Classroom Questionnaires

Memoria de Proyecto Final de Máster

Desarrollo de Sitios y Aplicaciones Web

Autor: José Carlos García Bermúdez

Consultor: Carlos Caballero González

Profesor: César Pablo Córcoles Briongos

6 de Enero de 2020

A Silvia, por su apoyo constante.

A Diego y Rebeca, por todo el tiempo que les he robado.

"Cada hora, cada minuto, cada segundo, está por escribir, y sobre todo… está por vivir."

José Sacristán

Abstract

El proyecto consiste en el desarrollo de una aplicación web, donde los usuarios que se registren podrán cumplimentar cuestionarios para evaluar sus conocimientos. Los cuestionarios están formados por preguntas de tipo test, agrupadas por materias y grupos.

En función del perfil asignado, el usuario podrá proponer preguntas, que una vez revisadas formarán parte de los cuestionarios que se generen. El objetivo de la aplicación es que pueda ser utilizada en el aula por el alumnado, trabajando de forma colaborativa, para afianzar sus conocimientos.

La aplicación se ha desarrollado con el framework de código abierto Laravel, que utiliza PHP para la implementación del código y MySQL como gestor de base de datos.

Palabras clave:

Trabajo Final de Máster, Cuestionarios, Preguntas, Test, Alumnado, Aula, Laravel, PHP, MySQL

Abstract (English Version)

The project consists in the development of a web application, where register users can fill in questionnaires to evaluate their knowledge. The questionnaires are made up of test questions, grouped by subjects and units.

Depending on the assigned profile, the user may propose questions, which once reviewed will be part of the generated questionnaires. The purpose of the application is that it can be used in the classroom by students, working collaboratively, to strengthen their knowledge.

The application has been developed with the Laravel open source framework, which uses PHP for code implementation and MySQL as database manager.

Keywords:

Final Master's Work, Questionnaires, Questions, Test, Students, Classroom, Laravel, PHP, MySQL

Índice

Figuras y Tablas

Índice de Figuras

Índice de Tablas

1. Introducción/Prefacio

La idea de este proyecto surge ante mi necesidad como profesor de preparar cuestionarios que permitan al alumnado reforzar su aprendizaje.

En el mercado hay plataformas gratuitas, como Kahoot!, que permiten al profesor la creación de cuestionarios, con el que los alumnos y alumnas compiten en ver quién responde más rápido a las preguntas propuestas.

La justificación funcional de mi proyecto surge tras el uso de herramientas como Kahoot!, donde el profesorado crea unos cuestionarios con los que el alumnado compite para ver quién responde más rápido, lo que me ha hecho ver la necesidad de este proyecto por los siguientes motivos:

- En primer lugar, considero que en algunos casos es contraproducente que se premie la rapidez en la resolución de cada pregunta, ya que la realidad del aula es muy diversa, y es habitual encontrar alumnado que requiere de adaptaciones curriculares que le permitan disponer de más tiempo a la hora de realizar sus tareas.
- Por otro lado, pretendo que sea una herramienta colaborativa, donde el alumnado participe elaborando preguntas de las diferentes unidades del curso, que serán incorporadas a los cuestionarios que realicen el resto de compañeros.

En cuanto a la justificación técnica, he decidido desarrollar este proyecto Laravel por los siguientes motivos:

- El uso de un framework de desarrollo nos proporciona una base estandarizada sobre la que iniciar el desarrollo, con la consiguiente reducción de tiempos de desarrollo y mantenimiento.
- Utiliza PHP para la implementación del código, uno de los lenguajes más utilizados en el desarrollo web.
- Aplica el patrón Modelo-Vista-Controlador, lo que facilita la organización del código.
- Facilita el manejo de los datos mediante Eloquent, por lo que la interacción con las bases de datos es totalmente orientada a objetos, siendo compatible con la gran mayoría de bases de datos del mercado.
- Es modular, con un amplio sistema de paquetes para extender la funcionalidad de forma fácil, robusta y segura.
- Facilita el manejo de las rutas de las aplicaciones, así como la generación de URLs amigables.
- Incluye el sistema de plantillas Blade, que mejoran el aspecto de la aplicación e incluyen un sistema de caché que las hace más rápidas, lo que mejora el rendimiento de la aplicación.
- Reduce la curva de aprendizaje, en comparación con otros frameworks PHP.
- Hay disponible una abundante documentación.

2. Objetivos

El principal objetivo es la elaboración de una herramienta de trabajo que permita al alumnado participar en su proceso de aprendizaje: creando preguntas, familiarizándose con un entorno de trabajo en grupo, afianzando y autoevaluando sus conocimientos.

3. Contenidos

El listado detallado de las funcionalidades que incluirá la aplicación es el siguiente:

- Registro de Usuarios. El registro del usuario (usuario Estándar) le permitirá el acceso a la aplicación.
- Acceso al Sistema. Mediante usuario/contraseña, el usuario podrá acceder al sistema a la funcionalidad que tenga disponible en función de su perfil.
- Perfil del Usuario. Cada usuario podrá editar su información de registro, y consultar el histórico de cuestionarios realizados.
- Generación de Cuestionarios. Cualquier usuario registrado podrá realizar un cuestionario. Para generarlo, se seleccionará aleatoriamente un conjunto de preguntas en base a los parámetros indicados (Materia y Grupo). Una vez completado, el resultado será registrado en el historial del usuario.
- Gestión de Preguntas. El usuario Editor podrá crear nuevas preguntas, que estarán asignadas a una Materia y Grupo. Cada pregunta tendrá un texto, imagen (opcional) y cuatro opciones (sólo una será la correcta). Las preguntas se crean en estado 'Propuesta', y no serán incorporadas a los cuestionarios hasta que hayan sido aprobadas en su revisión.
- Revisión de Preguntas. El usuario Revisor validará las preguntas propuestas, y las aprobará o rechazará.
- Gestión de Materias. El Administrador podrá crear, modificar y borrar las materias, que son un primer nivel de agrupación de preguntas.
- Gestión de Grupos. Para cada Materia, el Administrador podrá crear, modificar y borrar grupos, que son un segundo nivel de agrupación de preguntas.
- Gestión de Usuarios. El usuario Administrador podrá asignar a los usuarios registrados los permisos de Editor, Revisor o Administrador.

4. Metodología

El proyecto tiene un **enfoque profesionalizador**, donde como profesional en formación mejorar las competencias iniciales cumpliendo distintas etapas de formación y desarrollo de conocimientos y habilidades.

Se aplicará una **metodología iterativa e incremental**, donde se solaparán diferentes fases de desarrollo (en lugar de llevar a cabo una planificación secuencial o de cascada), se planificarán pequeños bloques del proyecto, y se irá revisando y mejorando lo anterior.

5. Arquitectura de la Aplicación

Al tratarse de una aplicación web, este proyecto tiene una arquitectura cliente/servidor, donde la aplicación web:

- Es proporcionada por un servidor web.
- Es utilizada por los usuarios a través de navegadores.

El servidor web genera de forma dinámica las páginas que los clientes solicitan, para ello utiliza la información almacenada en una base de datos.

Esta aplicación utiliza una arquitectura cliente/servidor basada en un modelo de tres capas:

- Capa de presentación (cliente)
  - ✓ Recoge la información del usuario
  - ✓ Envía información a la capa de proceso
  - ✓ Recibe los resultados de la capa de proceso
  - ✓ Genera la presentación para el usuario
- Capa de proceso (servidor web)
  - ✓ Recibe la entrada de datos de la capa de presentación
  - ✓ Interactúa con la capa de datos para realizar operaciones
  - ✓ Envía los resultados procesados a la capa de presentación
- Capa de datos (servidor de base de datos)
  - ✓ Mantiene y recupera los datos
  - ✓ Asegura su integridad

Patrón de Diseño

Para el desarrollo de la aplicación se utilizará el Framework Laravel, que implementa el **patrón de diseño MVC**.

Este patrón MVC divide las aplicaciones en tres niveles de abstracción:

- **Modelo**: Responsable de la comunicación con la base de datos.
- **Vista**: Encargada de mostrar la información al usuario.
- **Controlador**: Intermediario entre la vista y el modelo.

El funcionamiento básico del patrón MVC es el siguiente:

- El usuario realiza una petición.
- El controlador captura la petición, y hace la llamada al modelo correspondiente.
- El modelo se encarga de interactuar con la base de datos.
- El controlador recibe la información y la envía a la vista.
- La vista muestra la información.

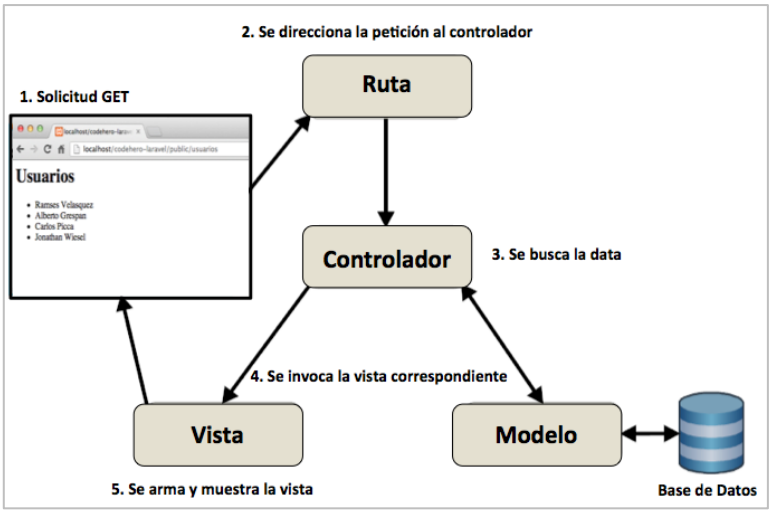

Figura 1. Esquema patrón de diseño MVC.

6. Plataforma de Desarrollo

Recursos

Los recursos tecnológicos que estoy utilizando para el desarrollo del proyecto son:

Hardware

Las características del equipo que estoy utilizando para el desarrollo de esta aplicación son:

- Ordenador portátil Acer Aspire E1-572G
- Procesador Intel Core i7-4500U 1,8Ghz
- Memoria RAM 8 GB DDR3
- Gráfica AMD Radeon con 1GB VRAM

Software

El software utilizado es:

- Sistema operativos Windows 10 Professional
- IDE JetBrains PhpStorm 2019.1.3
- Servidor XAMPP 7.1.30, donde utilizo:
  - ✓ Servidor web Apache
  - ✓ Servidor de bases de datos MySQL
  - ✓ Administración de bases de datos phpMyAdmin
- Framework Laravel 5.8.18

Preparación del Entorno

Framework Laravel

Para la instalación de Laravel se ha seguido la guía oficial de instalación [1].

Laravel utiliza Composer para gestionar sus dependencias. Para la instalación de Composer en Windows, lo más simple es utilizar el instalador 'Composer-Setup.exe', que se puede descargar de la página:

https://getcomposer.org/download/

Para instalar Laravel utilizando Composer, se debe ejecutar desde consola el comando:

composer global require laravel/installer

Posteriormente se creará un nuevo proyecto de Laravel (por ejemplo 'quest'), ejecutando el comando:

laravel new quest

Para crear las plantillas de registro e inicio de sesión, así como definir las rutas para que apunten al controlador de autenticación, hay que ejecutar el comando:

php artisan make:auth

Para usar el servidor de desarrollo de PHP de la instalación local, sobre el directorio del proyecto se ejecuta:

php artisan serve

Con lo que se inicia el servidor local, en el puerto 8000, y ya estará accesible la aplicación por defecto.

http://localhost:8000

Base de Datos MySQL

Para la instalación de la base de datos MySQL se utilizará el servidor XAMPP [2], que además incluye el servidor web Apache para la administración de la base de datos a través de phpMyAdmin. El panel de control de XAMPP permite iniciar ambos servicios.

En MySQL hay que crear la base de datos 'quest', que contendrá las tablas y relaciones de la aplicación.

Configuración Inicial

En el archivo de configuración '.env' de Laravel hay que configurar los parámetros de acceso a la base datos.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=quest
DB_USERNAME=root
DB_PASSWORD=null
```

Laravel dispone de un mecanismo propio, denominado migraciones [3], que permite la creación y actualización de las tablas de la base de datos sin necesidad de utilizar otras herramientas.

Para crear el archivo que me permitirá definir la estructura de cada tabla (por ejemplo 'materias'), se ejecutará:

php artisan make:migration create_materias_table --create=materias

Con ello se habrá creado el archivo 'create_materias_table.php' en el directorio 'database/migrations' del proyecto, donde se definirán los campos de la tabla 'materias'.

```
Schema::create( table: 'materias', function (Blueprint $table) {
    $table->bigIncrements( column: 'id');
    $table->string( column: 'materia', length: 100);
    $table->string( column: 'imagen', length: 50)->nullable();
    $table->timestamps();
});
```

Una vez creados los archivos de todas las tablas, la creación se realizará ejecutando el comando:

php artisan migrate

7. Planificación

Para la planificación del proyecto he tomado como base las PECs que se realizarán durante el curso. Más adelante se planificarán entregas intermedias si se estima necesario, pero manteniendo los hitos principales, que se detallan a continuación:

PEC 1. 1 de octubre de 2019

En esta primera entrega se realizará la definición formal del proyecto:

- Contexto.
- Justificación.
- Objetivos.
- Metodología.
- Planificación.
- Funcionalidad a cubrir.

PEC 2. 30 de octubre de 2019

En esta segunda fase se realizarán las siguientes tareas:

- Revisión de requisitos.
- Selección de herramientas.
- Diseño del modelo de base de datos.
- Diseño de la interfaz de usuario.
- Definición del flujo de navegación.
- Consolidación de los métodos de trabajo.
- Inicio del desarrollo.

PEC 3. 8 de diciembre de 2019

En esta tercera fase se realizará la mayor parte de la implementación y prueba de la aplicación, que tendrá como resultado el lanzamiento de la versión beta.

Entrega Final. 6 de enero de 2019

En esta última fase se finalizará el proyecto, por lo que se realizarán las siguientes tareas:

- Completar las pruebas.
- Realizar las correcciones y mejoras necesarias.
- Entrega final de la aplicación.
- Entrega final de la documentación correspondiente.

8. Proceso de Trabajo/Desarrollo

En una **primera fase** se realiza la definición de los objetivos principales del proyecto, tomando como base la idea de realizar una aplicación que facilite la participación del alumnado en la creación de un repositorio compartido de preguntas para chequear los conocimientos adquiridos a lo largo de cada unidad.

Para la implementación del proyecto, y teniendo en cuenta que la aplicación debería ser accedida por los usuarios desde diferentes dispositivos sin necesidad de realizar ningún tipo de instalación, vi que una aplicación web era la mejor opción.

A la hora de elegir el lenguaje a utilizar, herramientas, entorno de desarrollo, etc., de las diferentes alternativas que hemos visto a lo largo del máster, me decidí por el Framework Laravel ya que estructuraba muy bien el código y facilitaba su desarrollo.

En una **segunda fase** he hecho la revisión de los requisitos del proyecto, eliminando algunos que me resultaban demasiado ambiciosos (teniendo en cuenta los plazos con los que contamos), e introduciendo otros nuevos que podrían introducir mayor vistosidad a la aplicación (por ejemplo, que el usuario pueda adjuntar una imagen con cada pregunta).

En esta segunda fase también he seleccionado las herramientas a utilizar (como el IDE PhpStorm), he realizado el diseño de la base de datos, he buscado la solución técnica a determinados problemas (como la gestión de imágenes, comportamiento de la aplicación a diferente tipo de dispositivos…), y se ha iniciado el desarrollo de algunos mantenimientos que servirán de base para el resto de la aplicación.

En la **tercera fase** se realiza la mayor parte de la implementación y prueba de la aplicación, teniendo como resultado el lanzamiento de la versión beta.

Finalmente, en una **cuarta fase** se finaliza el proyecto, completando las pruebas y realizando las correcciones y mejoras necesarias. Se realiza el despliegue de la aplicación y la entrega de la documentación correspondiente.

9. Diagramas UML

Modelo Relacional (Base de Datos)

El siguiente diagrama muestra la estructura de la base de datos, incluyendo sus tablas y relaciones.

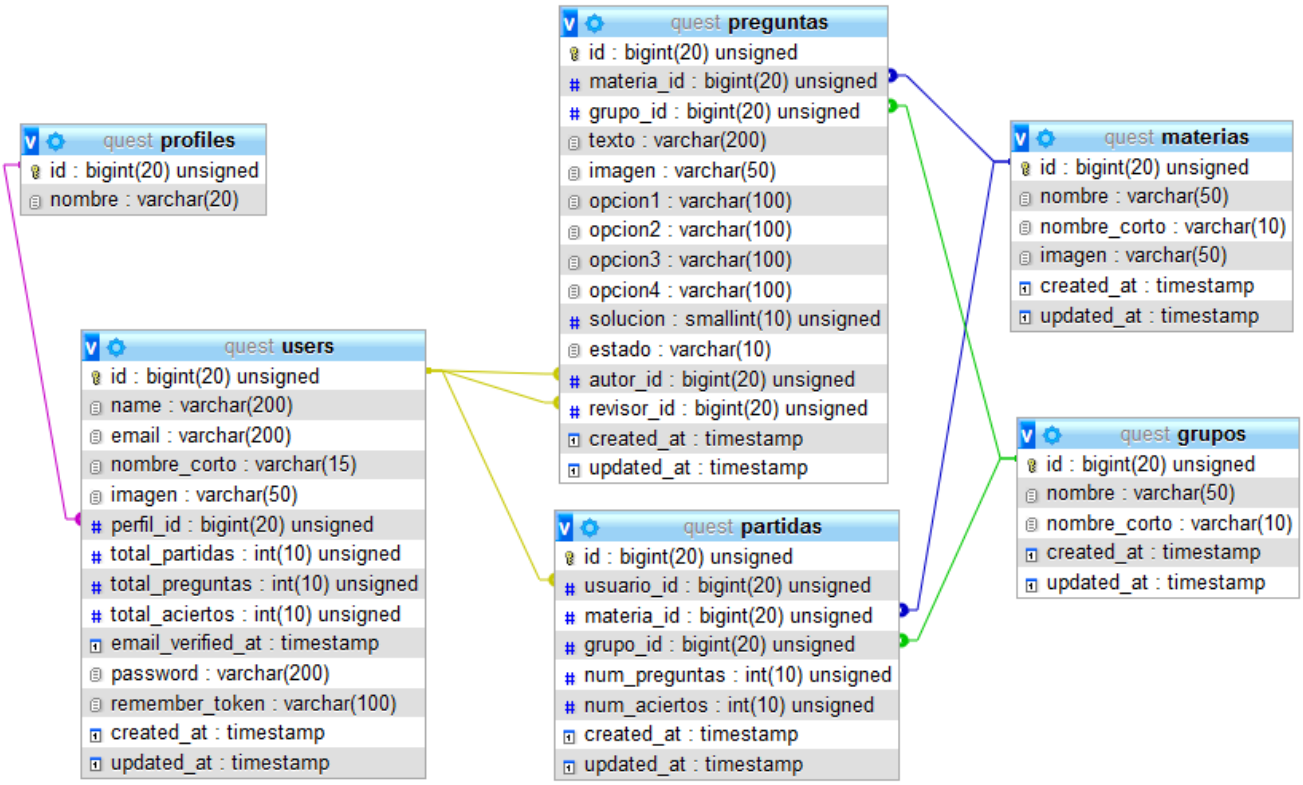

Figura 2. Diagrama Base de Datos. Modelo Relacional

Casos de Uso (Perfiles Usuarios)

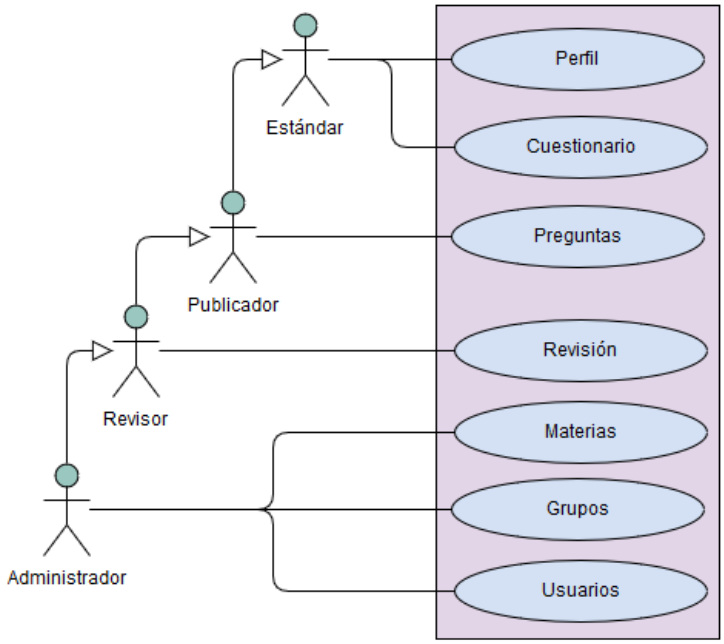

Figura 3. Diagrama de Casos de Uso. Perfiles de Usuarios.

Diagrama de Estados (Preguntas)

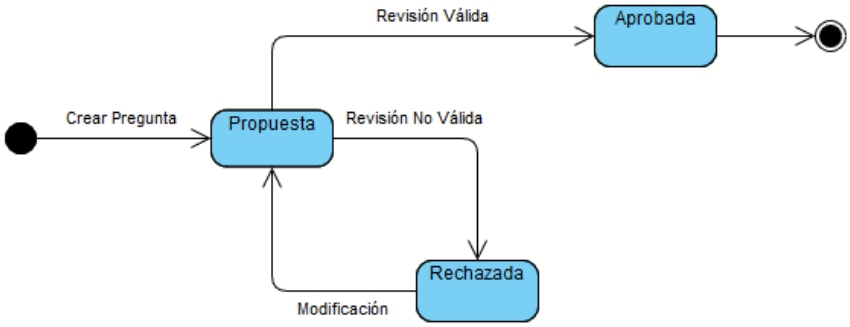

Figura 4. Diagrama de Estados. Preguntas.

10. Prototipos

A continuación se muestran los prototipos Hi-Fi creados durante el desarrollo de la aplicación.

Debido a la metodología que estoy utilizando, iterativa e incremental, éstos han ido sufriendo diversas variaciones, por lo que aquí se muestra lo que se podría considerar una primera versión de la aplicación.

Para su desarrollo no se ha utilizado ninguna herramienta de prototipado, sino que las capturas corresponden al propio desarrollo de la aplicación, ya que se están solapando las etapas de diseño e implementación.

Inicio

Si el usuario no está registrado, en la página de inicio solo podrá hacer login o registrarse.

Figura 5. Página de inicio. Usuario no identificado.

Registro

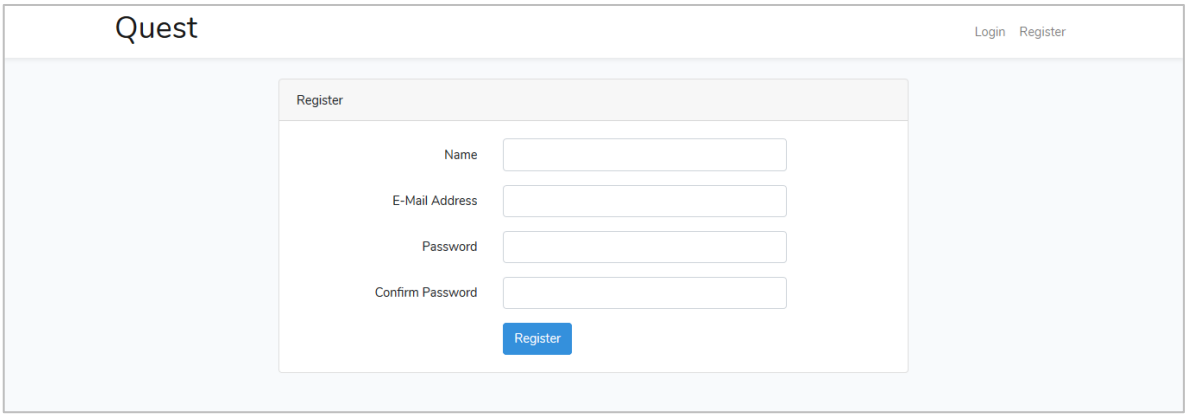

Figura 6. Página de registro.

Login

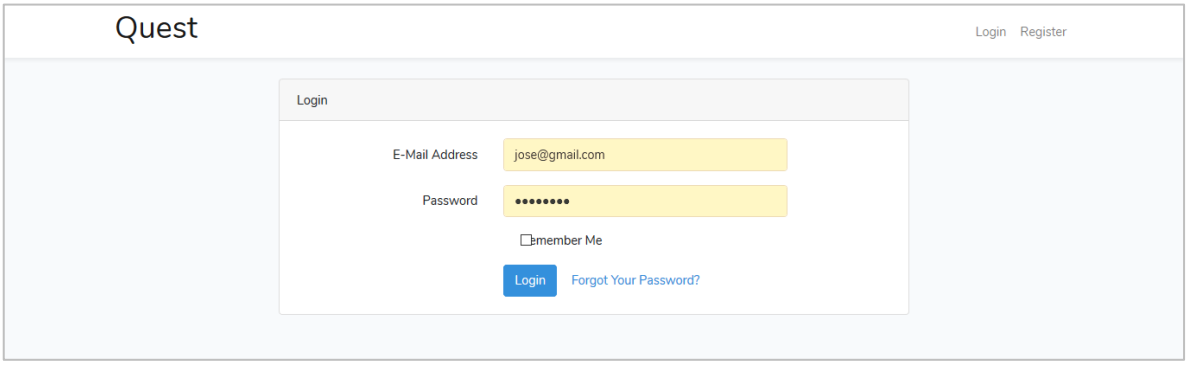

Figura 7. Página de login.

Usuario Identificado

Una vez que el usuario se ha identificado, en función de su perfil, el menú mostrará las opciones que tendrá disponibles. Se muestran las correspondientes al 'Administrador' que tendrá acceso a toda la funcionalidad, en el siguiente apartado (perfiles de usuario) se mostrará en detalle la funcionalidad disponible para cada perfil.

Cualquier usuario que se haya identificado podrá editar sus datos personales y consultar el listado de sus partidas, pudiendo acceder respectivamente a las páginas de edición de usuario e histórico de partidas del usuario, que se detallan más adelante.

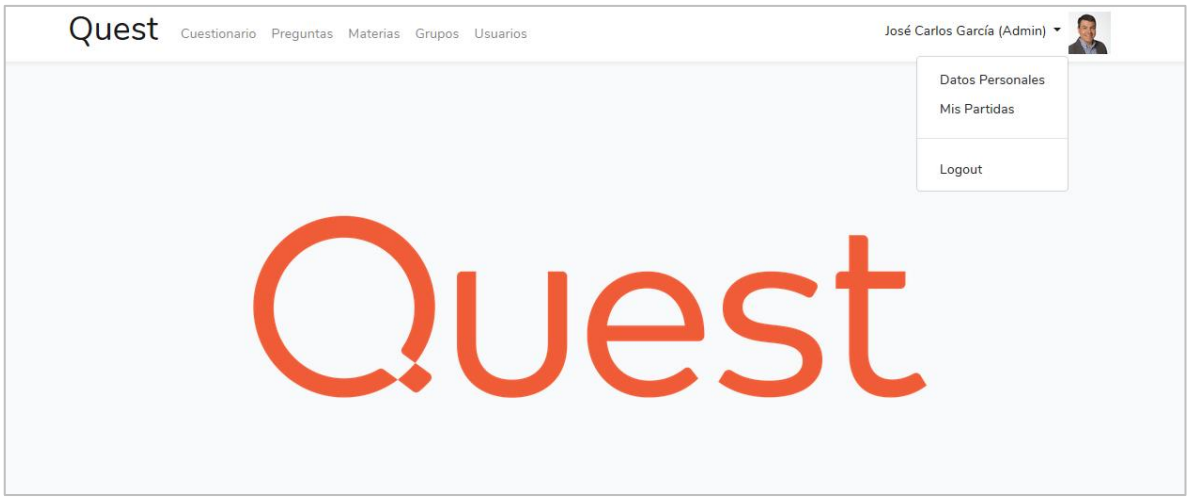

Figura 8. Página de inicio. Usuario identificado.

Cuestionario

Inicio

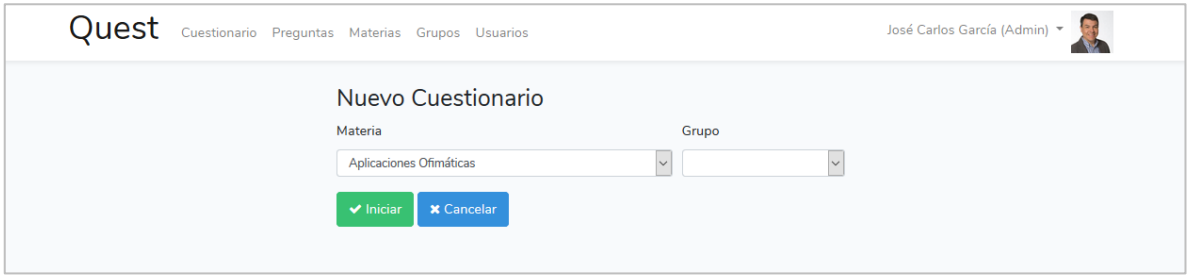

Figura 9. Página Cuestionario. Inicio.

Pregunta

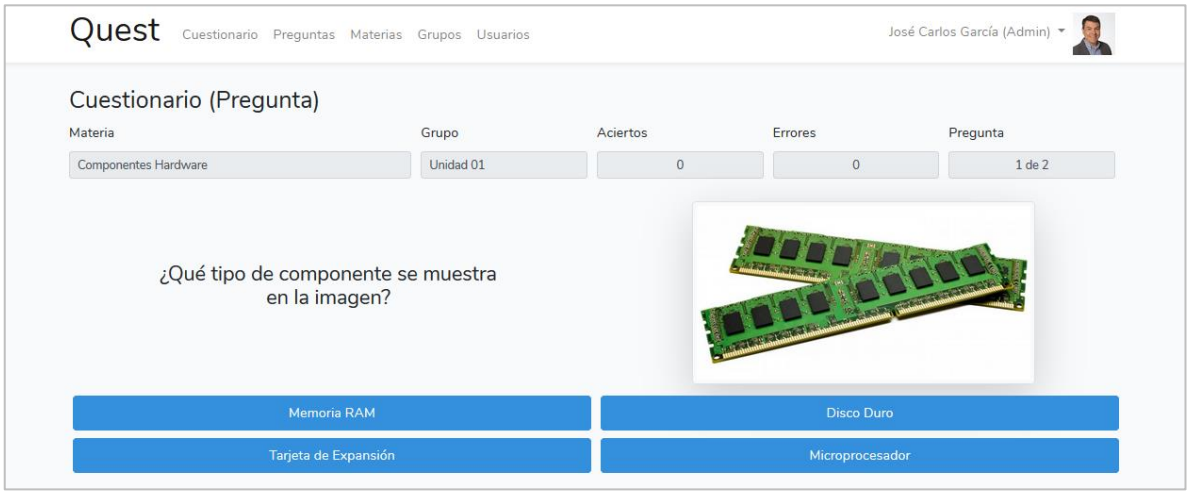

Figura 10. Página Cuestionario. Pregunta.

Respuesta Correcta

Figura 11. Página Cuestionario. Respuesta correcta.

Respuesta Errónea

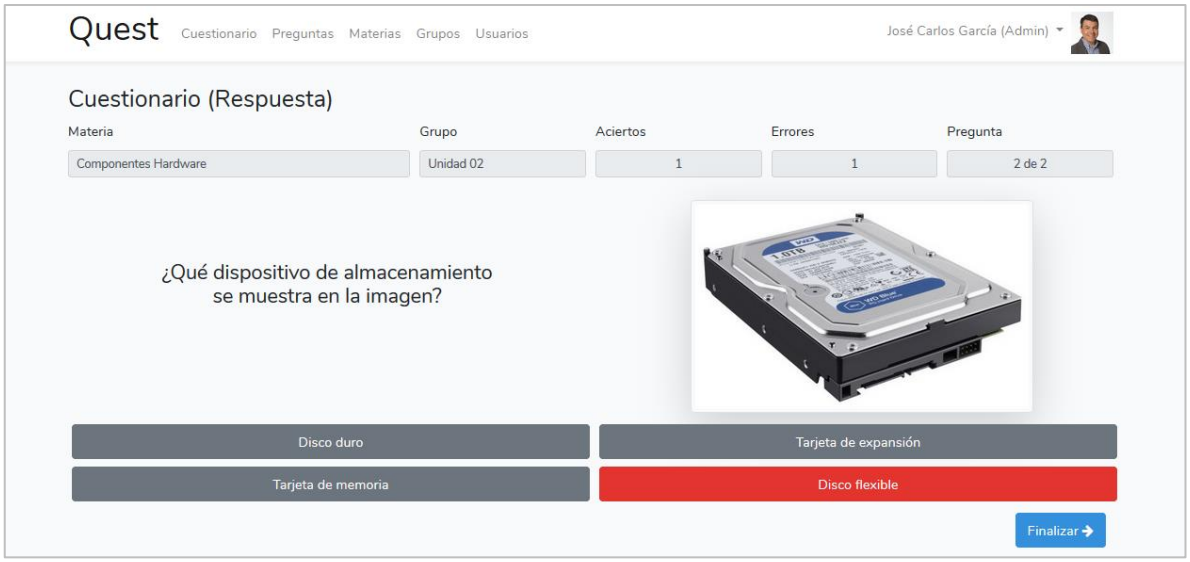

Figura 12. Página Cuestionario. Respuesta errónea.

Resultado

Figura 13. Página Cuestionario. Resultado final.

Preguntas

Listado

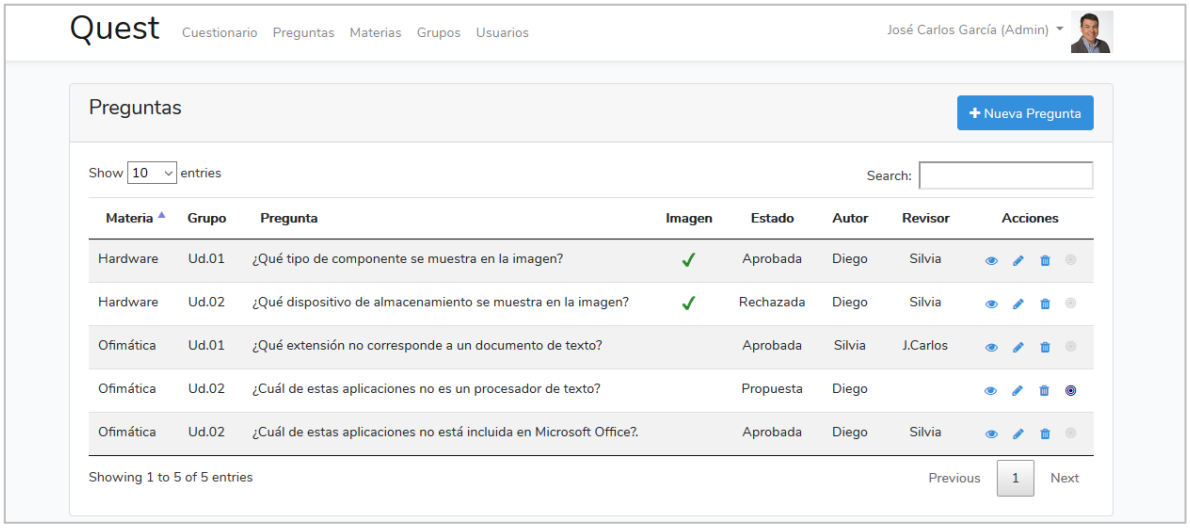

Figura 14. Gestión de Preguntas. Página de Listado.

Detalle

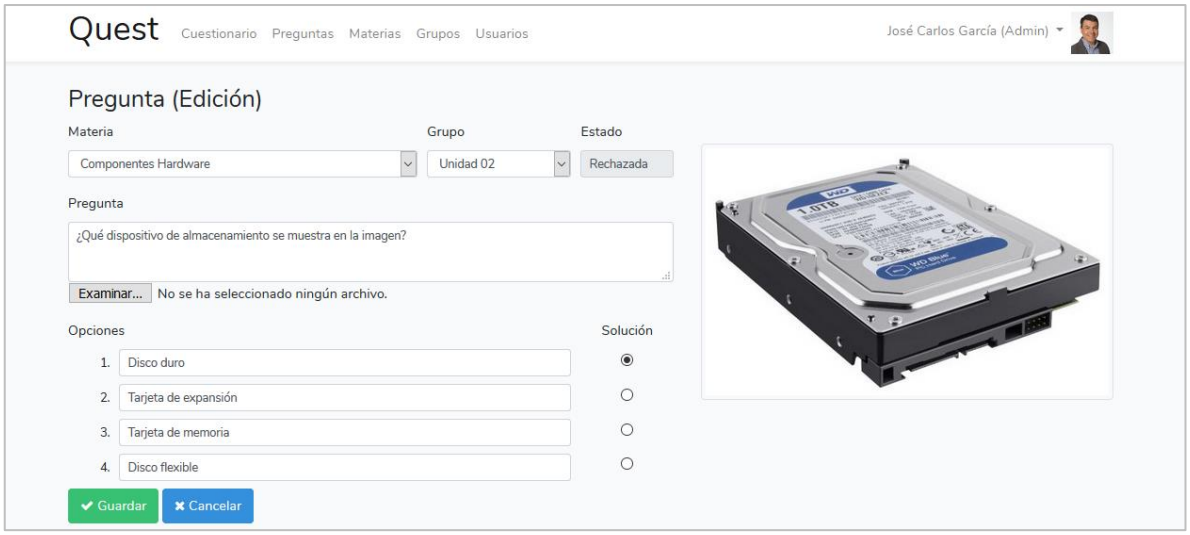

Figura 15. Gestión de Preguntas. Página de Detalle.

Revisión

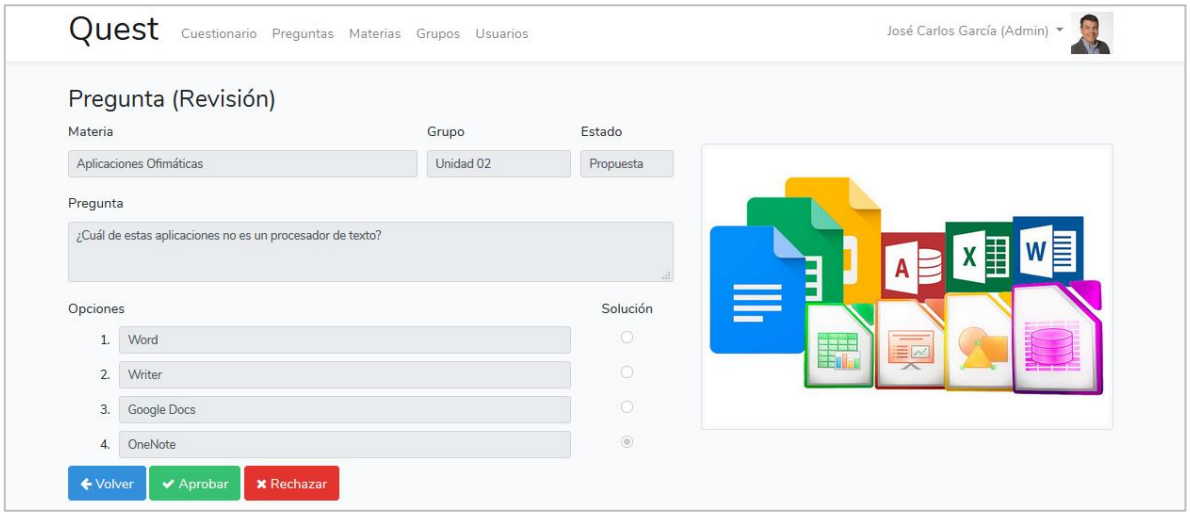

Figura 16. Gestión de Preguntas. Página de Revisión.

Materias

Listado

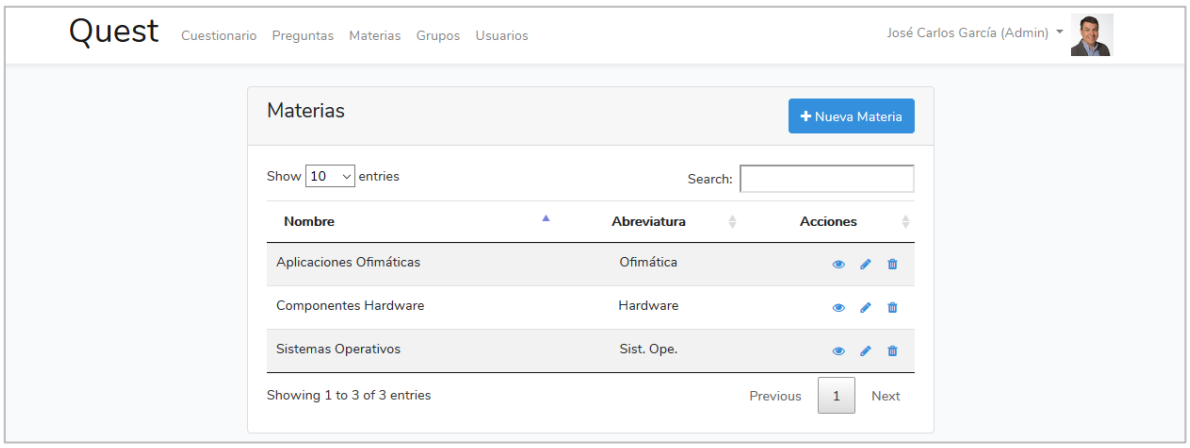

Figura 17. Gestión de Materias. Página de Listado.

Detalle

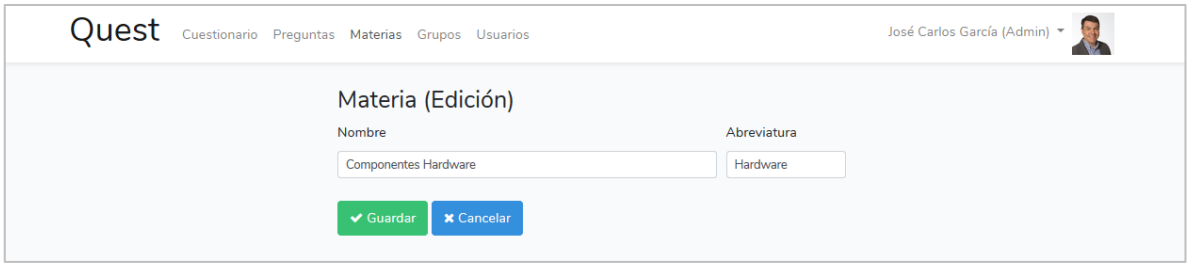

Figura 18. Gestión de Materias. Página de Detalle.

Grupos

Listado

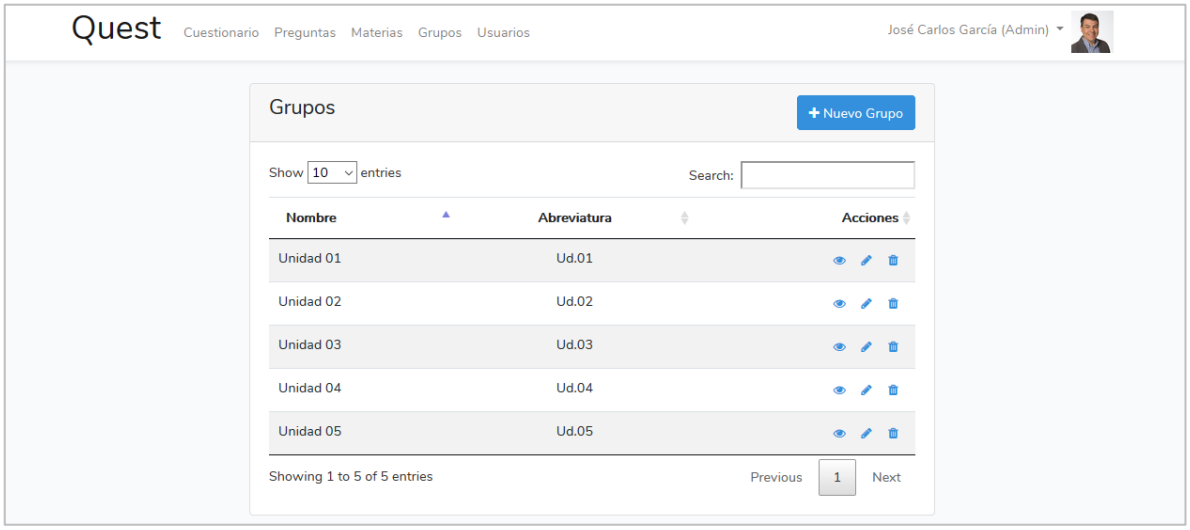

Figura 19. Gestión de Grupos. Página de Listado.

Detalle

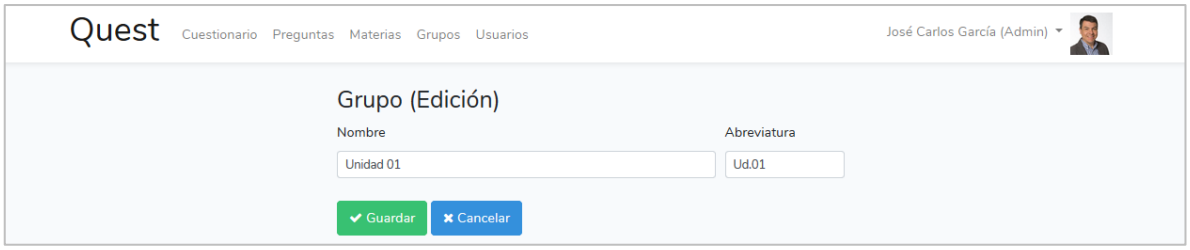

Figura 20. Gestión de Grupos. Página de Detalle.

Usuarios

Listado

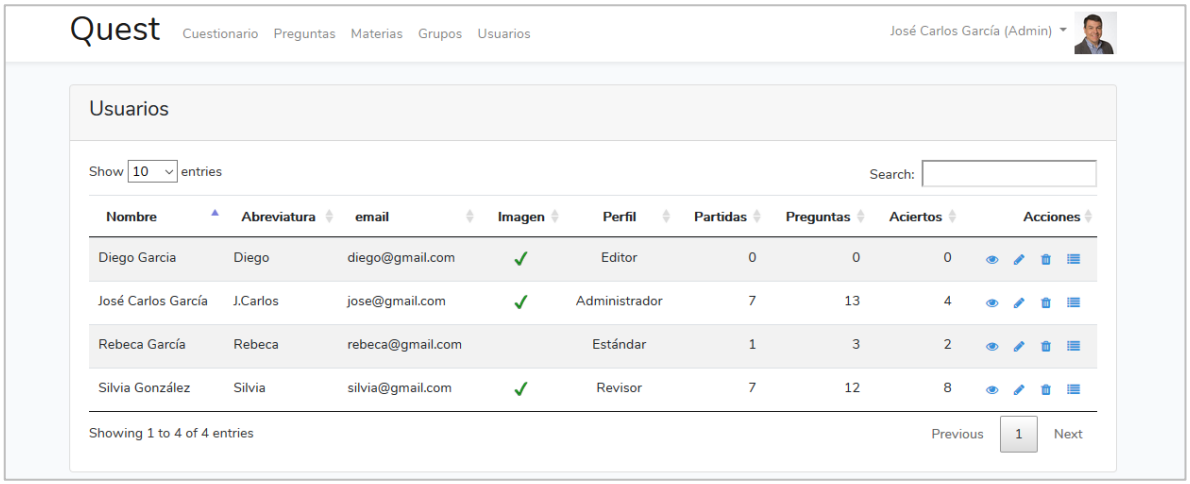

Figura 21. Gestión de Usuarios. Página de Listado.

Detalle

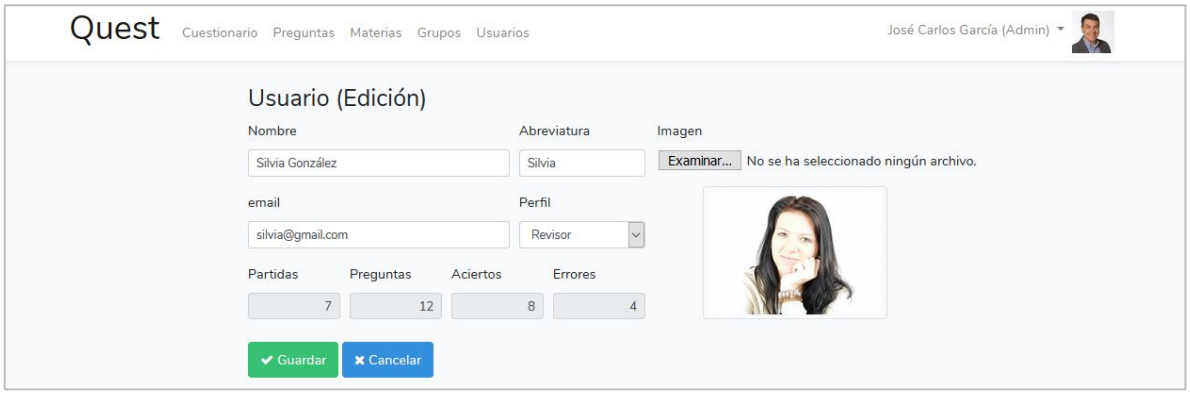

Figura 22. Gestión de Usuarios. Página de Detalle.

Histórico de Partidas

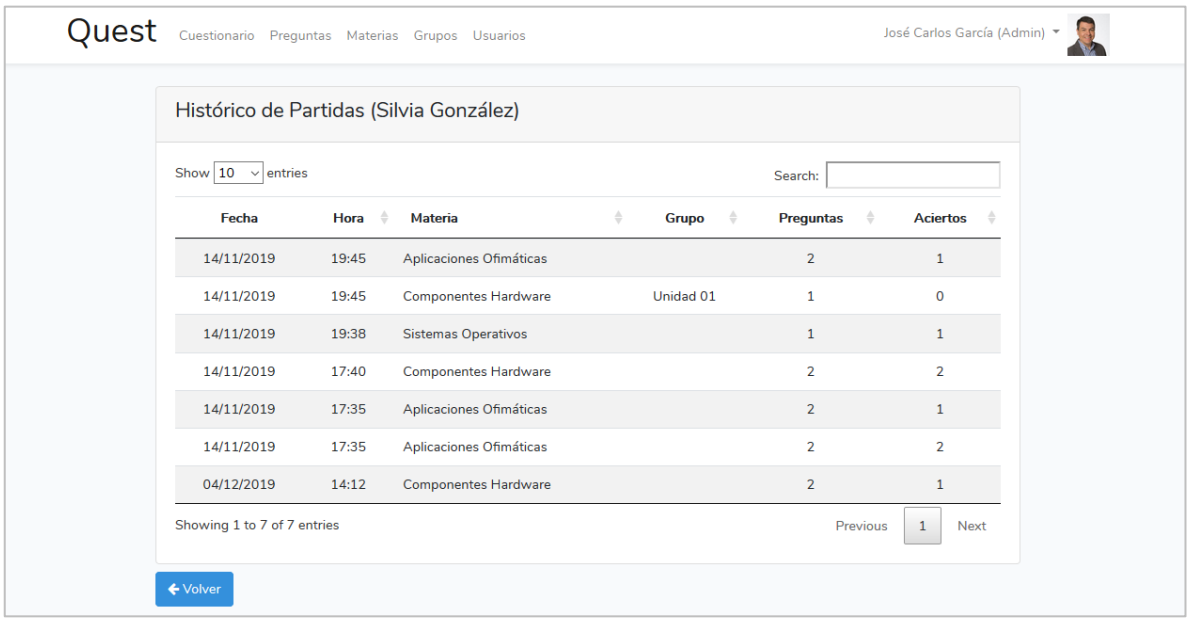

Figura 23. Gestión de Usuarios. Página de Histórico de Partidas.

11. Perfiles de Usuario

Uno de los objetivos de la aplicación es que pueda ser utilizada en el aula por el alumnado, trabajando de forma colaborativa; por tanto, determinados usuarios podrán proponer preguntas, que una vez revisadas formarán parte de los cuestionarios que se generen.

Para implementar este comportamiento, los usuarios de la aplicación deben registrarse previamente, y según se detalla en el diagrama de casos de uso, se les asignará uno de los siguientes perfiles:

- **Estándar**: Es el perfil por defecto de cualquier usuario registrado, que le permitirá la generación de cuestionarios.
- **Editor**: Podrá crear nuevas preguntas, en estado 'Propuesta', que estarán pendientes de revisión. Sólo tendrá acceso a sus propias preguntas, que podrá modificar o borrar si no están aprobadas.
- **Revisor**: Además de las opciones anteriores, el revisor tendrá acceso a las preguntas de todos los usuarios. El revisor tendrá disponible la opción 'Revisar', que estará activa para las preguntas en estado 'Propuesta', y que le permitirá pasarlas al estado 'Aprobada' o 'Rechazada'. Sólo las preguntas aprobadas serán incluidas en los cuestionarios.
- **Administrador**: Además de las opciones anteriores, el administrador es el único que tendrá acceso a la gestión de materias, grupos y usuarios.

Los perfiles son acumulativos, es decir, el Revisor también es Editor, el Administrador también es Revisor y Editor, y todos los perfiles incluyen los permisos del usuario Estándar.

Usuario No Registrado

En la página inicial tendrá que hacer login o registrarse, pero no tendrá disponible ninguna otra funcionalidad.

Figura 24. Menú usuario no registrado.

Usuario Estándar

Si el usuario que ha accedido a la aplicación tiene perfil 'Estándar', en el menú de la aplicación tendrá disponible la opción de realizar cuestionarios.

Figura 25. Menú usuario 'Estándar'.

Usuario Editor

Si el usuario que ha accedido a la aplicación tiene perfil 'Editor', en el menú de la aplicación tendrá además la opción de acceder a la gestión de preguntas.

Figura 26. Menú usuario 'Editor'.

El editor sólo tendrá acceso a sus propias preguntas, que no podrá modificar o borrar si ya están aprobadas.

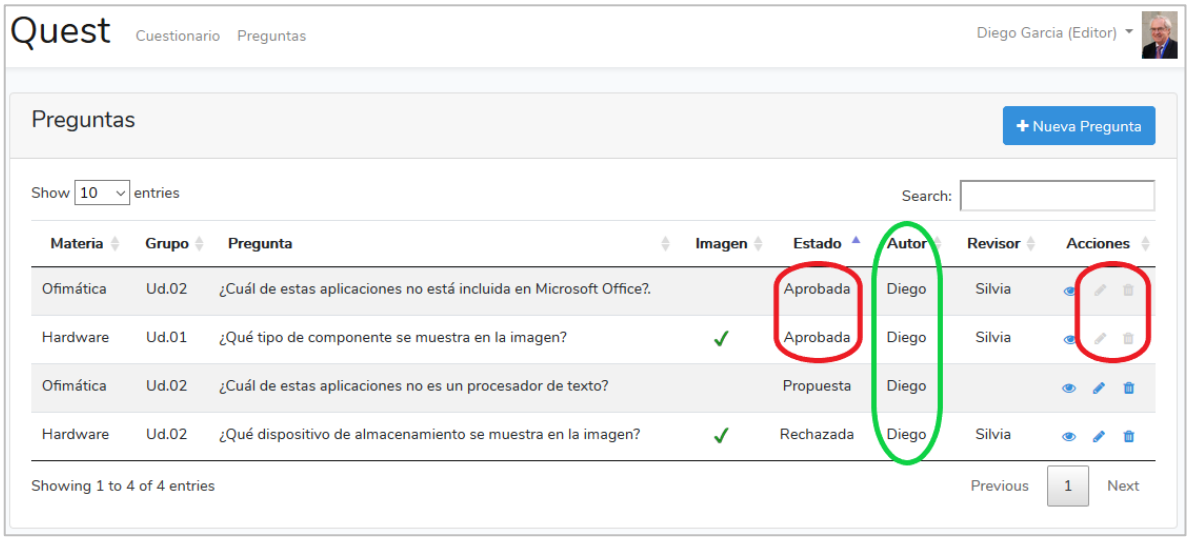

Figura 27. Página Listado Preguntas. Usuario 'Editor'.

Usuario Revisor

Si el usuario que ha accedido a la aplicación tiene perfil 'Revisor', al igual que el 'Editor' tendrá la opción de acceder a la gestión de preguntas, pero en este caso se mostrarán las preguntas de todos los usuarios, y tendrá disponible la opción 'Revisar', que estará activa para las preguntas en estado 'Propuesta'.

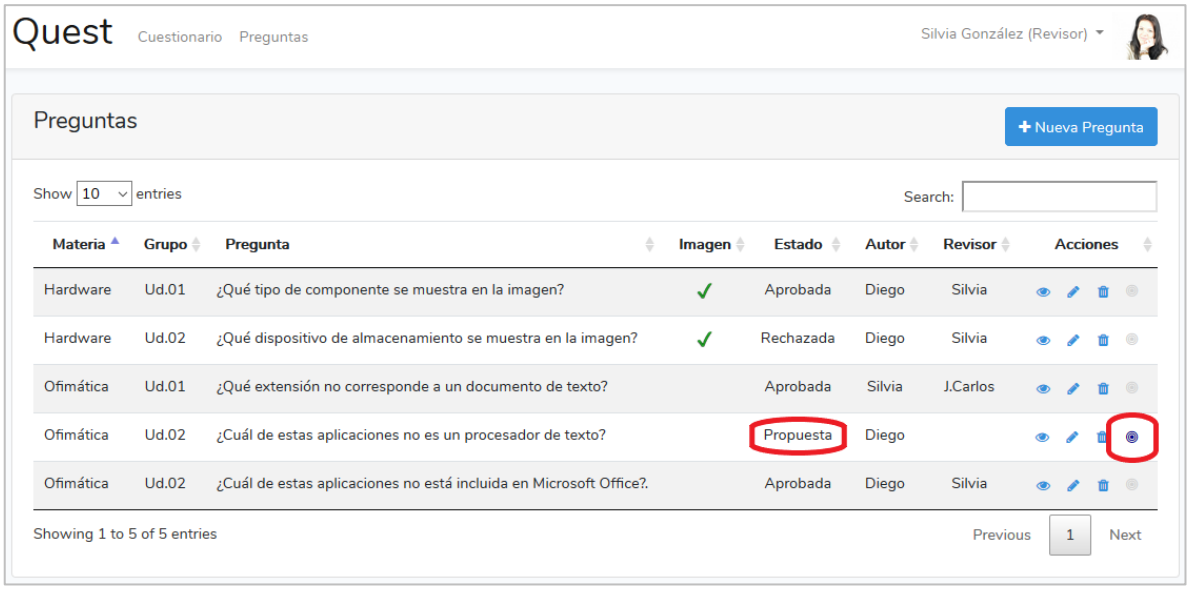

Figura 28. Página Listado Preguntas. Usuario 'Revisor'.

Usuario Administrador

Si el usuario que ha accedido a la aplicación tiene perfil 'Administrador', tendrá disponibles todas las opciones de la aplicación.

Figura 29. Menú usuario 'Administrador'.

12. Usabilidad/UX

Para mejorar la usabilidad de la aplicación y conseguir una buena experiencia del usuario, se han tenido en cuenta los siguientes aspectos:

- Filtrado y paginación de listas.
- Diseño de formularios.
- Flujo de navegación.

Filtrado y Paginación de Listas

Las páginas de listado ofrecen la posibilidad de:

- Indicar el número de registros que debe mostrar cada página.
- Indicar un filtro (que se aplicará a cualquier columna) para restringir los registros que se muestran.
- Pulsar sobre los botones de navegación en la parte inferior derecha de la lista.

Se ha seleccionado el tipo de paginación numerada sobre otras disponibles (scroll infinito, o botón 'cargar más') por presentar las siguientes ventajas:

- El usuario tiene una visión global de la información, ya que conoce el número registros y de páginas.
- Es muy flexible, ya que puede verlo todo (sin filtro) o una parte (aplicando un filtro), y puede configurar el tamaño de cada página (indicando el número de registros a mostrar).
- Es muy eficiente si el usuario está buscando algo concreto.
- La navegación es muy dinámica.

Diseño de Formularios

A la hora de definir la estructura y organización de los formularios, no sólo se ha tenido en cuenta su diseño visual (layout), sino también se ha optimizado la información que contiene. Se han aplicado los siguientes criterios: [4]

- **Solicitar sólo los campos necesarios**. Cada campo que se solicite debe estar totalmente justificado, evitando aquellos que sean innecesarios. Por ejemplo, el formulario de registro únicamente solicita el nombre, dirección de correo y contraseña.
- **Agrupar la información**. Al organizar de forma lógica la información que está relacionada se mejora la usabilidad. La información que se solicite estará agrupada en bloques de información que tengan sentido.
- **Campos en varias columnas**. Para que la información se muestre de forma más natural, y debido a que no hay gran número de campos en cada página, he optado por una organización de varias columnas en cada fila, que se irán completando de izquierda a derecha y de arriba hacia abajo.
- **Tamaño del campo adecuado**. El tamaño de cada campo debe estar acorde con el número de caracteres que puede contener. Por ejemplo, el nombre del usuario debe ser mucho más largo que su abreviatura.
- **Uso del tabulador**. Asegurar el adecuado movimiento del cursor si el usuario hace uso del tabulador.
- **Etiquetas.** Hay varios aspectos en relación a las etiquetas.
    - ✓ **Cortas y descriptivas**. Las etiquetas de los campos serán cortas, y lo suficientemente descriptivas como para que el usuario no necesite de ninguna aclaración adicional. La información será sencilla y breve.
    - ✓ **Ubicación.** He decidido colocar las etiquetas en la parte superior de los campos, alineadas a la izquierda, de forma que el usuario no tenga que buscar por separado una etiqueta y luego asociarla al campo correspondiente. Esta asociación es mucho más eficiente con el movimiento natural hacia abajo.
    - ✓ **Relación con campos.** Para mejorar la legibilidad de los formularios, la separación de una etiqueta con el campo con el que está asociada debe ser menor que la separación con los campos de la fila superior. Esto tiene relación con las leyes de la Gestalt, concretamente con la ley de proximidad [5], que dice: "Los elementos aislados, pero con cierta cercanía tienden a ser considerados como grupos.".

- **Enfocar el campo elegido**. Hay que destacar el campo que tiene el foco, para captar la atención del usuario sobre el campo a rellenar. En este caso se ha optado por resaltar el borde, más grueso y azul.
- **Campos preseleccionados**. Siempre que sea posible, utilizar valores conocidos para los campos. Por ejemplo, los que el usuario ya haya indicado en el registro.
- **Campos desplegables**. Utilizar campos desplegables (combo-box) si los posibles valores están predefinidos. Por ejemplo, los tipos de perfiles de un usuario.
- **Campos no editables**. Proteger aquellos campos que no puedan ser modificados por el usuario, y mostrarlos con otro aspecto, para que los identifique rápidamente. En este caso se ha optado por mostrarlos con fondo gris.
- **Ubicación de los botones**. Los botones de cada formulario se ubicarán de forma estándar en la misma posición, en este caso en la parte inferior izquierda de cada página.

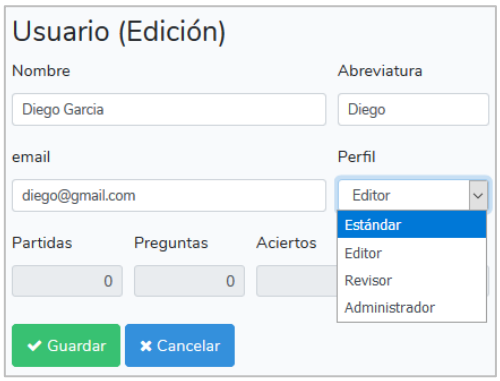

Figura 30. Ejemplo diseño formulario.

Flujo de Navegación

Se ha homogeneizado el flujo de navegación, de forma que cualquier tipo de gestión (preguntas, materias, grupos, usuarios, etc.) se realice de igual forma. Habrá una página de listado inicial, una página de detalle para cada operación (creación, consulta y edición), y para el borrado se mostrará un mensaje de confirmación. En caso de existir alguna acción adicional, se mostrará a la derecha.

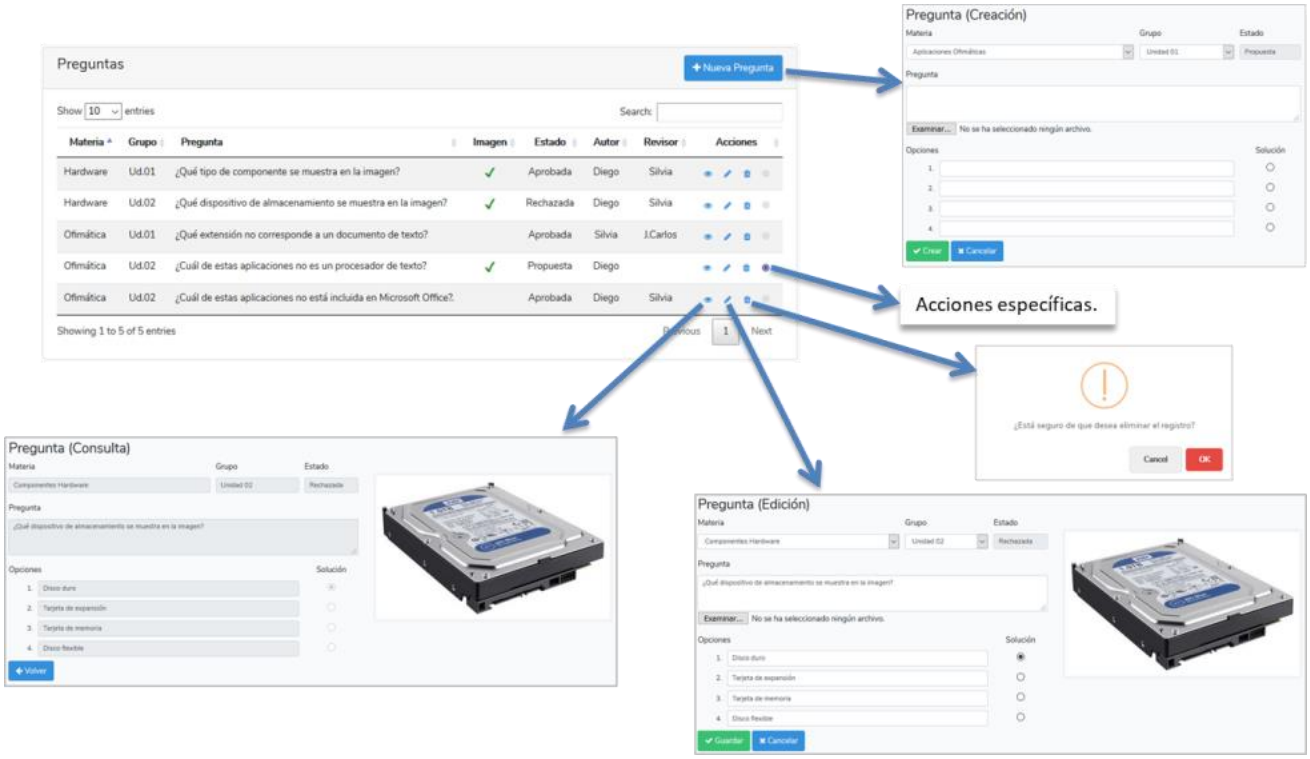

Figura 31. Flujo de navegación.

Página de Listado

Inicialmente se accederá a la página que muestra un listado de todos los elementos.

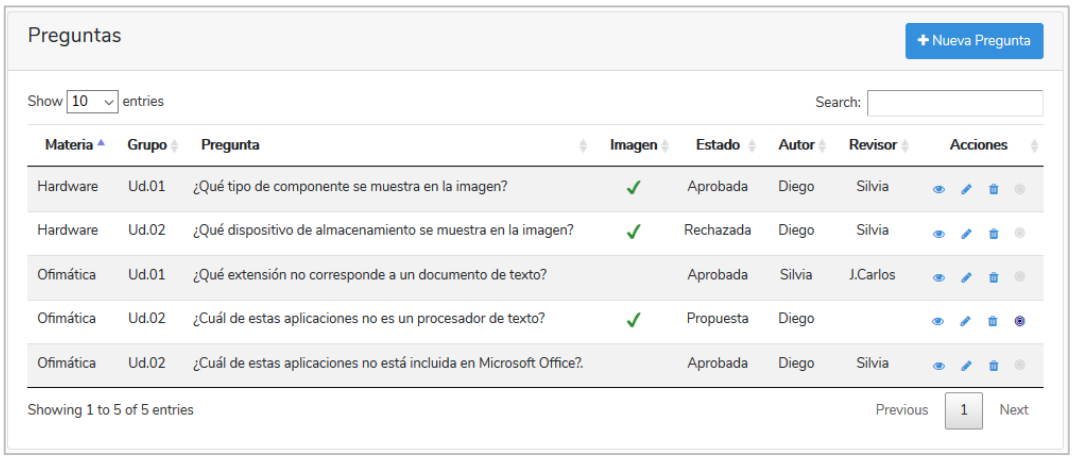

Figura 32. Ejemplo de página de listado.

El usuario podrá realizar las siguientes operaciones:

- Indicar el número de registros que debe mostrar cada página.
- Indicar un filtro (que se aplicará a cualquier columna) para restringir los registros que se muestran.
- Pulsar sobre los títulos de cada columna, para ordenar por dicha columna en ambos sentidos (asc/desc).
- Pulsar sobre los botones de navegación en la parte inferior derecha de la lista.
- Pulsar el botón de la parte superior derecha, para acceder a la página de creación de un nuevo elemento.
- Pulsar sobre los enlaces de acciones de cada fila, para consultar, modificar o borrar el registro.

Página de Creación

Mostrará los campos vacíos, excepto los desplegables que sean obligatorios que mostrarán un valor por defecto. Las acciones disponibles son 'Crear' y 'Cancelar'.

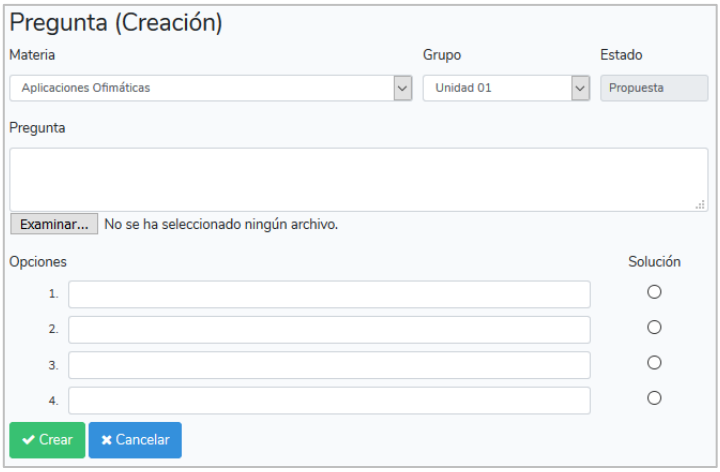

Figura 33. Ejemplo de página de creación.

Confirmación del Borrado

Al pulsar sobre el enlace de borrado se mostrará un mensaje de confirmación.

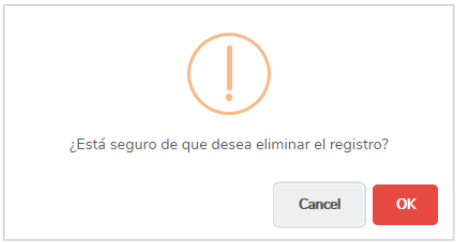

Figura 34. Ejemplo de mensaje de confirmación.

Página de Consulta

Mostrará los campos informados y protegidos. La única acción disponible es 'Volver'.

Figura 35. Ejemplo de página de consulta.

Página de Edición

Mostrará los campos informados, y estarán habilitados o protegidos en función de que puedan ser modificados o no por el usuario. Las acciones disponibles son 'Guardar' y 'Cancelar'.

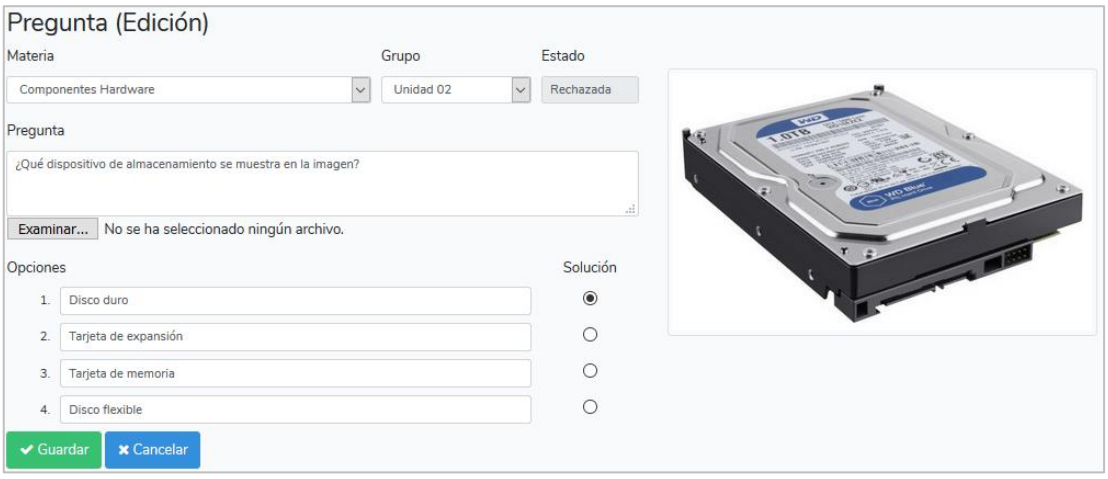

Figura 36. Ejemplo de página de creación

Acciones Específicas

En ocasiones es necesario gestionar algún tipo de acción específica, como en la gestión de Preguntas, donde el usuario con perfil 'Revisor' podrá revisar las preguntas en estado 'Propuesta', para aprobarlas o rechazarlas.

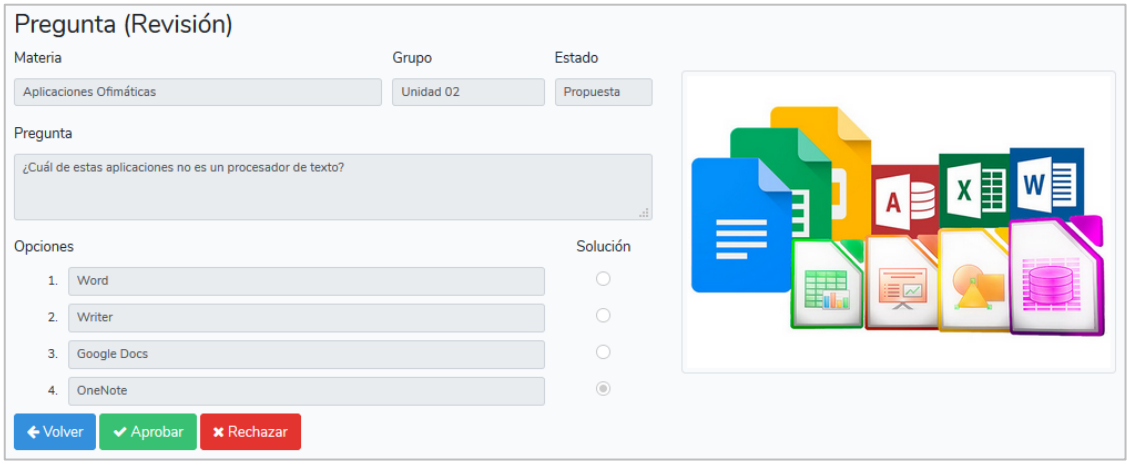

Figura 37. Ejemplo de página de acción específica (Revisión de pregunta).

13. Seguridad

Los usuarios que deseen acceder al sistema deben identificarse mediante usuario y contraseña, que será validada sobre la tabla 'users' de la base de datos.

Como medida de seguridad, durante el registro del usuario, la contraseña es encriptada a la hora de almacenar la información en la base de datos, por lo que no es posible acceder a su valor a través de una consulta SQL.

Por ejemplo, si el usuario indica como contraseña el valor '12345678', la información que se almacenará en la base de datos será:

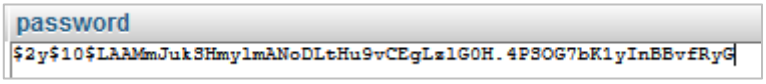

Figura 38. Ejemplo de clave encriptada.

Por otro lado, la contraseña debe tener una longitud mínima de 8 caracteres, aunque no implementa ningún mecanismo adicional para asegurar su complejidad, o que se bloquee tras un determinado número de intentos fallidos.

14. Tests

Como ya se ha indicado anteriormente en el apartado 'Usabilidad/UX', se ha homogeneizado el flujo de navegación, de forma que cualquier tipo de gestión (preguntas, materias, grupos, usuarios, etc.) se realice de igual forma. Esto, además de una mejora para el usuario, supone una ventaja para el desarrollo (por la reutilización de código) y por tanto una mejora a la hora de detectar y corregir errores.

Debido al alcance del proyecto y a lo reducido del equipo de trabajo, no se han definido diferentes tipos de pruebas (unitaria, integrada, producto, etc.), ni se ha generado documentación específica para ello (ciclos de prueba, condiciones de prueba, resultados esperados, resultados obtenidos, etc.).

Las pruebas se han ido realizando a medida que avanzaba el desarrollo, de forma que la reutilización del código fuese lo más efectiva posible.

A continuación se detallan las principales pruebas realizadas, y el correspondiente funcionamiento que ha tenido la aplicación.

Condición de Prueba	Resultado Esperado
Registro	
No se informa ningún campo.	Error indicando todos los campos obligatorios que no están informados.
Contraseña muy pequeña.	Error indicando que al menos debe tener ocho caracteres.
Confirmación de contraseña distinta a la contraseña.	Error indicando que ambos valores deben ser iguales.
Correo electrónico igual al de otro usuario ya registrado.	Error indicando que ya hay otro usuario con esa dirección de correo.
Se informan todos los campos con valores corectos.	Se realiza el registro del nuevo usuario.
Login	
No se informa ningún campo.	Error indicando todos los campos obligatorios que no están informados.
Usuario con contraseña incorrecta.	Error indicando que el usuario/contraseña no son correctos.
Usuario con contraseña correcta.	Se accede al sistema, mostrando las opciones de menú correspondientes a su perfil.
Listado	
Si no hay datos.	No se muestra ningún elemento en la lista. No hay filtros ni botones de navegación. Está disponible el botón 'Nuevo'.
Sí hay datos.	Se muestran los elementos de la lista. Se puede filtrar, ordenar y cambiar de página. Están disponibles las diferentes acciones para cada registro.
Pulsar botón 'Nuevo'	Se accede a la página de detalle para la creación de un nuevo registro.
Pulsar enlace 'Consulta' de un registro.	Se accede a la página de detalle para consultar dicho registro.
Pulsar enlace 'Edición' de un registro.	Se accede a la página de detalle para modificar dicho registro.
Pulsar enlace 'Borrado' de un registro.	Se muestra un mensaje de confirmación antes de borrar dicho registro.
Creación	
Aspecto inicial.	Mostrará los campos vacíos, excepto los desplegables que sean obligatorios que mostrarán un valor por defecto.
Pulsar botón 'Cancelar'	Se vuelve a la página de listado, sin realizar ningún cambio en la base de datos.
Pulsar botón 'Crear' sin informar ningún campo.	Error indicando todos los campos obligatorios que no están informados.
Pulsar botón 'Crear' informando todos los campos de forma errónea.	Error indicando los errores de validación de cada campo.
Pulsar botón 'Crear' informando todos los campos correctamente.	Se crea el nuevo registro y se vuelve al listado con la información actualizada.
Pulsar botón 'Crear' informando sólo los campos obligatorios.	Se crea el nuevo registro y se vuelve al listado con la información actualizada.
Consulta	
Aspecto inicial.	Mostrará los campos informados y protegidos.
Pulsar botón 'Volver'	Se vuelve a la página de listado, sin realizar ningún cambio en la base de datos.

Edición	
Aspecto inicial.	Mostrará los campos informados, y estarán habilitados o protegidos en función de que puedan ser modificados o no por el usuario.
Pulsar botón 'Cancelar'	Se vuelve a la página de listado, sin realizar ningún cambio en la base de datos.
Pulsar botón 'Guardar' borrando algún campo obligatorio.	Error indicando todos los campos obligatorios que no están informados.
Pulsar botón 'Guardar' informando los campos de forma errónea.	Error indicando los errores de validación de cada campo.
Pulsar botón 'Guardar' informando todos los campos correctamente.	Se actualiza el registro y se vuelve al listado con la información actualizada.
Pulsar botón 'Guardar' informando sólo los campos obligatorios.	Se actualiza el registro y se vuelve al listado con la información actualizada.
Borrado	
Aspecto inicial.	Se mostrará un mensaje de confirmación, con las opciones 'Ok' y 'Cancel'.
Pulsar botón 'Cancel'	Se vuelve a la página de listado, sin realizar ningún cambio en la base de datos.
Pulsar botón 'Ok'	Se borra el registro y se vuelve a la página de listado con la información actualizada.

Tabla 1. Condiciones de prueba y resultados esperados.

15. Versiones de la Aplicación

Las versiones de la aplicación que se han preparado durante su desarrollo son las siguientes:

Versión	Fecha	Descripción
1.0	30/10/2019	Entrega PEC2. Incluye el login y parte de la funcionalidad.
Beta	08/12/2019	Entrega PEC3. Incluye toda la funcionalidad de la aplicación. Se han realizado la mayor parte de las pruebas.
Alpha	06/01/2020	Entrega Final. Se han completado todas las pruebas de la aplicación.

Tabla 2. Versiones de la aplicación.

16. Instalación

Requisitos

Para la instalación de la aplicación no hay unos **requisitos hardware** específicos.

En cuanto a los **requisitos software**, es necesario instalar un servidor XAMPP, preferiblemente la versión 7.1 (utilizada para el desarrollo) o superior.

Instrucciones

Creación de la Base de Datos

Sobre el servidor de base datos MySQL que se vaya a utilizar (puede ser el que incorpora XAMPP, accesible a través de phpMyAdmin), hay que crear la base de datos con el nombre que se desee (por ejemplo 'quest'), donde se importará el archivo 'quest.sql' que se proporciona, y que contiene las tablas y relaciones utilizadas por la aplicación.

Instalación y Configuración

Para la instalación y configuración de la aplicación es necesario realizar las siguientes tareas:

- Copiar los archivos de la aplicación a la carpeta pública del servidor.
- Modificar el archivo '.env' para configurar los parámetros de conexión a la base de datos del servidor. Estos valores dependerán de la ubicación de la base de datos, de su nombre, y del usuario/contraseña con el que se vaya a acceder.

```
APP_ENV=production
APP_DEBUG=false

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=quest
DB_USERNAME=jgarber117
DB_PASSWORD=xxxxxxxx
```

- Dar permisos de escritura al directorio '/storage' y todos sus subdirectorios.
- Añadir el archivo .htaccess, para que las peticiones se redirijan a la carpeta '/public', o incluir '/public' al final de la url con la que se acceda al servidor.

17. Bugs

Se han detectado las siguientes incidencias, que aunque no puedan ser calificadas como errores de código, sí que deberían haberse tenido en cuenta durante la fase de diseño. No han sido corregidos por la falta de tiempo y el escaso impacto que tienen para el resto de la aplicación:

- Cuando se asigna una imagen a una pregunta, no hay ninguna opción para hacer que luego dicha pregunta no tenga ninguna imagen asociada. Se elimina al asignarle una nueva imagen.
- Al seleccionar la imagen que será asignada a una pregunta, no se muestra una previsualización de dicha imagen. El resultado final no se ve hasta que posteriormente se accede al detalle de la pregunta para su consulta o edición.

18. Proyección a Futuro

Se han identificado las siguientes mejoras a realizar en una posible ampliación de este proyecto:

- Completar el mecanismo de registro que proporciona Laravel, para habilitar el envío de correo electrónico para el reseteo de la contraseña.
- Hacer que la aplicación pueda gestionar varios idiomas.
- Modificar la estructura de mensajes de error que proporciona Laravel por defecto, de forma que:
  - ✓ Se muestren junto a los campos concretos donde se produce el error, no en un mensaje general.
  - ✓ Se utilice como parte del mensaje el texto de la etiqueta asociada al campo, en lugar del nombre que tiene el campo en la base de datos.
  - ✓ El texto que se incluye como parte del error no se muestre en inglés.
- Habilitar la validación de cada campo en el mismo momento de completarlo y no en el envío del formulario.
- Realizar pruebas de concurrencia y rendimiento, con cierto volumen de usuarios utilizando la aplicación de forma simultánea.

19. Conclusiones

Tras la realización del proyecto hago una valoración muy positiva, tanto de los resultados obtenidos como por el desarrollo del proyecto en su conjunto.

Este proyecto suponía un gran reto para mí, ya que era la primera vez que me enfrentaba a un desarrollo web de cierta envergadura, y con unos plazos muy ajustados. Por tanto, era muy importante elegir una temática interesante, con un alcance flexible y una tecnología que permitiese un desarrollo robusto y ágil.

En cuanto al **desarrollo del proyecto**, pienso que en su momento realicé una buena elección de la temática, que considero muy interesante por la aplicación práctica que puede tener en clase con el alumnado. Por otro lado, la tecnología utilizada ha superado con creces las expectativas que tenía inicialmente con Laravel, ya que he visto cómo aporta soluciones simples ante determinadas necesidades técnicas.

Me preocupaba enormemente que una vez iniciado el proyecto encontrase ciertas dificultades técnicas que me impidiesen su desarrollo, o que la solución fuese tan compleja que no me permitiese finalizar en los plazos previstos. Pero no ha sido así, sino todo lo contrario, cada vez que tenía una determinada necesidad, veía como la solución que me proporcionaba el framework era aún mejor que la esperada inicialmente. Por ejemplo, la gestión y almacenamiento de imágenes, validación de la información introducida por el usuario, gestión de excepciones, registro y login de usuarios, etc.

También estoy muy satisfecho con el **resultado obtenido**, ya que he podido desarrollar toda la funcionalidad que había previsto inicialmente, aunque han ido surgiendo nuevas ideas que he tenido que dejar para futuras ampliaciones. Entre los resultados obtenidos, destacaría el uso de imágenes por parte del usuario, la gestión de perfiles de usuarios y los estados por los que puede pasar cada pregunta.

En definitiva, este proyecto ha supuesto un reto para mí, ya que además de satisfacer una necesidad profesional, me ha permitido poner en práctica muchos de los conocimientos adquiridos durante la realización del Máster.

Anexo 1. Entregables del Proyecto

La lista de archivos entregados es la siguiente:

Archivo	Descripción
PAC_FINAL_PRJ_GarciaBermudez_JoseCarlos.rar	Código de la aplicación y exportación de la base de datos.
PAC_FINAL_MEM_GarciaBermudez_JoseCarlos.pdf	Memoria del proyecto.
PAC_FINAL_PRS_GarciaBermudez_JoseCarlos.pdf	Presentación del proyecto, en formato PDF.
PAC_FINAL_PRS_GarciaBermudez_JoseCarlos.pptx	Presentación del proyecto, en formato PowerPoint.
PAC_FINAL_VID_GarciaBermudez_JoseCarlos.mp4	Vídeo de presentación.
PAC_FINAL_EVA_GarciaBermudez_JoseCarlos.docx	Autoinforme de evaluación.

Tabla 3. Entregables del proyecto.

Anexo 2. Código Fuente

Para la configuración inicial y el desarrollo base del proyecto, se han seguido las indicaciones del desarrollo de una lista de tareas de la documentación oficial de Laravel [6][7].

A continuación se muestran extractos del código fuente, que muestran partes relevantes del desarrollo.

Estructura de la Aplicación

El Framework Laravel, que implementa el **patrón de diseño MVC**, divide las aplicaciones en tres niveles de abstracción (Modelo, Vista y Controlador), que a nivel de código se organiza de la siguiente forma:

Vistas

Las vistas, encargadas de mostrar la información al usuario, se ubican en el directorio '/resources/views'.

Laravel utiliza un sistema de plantillas llamado Blade [8], que contienen la parte del código que se repite en las diferentes vistas (cabecera, barra de navegación, pie…), que no tendría sentido implementar en cada una. Cada vista tendrá la extensión '.blade.php'.

La carpeta '/resources/views' contiene:

- Plantilla. Para el proyecto se ha trabajado con la plantilla por defecto:

  ```
  /resources/views/layouts/app.blade.php
  ```

- Página de inicio.

  ```
  /resources/views/home.blade.php
  ```

- Una carpeta para cada entidad que se gestiona (preguntas, materias, grupos…), que de forma estándar contendrá las vistas:

  ```
  index.blade.php     Lista de elementos.
  create.blade.php    Creación de un nuevo elemento.
  edit.blade.php      Edición de un elemento.
  show.blade.php      Consulta de un elemento.
  ```

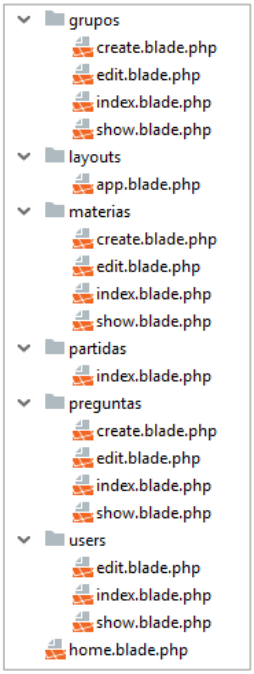

Figura 39. Directorio de Vistas

Controladores

Los controladores de Laravel [9], que agrupan en una clase un conjunto de peticiones HTTP relacionadas con una entidad, se almacenan en el directorio 'app/Http/Controllers'.

Hay un controlador para cada una de las entidades que se gestionan:

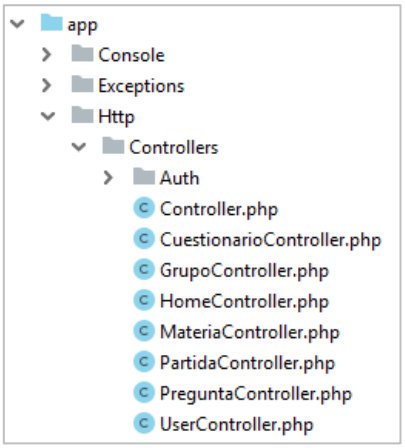

Figura 40. Directorio de Controladores

A continuación se muestra el código del controlador 'MateriaController.php', que incluye los métodos estándar para gestionar las peticiones que se realicen sobre la tabla de 'Materias'.

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Materia;

class MateriaController extends Controller
{
    // Crea una nueva instancia del controlador
    public function __construct()
    {
        // Todas las operaciones requieren que el usuario esté registrado
        $this->middleware('auth');
    }

    // Muestra la lista con todos los registros
    public function index(Request $request)
    {
        // Obtenemos todos los registros ordenados
        $materias = Materia::orderBy('nombre')->get();

        // Pasamos a la vista los registros obtenidos
        return view('materias.index', compact('materias'));
    }

    // Página de creación
    public function create()
    {
        return view('materias.create');
    }

    // Función que almacena el nuevo registro
    public function store(Request $request)
    {
       request()->validate([
            'nombre' => 'required|max:50',
            'nombre_corto' => 'required|max:10',
        ]);

        // Insertar el registro en la base de datos
        $materia = new Materia;

        $materia->nombre = $request->nombre;
        $materia->nombre_corto = $request->nombre_corto;

        $materia->save();

        return redirect('/materias');
    }

    // Muestra el detalle de la materia
    public function show($id)
    {
        $materia = Materia::find($id);

        return view('materias.show', compact('materia'));
    }

    // Función de borrado
    public function destroy(Request $request, Materia $materia)
    {
        try {
            // Borrar el registro de la base de datos
            $materia->delete();
        }
        catch (\Exception  $exception) {
            if ($exception->getCode() == 23000)
                $mensaje = 'No es posible la operación. Tiene información asociada.';
            else
                $mensaje = $exception->getMessage();

            return back()->withError($mensaje)->withInput();
        }

        return redirect('/materias');
    }
```

```
        // Página de edición
        public function edit($id)
        {
            $materia = Materia::find($id);

            return view('materias.edit', compact('materia'));
        }

        // Función de actualización
        public function update(Request $request, $id)
        {
            request()->validate([
                'nombre' => 'required|max:50',
                'nombre_corto' => 'required|max:10',
            ]);

            // Buscar el registro que se va a modificar
            $materia = Materia::find($id);

            // Cargar la información del registro a actualizar
            $materia->nombre = $request->nombre;
            $materia->nombre_corto = $request->nombre_corto;

            $materia->save();

            return redirect('/materias');
        }
    }
```

Modelos

Los modelos son los responsables de la comunicación con la base de datos. Cada tabla que se gestiona en la aplicación tendrá un modelo asociado, que se genera de forma automática ejecutando el siguiente comando (en este caso para la tabla 'Materia').

```
php artisan make:model Materia
```

Los modelos se ubican en el directorio '/app':

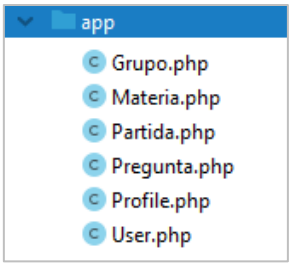

Figura 41. Directorio de Modelos.

Los modelos extienden de la clase 'Model', y por defecto no necesitan de implementar código específico.

Por ejemplo, el modelo para la tabla Materia es el siguiente.

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Materia extends Model
{
    //
}
```

Rutas

En Laravel es posible centralizar la definición de las rutas [9][10] de la aplicación en el archivo '/routes/web.php', con lo que estarán disponibles a través de la URL del navegador. Las rutas siempre se declaran usando la clase Route, seguida del verbo HTTP correspondiente. Se pueden registrar rutas que respondan a cualquier verbo HTTP.

Para el proyecto he utilizado la posibilidad que ofrece Laravel de definir las rutas como recursos, creando por defecto rutas para gestionar todos los tipos de peticiones RESTful para una entidad.

Por ejemplo, si se define el siguiente recurso para gestionar las materias:

```
Route::resource('materias', 'MateriaController');
```

Implícitamente se estarían generando las siguientes rutas:

Ruta	Verbo	Acción	Nombre Ruta
/materias	GET	index	materias.index
/materias/create	GET	create	materias.create
/materias	POST	store	materias.store
/materias/{id}	GET	show	materias.show
/materias/{id}/edit	GET	edit	materias.edit
/materias/{id}	PUT/PATCH	update	materias.update
/materias/{id}	DELETE	destroy	materias.destroy

Tabla 4. Rutas generadas por un 'resource'.

Además de las rutas definidas por defecto, también ha sido necesario definir otras rutas personalizadas. A continuación se muestra la declaración completa de rutas del archivo 'web.php'.

```
<?php

use Illuminate\Http\Request;
use App\Post;

Auth::routes();

// Página de inicio
Route::get('/', 'HomeController@show');
Route::get('/home', 'HomeController@show');

Route::resource('materias', 'MateriaController');                    // Gestión de materias
Route::resource('grupos', 'GrupoController');                        // Gestión de grupos

Route::resource('preguntas', 'PreguntaController');                  // Gestión de preguntas
Route::get('preguntas/{id}/review', 'PreguntaController@review');    // Revisión de una pregunta
Route::post('preguntas/{id}/approve', 'PreguntaController@approve');// Aprobación preg. revisada
Route::get('preguntas/{id}/refuse', 'PreguntaController@refuse');    // Rechazo pregunta revisada

Route::resource('users', 'UserController');                          // Gestión de usuarios
Route::get('users/perfil/{id}', 'UserController@edit_perfil');       // Edición usuario conectado

// Listado de partidas de un usuario
Route::get('partidas/usuario/{id}', 'PartidaController@usuario');
// Listado de partidas del usuario conectado
Route::get('partidas/perfil/{id}', 'PartidaController@perfil');

// Inicio cuestionario
Route::get('cuestionario/inicio', 'CuestionarioController@inicio');
// Carga inicial de preguntas
Route::post('cuestionario/carga', 'CuestionarioController@carga');
// Muestra siguiente pregunta
Route::get('cuestionario/siguiente', 'CuestionarioController@siguiente');
// Comprueba respuesta
Route::put('cuestionario/{id}/{respuesta}', 'CuestionarioController@comprobacion');
```

Plantilla

La plantilla de la aplicación 'app.blade.php' contiene la parte del código que se repite en las diferentes vistas (cabecera, barra de navegación, pie…).

A continuación se muestra el código de la plantilla, donde hay que destacar:

- En el 'head' se incluyen referencias a librerías externas, centralizadas en un único punto de la aplicación.
- En el menú hay opciones definidas entre las instrucciones '@auth' y '@endauth' que sólo se mostrarán si el usuario se ha identificado. De igual forma, la instrucción '@guest' se utilizará para opciones en las que el usuario aún no esté identificado.
- Si el usuario se haya identificado, también hay casos donde se comprueba su perfil 'Auth::user()->perfil_id' para mostrar o no una determinada opción.

- Al final del archivo, dentro de la etiqueta 'main', se encuentra la instrucción '@yield('content')' que es la que incluirá el contenido de la vista que se desee mostrar. Para ello, la vista correspondiente tendrá las etiquetas '@section('content')' y '@endsection' para delimitar el código a incluir.

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- CSRF Token -->
  <meta name="csrf-token" content="{{ csrf_token() }}">

  <title>{{ config('app.name', 'Laravel') }}</title>

  <!-- Scripts -->
  <script src="{{ asset('js/app.js') }}" defer></script>

  <!-- Fonts -->
  <link rel="dns-prefetch" href="//fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet">
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.4.0/css/font-awesome.min.css"
        rel='stylesheet' type='text/css'>

  <!-- Styles -->
  <link href="{{ asset('css/app.css') }}" rel="stylesheet">

  <!-- DataTable -->
  <script src="https://code.jquery.com/jquery-3.3.1.js"></script>
  <link rel="stylesheet" type="text/css"
        href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.css">
  <script type="text/javascript" charset="utf8"
          src="https://cdn.datatables.net/1.10.19/js/jquery.dataTables.js" defer></script>

  <!-- SweetAlert -->
  <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
</head>

<body>
  <div id="app">
    <nav class="navbar navbar-expand-md navbar-light bg-white shadow-sm">
      <div class="container">
        <a class="navbar-brand" href="{{ url('/') }}">
          <h1>Quest</h1>
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
                data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
                aria-expanded="false" aria-label="{{ __('Toggle navigation') }}">
          <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <!-- Left Side Of Navbar -->
          <ul class="navbar-nav mr-auto">
            @auth
              <li class="nav-item">
                <a class="nav-link" href="{{ url('cuestionario/inicio') }}">Cuestionario</a>
              </li>

              @if(Auth::user()->perfil_id > config('const.perfil.estandar'))
                <li class="nav-item">
                  <a class="nav-link" href="{{ url('preguntas') }}">Preguntas</a>
                </li>
              @endif

              @if(Auth::user()->perfil_id == config('const.perfil.administrador'))
                <li class="nav-item">
                  <a class="nav-link" href="{{ url('materias') }}">Materias</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="{{ url('grupos') }}">Grupos</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link" href="{{ url('users') }}">Usuarios</a>
                </li>
              @endif
            @endauth
          </ul>
```

```
          <!-- Right Side Of Navbar -->
          <ul class="navbar-nav ml-auto">
            <!-- Authentication Links -->
            @guest
              <li class="nav-item">
                <a class="nav-link" href="{{ route('login') }}">{{ __('Login') }}</a>
              </li>
              @if (Route::has('register'))
                <li class="nav-item">
                  <a class="nav-link" href="{{ route('register') }}">{{ __('Register') }}</a>
                </li>
              @endif
            @else
              <li class="nav-item dropdown">
                <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button"
                   data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" v-pre>
                  {{ Auth::user()->name}} {{Auth::user()->perfil_id ==
config('const.perfil.administrador') ? '(Admin)' : (Auth::user()->perfil_id ==
config('const.perfil.revisor') ? '(Revisor)' : (Auth::user()->perfil_id ==
config('const.perfil.editor') ? '(Editor)' : ''))}}
                  <span class="caret"></span>
                </a>

                <div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">
                  <a href="{{ url('/users/perfil/' . Auth::user()->id) }}" class="dropdown-item">
                    Datos Personales
                  </a>

                  <a href="{{url('/partidas/perfil/'.Auth::user()->id)}}" class="dropdown-item">
                    Mis Partidas
                  </a>

                  <hr>

                  <a class="dropdown-item" href="{{ route('logout') }}"
                     onclick="event.preventDefault();
                           document.getElementById('logout-form').submit();">
                    {{ __('Logout') }}
                  </a>

                  <form id="logout-form" action="{{ route('logout') }}"
                        method="POST" style="display: none;">
                    @csrf
                  </form>
                </div>
              </li>

              <li class="nav-item">
                <img src="{{ (Auth::user()->imagen!==null)?
asset('images/usuarios/'.Auth::user()->imagen) : asset('images/'.'no-user-image.png') }}"
                     class="" style="max-height:45px; width:auto">
              </li>
            @endguest
          </ul>
        </div>
      </div>
    </nav>

    <main class="py-4">
      @yield('content')
    </main>
  </div>
</body>
</html>
```

Validaciones y Mensajes de Error

En Laravel la implementación de las validaciones de los campos de la vista se realiza de forma muy simple en el controlador. Para ello basta con llamar al método '**request()->validate(…)**' al inicio del método correspondiente, al que se pasará como parámetro el array de campos con sus correspondientes validaciones.

En la aplicación las validaciones se implementan al inicio de los métodos 'store' y 'update' que realizan respectivamente las operaciones de inserción y actualización sobre la base de datos.

Por ejemplo, la validación de una nueva pregunta que se desea insertar sería:

```
request()->validate([
     'materia_id' => 'required',
     'grupo_id' => 'required',
     'texto' => 'required|max:200',
     'imagen' => 'image|mimes:jpeg,png,jpg,gif,svg|max:2048',
     'opcion1' => 'required|max:100',
     'opcion2' => 'required|max:100',
     'opcion3' => 'required|max:100',
     'opcion4' => 'required|max:100',
     'solucion' => 'required',
 ]);
```

Si alguna de estas validaciones no es correcta, en la vista se mostrará el mensaje de error correspondiente. Para ello, la vista tendrá que incluir la instrucción '**@include('common.errors')**'.

Constantes

En el archivo '/config/const.php' se realiza la declaración de las constantes que serán consultadas en diferentes puntos del proyecto, de forma que se mejora la legibilidad y mantenibilidad del código.

```
<?php
return [
    "cuestionario" => [
        "min_preguntas" => 2,
        "max_preguntas" => 10
    ],
    "perfil" => [
        "estandar" => 0,
        "editor" => 1,
        "revisor" => 2,
        "administrador" => 3
    ],
    "estado_pregunta" => [
        "propuesta" => "Propuesta",
        "aprobada" => "Aprobada",
        "rechazada" => "Rechazada"
]
];
```

Figura 42. Código del archivo de constantes.

Integridad Referencial

La definición de la estructura de cada tabla de la base de datos se encuentra en los archivos de migraciones, en el directorio 'database/migrations' del proyecto, e incluyen la definición de las relaciones de integridad entre las diferentes tablas.

Por ejemplo, la tabla 'preguntas' define relaciones de integridad con la tabla materias, grupos y users, indicando en cada caso que si en esas tablas se va a borrar un registro cuyo 'id' es utilizado en 'preguntas', que no se permita la operación (cláusula 'restrict').

```
$table->foreign( columns: 'materia_id')
    ->references( columns: 'id')->on( table: 'materias')->onDelete( action: 'restrict');

$table->foreign( columns: 'grupo_id')
    ->references( columns: 'id')->on( table: 'grupos')->onDelete( action: 'restrict');

$table->foreign( columns: 'autor_id')
    ->references( columns: 'id')->on( table: 'users')->onDelete( action: 'restrict');

$table->foreign( columns: 'revisor_id')
    ->references( columns: 'id')->on( table: 'users')->onDelete( action: 'restrict');
```

Figura 43. Ejemplo definición integridad referencial.

Control de Excepciones

En el proyecto también se hace un control de excepciones para controlar determinados errores, principalmente los producidos por errores de integridad referencial al intentar eliminar un registro.

Con el uso de bloques 'try / catch' en el controlador, se evita perder el control sobre el flujo de ejecución de la aplicación.

A continuación se muestra un ejemplo de código del método 'destroy', para el borrado de un registro de la base de datos. La lógica se ha incluido dentro de un bloque 'try', de forma que si se produce alguna excepción, el flujo de ejecución saltará al bloque 'catch', y en función del código de error se devolverá a la vista el mensaje correspondiente.

```
public function destroy(Request $request, Pregunta $pregunta)
{
    try {
        // Borrar el registro de la base de datos
        $pregunta->delete();

        // Si tiene alguna imagen asociada
        if ($pregunta->imagen !== null)
        {
            // Borrar el archivo de la carpeta pública
            $ruta_acceso_imagen = public_path('images/preguntas/').$pregunta->imagen;
            unlink($ruta_acceso_imagen);
        }
    }
    catch (\Exception  $exception) {
        if ($exception->getCode() == 23000)
            $mensaje = 'No es posible la operación. El registro tiene información asociada.';
        else
            $mensaje = $exception->getMessage();

        return back()->withError($mensaje)->withInput();
    }

    return redirect('/preguntas');
}
```

Como ejemplo, al intentar borrar una materia de la que ya se han creado preguntas, se muestra el correspondiente mensaje de error. Aunque se ha producido una excepción, ésta ha sido controlada, y en ningún momento se ha perdido el control del flujo de ejecución.

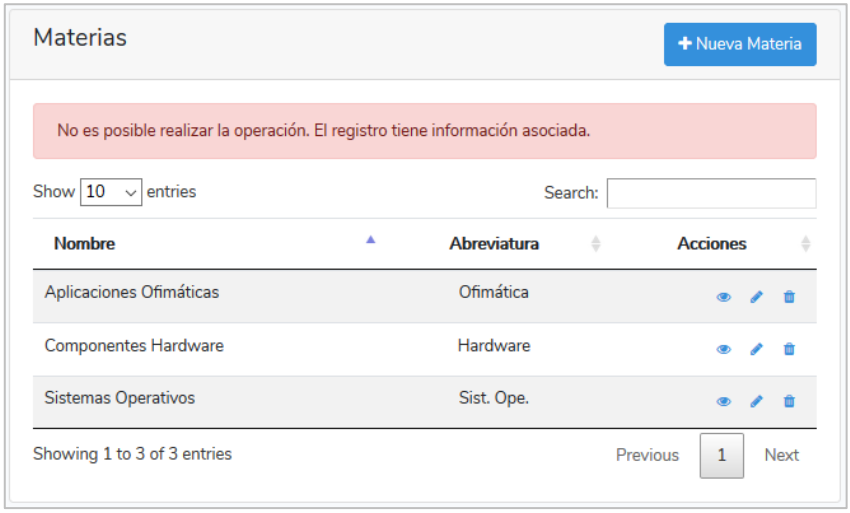

Figura 44. Ejemplo mensaje gestionado por una excepción.

Gestión de Imágenes

Las imágenes utilizadas en el proyecto se almacenan en el directorio '/public/images', donde se organizan de la siguiente forma:

- En ese directorio se encuentran las imágenes que se utilizan en diferentes puntos de la aplicación (logo, check, etc.)
- El directorio 'preguntas' contiene las imágenes que los usuarios de la aplicación han asociado a las diferentes preguntas que han publicado.
- El directorio 'usuarios' contiene las imágenes que los usuarios han asociado a su propio perfil.

Para las imágenes asociadas a preguntas y usuarios creados en la base de datos, en la tabla correspondiente sólo se almacena el nombre de la imagen.

Vista

El formulario incluye un control de tipo 'file', para que el usuario pueda seleccionar una imagen de su equipo local.

```
<input type="file" name="imagen">
```

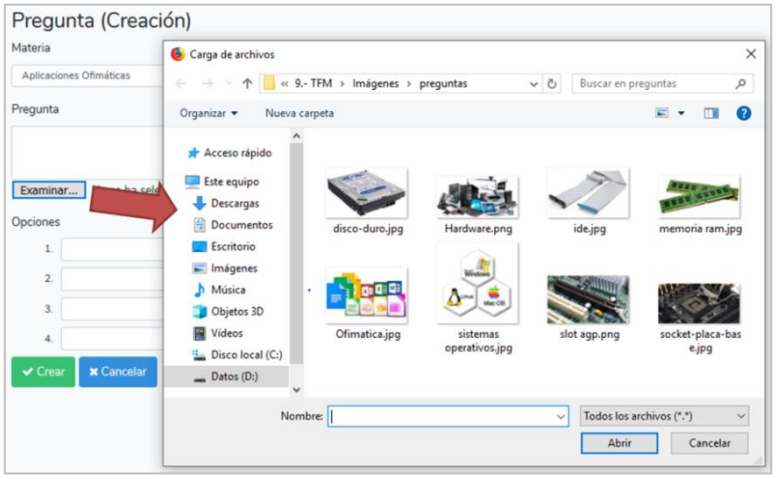

Figura 45. Selección de imagen desde equipo local.

Controlador

En la **creación** de un nuevo registro, en el controlador se realizan las siguientes operaciones:

- Se valida el tipo de archivo seleccionado y su tamaño.
- Se genera el nombre de la imagen (a partir de la fecha/hora actual y la extensión original).
- Se mueve la imagen a la carpeta correspondiente del servidor, utilizando el nombre generado.
- El nombre asignado a la imagen se almacena en la base de datos junto al resto de campos.

```php
// Función que almacena el nuevo registro
public function store(Request $request)
{
    request()->validate([
        'materia_id' => 'required',
        'grupo_id' => 'required',
        'texto' => 'required|max:200',
        'imagen' => 'image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        'opcion1' =>
```

```php
// Comprobar si se ha indicado una imagen
$imageName = null;
if ($request->imagen !== null)
{
    // Copiar la imagen a la carpeta pública
    $imageName = time().'.'.request()->imagen->getClientOriginalExtension();
    request()->imagen->move(public_path('images/preguntas'), $imageName);
}
```

Figura 46. Código para el almacenamiento de una imagen en el servidor.

En el **borrado**:

- Se borra el registro de la base de datos.
- Se borra el archivo de la carpeta pública.

```php
// Función de borrado
public function destroy(Request $request, Pregunta $pregunta)
{
    try {
        // Borrar el registro de la base de datos
        $pregunta->delete();

        // Si tiene alguna imagen asociada
        if ($pregunta->imagen !== null)
        {
            // Borrar el archivo de la carpeta pública
            $ruta_acceso_imagen = public_path('images/preguntas/').$pregunta->imagen;
            unlink($ruta_acceso_imagen);
        }
```

Figura 47. Código para el borrado de una imagen del servidor.

En la **edición**, el proceso es muy parecido a la creación, teniendo en cuenta que si se ha seleccionado una nueva imagen hay que borrar la imagen anterior de la carpeta pública.

```
// Comprobar si se ha indicado una nueva imagen
if ($request->imagen === null)
{
    // No se ha indicado una nueva imagen

    // Se mantendrá el nombre que tenía la imagen (será null si no había imagen)
    $imageName = $pregunta->imagen;
}
else
{
    // Se ha indicado una nueva imagen

    // Copiar la nueva imagen a la carpeta pública
    $imageName = time().'.'.request()->imagen->getClientOriginalExtension();
    request()->imagen->move(public_path('images/preguntas'), $imageName);

    // Comprobar si anteriormente se había indicado una imagen
    if ($pregunta->imagen !== null)
    {
        // Borrar la imagen anterior de la carpeta pública
        $ruta_acceso_imagen = public_path('images/preguntas/').$pregunta->imagen;
        unlink($ruta_acceso_imagen);
    }
}
```

Figura 48. Código para la actualización de una imagen del servidor.

Anexo 3. Librerías Externas

Para el desarrollo del proyecto se ha hecho uso de las librerías externas Font Awesome y SweetAlert2.

Font Awesome

La librería Font Awesome ha sido utilizada para mostrar iconos en los botones y enlaces de la aplicación.

La librería ha sido incluida en la cabecera de la plantilla de la aplicación, a través de la etiqueta:

```
<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.4.0/css/font-
awesome.min.css" rel='stylesheet' type='text/css'>
```

Para mostrar un icono como texto de un botón o de un enlace, se incluyen etiquetas como las siguientes:

```
<i class="fa fa-btn fa-check"></i>
<i class="fa fa-btn fa-pencil"></i>
```

Este será el aspecto de los iconos en los botones, y en los enlaces de las listas.

Figura 49. Iconos de la librería 'Font Awesome'.

SweetAlert2

La librería SweetAlert2 ha sido utilizada para mostrar mensajes mejorados al usuario, con un aspecto mucho más atractivo que los mensajes estándar.

La librería ha sido incluida en la cabecera de la plantilla de la aplicación, a través de la etiqueta:

```
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
```

Como ejemplo de uso, para mostrar un mensaje de confirmación cuando se pulse el botón de borrado, se debe asociar un nombre de clase a dicho botón (en este caso 'confirm').

```
<button type="submit" id="delete-pregunta-{{ $pregunta->id }}" class="btn btn-link btn-sm confirm" title="Borrar">
    <i class="fa fa-btn fa-trash"></i>
</button>
```

Más adelante, se asociará a dicho identificador de clase ('confirm') el código JavaScript de esta librería.

```
<script type="text/javascript">
    $(document).ready( function () {
        $('#tabla_preguntas').DataTable();

        $(".confirm").on("click", function(e){
            e.preventDefault();
            swal({
                icon: "warning",
                text: "¿Está seguro de que desea eliminar el registro?",
                buttons: true,
                dangerMode: true
            })
            .then ((willDelete) => {
                if (willDelete) {
                    $(this).closest("form").submit();
                }
            });
        });
    });
</script>
```

Figura 50. Código para mostrar mensajes con la librería SweetAler2.

En este caso se ha configurado el icono, texto, botones y modo. Si se confirma, se hará submit del formulario, en caso contrario se permanecerá en la vista actual.

El mensaje que se mostrará es el siguiente:

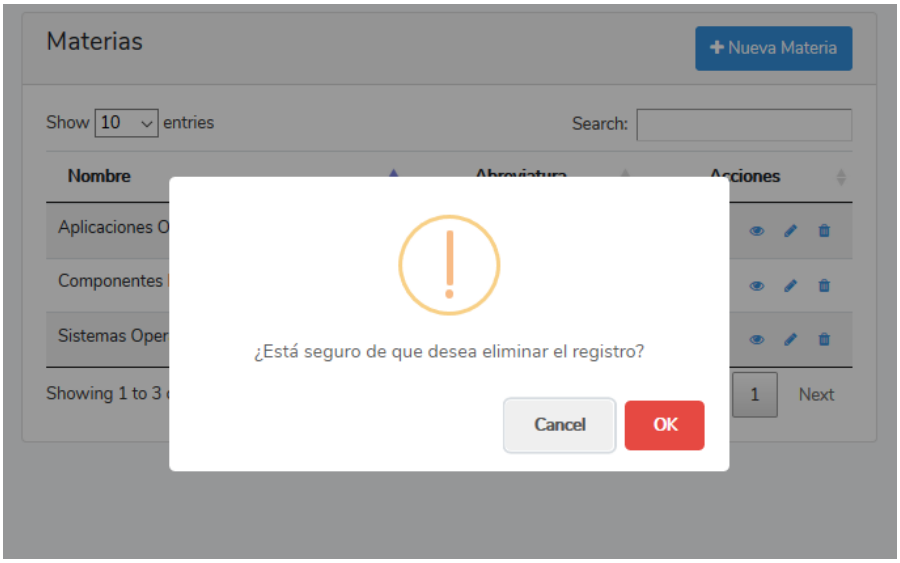

Figura 51. Ejemplo de mensaje con la librería SweetAler2.

Anexo 5. Guía de Usuario

La página de inicio de la aplicación muestra las opciones que están disponibles inicialmente. El usuario, al no estar aún identificado, solo podrá hacer login o registrarse.

Figura 52. Página de inicio. Usuario no identificado.

Registro

Para el registro de nuevos usuarios es necesario indicar nombre, dirección de correo electrónico y contraseña.

Por defecto se le asignará un perfil de usuario 'Estándar'.

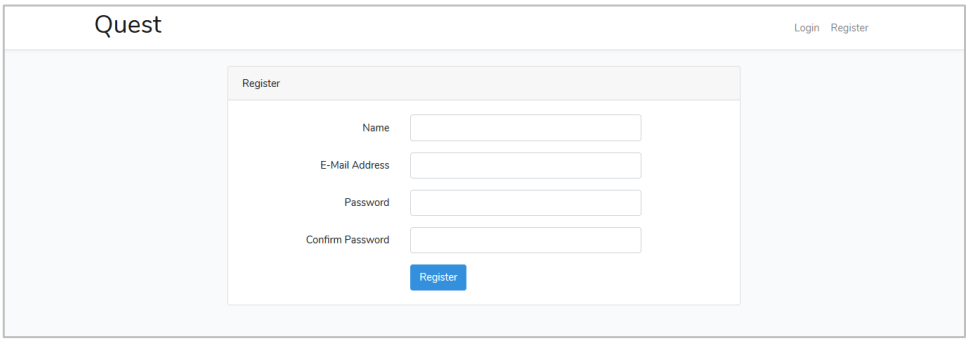

Figura 53. Página de registro.

Login

Los usuarios registrados en el sistema podrán identificarse a través de la página de login.

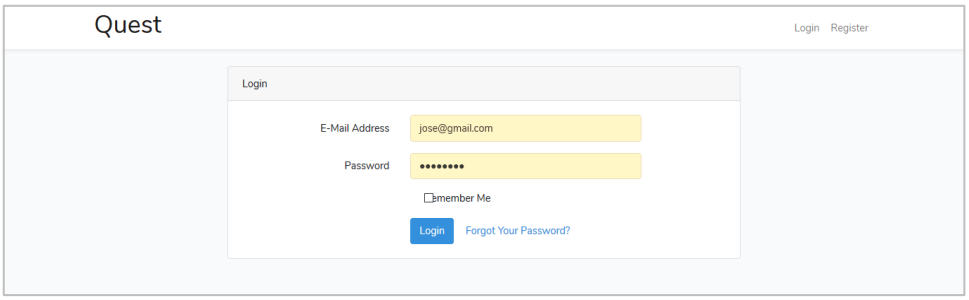

Figura 54. Página de login.

Usuario Identificado

Una vez identificado el usuario, el menú mostrará las opciones que tendrá disponibles en función de su perfil:

- **Estándar:** Sólo tendrá disponible la opción de realizar cuestionarios.

Figura 55. Menú usuario 'Estándar'.

- **Editor:** Tendrá además la opción de acceder a la gestión de preguntas.

Figura 56. Menú usuario 'Editor'.

- **Revisor**: Al igual que el Editor, tendrá la opción de acceder a la gestión de preguntas, pero con mayores privilegios (como se indicará más adelante).

Figura 57. Menú usuario 'Revisor'.

- **Administrador**: Tendrá disponibles todas las opciones de la aplicación.

Figura 58. Menú usuario 'Administrador'.

Datos Personales

Cualquier usuario que se haya identificado podrá editar sus datos personales y consultar el listado de sus partidas, pudiendo acceder respectivamente a las páginas de edición de usuario e histórico de partidas del usuario.

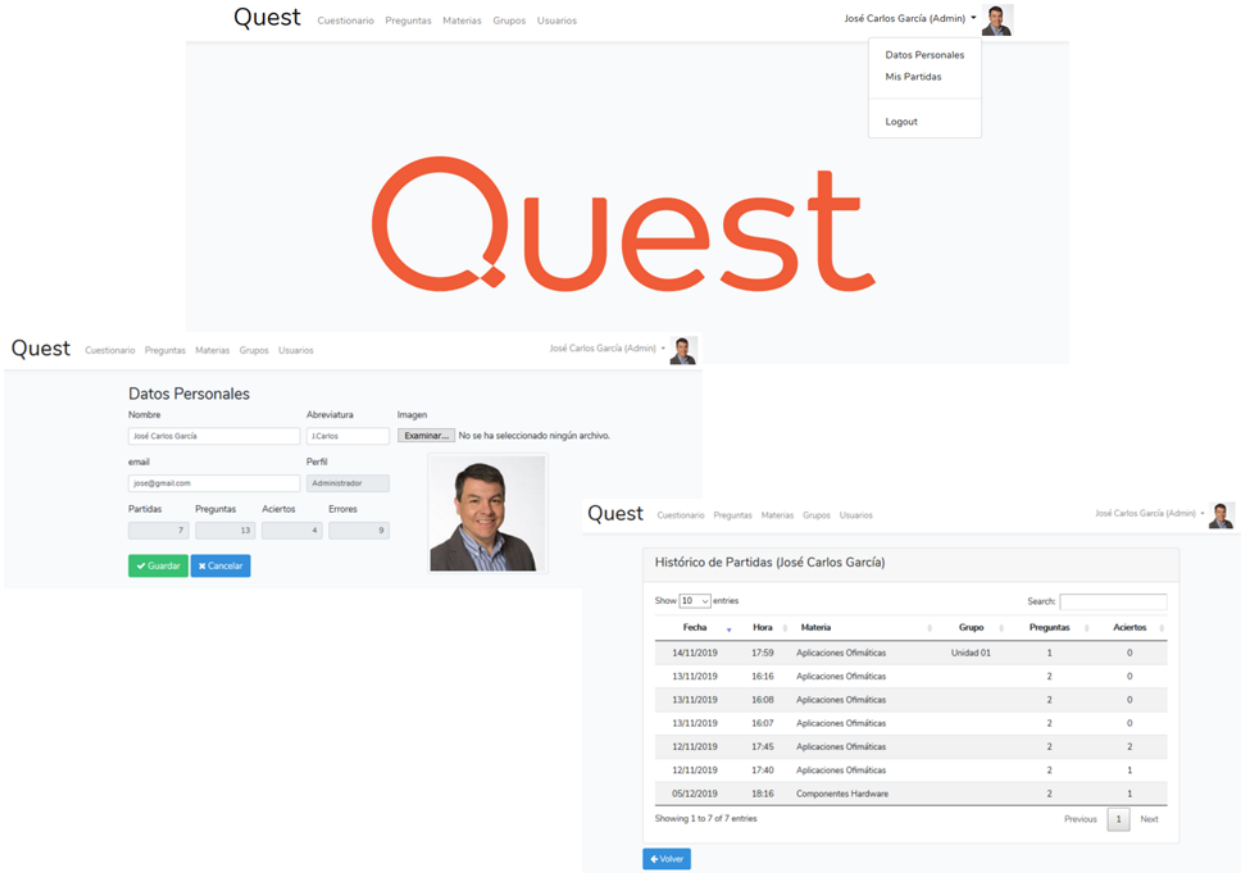

Figura 59. Información del usuario conectado.

Cuestionario

Al iniciar un cuestionario, se deberá indicar la Materia, y opcionalmente alguno de los grupos.

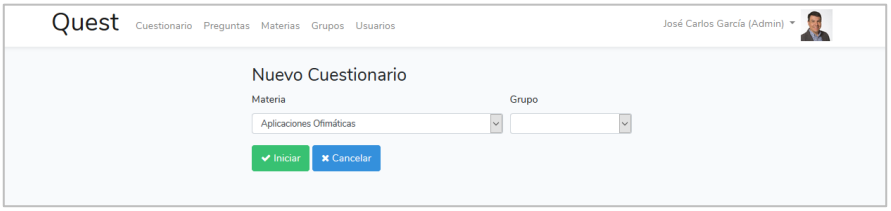

Figura 60. Inicio del cuestionario.

Cada pregunta mostrará cuatro posibles respuestas, y opcionalmente una imagen.

Figura 61. Pregunta del cuestionario.

Al pulsar sobre una de las opciones, se mostrará en verde si es correcta y en rojo si es errónea, y se incrementará el contador correspondiente.

Figura 62. Respuesta correcta del cuestionario.

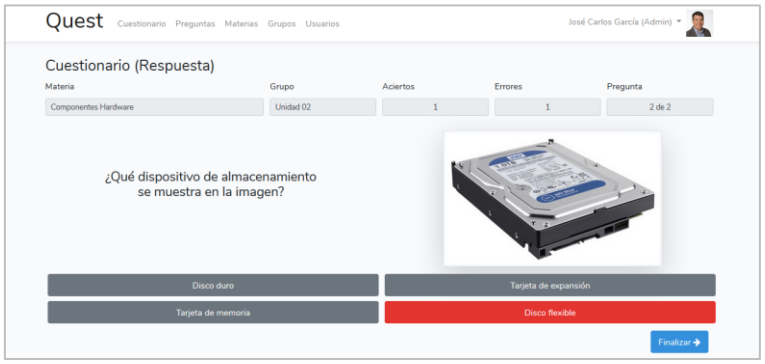

Figura 63. Respuesta errónea del cuestionario.

Al finalizar el cuestionario se indicará el total de preguntas, aciertos y errores.

Figura 64. Resultado final del cuestionario.

En función del porcentaje de aciertos, se mostrará un mensaje y una imagen diferentes.

Figura 65. Imágenes del resultado del cuestionario.

Gestión de Entidades

La gestión de todas las entidades de la aplicación (preguntas, materias, grupos y usuarios) se realiza de la misma forma:

- Una página de listado inicial, que muestra todos los elementos.
- Una página de detalle para las operaciones de creación, consulta y edición.
- Mensaje de confirmación antes de realizar el borrado de un elemento.
- En caso de existir alguna acción adicional, el enlace se mostrará a la derecha de la lista.

Flujo de Navegación

Este es el flujo de navegación para la gestión de preguntas.

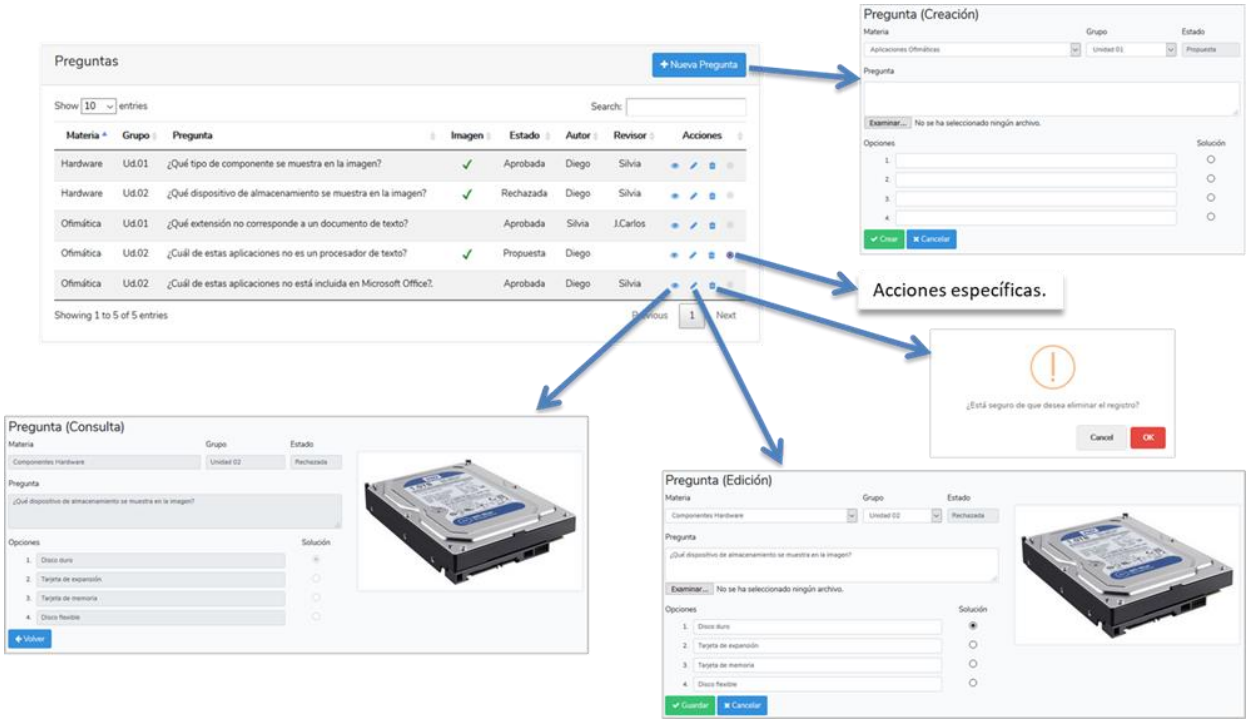

Figura 66. Flujo de navegación.

Página de Listado

Inicialmente se accederá a la página que muestra un listado de todos los elementos.

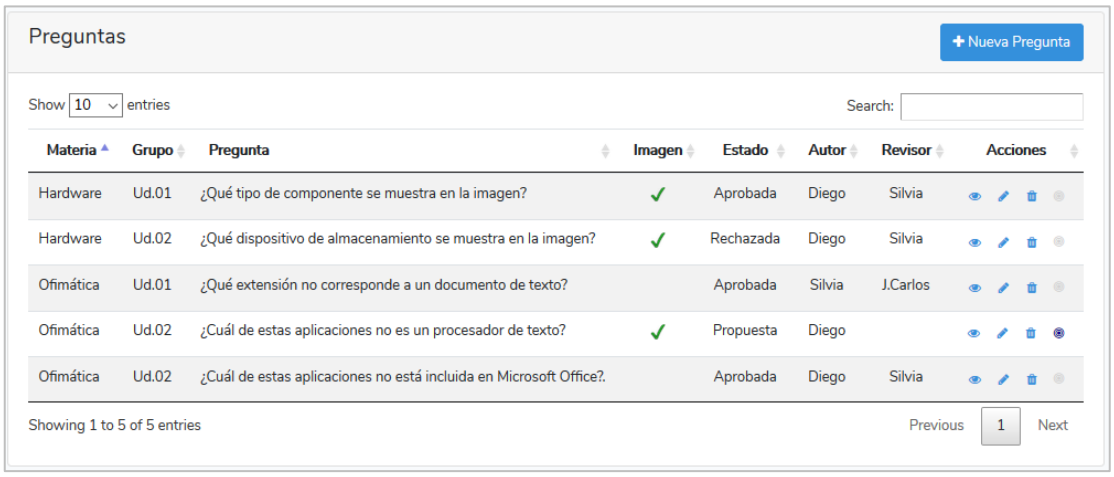

Figura 67. Página de listado.

El usuario podrá realizar las siguientes operaciones:

- Indicar el número de registros que debe mostrar cada página.
- Indicar un filtro (que se aplicará a cualquier columna) para restringir los registros que se muestran.
- Pulsar sobre los títulos de cada columna, para ordenar por dicha columna en ambos sentidos (asc/desc).
- Pulsar sobre los botones de navegación en la parte inferior derecha de la lista.
- Pulsar el botón de la parte superior derecha, para acceder a la página de creación de un nuevo elemento.
- Pulsar sobre los enlaces de acciones de cada fila, para consultar, modificar o borrar el registro.

Página de Creación

Mostrará los campos vacíos, excepto los desplegables que sean obligatorios que mostrarán un valor por defecto. Las acciones disponibles son 'Crear' y 'Cancelar'.

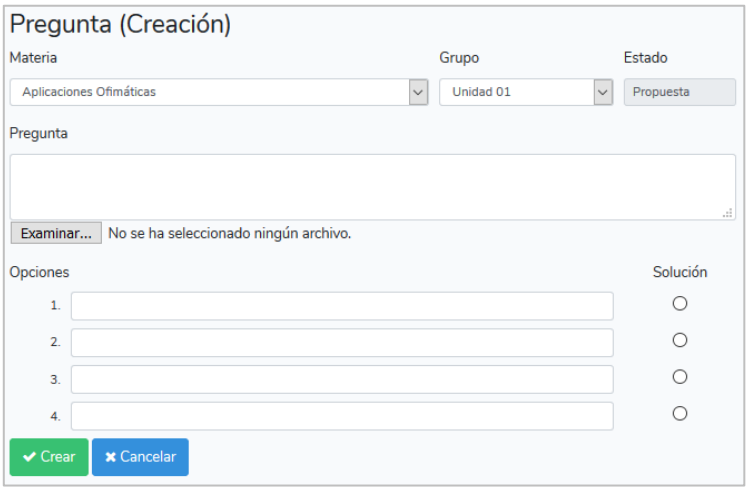

Figura 68. Página de creación.

Confirmación del Borrado

Al pulsar sobre el enlace de borrado se mostrará un mensaje de confirmación antes de ejecutar la acción.

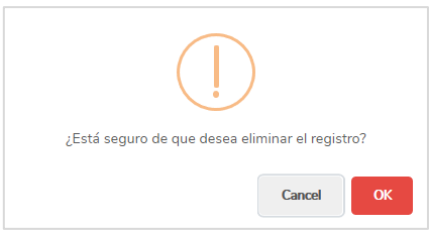

Figura 69. Mensaje de confirmación.

Página de Consulta

Mostrará los campos informados y protegidos. La única acción disponible es 'Volver'.

Figura 70. Página de consulta.

Página de Edición

Mostrará los campos informados, y estarán habilitados o protegidos en función de que puedan ser modificados o no por el usuario. Las acciones disponibles son 'Guardar' y 'Cancelar'.

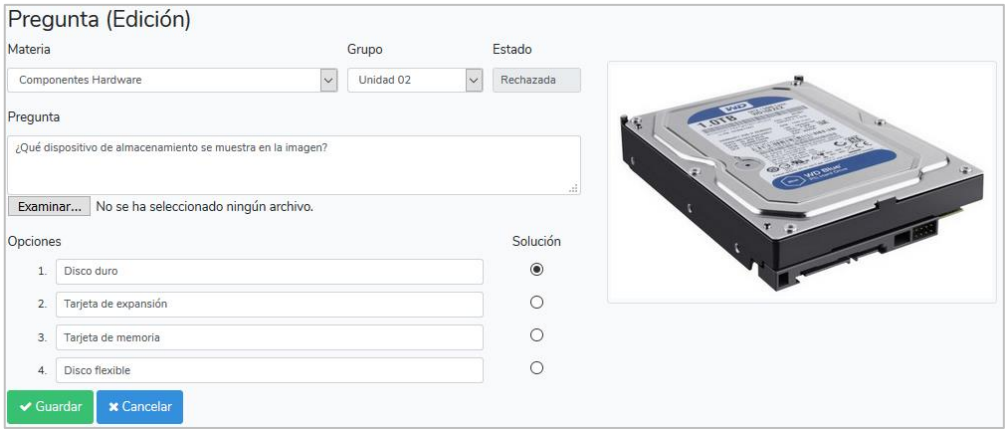

Figura 71. Página de edición.

Gestión Preguntas

La gestión de preguntas tendrá un comportamiento diferente en función del perfil del usuario conectado.

El **Editor** sólo tendrá acceso a sus propias preguntas, que no podrá modificar o borrar si ya están aprobadas.

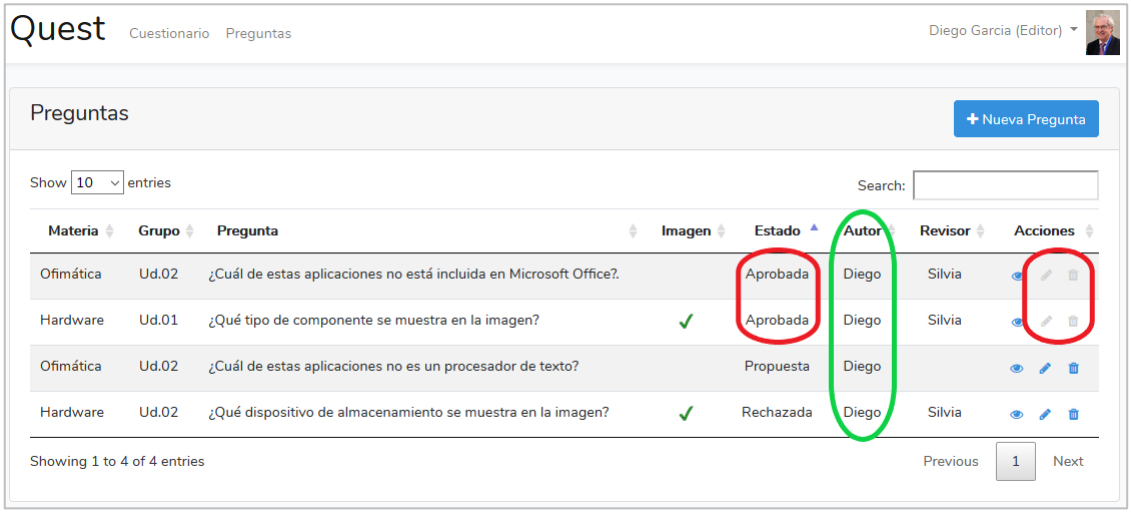

Figura 72. Acciones disponibles para el 'Editor' de preguntas.

El **Revisor** tendrá acceso al listado de preguntas de todos los usuarios, y tendrá disponible la opción 'Revisar', que estará activa para las preguntas en estado 'Propuesta'.

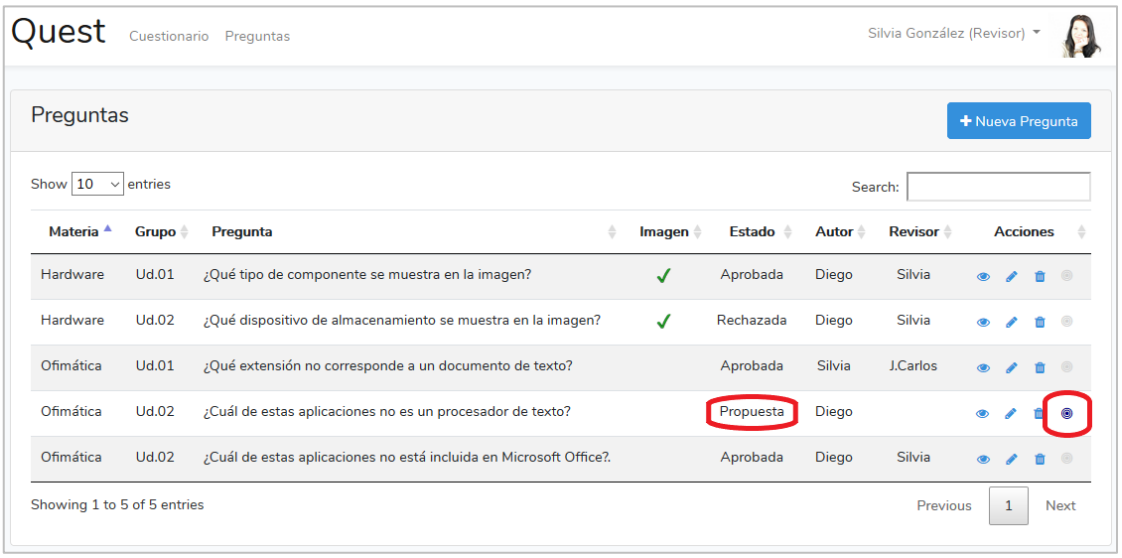

Figura 73. Acciones disponibles para el 'Revisor' de preguntas.

Para revisar una pregunta, se mostrará el detalle correspondiente, teniendo la opción aprobarla, rechazarla, o volver al listado anterior sin realizar ninguna acción.

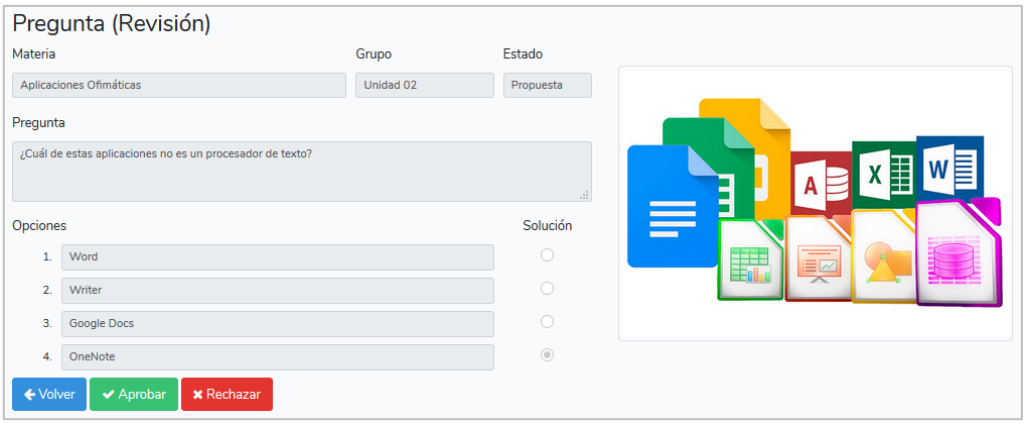

Figura 74. Revisión de una pregunta.

Anexo 8. Glosario

A continuación se muestra el glosario de términos y acrónimos utilizados en el proyecto.

A

Array

Conjunto de elementos de un mismo tipo, organizado en filas y columnas, que pueden ser gestionados de forma conjunta.

D

DELETE

Método HTTP para eliminar un recurso del servidor.

E

Encriptación

Mecanismo de seguridad por el cual la información se vuelve ilegible tras la aplicación de un algoritmo, de forma que sólo podrá ser interpretada si se conoce dicho algoritmo.

G

GET

Método HTTP para la obtención de un recurso del servidor.

H

HTTP

De forma general, es un protocolo para transferencia de hipertexto.

HTTP (Peticiones)

Las peticiones HTTP se refieren a las operaciones que se pueden realizar con HTTP sobre un servidor web.

I

IDE

Entorno de desarrollo integrado, que proporciona el conjunto de herramientas necesarias para el desarrollo de aplicaciones.

L

Layout

Esquema de distribución de los elementos dentro un diseño.

M

MVC

Patrón de arquitectura de software, que separa datos, lógica de negocio y su representación, proponiendo la construcción de tres tipos de componentes: modelo, vista y controlador.

P

POST

Método HTTP para la creación de un recurso en el servidor.

PUT

Método HTTP para cambiar el estado de un recurso del servidor o actualizarlo.

R

RESTful

Se refiere a los servicios web que utilizan los métodos HTTP de manera explícita (POST, GET, PUT, DELETE).

S

Scroll

Se refiere al desplazamiento de los contenidos.

Submit

Envío al servidor de la información contenida en un formulario web.

U

UML

Siglas de Unified Modeling Language. Es un lenguaje gráfico para especificar, construir y documentar un sistema.

Anexo 9. Bibliografía

[1] Laravel LLC. Laravel Installation. https://laravel.com/docs/5.8
Disponible el 25 de septiembre de 2019

[2] Apache Friends. XAMPP. https://www.apachefriends.org/es/index.html
Disponible el 25 de septiembre de 2019

[3] Laravel LLC. Database Migrations. https://laravel.com/docs/5.8/migrations
Disponible el 30 de septiembre de 2019

[4] David Gil Ripoll. Usabilidad y Experiencia de Usuario. http://www.usabilidad-ux.com/
Disponible el 26 de octubre de 2019

[5] Cris Busquets. Principios Gestalt Aplicados al Diseño UI/UX.
https://www.uifrommars.com/principios-gestalt-diseno-web/
Disponible el 26 de octubre de 2019

[6] Laravel LLC. Basic Task List. https://laravel.com/docs/5.2/quickstart
Disponible el 27 de octubre de 2019

[7] Laravel LLC. Intermediate Task List. https://laravel.com/docs/5.2/quickstart-intermediate
Disponible el 30 de octubre de 2019

[8] Laravel LLC. Views. https://laravel.com/docs/5.8/views
Disponible el 5 de noviembre de 2019

[9] Laravel LLC. Controllers. https://laravel.com/docs/5.8/controllers
Disponible el 5 de noviembre de 2019

[10] Laravel LLC. Routing. https://laravel.com/docs/5.8/routing
Disponible el 5 de noviembre de 2019

Anexo 10. Vita

Formación Académica

- **Ingeniería Técnica de Telecomunicaciones**
  Especialidad Equipos Electrónicos
  Universidad de Alcalá de Henares, Madrid. 1992
- **Técnico Superior en Administración de Sistemas Informáticos en Red**
  I.E.S. Politécnico Jesús Marín, Málaga. 2013
- **Técnico Superior en Desarrollo de Aplicaciones Multiplataforma**
  I.E.S. Aguadulce, Almería. 2015

Experiencia Profesional

- **Profesor Técnico de Formación Profesional**
  Especialidad Sistemas y Aplicaciones Informáticas
  Desde 2015
- **Docente Formación Profesional para el Empleo**
  Junta de Andalucía / Servicio de Empleo Público Estatal
  2011 - 2015
- **Responsable de Informática**
  IndeK S.A.
  2007 - 2010
- **Programador / Analista / Jefe de Equipo**
  Coritel S.A.
  1993 - 2006

Publicaciones

- **Diseño de Elementos Software con Tecnologías Basadas en Componentes**
  IC Editorial
  2014