



---

# Universidad de Valladolid

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE  
TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE LAS TECNOLOGÍAS DE LAS  
TELECOMUNICACIONES

Revisión de pruebas de software (Software Testing)  
y su aplicación a un Proyecto de Desarrollo de una  
Aplicación Web para la generación de Cuestionarios

---

AUTORA:

Patricia Briongos Pérez

TUTORES:

Dña. María Ángeles Pérez Juárez

Dña. Míriam Antón Rodríguez

Valladolid,

## **DESCRIPCIÓN DEL TFG**

**TÍTULO:** Revisión de pruebas de software (Software Testing) y su aplicación a un Proyecto de Desarrollo de una Aplicación Web para la generación de Cuestionario

**AUTOR:** Patricia Briongos Pérez

**TUTORAS:** M.<sup>a</sup> Ángeles Pérez Juárez y Míriam Antón Rodríguez

**DEPARTAMENTO:** Dpto. de Teoría de la Señal y Comunicaciones e Ingeniería Telemática

E.T.S.I. Telecomunicación, Universidad de Valladolid

## **MIEMBROS DEL TRIBUNAL**

**PRESIDENTE:** M.<sup>a</sup> Ángeles Pérez Juárez

**VOCAL:** Míriam Antón Rodríguez

**SECRETARIO:** David González Ortega

**SUPLENTE 1:** Javier Manuel Aguiar Pérez

**SUPLENTE 2:** Mario Martínez Zarzuela

**FECHA DE LECTURA:**

**CALIFICACIÓN:**

## Resumen

En todo el proyecto de desarrollo software es imprescindible realizar pruebas (*Software Testing*) para garantizar un nivel adecuado de calidad. En este Trabajo Fin de Grado se ha desarrollado una aplicación web que funciona como una plataforma de cuestionarios para ser utilizada por profesores y alumnos. La aplicación web permite a los usuarios con el rol de profesor registrar a nuevos alumnos, crear preguntas y, a partir de ellas, construir cuestionarios. Estos cuestionarios podrán ser asignados a los alumnos registrados en la plataforma. Los alumnos podrán acceder a un espacio personal donde verán los cuestionarios pendientes por responder que tengan, y podrán completarlos. También podrán ver las calificaciones o puntuaciones que hayan obtenido en los cuestionarios que hayan rellenado y enviado. El Trabajo Fin de Grado documenta todas las tareas realizadas en relación a la captura de requisitos y análisis y al desarrollo de la aplicación web. Como parte del trabajo, también se hará un estudio teórico del *Software Testing* para entender en qué consiste y por qué es necesario aplicarlo en el desarrollo de aplicaciones web. Para completar el trabajo se realizará un test de pruebas a la aplicación web desarrollada.

**Palabras clave:** Aplicación web, *Software Testing*, Profesores, Alumnos, Plataforma de Cuestionarios, Preguntas, PHPMyAdmin, PHP, SQL, JavaScript, CSS, HTML.

## Abstract

Every software development project needs to perform tests (Software Testing) to ensure an adequate level of quality. In this End-Of-Degree Project, a web application has been developed. It works as a questionnaire platform to be used by teachers and students. The web application allows users with the role of teachers to register new students, create questions and, from them, build questionnaires. These questionnaires may be assigned to students registered on the platform. The students will be able to access a personal space where they will see the questionnaires pending to be answered, and they will be able to complete them. Students can also see the qualifications or scores they have obtained in the questionnaires they have filed out and submitted. The End-Of-Degree Project documents all the tasks performed in relation to the capture of requirements and analysis and development of the web application. As part of the work as well, a theoretical study of Software Testing will be done to understand what it is and why it is necessary to apply it in the development of web applications. To complete the work, some tests will be carried out on the developed web application.

**Key words:** Web application, Software Testing, Teachers, Students, Questionnaire Platform, Questions, PHPMyAdmin, PHP, SQL, JavaScript, CSS, HTML.

## Agradecimientos

A mis tutoras de TFG: María Ángeles Pérez Juárez y Míriam Antón Rodríguez, por todas las dudas resueltas y toda la ayuda proporcionada.

A mis compañeros de carrera Mikel Barrio Conde, Rubén Blanco Pérez y Miguel Alonso Felipe, por ayudarme cuando estaba dando mis primeros pasos en el desarrollo web.

## Dedicatoria

A todos los que saben lo que hay tras estas páginas.

## Índice de Contenido

<b>Resumen</b> .....	<b>3</b>
<b>Abstract</b> .....	<b>4</b>
<b>Agradecimientos</b> .....	<b>5</b>
<b>Dedicatoria</b> .....	<b>6</b>
<b>1. INTRODUCCIÓN</b> .....	<b>14</b>
<b>2. EL SOFTWARE TESTING</b> .....	<b>16</b>
2.1 ¿Por qué es necesario el <i>software testing</i> ? .....	16
2.2 Principios y proceso del <i>software testing</i> .....	21
2.2.1 Planificación y control .....	22
2.2.2 Análisis y diseño.....	22
2.2.3 Implementación y ejecución .....	22
2.2.4 Criterio de parada e informe de resultados .....	23
2.2.5 Finalización del test .....	23
2.3 Tipos pruebas según su funcionalidad.....	24
2.3.1 Pruebas funcionales .....	24
2.3.2 Pruebas no funcionales .....	29
2.3.3 Pruebas estructurales.....	29
2.3.4 Pruebas relacionadas con los cambios.....	30
2.4 Otros tipos de pruebas .....	30
2.4.1 Pruebas automatizadas .....	30
2.4.2 Pruebas de rendimiento .....	31
2.5 Pruebas automatizadas.....	32
2.5.1 Niveles de las pruebas automatizadas .....	32
<b>3. LA APLICACIÓN WEB</b> .....	<b>34</b>
3.1 Requisitos funcionales y no funcionales de la aplicación web .....	35
3.2 Diagrama de Casos de Uso.....	37
3.2.1 Descripción de los Casos de Uso .....	39
3.3 Tablas de la base de datos .....	65
3.3.1 Admin .....	66
3.3.2 Profesores .....	67
3.3.3 Alumnos.....	68

3.3.4 Preguntas .....	69
3.3.5 Posibles respuestas .....	70
3.3.6 Cuestionarios.....	71
3.3.7 Pregunta_cuestionario .....	72
3.3.8 Cuestionario_alumno .....	72
3.3.9 Respuesta_alumno.....	73
<b>4. HERRAMIENTAS EMPLEADAS PARA EL DESARROLLO DE LA APLICACIÓN WEB.....</b>	<b>75</b>
4.1 La parte cliente .....	75
4.1.1 HTML .....	75
4.1.2 CSS.....	77
4.1.3 JavaScript.....	79
4.2 La parte del servidor .....	83
4.2.1 PHP .....	83
4.2.2 MySQL .....	85
4.2.3 SQL.....	85
4.3. Otras herramientas .....	87
4.3.1 Visual Studio Code.....	88
4.3.2 MAMP.....	88
4.3.3 PHPMyAdmin .....	89
<b>5. MANUAL DE USUARIO .....</b>	<b>91</b>
5.1 Página de inicio de sesión .....	91
5.2 Menú Principal de Profesores.....	91
5.2.1 Editar perfil propio .....	93
5.2.2 Cerrar Sesión .....	94
5.2.3 Barra de herramientas .....	95
5.2.4 Gestionar Preguntas.....	95
5.2.5 Gestionar Cuestionarios .....	101
5.2.6 Gestionar Alumnos.....	104
5.2.7 Asignar Cuestionarios.....	108
5.2.8 Ver Cuestionarios Sin Corregir .....	109
5.2.9 Ver Cuestionarios Corregidos.....	110
5.3 Menú Principal Alumnos.....	111



5.3.1 Realizar Cuestionario.....	112
554 Menú Principal de Administrador.....	112
6.4.1 Gestionar Profesores.....	113
<b>6. PRUEBAS DE SOFTWARE: SELENIUM .....</b>	<b>116</b>
6.1 Pruebas con Selenium IDE .....	117
6.1.1 Instalación de las herramientas necesarias y manejo de Selenium IDE .....	118
6.1.2 Casos de Prueba .....	120
6.1.3 Conclusiones de las pruebas realizadas .....	123
6.2 Pruebas con Selenium WebDriver .....	124
6.2.1 Instalación de las herramientas necesarias para Selenium WebDriver.....	124
6.2.2 Casos de Prueba .....	127
6.2.3 Conclusiones de las pruebas realizadas .....	133
6.3 Conclusión y comparación entre Selenium IDE y Selenium WebDriver.....	134
<b>7. ESTUDIO ECONÓMICO .....</b>	<b>136</b>
<b>8. CONCLUSIONES .....</b>	<b>137</b>
<b>9. LÍNEAS FUTURAS .....</b>	<b>139</b>
<b>Referencias .....</b>	<b>140</b>
<b>ANEXO I: Funciones útiles para el manejo de Selenium con Python .....</b>	<b>144</b>

## Índice de Ilustraciones

Ilustración 1: Tipo de errores y defectos. Etapas en las que pueden aparecer (Graham, van Veenendaal, Evans, & Black, 2010).....	18
Ilustración 2: Evolución del coste de detectar fallos en el software con el tiempo (Graham, van Veenendaal, Evans, & Black, 2010) .....	19
Ilustración 3: Estructura del Software Testing según la norma ISO/IEC/IEEE 29119 (Tuya, 2017).....	19
Ilustración 4: Competencias de los clientes y el proveedor durante el proceso de creación del Software (Instituto Tecnológico de Toluca, 2019) .....	20
Ilustración 5: Proceso básico de diseño de prueba (Toledo Rodríguez, 2014) .....	25
Ilustración 6: Ejemplo de Diagrama de Casos de Uso (Realizada con app.creately.com) .....	27
Ilustración 7: Ejemplo de grafos causa-efecto (Toledo Rodríguez, 2014).....	28
Ilustración 8: Niveles de pruebas automatizadas según la pirámide de Cohn (Grazas, 2019).....	32
Ilustración 9: Diagrama de Casos de Uso .....	38
Ilustración 10: Esquema de las tablas de la base de datos .....	66
Ilustración 11: Tabla admin .....	67
Ilustración 12: Tabla profesores .....	67
Ilustración 13: Tabla alumnos.....	68
Ilustración 14: Tabla preguntas .....	69
Ilustración 15: Tabla posibles respuestas.....	70
Ilustración 16: Tabla cuestionarios.....	71
Ilustración 17: Tabla pregunta_cuestionario .....	72
Ilustración 18: Tabla cuestionario_alumno .....	73
Ilustración 19: Tabla respuesta_alumno .....	74
Ilustración 20: Ejemplo de código CSS .....	79
Ilustración 21: Ejemplo de pantalla de Visual Studio Code.....	88
Ilustración 22: Interfaz de MAMP .....	89
Ilustración 23: Interfaz gráfica de phpMyAdmin.....	90
Ilustración 24: Pantalla de Inicio de Sesión .....	91
Ilustración 25: Registrar Alumno (Flaticon, 2019).....	92
Ilustración 26: Ver Alumnos (Flaticon, 2019) .....	92
Ilustración 27: Ver Preguntas (Flaticon, 2019) .....	92
Ilustración 28: Ver Cuestionarios (Flaticon, 2019) .....	93
Ilustración 29: Corregir Cuestionarios (Flaticon, 2019).....	93
Ilustración 30: Menú Principal del Profesor .....	93
Ilustración 31: Cabecera del profesor .....	94
Ilustración 32: Editar Perfil Usuario.....	94
Ilustración 33: Cerrar Sesión.....	95
Ilustración 34: Barra de herramientas.....	95

Ilustración 35: Menús desplegables de la barra de herramientas .....	95
Ilustración 36: Crear Pregunta.....	96
Ilustración 37: Pregunta con respuesta Verdadero/Falso.....	97
Ilustración 38: Pregunta con varias respuestas.....	98
Ilustración 39: Crear Pregunta gradual con 3 opciones .....	99
Ilustración 40: Ver las preguntas que hay registradas en el sistema .....	100
Ilustración 41: Ver Pregunta tipo 0,1,2 .....	100
Ilustración 42: Ver pregunta tipo 3 y 4.....	101
Ilustración 43: Editar Pregunta.....	101
Ilustración 44: Crear Cuestionario.....	102
Ilustración 45: Ver los cuestionarios que existen en el sistema.....	102
Ilustración 46: Ver Cuestionario .....	103
Ilustración 47: Editar Cuestionario .....	104
Ilustración 48: Registrar usuario.....	105
Ilustración 49: Ver Alumnos .....	106
Ilustración 50: Ver Cuestionarios Pendientes de un alumno .....	106
Ilustración 51: Editar Alumno .....	107
Ilustración 52: Ver Alumno .....	107
Ilustración 53: Lista de cuestionarios que el profesor puede asignar a los alumnos ..	108
Ilustración 54: Lista de alumnos a los que el profesor puede asignar los cuestionarios que haya elegido.....	109
Ilustración 55: Ver Cuestionarios sin Corregir.....	109
Ilustración 56: Corregir cuestionario de un alumno.....	110
Ilustración 57: Ver Cuestionarios Corregidos .....	111
Ilustración 58: Página Principal del Alumno .....	111
Ilustración 59: Realizar Cuestionario .....	112
Ilustración 60: Menú Principal del Administrador .....	113
Ilustración 61: Registrar Profesor .....	114
Ilustración 62: Listar Profesores .....	114
Ilustración 63: flujo de trabajo de Selenium IDE (Selenium, 2019).....	117
Ilustración 64: Instalación de Selenium IDE en el navegador Google Chrome .....	118
Ilustración 65: Icono Selenium IDE en la barra de herramientas del navegador.....	118
Ilustración 66: Pantalla de trabajo con Selenium IDE .....	119
Ilustración 67: Abrir un proyecto existente con Selenium IDE.....	119
Ilustración 68: Caso inicio_sesion.....	120
Ilustración 69: Caso crear_alumno .....	121
Ilustración 70: Comandos grabados al ejecutar el Caso crear_alumno .....	122
Ilustración 71: prueba editar_alumno fallida .....	123
Ilustración 72: Acceder al intérprete de Python .....	125
Ilustración 73: instalación de la biblioteca Selenium .....	126

Ilustración 74: Esquema de paso de datos desde Selenium hasta Firefox (Guru99, 2019)	127
Ilustración 75: script cuestionario_completo.py	131
Ilustración 76: Resultado correcto del script de prueba cuestionario_completo	132
Ilustración 77: Resultado incorrecto debido a un error en la escritura del nombre por el que se busca al elemento en el script iniciar_sesion.py	133
Ilustración 78: Resultado incorrecto debido a la falta del atributo name en el botón de un formulario	133

## Índice de Tablas

Tabla 1: Requisitos funcionales .....	36
Tabla 2: Requisitos no funcionales .....	37
Tabla 3: Caso de Uso: Gestionar Preguntas .....	40
Tabla 4: Caso de Uso: Crear Pregunta .....	41
Tabla 5: Caso de Uso: Ver Pregunta .....	42
Tabla 6: Caso de Uso: Editar pregunta .....	43
Tabla 7: Caso de Uso: Eliminar pregunta.....	44
Tabla 8: Caso de Uso: Gestionar Alumnos.....	45
Tabla 9: Caso de Uso: Crear Alumno .....	46
Tabla 10: Caso de Uso: Ver Alumno .....	46
Tabla 11: Caso de Uso: Editar Alumno .....	47
Tabla 12: Caso de Uso: Eliminar Alumno.....	48
Tabla 13: Caso de Uso: Gestionar Cuestionarios.....	49
Tabla 14: Caso de Uso: Crear Cuestionario .....	50
Tabla 15: Caso de Uso: Ver Cuestionario .....	51
Tabla 16: Caso de Uso: Editar Cuestionario .....	52
Tabla 17: Caso de Uso: Eliminar Cuestionario.....	53
Tabla 18: Caso de Uso Asignar Cuestionarios .....	54
Tabla 19: Caso de Uso: Ver Cuestionarios Respondidos .....	55
Tabla 20: Caso de Uso: Corregir Cuestionario.....	56
Tabla 21: Caso de Uso: Eliminar Cuestionarios Pendientes .....	58
Tabla 22. Caso de Uso: Ver Cuestionarios Pendientes de Responder .....	58
Tabla 23: Caso de Uso: Rellenar Cuestionario.....	59
Tabla 24: Caso de Uso: VerCalificacionCuestionariosRespondidos .....	60
Tabla 25: Caso de Uso: GestionarProfesores .....	61
Tabla 26: Caso de Uso: CrearProfesor .....	62
Tabla 27: Caso de Uso: Ver Profesor .....	63
Tabla 28: Caso de Uso: Editar Profesor .....	64
Tabla 29: Caso de Uso: Eliminar Profesor.....	65
Tabla 30: Ventajas y desventajas de Selenium IDE y Selenium WebDriver .....	134

# 1. INTRODUCCIÓN

En los últimos años, las Aplicaciones Web están presentes en nuestras vidas de forma habitual. Algo que en el siglo pasado era inimaginable se ha convertido para nosotros en algo tan cotidiano como comer o salir a la calle. Uno de los entornos en los que se emplean las aplicaciones web son los centros educativos, donde los métodos de enseñanza han ido cambiando con el paso de los años y se han ido adaptando para incorporar las nuevas tecnologías.

No es inusual que los alumnos (tanto en colegios, como institutos o universidades) tengan que realizar diversas acciones como entregar sus deberes a través de un campus virtual o plataforma similar. Este es el objetivo de este Trabajo Fin de Grado: diseñar e implementar una plataforma que permita a profesores y alumnos estar en contacto. Los profesores podrán asignar cuestionarios que ellos mismos hayan diseñado a sus alumnos; mientras que estos podrán responder a los cuestionarios que se les haya asignado.

Pero ¿cómo se sabe que el *software* que compone la plataforma de cuestionarios con la que se va a trabajar es fiable? ¿Es posible construir *software* sin fallos? Esa es la pregunta que mueve miles de trabajos relacionados con el *software testing* y muchas investigaciones realizadas por empresas que se dedican en gran parte a esta tarea. También es uno de los objetivos de este Trabajo de Fin de Grado: concienciar sobre la importancia del *software testing*, describir las múltiples formas de aplicarlo y hacer una demostración de su uso. Pero antes de comenzar, hay que destacar algo sobre lo que han hecho hincapié todas las fuentes bibliográficas que se han consultado para la realización de este TFG: el *software* perfecto no existe.

El presente TFG está compuesto por siete capítulos. En este primer capítulo se pretende situar al lector en el contexto en el que se va a desarrollar todo el trabajo, proporcionándole una visión global.

En el segundo capítulo, se describe el marco teórico del *software testing*: qué es y por qué es necesario, qué tipo de pruebas se pueden llevar a cabo, etc.

En el tercer capítulo se describe lo relativo a las pruebas de *software* que se van a realizar a la Aplicación Web que se ha diseñado: los niveles de pruebas automatizadas y el tipo de pruebas automatizadas en las que se ha centrado este Trabajo Fin de Grado.

En el cuarto capítulo se describe de manera detallada el objetivo de la Aplicación Web y los pilares sobre los que se va a construir la aplicación web: sus requisitos funcionales y no funcionales, la organización de su base de datos y la descripción de los casos de uso que la forman.

En el quinto capítulo se describen las herramientas de las que se ha hecho uso para desarrollar la página web (programas, lenguajes de programación, herramientas externas, etc.).

En el sexto capítulo se muestra una guía de usuario, con el fin de que la persona que vaya a manejar la Aplicación Web sepa y comprenda su funcionamiento. El objetivo de este capítulo es familiarizar al usuario con la plataforma y que sirva asimismo como un manual de usuario de la misma.

En el séptimo capítulo se describe cómo se han llevado a cabo una serie de pruebas de software sobre la aplicación que se ha desarrollado. A continuación, se extraen las conclusiones sobre las pruebas realizadas a la Aplicación Web, poniéndose de manifiesto cómo en ocasiones, se hace uso de técnicas de testeo de software sin ser consciente de ello.

En el octavo capítulo se ha elaborado un presupuesto, en el que se indica el coste que hubiera tenido este proyecto en el mercado de las Aplicaciones Web.

Los capítulos noveno y décimo presentan respectivamente las conclusiones obtenidas con la realización de este Trabajo de Fin de Grado, así como las líneas futuras en las que se podría trabajar en este proyecto.

## 2. EL SOFTWARE TESTING

¿Qué es el *software testing*? El *software testing* (o testeo del *software*) es el proceso mediante el cual se determina la corrección, completitud, seguridad y calidad del *software*. Se denomina *software* al conjunto de datos, documentación, programas, procedimientos y reglas que componen un sistema informático. Habitualmente el *software testing* se descompone en diferentes etapas, cada una de ellas focalizada en un tipo de fallo: cada etapa está especializada en un tipo de testeo distinto. Más adelante se van a describir cada una de esas etapas, los objetivos particulares que persiguen y los mecanismos más habituales en cada una de ellas (Bolaños Alonso, Sierra Alonso, & Alarcón Rodríguez, 2008)

En este capítulo del TFG se van a presentar los conceptos básicos y el objetivo que persigue *software testing* con la finalidad de que el lector se familiarice y conozca este concepto.

### 2.1 ¿Por qué es necesario el *software testing*?

Hacer pruebas sobre algo significa comprobar que todo funciona bien. El *software testing* es necesario porque es humano cometer errores, y el campo del *software* no es ajeno a ello. Los errores pueden tener distintos niveles de gravedad, pero es importante no subestimar ninguno de ellos. Algunos fallos pueden tener como origen un mal planteamiento o un punto ciego en el programa, lo que les harán difíciles de detectar. A menudo, el *software testing* se realiza por personas ajenas a la programación del *software*, para aportar una visión crítica y externa al estudio.

Vivimos en un mundo donde nos rodean objetos cotidianos que están programados. Y aunque no es habitual, tampoco es extraño tener problemas con alguno de ellos de vez en cuando: se ha producido un error en el *software*. Seguramente se hayan realizado pruebas de *software* sobre ese aparato, pero algún caso o alguna combinación no ha sido considerada. El *software* perfecto no existe. De ser así, Windows, Android o cualquier otro sistema operativo que empleáremos no mandaría actualizaciones a nuestros terminales cada cierto tiempo o no tendríamos ningún problema con nuestros aparatos electrónicos. Entonces, ¿para qué realizar pruebas sobre el *software*, si es imposible que desaparezcan todos los fallos? Para minimizar la cantidad de fallos cuanto sea posible.

Un **riesgo** es algo que no ha ocurrido todavía y no tiene por qué ocurrir en un futuro, pero existe la posibilidad de que ocurra: es un problema en potencia. Los riesgos



nos rodean día a día en todos los ámbitos: hay riesgos cuando cruzamos una carretera, cuando sacamos algo caliente del microondas o cuando cogemos el coche, pero eso no implica que vaya a ocurrir nada malo. No obstante, se pueden realizar una serie de acciones que disminuyan significativamente la probabilidad de que las cosas vayan mal.

Un mismo fallo puede no tener las mismas consecuencias en distintas situaciones: un error tipográfico en la cuenta de usuario de un alumno de clase no es lo mismo a que se produzca en el billete de avión que se ha impreso para ir de vacaciones. Aunque el fallo sea el mismo, uno de ellos puede costarte el hecho de quedarte sin vacaciones. Por eso, es importante ser tan estricto en la detección de fallos como la situación lo requiera.

Un **error** da lugar a un defecto. El uso incorrecto de un *software* puede llevar a un comportamiento inesperado. Un **defecto** en la construcción del *software* también provocará un fallo. Puede ocurrir que un error o un fallo den lugar a un defecto, pero este no sea visible. Esto nos puede llevar a dos situaciones: que sea algo insignificante o, por el contrario, que aparezca un comportamiento extraño del que no se sepa su origen ni cómo ponerlo fin (Graham, van Veenendaal, Evans, & Black, 2010).

Existen otra serie de fallos que van más allá de las competencias del programador o del equipo de testeo del *software*, pero que hay que tener en cuenta si se está diseñando un sistema en conjunto: radiaciones electromagnéticas que puedan interferir en las comunicaciones que se están llevando a cabo, el estado de los componentes físicos, etc.

¿En qué punto del diseño del *software* aparecen los fallos? Se pueden producir en cualquiera de las fases desde que se comienza a plantear un proyecto hasta que este se termina. En la Ilustración 1 se pueden ver esquemas de qué tipos de fallos aparecen en distintas etapas del proceso de creación y diseño del *software*.

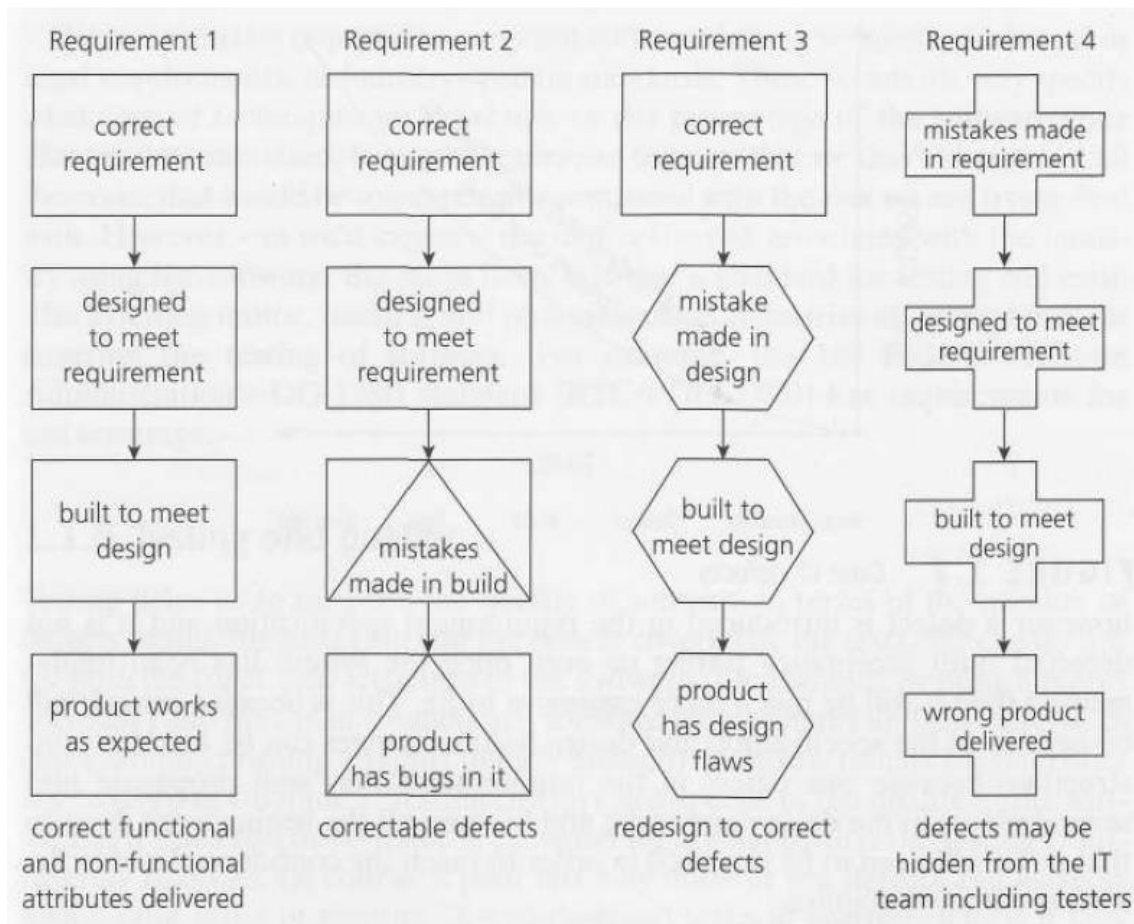


Ilustración 1: Tipo de errores y defectos. Etapas en las que pueden aparecer (Graham, van Veenendaal, Evans, & Black, 2010)

Todos los fallos tienen un coste, bien sea económico o de tiempo (que se puede traducir en económico también), pero no todos los fallos cuestan lo mismo. Cuánto más avanzado esté el desarrollo del *software*, más costoso será el resolver un fallo. En la Ilustración 2 se puede ver el crecimiento del coste según se avanza en el tiempo (Graham, van Veenendaal, Evans, & Black, 2010).

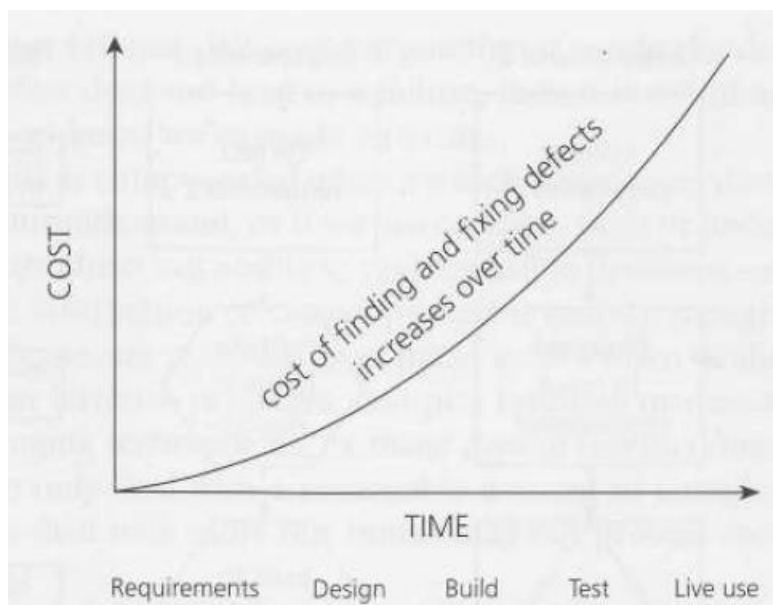


Ilustración 2: Evolución del coste de detectar fallos en el software con el tiempo (Graham, van Veenendaal, Evans, & Black, 2010)

El *software testing* está definido en el marco legal y existen estándares que lo regulan. Estos estándares marcan los porcentajes del código total que es necesario probar para que se considere apto para la comercialización y otros parámetros. La **ISO/ICE/IEEE 29119 Software Testing** tiene como propósito unificar e integrar la normativa actual que está fragmentada en distintos estándares usados por diferentes asociaciones: BSI, IEEE y ISO/IEC JTC1/SC7. En la Ilustración 3 se puede ver un esquema de los diferentes estándares que existen, qué regula cada uno de ellos y cómo están relacionados entre sí (Tuya, 2017).

## ISO/IEC/IEEE 29119 Software Testing - Estructura

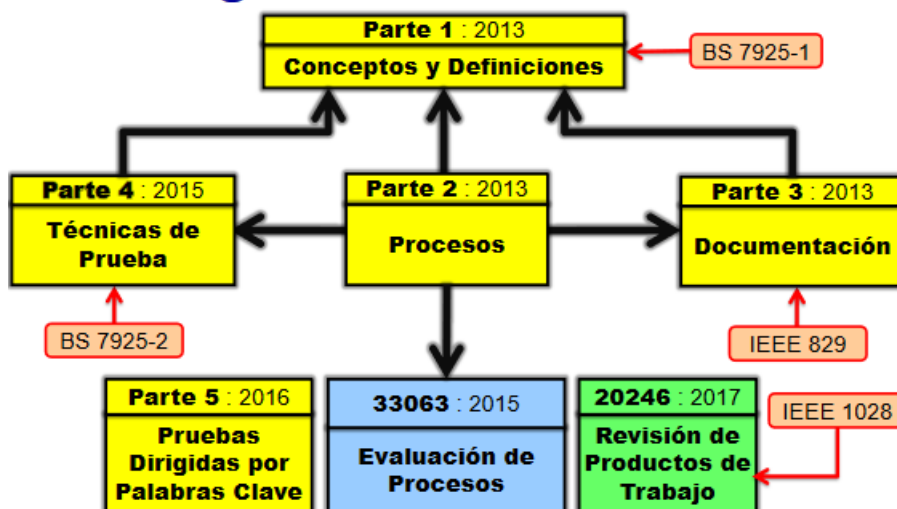


Ilustración 3: Estructura del Software Testing según la norma ISO/IEC/IEEE 29119 (Tuya, 2017)

Las pruebas que se realizan al *software* proporcionan una medida de la **calidad** del *software* en función de los defectos encontrados, el número de tests que ha pasado, etc. Para que un test sea considerado de buena calidad, y que este proporcione unas garantías, debe cumplir múltiples requisitos. Por otra parte, que un programa pase muchos tests no implica que ese programa no tenga fallos.

Pero ¿qué es la **calidad** en términos de *software*? Cuando se diseña un *software* se quiere que cumpla unas determinadas especificaciones (validación) y que esas especificaciones funcionen adecuadamente (verificación). Las empresas que se dedican al campo del *software testing* deben asegurarse de llegar a un acuerdo claro con su cliente cuando hablan de la calidad del programa que se va a diseñar. El factor económico tiene una gran importancia, porque un cliente paga por una serie de condiciones, y esas expectativas deben ser cubiertas. Existen numerosas empresas a nivel mundial que se dedican al *software testing* y que trabajan para bancos, aseguradoras o compañías telefónicas bien conocidas (Graham, van Veenendaal, Evans, & Black, 2010).

En la Ilustración 4 se pueden ver las competencias del cliente, que demanda un *software* que cumple una serie de requisitos; y el proveedor, que se encargará de satisfacer todas las peticiones del cliente, y que está compuesto por múltiples, equipos, cada uno de ellos con una tarea.

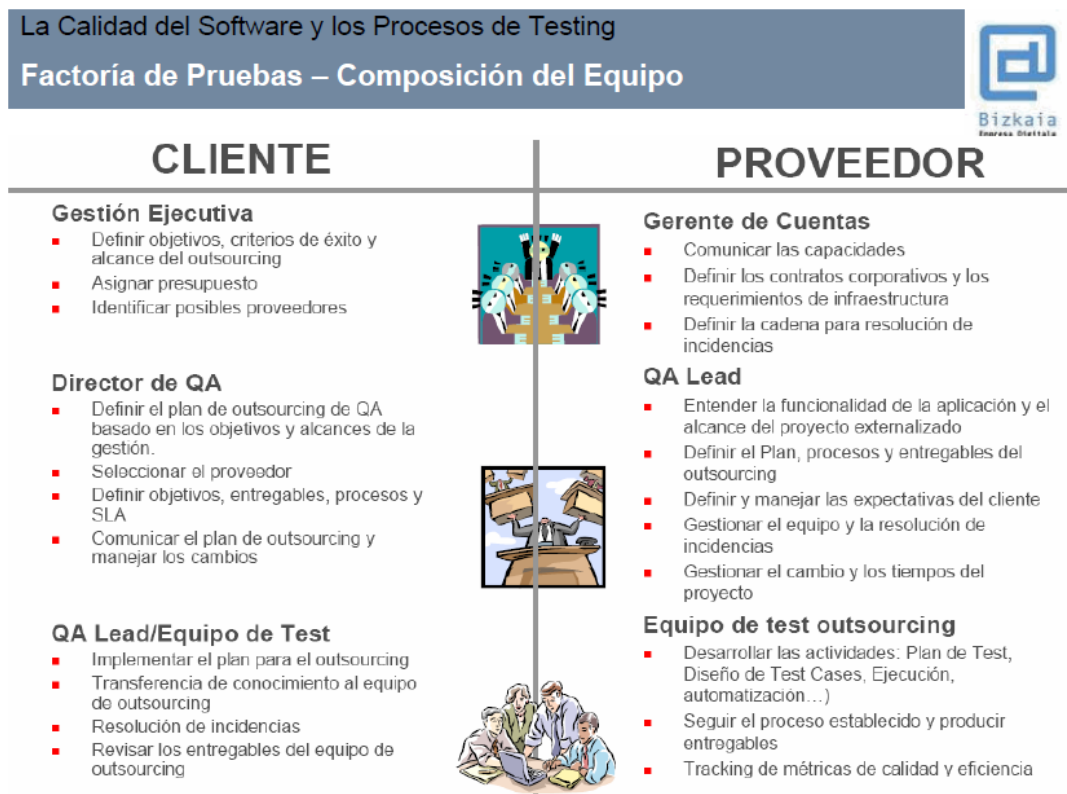


Ilustración 4: Competencias de los clientes y el proveedor durante el proceso de creación del Software (Instituto Tecnológico de Toluca, 2019)

Al inicio de un proyecto, es útil tener cerca a personas que hayan trabajado en algo similar con anterioridad: los fallos tienden a repetirse, y evitar los fallos es también una parte esencial del *software testing*.

¿Cómo se sabe en qué momento hay que dejar de hacer pruebas sobre un *software*?  
¿Cuándo se sabe que se ha testeado todo? No es posible. No se puede crear un *software* 100% libre de fallos. El hecho de no encontrar fallos no implica que no los haya. El *software testing* busca aplicar la máxima cantidad de pruebas para que el riesgo de que la aplicación falle sea mínimo.

## 2.2 Principios y proceso del *software testing*

Todo test que se realiza, sea de *software*, un examen sobre una materia, etc. cumple una serie de requisitos:

- Ha sido planeado con antelación.
- Se busca cumplir unos objetivos mínimos: o se pasan o no se pasan.

El *software testing* se asienta sobre una serie de pilares:

1. **Mostrar la presencia de defectos:** el *software testing* puede demostrar que hay fallos, pero no puede asegurar que no haya defectos.
2. **Un testeo exhaustivo es imposible:** realizar todas las combinaciones posibles de casos de estudio no es factible.
3. **Testeo anticipado:** El *software testing* debe comenzar tan pronto como se inicie un proyecto.
4. **La paradoja del pesticida:** si un test se repite muchas veces sobre un mismo código, llegará un momento en el que no se encuentren nuevos fallos. Hay que variar los tests para evitar que el programa se haga inmune a ellos.
5. **El testeo depende del contexto:** no es lo mismo testear una aplicación para uso personal que una aplicación que va a usarse en una multinacional.
6. **La falacia de la ausencia de errores:** la ausencia de fallos no sirve de nada si la aplicación diseñada no cumple con las funciones para la que fue pensada.

Cuando se inicia un proyecto, es importante tener en cuenta el tiempo que se va a dedicar a cada una de las tareas, aunque sea de manera aproximada. El proceso de testeo se puede descomponer en distintas fases, y no hay un modelo establecido como principal. Aquí se va a describir uno en concreto. Los pasos por los que está formado son: **planificación y control, análisis y diseño, implementación y ejecución, criterio de parada e informe de resultados, finalización del testeo** (Bolaños Alonso, Sierra Alonso, & Alarcón Rodríguez, 2008).

### 2.2.1 Planificación y control

Lo primero de todo cuando se quiere diseñar un plan de *software testing* es asegurarse de que se ha comprendido adecuadamente el fin que busca el cliente. Teniendo el fin claro, se podrán fijar las políticas de testeo y la estrategia a seguir. Las competencias de esta etapa son:

- Determinar los riesgos e identificar los objetivos de las pruebas que se van a realizar.
- Determinar las técnicas que se van a emplear y la cobertura que van a ofrecer.
- Implementar las políticas y estrategias elegidas. Justificar por qué se han elegido y se considera que son las mejores.
- Diseñar un esquema a seguir para el análisis que se va a realizar. El orden en el que se van a aplicar las diferentes pruebas y a qué va orientada cada una de ellas.
- Fijar el criterio de finalización: saber qué se espera al final de cada prueba porque se considera que el *software* cumple con lo que se espera de él (Graham, van Veenendaal, Evans, & Black, 2010).

### 2.2.2 Análisis y diseño

En esta fase, los objetivos del *software testing* se transforman en condiciones tangibles: se construyen los *scripts* que van a comprobar la calidad del software. Las competencias de esta etapa son:

- Revisar las bases que se han fijado anteriormente y examinar las especificaciones que se han pedido para el *software*. El objetivo de este paso es encontrar posibles ambigüedades en todo lo que se haya hecho anteriormente.
- Identificar las condiciones de las pruebas que se van a realizar, ¿cómo se va a enfocar el problema?
- Diseñar un test que se centre en alguno de los factores de riesgo del *software* (Graham, van Veenendaal, Evans, & Black, 2010).

### 2.2.3 Implementación y ejecución

Se convierten las condiciones de test en **casos de prueba**. También se fija el entorno en el que se van a ejecutar los casos de pruebas. En este paso se suelen detectar muchos de los fallos del *software*. Las competencias de esta etapa son:

- Priorizar ciertos casos de prueba sobre otros. Escribir las instrucciones de ejecución de los tests: puede haber más de uno. En alguna situación puede que

se requiera ejecutar distintos casos de prueba de manera autónoma → test automatizados.

- Implementar y verificar el entorno.
- Ejecutar los test sobre los casos de prueba.
- Registrar las salidas obtenidas y nombrarlas de forma correcta indicando la versión de prueba que es, es decir, construir un log de pruebas.
- Comparar los resultados obtenidos con los esperados.
- Hacer un informe con los incidentes detectados. Los incidentes ayudan a obtener información sobre los defectos que se han producido. Con ellos se podrá saber si el problema ha sido a causa del *software*, los datos de prueba, la forma de ejecución, etc. También se busca mejorar la calidad del propio test de prueba con cada ejecución del mismo.
- Repetir los pasos anteriores tantas veces como sea necesario hasta obtener un resultado satisfactorio (Graham, van Veenendaal, Evans, & Black, 2010).

#### 2.2.4 Criterio de parada e informe de resultados

Esta etapa no es una etapa final que se realice después de realizar todos los test necesarios. Esta etapa se realiza para cada uno de los test, para saber si ya se ha ejecutado el test en cuestión las veces suficientes. Es importante definir el criterio de parada adecuadamente, ya que va a ser el que indique cuándo parar. También hay que tener en cuenta el criterio de aceptación, que indica si el *software* ha superado el test o no. Las competencias de esta etapa son:

- Comprobar los ficheros log del test y el valor de parada del test, para asegurarse de que no ha habido un error y el test ha parado de ejecutarse sin haber obtenido los resultados necesarios (aunque eso no implique que sean los correctos y haga falta volver a ejecutarlo).
- Verificar si son necesarias más repeticiones del test o por el contrario se han obtenido los resultados esperados y se puede pasar a la siguiente etapa.
- Escribir un informe sobre los resultados obtenidos donde se explique todo lo que ha pasado y las conclusiones a las que se ha llegado: una memoria sobre todo el proceso (Graham, van Veenendaal, Evans, & Black, 2010).

#### 2.2.5 Finalización del test

Para dar por finalizado todo el proceso del *software testing* hay que comprobar que todos los resultados obtenidos de los múltiples tests y de sus repeticiones cumplen todos los criterios marcados. Además, el *software* acabado debe funcionar

correctamente, es decir, debe estar listo para su distribución. El test se dará por terminado cuando la parte del cliente dé por bueno los resultados que se le han proporcionado (Graham, van Veenendaal, Evans, & Black, 2010).

### 2.3 Tipos pruebas según su funcionalidad

Cuando se quieren buscar fallos en un *software*, la manera habitual de proceder es comenzar a realizar pruebas desde dentro hacia fuera: analizar cada una de las unidades más pequeñas de forma individual y acabar analizando el sistema en global. El orden más lógico para seguir sería:

1. Pruebas unitarias: se comprueba la funcionalidad de cada clase o módulo por separado.
2. Pruebas de integración: se agrupan módulos o clases y se analiza el flujo que circula entre ellas.
3. Pruebas de validación: se comprueba que lo analizado hasta el momento cumple los requisitos impuestos por el usuario.
4. Pruebas de sistema: se despliega el *software* diseñado en el entorno donde va a trabajar.
5. Prueba de aceptación: el usuario da el visto bueno al producto que ha encargado y verifica si cumple los requisitos que debería.

Se definen tipos de niveles por organización: cada uno de ellos se encargará de sus objetivos en un cierto momento. Se pueden realizar pruebas de distintos tipos: **funcionales, no funcionales, estructurales o relacionadas con los cambios**. Algunas de ellas tiene lugar en el nivel de diseño del *software*, mientras que otras se tienen en cuenta en todos los niveles, como es el caso de las **pruebas estructurales**.

Un tipo de test está focalizado en un objetivo particular: una función, un componente... pero de forma particular y específica. A continuación, se van a explicar algunos de los tipos de pruebas que existen (Graham, van Veenendaal, Evans, & Black, 2010).

#### 2.3.1 Pruebas funcionales

¿Qué es “la **función**” de un sistema? La respuesta es simple: lo que hace. Habitualmente una serie de especificaciones describen las funcionalidades de un sistema. Hay ciertas



funciones, que, aunque no se hayan especificado, se sobreentiende que el sistema debe ser capaz de realizar. Pero es difícil comprobar que todo vaya bien si no se tienen las funcionalidades acotadas. Por eso, es importante capturar y definir una lista de **requisitos funcionales** cuando se trabaja en un proyecto *software*. Habitualmente las pruebas funcionales también son llamadas pruebas de caja negra, ya que lo que interesa es ver el comportamiento del sistema como conjunto, no lo que pasa en su interior (Graham, van Veenendaal, Evans, & Black, 2010).

### Tipos de prueba de caja negra

- Partición de equivalencia: se basa en clasificar los casos de entrada de un programa en clases y comprobar el funcionamiento interno del bloque en función de cada uno de los conjuntos. La entrada puede ser un conjunto de rangos, valores específicos o valores lógicos.
- Análisis de valores límite: a partir de los resultados del caso anterior, se busca el valor límite que hace que el programa funcione de manera adecuada (Bolaños Alonso, Sierra Alonso, & Alarcón Rodríguez, 2008).

Para comenzar a realizar una prueba funcional es necesario presentar el catálogo de requisitos funcionales del *software* que se va a diseñar.

A continuación, se van a plantear algunas de las estrategias para testear el *software*.

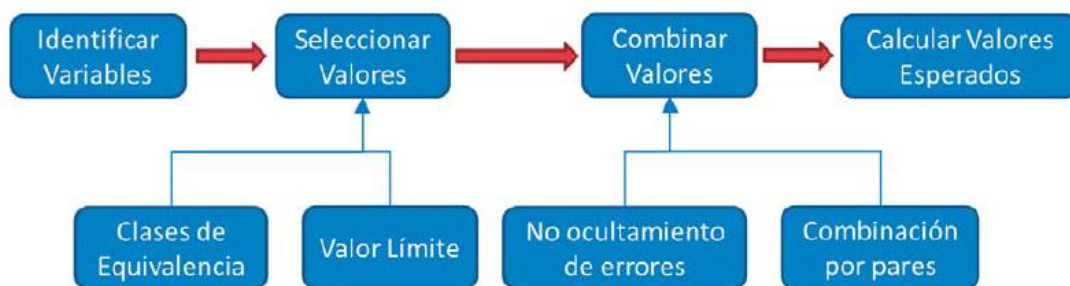


Ilustración 5: Proceso básico de diseño de prueba (Toledo Rodríguez, 2014)

La Ilustración 5 muestra un posible esquema de trabajo para diseñar datos de prueba: se prepara un conjunto de valores de entrada para los que se quiere examinar el comportamiento del sistema, y poder localizar así los posibles fallos que se puedan dar.

Lo primero es **identificar las variables** que participan en el caso de diseño que se va a testear. Una vez se hayan identificado, se definirá un rango de valores típicos para dichas variables. A menudo, un conjunto de valores diferentes obtiene una salida similar, por lo que: se pueden englobar en un mismo caso de estudio. Esto es útil porque evita que haya casos de prueba muy grandes. Por último, se comprueba que los valores obtenidos son adecuados respecto a los valores esperados.

1. **Identificación de variables.** Cuando se estudia un caso de diseño hay que tener muy claro cuáles son las variables que afectan a la funcionalidad en concreto que se quiere probar. Puede haber variables del programa que no afecten de ninguna forma a la funcionalidad que se está analizando, por lo que no tendría sentido incluirlas en el estudio. Por ejemplo, si se están buscando fallos en una página web donde hay que introducir los datos de un usuario que se quiere registrar, sería lógico comprobar el valor de las variables: nombre, apellido, dirección, fecha de nacimiento, DNI...

2. **Selección de valores** Una vez que se saben las variables que se quieren estudiar, se tendrá que saber el valor típico y adecuado de las mismas. Siguiendo con el ejemplo anterior, cada una de las variables deberá tener un valor concreto. El nombre deberá ser una cadena de caracteres, la fecha de nacimiento deberá estar compuesta por día, mes y año, o el DNI deberá ser una cadena de ocho números seguida de una letra. Si alguna de las variables contiene un valor diferente al esperado, se tendrá que avisar de ello al usuario, para evitar así posibles errores futuros.

3. **Combinación de valores.** Como ya se ha comentado antes, hay muchos casos que son similares entre sí. No es necesario analizar todas las combinaciones posibles (en parte porque en ocasiones puede ser imposible), a veces se pueden optimizar las tareas de comprobación.

4. **Calcular valores esperados.** En esta fase hay que diseñar el oráculo de prueba: hay que comparar lo obtenido tras introducir la entrada con lo que se esperaba obtener. En el caso del registro de usuario, se puede hacer la comprobación de que el usuario en cuestión sea mayor de dieciocho años para que sea posible finalizar el registro. Si esto no se cumple, la operación se cancelará porque no se ha obtenido el valor esperado en el campo de fecha de nacimiento (Toledo Rodríguez, 2014).

### ¿Cómo identificar los casos de prueba que se necesitan para testear el *software*?

Se ha descrito cómo se tienen que tratar los datos de entrada con los que se va a testear el *software*, pero ¿qué casos de prueba hay que definir? ¿Cómo hay que definirlos? La forma de afrontar el *software testing* depende del carácter del caso de prueba.

- Representación de Caso de Uso (funcionalidad del sistema)

Un caso de uso es un elemento de análisis que describe la interacción usuario-sistema. El artefacto UML denominado Diagrama de Caso de Uso representa los Casos de Uso: muestra a los actores involucrados en cada caso de uso, e incluso las interdependencias entre los diferentes actores o casos de uso. Formalizar los casos de uso es algo muy importante para dejar claros los estados por los que se pueden pasar y establecer las relaciones entre los actores y el sistema.

Se denomina **actor** a **quién** puede desencadenar un Caso de Uso.

En el ejemplo de la Ilustración 6, el sistema es una biblioteca. Los casos de uso que se pueden analizar en busca de fallos que depurar son **registrar préstamo, registrar devolución y realizar reserva por internet**. Los actores que interactúan con el sistema son el bibliotecario y el usuario de la biblioteca (Toledo Rodríguez, 2014).

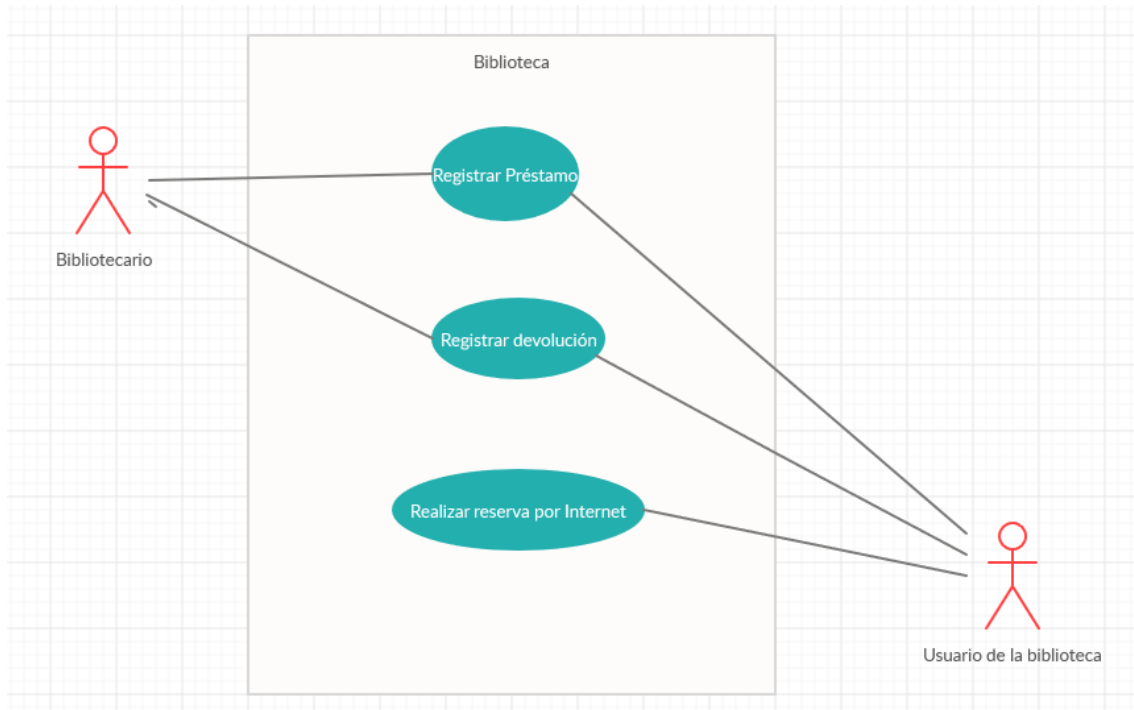


Ilustración 6: Ejemplo de Diagrama de Casos de Uso (Realizada con app.creately.com)

- Técnica de tablas de decisión.

Esta estrategia de *software testing* es útil cuando el programa que se quiere probar está basado en decisiones lógicas: programas donde dominen estructuras de bucles tipo *if-then-else*. ¿Por qué? Cuando la cantidad de estructuras de este tipo es abundante, la probabilidad de que se produzca un problema en el programa debido a una mala definición de una condición es alta. Se pueden modelar una serie de entradas que comprueben si se cumplen todas las condiciones que se han puesto. Las **tablas de decisión** permiten realizar estas pruebas, aunque existen otros métodos como puede ser el **grafo de causa-efecto**.

En las **tablas de decisión** se combinan datos que conformarán los casos de prueba y se agruparán en una matriz.

El **grafo de causa-efecto** se basa en realizar diagramas que representen la relación lógica entre distintas causas y posibles efectos. Se basa en sentencias del tipo “Si A, entonces B”, “Si A o C, entonces B”, etc. En la Ilustración 7 se puede ver algún ejemplo de estos grafos. Estos casos son muy sencillos, pero pueden llegar a representar situaciones realmente complejas (Toledo Rodríguez, 2014).

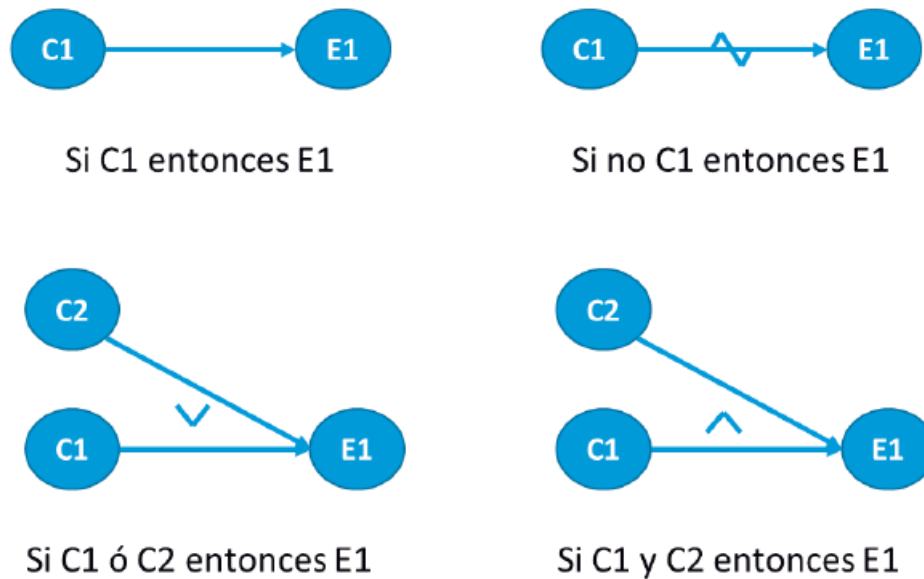


Ilustración 7: Ejemplo de grafos causa-efecto (Toledo Rodríguez, 2014)

- Técnica de máquina de estados

Una máquina de estados es una forma de modelar el comportamiento de un sistema con entradas y salidas. Las salidas del sistema dependen tanto del valor de la entrada actual como de los anteriores valores de las entradas: depende del estado en el que se encuentre. Con este tipo de pruebas se puede comprobar que el programa pasa de forma correcta de un estado a otro (de un caso de uso a otro), cuando la entrada es la esperada (Toledo Rodríguez, 2014).

- Matriz CRUD

Un patrón CRUD (*create, read, update, delete*) permite modelar a una entidad de un sistema de información. Esta herramienta es útil para modelar casos de uso en los que sea necesario poder crear, leer, modificar y borrar ciertas entidades. En el ejemplo de la biblioteca que se ha descrito anteriormente, se podría seguir un patrón CRUD para el caso de uso "Registrar usuario", donde sería necesario poder crear al usuario, poder consultar información sobre el usuario, poder modificar alguno de los datos del usuario y poder borrar al usuario si este se diera de baja en la plataforma de la biblioteca. Este tipo de recursos permite organizar al programador el *software* que va a diseñar (Toledo Rodríguez, 2014).

Este primer grupo de técnicas son un conjunto de herramientas que permiten al programador evitar fallos aparentemente simples, pero un fallo en la etapa de diseño de *software* puede acarrear fallos más complejos en futuras etapas. Partir de una buena organización y una base sólida cuando se trata de construir un *software* es muy importante y puede decrementar la probabilidad de encontrar fallos graves más adelante (Toledo Rodríguez, 2014).

### 2.3.2 Pruebas no funcionales

Las pruebas no funcionales se encargan de comprobar la calidad de las características que debe cumplir el sistema: cómo de bien reacciona el sistema ante las entradas o cómo de rápido es. Algunas de las propiedades que comprueban los test no funcionales son:

- **Fiabilidad:** comprueba cómo de tolerante es a los fallos, cuánto tarda en recuperarse ante un fallo o cómo de robusto es.
- **Usabilidad:** determina lo fácil de usar que es y si es amigable.
- **Eficiencia:** cuántos recursos emplea.
- **Mantenibilidad:** cómo de fácil será introducir cambios futuros y si es fácil buscar fallos o no.
- **Portabilidad:** si permite adaptarse a otros sistemas operativos, coexistir con otras aplicaciones o cómo de fácil es de instalar (Graham, van Veenendaal, Evans, & Black, 2010).

### 2.3.3 Pruebas estructurales

Si las pruebas funcionales y no funcionales se pueden llamar de “caja negra”, al test estructural se lo puede llamar de “caja blanca”. Ahora sí interesa conocer el interior del sistema cuando se comprueban las funcionalidades. Se quiere comprobar la estructura del sistema y de los componentes. Estas pruebas habitualmente se basan en modelos de control de flujo (Graham, van Veenendaal, Evans, & Black, 2010).

#### **Tipos de prueba de caja blanca**

- **Pruebas de interfaces entre módulos y clases:** se analiza el flujo que atraviesa interfaces, funciones o métodos.
- **Pruebas de estructuras de datos:** se comprueba que, durante toda la ejecución del programa, no se produzca ningún desbordamiento sobre ninguna variable o que ninguna variable contenga un valor no válido que pueda bloquear al programa.
- **Prueba de camino básico:** se emplea en código con bucles o con estructura secuencial y sirve para comprobar que los pasos son ejecutados, y que todas las posibles opciones funcionan de manera correcta.
- **Pruebas de condición:** se comprueba que las condiciones que hacen que un programa salte de un punto a otro están bien definidas (Bolaños Alonso, Sierra Alonso, & Alarcón Rodríguez, 2008).

### 2.3.4 Pruebas relacionadas con los cambios

¿Cómo afectan los cambios al sistema? ¿Qué ocurre si se cambia algo en el *software*?

En ese caso, puede ser necesario rehacer todos los test que sean necesarios: **re-testing**. Cuando el *software* falla y se encuentra el defecto, se modifica para que todo vaya bien. Se ha creado entonces una nueva versión de *software*. Esta nueva versión tendrá que pasar de nuevo todos los test. A este proceso se le llama confirmación de prueba.

En otras ocasiones, se pasará un test de nuevo a la misma versión de *software*. A esto se le llama **test regresivo** y se emplea cuando no se está seguro de que los resultados obtenidos anteriormente sean correctos (aunque lo parezcan) (Graham, van Veenendaal, Evans, & Black, 2010).

## 2.4 Otros tipos de pruebas

### 2.4.1 Pruebas automatizadas

Se ha hablado de distintos tipos de pruebas, pero ¿cómo hay que llevarlas a cabo? ¿Es necesario introducir de forma manual uno a uno cada posible caso? No, o al menos, no siempre. Se habla entonces de la **automatización de pruebas**. Pero para esto es necesario planear todo con cuidado: automatizar pruebas no implica conseguir el resultado que se espera o que todos los fallos desaparezcan. La automatización, si se hace bien, permite al usuario ahorrarse una gran cantidad de comprobaciones, y, por tanto, mucho tiempo. También disminuye la cantidad de errores humanos.

Las **pruebas de regresión** son un pilar importante en la automatización del *software testing*. Son un conjunto de pruebas planificadas que se ejecutan periódicamente y tienen como objetivo ver que el programa no sufre ningún fallo que antes no hubiera. Este tipo de pruebas se realizan habitualmente en *software* con más de una versión. Por ejemplo, se tiene un programa cuya versión 2.1 funciona correctamente. Tras instalar la siguiente versión, aparecen una serie de fallos que antes no había. Esto es detectado con las pruebas de regresión.

Las pruebas automatizadas se realizan a través de un *script* donde se han especificado los pasos que se tienen que dar. Algunas de las herramientas que se emplean en este campo son **Selenium**, útil en aplicaciones web; o **Junit**, que se aplica directamente sobre el código fuente de un programa (SeleniumHQ, SeleniumHQ.org, 2019).

Existen múltiples paradigmas para la automatización de pruebas. En el capítulo 7 se pueden ver las pruebas que se han realizado en este Trabajo de Fin de Grado. Los dos tipos de pruebas que se han realizado pertenecen a las categorías que se van a describir a continuación: **scripting** (Apartado 7.2 Pruebas con Selenium WebDriver) y **Record and Playback** (Apartado 7.1 Pruebas con Selenium IDE).

- **Scripting**: se definen los casos de prueba a partir de un *script* que logra ejecutar acciones sobre el sistema que se quiere probar. Las herramientas que se emplean pueden ser específicas para cada lenguaje (como **Selenium**), o una biblioteca o API del lenguaje (como **Junit**).
- **Record and Playback**: en ocasiones, los *scripts* necesarios para probar el *software* pueden ser muy complejos, por lo que se recurre al paradigma de “grabar y ejecutar”. El usuario realiza una serie de acciones sobre el *software* que se quiere probar mientras se desarrolla la acción de **grabar**. Las acciones que se han realizado se pueden convertir en un *script* con la acción **ejecutar**. Este paradigma es útil en aplicaciones web o aplicaciones móviles, donde el usuario puede desempeñar diferentes tareas fácilmente (Toledo Rodríguez, 2014).

#### 2.4.2 Pruebas de rendimiento

Las pruebas de rendimiento se utilizan para conocer los límites de los programas: cuál es su cuello de botella y cómo poder mejorarlo. Por ejemplo, se quiere comprobar cuantos usuarios puede haber conectados simultáneamente a una página web sin que ésta deje cumplir todas sus funciones. A menudo, el equipo que realiza estas pruebas no tiene a su disposición tantas cuentas de usuario como para conseguir “colgar” la aplicación, por eso se opta por automatizar las interacciones que se realizan a nivel de interfaz gráfica. Esta automatización es bastante más compleja que la automatización de pruebas funcionales, pero es necesaria para llevar a cabo las pruebas de rendimiento.

Las pruebas de rendimiento intentan dar respuesta a: cuándo dejará la aplicación de dar respuesta a sus usuarios debido a la carga, cuántos usuarios puede haber conectados, cuánto tiempo tarda el sistema en recuperarse en caso de sufrir una saturación o que puntos son los más delicados y por tanto actúan como cuello de botella. También hay que tener en cuenta, cuáles de todas las posibles acciones que puede realizar el usuario son las **más habituales**, y cuáles son las que **conlleven más carga de recursos**. Quizá no tenga sentido simular a mil usuarios realizando la acción que consume más recursos si el uso de esta es inusual. Hay que estudiar los posibles escenarios antes de llevar a cabo ningún tipo de *software testing*.

Algunas de las herramientas más habituales para este tipo de pruebas son **OpenSTA** o **JMeter** (ambas de código abierto) (Apache JMeter, 2020) (OpenSTA, 2020).

A diferencia del resto de pruebas de las que se ha hablado hasta el momento, las pruebas de rendimiento sí es aconsejable realizarlas en las fases finales del diseño del *software*. Someter a una sobrecarga a un sistema con fallos puede ser contraproducente si lo que se quiere comprobar es la capacidad de aguante que tiene el propio sistema (Toledo Rodríguez, 2014).

## 2.5 Pruebas automatizadas

Hasta ahora se ha hablado del *software testing* y por qué es necesario. También se han mencionado los principios en los que se basa, su ciclo de vida y los tipos de pruebas que se pueden realizar según la funcionalidad que se quiera comprobar. A continuación, se va a profundizar en las pruebas automatizadas (ya que va a ser el tipo de pruebas que se van a realizar en este Trabajo de Fin de Grado): los diferentes niveles en los que se descomponen, y en cuál de ellos se va a centrar este trabajo más adelante.

### 2.5.1 Niveles de las pruebas automatizadas

Cuando se habla de pruebas automatizadas, se puede hacer referencia a tres niveles diferentes. En la Ilustración 8 se puede ver la pirámide de Cohn, que representa estos niveles.

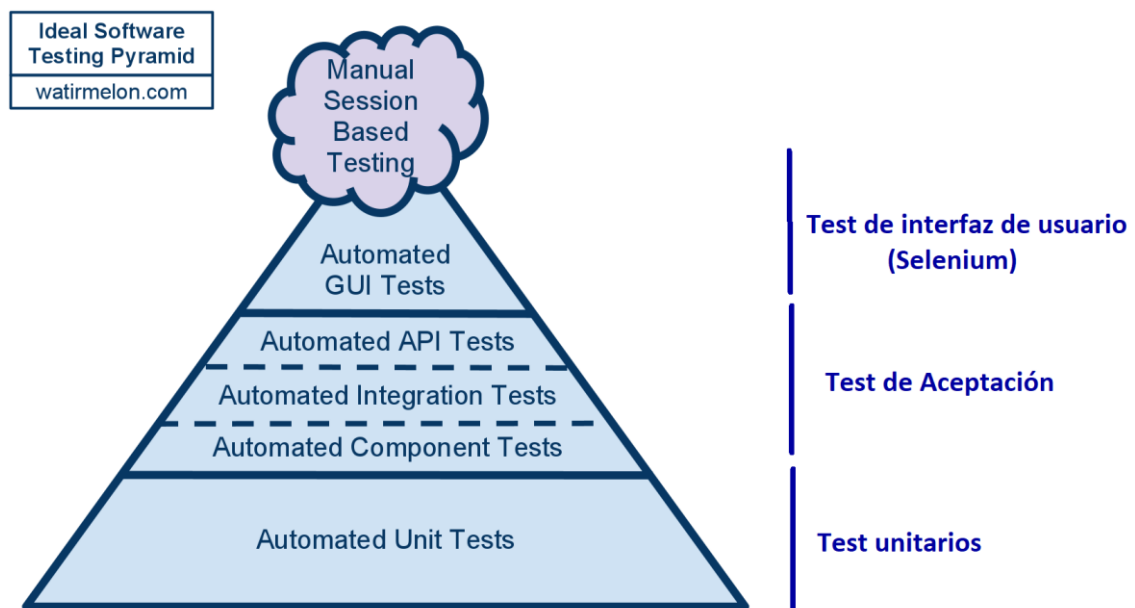


Ilustración 8: Niveles de pruebas automatizadas según la pirámide de Cohn (Grazas, 2019)



En el nivel más bajo de la pirámide se encuentran las **pruebas automatizadas unitarias**, en el segundo nivel se encuentran las **pruebas de aceptación** y en la parte superior de la pirámide están **las pruebas relacionadas con la interfaz de usuario**. Este último nivel es el que se van a centrar las pruebas que se realicen en este trabajo, más adelante. Las pruebas que se encuentran en la cúspide de la pirámide requerirán más esfuerzos y serán más complejos, por lo que se realizará un menor número de pruebas. Sin embargo, en los niveles más bajos de la pirámide es recomendable realizar un mayor número de pruebas automatizadas (Cohn, 2010).

Por lo tanto, según la pirámide de Cohn, una situación ideal de *software testing* sería aquella en la que hubiera:

- Muchas pruebas unitarias automatizados.
- Bastantes pruebas a nivel de integración de componentes y API.
- Pocas pruebas a nivel de interfaz de usuario.

¿De qué sirve realizar pruebas automatizadas a la interfaz de usuario? Para comprobar que el usuario no tenga ningún problema cuando vaya a interactuar con la aplicación que se está comprobando (en este caso, una aplicación web).

El programa que se va a emplear para realizar estas pruebas se llama **Selenium** y se centra en el tercer nivel de la pirámide. Selenium ofrece muchos tipos de pruebas a nivel de interfaz de usuario. Este Trabajo de Fin de Grado se va a centrar en dos tipos concretos: el *Play and Record* (Apartado 7.1 Pruebas con Selenium IDE) y el *Scripting* (Apartado 7.2 Pruebas con Selenium WebDriver).

### 3. LA APLICACIÓN WEB

Se va a diseñar una aplicación web que funcione como plataforma de diferentes tipos de encuestas y cuestionarios para profesores y alumnos. Los profesores serán los encargados de gestionar las preguntas de forma individual, y a partir de ellas generar cuestionarios. Esos cuestionarios podrán ser asignados al alumno o al grupo de alumnos que ellos quieran. También podrán gestionar a los alumnos de la plataforma: podrán registrar nuevos alumnos, ver los alumnos que hay registrados, modificar los datos de aquellos alumnos que ellos mismos hayan registrado, así como eliminarlos. Por último: el profesor podrá ver los cuestionarios que le han entregado los alumnos, y corregirlos, asignándolos una calificación si resulta pertinente.

La plataforma de cuestionarios tendrá un administrador, que se encargará de gestionar a los profesores: podrá registrarlos, verlos, editarlos y eliminarlos.

Por otro lado, los alumnos podrán ver los cuestionarios que se les han asignado, y realizarlos. También podrán ver las calificaciones que se les ha puesto en los cuestionarios que hayan realizado y que los profesores ya hayan corregido.

Como los alumnos y profesores deben estar registrados en la aplicación web para poder acceder a ella, será necesario que cada uno de ellos tenga un nombre de usuario y una contraseña. El nombre de usuario tendrá que ser único, ya que es la llave de acceso a la plataforma de cuestionarios.

La plataforma de cuestionarios se puede usar con dos fines diferentes: evaluar los conocimientos del alumno a través de **cuestionarios o tests**; o como **plataforma de encuestas**, para preguntar a los alumnos acerca de sus opiniones sobre sus profesores, las clases o cualquier otro asunto.

La plataforma plantea la posibilidad de trabajar con preguntas con cinco tipos de respuestas diferentes. Algunas de ellas están pensadas para los cuestionarios de evaluación, mientras que otras están más orientadas a las encuestas de opinión. No obstante, todas ellas se pueden utilizar de forma indistinta. Para los **cuestionarios de evaluación**, los profesores pueden crear:

- Preguntas con respuesta abierta: la respuesta podrá ser libre. El alumno tendrá un área de texto para escribir en ella lo que quiera.
- Preguntas de verdadero/falso: la pregunta tendrá dos posibles soluciones, entre las que el alumno deberá elegir la correcta.
- Pregunta sí/no: similar a la anterior.

- Preguntas con varias respuestas: el profesor podrá elegir el número de respuestas posibles: 2, 3, 4, 5 o 6. También podrá escribir el texto asociado a cada una de las respuestas.

Para las preguntas de verdadero/falso, sí/no o varias respuestas, el profesor podrá indicar la respuesta correcta. De esa forma, cuando vaya a corregir el cuestionario, podrá ver un indicador donde se marque cuántas preguntas ha acertado el alumno. Otras preguntas, como la pregunta con respuesta abierta, no podrán ser evaluadas de manera automática.

El tipo de preguntas recomendadas para las **encuestas de opinión** son:

- Preguntas sí/no.
- Preguntas graduales: el profesor podrá elegir una escala de 3, 5 o 10 elementos y asignar un texto a cada uno de los niveles, como, por ejemplo: “bien, regular o mal” en el caso de elegirse una escala de 3 elementos; o “nunca, casi nunca, a veces, frecuentemente o siempre” en el caso de elegirse una escala de 5 elementos.

Los cuestionarios de evaluación o las encuestas de opinión que gestione un profesor pondrán estar compuestos por todo tipo de preguntas combinándolas como resulte necesario. Pero solo podrán asignar a los alumnos cuestionarios o encuestas que hayan sido creados por ellos mismos.

### 3.1 Requisitos funcionales y no funcionales de la aplicación web

Los **requisitos funcionales** son las tareas que el sistema debe ser capaz de realizar. Su captura es imprescindible para que queden bien definidas las funcionalidades que debe llevar a cabo el sistema. A continuación, en la tabla, se muestran los requisitos funcionales de la aplicación web que se va a hacer (Pérez Juárez, 2017):

Código del requisito	Descripción del requisito funcional
FRQ-001	El sistema deberá identificar dos roles diferentes de usuarios: profesores y alumnos.
FRQ-002	El sistema deberá permitir a los usuarios de tipo “profesor” gestionar (crear, leer, actualizar y borrar) preguntas, que más tarde podrán pasar a formar parte de un cuestionario o encuesta. Las preguntas podrán ser de diferentes tipos:

	respuesta abierta, verdadero/falso, sí/no, varias opciones y graduales.
<b>FRQ-003</b>	El sistema deberá permitir al usuario de tipo “profesor” marcar la respuesta correcta asociada a una pregunta (para las preguntas de tipo verdadero/falso, sí/no o varias respuestas.
<b>FRQ-004</b>	El sistema deberá permitir a los usuarios de tipo “profesor” gestionar cuestionarios: el profesor podrá generar cuestionarios y encuestas a partir de las preguntas que haya en la base de datos del sistema.
<b>FRQ-005</b>	El sistema deberá permitir a los usuarios de tipo “profesor” gestionar a los alumnos: será el encargado de registrarles, introducir sus datos (nombre de usuario, nombre, primer apellido, segundo apellido, dirección, ciudad, correo electrónico, teléfono, clase y contraseña), modificarlos, y eliminarlos.
<b>FRQ-006</b>	El sistema deberá permitir a los usuarios de tipo “administrador” registrar a nuevos profesores.
<b>FRQ-007</b>	El sistema deberá permitir a los usuarios de tipo “profesor” ver los cuestionarios y encuestas entregados por los alumnos.
<b>FRQ-008</b>	El sistema deberá permitir a los usuarios de tipo “profesor” asignar a cada alumno los cuestionarios y encuestas que debe realizar.
<b>FRQ-009</b>	El sistema deberá permitir a los usuarios de tipo “alumno” ver los cuestionarios y encuestas que tienen pendientes de realizar.
<b>FRQ-010</b>	El sistema deberá permitir a los usuarios de tipo “alumno” rellenar y entregar los cuestionarios y encuestas que tengan que hacer.
<b>FRQ-011</b>	El sistema deberá permitir a los usuarios de tipo “alumno” consultar los resultados obtenidos en los cuestionarios y encuestas realizados.

*Tabla 1: Requisitos funcionales*

Los **requisitos no funcionales** son características requeridas del sistema, del proceso de desarrollo o del servicio prestado, que señalan una restricción del mismo. A

continuación, se muestra una tabla con los requisitos no funcionales del sistema (Pérez Juárez, 2017):

<b>Código del requisito</b>	<b>Descripción del requisito no funcional</b>
<b>NFRQ-001</b>	El sistema deberá tener una interfaz usable que permita al usuario acceder a los cuestionarios y encuestas y a toda la información relacionada.
<b>NFRQ-002</b>	El sistema deberá tener una interfaz que se adapte al tamaño disponible para renderizar el contenido web y se visualice de forma adecuada en las pantallas de los diferentes dispositivos móviles.
<b>NFRQ-003</b>	El sistema deberá permitir el acceso de un usuario a la plataforma mediante su nombre de usuario y contraseña. Un usuario solo podrá tener acceso a la información que le corresponde.
<b>NFRQ-004</b>	El sistema deberá garantizar que la información sobre un cuestionario o encuesta respondido (respuestas dadas y puntuación) solo sea visible para el alumno y el profesor involucrados.

*Tabla 2: Requisitos no funcionales*

### 3.2 Diagrama de Casos de Uso

Los diagramas de caso de uso resumen el comportamiento del sistema y de sus actores, mostrando los límites del sistema: permiten visualizar los requisitos funcionales del sistema. (Pérez Juárez, 2017).

En el sistema de la Plataforma de Cuestionarios hay tres actores diferentes: administrador, profesor y alumno. Cada uno de ellos tiene asociados diferentes casos de uso. En el Diagrama de Casos de Uso de la Ilustración 9 se puede ver de manera detallada todos los casos de uso que pueden realizar cada uno de los actores.

En los apartados siguientes se van a describir todos los casos de uso en detalle.

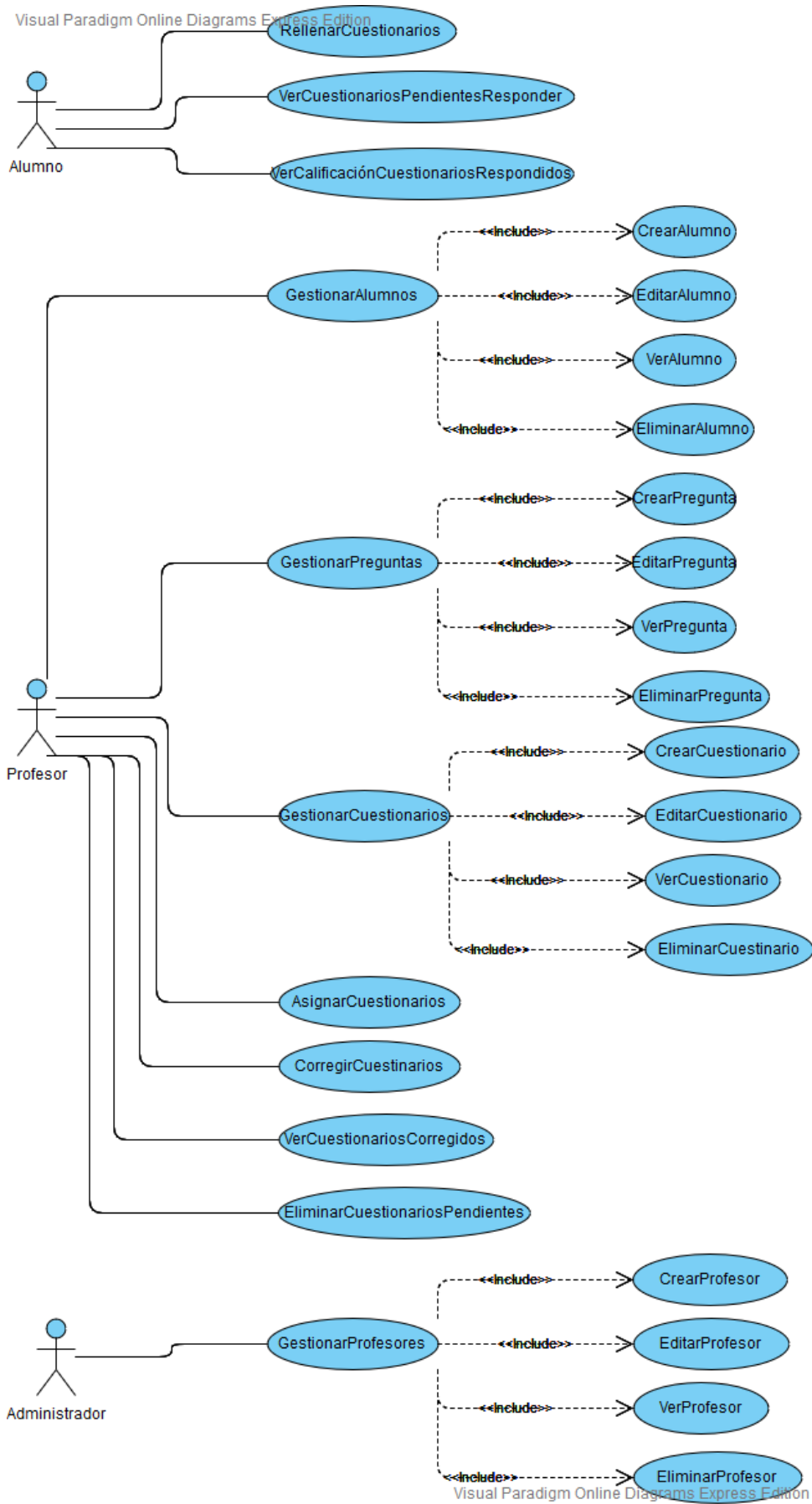


Ilustración 9: Diagrama de Casos de Uso

### 3.2.1 Descripción de los Casos de Uso

Los casos de uso describen el comportamiento del sistema al afrontar un requisito funcional. A continuación, se van a describir los once casos de uso identificados en el sistema.

#### 3.2.1.1 Gestionar Preguntas

El caso de uso **Gestionar Preguntas** es de tipo CRUD (*Create, Retrieve, Update, Delete*), es decir, que permite al actor que lo realiza, en este caso el profesor, crear, recuperar, actualizar o eliminar preguntas (en este caso). En la tabla siguiente se puede ver la descripción de este caso de uso (Pérez Juárez, 2017):

Identificador		Gestionar Preguntas	
<b>Descripción</b>	El profesor podrá añadir, consultar, modificar o borrar preguntas de varios tipos: abierta, verdadero/falso, sí/no, varias opciones, respuesta gradual con números o respuesta gradual con palabras.		
<b>Actores</b>	Profesor.		
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor.		
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>	
	1	El profesor elige la opción "Preguntas".	
	2	El sistema presenta las diferentes opciones para gestionar las preguntas.	
	3a	El Profesor elige la opción "Crear Preguntar".	
	4a	<<include>> Crear Pregunta.	
	3b	El Profesor elige la opción "Visualizar Preguntar".	
	4b	<<include>> Visualizar Pregunta.	
	3c	El Profesor elige la opción "Modificar Pregunta".	

	4c	<<include>> Modificar Pregunta.
	3d	El Administrador elige la opción “Eliminar Pregunta”.
	4d	<<include>> Eliminar Pregunta.
<b>Postcondición</b>	Las de los Casos de Uso incluidos.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-3	Si el profesor pulsa Cancelar, el Caso de Uso termina sin ninguna postcondición.

Tabla 3: Caso de Uso: Gestionar Preguntas

### 3.2.1.1.1 CrearPregunta

Identificador		Crear Pregunta
<b>Descripción</b>	El profesor podrá crear cualquier tipo de pregunta y seleccionar el tipo de respuesta que quiera.	
<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor.	
<b>Escenario principal de éxito</b>	Paso	Acción
	1	El profesor elige la opción “Crear pregunta”.
	2	El sistema muestra una pantalla con todos los campos necesarios para poder crear una pregunta: un espacio de texto para introducir el enunciado y un menú dinámico para elegir el tipo de respuesta.
	3	El profesor completa el campo del enunciado y selecciona el tipo de pregunta que quiere: de respuesta abierta, Sí/No, Verdadero/Falso, con varias respuestas posibles (y una de ellas correcta) y graduales (con varias opciones posibles).
	4	Si la pregunta que ha seleccionado el profesor es de respuesta Sí/No, Verdadero/Falso o con varias



		respuestas (y una correcta); el Profesor deberá indicar cuál de todas las respuestas es la correcta.
	5	Si la pregunta que ha seleccionado el profesor es de varias respuestas (con corrección o sin corrección), se podrá elegir el número de posibles respuestas que se quiere que tenga la pregunta y se podrá introducir el texto asociado a cada una de ellas.
	6	El sistema crea la nueva pregunta y la registra, asociándola un identificador nuevo y único. También registra cada una de las respuestas asociadas a la pregunta y asigna un identificador único a cada una de ellas.
<b>Postcondición</b>	La nueva pregunta y sus respuestas han sido añadidas al sistema.	
Flujo Alternativo	Paso	Acción
	1-3	Si el Profesor pulsa "Cancelar" el Caso de Uso termina sin ninguna postcondición.

Tabla 4: Caso de Uso: Crear Pregunta

### 3.2.1.1.2 VerPregunta

Identificador	Ver pregunta	
<b>Descripción</b>	El profesor podrá ver los detalles de las preguntas que hay en el sistema: su enunciado y el tipo de respuesta que tiene.	
<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen preguntas en el sistema, al menos una.	
	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción "Ver Preguntas".
	2	El sistema muestra una lista con todas las preguntas que hay registradas en la plataforma.

<b>Escenario principal de éxito</b>	3	El profesor pulsa el enunciado de la pregunta de la que quiere ver toda la información; o pulsa el botón “pregunta”.
	4	El sistema muestra la información detallada sobre la pregunta seleccionada (enunciado y tipo de respuesta asociada).
<b>Postcondición</b>	Los datos de la pregunta seleccionada han sido visualizados por el profesor.	

Tabla 5: Caso de Uso: Ver Pregunta

### 3.2.1.1.3 EditarPregunta

Identificador		Editar Pregunta
<b>Descripción</b>	El profesor podrá editar los datos de cualquier pregunta que haya sido creada por él mismo.	
<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen preguntas en el sistema creadas por el profesor, al menos una.	
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción “Editar Pregunta”.
	2	El sistema muestra una lista con todas las preguntas que hay registradas en la plataforma.
	3	El profesor selecciona la opción <b>Editar pregunta</b> que corresponde a la pregunta de la que quiere modificar algún dato.
	4	El sistema muestra todos los campos de la pregunta rellenos con la información actual de la pregunta (enunciado, tipo de respuesta, valores de las respuestas y respuesta marcada como correcta) como campos editables.

	5	El profesor modifica el valor de los campos que quiera y pulsa el botón <b>Cambiar</b> .
	6	El sistema registra las modificaciones realizadas en los datos de la pregunta.
<b>Postcondición</b>	La información de la pregunta seleccionada ha sido modificada en el sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-5	Si el Profesor pulsa el botón “Cancelar”, el Caso de Uso termina sin ninguna postcondición.

Tabla 6: Caso de Uso: Editar pregunta

#### 3.2.1.1.4 EliminarPregunta

Identificador		Eliminar Pregunta	
<b>Descripción</b>	El profesor podrá eliminar una pregunta que hay sido creada por él mismo.		
<b>Actores</b>	Profesor.		
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen preguntas en el sistema creadas por el profesor, al menos una.		
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>	
	1	El profesor elige la opción “Eliminar Pregunta”	
	2	El sistema muestra una lista con todas las preguntas que hay registradas en la plataforma.	
	3	El profesor selecciona la opción <b>Eliminar pregunta</b> que corresponde a la pregunta que quiere eliminar.	
	4	El sistema busca la pregunta seleccionada a través de su identificador, así como las respuestas que tiene vinculadas. Una vez identificadas todas las respuestas y la pregunta, las elimina del sistema.	

<b>Postcondición</b>	La pregunta ha sido eliminada del sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-3	Si el Profesor pulsa el botón “Atrás” del buscador, el Caso de Uso termina sin ninguna postcondición.

Tabla 7: Caso de Uso: Eliminar pregunta

### 3.2.1.2 Gestionar Alumnos

El caso de uso **Gestionar Alumnos** es de tipo CRUD (*Create, Retrieve, Update, Delete*), es decir, que permite al actor que lo realiza, en concreto al actor profesor registrar en el sistema a nuevos alumnos, ver la información sobre cada uno de los alumnos, editar la información sobre ellos y eliminar a los alumnos que quiera.

Identificador		Gestionar Alumnos	
<b>Descripción</b>	El profesor podrá añadir, consultar, modificar o borrar alumnos.		
<b>Actores</b>	Profesor.		
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor.		
<b>Escenario principal éxito</b>	<b>Paso</b>	<b>Acción</b>	
	1	El profesor elige la opción “Alumnos”.	
	2	El sistema muestra un menú con las opciones disponibles para gestionar alumnos de la plataforma.	
	3a	El profesor elige la opción “Crear Alumno”.	
	4a	<<include>> Crear alumno.	
	3b	El Profesor elige la opción “Ver Alumno”.	
	4b	<<include>> Ver alumno.	
	3c	El Profesor elige la opción “Editar Alumno”.	
	4c	<<include>> Editar alumno.	
	3d	El profesor elige la opción “Eliminar Alumno”.	

	4d	<<include>> Eliminar alumno.
<b>Postcondición</b>	Las de los Casos de Uso incluidos.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-3	Si el profesor pulsa Cancelar, el Caso de Uso termina sin ninguna postcondición.

Tabla 8: Caso de Uso: Gestionar Alumnos

### 3.2.1.2.1 CrearAlumno

Identificador		CrearAlumno	
<b>Descripción</b>	El profesor podrá registrar a cualquier alumno.		
<b>Actores</b>	Profesor.		
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor.		
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>	
	1	El profesor elige la opción "Registrar Alumno".	
	2	El sistema muestra una pantalla con todos los campos necesarios para poder registrar a un nuevo alumno: nombre de usuario, nombre, primer apellido, segundo apellido, dirección, ciudad, teléfono, correo electrónico, clase y contraseña.	
	3	El profesor completa todos los campos obligatorios, y los optativos y selecciona el botón Registrar.	
	4	El sistema crea la cuenta del nuevo alumno y le asigna un identificador que se empleará en futuras ocasiones. A partir de ahora, el alumno puede acceder al sistema introduciendo su nombre de usuario y su contraseña.	
<b>Postcondición</b>	El nuevo alumno ha sido incluido en el sistema		
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>	
	1-4	Si el Profesor pulsa "Cancelar" el Caso de Uso termina sin ninguna postcondición.	

Tabla 9: Caso de Uso: Crear Alumno

3.2.1.2.2 VerAlumno

Identificador		Ver alumno	
<b>Descripción</b>	El profesor podrá ver los detalles de los alumnos registrados en el sistema.		
<b>Actores</b>	Profesor.		
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen alumnos en el sistema, al menos uno.		
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>	
	1	El profesor elige la opción "Ver Alumnos".	
	2	El sistema muestra una lista con todos los alumnos que hay registrados en la plataforma.	
	3	El profesor pulsa el nombre del usuario correspondiente al alumno del que quiere ver toda la información; o pulsa el botón <b>Ver Alumno</b> .	
	4	El sistema muestra la información detallada sobre el alumno seleccionado (nombre del usuario, nombre, primer apellido, segundo apellido, dirección, ciudad, correo electrónico, teléfono y clase).	
<b>Postcondición</b>	Los datos del alumno seleccionado han sido visualizados por el profesor.		

Tabla 10: Caso de Uso: Ver Alumno

3.2.1.2.3 EditarAlumno

Identificador		Editar Alumno	
<b>Descripción</b>	El profesor podrá editar los datos de cualquier alumno que haya sido registrado por él mismo.		
<b>Actores</b>	Profesor.		

<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen alumnos en el sistema añadidos por el profesor, al menos uno.	
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción "Editar Alumno".
	2	El sistema muestra una pantalla a todos los alumnos que hay registrados en la plataforma.
	3	El profesor selecciona la opción <b>Editar alumno</b> que corresponde al alumno del que quiere modificar algún dato.
	4	El sistema muestra todos los campos del usuario rellenos con la información actual del usuario (nombre, primer apellido, segundo apellido, dirección, ciudad, teléfono, correo y clase) como campos editables.
	5	El profesor modifica los campos que quiera y pulsa el botón <b>Cambiar</b> .
6	El sistema registra las modificaciones realizadas en los datos del alumno.	
<b>Postcondición</b>	La información del alumno seleccionado ha sido modificada en el sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-4	Si el Profesor pulsa "Borrar", los datos del alumno que haya modificado vuelven al estado inicial (antes de que los cambiara).

Tabla 11: Caso de Uso: Editar Alumno

#### 3.2.1.2.4 EliminarAlumno

Identificador

Eliminar Alumno

<b>Descripción</b>	El profesor podrá eliminar un alumno que haya sido registrado por él mismo.	
<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen alumnos en el sistema añadidos por el profesor, al menos uno.	
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción “Eliminar Alumno”.
	2	El sistema muestra una lista con los alumnos que hay registrados en la plataforma.
	3	El profesor selecciona la opción <b>Eliminar alumno</b> que corresponde al alumno que quiere eliminar.
	4	El sistema busca al alumno seleccionado a través de su identificador y elimina su cuenta y todo lo relacionado con ella.
<b>Postcondición</b>	El alumno ha sido eliminado del sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-3	Si el Profesor pulsa el botón “Atrás” del buscador, el Caso de Uso termina sin ninguna postcondición.

Tabla 12: Caso de Uso: Eliminar Alumno.

### 3.2.1.3 Gestionar Cuestionarios

El caso de uso **Gestionar Cuestionarios** es de tipo CRUD (*Create, Retrieve, Update, Delete*), es decir, que permite al actor que lo realiza, en este caso el profesor, entre otras acciones, registrar en la plataforma los cuestionarios que más tarde se podrán asignar a los alumnos.

<b>Identificador</b>	<b>Gestionar Cuestionarios</b>
<b>Descripción</b>	El profesor podrá añadir, consultar, modificar o borrar cuestionarios.



<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor.	
<b>Escenario principal éxito</b>	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción "Cuestionarios".
	2	El sistema muestra un menú con las opciones disponibles para gestionar cuestionarios de la plataforma.
	3a	El profesor elige la opción "Crear Cuestionario".
	4a	<<include>> Crear cuestionario.
	3b	El Profesor elige la opción "Ver Cuestionario".
	4b	<<include>> Ver cuestionario.
	3c	El Profesor elige la opción "Editar Cuestionario".
	4c	<<include>> Editar cuestionario.
	3d	El profesor elige la opción "Eliminar Cuestionario".
4d	<<include>> Eliminar cuestionario.	
<b>Postcondición</b>	Las de los Casos de Uso incluidos.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-3	Si el profesor pulsa Cancelar, el Caso de Uso termina sin ninguna postcondición.

Tabla 13: Caso de Uso: Gestionar Cuestionarios

### 3.2.1.3.1 CrearCuestionario

Identificador	Crear Cuestionario
<b>Descripción</b>	El profesor podrá crear un cuestionario a partir de las preguntas que hay registradas en el sistema.
<b>Actores</b>	Profesor.

<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen preguntas en el sistema, al menos una.	
<b>Escenario principal éxito</b>	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción "Crear Cuestionario".
	2	El sistema muestra una pantalla con todos los campos necesarios para poder crear el nuevo cuestionario: el título y las preguntas que hay en el sistema; de las que podrá elegir tantas como quiera.
	3	El profesor completa todos los campos obligatorios (el título del cuestionario), y los optativos (las preguntas que se quieren incluir en el cuestionario) y selecciona el botón <b>Crear</b> .
	4	El sistema crea el nuevo cuestionario con las preguntas escogidas y lo registra en el sistema.
<b>Postcondición</b>	El nuevo cuestionario está registrado en el sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-4	Si el Profesor pulsa "Cancelar" el Caso de Uso termina sin ninguna postcondición.

Tabla 14: Caso de Uso: Crear Cuestionario

### 3.2.1.3.2 VerCuestionario

<b>Identificador</b>	<b>Ver cuestionario</b>	
<b>Descripción</b>	El profesor podrá ver los detalles del cuestionario seleccionado: su título y las preguntas que lo componen.	
<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen cuestionarios en el sistema, al menos uno.	
	<b>Paso</b>	<b>Acción</b>

<b>Escenario principal de éxito</b>	1	El profesor elige la opción “Ver Cuestionarios” de la barra de opciones.
	2	El sistema muestra una pantalla con todos los cuestionarios que hay registrados en la plataforma.
	3	El profesor pulsa el título del cuestionario del que quiere ver toda la información; o pulsa el botón <b>Ver Cuestionario</b> .
	4	El sistema muestra la información detallada sobre el cuestionario seleccionado (título y preguntas que lo conforman).
<b>Postcondición</b>	Los datos del cuestionario han sido visualizados con éxito.	

Tabla 15: Caso de Uso: Ver Cuestionario

### 3.2.1.3.3 EditarCuestionario

Identificador		Editar Cuestionario	
<b>Descripción</b>	El profesor podrá editar los datos de cualquier cuestionario que haya sido creado por él mismo.		
<b>Actores</b>	Profesor.		
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen cuestionarios en el sistema añadidos por el profesor, al menos uno.		
<b>Escenario principal de</b>	<b>Paso</b>	<b>Acción</b>	
	1	El profesor elige la opción “Editar Cuestionario”.	
	2	El sistema muestra una pantalla con todos los cuestionarios que hay registrados en la plataforma.	
	3	El profesor selecciona la opción <b>Editar cuestionario</b> que corresponde al cuestionario del que quiere modificar algún dato.	

<b>éxito</b>	4	El sistema muestra todos los campos del cuestionario rellenos con la información actual del mismo como campos editables.
	5	El profesor modifica los campos que quiera y pulsa el botón <b>Cambiar</b> .
	6	El sistema registra las modificaciones realizadas en los datos del cuestionario.
<b>Postcondición</b>	La información del cuestionario seleccionado ha sido modificada en el sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-5	Si el Profesor pulsa "Borrar", los datos del cuestionario que se hayan modificado vuelven al estado inicial (antes de que los cambiara).
	1-5	Si el Profesor pulsa el botón "Cancelar", el Caso de Uso termina sin ninguna postcondición.

Tabla 16: Caso de Uso: Editar Cuestionario

#### 3.2.1.3.4 EliminarCuestionario

Identificador	Eliminar Cuestionario	
<b>Descripción</b>	El profesor podrá eliminar un cuestionario que haya sido creado por él mismo.	
<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Profesor. Existen cuestionarios en el sistema añadidos por el profesor, al menos uno.	
	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción "Eliminar Cuestionario".
	2	El sistema muestra una pantalla con todos los cuestionarios que hay registrados en la plataforma.

<b>Escenario principal de éxito</b>	3	El profesor selecciona la opción <b>Eliminar cuestionario</b> que corresponde al cuestionario que quiere eliminar.
	4	El sistema busca al cuestionario seleccionado a través de su identificador, y lo elimina.
<b>Postcondición</b>	El cuestionario ha sido eliminado del sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-3	Si el Profesor pulsa el botón “Atrás” del buscador, el Caso de Uso termina sin ninguna postcondición.

Tabla 17: Caso de Uso: Eliminar Cuestionario

### 3.2.1.4 AsignarCuestionarios

El caso de uso **Asignar Cuestionarios** permite al actor con rol de profesor asignar a cada uno de los alumnos los cuestionarios que quiere que hagan. En la siguiente tabla se puede ver la descripción del caso de uso (Pérez Juárez, 2017).

Identificador		Asignar Cuestionarios
<b>Descripción</b>	El profesor podrá asignar uno o más cuestionarios de la plataforma a uno o más alumnos.	
<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como profesor. Hay cuestionarios creados en el sistema por el profesor y alumnos registrados, al menos un cuestionario y un alumno.	
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción “Asignar Cuestionarios”.
	2	El sistema muestra una lista con el título de todos los cuestionarios que hay registrados en el sistema.
	3	El Profesor selecciona los cuestionarios que quiere asignar.
	4	El sistema muestra una lista con todos los alumnos que hay registrados en el sistema.

	5	El Profesor selecciona los alumnos a los que quiere asignar los cuestionarios que ha seleccionado previamente.
	6	El sistema asigna los cuestionarios a los alumnos que deben realizarlos.
<b>Postcondición</b>	Los alumnos tienen asignados los cuestionarios que les corresponden y pueden acceder a ellos para completarlos.	
<b>Flujo Alternativo</b>	Paso	Acción
	1-6	Si el Profesor pulsa "Cancelar" el Caso de Uso finalizará sin lograr ninguna postcondición por lo que no se asignará ningún cuestionario a ningún alumno.

Tabla 18: Caso de Uso Asignar Cuestionarios

### 3.2.1.5 VerCuestionariosCorregidos

El caso de uso **Ver Cuestionarios Corregidos** permite al actor con rol de profesor ver una lista donde aparecen los cuestionarios que sus alumnos han respondido. Esta lista se divide en dos categorías: **Cuestionarios Corregidos** y **Cuestionarios sin Corregir**.

Identificador	Ver Cuestionarios Respondidos	
<b>Descripción</b>	El profesor podrá ver los cuestionarios que sus alumnos han respondido.	
<b>Actores</b>	Profesor.	
<b>Precondición</b>	El sistema ha reconocido al usuario como profesor. Hay cuestionarios creados en el sistema y han sido respondidos por los alumnos, al menos un cuestionario ha sido respondido por un alumno.	
	<b>Paso</b>	<b>Acción</b>
	1ª	El profesor elige la opción "Corregir".
	2ª	El sistema muestra una lista con todos los cuestionarios respondidos que aún no han sido corregidos por el profesor. En ellos se muestra el

<b>Escenario principal de éxito</b>		título del cuestionario, el alumno que lo ha realizado, la clase a la que pertenece y si el profesor quiere ver el cuestionario.
	3ª	El Profesor ve los cuestionarios sin corregir.
	1b	El profesor elige la opción "Ver Corregidos".
	2b	El sistema muestra una lista con todos los cuestionarios respondidos que ya han sido corregidos por el profesor. En ellos se muestra el título del cuestionario, el alumno que lo ha realizado, la clase a la que pertenece, la puntuación obtenida y si el profesor quiere ver el cuestionario.
	3b	El profesor ve los cuestionarios corregidos.
<b>Postcondición</b>	El profesor accede a los cuestionarios que le han entregado sus alumnos, tanto los que ya ha corregido, como los que aún no ha corregido.	
<b>Flujo Alternativo</b>	Paso	Acción
	1-3	Si el Profesor pulsa "Atrás" en el buscador, o el logo de la página, o cualquiera de las pestañas con diferentes opciones de la aplicación web, el Caso de Uso finalizará sin ninguna postcondición.

Tabla 19: Caso de Uso: Ver Cuestionarios Respondidos

### 3.2.1.6 CorregirCuestionarios

El caso de uso **Corregir Cuestionarios** permite al actor con rol de profesor asignar una calificación a los cuestionarios realizados por los alumnos.

Identificador	Corregir Cuestionario
<b>Descripción</b>	El profesor podrá corregir los cuestionarios que sus alumnos han realizado y entregado.
<b>Actores</b>	Profesor.

<b>Precondición</b>	El sistema ha reconocido al usuario como profesor. Hay cuestionarios creados en el sistema y han sido respondidos por los alumnos, al menos un cuestionario ha sido respondido por un alumno.	
<b>Escenario principal de éxito</b>	Paso	Acción
	1	El profesor elige la opción "Corregir".
	2	El sistema muestra una lista con todos los cuestionarios respondidos que aún no han sido corregidos por el profesor. En ellos se muestra el título del cuestionario, el alumno que lo ha realizado, la clase a la que pertenece y si el profesor quiere ver el cuestionario.
	3	El Profesor selecciona la opción "Ver Cuestionario".
	4	El sistema le muestra al profesor el cuestionario y las respuestas del alumno.
	5	El profesor selecciona la opción "Evaluar".
	6	El sistema le vuelve a mostrar el cuestionario con las respuestas del alumno, y permite al profesor introducir la calificación que le quiera dar (puede ser tanto numérica, como con palabra: suspenso, aprobado, bien, etc.)
	7	El profesor selecciona la opción "Terminar de Evaluar".
	8	El sistema almacena la calificación del alumno relacionada con ese cuestionario.
<b>Postcondición</b>	El profesor ha corregido el cuestionario del alumno y el resultado ha sido almacenado en el sistema.	
<b>Flujo Alternativo</b>	Paso	Acción
	1-7	Si el Profesor pulsa "Cancelar", el Caso de Uso finalizará sin ninguna postcondición.

Tabla 20: Caso de Uso: Corregir Cuestionario



### 3.2.1.7 EliminarCuestionariosPendientes

El caso de uso **Eliminar Cuestionarios Pendientes** permite al actor con rol de profesor eliminar los cuestionarios que ha asignado a un alumno y éste no ha resuelto todavía.

Identificador		Eliminar Cuestionarios Pendientes	
<b>Descripción</b>	El profesor podrá eliminar de entre uno a todos los cuestionarios que él mismo ha asignado, a un alumno en concreto. Los cuestionarios asignados que quiera eliminar no deben haber sido resueltos todavía.		
<b>Actores</b>	Profesor		
<b>Precondición</b>	El sistema ha reconocido al usuario como profesor.		
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>	
	<b>1</b>	El profesor va a “Ver Alumnos”.	
	<b>2</b>	El sistema muestra una lista con todos los alumnos registrados en el sistema.	
	<b>3</b>	El profesor selecciona “Ver Cuestionarios Pendientes”, relacionado con el alumno del que quiere ver los cuestionarios que tiene pendientes de resolver.	
	<b>4</b>	El sistema muestra los cuestionarios pendientes del alumno seleccionado.	
	<b>5</b>	El profesor selecciona los cuestionarios pendientes de resolver del alumno que quiere eliminar y selecciona “Eliminar”.	
<b>6</b>	El sistema elimina los cuestionarios seleccionados de la lista de cuestionarios pendientes de resolver por el alumno.		
<b>Postcondición</b>	Los cuestionarios seleccionados ya no están en la lista de cuestionarios pendientes del alumno.		
<b>Flujo</b>	Paso	Acción	

<b>Alternativo</b>	<b>4.1</b>	El sistema no muestra ningún cuestionario pendiente porque el alumno no tiene cuestionarios pendientes de responder.
	<b>5.1</b>	El profesor selecciona la opción “Cancelar” y el Caso de Uso termina sin aplicarse la postcondición.
	<b>5.2</b>	El profesor selecciona la casilla “Todo” para eliminar todos los cuestionarios que el alumno tiene asignados.
	<b>4-5</b>	El profesor selecciona la opción “Cancelar” y el Caso de Uso termina sin aplicarse la postcondición.

Tabla 2121: Caso de Uso: Eliminar Cuestionarios Pendientes

### 3.2.1.8 VerCuestionariosPendientesResponder

El caso de uso **Ver Cuestionarios Pendientes de Responder** permite al actor con rol de alumno ver los cuestionarios que los profesores le han asignado.

<b>Identificador</b>		<b>Ver Cuestionarios Pendientes de Responder</b>	
<b>Descripción</b>	El alumno podrá ver los cuestionarios que aún no ha respondido.		
<b>Actores</b>	Alumno		
<b>Precondición</b>	El sistema ha reconocido al usuario como alumno.		
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>	
	1	El sistema muestra al alumno la lista de los cuestionarios que tiene pendientes de resolver.	
<b>Postcondición</b>	El alumno sabe cuáles son los cuestionarios que le quedan por hacer.		
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>	
	1	Si el alumno no tiene cuestionarios pendientes por responder, el sistema muestra un mensaje indicando que no hay cuestionarios pendientes.	

Tabla 22. Caso de Uso: Ver Cuestionarios Pendientes de Responder

### 3.2.1.9 Rellenar Cuestionario

El caso de uso **Rellenar Cuestionario** permite al actor con rol de alumno completar la lista de cuestionarios que tiene pendientes por realizar, y entregárselos resueltos al profesor.

Identificador		Rellenar cuestionario	
<b>Descripción</b>	El alumno podrá hacer los cuestionarios que los profesores le hayan asignado.		
<b>Actores</b>	Alumno		
<b>Precondición</b>	El sistema ha reconocido al usuario como alumno. El alumno tiene cuestionarios pendientes por responder, al menos uno.		
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>	
	1	El alumno selecciona la opción "Realizar Cuestionario" correspondiente al cuestionario que quiere hacer.	
	2	El sistema muestra al alumno el cuestionario que tiene que rellenar: las preguntas que lo forman y las posibles respuestas para cada una de ellas (en caso de estar predefinidas).	
	3	El alumno completa las respuestas para cada una de las preguntas. Cuando finaliza, pulsa el botón "Enviar".	
	4	El sistema almacena las respuestas correspondientes a cada pregunta y se las asocia al alumno.	
<b>Postcondición</b>	El cuestionario ha sido completado correctamente y las respuestas han sido enviadas al profesor responsable del cuestionario, para que más tarde pueda corregirlo.		
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>	
	1-3	Si el alumno pulsa el botón "Atrás" del navegador o el logo de la Plataforma de Cuestionarios, el Caso de Uso terminará sin ninguna postcondición.	

Tabla 23: Caso de Uso: Rellenar Cuestionario

### 3.2.1.10 VerCalificacionCuestionariosRespondidos

El caso de uso **Ver Calificación Cuestionarios Respondidos** permite al actor con rol de alumno ver la calificación que le han puesto los profesores a los cuestionarios que ya ha respondido.

Identificador		Ver Calificación Cuestionarios Respondidos	
<b>Descripción</b>	El alumno podrá ver la calificación de los cuestionarios que le han asignado los profesores en los cuestionarios que ya ha respondido.		
<b>Actores</b>	Alumno		
<b>Precondición</b>	El sistema ha reconocido al usuario como alumno.		
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>	
	1	El sistema muestra al alumno la lista de los cuestionarios que ya ha respondido y han sido evaluados. Aparece el título de los cuestionarios y su puntuación.	
<b>Postcondición</b>	El alumno sabe cuáles han sido sus calificaciones en los cuestionarios que ya ha entregado.		
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>	
	1	Si el alumno no tiene cuestionarios corregidos, el sistema muestra un mensaje indicando que no hay cuestionarios corregidos.	

Tabla 24: Caso de Uso: VerCalificacionCuestionariosRespondidos

### 3.2.1.11 GestionarProfesores

El caso de uso **Gestionar Profesores** es de tipo CRUD (*Create, Retireve, Update, Delete*), es decir, que permite al actor que lo realiza, en este caso al actor Administrador, registrar en el sistema a nuevos profesores, ver la información sobre cada uno de los profesores, editar la información sobre ellos y eliminar a los profesores que quiera.

Identificador	Gestionar Profesor
---------------	--------------------

<b>Descripción</b>	El administrador podrá añadir, consultar, modificar o borrar profesores.	
<b>Actores</b>	Administrador.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Administrador.	
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>
	1	El administrador elige la opción "Profesores".
	2	El sistema muestra un menú con las opciones disponibles para gestionar profesores de la plataforma.
	3a	El administrador elige la opción "Crear Profesor".
	4a	<<include>> Crear profesor.
	3b	El administrador elige la opción "Ver Profesor".
	4b	<<include>> Ver profesor.
	3c	El administrador elige la opción "Editar Profesor".
	4c	<<include>> Editar profesor.
	3d	El administrador elige la opción "Eliminar Profesor".
4d	<<include>> Eliminar profesor.	
<b>Postcondición</b>	Las de los Casos de Uso incluidos.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-3	Si el administrador pulsa Cancelar, el Caso de Uso termina sin ninguna postcondición.

Tabla 25: Caso de Uso: GestionarProfesores

### 3.2.1.10.1 CrearProfesor

<b>Identificador</b>	<b>CrearProfesor</b>
<b>Descripción</b>	El administrador podrá registrar a cualquier profesor.

<b>Actores</b>	Administrador.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Administrador.	
<b>Escenario principal de éxito</b>	<b>Paso</b>	<b>Acción</b>
	1	El profesor elige la opción "Registrar Profesor".
	2	El sistema muestra una pantalla con todos los campos necesarios para poder registrar a un nuevo profesor: nombre del usuario, nombre, primer apellido, segundo apellido, dirección, ciudad, teléfono, correo y contraseña.
	3	El administrador completa todos los campos obligatorios y selecciona el botón Registrar.
	4	El sistema crea la cuenta del nuevo profesor y le asigna un identificador que se empleará en futuras ocasiones.
<b>Postcondición</b>	El nuevo profesor ha sido incluido en el sistema. A partir de ahora, el profesor puede acceder al sistema introduciendo su nombre de usuario y su contraseña.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-4	Si el Profesor pulsa "Cancelar" el Caso de Uso termina sin ninguna postcondición.

Tabla 26: Caso de Uso: CrearProfesor

### 3.2.1.10.2 VerProfesor

<b>Identificador</b>	<b>Ver profesor</b>	
<b>Descripción</b>	El administrador podrá ver los detalles de los profesores registrados en el sistema.	
<b>Actores</b>	Administrador.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Administrador. Existen profesores en el sistema, al menos uno.	
	<b>Paso</b>	<b>Acción</b>

<b>Escenario principal éxito</b>	<b>de</b>	1	El administrador elige la opción “Ver Profesores”.
		2	El sistema muestra una lista con todos los profesores que hay registrados en la plataforma.
		3	El administrador pulsa el nombre del profesor del que quiere ver toda la información; o pulsa el botón <b>Ver Profesor</b> .
		4	El sistema muestra la información detallada sobre el profesor seleccionado (nombre del usuario, nombre, primer apellido, segundo apellido, dirección, ciudad, correo electrónico y teléfono).
<b>Postcondición</b>		Los datos del profesor seleccionado han sido visualizados por el administrador.	

Tabla 27: Caso de Uso: Ver Profesor

### 3.2.1.10.3 EditarProfesor

Identificador		Editar Profesor	
<b>Descripción</b>	El administrador podrá editar los datos de cualquier profesor que haya sido registrado.		
<b>Actores</b>	Administrador.		
<b>Precondición</b>	El sistema ha reconocido al usuario como Administrador. Existen profesores en el sistema, al menos uno.		
<b>Escenario principal éxito</b>	<b>de</b>	<b>Paso</b>	<b>Acción</b>
		1	El administrador elige la opción “Editar Profesor”.
		2	El sistema muestra una pantalla a todos los profesores que hay registrados en la plataforma.
		3	El administrador selecciona la opción <b>Editar profesor</b> que corresponde al profesor del que quiere modificar algún dato.
		4	El sistema muestra todos los campos del profesor rellenos con la información actual del usuario

		(nombre, primer apellido, segundo apellido, dirección, ciudad, teléfono y correo) como campos editables.
	5	El administrador modifica los campos que quiera y pulsa el botón <b>Cambiar</b> .
	6	El sistema registra las modificaciones realizadas en los datos del profesor.
<b>Postcondición</b>	La información del profesor seleccionado ha sido modificada en el sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-4	Si el Administrador pulsa "Borrar", los datos del alumno que haya modificado vuelven al estado inicial (antes de que los cambiara).

Tabla 28: Caso de Uso: Editar Profesor

#### 3.2.1.10.4 EliminarProfesor

Identificador	EliminarProfesor	
<b>Descripción</b>	El administrador elimina un profesor de la plataforma de cuestionarios.	
<b>Actores</b>	Administrador.	
<b>Precondición</b>	El sistema ha reconocido al usuario como Administrador. Existen profesores en el sistema, al menos uno.	
	<b>Paso</b>	<b>Acción</b>
	1	El administrador elige la opción "Eliminar Profesor".
	2	El sistema muestra una lista con los profesores que hay registrados en la plataforma.
	3	El administrador selecciona la opción <b>Eliminar profesor</b> que corresponde al profesor del que quiere eliminar.



<b>Escenario principal de éxito</b>	4	El sistema busca al profesor seleccionado a través de su identificador y elimina su cuenta.
<b>Postcondición</b>	El profesor ha sido eliminado del sistema.	
<b>Flujo Alternativo</b>	<b>Paso</b>	<b>Acción</b>
	1-3	Si el Administrador pulsa el botón “Atrás” del buscador, el Caso de Uso termina sin ninguna postcondición.

Tabla 29: Caso de Uso: Eliminar Profesor

### 3.3 Tablas de la base de datos

Una **base de datos** está formada por un conjunto de información relacionada entre sí. Las bases de datos relacionales representan el modelo de bases de datos más empleado actualmente. Una **base de datos relacional** se caracteriza por representar los datos y las relaciones entre ellos a través de **tablas**. Cada tabla tiene un nombre único que la identifica y una serie de filas que representan los registros de dicha tabla (Jiménez Capel, 2015).

Una base de datos eficiente es aquella que está **normalizada**: cumple ciertas reglas que indican cómo se deben almacenar los datos en ellas. Estas reglas reciben el nombre de formas normales, y son tres (Unidade de Apoio para el Aprendizaje, 2019):

- **Primera forma normal (1FN)**: la primera forma está compuesta por dos indicaciones.
  - Todos los atributos y valores almacenados en las columnas deben ser indivisibles.
  - No deben existir grupos de valores repetidos.
- **Segunda forma normal (2FN)**: la segunda forma indica que no tienen que existir dependencias funcionales parciales, es decir, todos los valores de las columnas de una fila tienen que depender de la clave primaria de la propia fila.
- **Tercera forma normal (3FN)**: la tercera forma indica que no deben existir dependencias transitivas entre las columnas de una tabla, es decir, las columnas que no forman parte de la clave primaria deben depender solo de la clave.

En la Ilustración 10 se puede ver el esquema de la base de datos que se ha diseñado para la aplicación web de la plataforma de cuestionarios. La base de datos con la que se ha trabajado está formada por nueve tablas diferentes: admin, profesores, alumnos, preguntas, posibles\_respuestas, cuestionarios, pregunta\_cuestionario, respuesta\_alumno y cuestionario\_alumno. A continuación, se describirá cada una de las tablas, los campos que la forman y como están relacionadas entre sí.

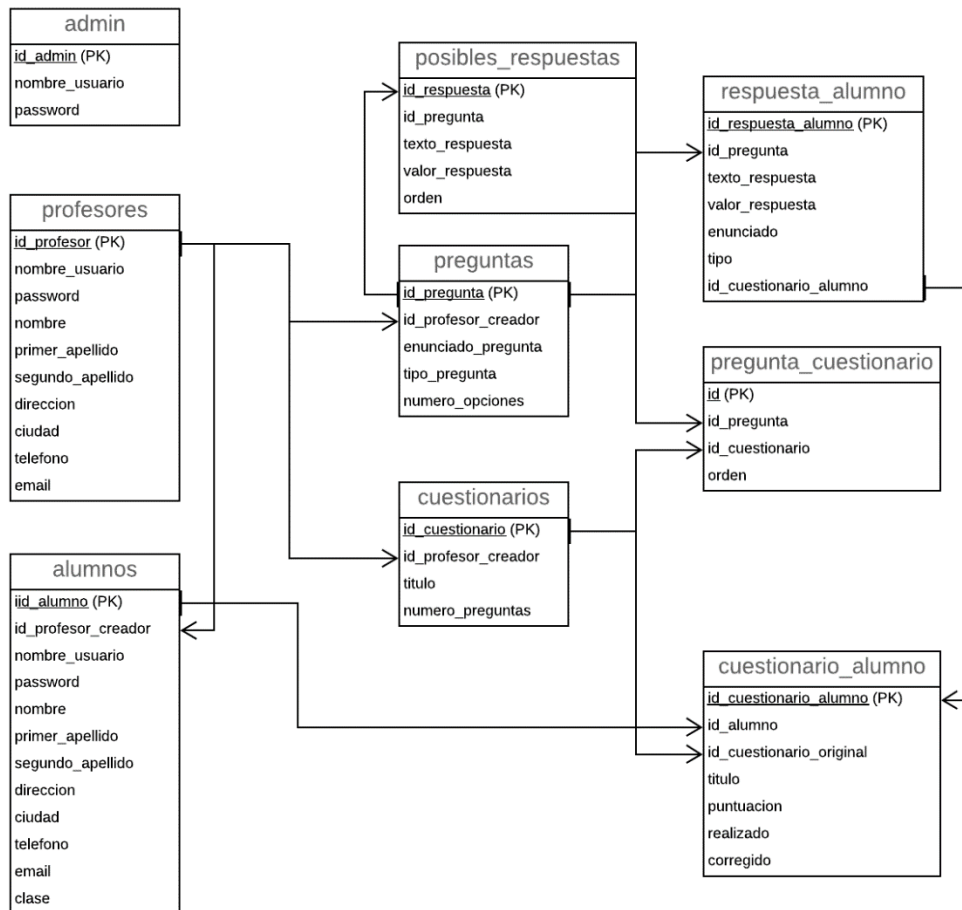


Ilustración 10: Esquema de las tablas de la base de datos

### 3.3.1 Admin

La tabla **admin** contiene la información relacionada con el administrador de la plataforma de cuestionarios. El administrador es el usuario que gestiona a los profesores de la plataforma. En la Ilustración 11 se pueden ver sus campos.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	<b>id_admin</b> 🔑	int(1)			No	Ninguna		AUTO_INCREMENT
2	<b>nombre_usuario</b>	varchar(5)	utf8_spanish_ci		No	Ninguna		
3	<b>password</b>	varchar(5)	utf8_spanish_ci		No	Ninguna		

Ilustración 11: Tabla admin

- **id\_admin**: es la clave primaria a partir de la que se va a identificar al administrador.
- **nombre\_usuario**: nombre que emplea el administrador para acceder a la plataforma.
- **password**: contraseña que emplea el administrador, junto al nombre de usuario, para acceder a la plataforma de cuestionarios.

### 3.3.2 Profesores

La tabla **profesores** contiene toda la información relacionada con los usuarios profesor. Los profesores pueden gestionar alumnos, preguntas y cuestionarios; así como asignar cuestionarios a los alumnos (y eliminar dicha asignación si el cuestionario aún no se ha realizado), ver los cuestionarios que sus alumnos han entregado, corregirlos y ver los cuestionarios que ya han corregido. Los campos de esta tabla se pueden ver en la Ilustración 12.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	<b>id_profesor</b> 🔑	int(5)			No	Ninguna		AUTO_INCREMENT
2	<b>nombre_usuario</b>	varchar(15)	utf8_spanish_ci		No	Ninguna		
3	<b>password</b>	varchar(20)	utf8_spanish_ci		No	Ninguna		
4	<b>nombre</b>	varchar(20)	utf8_spanish_ci		No	Ninguna		
5	<b>primer_apellido</b>	varchar(20)	utf8_spanish_ci		No	Ninguna		
6	<b>segundo_apellido</b>	varchar(20)	utf8_spanish_ci		No	Ninguna		
7	<b>direccion</b>	varchar(40)	utf8_spanish_ci		No	Ninguna		
8	<b>ciudad</b>	varchar(15)	utf8_spanish_ci		No	Ninguna		
9	<b>telefono</b>	int(15)			No	Ninguna		
10	<b>email</b>	varchar(40)	utf8_spanish_ci		No	Ninguna		

Ilustración 12: Tabla profesores

- **id\_profesor**: clave primaria de identificación de los profesores.
- **nombre\_usuario**: nombre que emplea el profesor para acceder a la plataforma.
- **password**: contraseña que emplea el profesor, junto al nombre de usuario, para acceder a la plataforma de cuestionarios.

- nombre: nombre propio del profesor.
- primer\_apellido: primer apellido del profesor.
- segundo\_apellido: segundo apellido del profesor.
- direccion: dirección del colegio dónde el profesor trabaja.
- ciudad: ciudad del colegio dónde el profesor trabaja.
- telefono: número de teléfono de contacto con el profesor en el colegio.
- email: dirección de correo de contacto del profesor en el colegio.

### 3.3.3 Alumnos

La tabla **alumnos** contiene toda la información relacionada con los usuarios alumno. Los alumnos pueden responder a los cuestionarios que se les han asignado y ver la calificación de los cuestionarios que ya han respondido y ya han sido corregidos por los profesores. Los campos de esta tabla se pueden ver en la Ilustración 13.


#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id_alumno 	int(5)			No	Ninguna		AUTO_INCREMENT
2	id_profesor_creador	int(5)			No	Ninguna		
3	nombre_usuario	varchar(20)	utf8_spanish_ci		No	Ninguna		
4	password	varchar(20)	utf8_spanish_ci		No	Ninguna		
5	nombre	varchar(20)	utf8_spanish_ci		No	Ninguna		
6	primer_apellido	varchar(20)	utf8_spanish_ci		No	Ninguna		
7	segundo_apellido	varchar(20)	utf8_spanish_ci		No	Ninguna		
8	direccion	varchar(50)	utf8_spanish_ci		No	Ninguna		
9	ciudad	varchar(20)	utf8_spanish_ci		No	Ninguna		
10	telefono	int(15)			No	Ninguna		
11	email	varchar(40)	utf8_spanish_ci		No	Ninguna		
12	clase	varchar(5)	utf8_spanish_ci		No	Ninguna		

Ilustración 13: Tabla alumnos

- id\_alumno: clave primaria de identificación de los alumnos.
- id\_profesor\_creador: identificador del profesor que ha registrado al alumno en la plataforma de cuestionarios.
- nombre\_usuario: nombre que emplea el alumno para acceder a la plataforma.
- password: contraseña que emplea el alumno, junto al nombre de usuario, para acceder a la plataforma de cuestionarios.

- nombre: nombre propio del alumno.
- primer\_apellido: primer apellido del alumno.
- segundo\_apellido: segundo apellido del alumno.
- direccion: dirección del alumno.
- ciudad: ciudad del alumno.
- telefono: número de teléfono del alumno.
- email: dirección de correo del alumno.
- clase: clase del alumno

### 3.3.4 Preguntas

La tabla **preguntas** contiene toda la información sobre cada una de las preguntas que se han registrado en la plataforma de cuestionarios. Las preguntas registradas pueden ser de diferentes tipos, y solo el profesor que las ha creado puede gestionarlas. Cada pregunta tiene asociada una o más **posibles respuestas** (tabla de la que se hablará más tarde, en el apartado 4.3.5 Posibles respuestas). Los campos de la tabla **preguntas** se puede ver en la Ilustración 14.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id_pregunta 	int(5)			No	Ninguna		AUTO_INCREMENT
2	id_profesor_creador	int(5)			No	Ninguna		
3	enunciado_pregunta	varchar(100)	utf8_spanish_ci		No	Ninguna		
4	tipo_pregunta	int(2)			No	Ninguna		
5	numero_opciones	int(2)			No	Ninguna		

Ilustración 14: Tabla preguntas

- id\_pregunta: clave primaria de identificación de las preguntas.
- id\_profesor\_creador: identificador del profesor que registró la pregunta en la plataforma de cuestionarios.
- enunciado\_pregunta: texto que indica el enunciado de la pregunta.
- tipo\_pregunta: indica el tipo de respuesta que se espera de esta pregunta. Existen los siguientes valores:
  - 0: la respuesta relacionada con esa pregunta es abierta: un texto que será introducido por el alumno.

- 1: la respuesta relacionada con esa pregunta puede ser Sí o No.
  - 2: la respuesta relacionada con esa pregunta puede ser Verdadero o Falso.
  - 3: la respuesta relacionada con esa pregunta puede ser una entre varias posibles, donde cada una de ellas es una posible respuesta válida a una pregunta. Solo una de las respuestas introducidas puede marcarse como correcta.
  - 4: la respuesta relacionada con esa pregunta puede ser una entre varias posibles, donde cada una de las respuestas representa un grado de conformidad, habitualidad, etc.
- numero\_opciones: indica el número de opciones que puede tener asociadas la pregunta.

### 3.3.5 Posibles respuestas

La tabla de **posibles\_respuestas** contiene la información de cada una de las posibles respuestas asociadas a una pregunta. Por ejemplo, si una pregunta tiene cinco respuestas asociadas, se crearán cinco entradas en la tabla posibles\_respuestas: una por cada posible respuesta. En la Ilustración 15 se puede ver la tabla posibles\_respuestas.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id_respuesta	int(5)			No	Ninguna		AUTO_INCREMENT
2	id_pregunta	int(5)			No	Ninguna		
3	texto_respuesta	varchar(100)	utf8_spanish_ci		No	Ninguna		
4	valor_respuesta	int(2)			No	Ninguna		
5	orden	int(2)			No	Ninguna		

Ilustración 15: Tabla posibles respuestas

- id\_respuesta: clave primaria que sirve como identificador de cada una de las respuestas.
- id\_pregunta: asocia cada una de las respuestas con la pregunta a la que corresponden. Este identificador se emplea para localizar a todas las posibles respuestas que pertenecen a una pregunta.
- texto\_respuesta: contiene el texto de cada una de las respuestas. Este texto puede haber sido asignado por el profesor, o puede estar predeterminado por el sistema (como ocurre en los casos de las preguntas con respuestas tipo Sí/No y Verdadero/Falso).

- **valor\_respuesta:** contiene cuatro posibles valores:
  - -1: la respuesta no es correcta ni incorrecta, lo tendrá que decidir el profesor. Este valor de respuesta está asociado a las preguntas de tipo 0.
  - 0: la respuesta es incorrecta. Este valor de respuesta está asociado a las preguntas de tipo 1, 2 y 3.
  - 1: la respuesta es correcta. Este valor de respuesta está asociado a las preguntas de tipo 1, 2 y 3.
  - 2: la respuesta no es correcta ni incorrecta. Este valor está asociado a las preguntas de tipo 4.
- **orden:** indica el orden de las respuestas.

### 3.3.6 Cuestionarios

La tabla de **cuestionarios** contiene la información de cada uno de los cuestionarios que hay en la plataforma de cuestionarios. Un cuestionario está formado por un título y por una serie de preguntas. Para poder relacionar al cuestionario con las preguntas que lo forman, se necesita una tabla auxiliar: **pregunta\_cuestionario**. Esta tabla relaciona una pregunta con un cuestionario al que pertenece. Una pregunta puede pertenecer a más de un cuestionario.

La tabla **cuestionarios** también va a estar relacionada con la tabla **cuestionario\_alumno** (de la que se hablará en el apartado 4.3.8). En la Ilustración 16 se puede ver la tabla **cuestionarios**.


#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	<b>id_cuestionario</b> 	int(5)			No	Ninguna		AUTO_INCREMENT
2	<b>id_profesor_creador</b>	int(5)			No	Ninguna		
3	<b>título</b>	varchar(30)	utf8_spanish_ci		No	Ninguna		
4	<b>numero_preguntas</b>	int(2)			No	Ninguna		


Ilustración 16: Tabla cuestionarios

- **id\_cuestionario:** clave primaria de identificación de los cuestionarios.
- **id\_profesor\_creador:** identificador del usuario que ha creado el cuestionario. Este campo es importante, porque un profesor solo podrá asignar a los alumnos los cuestionarios que él mismo haya creado.
- **título:** nombre que se le ha asignado al cuestionario cuando se ha creado.

- `numero_preguntas`: número de preguntas que componen al cuestionario.

### 3.3.7 `Pregunta_cuestionario`

La tabla **`pregunta_cuestionario`** sirve como relación entre las tablas **`preguntas`** y **`cuestionarios`**. Como se ha comentado en el apartado 4.3.6, los cuestionarios pueden estar formados por diferentes preguntas, y una pregunta puede aparecer en más de un cuestionario. ¿Cómo se sabe qué preguntas forman cada cuestionario? A través de la tabla **`pregunta_cuestionario`**. En la Ilustración 17 se puede ver esta tabla.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	<b>id</b> 	int(5)			No	Ninguna		AUTO_INCREMENT
2	<b>id_pregunta</b>	int(5)			No	Ninguna		
3	<b>id_cuestionario</b>	int(5)			No	Ninguna		
4	<b>orden</b>	int(2)			No	Ninguna		

*Ilustración 17: Tabla `pregunta_cuestionario`*

- `id`: clave primaria de la tabla `pregunta_cuestionario`.
- `id_pregunta`: identificador de la pregunta que se va a relacionar con un cuestionario.
- `id_cuestionario`: identificador del cuestionario al que pertenece una pregunta.
- `orden`: el orden que ocupa la pregunta identificada por `id_pregunta` en el cuestionario identificado por `id_cuestionario`.

### 3.3.8 `Cuestionario_alumno`

La tabla **`cuestionario_alumno`** contiene la información relacionada con un cuestionario que se ha asignado a un alumno. Cuando un profesor asigna uno o más cuestionarios a uno o más alumnos, se crea una fila en la tabla **`cuestionario_alumno`**. El fin de esta tabla es almacenar datos relacionados con el cuestionario que el alumno tiene asignado: si lo ha realizado o no, si ha sido corregido o no, etc. También se va a almacenar el valor del título del cuestionario, para que, en caso de que el cuestionario original sea eliminado, se pueda tener este dato (y no se pierda). En la Ilustración 18 se puede ver la tabla de **`cuestionario_alumno`**.



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	<b>id_cuestionario_alumno</b> 🗝️	int(5)			No	Ninguna		AUTO_INCREMENT
2	<b>id_cuestionario_original</b>	int(5)			No	Ninguna		
3	<b>id_alumno</b>	int(5)			No	Ninguna		
4	<b>puntuacion</b>	varchar(15)	utf8_spanish_ci		Sí	NULL		
5	<b>corregido</b>	int(1)			No	0		
6	<b>realizado</b>	int(1)			No	0		
7	<b>titulo</b>	varchar(30)	utf8_spanish_ci		No	Ninguna		

Ilustración 18: Tabla cuestionario\_alumno

- **id\_cuestionario\_alumno**: clave primaria que identifica a cada uno de los cuestionarios asignados a los alumnos.
- **id\_cuestionario\_original**: identificador del cuestionario original con el que está relacionado.
- **id\_alumno**: identificador del alumno al que se le ha asignado el cuestionario.
- **puntuacion**: indica la calificación que el usuario profesor pone a las respuestas dadas por el alumno. Puede ser tanto en formato numérico, como con palabras (Sobresaliente, bien, no evaluable, etc.)
- **titulo**: indica el título del cuestionario original. Se almacena de forma redundante para que en caso de que se elimine el cuestionario original, este dato no se pierda. Es necesario tenerlo siempre, porque el título aparecerá cuando se corrija un cuestionario o cuando se quieran ver los cuestionarios corregidos.
- **realizado**: *flag* que indica si el cuestionario ha sido realizado por el alumno o no (por defecto tiene el valor 0).
  - 0 → El cuestionario no ha sido realizado.
  - 1 → El cuestionario ha sido realizado.
- **corregido**: *flag* que indica si el cuestionario ha sido corregido por el profesor o no (por defecto tiene el valor 0).
  - 0 → El cuestionario no ha sido corregido.
  - 1 → El cuestionario ha sido corregido.

### 3.3.9 Respuesta\_alumno

La tabla **respuesta\_alumno** contiene la información sobre las respuestas dadas por los alumnos a cada una de las preguntas que forman un cuestionario. En ella se almacena

el identificador de la **pregunta** a la que corresponde la respuesta dada, el texto de la respuesta y el enunciado, el valor de la respuesta seleccionada, el tipo de pregunta a la que corresponde y el identificador de la tabla **cuestionario\_alumno** con las que está relacionada. Algunos de los datos mencionados pueden parecer innecesarios, pero no lo son. Se han almacenado de manera redundante para poder conservar su contenido, en caso de que las preguntas con las que están relacionadas las respuestas de los alumnos sean eliminadas. En la Ilustración 19 se puede ver la tabla **respuesta\_alumnos**.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	id_respuesta_alumno	int(5)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
2	id_cuestionario_alumno	int(5)			No	Ninguna			Cambiar Eliminar Más
3	id_pregunta	int(5)			No	Ninguna			Cambiar Eliminar Más
4	texto_respuesta	varchar(50)	utf8_spanish_ci		No	Ninguna			Cambiar Eliminar Más
5	valor_respuesta	int(2)			No	Ninguna			Cambiar Eliminar Más
6	enunciado	varchar(1000)	utf8_spanish_ci		No	Ninguna			Cambiar Eliminar Más
7	tipo_pregunta	int(1)			No	Ninguna			Cambiar Eliminar Más

Ilustración 19: Tabla respuesta\_alumno

- id\_respuesta\_alumno: clave primaria que identifica a cada una de las respuestas dadas por el alumno.
- id\_cuestionario\_alumno: relaciona la respuesta dada por el alumno con el cuestionario al que pertenece.
- id\_pregunta: relaciona la respuesta con la pregunta original a la que pertenece.
- texto\_respuesta: almacena el texto asociado a la respuesta.
- valor\_respuesta: contiene cuatro posibles valores:
  - -1: la respuesta no es correcta ni incorrecta, lo tendrá que decidir el profesor. Este valor de respuesta está asociado a las preguntas de tipo 0.
  - 0: la respuesta es incorrecta. Este valor de respuesta está asociado a las preguntas de tipo 1, 2 y 3.
  - 1: la respuesta es correcta. Este valor de respuesta está asociado a las preguntas de tipo 1, 2 y 3.
  - 2: la respuesta no es correcta ni incorrecta. Este valor está asociado a las preguntas de tipo 4.
- enunciado: contiene el enunciado relacionado con la respuesta que el alumno ha dado.
- tipo\_pregunta: indica el tipo de pregunta con el que está asociada la respuesta del alumno.

## 4. HERRAMIENTAS EMPLEADAS PARA EL DESARROLLO DE LA APLICACIÓN WEB

Para desarrollar una aplicación web son necesarias diferentes herramientas y lenguajes de programación. Una aplicación web se descompone en dos partes: la parte **cliente** y la parte **servidor**. Cada una de las partes tiene un objetivo determinado, y es necesario emplear distintos tipos de lenguajes de programación en cada una de ellas.

### 4.1 La parte cliente

La parte cliente es el parte con la que interactúa el usuario para comunicarse con un servidor web y poder intercambiar recursos con él. Para construir esta parte de la aplicación web se emplean lenguajes como **HTML** (*Hypertext Markup Language*), **CSS** (*Cascading Style Sheets*) o *JavaScript*. HTML es la parte encargada de la estructura, CSS se encarga de dar una presentación visual a dicha estructura y JavaScript hace todo lo demás (más adelante se pondrán ejemplos, en los apartados 5.1.1 HTML, 5.1.2 CSS y 5.1.3 *JavaScript*) (Gauchat, 2012)

#### 4.1.1 HTML

La primera publicación en la que se habló de HTML como tal apareció en 1991, bajo el nombre de “*HTML Tags*” y fue escrita por Tim Berners-Lee. En dicha publicación se describieron los 18 elementos que formaban la primera versión de HTML y que fue la propulsora del HTML como se conoce hoy. Aunque desde 1991 hasta la actualidad ha pasado por diferentes versiones, y cada una de ellas ha aportado mejoras y funcionalidades extra (Berners-Lee, 1992).

HTML es el lenguaje usado para crear páginas web. Aunque existen muchas versiones, la empleada actualmente es **HTML5**. HTML no es un lenguaje de programación, es un lenguaje de marcado: un sistema para identificar y describir los diferentes componentes de un documento, como pueden ser su cabecera, los párrafos de los que se compone, etc. Las características por defecto de cada elemento de HTML están definidas en el navegador y pueden ser modificadas por el diseñador de la página web a través de CSS (apartado 5.1.2 CSS) (Robbins, 2012).

Después de la primera versión de HTML, aparecieron HTML2 (noviembre, 1995), HTML3 (enero, 1997), HTML4 (diciembre, 1997) y HTML5 (octubre, 2014). Estas dos últimas versiones tuvieron a su vez versiones intermedias. La versión más actual es HTML5.2, publicada en diciembre de 2017.

Los navegadores web reciben los ficheros HTML de un servidor web o desde un almacenamiento local y “convierten” los ficheros en una página web multimedia. HTML describe la estructura de la página web. Un documento HTML está compuesto por **elementos o bloques** que conforman **la estructura de la página web**. Las imágenes o formularios interactivos son elementos que también forman parte del resultado final de la página web. HTML proporciona al programador los medios necesarios para crear documentos estructurados a partir de campos semánticos de texto como: la cabecera (*heading*), párrafos, listas, enlaces, etc. Estos elementos son delimitados con etiquetas de apertura y cierre. Por ejemplo, para marcar el comienzo y final de la cabecera, se utilizan las etiquetas `<header></header>` (Consortium, 2019).

Un fichero HTML tiene una estructura de elementos anidados, y se les reconoce porque también van acompañados de los caracteres `<>`. Como se ha indicado en el ejemplo de la cabecera, los caracteres delimitan el comienzo y el final de un bloque o elemento. A mayores, algunas etiquetas pueden contener atributos, como puede ser la etiqueta ***input***, en la que se puede indicar qué tipo de entrada se debería introducir, su valor u otras informaciones. En las líneas de código mostradas a continuación se puede ver un ejemplo de ello. En este caso, la etiqueta del elemento tipo ***input*** tiene asociados un conjunto de atributos que indican características sobre dicho elemento. El tipo: un *radioButton*; el nombre de dicho *radioButton* (tipo\_respuesta); el valor asignado al *radioButton* (“0”); un identificador relacionado con él (tipo\_respuesta\_0); una función de *JavaScript* que se ejecutará en caso de pulsar dicho *radioButton* (mostrarReferencia();) y el texto que acompañará a dicho *radioButton* (Respuesta abierta) (w3schools, 2015) .

```
<input type="radio" name="tipo_respuesta" value="0" id="tipo_respuesta" onclick="mostrarReferencia();" > Respuesta abierta.
```

También existen otras etiquetas como `<p>` para introducir nuevos párrafos, `<a>` para crear enlaces. En este último caso se pueden añadir atributos como, por ejemplo: `<a href =>` para introducir direcciones URL en el enlace creado, como se puede ver a continuación. En la línea mostrada, se ha introducido un enlace al fichero *registrar\_pregunta.php*.

```
<a href="registrar_pregunta.php">
```

HTML5 apareció con el fin de facilitar la implementación de las aplicaciones web. Algunas de las mejoras que introduce son:

- Aparición de nuevos elementos estructuradores como *section*, *article*, *aside*, *header*, *footer* o *nav*. Estos elementos permiten organizar la página web fácilmente (WHATWG (Apple, 2019)).
- Ampliación del elemento *input*: permite más tipos de datos que en versiones anteriores, como por ejemplo **emails**, **urls**, **colores**, etc (WHATWG (Apple, 2019)).
- Permite trabajar con elementos como el **audio** o **vídeo**, permitiendo poder controlar la reproducción de dichos elementos sin tener que recurrir a plugins como puede ser *Flash* (WHATWG (Apple, 2019)).

#### 4.1.2 CSS

El 10 de octubre de 1994, el colaborador del CERN, Hakon Wium Lie, propuso la primera versión de CSS. Muchos lenguajes de hoja de estilo habían sido propuestos como representante de este campo hasta el momento, pero CSS acabó imponiéndose frente a los demás, consiguiendo ser el lenguaje referente en cuanto a hoja de estilo (Wium Lie & Bos, 2018).

CSS se utiliza para describir la presentación de un documento cuya estructura está especificada con algún lenguaje marcado, como por ejemplo HTML. Permite la organización y presentación del contenido de una página web: su diseño, los colores utilizados o el tipo de fuente que se emplea (W3C, 2019).

CSS también permite particularizar el estilo de una página web cuando se accede a la misma desde diferentes medios: se puede acceder a una página web con un estilo pensado para ver por pantalla o para la impresión, para percibir su contenido de forma auditiva (por voz), o incluso en un formato Braille para todos aquellos dispositivos táctiles que estén habilitados para ello (Clark, 2019).

Al igual que ocurre con HTML y con el fin de corregir errores e incorporar nuevas funcionalidades, han ido apareciendo nuevas versiones de CSS con el paso de los años, hasta llegar a la versión actual: CSS3 (Krall, 2019).

¿Cómo y dónde se emplea CSS? Aunque el navegador de internet utilizado habitualmente posee unos estilos por defecto, es habitual que dichos estilos no cubran las necesidades del usuario. CSS permite a los diseñadores web modificar los estilos y dar a la página web la apariencia que quieran. CSS organiza el espacio de la página web por **elementos**, que pueden ser: **elementos block** o **elementos inline**. Los primeros organizan la página en bloques que se van posicionando uno sobre otro en orden descendente; mientras que los otros permiten posicionar los bloques uno al lado de otro, siguiendo una línea horizontal (Gauchat, 2012).

Aunque CSS se puede mezclar con HTML, habitualmente se crea un fichero (o varios) específico en el que se recoge todo el código CSS. Para que sea efectivo, se utiliza en elemento de HTML `<link>` para insertar este fichero en los documentos que lo empleen. Esto es realmente útil, porque permite al usuario usar estilos de elementos declarados en un único punto desde cualquier parte o fichero de la aplicación web. Así, solo es necesario cambiar lo que se quiera en el fichero `.css`, y este cambio aparecerá en todos los puntos desde donde se haga referencia a dicho elemento (Gauchat, 2012).

A continuación, se muestra un ejemplo de código CSS (Ilustración 20). En el código se puede ver como se declara un estilo para un bloque `.form-consulta`. Cada vez que se haga referencia a este tipo de bloque en la página web, el mismo aparecerá cumpliendo las condiciones descritas.

- **max-width:** es una propiedad que fija la anchura máxima del bloque a 450 píxeles.
- **background:** es una propiedad que define el color del fondo. Hay diferentes nomenclaturas para hacer referencia a los colores: a través de código rgb (rgb(24,180,201)), con palabras clave (black) o con el símbolo almohadilla (#) seguido de un número que represente a un color (#f00). Por defecto es blanco.
- **padding:** define un espaciado entre el contenedor y el contenido del tamaño que se quiera. Habitualmente se indica en píxeles o porcentajes.
- **font-family:** define la fuente que se va a emplear, en caso de que haya texto.
- **color:** define el color del texto. Por defecto es negro.
- **margin-bottom:** define el espacio que se deja entre el elemento al que se le asigna el *margin-bottom* y el elemento situado en la parte inferior.
- **float:** esta propiedad especifica la alineación de un elemento: en este caso, los elementos de tipo `.form-consulta` aparecen centrados en la pantalla.
- **border-radius:** el uso de esta propiedad es muy específico. En este caso se debe a que los bordes del formulario (dónde se emplea el estilo `.form-consulta`), tienen forma redondeada en lugar de la tradicional esquina. Los valores de esta propiedad indican los píxeles de curvatura que se quiere.
- **border:** indica el grosor del borde de la caja, así como su color.
- **margin-top:** indica el espaciado de separación que se deja respecto al margen superior de la página, o respecto al elemento que esté por encima de este bloque.

```

3  .form-consulta {
4      max-width: 450px;
5      background: #1a4a53;
6      padding: 25px;
7      font-family: 'Source Sans Pro', sans-serif;
8      color: #ffffff;
9      margin-bottom: 15px;
10     float: center;
11     border-radius: 35px 0px 35px 0px;
12     -moz-border-radius: 35px 0px 35px 0px;
13     -webkit-border-radius: 0px 0px 35px 35px;
14     border: 2px solid #000000;
15     margin-top: -50px;
16 }

```

Ilustración 20: Ejemplo de código CSS

Programar ficheros CSS puede llegar a ser muy complicado y tedioso, por ello, existen multitud de plantillas (de código abierto o de pago) que ofrecen al usuario un diseño amigable sin necesidad de tener que especificar, elemento a elemento, el aspecto de la web.

#### 4.1.3 JavaScript

Al inicio de los sitios web, HTML y CSS eran suficiente para llevar a cabo el desarrollo de las páginas web, pero según fue avanzando la tecnología, la aparición de nuevos lenguajes se hizo necesaria. Se necesitaba que las páginas web y los usuarios de estas pudieran interactuar entre sí, también se quería que las páginas web tuvieran un comportamiento dinámico que gustara al usuario final. La solución que se propuso fue *JavaScript* (Negrino & Smith, 2005).

*JavaScript* (formalmente conocido como *ECMAScript*) fue desarrollado por **Brendan Eich** (trabajador de la compañía Netscape) en 1995. Pese a llevar en su nombre la palabra *Java*, este lenguaje no tiene nada que ver con su homónimo. *JavaScript* es un lenguaje interpretado y tiene como único fin el permitir implementar páginas web que interactúen con sus usuarios. No se debe concebir *JavaScript* para un uso que no esté relacionado con las páginas web. *JavaScript* fue utilizado principalmente en formularios web (tenía la capacidad de validar los datos introducidos en ellos sin la necesidad de pasarlos por un servidor web) y en el manejo de imágenes (Suehring, 2008).

Microsoft incluyó soporte para *JavaScript* en su navegador de Internet (Internet Explorer) en 1996. Internet Explorer también daba soporte a otro lenguaje similar (pero

no igual) a *JavaScript*: VBScript. El problema que se planteó entonces fue detectar qué lenguaje empleaba el navegador: *JavaScript* o VBScript. (Suehring, 2008).

Con el paso de los años han aparecido muchos estándares para *JavaScript*. A continuación, se va a describir uno de los más relevantes hoy: el DOM (*Document Object Model*). Fue desarrollado por W3C (*World Wide Web*) y lo definió como: “una interfaz neutra de plataforma y lenguaje que permite a los programadores y los scripts acceder dinámicamente para actualizar el contenido, la estructura y el estilo de los documentos”. El DOM crea una estructura para documentos HTML y permite escribir sobre estos objetos (Suehring, 2008).

*JavaScript* no puede funcionar sin HTML ni CSS: los tres lenguajes están ligados entre sí y conforman la base de las páginas web. HTML proporciona el nivel estructural, CSS proporciona el nivel de presentación y *JavaScript* proporciona la interacción usuario-web (Sawyer McFarland, 2008).

Todo el código *JavaScript* que aparezca en un documento HTML debe comenzar y terminar con las etiquetas `<script></script>` o vincularse a un manejador de evento. Otra de las cosas que permite *JavaScript* (a parte de la validación de formularios o el manejo de imágenes) es la de crear páginas HTML personalizadas en función de las elecciones del usuario. *JavaScript* permite al diseñador web realizar todas estas acciones a través de los objetos, las propiedades y los métodos (Negrino & Smith, 2005).

- **Objetos:** se llaman objetos a las ventanas, formularios, elementos de formularios (botones, campos para rellenar, etc.), etc.
- **Propiedades:** describen a un objeto, cómo, por ejemplo, el título de una ventana de navegación, o el botón de validación de un formulario. Las propiedades pueden modificar a los objetos.
- **Métodos:** son las acciones que pueden realizar los objetos. Un ejemplo de ello sería el método `click()` para un botón.

En la aplicación web que se ha llevado a cabo en este Trabajo Fin de Grado, se ha empleado *JavaScript* para crear páginas HTML personalizadas en función de las elecciones del usuario. Concretamente, en la sección **Crear Pregunta**. A continuación, se van a mostrar dos partes de dos ficheros diferentes. La primera de ellas corresponde a la llamada a una función *JavaScript* a través de un `radioButton` que pertenece a un formulario HTML. La segunda de ellas corresponde a la declaración de la función llamada (escrita en *JavaScript*), que realiza la acción que le corresponde.

```
<input type="radio" name="tipo_pregunta" value="2" id="tipo_pregunta" onclick="mostrarReferencia();" > Verdadero o falso.  
<div id="desdeotro_2" style="display:none;" >  
<br>
```



```

<p>
  <input type="radio" name="respuesta_2" value="0" class="campo-form_2">
Verdadero
  <br>
  <br>
  <input type="radio" name="respuesta_2" value="1" class="campo-form_2">
Falso
</p>
</div>

```

En las líneas de código superiores de HTML, se puede ver:

- En las dos primeras líneas, como al pulsar el *radioButton* asociado a la opción **Verdadero o Falso**, se asignará el valor dos a la variable `tipo_pregunta` y llamará a la función `mostrarReferencia()`. ¿Qué hace esta función? El usuario profesor puede crear cinco tipos de preguntas diferentes, cada una de ellas con una serie de respuestas. Se quiere que el contenido correspondiente a cada uno de los tipos de respuestas varíe en función del tipo de pregunta que se haya elegido.
- En el resto de las líneas del código, se muestra el contenido que aparece entre las etiquetas HTML `<div></div>` y que tiene el identificador "desdeotro\_2". Este identificador varía para cada uno de los menús que se quiere desplegar según se haya elegido un tipo de pregunta u otra.

La función `mostrarReferencia()` es la siguiente (en *JavaScript*):

```

function mostrarReferencia(){
//Si la opcion con id Conocido_1 (dentro del documento > formulario con n
ame fcontacto > y a la vez dentro del array de Conocido) esta activad
a
if (document.formConsulta.tipo_respuesta[1].checked == true) {
//muestra (cambiando la propiedad display del estilo) el div con id 'desd
eotro'
  document.getElementById('desdeotro_1').style.display='block';
  document.getElementById('desdeotro_2').style.display='none';
  document.getElementById('desdeotro_3').style.display='none';
  document.getElementById('desdeotro_4').style.display='none';
  document.getElementById('desdeotro_5').style.display='none';
//por el contrario, si no esta seleccionada
}
else if (document.formConsulta.tipo_respuesta[2].checked == true) {
//muestra (cambiando la propiedad display del estilo) el div con id 'desd
eotro'
  document.getElementById('desdeotro_1').style.display='none';
  document.getElementById('desdeotro_2').style.display='block';
  document.getElementById('desdeotro_3').style.display='none';
  document.getElementById('desdeotro_4').style.display='none';
}
}

```

```

        document.getElementById('desdeotro_5').style.display='none';
//por el contrario, si no esta seleccionada
    }
else if (document.formConsulta.tipo_respuesta[3].checked == true) {
//muestra (cambiando la propiedad display del estilo) el div con id 'desd
eotro'
    document.getElementById('desdeotro_1').style.display='none';
    document.getElementById('desdeotro_2').style.display='none';
    document.getElementById('desdeotro_3').style.display='block';
    document.getElementById('desdeotro_4').style.display='none';
    document.getElementById('desdeotro_5').style.display='none';
//por el contrario, si no esta seleccionada
}
else if (document.formConsulta.tipo_respuesta[4].checked == true) {
    document.getElementById('desdeotro_1').style.display='none';
    document.getElementById('desdeotro_2').style.display='none';
    document.getElementById('desdeotro_3').style.display='none';
    document.getElementById('desdeotro_4').style.display='block';
    document.getElementById('desdeotro_5').style.display='none';
}
else if (document.formConsulta.tipo_respuesta[5].checked == true) {
    document.getElementById('desdeotro_1').style.display='none';
    document.getElementById('desdeotro_2').style.display='none';
    document.getElementById('desdeotro_3').style.display='none';
    document.getElementById('desdeotro_4').style.display='none';
    document.getElementById('desdeotro_5').style.display='block';
}
else {
//oculta el div con id 'desdeotro'
    document.getElementById('desdeotro_1').style.display='none';
    document.getElementById('desdeotro_2').style.display='none';
    document.getElementById('desdeotro_3').style.display='none';
    document.getElementById('desdeotro_4').style.display='none';
    document.getElementById('desdeotro_5').style.display='none';
}
}
}

```

Cada uno de los elementos del bucle anidado *if* corresponde a que el usuario elija uno de los diferentes tipos de pregunta. Si elige el primer tipo: mostrará una serie de opciones, si elige el segundo, otras, etc. Si el usuario no elige ningún tipo de pregunta, todos los menús desplegables aparecerán ocultos.

Las sentencias de tipo `document.getElementById('desdeotro_x').style.display = 'none'` indican que los menús desplegables estén ocultos. Las sentencias de tipo `document.getElementById('desdeotro_x').style.display = 'block'` indican que los menús desplegables sean visibles y se muestren como elementos de bloque.

## 4.2 La parte del servidor

La parte servidor es el programa que está esperando constantemente solicitudes que provengan de la parte cliente. Las solicitudes son recibidas a través de una conexión http (o https). A su vez, la parte servidor se comunica con la base de datos donde se almacena toda la información sobre la que se realizan las peticiones. Para construir la parte servidor se emplean lenguajes como **PHP** (*Hypertext Preprocessor*) y **SQL** (*Structured Query Language*); y herramientas como **MySQL** (Gosselin, Kokoska, & Easterbrooks, 2010).

### 4.2.1 PHP

Originalmente, **PHP** significaba *Personal Home Page*, significado que ha cambiado en el presente, como se ha indicado anteriormente. Apareció en 1994, gracias a Rasmus Lerdorf. El objetivo de PHP era atraer a visitantes a consultar su currículum, que había “colgado” en internet. Con el paso de los años, PHP fue avanzando, hasta cambiar el significado a sus siglas y consolidar su finalidad: manejar datos antes de convertirse en HTML (Ullman, 2010).

Según la página oficial de PHP, se define como la capacidad de escribir dentro del lenguaje HTML, que compone todas las páginas web. PHP es un **lenguaje de scripts**, lo que significa que no es un lenguaje independiente: está diseñado para que actúe después de que una acción tenga lugar, por ejemplo que un usuario cumplimente un formulario o quiera acceder a una URL (Ullman, 2010).

PHP pertenece a la llamada “tecnología del lado del servidor”: ocurre en el lado del servidor. ¿Qué es un servidor? Una aplicación que se ha configurado para que proporcione al usuario las páginas y el contenido que demanda. Además, PHP es multiplataforma: es compatible para ordenadores o terminales que funcionen con sistemas operativos Windows, Linux, MacOs, etc (Ullman, 2010).

La expansión y aceptación de PHP ha sido tal que en la actualidad se emplea en más de las tres cuartas partes del total de webs en todo el mundo. Como otros lenguajes, PHP ha ido introduciendo mejoras en diferentes versiones. La última es PHP7. Existen competidores de PHP, como Python o Ruby. No obstante, PHP aporta una serie de cualidades que le hacen líder (Thomson & Welling, 2017):

- Buen rendimiento: es muy rápido (Thomson & Welling, 2017).

- Escalabilidad: se puede implementar en un gran número de servidores diferentes (Thomson & Welling, 2017).
- Integración de base de datos: PHP dispone de conexiones propias a sistemas de bases de datos. El uso de ODBC (*Open DataBase Connectivity*, es un estándar de conectividad abierta de bases de datos), permite establecer una conexión con cualquier base de datos que suministre un controlador ODBC (Thomson & Welling, 2017).
- Bibliotecas incorporadas: PHP incorpora funciones para realizar múltiples tareas en las páginas web (Thomson & Welling, 2017).
- Coste: PHP es código abierto, y por tanto gratuito (Thomson & Welling, 2017).
- Fácil: PHP se basa en otros lenguajes, por lo que, si el programador que lo usa está familiarizado con otros lenguajes de programación, no le costará mucho trabajo aprender a utilizar PHP (Thomson & Welling, 2017).
- Compatible con la orientación a objetos: muy útil si se quiere emplear junto a Java o C++ (Thomson & Welling, 2017).
- Portabilidad: PHP está disponible para sistemas que funcionen con diferentes sistemas operativos: Microsoft, Linux, MacOs, etc (Thomson & Welling, 2017).

PHP ha estado presente en casi todos los ficheros necesarios para el desarrollo de la aplicación web de este Trabajo Fin de Grado como medio de manejo de los datos obtenidos de la aplicación web. Con PHP se han podido pasar los valores introducidos por el usuario a la base de datos, y almacenarlos en ella. A continuación, se van a mostrar algunos fragmentos de código en los que se ha empleado PHP:

1. En la edición de un campo de los datos del alumno, en la pantalla **Editar Alumno** explicada en el punto 4.2.6 de esta memoria. En este caso, PHP sirve para mostrar el valor actual almacenado en la base de datos en el campo nombre\_usuario correspondiente al alumno.

```
<p><br><?php echo'Nombre usuario:?'><span>*</span>
<input type="text" name="nombre_usuario" value ="<?php echo $fila['nombre_usuario']?>" class="campo-form" maxlength="30" size="30" required>
</p>
```

2. Para trabajar con los datos que el usuario ha introducido en un formulario. En este caso, los valores que el usuario profesor ha introducido en el formulario de registro de una pregunta son almacenados en variables, con las que más tarde se podrá trabajar.

```
<?php
```

```
$enunciado_pregunta=$_POST['enunciado_pregunta'];
$tipo_respuesta=$_POST['tipo_respuesta'];
$id_usuario=$_SESSION['id_usuario'];
¿>
```

#### 4.2.2 MySQL

MySQL no es una base de datos, sino que es un *software* que permite al programador crear, mantener y gestionar un conjunto de datos que conforman una base de datos. Para gestionar esas bases de datos hace uso del lenguaje SQL. MySQL se puede considerar un Sistema Gestor de Base de Datos (DBMS, *Database Management System*) (Reese, Jay Yarger, King, & Williams, 2002).

Una base de datos permite almacenar, buscar, recuperar o modificar datos. MySQL controla el acceso a la base de datos para asegurar que solo los usuarios con los permisos necesarios pueden acceder a los datos que les corresponden y que más de un usuario pueda acceder a la base de datos y consultar lo que desea (MySQL es un servidor multiusuario). MySQL emplea un lenguaje de programación para llevar a cabo todas estas acciones: SQL (Thomson & Welling, 2017).

#### 4.2.3 SQL

**SQL** (Structured Query Language) es un lenguaje de dominio específico (se utiliza para un uso concreto) que se emplea para gestionar datos que pertenecen a bases de datos relacionales. Este lenguaje presenta dos grandes ventajas frente a antiguas versiones que se empleaban para lo mismo: se puede acceder a muchos elementos de la base de datos con un solo comando y hace posible identificar un elemento de una tabla de la base de datos a partir de cualquiera de sus campos (no solo de su clave primaria) (Microsoft, 2019).

SQL fue desarrollado por Donald D. Chamberlin y Raymond F. Boyce, ambos trabajadores de la empresa IBM, sobre los años setenta. El nombre original que se le dio a este lenguaje fue SEQUEL (*Structured English Query Language*), y su función era manipular los datos almacenados en la base de datos de IBM. No fue hasta 1986, cuando la ANSI (*American National Standards Institute*) y la ISO (*International Organization for Standardization*) aceptaron al lenguaje SQL como estándar (Chamberlin, October-December 2012).

SQL está compuesto por tres principales módulos de lenguaje: DML (*Data Manipulation Language*), DDL (*Data Definition Language*) y DCL (*Data Control Language*) (Rockoff, 2017).

- DML: este módulo permite recuperar, modificar, añadir o eliminar datos de la base de datos.
- DDL: este módulo permite crear y modificar el diseño de las tablas de la base de datos.
- DCL: este módulo se encarga de la seguridad de la base de datos.

Una base de datos relacional (como es la base de datos con la que se trabaja en este Trabajo Fin de Grado) tiene ciertos elementos clave. Uno de ellos es la **clave primaria**. La clave primaria de una tabla es un identificador único que es necesario por dos razones (Rockoff, 2017):

- Permiten identificar de manera única a una fila de una tabla de la base de datos. Un campo de una tabla que se ha definido como clave primaria, asegura que todos los componentes de esa columna tendrán un valor único para cada fila (Rockoff, 2017).
- Permite relacionar fácilmente una tabla con otra (Rockoff, 2017).

Para recuperar datos de la base de datos con SQL, se emplean un conjunto de palabras clave, que conforman una sentencia. Por ejemplo, si se quiere seleccionar todos los elementos de la tabla **alumnos**, la sentencia SQL sería: *SELECT \* FROM alumnos;* Es importante tener en cuenta que las palabras clave no son sensibles al uso de mayúsculas o minúsculas. Además, las sentencias SQL se pueden escribir en diferentes líneas, teniendo el mismo efecto. Esto puede ser útil en aquellas ocasiones donde la sentencia es muy larga, y conviene verla a golpe de vista (Rockoff, 2017).

Las sentencias SQL se pueden hacer más o menos complejas; más o menos específicas. A continuación, se van a mostrar algunas sentencias SQL que se han usado en el desarrollo de la aplicación web de este Trabajo Fin de Grado.

En este primer ejemplo se puede ver la sentencia SQL necesaria para crear una nueva fila en la tabla **alumnos**. **INSERT INTO** indica que se cree una nueva fila. Los campos que aparecen entre paréntesis se corresponden a los atributos de la tabla **alumnos**. Los campos precedidos por la palabra **VALUES** son los valores que se van a almacenar en los diversos campos de la tabla. Cada vez que se registre a un alumno, la variable **\$nombre** tendrá un valor, la variable **\$ciudad** otro, etc.

```
INSERT INTO alumnos (id_profesor_creador, nombre_usuario, password, nombre, primer_apellido, segundo_apellido, direccion, ciudad, email, telefono,
```

```
class)VALUES('$id_profesor_creador','$nombre_usuario','$password','$nombre','$primer_apellido','$segundo_apellido','$direccion','$ciudad','$email','$telefono','$clase');
```

La sentencia mostrada debajo extrae de la tabla **cuestionarios** todas las filas cuya columna **id\_profesor\_creador** tenga el valor de la variable **\$id\_profesor**. El valor de la variable **\$id\_profesor** puede variar dependiendo de quién esté utilizando la aplicación web, por lo que en función de su valor, se seleccionarán unos cuestionarios u otros.

```
SELECT * FROM cuestionarios WHERE id_profesor_creador = $id_profesor;
```

La siguiente sentencia utiliza la palabra clave **UPDATE** y sirve para actualizar el valor de una fila de la tabla **profesores**. La palabra clave **SET** indica que el valor de los atributos de la tabla **profesores** serán reemplazados por los valores almacenados en las variables a las que se les está igualando. Por ejemplo, el atributo **primer\_apellido**, pasará a contener el valor que almacena la variable **\$primer\_apellido**. La palabra clave **WHERE** indica que las filas cuya columna **id\_profesor** valga lo mismo que **\$id\_profesor**, serán modificadas. Como **id\_profesor** es una clave primaria, solo se cambiará una fila, que identifica a un único usuario.

```
UPDATE profesores SET nombre=".$nombre.",primer_apellido=".$primer_apellido.",segundo_apellido=".$segundo_apellido.",direccion=".$direccion.",ciudad=".$ciudad.",telefono=".$telefono.",email=".$email." WHERE id_profesor = ".$id_profesor."
```

El último ejemplo mostrado elimina una o más filas de una tabla de la base de datos. En este caso, se eliminará la pregunta de la tabla **preguntas** cuyo valor de la columna **id\_pregunta** sea igual al valor de la variable **\$id\_pregunta**. De nuevo en este ejemplo, solo se eliminará una fila, dado que **id\_pregunta** es una clave primaria.

```
DELETE FROM preguntas where id_pregunta = $id_pregunta
```

### 4.3. Otras herramientas

En los apartados anteriores se han listado los diferentes lenguajes de programación empleados para desarrollar la aplicación web de este Trabajo Fin de Grado, pero no solo ha sido necesario el uso de lenguajes de programación: también se ha tenido que recurrir a diversas herramientas y programas que se van a describir a continuación.

### 4.3.1 Visual Studio Code

Para llevar a cabo la aplicación web se ha empleado *Visual Studio Code*, un editor de código fuente (véase Ilustración 21) que se puede emplear para trabajar con múltiples lenguajes de programación. Desde *Visual Studio Code* se pueden abrir uno o más ficheros y trabajar sobre ellos, y además permite al usuario trabajar de forma simultánea con más de un lenguaje de programación. (Code, 2019).

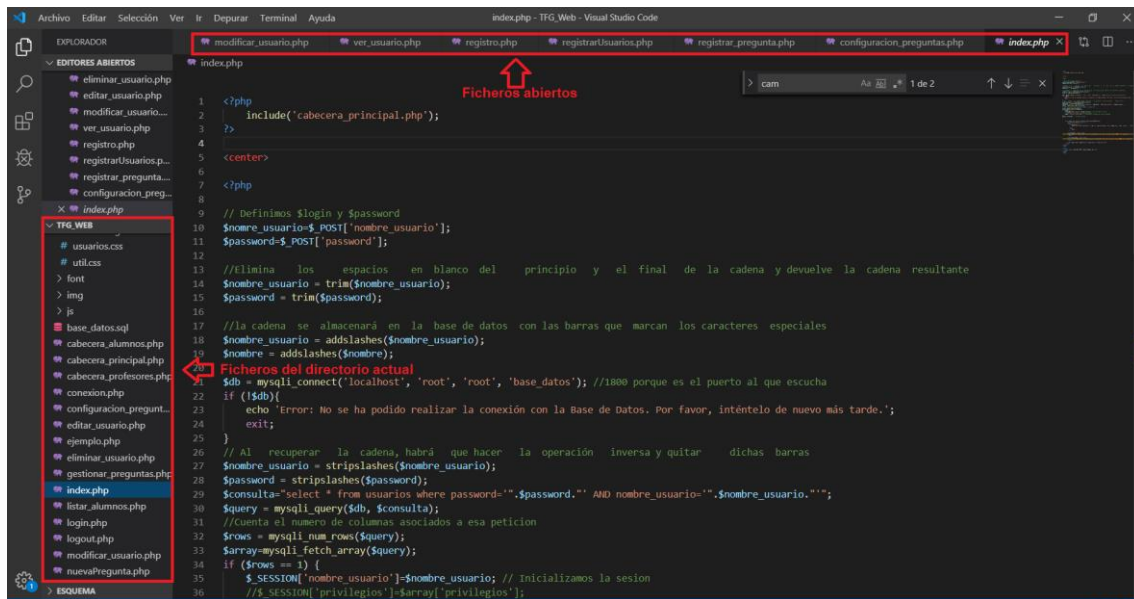


Ilustración 21: Ejemplo de pantalla de Visual Studio Code

### 4.3.2 MAMP

MAMP (*macOS, Apache, MySQL, PHP, Perl Python*) es un *software* gratuito y de código abierto (véase Ilustración 22) que permite ejecutar sitios web dinámicos (como es la aplicación web que se ha diseñado en este Trabajo de Fin de Grado), compatible con los sistemas operativos Windows y macOS (PCQuest, 2019).

El nombre del *software* engloba a los componentes del sistema: el sistema operativo **macOS**, el servidor web **Apache**, el Sistema Gestor de Base de Datos **MySQL**, y los diferentes lenguajes de programación empleados para el desarrollo web: **PHP, Perl y Python** (Codex, 2019).



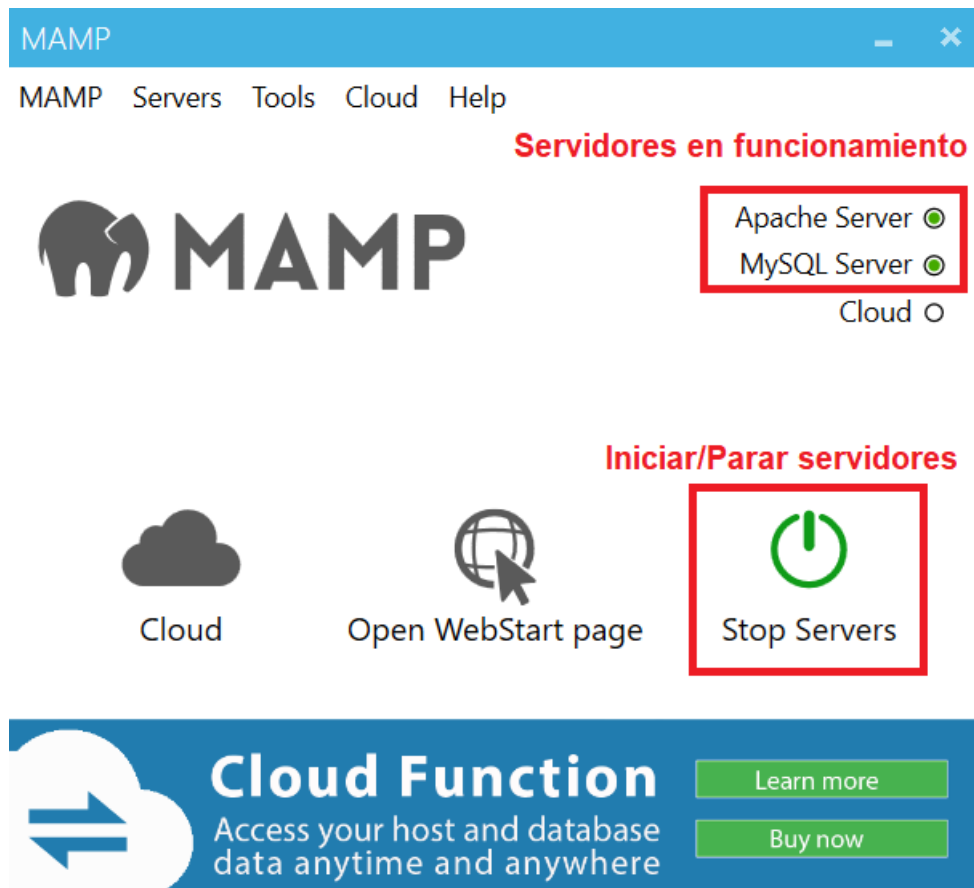


Ilustración 22: Interfaz de MAMP

#### 4.3.3 PHPMysqlAdmin

**PHPMysqlAdmin** (véase Ilustración 23) es una herramienta de administración de bases de datos de código abierto. Es una de las aplicaciones más empleadas en la administración de MySQL. Proporciona al usuario que trabaja con ella: una interfaz web, un gestor de bases de datos (tanto de tipo MySQL como MariaDB), la opción de exportar las bases de datos a distintos formatos, administrar múltiples servidores, etc. (phpMyAdmin, 2019).

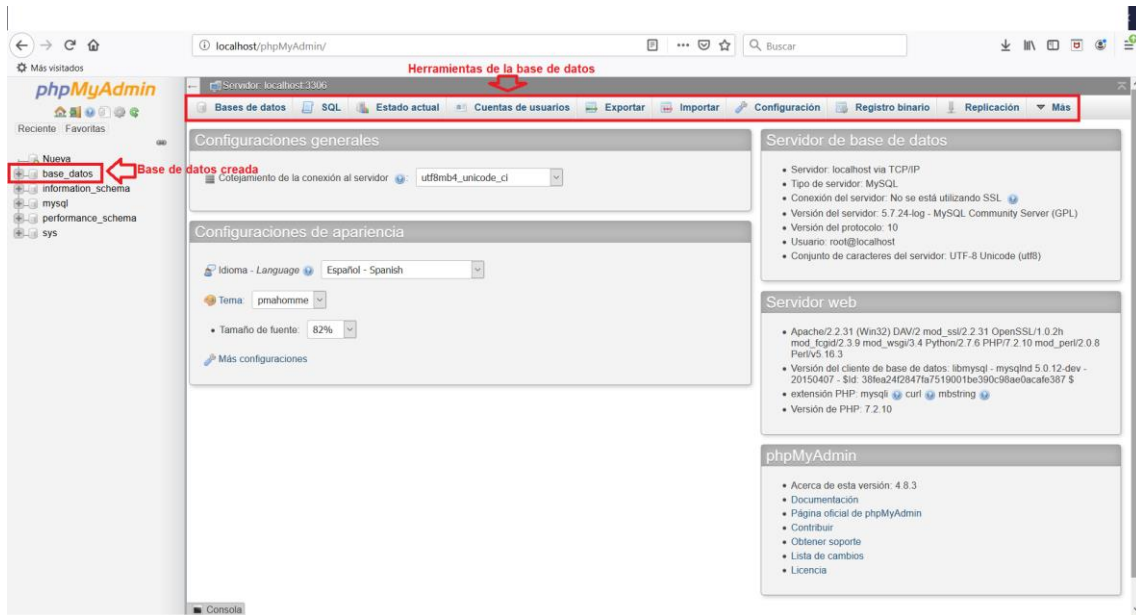


Ilustración 23: Interfaz gráfica de phpMyAdmin

## 5. MANUAL DE USUARIO

Ya se han descrito los requisitos funcionales y no funcionales de la aplicación web, los casos de uso y las tablas que forman la base de datos. Ahora se va a explicar, a través de capturas de pantalla reales, las diferentes funcionalidades de la aplicación web, con el fin de que el usuario aprenda a utilizar la misma y se familiarice con ella.

### 5.1 Página de inicio de sesión

Lo primero con lo que se va a encontrar el usuario de la Plataforma de Cuestionarios es una pantalla que le permitirá iniciar sesión en la plataforma. Para ello, el usuario deberá tener una cuenta de usuario. Introduciendo su **nombre de usuario** y su **contraseña** en los campos correspondientes, el usuario accederá a su cuenta personal (véase Ilustración 24).



*Ilustración 24: Pantalla de Inicio de Sesión*

Como se ha comentado anteriormente, existen dos roles diferentes en la plataforma: alumno y profesor. La siguiente pantalla (página principal del usuario) variará en función del rol que tenga.

### 5.2 Menú Principal de Profesores

Si el usuario que ha iniciado sesión es un profesor, podrá realizar las siguientes tareas: gestionar alumnos, preguntas y cuestionarios; asignar cuestionarios a los alumnos; ver los cuestionarios que tiene pendientes de corregir y ver los cuestionarios que ya ha

corregido. Todas estas tareas las puede seleccionar desde el menú principal de profesores. También podrá llevar a cabo otras tareas, que se describirán más adelante (véase Ilustración 30).

En la página principal del profesor, se pueden ver una serie de iconos que permiten un acceso rápido a las acciones que se desarrollan más a menudo:



- Registrar Alumno: permite al usuario “profesor” registrar a un nuevo alumno (véase Ilustración 25).

*Ilustración 25: Registrar Alumno (Flaticon, 2019)*



- Alumnos: permite al usuario “profesor” ver la lista de alumnos registrados en la plataforma (véase Ilustración 26).

*Ilustración 26: Ver Alumnos (Flaticon, 2019)*



- Preguntas: permite al usuario “profesor” ver las preguntas registradas en el sistema (véase Ilustración 27)

*Ilustración 27: Ver Preguntas (Flaticon, 2019)*



- Cuestionarios: permite al usuario “profesor” ver los cuestionarios registrados en el sistema (véase Ilustración 28).

Ilustración 28: Ver Cuestionarios (Flaticon, 2019)



- Corregir: permite al usuario “profesor” ver los cuestionarios que le han entregado sus alumnos y tiene pendientes de corregir (véase Ilustración 29).

Ilustración 29: Corregir Cuestionarios (Flaticon, 2019)



Opciones disponibles:



Ilustración 30: Menú Principal del Profesor

### 5.2.1 Editar perfil propio

El profesor podrá cambiar cualquier dato referente a su perfil. Para ello, tendrá que pulsar la palabra *Perfil* que aparece en la parte superior derecha de la pantalla (en la

parte de la cabecera de la aplicación web) (véase Ilustración 31). Esta opción de editar el perfil personal, también está disponible en el menú principal de alumnos.



Ilustración 31: Cabecera del profesor

Pulsando en *Perfil*, aparecerá la pantalla mostrada en la Ilustración 32. Todos los campos del usuario pueden ser cambiados a excepción del nombre de usuario, la contraseña, el rol que desempeña (profesor o alumno) y la clase a la que pertenece (en el caso de los usuarios de tipo “alumno”).



**Datos del usuario:**

Nombre\*

Primer apellido\*

Segundo apellido\*

Dirección\*

Ciudad\*

Email\*

Teléfono\*

Ilustración 32: Editar Perfil Usuario

## 5.2.2 Cerrar Sesión

Si el usuario quiere cerrar su sesión, debe pulsar el texto *Cerrar Sesión* que se encuentra al lado de *Perfil*. Ambas opciones están situadas en una zona de la pantalla donde en

todo momento se podrá ver el **nombre de usuario** que actualmente tiene la sesión iniciada (véase Ilustración 33).



Ilustración 33: Cerrar Sesión

### 5.2.3 Barra de herramientas

Separando la cabecera del cuerpo de la aplicación, hay una barra desde donde se puede acceder a cualquiera de las opciones que proporciona la aplicación (véase Ilustración 34). Algunas de estas opciones de la barra de herramientas tienen un menú desplegable. Este es el caso de las opciones **Gestionar Preguntas**, **Gestionar Cuestionarios** y **Cuestionar Alumnos**.

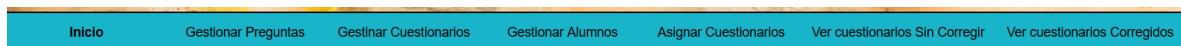


Ilustración 34: Barra de herramientas

Los menús desplegables se pueden ver en la Ilustración 35.



Ilustración 35: Menús desplegables de la barra de herramientas

### 5.2.4 Gestionar Preguntas

La pestaña *Preguntas* ofrece al usuario “profesor” dos opciones: **crear pregunta o ver preguntas**. La primera de las opciones, **Crear Pregunta**, lleva al usuario a la pantalla mostrada en la Ilustración 36. En ella hay un espacio para que se introduzca el enunciado de la pregunta que se va a crear y una serie de opciones para que el profesor seleccione el tipo de respuesta que quiere asociar a la pregunta. En la Ilustración 36 se muestra un

ejemplo de pregunta con respuesta abierta (primera opción seleccionada). Pero se pueden elegir otras opciones:

PLATAFORMA DE CUESTIONARIOS

Inicio Preguntas Cuestionarios Alumnos Asignar Corregir Ver Corregidos

Bienvenid@ : Patricia Perfil Cerrar sesión

Por favor, rellene los siguientes datos:

Enunciado de la pregunta: \*

Cita cinco países de Europa que no estén en la unión europea.

Tipo de pregunta:

- Respuesta abierta.
- Si o no.
- Verdadero o falso.
- Varias opciones.
- Gradual.

CREAR CANCELAR

Ilustración 36: Crear Pregunta

En la Ilustración 37 se muestra el formulario a rellenar cuando la pregunta tiene asociada una respuesta de tipo verdadero/falso (igual a la de respuesta sí/no). Al lado de las palabras Verdadero y Falso aparece un botón y seleccionarle implica señalar esa respuesta como la correcta. Más adelante se explicará la utilidad de esta acción.



**Por favor, rellene los siguientes datos:**

Enunciado de la pregunta: \*

El país más pequeño de mundo es Andorra.

**Tipo de pregunta:**

- Respuesta abierta.
- Sí o no.
- Verdadero o falso.
  - Verdadero
  - Falso
- Varias opciones.
- Gradual.

**CREAR** **CANCELAR**

*Ilustración 37: Pregunta con respuesta Verdadero/Falso*

En la Ilustración 38 se muestra el formulario a rellenar cuando la pregunta tiene varias respuestas posibles. Como se puede ver, el usuario puede elegir el número de respuestas que quiere que tenga su pregunta (entre 2 y 6). También puede introducir el texto que quiere que tenga cada una de las respuestas.

**Por favor, rellene los siguientes datos:**

Enunciado de la pregunta: \*

¿Cuál es la capital de Polonia?

**Tipo de pregunta:**

- Respuesta abierta.
- Sí o no.
- Verdadero o falso.
- Varias opciones.
  - 2 Respuestas.
    - Cracovia
    - Varsovia
  - 3 Respuestas.
  - 4 Respuestas.
  - 5 Respuestas.
  - 6 Respuestas.
- Gradual.

**CREAR** **CANCELAR**

*Ilustración 38: Pregunta con varias respuestas*

Por último, se puede ver una captura del formulario para preguntas con respuestas graduales (véase Ilustración 39). Es similar al caso anterior, pero no se puede marcar ninguna respuesta como correcta (ya que está pensado para encuestas de opinión a los alumnos) y solo permite elegir un rango de tres, cinco o diez respuestas. El texto asociado a cada una de las respuestas también puede ser introducido por el usuario a su libre elección.

**Por favor, rellene los siguientes datos:**

Enunciado de la pregunta: \*

¿Cómo evaluarías la apariencia de la Aplicación?

**Tipo de pregunta:**

- Respuesta abierta.
- Sí o no.
- Verdadero o falso.
- Varias opciones.
- Gradual.
  - De 1 a 3.
    - 1. Buena
    - 2. Regular
    - 3. Mala
  - De 1 a 5.
  - De 1 a 10.

**CREAR** **CANCELAR**

*Ilustración 39: Crear Pregunta gradual con 3 opciones*

Otra de las posibilidades que se tiene con las preguntas son las de ver, editar y eliminarlas. Para acceder a estas opciones, hay que ir a Preguntas > Ver Preguntas. La pantalla que aparece a continuación muestra todas las preguntas que hay registradas en el sistema (véase Ilustración 40). Se puede apreciar dos casos diferentes: algunas preguntas permiten al usuario “profesor” verlas, editarlas y eliminarlas; mientras que

otras solo pueden ser vistas. Esto es debido a que un profesor solo podrá editar o eliminar aquellas preguntas que él mismo haya creado. En la parte superior de la pantalla se muestra un mensaje advirtiendo que si se elimina una pregunta, esta desaparecerá de todos los cuestionarios que la contengan.

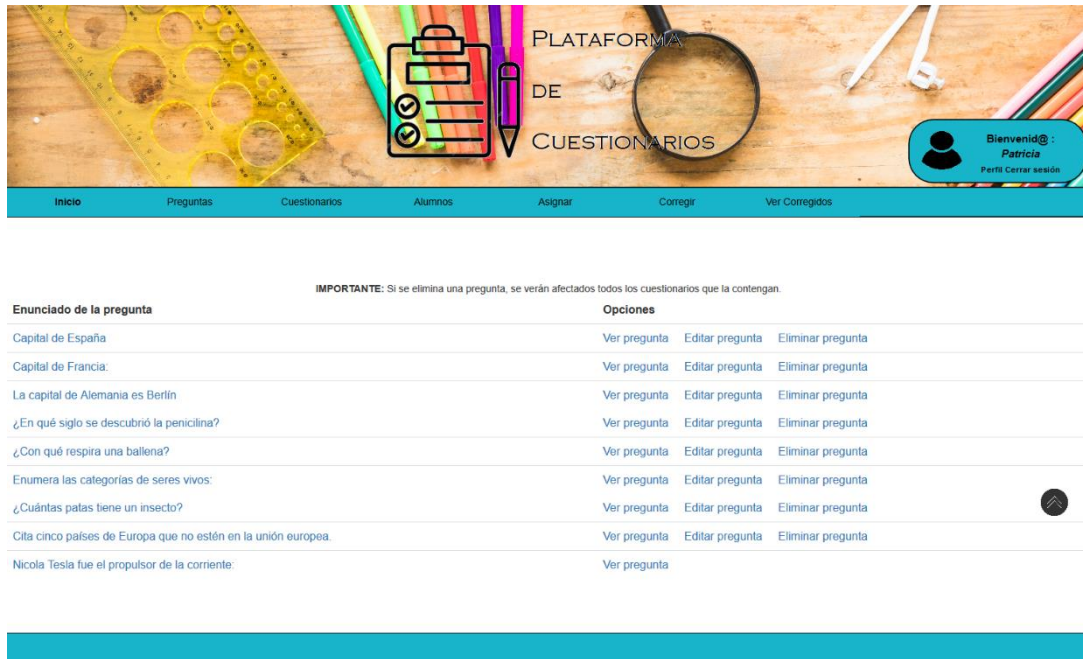


Ilustración 40: Ver las preguntas que hay registradas en el sistema

Si se selecciona la opción **Ver Pregunta**, el usuario puede ver el enunciado de la pregunta y el tipo de respuesta que tiene (para las preguntas de respuesta abierta, sí/no o verdadero/falso). En el caso de la Ilustración 41, la pregunta tiene una respuesta con varias opciones.



Ilustración 41: Ver Pregunta tipo 0,1,2

Si la pregunta que se quiere ver es una pregunta con varias opciones, se muestra también las posibles respuestas que tiene asociadas y se indica de color verde, seguida del símbolo "<<", cuál de todas las respuestas es la correcta (véase Ilustración 42).



Ilustración 42: Ver pregunta tipo 3 y 4

Si se selecciona la opción **Editar Pregunta**, aparece en pantalla el enunciado original de la pregunta y todas las posibles opciones de respuesta. Así, el profesor podrá volver a definir el tipo de respuesta que quiere (véase Ilustración 43).

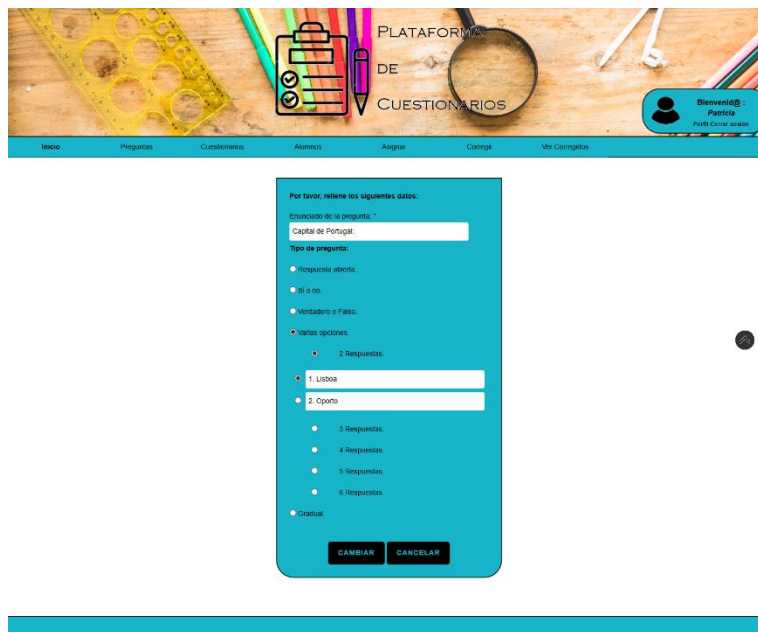


Ilustración 43: Editar Pregunta

Por último, el profesor podrá eliminar cualquier pregunta que haya creado. Al **eliminar una pregunta**, se eliminan todas las respuestas asociadas a ésta.

### 5.2.5 Gestionar Cuestionarios

Al igual que con las preguntas, el profesor puede crear y ver cuestionarios. Para crear un cuestionario hay que ir a Cuestionarios > Crear Cuestionario (véase Ilustración 44). Un

cuestionario necesita un título y la selección de las preguntas que se quieren incluir en él. El usuario podrá introducir en el cuestionario tanto preguntas creadas por él como por otro profesor.

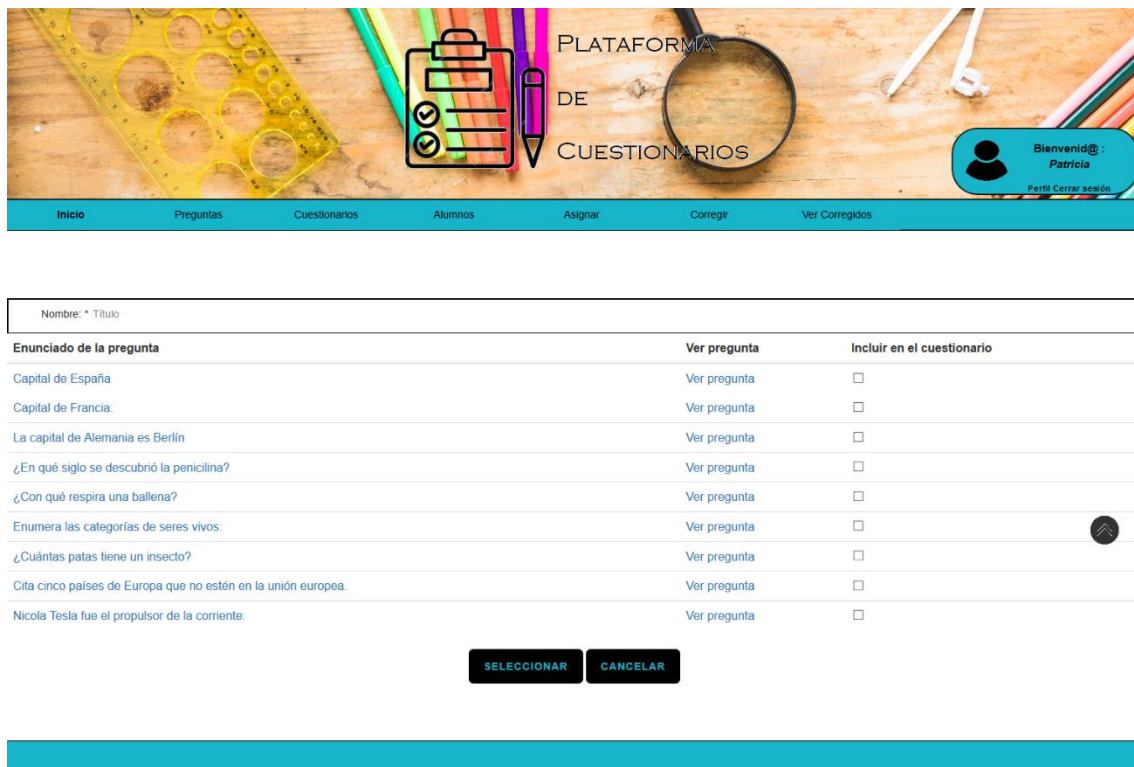


Ilustración 44: Crear Cuestionario

Si se quiere ver, editar o eliminar cuestionarios, hay que seleccionar Cuestionarios > Ver Cuestionario. Y aparece una pantalla similar a la de las preguntas (véase Ilustración 45). De nuevo, solo los cuestionarios creados por el usuario pueden ser modificados o eliminados por él.

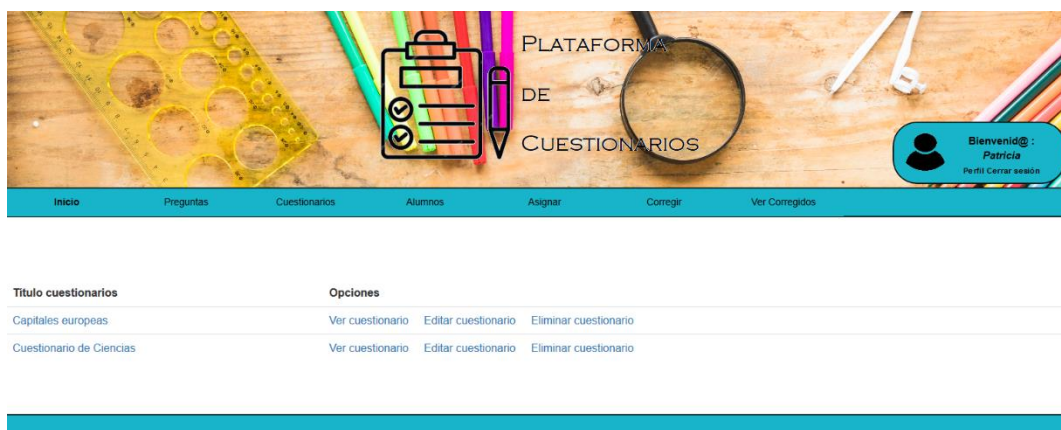


Ilustración 45: Ver los cuestionarios que existen en el sistema

Al seleccionar la opción **Ver Cuestionario**, una pantalla muestra la información sobre él: su título y el enunciado de las preguntas que lo componen. A demás, en las preguntas

de tipo 1, 2 y 3; se marca de color verde, seguida del símbolo “<<” cuál es la respuesta correcta (véase Ilustración 46).

Titulo cuestionarios	Preguntas
Capitales de la Unión Europea	1. Capital de España
	Madrid <<
	Barcelona
	2. Capital de Francia:
	1. París <<
	2. Burdeos
	3. Lyon
	3. La capital de Alemania es Berlín

Ilustración 46: Ver Cuestionario

Para editar un cuestionario que ya está creado, se selecciona **Editar Cuestionario**. Al seleccionar esa opción, aparecerá de nuevo la lista con todas las preguntas registradas en el sistema, con las preguntas que actualmente pertenecen al cuestionario marcadas. El profesor podrá introducir las nuevas preguntas que quiera, o quitar alguna de las preguntas que haya (véase Ilustración 47).



Nombre: \* Cuestionario de Ciencias

Título del Cuestionario	Incluir en el cuestionario
Capital de España	Ver pregunta <input type="checkbox"/>
Capital de Francia:	Ver pregunta <input type="checkbox"/>
La capital de Alemania es Berlín	Ver pregunta <input type="checkbox"/>
¿En qué siglo se descubrió la penicilina?	Ver pregunta <input checked="" type="checkbox"/>
¿Con qué respira una ballena?	Ver pregunta <input type="checkbox"/>
Enumera las categorías de seres vivos:	Ver pregunta <input type="checkbox"/>
¿Cuántas patas tiene un insecto?	Ver pregunta <input checked="" type="checkbox"/>
Cita cinco países de Europa que no estén en la unión europea.	Ver pregunta <input type="checkbox"/>
Nicola Tesla fue el propulsor de la corriente:	Ver pregunta <input type="checkbox"/>

Ilustración 47: Editar Cuestionario

Para eliminar un cuestionario, se selecciona la opción **Eliminar Cuestionario**. Si se elimina un cuestionario, todos los cuestionarios pendientes de realizar por los alumnos que correspondan al cuestionario en cuestión, desaparecerán.

### 5.2.6 Gestionar Alumnos

La pestaña **Alumnos** permite al usuario “profesor” registrar a nuevos usuarios: alumnos. Los alumnos registrados, podrán ser vistos por cualquier profesor del sistema y editados y eliminados por el profesor que los creó.

Existe a mayores una función que permite ver los cuestionarios pendientes de un alumno. Esta función se describirá más adelante.

Para registrar a un alumno se va a Alumnos > Crear Alumno. Para registrar a un usuario es necesario introducir de manera obligatoria los campos: **nombre usuario, primer apellido, segundo apellido, dirección, ciudad, email** (que debe cumplir el formato nombre@correo.dominio), **número de teléfono** (nueve cifras como máximo), **clase y contraseña**. En la Ilustración 48 se puede ver el formulario correspondiente al registro de un alumno. No se podrá registrar un alumno que tenga el mismo nombre de usuario que uno que ya esté registrado en el sistema, ya que es la clave de acceso a la plataforma.





Por favor, rellene los siguientes datos:

Nombre Usuario: \*

Nombre: \*

Primer Apellido: \*

Segundo Apellido: \*

Dirección: \*

Ciudad: \*

Email: \*

Teléfono: \*

Clase:

Password: \*

Ilustración 48: Registrar usuario

Para ver a todos los alumnos registrados en el sistema, se va a Alumnos > Ver Alumno (véase Ilustración 49). Aparecen las mismas opciones que en las preguntas y cuestionarios (ver, editar y eliminar alumno), pero en esta ocasión se añade una nueva: **Ver Cuestionarios Pendientes**. Esta opción va a permitir al profesor ver los cuestionarios que el alumno tiene asignados pero que todavía no ha respondido.

Un profesor solo puede ver los cuestionarios que el alumno tiene pendientes y que él mismo le ha asignado (independientemente de que sea el usuario creador del alumno o no) (véase Ilustración 50). El profesor tiene la posibilidad de “desasignar” alguno de los cuestionarios que ha asignado a los alumnos. Esto puede ocurrir porque el profesor hubiera cometido un error al asignarlo, porque cambie de opinión sobre el cuestionario que ha asignado, etc.



Ilustración 49: Ver Alumnos

Sin embargo, el profesor creador del usuario cuenta con un privilegio sobre los alumnos que él mismo ha creado: puede **eliminar todos los cuestionarios** que ese alumno tenga asignados. Una posible situación en la que esta opción podría ser útil es la siguiente: en caso de que finalice el curso, y el alumno pase a una nueva clase, con nuevos profesores y nuevas asignaturas, no tiene sentido que se almacenen los cuestionarios del curso anterior, en caso de que le hubiera quedado alguno sin realizar. Por ello es interesante la opción de poder “poner a cero” la lista de cuestionarios de un alumno. Para ello, hay que seleccionar el *checkbox* **Eliminar Todo** (véase Ilustración 50).

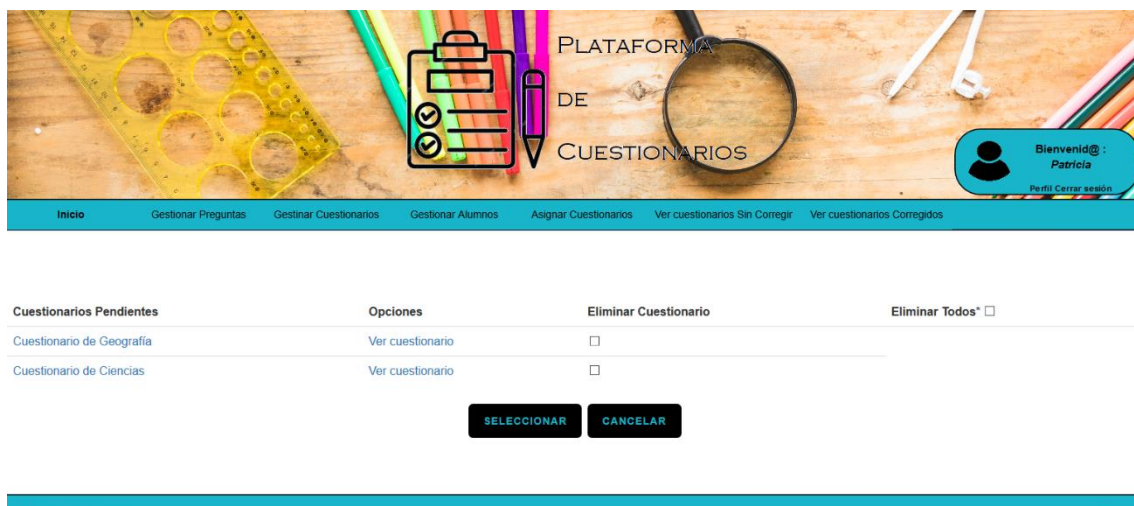


Ilustración 50: Ver Cuestionarios Pendientes de un alumno

Otra de las opciones que tiene el profesor es la de editar un alumno. Una vez seleccione al alumno que quiere editar, aparecerá el formulario del alumno completo con los datos actuales del mismo. El profesor podrá cambiar los campos que se muestran (todos menos la contraseña) y efectuar el cambio pulsando **Cambiar**. En caso de no querer

realizar ningún cambio y querer restablecer los valores iniciales, el profesor podrá pulsar el botón **Borrar** (véase Ilustración 51).

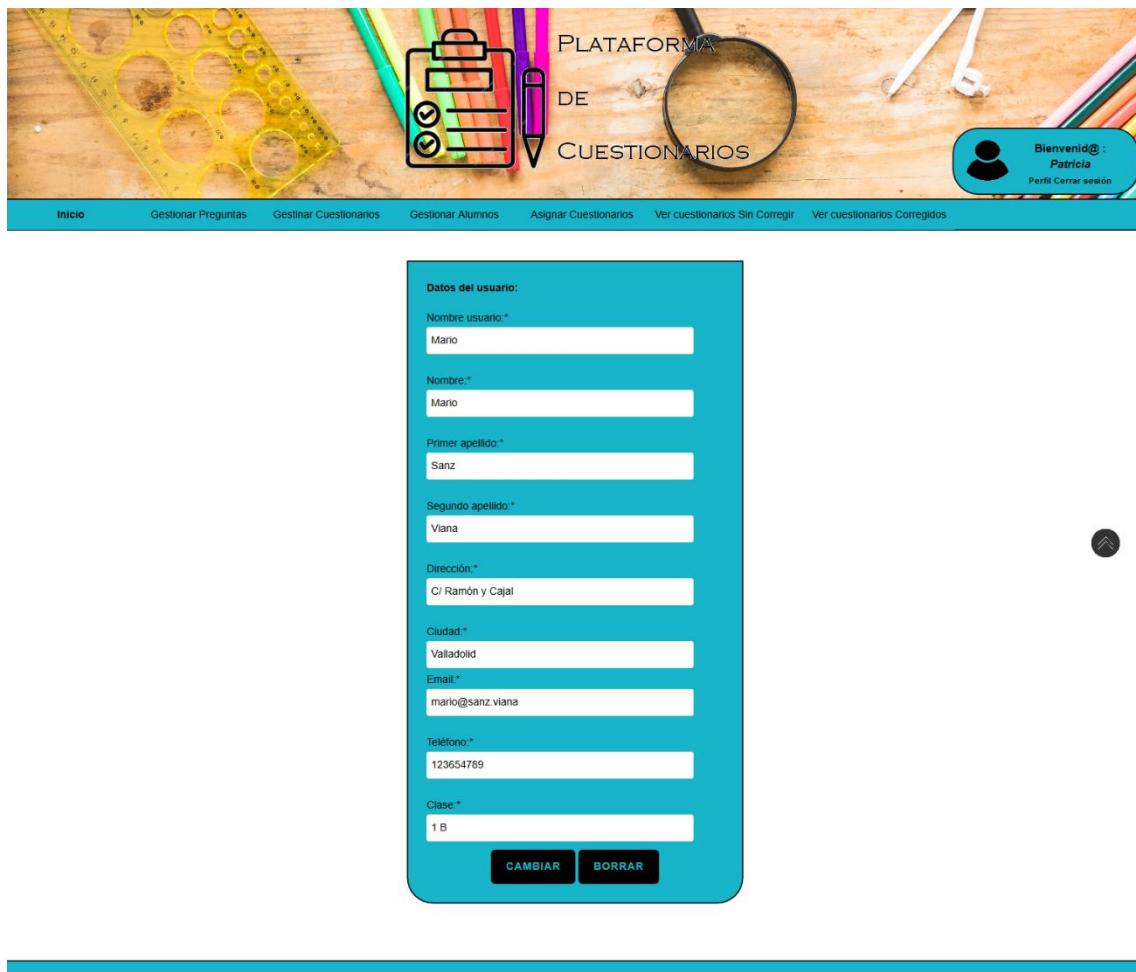


Ilustración 51: Editar Alumno

El profesor podrá ver la información de un alumno: su nombre y apellidos, su dirección, correo, teléfono y clase. Esto es posible con la opción **Ver Alumno** (véase Ilustración 52).



Ilustración 52: Ver Alumno

Por último, el profesor podrá eliminar a aquellos alumnos que él mismo haya registrado con la opción **Eliminar Alumno**.

### 5.2.7 Asignar Cuestionarios

La finalidad de la Plataforma de Cuestionarios es que los alumnos puedan responder a los cuestionarios o encuestas que los profesores les asignen. Esto es posible a través de la opción **Asignar Cuestionarios**. Cuando el profesor selecciona la opción, aparece una lista con todos los cuestionarios que él mismo ha creado. Un profesor solo puede asignar cuestionarios que hayan sido creados por él (véase Ilustración 53). Cuando haya elegido los cuestionarios que quiere asignar (con los *checkbox*), pulsará **Seleccionar**.



Ilustración 53: Lista de cuestionarios que el profesor puede asignar a los alumnos

La siguiente pantalla mostrará al profesor los alumnos que hay registrados en el sistema. Podrá elegir a todos los que quiera. Los alumnos que seleccione pasarán a tener en su lista de cuestionarios pendientes los cuestionarios que haya elegido (véase Ilustración 54).

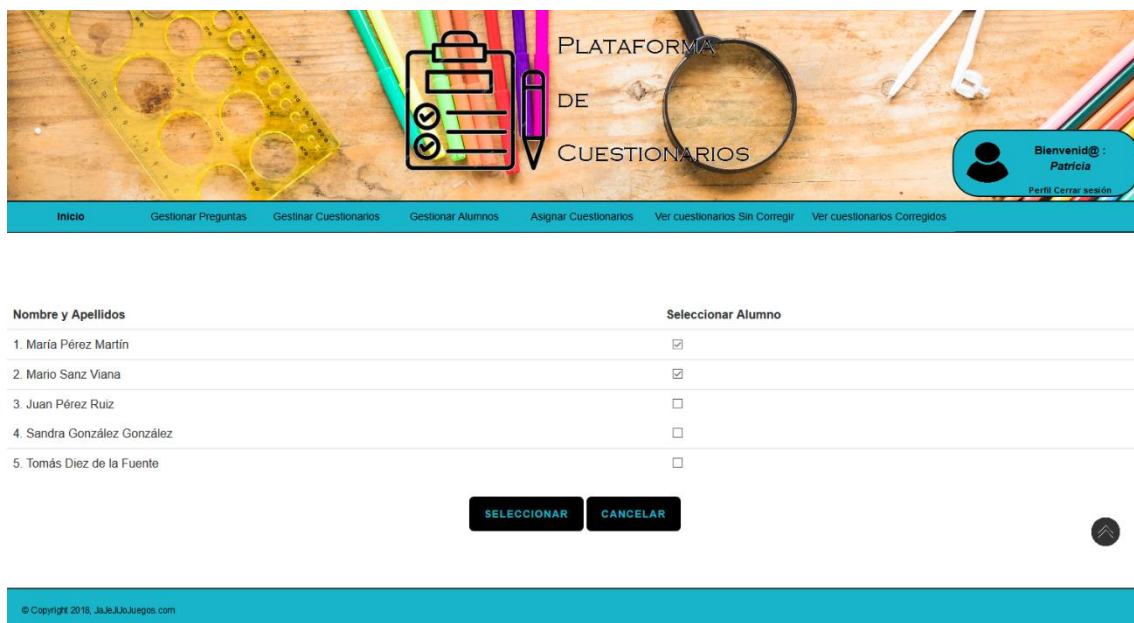


Ilustración 54: Lista de alumnos a los que el profesor puede asignar los cuestionarios que haya elegido

## 5.2.8 Ver Cuestionarios Sin Corregir

La opción **Ver Cuestionarios Sin Corregir** muestra al profesor una lista con los cuestionarios que él ha asignado a los alumnos y ya han sido contestados. Desde esa pantalla (véase Ilustración 55), el profesor podrá elegir el cuestionario que quiere ver en detalle.

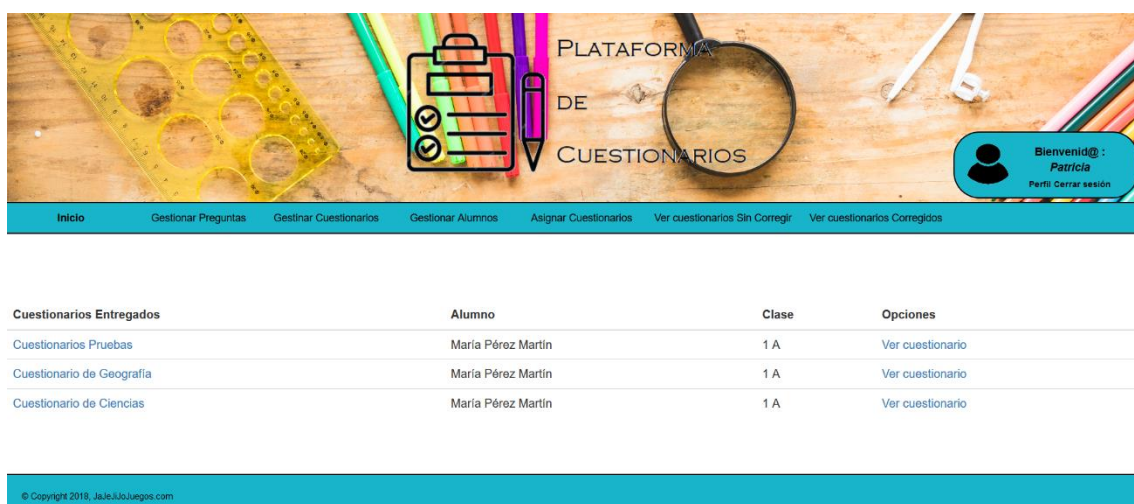


Ilustración 55: Ver Cuestionarios sin Corregir

Una vez se pulse **Ver Cuestionario**, aparecerá el cuestionario con el enunciado de las preguntas y las respuestas dadas por el alumno. También aparece la corrección provisional, que ha sido calculada a partir de los valores introducidos por el profesor al crear las preguntas e indicar cuál de todas las respuestas era la correcta. Aun así, la

corrección aparece como provisional porque hay preguntas que solo pueden ser corregidas por el profesor, como pueden ser las preguntas de respuesta abierta. (Ilustración 56).



Ilustración 56: Corregir cuestionario de un alumno

Para formalizar la calificación, el profesor deberá introducir la calificación final que quiere dar al cuestionario (en forma de número o palabras: Apto, Notable, etc.) y seleccionar la opción **“Terminar de Evaluar”**.

#### 5.2.9 Ver Cuestionarios Corregidos

La última opción a la que puede acceder el usuario profesor es a la de ver los cuestionarios que ya ha evaluado. Para ello, tiene que seleccionar la opción **“Ver Cuestionarios Corregidos”** del menú principal de profesores (Ilustración 57).



Ilustración 57: Ver Cuestionarios Corregidos

### 5.3 Menú Principal Alumnos

Si el usuario que inicia sesión tiene el rol de alumno, podrá realizar dos acciones diferentes: responder a los cuestionarios pendientes que tenga y ver la calificación obtenida en los cuestionarios que ya haya respondido y sus profesores hayan corregido.

La pantalla que verá será la mostrada a continuación (ver Ilustración 58):

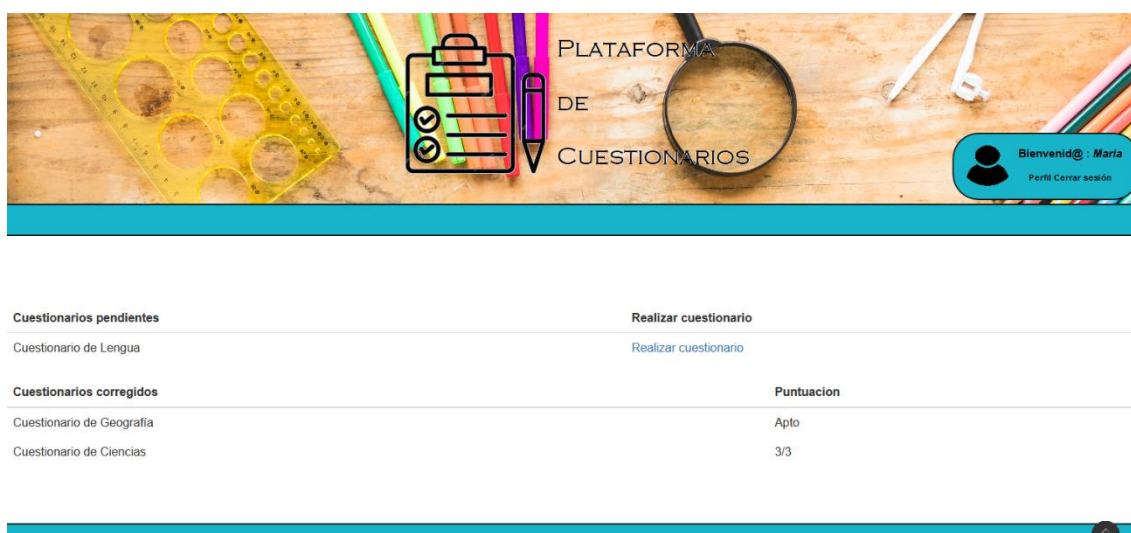


Ilustración 58: Página Principal del Alumno

Como los profesores, los alumnos también pueden editar su perfil desde **Editar Perfil**. No obstante, no podrán cambiar todos los campos que el profesor les ha asignado: el nombre de usuario, la clase a la que pertenecen o la contraseña no podrán ser modificados.

Como se puede ver en su página principal, tienen dos tablas diferentes, una de ellas llamada **Cuestionarios Pendientes** que les ofrece la opción de realizar el cuestionario, y otra llamada **Cuestionarios Corregidos**, que les permite ver las calificaciones obtenidas hasta el momento.

### 5.3.1 Realizar Cuestionario

Si el alumno selecciona la opción **Realizar Cuestionario**, aparecerá una pantalla que le muestre las preguntas con sus correspondientes espacios para las respuestas (véase Ilustración 59). El alumno podrá completar los campos, y cuando considere que lo ha acabado, seleccionar **Enviar**. En ese punto, se le redirigirá a la página principal y el cuestionario que acaba de completar desaparecerá de la lista de cuestionarios pendientes.

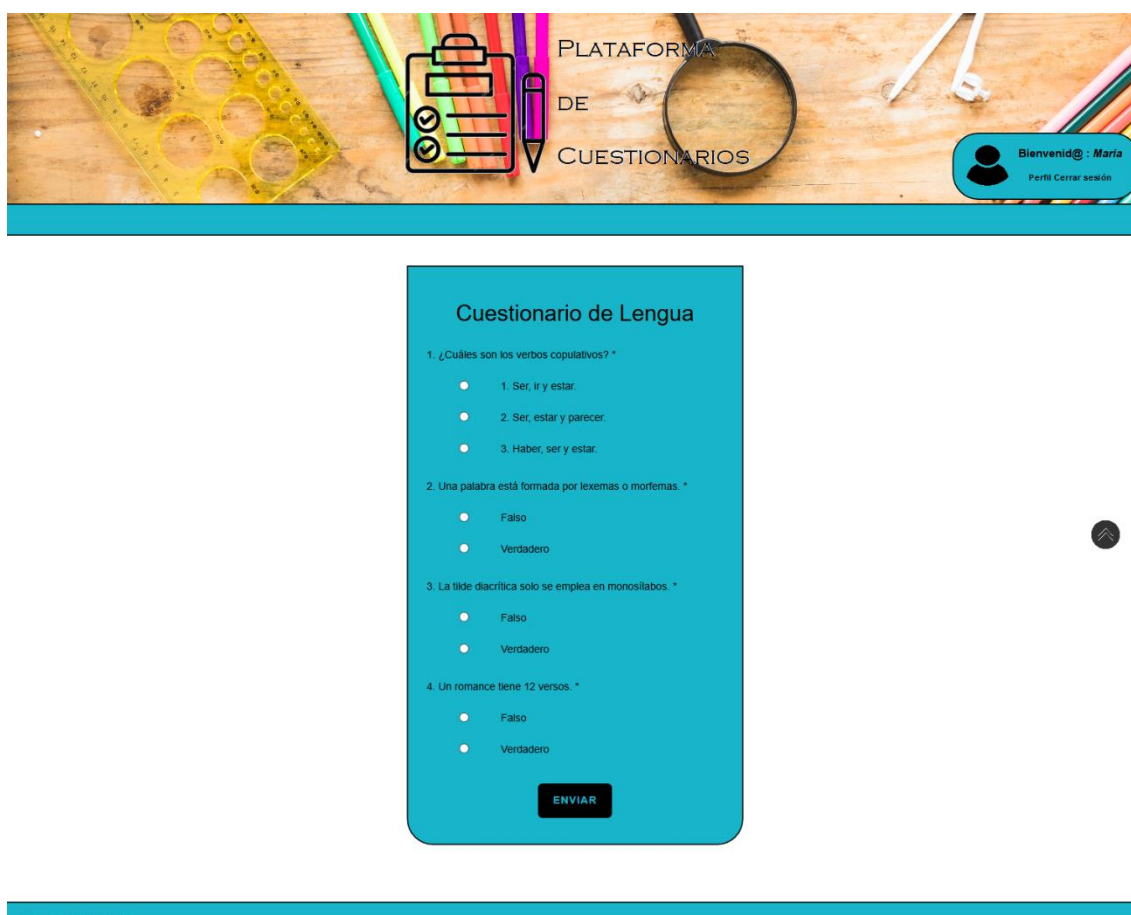


Ilustración 59: Realizar Cuestionario



Si el usuario que ha iniciado sesión es el administrador podrá realizar la tarea de gestionar profesores. Su página principal se puede ver en la Ilustración 60.

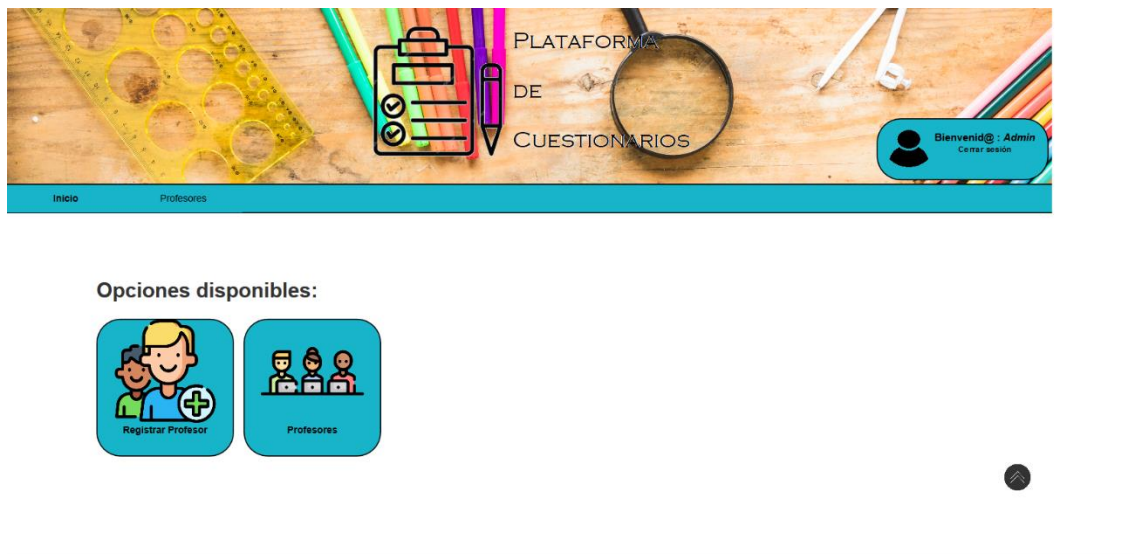


Ilustración 60: Menú Principal del Administrador

#### 6.4.1 Gestionar Profesores

La tarea del Administrador es la de gestionar los profesores de la plataforma. Si se elige la opción **Registrar Profesores**, aparecerá una pantalla similar a la de registrar alumnos (véase Ilustración 61). Es necesario introducir los datos: nombre usuario, nombre, primer apellido, segundo apellido, dirección, ciudad, email, teléfono y contraseña. Si existe en el sistema algún otro usuario con el mismo nombre de usuario, aparecerá un mensaje de error, indicando que no es posible registrar al nuevo profesor.



Por favor, rellene los siguientes datos:

Nombre Usuario: \*

Nombre: \*

Primer Apellido: \*

Segundo Apellido: \*

Dirección: \*

Ciudad: \*

Email: \*

Teléfono: \*

Password: \*

Ilustración 61: Registrar Profesor

Si se elige la opción **Profesores**, aparecerá una tabla mostrando todos los profesores que hay registrados (véase Ilustración 62).



Nombre Profesor	Dirección	Teléfono	Correo	Opciones
Patricia Briongos Pérez	C/ Jaén Burgos	789654123	patriciabriongos22@gmail.com	<a href="#">Editar profesor</a> <a href="#">Ver profesor</a> <a href="#">Eliminar profesor</a>
pilar pilar pilar	pilar pilar	741258	pilar@pilar.pilar	<a href="#">Editar profesor</a> <a href="#">Ver profesor</a> <a href="#">Eliminar profesor</a>
Pedro Gómez Martínez	C/ Junco Valladolid	741258	pedro@gomez.martinez	<a href="#">Editar profesor</a> <a href="#">Ver profesor</a> <a href="#">Eliminar profesor</a>
Yolanda Ruiz Ruiz	C/ Mar Valladolid	741258	yolanda@ruiz.ruiz	<a href="#">Editar profesor</a> <a href="#">Ver profesor</a> <a href="#">Eliminar profesor</a>

Ilustración 62: Listar Profesores

Como en el caso de los alumnos, se pueden editar los datos de los profesores, ver los datos de cada uno de ellos, y eliminarlos.

## 6. PRUEBAS DE SOFTWARE: SELENIUM

Selenium es un *framework* para realizar pruebas sobre aplicaciones web. Se pueden escribir scripts en distintos lenguajes como C#, Java, PHP, Python, Perl, Ruby, etc., o, por el contrario, grabar y reproducir (*record & PlayBack*) pruebas con el navegador de internet empleado (es compatible con Windows, Linux y macOS) (SeleniumHQ, SeleniumHQ Browser Automation, 2019).

Selenium fue desarrollado por Jason Huggins en el año 2004 como una herramienta interna para la empresa *ThoughtWorks*. Poco a poco, otros programadores se unieron al proyecto de Huggins y continuaron con el desarrollo de esta herramienta, a la que llamaron Selenium-RC (*Remote Control*). En 2007, Huggins empezó a trabajar con Google, y orientaron el *software* que tenían hasta el momento hacia el testeo automático de páginas web. Así apareció Selenium *WebDriver* (2009) (SeleniumHQ, SeleniumHQ Browser Automation, 2019).

Hoy, el *software* desarrollado por Selenium cuenta con distintas modalidades: Selenium IDE, Selenium WebDriver, Selenium Grid y Selenium-RC:

- **Selenium IDE:** proporciona al usuario un entorno de desarrollo con la capacidad de grabar/reproducir pruebas funcionales sin la necesidad de realizar un *script*. Para trabajar con esta herramienta, basta con instalar un complemento en el navegador que se vaya a emplear para llevar a cabo las pruebas (Google Chrome o Firefox). Esta versión de Selenium apareció en 2006, y fue desarrollada por Shinya Kasatani (Colantonio, 2019).
- **Selenium WebDriver:** permite al usuario escribir *scripts* y enviárselos al navegador de internet, que los ejecuta y lleva a cabo lo que está escrito en ellos. Esto ocurre a través de un controlador que se encuentra en el navegador web y que permite al *script* acceder a la aplicación que se va a ejecutar. No es necesario emplear un servidor específico para que Selenium WebDriver funcione (Stewart, 2019).
- **Selenium Grid:** es un servidor que permite realizar pruebas en navegadores web que se ejecutan en máquinas remotas. Permite ejecutar pruebas en paralelo sobre múltiples máquinas y diferentes navegadores (SeleniumHQ, SeleniumHQ Browser Automation, 2019).

En este Trabajo de Fin de Grado se van a utilizar dos de las cuatro modalidades de Selenium:

- **Selenium IDE:** se ha empleado esta herramienta para familiarizarse con Selenium. Con ella se han grabado escenarios de prueba y luego se han

reproducido para ver que funcionaban correctamente. Pero Selenium IDE es una herramienta que no cubre todas las necesidades y presenta algunas carencias: al no basarse en la ejecución de ningún *script*, hay elementos a los que no es posible acceder, y eso puede dar lugar a **falsos fallos**. Como consecuencia, no es una herramienta que permita comprobar de forma definitiva y completa que la aplicación web funciona adecuadamente.

- **Selenium WebDriver**: una vez que se ha entendido lo que hace Selenium IDE, se han programado *scripts* en Python, que permiten solventar y completar las carencias que tiene Selenium IDE: de esta forma se pueden localizar todos los elementos de la página web con precisión, sin tener peligro de obtener falsos fallos.

En los apartados 7.1 y 7.2 se detallan los procedimientos y resultados obtenidos al usar estas herramientas que se han descrito.

## 6.1 Pruebas con Selenium IDE

Para usar **Selenium IDE** (Integrated Development Environment), hace falta instalar un plugin en el navegador que se vaya a emplear (en este caso, Google Chrome). El flujo de Selenium IDE es el mostrado en la Ilustración 63 (Selenium, 2019).

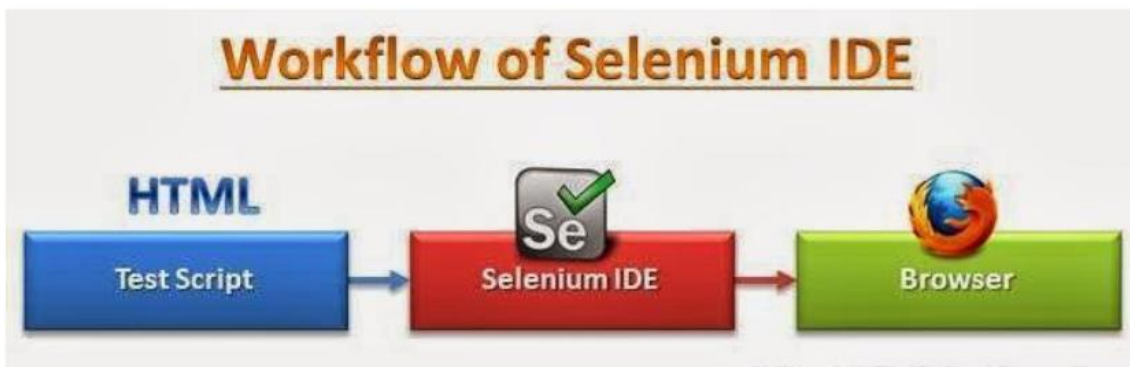


Ilustración 63: flujo de trabajo de Selenium IDE (Selenium, 2019)

Para probar la aplicación web que se ha desarrollado a lo largo este Trabajo de Fin de Grado se ha creado un proyecto en el plugin del navegador llamado **Pruebas\_Aplicacion\_Web**. Este proyecto se encuentra almacenado en un fichero llamado **Pruebas\_Aplicacion\_Wen.side**. En él se han grabado diferentes casos de prueba con Selenium IDE. Cada uno de los casos de prueba realizan una actividad específica en la aplicación web: inicio de sesión, registro de un alumno, edición del perfil personal, etc. El fin del uso de Selenium IDE es probar que, al reproducir la grabación que se ha realizado, el *software* reproduce de manera automática las instrucciones

introducidas, lo que implica que la aplicación web está funcionando de manera correcta. Si no lo hiciera, al reproducir la lista de instrucciones que tiene que llevar acabo, se produciría una pausa en una de ellas, y no se podría completar la reproducción.

### 6.1.1 Instalación de las herramientas necesarias y manejo de Selenium IDE

A continuación, se van a describir los pasos que debe seguir el usuario para hacer uso de Selenium IDE. Lo primero que hay que hacer es instalar el plugin necesario en el buscador que se va a emplear. Para ello se seguirán los siguientes pasos:

1. Abrir el navegador que se va a emplear (Google Chrome) y acceder a *Google Chrome Store*.
2. Buscar en **Extensiones** el *software* "Selenium IDE" y añadir la extensión al navegador (véase Ilustración 64).



Ilustración 64: Instalación de Selenium IDE en el navegador Google Chrome

3. Hacer click en el icono de Selenium IDE que aparece en la barra de herramientas del navegador que se está empleando (véase Ilustración 65).



Ilustración 65: Icono Selenium IDE en la barra de herramientas del navegador.

4. Ver la página principal con la que se va a trabajar. En ella se pueden distinguir diferentes zonas (véase Ilustración 66):
  - Arriba a la izquierda de la pantalla se puede ver el título del proyecto que se ha creado: **Pruebas\_Aplicación\_Web**.
  - En la parte lateral izquierda se puede ver la lista de escenarios de prueba que se han grabado.
  - El cuerpo principal de la pantalla muestra los pasos que ha de llevar a cabo Selenium IDE para reproducir el caso de prueba que se ha grabado. El color verde indica los pasos que se han realizado con éxito, mientras que los pasos marcados en rojo indican que la reproducción del caso grabado se ha parado ahí, debido a un problema.

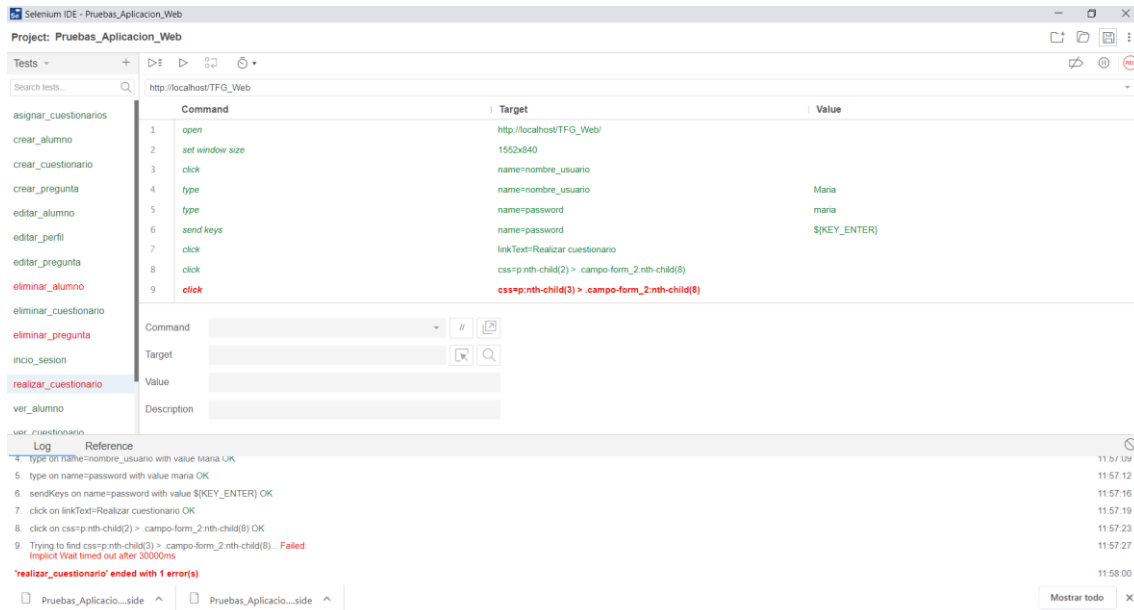


Ilustración 66: Pantalla de trabajo con Selenium IDE

Si se quiere abrir un proyecto que ya existe, como puede ser **Pruebas\_Aplicacion\_Web**, hay que seleccionar el icono de Selenium IDE que aparece en la parte superior del buscador (véase Ilustración 65). Aparecerá la pantalla mostrada en la Ilustración 67. Seleccionando la opción **Open an existing project**, se podrá seleccionar el fichero donde se haya guardado el proyecto que se quiere abrir. Los proyectos de Selenium IDE se guardan con la extensión **.side**. En este caso, el proyecto que se quiere abrir está guardado en el fichero **Proyecto\_Aplicacion\_Web.side**.

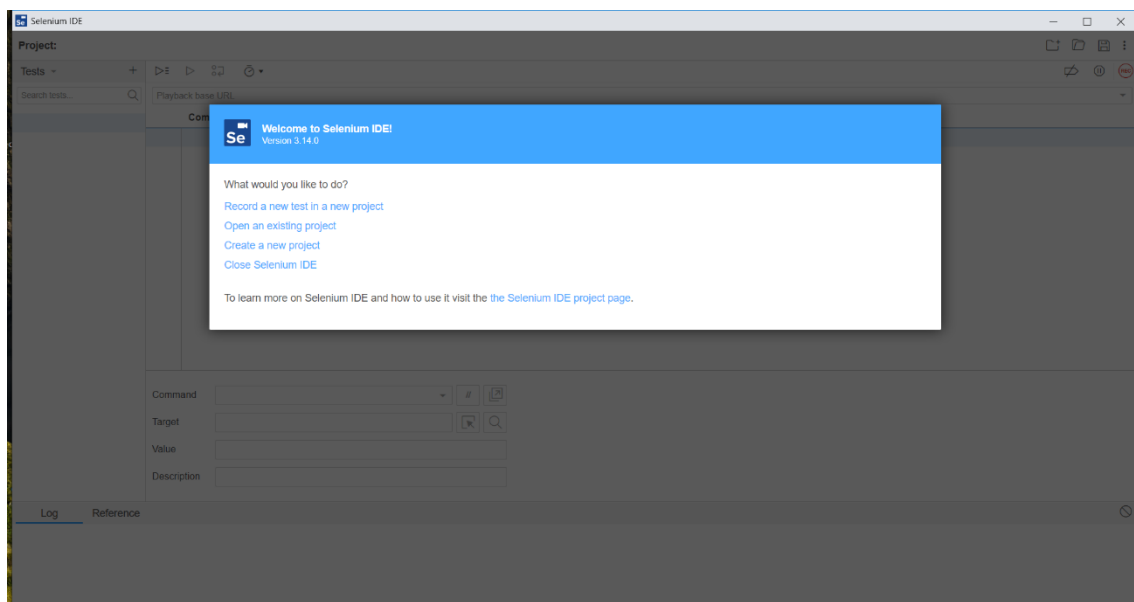


Ilustración 67: Abrir un proyecto existente con Selenium IDE

## 6.1.2 Casos de Prueba

Los siguientes apartados muestran las pruebas que se han realizado con Selenium IDE, y las conclusiones que se han extraído al emplear esta herramienta. No se han realizado todos los casos de prueba que componen a la Aplicación Web, porque con los que se han llevado a cabo han sido suficientes para extraer las conclusiones de esta herramienta: sirve para entrar en contacto con el mundo del *software testing*, pero no proporciona la precisión suficiente como para determinar si la Aplicación Web funciona de manera correcta o no. Por ello, ha sido necesario el uso de otra herramienta más precisa: Selenium WebDriver.

### 6.1.2.1 Prueba de inicio de sesión

El primer caso que se ha grabado ha sido el de inicio de sesión. Para grabar un caso hay que crear un nuevo caso de prueba y darle un nombre, en este caso se le ha llamado **inicio\_sesión** (véase Ilustración 68). Una vez creado, se indica la URL desde donde se quiere que se comience a probar el caso: **http://localhost/TFG\_Web**, que es donde está alojada la página web. Por último, para llevar a cabo la grabación, se pulsa el botón **REC** que aparece arriba a la derecha de la pantalla.

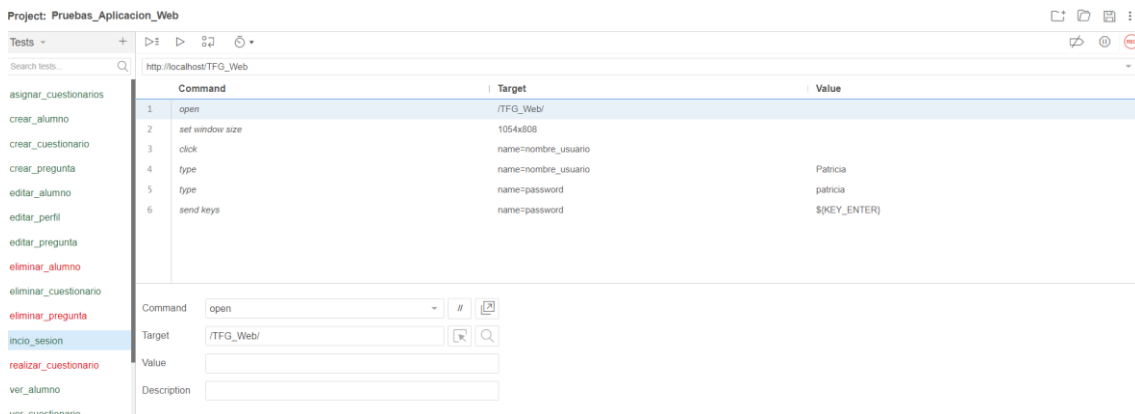


Ilustración 68: Caso inicio\_sesion

Automáticamente se abrirá una ventana en Google Chrome, en la URL que se ha indicado. Como en este caso se quiere probar que el inicio de sesión funciona correctamente, se introduce el nombre y la contraseña del usuario que quiere iniciar sesión. Para parar la grabación, una vez que se hayan realizado todas las acciones que se quieren probar, se pulsa el botón de **REC** otra vez, que ahora será un botón con el símbolo de **Pausa**.

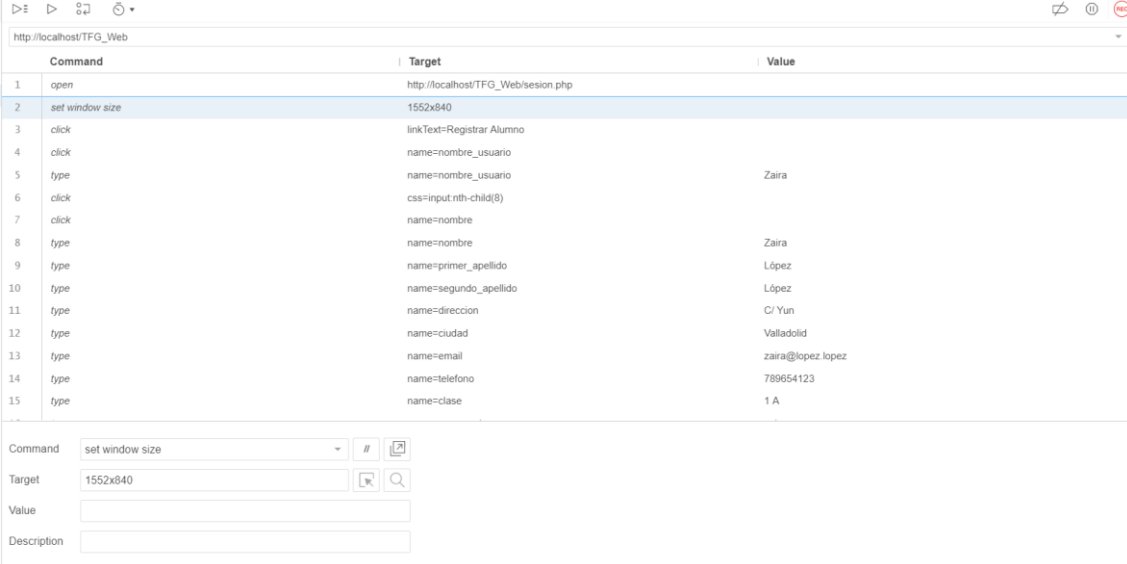
Para reproducir el caso que se ha grabado, se selecciona el botón del **Play** que aparece arriba del todo de la pantalla (véase Ilustración 68). Cada uno de los comandos grabados



se ejecutará automáticamente en una nueva ventana emergente. Con el botón del reloj que aparece a la derecha del **Play** se puede modificar la velocidad de reproducción de la prueba.

### 6.1.2.2 Prueba de crear alumno

Otro de los casos que se han grabado ha sido el de crear un nuevo alumno. El resultado de grabar el caso ha sido el mostrado en la Ilustración 69. La columna *command* indica el tipo de comando que se ejecuta en cada orden. La columna *Target* indica el contenido de cada acción. La columna *Value* indica el valor del contenido, si ese comando lo permite. Selenium IDE cuenta con un **Log** donde se almacenan las acciones realizadas y se indica si estas se han llevado a cabo con éxito o no.



Command	Target	Value
1 open	http://localhost/TFG_Web/sesion.php	
2 set window size	1552x840	
3 click	linkText=Registrar Alumno	
4 click	name=nombre_usuario	
5 type	name=nombre_usuario	Zaira
6 click	css=input:nth-child(8)	
7 click	name=nombre	
8 type	name=nombre	Zaira
9 type	name=primer_apellido	López
10 type	name=segundo_apellido	López
11 type	name=direccion	C/ Yun
12 type	name=ciudad	Valladolid
13 type	name=email	zaira@lopez.lopez
14 type	name=telefono	789654123
15 type	name=clase	1 A

Ilustración 69: Caso crear\_alumno

Cada una de las sentencias representa a uno de los comandos grabados (Ilustración 70). Como se puede ver al final de la lista de comandos, aparece en verde la frase **'crear\_alumno' completed successfully**. El caso se ha podido llevar a cabo correctamente, por lo que la aplicación web ha pasado la prueba.

Log	Reference
Running 'crear_alumno'	14:13:41
1. open on http://localhost/TFG_Web/sesion.php OK	14:13:41
2. setWindowSize on 1552x840 OK	14:13:44
3. click on linkText=Registrar Alumno OK	14:13:47
4. click on name=nombre_usuario OK	14:13:50
5. type on name=nombre_usuario with value Zaira OK	14:13:54
6. click on css=input:nth-child(8) OK	14:13:57
7. click on name=nombre OK	14:14:01
8. type on name=nombre with value Zaira OK	14:14:04
9. type on name=primer_apellido with value López OK	14:14:07
10. type on name=segundo_apellido with value López OK	14:14:10
11. type on name=direccion with value C/ Yun OK	14:14:13
12. type on name=ciudad with value Valladolid OK	14:14:16
13. type on name=email with value zaira@lopez lopez OK	14:14:20
14. type on name=telefono with value 789654123 OK	14:14:23
15. type on name=clave with value 1 A OK	14:14:26
16. type on name=password with value zaira OK	14:14:29
17. sendKeys on name=password with value \${KEY_ENTER} OK	14:14:32
'crear_alumno' completed successfully	14:14:33

Ilustración 70: Comandos grabados al ejecutar el Caso crear\_alumno

### 6.1.2.3 Prueba editar alumno

El caso de prueba **editar\_alumno**, tiene como fin gradar y reproducir el escenario de modificar los datos de un alumno en concreto de la aplicación web. Se parte del menú principal de la aplicación, el fichero **sesion.php**. Para editar a un alumno, el profesor primero tiene que seleccionar la opción **Ver Alumnos**, que se encuentra oculta tras la pestaña **Alumnos**.

Primero se graba el escenario, realizando los pasos que se van a querer reproducir más tarde, después, se reproduce el caso **editar\_alumno**. Pero en esta ocasión, el escenario no se va a completar con éxito. ¿Por qué?

Se está empleando una herramienta de **record & playback**, por lo que se reproducen las instrucciones grabadas a partir de clicks en la pantalla, no a través de la identificación de elementos de la página web. Por tanto, los elementos que se encuentran ocultos, como es en este caso el elemento **Ver Alumnos**, no podrán ser encontrados por Selenium IDE, a no ser que el usuario que está reproduciendo la prueba ayude al programa, aproximando el cursor a la zona de la pantalla donde se encuentra el elemento oculto.

También puede ocurrir que Selenium IDE tenga problemas para localizar otros elementos, aunque no estén ocultos. De nuevo, esto ocurre porque se ha grabado un escenario de prueba y se ha reproducido, por lo que la forma de localizar a los elementos no es exacta.

Si no se ayudara a Selenium IDE, el resultado de la misma prueba sería el mostrado en la Ilustración 71. En este caso el error concreto que se produce es debido a que Selenium IDE no puede localizar **Ver Alumnos**. Pasados treinta segundos, si Selenium IDE no ha localizado el elemento que busca, determina que el test ha sido fallido.

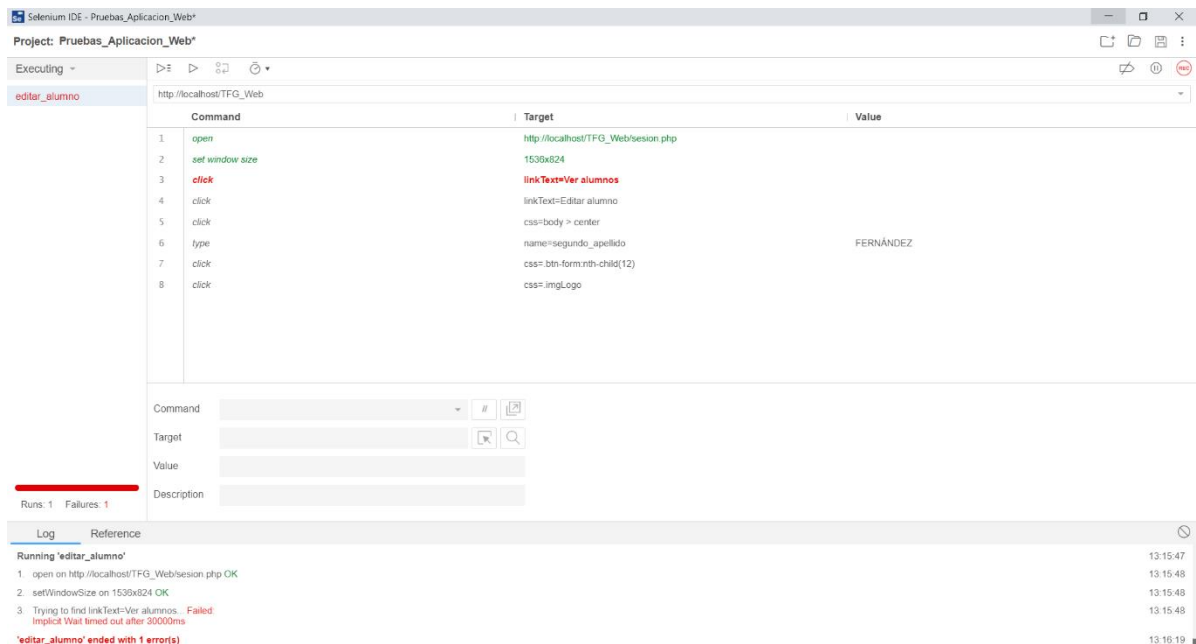


Ilustración 71: prueba editar\_alumno fallida

### 6.1.3 Conclusiones de las pruebas realizadas

Las **pruebas realizadas con Selenium IDE** pertenecen a la categoría de pruebas automatizadas de *record & playback*. Se trata de unas pruebas automatizadas simples que no requieren el desarrollo de ningún *script*, tarea bastante complicada y que requiere cierta destreza en el mundo de las pruebas de *software*.

Al no ser necesario ningún *script*, localizar ciertos elementos puede ser algo complicado para Selenium IDE. Esto hace imposible la comprobación de ciertos casos de prueba. Por este motivo, no se han realizado todos los casos de prueba posibles de la Aplicación Web, ya que se ha pensado que realizarlos sin obtener ningún resultado concluyente no tiene sentido.

Los escenarios de prueba grabados se han completado satisfactoriamente, aunque si es verdad que algunos de ellos han necesitado de “ayuda externa”. Esto ha ocurrido en todos los casos que han requerido acceder a alguna de las opciones del menú de profesores que se encontraban dentro de los menús desplegados (*Registrar Alumno, Crear Cuestionario, Ver Pregunta, etc.*). ¿Por qué ocurre esto? Porque estas opciones no son “visibles” durante la reproducción de la grabación realizada. Por ello, si se quiere que la grabación funcione correctamente, es necesario pasar el cursor del ratón por la zona donde se encuentran las opciones del menú a las que se quiere acceder. Por tanto, una conclusión de estas pruebas de *software* es que **acceder a ciertos elementos de la página web puede ser complicado si estos están ocultos** o no son visibles directamente sobre la pantalla.

Por otra parte, se ha visto que **ciertas acciones**, como *Eliminar Alumno* o *Realizar Cuestionario* **dan un error si se reproducen sin haberse restaurado su valor inicial antes**. Se ha comprobado que no se puede eliminar algo que ya no existe: la grabación trata de buscar ese elemento en la página web, pero este ya no está. Al pasar unos segundos, donde la grabación trata de buscar el elemento de forma reiterada, Selenium IDE marca ese comando como erróneo. Para solventar este problema, hay que volver a crear el elemento que se eliminó mientras se grabó el caso de prueba.

Por este motivo, se ha decidido emplear otra herramienta para aplicar *software testing* a la Aplicación Web: **Selenium WebDriver**. En el apartado 7.2 se explica todo lo relativo a esta herramienta, los casos de prueba realizados y las conclusiones extraídas.

## 6.2 Pruebas con Selenium WebDriver

Como los resultados obtenidos a partir de Selenium IDE no han permitido comprobar el correcto funcionamiento de la Aplicación Web, se ha hecho uso de Selenium WebDriver. Selenium IDE no permitía localizar ciertos tipos de elementos de la página web, al basarse en un mecanismo de grabado y reproducción. Por ello se va a emplear **Selenium WebDriver**, que se basa en *scripts*.

Para llevar a cabo las pruebas de *software*, se ha elegido utilizar el lenguaje de programación **Python**, aunque se podrían haber elegido otros lenguajes como *Java*, *Php*, etc. A continuación, se va a describir como instalar todas las herramientas necesarias para utilizar **Selenium WebDriver**.

### 6.2.1 Instalación de las herramientas necesarias para Selenium WebDriver

Se va a trabajar con Selenium WebDriver y Python, por lo que se necesitan dos herramientas para realizar este tipo de pruebas:

- Un entorno Python: se necesita tener instalado en el ordenador Python, para poder trabajar con este lenguaje de programación.
- Una biblioteca para Python con las herramientas necesarias para emplear Selenium.

Antes de explicar el procedimiento de instalación, ¿qué es Python?

### 6.2.1.1 Python

Python es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, creado en 1991 por Guido van Rossum. Uno de los rasgos que caracterizan a este lenguaje es el uso de las tabulaciones como forma de organización del código: una tabulación incorrecta se traduce en el funcionamiento incorrecto del *script* (Kuhlman, 2019).

Se ha dicho que Python es un lenguaje interpretado, pero no es del todo cierto. También es compilado. El código original es compilado a código byte, y después es interpretado, pero esta acción es realizada de forma automática por Python, el usuario no tiene que pedirselo de manera explícita (Cassel & Gauld, Python projects, 2014).

Los programas de Python se pueden escribir en diferentes editores de texto, como NotePad++, el bloc de notas, etc. Pero siempre tienen que guardarse con la extensión **.py**. También es posible emplear Python introduciendo las instrucciones directamente en el intérprete. Para acceder al intérprete de Python en un ordenador que lo tiene instalado, basta con abrir la consola de comandos y teclear **Python** (véase Ilustración 72) (Cassel & Gauld, Python projects, 2014).

Desde el intérprete se pueden ejecutar comandos simples. Si en lugar de ejecutar comandos, se quiere ejecutar algún *script* que se tenga programado, basta con abrir la consola de comandos en el directorio en el que se encuentra el *script* que se quiere ejecutar y escribir una sentencia con el formato: **\$ python nombre\_script.py**.

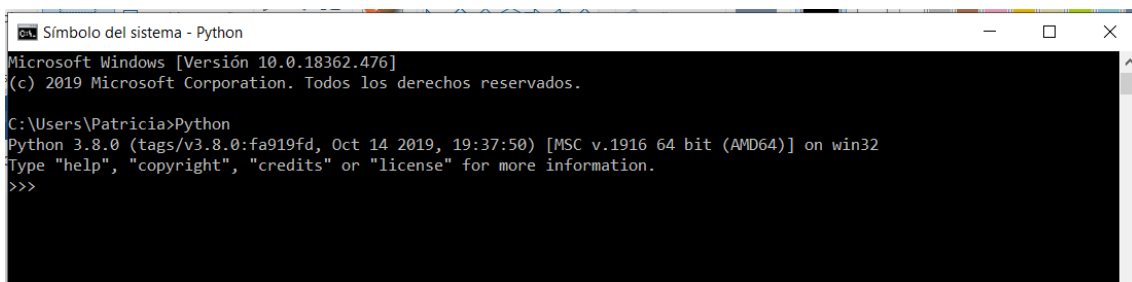


Ilustración 72: Acceder al intérprete de Python

Además, Python cuenta con el comando **help()**, que proporciona ayuda al usuario sobre cualquier comando que no sepa como emplear.

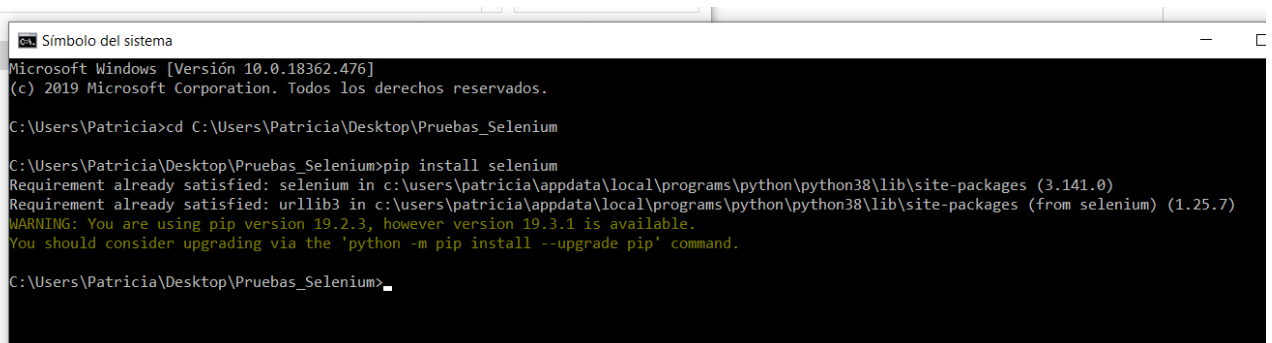
Pero lo realmente interesante de Python, es el uso de bibliotecas. Cuando se quieren hacer programas que implican tareas concretas, es necesario emplear bibliotecas específicas. Las bibliotecas vienen en “paquetes” (*package*): carpetas llenas de módulos necesarios para el desempeño de una tarea concreta. Todos los paquetes tienen un fichero llamado `__init__.py`, que se encuentra vacío y permite instalarle en el entorno python que se está empleando (Cassel & Gauld, Python projects, 2014).

En particular, cuando se emplee Python para programar los *scripts* de los casos de prueba que se quieren comprobar; antes va a ser necesaria la instalación del paquete Selenium en el entorno empleado.

### 6.2.1.2 Instalación de Python y la biblioteca Selenium

Para trabajar con Python, hay que descargar el programa al ordenador en el que se va a emplear. Una vez instalado Python, hay que descargar la biblioteca de Selenium, para poder trabajar con las funciones correspondientes a ella. Para ello se seguirán los siguientes pasos:

1. Se va a la página de Python: <https://www.python.org/downloads/>. En ella se selecciona la versión de Python que se quiere instalar.
2. Una vez instalado Python, es necesario instalar la biblioteca de Selenium. En la Ilustración 73 se puede ver la línea de comando que hay que ejecutar en la carpeta donde se va a trabajar con Selenium: **pip install selenium**.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18362.476]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Patricia>cd C:\Users\Patricia\Desktop\Pruebas_Selenium

C:\Users\Patricia\Desktop\Pruebas_Selenium>pip install selenium
Requirement already satisfied: selenium in c:\users\patricia\appdata\local\programs\python\python38\lib\site-packages (3.141.0)
Requirement already satisfied: urllib3 in c:\users\patricia\appdata\local\programs\python\python38\lib\site-packages (from selenium) (1.25.7)
WARNING: You are using pip version 19.2.3, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\Patricia\Desktop\Pruebas_Selenium>
```

Ilustración 73: instalación de la biblioteca Selenium

3. En la carpeta donde se encuentren los *scripts* de Python, es necesario que también se encuentren los ficheros **geckodriver.exe** y **geckodriver.log**. Geckodriver es un motor de navegación, incorporado en el navegador Mozilla Firefox. El Geckodriver actúa como un *proxy* (enlace) entre el cliente (Python) y el navegador (Mozilla Firefox). La ventaja de emplear Geckodriver es que emplea el protocolo **W3C WebDriver** para comunicar al navegador con Selenium, y este protocolo es universal: funciona con todas las versiones de Firefox. En la Ilustración 74 se puede ver el camino que siguen los datos desde que se programa el *script* hasta que se ejecutan en el buscador (Firefox). Marionette es un controlador de automatización para Geckodriver. Permite el control remoto de la interfaz de usuario de una página web. Gracias a Marionette se pueden controlar los menús de la página web y los campos de información que hay en

ellos. Esta va a ser la parte que permita replicar al *script* las acciones del usuario (Mozilla Source Tree Docs, 2019).



Ilustración 74: Esquema de paso de datos desde Selenium hasta Firefox (Guru99, 2019)

4. Por último, para ejecutar el *script* que se ha programado, se introduce en la consola de comandos la siguiente instrucción: **\$ python nombre\_fichero.py**.

### 6.2.2 Casos de Prueba

En los siguientes apartados se van a mostrar los resultados obtenidos al programar los *scripts* correspondientes a los casos de prueba. Se ha realizado un caso de prueba para cada uno de los casos de uso de la aplicación web, para comprobar el correcto funcionamiento de cada uno de ellos. En alguna ocasión, se han agrupado más de un Caso de Uso en el mismo *script*, ya que están relacionados entre sí. *También* se han hecho casos de prueba relacionados con escenarios que no se corresponden con ningún caso de uso como tal, pero que son acciones que se pueden realizar en la Aplicación Web, como pueden ser el inicio de sesión o la edición del perfil personal de un usuario de la plataforma de cuestionarios.

En el ANEXO I: Funciones útiles para el manejo de Selenium con Python, se recogen las funciones que se han empleado en los *scripts* programados para todos los casos de prueba: cuándo se emplea cada una de ellas, qué es lo que hacen y a qué tipo de elementos están orientadas.

A este documento se le ha adjuntado una carpeta con el nombre **Pruebas\_Selenium** en la que se encuentran todos los *scripts* que se han programado. Si se quieren ejecutar los *scripts* sin modificar nada en los mismos, se recomienda ejecutarlos con la base de datos en el estado en que esta se entrega. Esto se debe a que se han programado con una base de datos determinada, y ejecutarlos con otra diferente puede dar lugar a errores ajenos a los propios *scripts*. Por ejemplo: si se ejecuta el *script* relacionado con asignar los cuestionarios a los alumnos, y se indica que se seleccionen el segundo y tercer cuestionario de la lista de cuestionarios. ¿Qué ocurre si se utiliza una base de datos diferente a la que se ha empleado cuando se ha programado el *script*? Puede que en esa base de datos solo haya un cuestionario en la lista de cuestionarios que se quiera asignar a los alumnos y Selenium no sea capaz de localizar el segundo y tercer elemento de la lista.

El usuario puede ejecutar el *script* cuando quiera, pero con una base de datos diferente a la entregada, puede que sea necesario modificar los *scripts*. Se han incluido en todos los *scripts* comentarios descriptivos, indicando qué es lo que habría que cambiar y por qué valores habría que cambiarlos, con el fin de facilitar al usuario su edición, si ese fuera su propósito. También se incluye en cada uno de los *scripts* una descripción con lo que hace cada uno de ellos.

La carpeta **Pruebas\_Selenium** contiene los *scripts* que se van a listar a continuación. Los dos primeros no corresponden a ningún Caso de Uso, pero llevan a cabo acciones sencillas que han servido para la toma de contacto con Selenium Web Driver. Todos los demás *scripts* engloban a uno o más Casos de Uso:

- *iniciar\_sesion*: este *script* simula el inicio de sesión para un usuario concreto.
- *editar\_perfil\_propio*: este *script* simula el cambio de algún campo sobre el perfil propio de un usuario.
- *crear\_pregunta*: este *script* simula el Caso de Uso CrearPregunta.
- *crear\_cuestionario*: este *script* simula el Caso de Uso CrearCuestionario.
- *registrar\_alumno*: este *script* simula el Caso de Uso CrearAlumno.
- *registrar\_profesor*: este *script* simula el Caso de Uso CrearProfesor.
- *editar\_pregunta*: este *script* simula los Casos de Uso VerPregunta y EditarPregunta.
- *editar\_cuestionario*: este *script* simula los Casos de Uso VerCuestionario y EditarCuestionario.
- *editar\_alumno*: este *script* simula los Casos de Uso VerAlumno y EditarAlumno.
- *editar\_profesor*: este *script* simula los Casos de Uso VerProfesor y EditarProfesor.
- *cuestionario\_completo*: este *script* engloba varios Casos de Uso. En él se incluye una descripción inicial de todas las acciones que se van a llevar a cabo, y a lo largo del código se detalla dónde se produce cada una de ellas.
  - En primer lugar, el profesor inicia sesión.
  - El profesor lista los alumnos y consulta los cuestionarios pendientes del primer alumno de la lista, Luis (Caso de Uso VerCuestionariosPendientes).
  - El profesor elimina todos los cuestionarios pendientes del alumno.



- El profesor asigna un cuestionario al alumno (Caso de Uso AsignarCuestionarios).
  - El alumno Luis inicia sesión.
  - El alumno realiza el cuestionario que se le ha asignado (Caso de Uso RealizarCuestionario).
- `corregir_cuestionario`: este *script* simula el Caso de Uso CorregirCuestionario.
  - `ver_cuestionarios_respondidos`: este *script* simula el Caso de Uso VerCuestionariosCorregidos.

En los siguientes apartados, se van a explicar algunos de los *scripts* que se han programado: cómo se han programado, qué hace cada una de las sentencias, etc. También se van a analizar los errores que se han ido obteniendo según se iban programando los *scripts*, y cómo ha afectado esto a la aplicación.

#### 6.2.2.1 Cuestionario\_completo.py

El *script* mostrado en la Ilustración 75 contiene el código necesario del *script* más complejo y completo que se ha programado. En él hay ejemplos de todas las herramientas y sentencias que se han empleado en los demás *scripts*. Por eso se ha decidido explicarlo paso a paso.

Las líneas 1-10 son comentarios que describen las acciones que se van a llevar a cabo a lo largo de todo el *script*.

Las líneas 13-16 importan los módulos básicos que se van a emplear.

- **unittest**: proporciona el marco para organizar los casos de prueba. Se basa en JUnit (Java).
- **selenium.webdriver**: proporciona las implementaciones de WebDriver.
- **time**: proporciona el módulo que contiene las funciones relacionadas con el tiempo, como la función *sleep()*, que permite introducir pausas en unidades de segundo. Por ejemplo, *sleep(2.4)* introduce una pausa de 2.4 segundos.

La línea 17 indica que se hereda la clase de test de prueba de la prueba unitaria **TestCase**.

Las líneas 18 y 19 inicializan el caso de prueba abriendo el buscador Firefox.

Las líneas 22, 36 y 51 indican la URL a la que se quiere que se dirija el navegador.

La línea 25 comprueba que el título de la página es del de **Plataforma de Encuestas**.

Las líneas 26 y 28 buscan los elementos con el nombre que se indica. Este tipo de sentencias aparecen de manera recurrente en los *scripts*. Existen más métodos de búsqueda de elementos en páginas web, pero solo se ha empleado la búsqueda por nombre, por id y por un enlace parcial.

Las líneas 27 y 29 introducen el valor que se indica en los campos que se ha seleccionado. Este tipo de sentencia solo es válida para enviar información en forma de texto. Si lo que se quiere es seleccionar un *radioButton* o una *checkBox*, se emplean otras sentencias. Este tipo de sentencia también se emplea en múltiples ocasiones.

Las líneas 46, 59, 69 y 110 pulsán el botón que envía la información de los formularios que se rellenan.

Cuando se quiere seleccionar una o más *checkBox*, como ocurre en la línea 66, hay que tener en cuenta que se están seleccionando elementos de un vector, en este caso el vector **alumno[]**. Por ello, es necesario indicar entre los corchetes el o los elementos del vector que se quieren seleccionar. Lo mismo ocurre en el caso del *radioButton*. La única diferencia entre ambos es que en este último caso, no es posible seleccionar más de uno de los elementos del vector. En este *script* se puede ver la selección de *radioButtons* o *checkboxes* en las líneas 66, 95, 99, 103 y 107.

Las líneas 120 y 121 se encargan de poner fin al caso de prueba y cerrar la pestaña del buscador que se ha abierto al comprobar el funcionamiento del caso de prueba.

En el ANEXO I se puede leer de manera detallada la descripción de las funciones que se han empleado y cómo se combinan entre ellas para obtener los resultados que se quieren.

A lo largo del *script* se puede ver la sentencia **time.sleep(2)** en múltiples ocasiones. Esta sentencia permite introducir pausas de los segundos que se indican entre paréntesis. Es necesario introducir pausas entre sentencias por dos motivos:

- Por la visibilidad y claridad del test, ya que, sin estas pausas, habría acciones que se producirían muy rápido, y el usuario no sería capaz de percatarse de lo que está ocurriendo.
- En ocasiones, cuando Selenium busca un elemento, puede producirse un error debido a la falta de tiempo en la búsqueda. Introduciendo una pausa, Selenium tiene el tiempo suficiente para localizar los elementos que necesita.

```

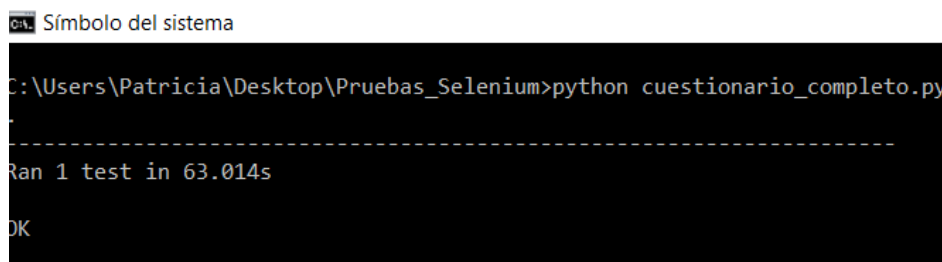
C:\Users\Patricia\Desktop\Pruebas_Selenium\cuestionario_completo.py - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?
registrar_alumno.py editar_alumno.py crear_cuestionario.py editar_cuestionario.py corregir_cuestionario.py registrar_pregunta.php editar_pregunta.php cuestionario_completo.py registrar_profesor.py editar_profesor.py ver_cuestio

1 #ASIGNACIÓN A LA ALUMNO LUIS DEL CUESTIONARIO DE CIENCIAS, REALIZACIÓN DEL CUESTIONARIO DE CIENCIAS
2 #En este fichero de prueba se realizan múltiples tareas:
3 #1. El profesor inicia sesión
4 #2. Se va a ver cuestionarios pendientes
5 #3. Se eliminan todos los cuestionarios que el alumno tenga pendientes
6 #4. asigna un cuestionario a un alumno
7 #5. El profesor cierra sesión
8 #6. El alumno inicia sesión
9 #7. El alumno realiza el cuestionario que se le ha asignado
10 #8. El alumno cierra sesión
11
12
13 import time
14 import unittest
15 from selenium import webdriver
16 from selenium.webdriver.common.keys import Keys
17 class PythonOrgSearch(unittest.TestCase):
18     def setUp(self):
19         self.driver = webdriver.Firefox()
20     def test_search_in_python_org(self):
21         driver = self.driver
22         driver.get("http://localhost/TFG_Web/index.php")
23
24         #1.inicio de sesión del profesor
25 self.assertIn("Plataforma de Encuestas", driver.title)
26 elem = driver.find_element_by_name("nombre_usuario")
27 elem.send_keys("PaTricia")
28 elem = driver.find_element_by_name("password")
29 elem.send_keys("patricia")
30 elem.send_keys(Keys.RETURN)
31 time.sleep(3)
32
33 #2. Después de iniciar sesión vamos a ver los cuestionarios pendientes del primer alumno de la lista y eliminar todos los cuestionarios que tiene pendientes
34 #Esto es necesario porque se le va asignar un tipo de cuestionario concreto, y como el alumno va a realizar el primer cuestionario que tenga asignado,
35 #es necesario eliminar todos aquellos que tuviera asignados anteriormente. Es como una especie de inicialización a cero
36 driver.get("http://localhost/TFG_Web/listar_alumnos.php")
37 time.sleep(1)
38 continue_link = driver.find_element_by_partial_link_text('Ver Cuestionarios Pendientes').click()
39 time.sleep(3)
40
41 #3. El campo cero del vector cuestionario_alumno[] corresponde a la opción eliminar todos los cuestionarios que el alumno tenga asignados, y es la opción que se va a elegir
42 checkbox = driver.find_elements_by_name("cuestionario_alumno")
43 checkbox[0].click()
44 #Se envía la información a la siguiente pantalla
45 #Se modifica el cuestionario
46 driver.find_element_by_name("submit").click()
47 time.sleep(2)
48
49 #4.Asignación de un cuestionario a un alumno:
50 # Se asigna el primer cuestionario de la lista (para asignar cualquier otro, basta con cambiar el índice del vector cuestionario[])
51 driver.get("http://localhost/TFG_Web/mostrar_cuestionarios.php")
52 time.sleep(1)
53 #checkbox es un vector que contiene todas las preguntas que se pueden añadir al cuestionario. En este caso se han elegido
54 #la primera (checkbox[0]), la segunda (checkbox[1]) y la tercera (checkbox[2])
55 checkbox = driver.find_elements_by_name("cuestionario[]")
56 checkbox[0].click()
57 #Se envía la información a la siguiente pantalla
58 #Se modifica el cuestionario
59 driver.find_element_by_name("submit").click()
60 time.sleep(2)
61
62 # Se asigna el cuestionario seleccionado al segundo alumno de la lista (para asignar cualquier otro, basta con cambiar el índice del vector alumno[])
63 #checkbox es un vector que contiene todas las preguntas que se pueden añadir al cuestionario. En este caso se han elegido
64 #la primera (checkbox[0]), la segunda (checkbox[1]) y la tercera (checkbox[2])
65 checkbox = driver.find_elements_by_name("alumno[]")
66 checkbox[0].click()
67 #Se envía la información a la siguiente pantalla
68 #Se modifica el cuestionario
69 driver.find_element_by_name("submit").click()
70 time.sleep(2)
71
72 #5. Cierro sesión del profesor
73 elem = driver.find_element_by_name("cerrar_sesion")
74 elem.send_keys(Keys.RETURN)
75 time.sleep(3)
76
77 #6.inicio de sesión del alumno
78 self.assertIn("Plataforma de Encuestas", driver.title)
79 elem = driver.find_element_by_name("nombre_usuario")
80 elem.send_keys("luis")
81 elem = driver.find_element_by_name("password")
82 elem.send_keys("luis")
83 elem.send_keys(Keys.RETURN)
84 time.sleep(3)
85
86 #7 Elección del cuestionario
87 time.sleep(1)
88 continue_link = driver.find_element_by_partial_link_text('Realizar cuestionario').click()
89 time.sleep(3)
90 #Se sabe que el cuestionario tiene una pregunta con dos opciones, una pregunta abierta, una pregunta con dos opciones y una pregunta abierta
91
92 #Realización del cuestionario y entrega
93 #En la primera pregunta se selecciona la primera respuesta
94 checkbox = driver.find_elements_by_name("respuesta[0]")
95 checkbox[0].click()
96 time.sleep(2)
97 #En la segunda pregunta se selecciona la segunda respuesta
98 checkbox = driver.find_elements_by_name("respuesta[1]")
99 checkbox[1].click()
100 time.sleep(2)
101 #En la tercera pregunta se escribe "respuesta 3"
102 checkbox = driver.find_elements_by_name("respuesta[2]")
103 checkbox[0].send_keys("respuesta 3")
104 time.sleep(2)
105 #En la cuarta pregunta se selecciona la primera pregunta
106 checkbox = driver.find_elements_by_name("respuesta[3]")
107 checkbox[0].click()
108 time.sleep(2)
109
110 driver.find_element_by_name("submit").click()
111 time.sleep(2)
112
113 #8. Cierro sesión del alumno
114 elem = driver.find_element_by_name("cerrar_sesion")
115 elem.send_keys(Keys.RETURN)
116 time.sleep(3)
117
118 assert "No results found." not in driver.page_source
119 def tearDown(self):
120     self.driver.close()
121 if __name__ == "__main__":
122     unittest.main()
123

```

Ilustración 75: script cuestionario\_completo.py

El resultado que se obtiene al ejecutar el comando `$ python cuestionario_completo.py` es el que puede observarse en la Ilustración 76:



```
Símbolo del sistema
C:\Users\Patricia\Desktop\Pruebas_Selenium>python cuestionario_completo.py
-----
Ran 1 test in 63.014s
OK
```

Ilustración 76: Resultado correcto del script de prueba cuestionario\_completo

En la imagen se puede ver que el test se ha pasado correctamente y que ha tardado 63.014 segundos en completarse.

#### 6.2.2.2 Error: elemento no encontrado

En el ejemplo anterior, se han buscado elementos de distintas maneras: a través de su nombre, de su id, o de una parte de un enlace. Pero ¿qué habría ocurrido si no hubiera ningún campo en la aplicación web con el nombre, id o parte del enlace por el que se buscaba al elemento? Se ha hecho la prueba en el *script iniciar\_sesion.py* (porque es más corto de ejecutar). El resultado se puede ver en la Ilustración 77. Selenium indica que no es capaz de localizar al elemento a través del nombre que se le ha indicado.

```
Símbolo del sistema
C:\Users\Patricia\Desktop\Pruebas_Selenium>python iniciar_sesion.py
E
=====
ERROR: test_search_in_python_org (__main__.PythonOrgSearch)
-----
Traceback (most recent call last):
  File "iniciar_sesion.py", line 16, in test_search_in_python_org
    elem = driver.find_element_by_name("nombre_usuario")
  File "C:\Users\Patricia\AppData\Local\Programs\Python\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 496, in find_element_by_name
    return self.find_element(by=By.NAME, value=name)
  File "C:\Users\Patricia\AppData\Local\Programs\Python\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 976, in find_element
    return self.execute(Command.FIND_ELEMENT, {
  File "C:\Users\Patricia\AppData\Local\Programs\Python\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 321, in execute
    self.error_handler.check_response(response)
  File "C:\Users\Patricia\AppData\Local\Programs\Python\Python38\lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 242, in check_response
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.NoSuchElementException: Message: Unable to locate element: [name="nombre_usuario"]
-----
Ran 1 test in 33.056s

FAILED (errors=1)
```

← No es capaz de encontrar el elemento nombre\_usuario

Ilustración 77: Resultado incorrecto debido a un error en la escritura del nombre por el que se busca al elemento en el script iniciar\_sesion.py

Lo mismo puede ocurrir si el elemento que se busca no existe. Esto ha ocurrido en los casos dónde ha sido necesario pulsar un botón, como por ejemplo en los casos en los que se envía la información de un cuestionario. El error se debe a que no ha sido posible localizar el botón, porque este no tiene un atributo **name** definido. El error obtenido es similar al anterior (Ilustración 78).

```
Símbolo del sistema
C:\Users\Patricia\Desktop\Pruebas_Selenium>python crear_cuestionario.py
E
=====
ERROR: test_search_in_python_org (__main__.PythonOrgSearch)
-----
Traceback (most recent call last):
  File "crear_cuestionario.py", line 28, in test_search_in_python_org
    driver.find_element_by_name("submit").click()
  File "C:\Users\Patricia\AppData\Local\Programs\Python\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 496, in find_element_by_name
    return self.find_element(by=By.NAME, value=name)
  File "C:\Users\Patricia\AppData\Local\Programs\Python\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 976, in find_element
    return self.execute(Command.FIND_ELEMENT, {
  File "C:\Users\Patricia\AppData\Local\Programs\Python\Python38\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 321, in execute
    self.error_handler.check_response(response)
  File "C:\Users\Patricia\AppData\Local\Programs\Python\Python38\lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 242, in check_response
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.NoSuchElementException: Message: Unable to locate element: [name="submit"]
-----
Ran 1 test in 26.942s

FAILED (errors=1)
```

Ilustración 78: Resultado incorrecto debido a la falta del atributo name en el botón de un formulario

### 6.2.3 Conclusiones de las pruebas realizadas

Las pruebas realizadas con Selenium WebDriver permiten al usuario comprobar que es posible localizar elementos en la aplicación web y solventar todos aquellos problemas relacionados con la mala definición de los elementos en la aplicación web. A través de los *scripts* que se han programado, se han podido reproducir los principales escenarios de uso de la Aplicación Web, y ver cómo quedaría esta reproducción desde el punto de vista del usuario final. En esta ocasión, ha sido más fácil llevar a cabo las pruebas que con Selenium IDE. Al poder especificar mediante código qué se quería buscar y cómo se querían tratar los datos y variables de la Aplicación Web, se ha tenido la posibilidad de reproducir Casos de Prueba más complejos y de manera satisfactoria.

### 6.3 Conclusión y comparación entre Selenium IDE y Selenium WebDriver

Cómo se ha podido ver en las pruebas realizadas por Selenium IDE y Selenium WebDriver, la primera herramienta (banco de pruebas en el documento **Pruebas\_Aplicacion\_Wen.side**) solo permite realizar una serie de pruebas muy limitadas. Su principal aportación ha sido la de toma de contacto y familiarización con Selenium, y ser la base a partir de la cual se ha utilizado Selenium WebDriver y se ha podido programar los *scripts* entregados en la carpeta **Pruebas\_Selenium**.

Selenium WebDriver es una herramienta mucho más versátil, que permite un manejo exacto de la Aplicación Web. Usado con destreza puede llegar a ser una herramienta realmente potente. En la Tabla 30 se puede ver una comparativa entre ambas herramientas.

	Ventajas	Desventajas
Selenium IDE	<ul style="list-style-type: none"> <li>• Fácil de utilizar.</li> <li>• No se necesitan conocimientos previos de programación.</li> </ul>	<ul style="list-style-type: none"> <li>• Opciones limitadas.</li> <li>• Puede dar lugar a falsos errores.</li> </ul>
Selenium WebDriver	<ul style="list-style-type: none"> <li>• Permite realizar pruebas de <i>software</i> de manera precisa.</li> <li>• Es fácil localizar elementos de la Aplicación Web.</li> <li>• Permite programar escenarios de uso complejos.</li> </ul>	<ul style="list-style-type: none"> <li>• Son necesarios conocimientos sobre lenguajes de programación.</li> <li>• Es necesario conocer las funciones de Selenium.</li> </ul>

Tabla 30: Ventajas y desventajas de Selenium IDE y Selenium WebDriver

Aunque solo se hayan realizado pruebas con Selenium IDE y Selenium WebDriver, implícitamente, también se han realizado muchas pruebas a medida que se desarrollaba la Aplicación Web. Como se ha dicho en el apartado 2.3 (Tipos de pruebas según su funcionalidad), **las pruebas funcionales consisten en comprobar si los requisitos funcionales se cumplen**. Para ver si esto es cierto, lo primero que se ha hecho es una tabla con los requisitos funcionales de la aplicación web que se ha diseñado. A lo largo del desarrollo de la aplicación, se han tenido estos requisitos funcionales presentes, para conseguir que todos ellos se cumplieran. Muchas veces, **mientras se desarrolla un proyecto, se está llevando a cabo *software testing* sin ser conscientes de ello**.

Una de las categorías de las **pruebas funcionales** era la **representación de los Casos de Uso**: formalizar los Casos de Uso es importante para dejar claros los estados por los que se puede pasar y establecer la relación actor-sistema. Para lograr esto, se han descrito los Casos de Uso que se querían tener en la aplicación antes de comenzar a programarla, para tener una base orientativa de lo que se debía esperar de ella. A partir de los Casos de Uso se ha construido la aplicación web, y una vez que esta se ha terminado, se han vuelto a consultar los Casos de Uso originales. Algunas de las acciones descritas al inicio del proyecto han cambiado respecto al resultado final: esto ha ocurrido porque, aunque se tenga una idea inicial sobre cómo llevar a cabo la implementación, a la hora de llevarlo a la práctica hay factores que pueden cambiar y afectar la implementación. Por ello, los Casos de Uso han sido actualizados al terminar la aplicación web, con el fin de que sean fieles a las funcionalidades reales de la aplicación.

Las **pruebas no funcionales** consisten en comprobar algunos aspectos de la aplicación: usabilidad, mantenibilidad, portabilidad, etc. La **usabilidad** define cómo de fácil y amigable es interactuar con la aplicación diseñada. Para probar esta característica, se ha dejado la aplicación a personas ajenas a ella y se les ha pedido que interactúen con la aplicación web. La finalidad de esto ha sido ver si alguien que usara por primera vez la aplicación tendría problemas para habituarse a ella. En función de las críticas recibidas y las sugerencias realizadas, se han ido cambiando partes de la interfaz de usuario hasta llegar a la configuración actual.

## 7. ESTUDIO ECONÓMICO

Para realizar el estudio económico sobre la Aplicación Web que se ha desarrollado, hay que tener en cuenta una serie de factores:

- Herramientas empleadas para el desarrollo: hardware (ordenador) y software (sistema operativo y programas empleados).
- Coste de las horas de trabajo empleadas.

El proyecto se ha desarrollado con un ordenador portátil que fue adquirido en febrero de 2018, hace 22 meses. Costó 850€ y se ha empleado durante seis meses. Por lo que el coste resultante del equipo hardware empleado es de 234,54€.

El software que se ha empleado ha sido:

- Sistema Operativo Windows y todas sus herramientas (Word).
- VisualStudioCode.
- MAMP.
- PhpMyAdmin.
- Selenium.

El Sistema Operativo y las herramientas vinculadas a él están incluidas en el coste del ordenador portátil, ya que vinieron incluidas en él. El resto de software que se ha empleado es abierto, por lo que su coste ha sido 0€.

Para desarrollar la Aplicación Web se estima haber empleado unas 350 horas. El salario de un desarrollador de aplicaciones web con un año de experiencia es de 12€. No obstante, mi experiencia como desarrolladora no llega a un año, por lo que la hora de trabajo costaría 8€ (indeed, 2020). Por tanto, el coste de las horas de trabajo asciende a 2.800€.

El coste total de la Aplicación Web desarrollada sería de 3.034,54€.



## 8. CONCLUSIONES

El *software testing* y el desarrollo de aplicaciones web generan una gran cantidad de empleos en la actualidad y representan una gran parte de la actividad de las empresas que se dedican al desarrollo de *software*. Por eso, obtener una visión global y una pequeña formación en ambos campos es algo bastante útil para las destrezas de un futuro ingeniero de telecomunicaciones.

A lo largo de este proyecto he podido aprender desde cero y realizar un estudio teórico, a qué se dedica, para qué sirve, qué persigue y cómo se organiza y clasifica el mundo del *software testing*. Realizar la investigación sobre este campo me ha hecho darme cuenta de lo amplio que es y de la cantidad de conocimientos previos que requiere una correcta utilización de las herramientas que permiten comprobar la calidad del *software*. Por otra parte, impresiona conocer la cantidad de personas que se encuentran detrás de que las cosas más cotidianas e inteligentes que nos rodean en nuestro día a día: que nuestro móvil funcione correctamente, que una aplicación funcione sin problemas, que el *software* integrado en el ascensor cumpla su cometido, etc.

Dedicarse al *software testing* no es algo que se pueda hacer sin una formación previa y especializada, es un campo muy amplio que requiere estudios específicos. Una parte del *software testing* es realizada por el equipo que diseña el producto final, otras pruebas son realizadas por equipos especializados ajenos al proyecto y otras pueden ser realizadas por personas inexpertas, que validen como consumidores el producto que se va a lanzar.

El desarrollo de aplicaciones web no tiene límites. El propio programador es el que debe decidir hasta dónde seguir. Las posibilidades que ofrecen las nuevas herramientas y los nuevos medios son infinitas: plantillas realizadas por expertos, *frameworks* que facilitan la tarea del programador, aplicaciones que facilitan el manejo de las bases de datos, etc. Sin la existencia de estas herramientas, el diseño y desarrollo de las aplicaciones web sería mucho más tedioso y complicado.

El constante avance de este campo hace que aparezcan nuevas herramientas que faciliten el cometido del programador constantemente, siendo difícil imaginar hasta qué punto se puede llegar.

En el ámbito personal, nunca había realizado una aplicación web, y un trabajo autodidacta ha sido fundamental para aprender a utilizar cada uno de los muchos lenguajes de programación y otras herramientas empleadas en el desarrollo de las aplicaciones web. Existen muchas herramientas, como pueden ser los *frameworks*, que implican tener una base de conocimientos previos antes de utilizarlos. Lo mismo ocurre con las plantillas de HTML y CSS que se pueden encontrar por internet o el uso de

plataformas como *Bootstrap*. Por este motivo, no se ha podido sacar beneficio de todas las ventajas que se pueden encontrar en Internet, pero se ha adquirido la capacidad de poder emplearlas en otros proyectos futuros.

Los navegadores que empleamos habitualmente para utilizar Internet poseen múltiples funcionalidades para moverse por una aplicación web. Conocer de cerca algo a lo que estamos tan habituados como son los sitios web, me ha hecho tener otra visión sobre la web y apreciar todo el trabajo que hay detrás de cada una de las páginas web que consulto en Internet.

Existen potentes herramientas como Selenium, que permiten al programador probar el correcto funcionamiento de las aplicaciones web. Un amplio conocimiento sobre este *framework* permite realizar un gran campo de pruebas sobre el *software* que se programa.

Ante todo, este proyecto ha sido algo útil y práctico, y me ha proporcionado conocimientos que puede que termine usando en un futuro. Además, ha cubierto las carencias que tenía sobre este campo y ha puesto a prueba mi capacidad de trabajo autónomo.

## 9. LÍNEAS FUTURAS

Cuando empecé a desarrollar este Trabajo Fin de Grado, no sabía nada sobre los lenguajes de programación y herramientas con las que he trabajado, por ello, existen ciertas cosas que se podrían mejorar, una vez sentadas las bases de la aplicación web.

- Usar plantillas ya realizadas en internet (de código abierto o de pago) en lugar del diseño empleado para la interfaz de usuario de la aplicación, con el fin de mejorar apariencia de la aplicación web.
- Usar *frameworks* como *Laravel* o similares, que permitieran optimizar el código, por ejemplo, empleando menos líneas de código de las que se emplean en este proyecto.
- Introducir funcionalidades extra a la aplicación web: que el usuario pudiera poner una fotografía personal en su perfil, incluir un chat o servicio de mensajería a través del que los profesores y alumnos pudieran comunicarse, una zona de notificaciones donde aparecieran los cuestionarios que tienen pendientes los alumnos y les indicaran los plazos que tienen para responder a cada uno de ellos, poder clasificar a los alumnos por clase para así poder asignar el mismo cuestionario a toda una clase y no a cada alumno de manera individual, etc.

## Referencias

- Antón Rodríguez, M., & Pérez Juárez, M. Á. (agosto de 2019). JavaScript. Valladolid, España: Universidad de Valladolid.
- Apache JMeter. (09 de 01 de 2020). *jmeter.apache.org*. Obtenido de <https://jmeter.apache.org/>
- Berners-Lee, T. (3 de 11 de 1992). *Tags used in HTML*. Obtenido de World Wide Web Consortium: <http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html>
- Bolaños Alonso, D., Sierra Alonso, A., & Alarcón Rodríguez, M. (2008). *Pruebas de software y JUnit. Un análisis en profundidad y práctico*. Madrid, España: Pearson.
- Cassel, L., & Gauld, A. (2014). *Python projects*. Indianapolis: John Wiley & Sons, Inc.
- Cassel, L., & Gauld, A. (2014). *Python Projects*. Indianapolis: John Wiley & Sons, Inc.
- Clark, S. (25 de 09 de 2019). *Web-based Mobile Apps of the Future Using HTML5, CSS and JavaScript*. Obtenido de HTML Goodies: <https://www.htmlgoodies.com/beyond/article.php/3893911/Web-based-Mobile-Apps-of-the-Future-Using-HTML-5-CSS-and-JavaScript.htm>
- Code, V. S. (15 de 09 de 2019). *Language Support in Visual Studio Code*. Obtenido de <https://code.visualstudio.com/docs/languages/overview>
- Codex, W. (15 de 09 de 2019). *Installing WordPress Locally on your Mac with MAMP*. Obtenido de [https://codex.wordpress.org/Installing\\_WordPress\\_Locally\\_on\\_Your\\_Mac\\_With\\_MAMP](https://codex.wordpress.org/Installing_WordPress_Locally_on_Your_Mac_With_MAMP)
- Cohn, M. (2010). *Succeeding with agile*. United States: Addison-Wesley.
- Colantonio, J. (15 de 10 de 2019). *Test Guild: Stunning Return of Selenium IDE*. Obtenido de <https://testguild.com/selenium-ide/>
- Consortium, W. W. (20 de 09 de 2019). *HTML 4.0 Specification - W3C Recommendation - Conformance: requirements and recommendations*. Obtenido de <https://www.w3.org/TR/REC-html40-971218/conform.html#deprecated>
- Chamberlin, D. D. (October-December 2012). Early History of SQL. *IEE Annals of the History of Computing*, 78-82.
- Flaticon. (28 de 10 de 2019). *Flaticon*. Obtenido de <https://www.flaticon.es/>
- Gauchat, J. D. (2012). *El gran libro de HTML5, CSS3 y JavaScript*. Barcelona: Marcombo.

- Gosselin, D., Kokoska, D., & Easterbrooks, R. (2010). *PHP programming with MySQL*. Boston, USA: Course Technology. Cengage Learning.
- Graham, D., van Veenendaal, E., Evans, I., & Black, R. (2010). *Foundations of software testing*.
- Grazas, J. (25 de 09 de 2019). *javiergarzas.com*. Obtenido de <https://www.javiergarzas.com/2015/01/automatizacion-pruebas.html>
- Guru99. (2 de 12 de 2019). *Gecko (Marionette) Driver Selenium: Download, Install, Use with Firefox* . Obtenido de <https://www.guru99.com/gecko-marionette-driver-selenium.html>
- indeed. (09 de 01 de 2020). *indeed*. Obtenido de Salarios para empleos de Desarrollador/a de software en España: <https://www.indeed.es/salaries/desarrollador-de-software-Salaries>
- Instituto Tecnológico de Toluca. (14 de 01 de 2019). *La Calidad del Software y los Procesos de Testing, Guías, Proyectos, Investigaciones de Informática General*. Obtenido de <https://www.docsity.com/es/la-calidad-del-software-y-los-procesos-de-testing/4391525/>
- Jiménez Capel, M. Y. (2015). *Bases de datos relacionales y modelado de datos*. Málaga: ic Editorial.
- Krall, C. (01 de 10 de 2019). *aprenderaprogramar.com*. Obtenido de Versiones CSS. Algo de historia y perspectiva. ¿Qué es el W3C? Recomendaciones oficiales: [https://www.aprenderaprogramar.com/index.php?option=com\\_content&view=article&id=726:versiones-css-algo-de-historia-y-perspectiva-ique-es-el-w3c-recomendaciones-oficiales-cu01022d&catid=75&Itemid=203](https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=726:versiones-css-algo-de-historia-y-perspectiva-ique-es-el-w3c-recomendaciones-oficiales-cu01022d&catid=75&Itemid=203)
- Kuhlman, D. (2 de 12 de 2019). *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. Obtenido de [https://web.archive.org/web/20120623165941/http://cutter.rexx.com/~dkuhlman/python\\_book\\_01.html](https://web.archive.org/web/20120623165941/http://cutter.rexx.com/~dkuhlman/python_book_01.html)
- Microsoft. (15 de 11 de 2019). *Structured Query Language (SQL)*. Obtenido de <https://docs.microsoft.com/en-us/sql/odbc/reference/structured-query-language-sql?redirectedfrom=MSDN&view=sql-server-ver15>
- Mozilla Source Tree Docs. (2 de 12 de 2019). *Introduction to Marionette*. Obtenido de <https://firefox-source-docs.mozilla.org/testing/marionette/Intro.html>
- Negrino, T., & Smith, D. (2005). *Guía de aprendizaje Java Script*. Madrid: Pearson, Pentice Hall.

- OpenSTA. (09 de 01 de 2020). *opensta.org*. Obtenido de <http://opensta.org/>
- PCQuest. (15 de 09 de 2019). *A free local WordPress Developments Environments*. Obtenido de <https://www.pcquest.com/4-local-wordpress-development-environments/>
- Pérez Juárez, M. Á. (2017). *Ingeniería de Sistemas de Software*. Valladolid, España: Universidad de Valladolid.
- phpMyAdmin. (15 de 09 de 2019). *phpMyAdmin*. Obtenido de [https://web.archive.org/web/20130216142335/http://www.phpmyadmin.net/home\\_page/index.php](https://web.archive.org/web/20130216142335/http://www.phpmyadmin.net/home_page/index.php)
- Pimentel, V. (30 de 06 de 2009). *nobbot*. Obtenido de Tecnología para las personas.: <https://www.nobbot.com/personas/las-novedades-de-html5-y-iv/>
- Reese, G., Jay Yarger, R., King, T., & Williams, H. E. (2002). *Managing and Using MySQL: Open source SQL database for managing information and Web sites*. O'Reilly Media, Inc.
- Robbins, J. (2012). *Learning web design: A beginner's guide to HTML, CSS, JavaScript and web graphics*. O'Reilly Media, Inc.
- Rockoff, L. (2017). *The language of SQL*. USA: Pearson Education, Inc.
- Sawyer McFarland, D. (2008). *JavaScript*. Madrid: Anaya.
- Selenium, T. (15 de 10 de 2019). *Tutorial Selenium*. Obtenido de <https://www.tutorialselenium.com/selenium-ide/>
- SeleniumHQ. (28 de 10 de 2019). *SeleniumHQ.org*. Obtenido de <https://www.seleniumhq.org/>
- SeleniumHQ. (15 de 10 de 2019). *SelenoumHQ Browser Automation*. Obtenido de <https://www.seleniumhq.org/>
- Stewart, S. (15 de 10 de 2019). *The Architecture of Open Source Applications: Selenium WebDriver*. Obtenido de <http://www.aosabook.org/en/selenium.html>
- Suehring, S. (2008). *JavaScript, Paso a paso*. Madrid: Grupo Anaya.
- Thomson, L., & Welling, L. (2017). *Desarrollo Web con PHP y MySQL*. Madrid: Anaya.
- Toledo Rodríguez, F. (2014). *Introducción a las Pruebas de Sistemas de Información*. Montevideo, Uruguay.

Tuya, J. (16 de marzo de 2017). *Universidad de Oviedo*. Obtenido de <https://in2test.lsi.uniovi.es/gt26/presentations/Tuya-ISO29119-Seguridad-20170316.pdf>

Ullman, L. (2010). *PHP, Paso a paso*. Madrid: Anaya.

*Unidade de Apoyo para el Aprendizaje*. (15 de 11 de 2019). Obtenido de [https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/872/mod\\_resource/content/1/contenido/index.html](https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/872/mod_resource/content/1/contenido/index.html)

w3. (20 de 09 de 2019). *HTML: The Markup Language (an HTML language reference)*. . Obtenido de <https://www.w3.org/TR/html/syntax.html#doctype-syntax>

W3C. (25 de 09 de 2019). *W3C*. Obtenido de What is CSS?: <https://www.w3.org/standards/webdesign/htmlcss#whatcss>

w3schools. (16 de 3 de 2015). *w3schools*. Obtenido de HTML Elements: [https://www.w3schools.com/html/html\\_elements.asp](https://www.w3schools.com/html/html_elements.asp)

WHATWG (Apple, G. M. (28 de 11 de 2019). *HTML: Living Standard*. Obtenido de <https://html.spec.whatwg.org/>

Wium Lie, H., & Bos, B. (13 de 09 de 2018). *W3C*. Obtenido de Cascading Style Sheets, level 1: <https://www.w3.org/TR/CSS1/>

## ANEXO I: Funciones útiles para el manejo de Selenium con Python

- **driver.get(URL)**: permite abrir una URL en el navegador de internet.
- **element = driver.find\_element\_by\_id("id")**: permite encontrar elementos del fichero HTML a través de su identificador.
- **element = driver.find\_element\_by\_name("name")**: permite encontrar elementos del fichero HTML a través de su nombre.
- **element = driver.find\_element\_by\_partial\_link\_text("link")**: permite encontrar elementos del fichero HTML a través de un enlace parcial.
- **element.send\_keys("datos")**: permite introducir texto en campos de texto del fichero HTML.
- **element.clear()**: permite poner a cero un campo de texto. Es útil cuando se quiere editar el contenido de algún campo de texto, pero antes se quiere eliminar el contenido actual.
- **driver.find\_element\_by\_id("submit").click()**: permite localizar un elemento (por su id o por su nombre) y hacer click sobre él. Es especialmente útil para hacer click sobre botones.
- **time.sleep(n)**: introduce una pausa de n segundos.
- **self.assertIn("Title", driver.title)**: permite comprobar que el título de la página coincide con el indicado.