



TRABAJO FIN DE GRADO
INGENIERÍA EN INFORMÁTICA

Sistema de generación de encuestas

Autor

Pablo Lara Padilla (alumno)

Directores

Marcelino Cabrera Cuevas (tutor)



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, Septiembre de 2017

Sistema de generación de encuestas

Pablo Lara Padilla (alumno)

Palabras clave: desarrollo web, diseño responsive, encuestas

Resumen

Este proyecto busca ofrecer a los usuarios una plataforma online para la generación de encuestas de la forma más personalizable posible. Este software busca ser una alternativa al resto de plataformas presentes en el mercado. Además añade la posibilidad de que estas encuestas puedan ser accesibles desde cualquier dispositivo independientemente del tamaño de pantalla del mismo. Para su desarrollo se han utilizado tecnologías tan conocidas como PHP, JavaScript o Bootstrap.

Survey generator system

Pablo, Lara Padilla (student)

Keywords: web development, responsive design, surveys

Abstract

This project aims to offer to users an online platform for generate surveys in the most customizable way possible. This software aims to be an alternative to the rest of platforms available nowadays. Also adds the possibility that these surveys can be accessible from any device regardless of screen size. For its development have used well known technologies like PHP, JavaScript or Bootstrap.

Yo, **Pablo Lara Padilla**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75963255M, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Pablo Lara Padilla

Granada a 11 de Septiembre de 2017 .

D. **Marcelino Cabrera Cuevas (tutor)**, Profesor del Departamento Lenguajes y Sistemas Informaticos de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Sistema de generación de encuestas***, ha sido realizado bajo su supervisión por **Pablo Lara Padilla (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 12 de Septiembre de 2017.

Los directores:

Marcelino Cabrera Cuevas (tutor)

Índice general

1. Introducción	15
2. Especificación de requisitos	17
2.1. Usuarios	17
2.1.1. Requisitos de datos	17
2.1.2. Requisitos funcionales	18
2.1.3. Restricciones semánticas	18
2.2. Gestión espacios de trabajo	18
2.2.1. Requisitos de datos	18
2.2.2. Requisitos funcionales	19
2.2.3. Restricciones semánticas	19
2.3. Gestión de formularios	19
2.3.1. Requisitos de datos	19
2.3.2. Requisitos funcionales	19
2.3.3. Restricciones semánticas	20
2.4. Gestión de preguntas	20
2.4.1. Requisitos de datos	20
2.4.2. Requisitos funcionales	25
2.4.3. Restricciones semánticas	25
2.5. Respuestas/Envíos a un formulario	25
2.5.1. Requisitos de datos	25
2.5.2. Restricciones semánticas	26
3. Análisis	27
3.1. Casos de uso	27
3.1.1. Descripción de los casos de uso	29
3.2. Modelo conceptual	41
4. Diseño	46
4.1. Algunos conceptos antes de comenzar	46
4.2. URLs de la aplicación	48
4.3. Diagramas de clase	52

5. Implementación	64
5.1. Algoritmo para crear/actualizar el tipo Cudrícula	64
5.2. Algoritmo para almacenar una respuesta	65
5.3. Algoritmo para obtener las estadísticas globales de un formulario	66
5.4. Configuración del entorno	68
6. Manual	72
7. Conclusiones y trabajos futuros	81
7.1. Conclusiones	81
7.2. Trabajos futuros	81

Índice de figuras

1.1. Clases implicadas en el controlador WorkspaceController . . .	15
3.1. Diagrama de caso de uso básico del sistema.	27
3.2. Diagrama caso de uso extendido del sistema.	28
3.3. Modelo conceptual del sistema.	41
3.4. Modelo conceptual del sistema.	45
4.1. Clases implicadas en el controlador WorkspaceController . . .	52
4.2. Clases implicadas en el controlador WorkspaceController . . .	54
4.3. Clases implicadas en el controlador WorkspaceController . . .	56
4.4. Diagrama de clases de la relación entre los modelos y sus eventos	60
4.5. Clases implicadas en el controlador WorkspaceController . . .	62
6.1. Página de inicio de la aplicación.	72
6.2. Página para identificarse.	73
6.3. Página para registrarse.	73
6.4. Página principal de un espacio de trabajo.	74
6.5. Editar el nombre de un espacio de trabajo.	74
6.6. Editar el nombre de un espacio de trabajo.	75
6.7. Barra que muestra nuestros espacios de trabajo.	75
6.8. Modal para crear un nuevo espacio de trabajo.	76
6.9. Crear un nuevo formulario.	76
6.10. Página para diseñar nuestros formularios.	77
6.11. Modal para crear un nuevo espacio de trabajo.	77
6.12. Cambiar el nombre de un formulario.	78
6.13. Vista para compartir el formulario.	78
6.14. Gráficas de las estadísticas.	79
6.15. Ver una respuesta de forma individual.	79
6.16. Formulario público.	80

Índice de cuadros

3.1. Descripción CU.1 - Registrarse	29
3.2. Descripción CU.2 - Identificarse	30
3.3. Descripción CU.3 - Gestión de espacios de trabajo	30
3.4. Descripción CU.3.1 - Crear un espacio de trabajo	31
3.5. Descripción CU.3.2 - Cambiar el nombre del espacios de trabajo	32
3.6. Descripción CU.3.3 - Eliminar espacio de trabajo	32
3.7. Descripción CU.3.4 - Añadir un formulario	33
3.8. Descripción CU.3.5 - Eliminar un formulario	34
3.9. Descripción CU.3.6 - Acceder a un formulario	34
3.10. Descripción CU.4 - Gestión de formularios	34
3.11. Descripción 4.1 - Cambiar nombre de un formulario	35
3.12. Descripción 4.2 - Compartir formulario	35
3.13. Descripción 4.3 - Analizar	36
3.14. Descripción 4.3.1 - Ver estadísticas	36
3.15. Descripción 4.3.2 - Ver respuestas individualmente	37
3.16. Descripción 4.4 - Gestionar preguntas	37
3.17. Descripción 4.4.1 - Añadir pregunta	38
3.18. Descripción 4.4.2 - Eliminar pregunta	39
3.19. Descripción 4.4.3 - Modificar pregunta	39
3.20. Descripción 4.4.4 - Cambiar posición de una pregunta	40

Capítulo 1

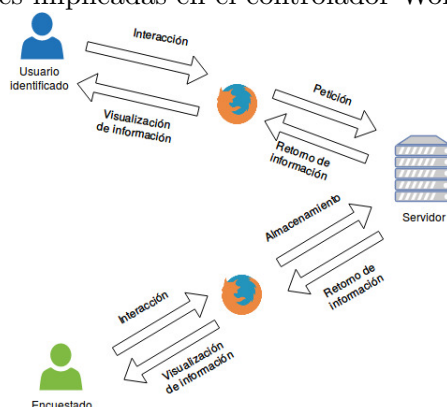
Introducción

La finalidad del proyecto es crear un sistema que permita la generación de encuestas por parte del usuario. El sistema deberá permitir que las encuestas se adapten lo máximo posible al tipo de usuario que va a rellenar la encuesta.

Para lograr que la aplicación pueda llegar al mayor número de usuarios, esta se va a desarrollar como una aplicación web. La aplicación dispondrá de dos partes bien diferenciadas: una parte para la gestión y generación de las encuestas por parte del usuario administrador, y otra parte en la que los usuarios podrán responder las preguntas generadas por el primero. Para que esta última parte no tenga limitaciones en cuanto al público que debe llegar, se desarrollará para que se adapte al máximo número de dispositivos del mercado, tratándose por ello de una interfaz *responsive*.

Básicamente, la arquitectura que debe seguir la aplicación es la que se muestra en la siguiente figura:

Figura 1.1: Clases implicadas en el controlador WorkspaceController



Debido a la variedad de tipos de pregunta que encontramos en una encuesta, será de vital importancia que el sistema se pueda extender con facilidad si surge el caso de que se deba añadir un nuevo tipo de pregunta con

unas características muy específicas.

Para comenzar con el desarrollo debo familiarizarme con los distintos sistemas de generación de encuestas que hay en la actualidad, y ver que funcionalidades ofrece al usuario, para así obtener un mínimo de requisitos a cumplir. Con todo esto, comencemos con el desarrollo del proyecto.

Capítulo 2

Especificación de requisitos

Para intentar describir de forma clara los requisitos que comprenden la aplicación, estos aparecen estructurados según el recurso que se gestiona y distinguiendo entre requisitos de datos, requisitos funcionales y restricciones semánticas aplicadas a dicho recurso según sea el caso. Además, en el caso de los requisitos de datos, para cada elemento a almacenar se especifica el tipo de requisito de dato, distinguiendo entre:

- Requisito de dato de entrada (RDE): son datos de entrada requeridos bien para ser almacenados o para ser procesados por la lógica del sistema.
- Requisito de dato de almacenamiento (RDA): datos que son almacenados de forma permanente en el sistema para su posterior procesamiento o salida.
- Requisito de dato de salida (RDS): datos que se mostrarán al usuario del sistema en un momento dado.

2.1. Usuarios

En el sistema solo hay un tipo de usuario. Este tipo de usuario debe poder gestionar espacios de trabajo, los cuales contendrán los formularios que puede diseñar el usuario.

2.1.1. Requisitos de datos

RD1. **Usuario:** un usuario se describe mediante:

- ID de usuario. **RDA.**
- Contraseña asociada al usuario. **RDE, RDA.**
- Fecha de creación. **RDA.**

- Fecha de la última actualización. **RDA**.
- Nombre del usuario. **RDE, RDA, RDS**.
- Email. **RDE, RDA, RDS**.
- ID del espacio de trabajo por defecto. **RDA**.

2.1.2. Requisitos funcionales

- **Usuarios:**

RF1. Registrar usuario.

RF2. Identificar usuario.

2.1.3. Restricciones semánticas

RS1. Cada ID es única y si un usuario es registrado después que otro su ID tiene que ser mayor.

RS2. El email del usuario ha de ser válido.

RS3. Todo usuario tiene que tener espacio de trabajo por defecto.

2.2. Gestión espacios de trabajo

Un espacio de trabajo representa una entidad contenedora de formularios, para la gestión de estos. Los espacios de trabajo permiten mantener de forma más estructurada y ordenada los formularios del usuario. Un usuario podrá tener tantos espacios de trabajo como desee, y por defecto tendrá asignado uno que no podrá ser eliminado.

2.2.1. Requisitos de datos

RD2. **Espacio de trabajo:** un espacio de trabajo se describe mediante:

- ID de espacio de trabajo. **RDA**.
- Nombre del espacio de trabajo. **RDE, RDA, RDS**.
- ID del usuario al que pertenece por defecto. **RDA**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

2.2.2. Requisitos funcionales

- **Espacios de trabajo:**

RF3. Crear un espacio de trabajo.

RF4. Actualizar el nombre de un espacio de trabajo.

RF5. Eliminar un espacio de trabajo.

RF6. Añadir un nuevo formulario al espacio de trabajo.

RF7. Eliminar un formulario del espacio de trabajo.

RF8. Acceder a un formulario.

2.2.3. Restricciones semánticas

RS4. Cada ID es única y si un espacio de trabajo creado después que otro su ID tiene que ser mayor.

RS5. El nombre del espacio de trabajo debe ser único para un mismo usuario.

RS6. El espacio de trabajo por defecto de un usuario no puede ser eliminado.

2.3. Gestión de formularios

Un formulario está compuesto de múltiples preguntas de distinto tipo, permitiendo la gestión de estas. Para cada formulario es posible ver las estadísticas existentes para ciertos tipos de pregunta, además de ver las respuestas de forma individual.

2.3.1. Requisitos de datos

RD3. **Formulario:** un formulario se describe mediante:

- ID de formulario. **RDA.**
- Nombre del formulario. **RDE, RDA, RDS.**
- ID del espacio de trabajo al que pertenece. **RDA.**
- Fecha de creación. **RDA.**
- Fecha de la última actualización. **RDA.**

2.3.2. Requisitos funcionales

- **Formularios:**

RF9. Actualizar el nombre de un formulario.

RF10. Añadir preguntas/texto al formulario.

- RF11. Eliminar preguntas/texto del formulario.
- RF12. Cambiar la posición de una pregunta/texto en el formulario.
- RF13. Ver URL pública del formulario.
- RF14. Ver estadísticas del formulario.
- RF15. Ver respuestas individuales del formulario.

2.3.3. Restricciones semánticas

- RS7. Cada ID es única y si un formulario es creado después que otro su ID tiene que ser mayor.
- RS8. El nombre del formulario debe ser único para un mismo espacio de trabajo.

2.4. Gestión de preguntas

En este sistema, las preguntas pueden tomar distinto tipo según las necesidades del diseñador de las encuestas. Cada una de estas preguntas tendrá unas características comunes a todas y otras específicas a si misma. Ciertas preguntas pueden requerir la existencia de elementos externos tales como opciones a elegir. Esta relación está ligada al tipo de pregunta y no a la pregunta en sí.

2.4.1. Requisitos de datos

RD4. **Pregunta:** una pregunta se describe mediante:

- ID de pregunta. **RDA**.
- Texto de la pregunta. **RDE, RDA, RDS**.
- Descripción de la pregunta. **RDE, RDA, RDS**.
- Icono del tipo de pregunta. **RDE, RDA, RDS**.
- Posición de la pregunta en el formulario. **RDE, RDA**.
- Tipo de pregunta **RDA**.
- ID de la pregunta del tipo específico. **RDA**.
- ID del formulario al que pertenece. **RDA**.
- ID del usuario que la actualizó la última vez. **RDA**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

A continuación se van a describir los requisitos de datos que deben cumplir los distintos tipos que puede tomar una pregunta.

RD5. **Tipo Respuesta corta:** una pregunta del tipo 'Respuesta corta' se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Número máximo de caracteres. **RDE, RDA, RDS**.
- Obligatoriedad de la pregunta. **RDE, RDA, RDS**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

RD6. **Tipo Respuesta larga:** una pregunta del tipo 'Respuesta larga' se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Número máximo de caracteres. **RDE, RDA, RDS**.
- Obligatoriedad de la pregunta. **RDE, RDA, RDS**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

RD7. **Tipo Varias respuestas:** una pregunta del tipo 'Varias respuestas' se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Obligatoriedad de la pregunta. **RDE, RDA, RDS**.
- Selección múltiple. **RDE, RDA, RDS**.
- Mostrar opciones en orden aleatorio. **RDE, RDA, RDS**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

RD8. **Tipo 'Desplegable':** una pregunta del tipo desplegable se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Obligatoriedad de la pregunta. **RDE, RDA, RDS**.
- Mostrar opciones alfabéticamente. **RDE, RDA, RDS**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

RD9. **Tipo Cuadrícula:** una pregunta del tipo 'Cuadrícula' se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Obligatoriedad de la pregunta. **RDE, RDA, RDS**.

- Selección múltiple. **RDE, RDA, RDS.**
- Fecha de creación. **RDA.**
- Fecha de la última actualización. **RDA.**

RD10. **Tipo 'Escala de opinión':** una pregunta del tipo 'Escala de opinión' se describe mediante:

- ID de tipo de pregunta. **RDA.**
- Obligatoriedad de la pregunta. **RDE, RDA, RDS.**
- Valor mínimo del rango. **RDE, RDA, RDS.**
- Valor máximo del rango. **RDE, RDA, RDS.**
- Fecha de creación. **RDA.**
- Fecha de la última actualización. **RDA.**

RD11. **Tipo Número:** una pregunta del tipo 'Número' se describe mediante:

- ID de tipo de pregunta. **RDA.**
- Obligatoriedad de la pregunta. **RDE, RDA, RDS.**
- Valor mínimo del rango. **RDE, RDA, RDS.**
- Valor máximo del rango. **RDE, RDA, RDS.**
- Fecha de creación. **RDA.**
- Fecha de la última actualización. **RDA.**

RD12. **Tipo Email:** una pregunta del tipo 'Email' se describe mediante:

- ID de tipo de pregunta. **RDA.**
- Obligatoriedad de la pregunta. **RDE, RDA, RDS.**
- Fecha de creación. **RDA.**
- Fecha de la última actualización. **RDA.**

RD13. **Tipo Página web:** una pregunta del tipo 'Página web' se describe mediante:

- ID de tipo de pregunta. **RDA.**
- Obligatoriedad de la pregunta. **RDE, RDA, RDS.**
- Fecha de creación. **RDA.**
- Fecha de la última actualización. **RDA.**

RD14. **Tipo Fecha:** una pregunta del tipo 'Fecha' se describe mediante:

- ID de tipo de pregunta. **RDA.**
- Obligatoriedad de la pregunta. **RDE, RDA, RDS.**

- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

RD15. **Tipo Hora:** una pregunta del tipo 'Hora' se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Obligatoriedad de la pregunta. **RDE, RDA, RDS**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

RD16. **Tipo Si/No:** una pregunta del tipo 'Si/No' se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Obligatoriedad de la pregunta. **RDE, RDA, RDS**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

RD17. **Tipo Legal:** una pregunta del tipo 'Legal' se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Obligatoriedad de la pregunta. **RDE, RDA, RDS**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

RD18. **Tipo Texto:** verdaderamente este no es un tipo de pregunta, sin embargo por cuestiones de diseño decidí tratarla como tal pues toma los atributos básico de una pregunta: un texto y una descripción. Se describe mediante:

- ID de tipo de pregunta. **RDA**.
- Fecha de creación. **RDA**.
- Fecha de la última actualización. **RDA**.

Tras haber descrito los atributos que deben tener tanto una pregunta como los tipos específicos de pregunta, se describen a continuación los requisitos de datos de elementos que puede contener un tipo de pregunta específico.

RD19. **Opción:** una opción puede estar asociada a los tipos de pregunta 'Desplegable' y 'Varias respuestas', y representa a una de las respuestas posibles que puede tomar esa pregunta. Se describe mediante:

- ID de opción. **RDA**.
- Valor que toma la opción. **RDE, RDA, RDS**.

- Posición de la opción en la lista. **RDA**.
- ID de la pregunta de un tipo específico. **RDA**.
- Tipo de la pregunta a la que pertenece. **RDA**.
- Fecha de la última actualización. **RDA**.

RD20. **Fila:** una fila puede estar asociada al tipo de pregunta 'Cuadrícula' y representa a una de las filas posibles de esa pregunta. Se describe mediante:

- ID de fila. **RDA**.
- Valor que toma la opción. **RDE, RDA, RDS**.
- Posición de la fila en la lista. **RDA**.
- ID de la pregunta de tipo 'Cuadrícula'. **RDA**.
- Fecha de la última actualización. **RDA**.

RD21. **Columna:** una columna puede estar asociada al tipo de pregunta 'Cuadrícula' y representa a una de las columnas posibles de esa pregunta. Se describe mediante:

- ID de columna. **RDA**.
- Valor que toma la columna. **RDE, RDA, RDS**.
- Posición de la columna en la lista. **RDA**.
- ID de la pregunta de tipo 'Cuadrícula'. **RDA**.
- Fecha de la última actualización. **RDA**.

RD22. **Imagen:** una imagen puede estar asociada a una pregunta y se describe mediante:

- ID de imagen. **RDA**.
- Nombre del fichero en el sistema. **RDE, RDA, RDS**.
- Nombre original del fichero. **RDA, RDS**.
- Tipo MIME del fichero'. **RDA**.
- Tipo de pregunta a la que pertenece **RDA**.
- ID de la pregunta del tipo específico a la que pertenece. **RDA**.
- Fecha de la última actualización. **RDA**.

RD23. **Vídeo:** un vídeo puede estar asociado a una pregunta y se describe mediante:

- ID de video. **RDA**.
- URL del video. **RDE, RDA, RDS**.
- Tipo de pregunta a la que pertenece **RDA**.
- ID de la pregunta del tipo específico a la que pertenece. **RDA**.
- Fecha de la última actualización. **RDA**.

2.4.2. Requisitos funcionales

- **Preguntas:**

RF16. Actualizar valor de los atributos de una pregunta.

RF17. Añadir una imagen/vídeo a la pregunta.

2.4.3. Restricciones semánticas

- **Pregunta:**

RS9. Cada ID es única y si una pregunta es creada después que otra su ID tiene que ser mayor.

RS10. Una pregunta debe tener un texto con la pregunta.

RS11. Una pregunta debe tomar un tipo específico entre los disponibles.

2.5. Respuestas/Envíos a un formulario

Una respuesta representa al envío realizado por un encuestado a un determinado formulario. Esta respuesta estará formada por las respuestas a las múltiples preguntas del formulario. Para una misma pregunta, pueden existir varias repuestas a dicha pregunta asociadas al envío. Esto es así porque existen tipos de preguntas que permiten la selección de varias opciones como respuesta.

2.5.1. Requisitos de datos

RD24. **Respuesta:** una respuesta representa a un envío individual de un formulario. Está formado por las respuestas a cada una de las preguntas que el encuestado a respondido del formulario:

- ID de respuesta. **RDA.**
- ID del formulario al que pertenece. **RDA.**
- Fecha de creación. **RDA.**
- Fecha de la última actualización. **RDA.**

RD25. **Respuesta a una pregunta:** está asociada a una respuesta (envío) de un formulario concreto, y almacena el valor que el encuestado a dado para una pregunta concreta del formulario:

- ID de respuesta a pregunta. **RDA.**
- ID de de la respuesta/envío a la que pertenece. **RDA.**
- ID de pregunta a la que pertenece. **RDA.**

- Valor de la respuesta para la pregunta a la que pertenece. **RDE, RDA, RDS.**
- Fecha de creación. **RDA.**
- Fecha de la última actualización. **RDA.**

2.5.2. Restricciones semánticas

- **Respuesta/envío:**

RS12. Cada ID es única y si una respuesta/envío es creada después que otra su ID tiene que ser mayor.

RS13. Debe estar asociado a un formulario concreto.

- **Respuesta a una pregunta:**

RS14. Cada ID es única y si una respuesta a una pregunta es creada después que otra su ID tiene que ser mayor.

RS15. Si se trata de una respuesta a una pregunta del tipo 'Cuadrícula', debe estar relacionada con una fila de la cuadrícula.

RS16. Puede haber varias respuestas a una pregunta dentro de una misma repuesta/envío a un formulario. Esto es así para las preguntas que permiten seleccionar múltiples opciones.

Capítulo 3

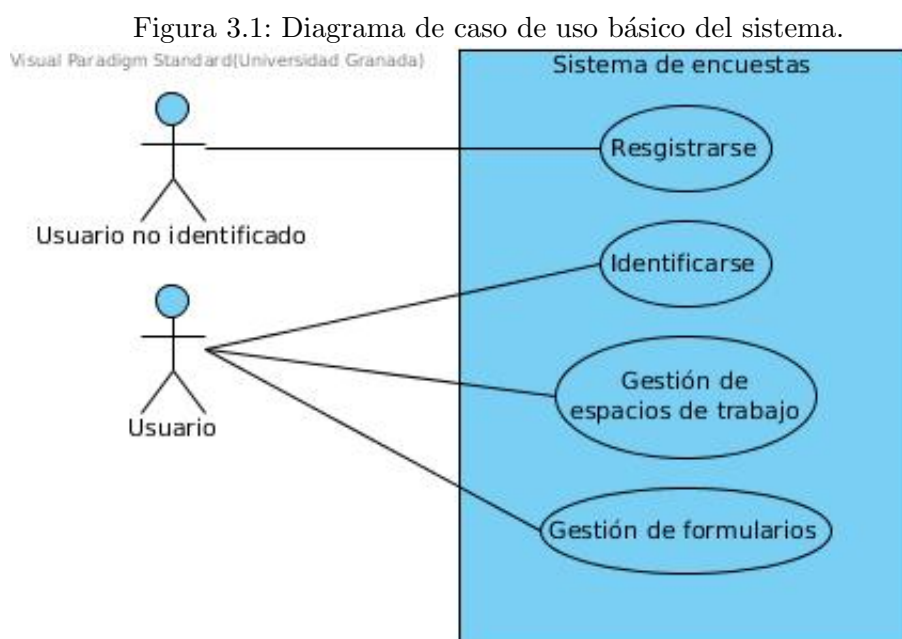
Análisis

Tras la especificación de requisitos, voy a continuar con el análisis del diseño propuesto para ir desarrollando qué elementos son necesarios para desarrollar el sistema. Para ello voy a hacer uso de herramientas de modelado que he visto durante el grado.

3.1. Casos de uso

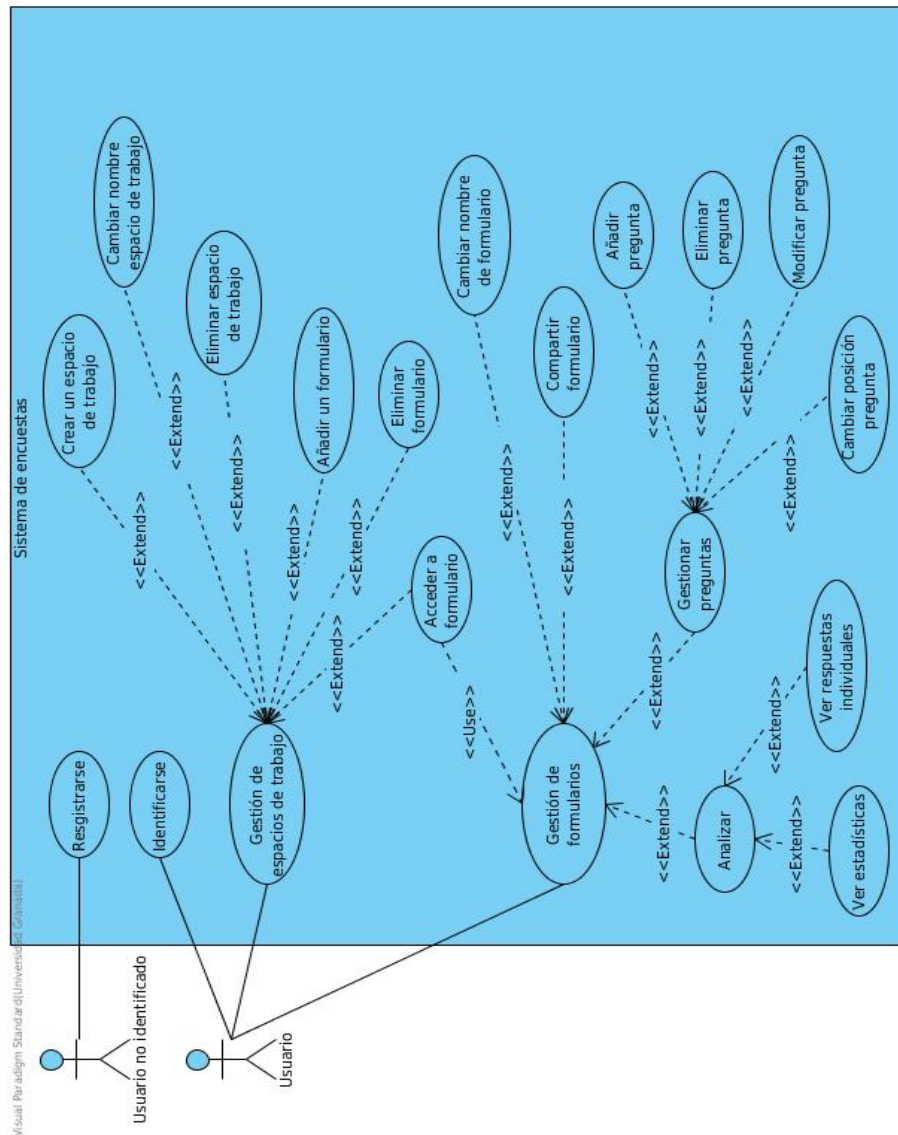
Para comenzar, a continuación se muestran los casos de uso que se han determinado a partir de la especificación de requisitos de apartados anteriores.

La siguiente figura muestra los casos de uso más básicos del sistema:



A continuación, se muestra un diagrama de casos de uso más detallado, añadiendo toda la funcionalidad restante al diagrama de casos de uso anterior:

Figura 3.2: Diagrama caso de uso extendido del sistema.



3.1.1. Descripción de los casos de uso

Tras haber determinado los casos de uso del sistema, voy a continuar con una descripción detallada de cada uno de ellos.

CU. 1	Registrarse	
Actores	Usuario no identificado	
Descripción	Un usuario no identificado debe tener la capacidad de registrarse en el sistema como nuevo usuario si aún no dispone de una cuenta.	
Secuencia normal	Paso	Acción
	1	El usuario no identificado hace clic en el enlace de la web principal para registrarse.
	2	Aparece la pantalla para el registro.
	3	El usuario introduce sus datos: nombre de usuario, email y contraseña.
Postcondición	4	El usuario hace clic sobre el botón de registrarse.
	Se crea en el sistema un usuario con los datos introducidos, se le asigna un espacio de trabajo por defecto y se le redirige a la página de gestión de este espacio de trabajo.	
Excepciones	Paso	Acción
	3	Si alguno de los datos es determinado erróneo por la validación.
Comentarios	Un dato introducido es incorrecto si: el email introducido ya está en uso o el email introducido no es un email.	

Cuadro 3.1: Descripción CU.1 - Registrarse

CU. 2	Identificarse	
Actores	Usuario no identificado	
Descripción	Un usuario no identificado debe tener la capacidad de identificarse en el sistema.	
Secuencia normal	Paso	Acción
	1	El usuario no identificado hace clic en el enlace de la web principal para identificarse.
	2	Aparece la pantalla para de identificación.
	3	El usuario introduce sus credenciales: email y contraseña.
	4	El usuario hace clic sobre el botón de identificarse.
Postcondición	El sistema comprueba las credenciales un usuario con los datos introducidos y le redirecciona a su espacio de trabajo por defecto.	
Excepciones	Paso	Acción
	4	Si las credenciales no son correctas, se le redirige a la pantalla de identificación de nuevo y se le muestra un mensaje indicando que los datos introducidos no son correctos.
Comentarios	Un dato introducido es incorrecto si: <ol style="list-style-type: none"> 1. El email introducido no está en la base de datos 2. La contraseña es errónea. 	

Cuadro 3.2: Descripción CU.2 - Identificarse

CU. 3	Gestión de espacios de trabajo
Actores	Usuario
Descripción	Un usuario identificado puede gestionar los espacios de trabajo de su cuenta de usuario.

Cuadro 3.3: Descripción CU.3 - Gestión de espacios de trabajo

CU. 3.1	Crear un espacio de trabajo	
Actores	Usuario	
Descripción	Un usuario puede crear un nuevo espacio de trabajo.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic sobre el botón para crear un nuevo espacio de trabajo.
	2	Aparece una pantalla en la que se debe indicar el nombre del espacio de trabajo.
	3	El usuario escribe el nombre del espacio de trabajo.
	4	El usuario hace clic sobre el botón para crear el nuevo espacio de trabajo.
Postcondición	Se crea en el sistema un nuevo espacio de trabajo asociado al usuario y con el nombre que este ha establecido.	
Excepciones	Paso	Acción
	4	Si el nombre del espacio de trabajo ya está en uso por otro espacio de trabajo del usuario se deniega la creación.
Comentarios	Un nombre introducido es incorrecto si: <ol style="list-style-type: none"> 1. El nombre introducido ya está en uso. 	

Cuadro 3.4: Descripción CU.3.1 - Crear un espacio de trabajo

CU. 3.2	Cambiar el nombre de un espacio de trabajo	
Actores	Usuario	
Descripción	Un usuario puede cambiar el nombre del espacio de trabajo actual.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic sobre el nombre del espacio de trabajo actual.
	2	Aparece un formulario para introducir el nombre.
	3	El usuario pulsa al botón para cambiar el nombre.
Postcondición	Se actualiza el nombre del espacio de trabajo en el sistema.	
Excepciones	Paso	Acción
	3	Si el nombre ya está en uso aparece un mensaje de error.
Comentarios	Un nombre introducido es incorrecto si: <ol style="list-style-type: none"> 1. El nombre introducido ya está en uso. 	

Cuadro 3.5: Descripción CU.3.2 - Cambiar el nombre del espacios de trabajo

CU. 3.3	Eliminar espacio de trabajo	
Actores	Usuario	
Descripción	Un usuario puede eliminar un espacio de trabajo entre los que tiene asociados.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic en el botón de eliminar el espacio de trabajo.
	2	Aparece un mensaje de aviso para confirmar la eliminación.
	3	El usuario hace clic sobre el botón de confirmar la eliminación.
Postcondición	Se elimina del sistema el espacio de trabajo.	
Excepciones	Paso	Acción
	1	Si el espacio a eliminar es el espacio de trabajo por defecto, el resto de pasos no se llevan a cabo.

Cuadro 3.6: Descripción CU.3.3 - Eliminar espacio de trabajo

CU. 3.4	Añadir un formulario	
Actores	Usuario	
Descripción	Un usuario puede añadir un nuevo formulario al espacio de trabajo actual.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic sobre el botón de crear un formulario.
	2	Aparece la pantalla para la creación de un nuevo formulario.
	3	El usuario introduce el nombre del nuevo formulario.
	4	El usuario hace clic sobre el botón de crear.
Postcondición	Se crea en el sistema un nuevo formulario con asociado al espacio de trabajo actual con el nombre que el usuario ha indicado	
Excepciones	Paso	Acción
	3	Si el nombre introducido ya está en uso para un formulario del espacio de trabajo no se lleva a cabo la acción y aparece un mensaje indicandolo.

Cuadro 3.7: Descripción CU.3.4 - Añadir un formulario

CU. 3.5	Eliminar un formulario	
Actores	Usuario	
Descripción	Un usuario puede eliminar un formulario del espacio de trabajo actual.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic en el botón para eliminar un formulario determinado.
	2	Aparece la pantalla de advertencia para confirmar la eliminación.
	3	El usuario hace clic sobre el botón para confirmar la eliminación.
Postcondición	El formulario es eliminado del sistema junto con las preguntas que tuviera asociadas.	
Excepciones	Paso	Acción
	3	Si el formulario no existe en el sistema.

Cuadro 3.8: Descripción CU.3.5 - Eliminar un formulario

CU. 3.6	Acceder a un formulario	
Actores	Usuario	
Descripción	Un usuario puede acceder a un formulario desde espacio de trabajo actual.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic sobre formulario.
Comentarios	El sistema redireccionará al usuario a la vista del formulario.	

Cuadro 3.9: Descripción CU.3.6 - Acceder a un formulario

CU. 4	Gestión de formularios	
Actores	Usuario	
Descripción	Un usuario puede gestionar los formularios que sus espacios de trabajo tiene asociado.	

Cuadro 3.10: Descripción CU.4 - Gestión de formularios

CU. 4.1	Cambiar nombre de un formulario	
Actores	Usuario	
Descripción	Un usuario puede cambiar el nombre del formulario actual.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic sobre el nombre del formulario actual.
	2	Aparece un formulario para introducir el nombre.
	3	El usuario pulsa al botón para cambiar el nombre.
Postcondición	Se actualiza el nombre del espacio de trabajo en el sistema.	
Excepciones	Paso	Acción
	3	Si ya existe un formulario asociado al mismo espacio de trabajo con ese nombre aparece un mensaje de error.
Comentarios	Un nombre introducido es incorrecto si: <ol style="list-style-type: none"> 1. El nombre introducido ya está en uso el mismo espacio de trabajo. 	

Cuadro 3.11: Descripción 4.1 - Cambiar nombre de un formulario

CU. 4.2	Compartir formulario	
Actores	Usuario	
Descripción	Un usuario puede obtener la URL que identifica de forma pública al formulario para poder ser rellenado por los posibles encuestados.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic en el enlace del panel que lleva a la pantalla de compartir.
	2	Aparece la pantalla con la URL para poder ser copiada.

Cuadro 3.12: Descripción 4.2 - Compartir formulario

CU. 4.3	Analizar	
Actores	Usuario	
Descripción	Un usuario puede visualizar y analizar los resultados que ha recibido para una determinada encuesta.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic en el enlace del panel que lleva a la pantalla de analizar.
	2	Aparece la pantalla el panel para visualizar los resultados obtenidos.

Cuadro 3.13: Descripción 4.3 - Analizar

CU. 4.3.1	Ver estadísticas	
Actores	Usuario	
Descripción	Un usuario puede ver las estadísticas globales de ciertos tipos de respuestas de las que se pueden realizar calculos y mostrar gráficas estadísticas.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic en el botón para visualizar las estadísticas.
	2	Aparece la pantalla con las estadísticas de las respuestas recogidas hasta el momento.
Excepciones	Paso	Acción
	2	Si aún no se ha almacenado ninguna respuesta, no aparecen las estadísticas.

Cuadro 3.14: Descripción 4.3.1 - Ver estadísticas

CU. 4.3.2	Ver respuestas individualmente	
Actores	Usuario	
Descripción	Un usuario puede ver las respuesta/envios enviadas por los encuestados de forma individual, pasando de respuesta/envío de uno en uno.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic en el botón para visualizar las respuesta individualmente.
	2	Aparece la pantalla con las respuestas a la preguntas para una respuesta/envío concreto.
Excepciones	Paso	Acción
	2	Si aún no se ha almacenado ninguna respuesta, no aparecen las estadísticas.

Cuadro 3.15: Descripción 4.3.2 - Ver respuestas individualmente

CU. 4.4	Gestionar preguntas
Actores	Usuario
Descripción	Un usuario puede gestionar las preguntas contenidas en un formulario: añadir, actualizar... .

Cuadro 3.16: Descripción 4.4 - Gestionar preguntas

CU. 4.4.1	Añadir pregunta	
Actores	Usuario	
Descripción	Un usuario puede añadir una nueva pregunta al formulario actual. Esta pregunta puede ser de uno de los tipos presentes.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic sobre uno de los tipos de pregunta.
	2	Aparece la pantalla para la introducción de los atributos de la pregunta.
	3	El usuario introduce los datos que desee y selecciona la configuración de la pregunta.
	4	El usuario hace clic sobre el botón de crear.
Postcondición	Se crea en el sistema una nueva pregunta asociada al formulario actual	
Excepciones	Paso	Acción
	4	Si no se ha introducido un texto para la pregunta se indica con un error antes del envío.
Comentarios	Las preguntas creadas se sitúan al final de la cola del resto de preguntas del formulario	

Cuadro 3.17: Descripción 4.4.1 - Añadir pregunta

CU. 4.4.2	Eliminar pregunta	
Actores	Usuario	
Descripción	Un usuario puede eliminar una pregunta del formulario actual.	
Secuencia normal	Paso	Acción
	1	El usuario identificado hace clic sobre el botón de eliminar la pregunta.
	2	Aparece una pantalla para confirmar la eliminación de la pregunta.
	3	El usuario hace clic sobre el botón de confirmar.
Postcondición	Se elimina del sistema la pregunta junto a todas las respuestas que tuviera asociada.	

Cuadro 3.18: Descripción 4.4.2 - Eliminar pregunta

CU. 4.4.3	Modificar pregunta	
Actores	Usuario	
Descripción	Un usuario puede modificar los atributos y configuración de una pregunta del formulario actual.	
Secuencia normal	Paso	Acción
	1	El usuario hace clic sobre la pregunta.
	2	Aparece una pantalla con los valores actuales de los atributos y configuración.
	3	El usuario modifica los datos que crea convenientes.
	4	El usuario hace clic sobre el botón de guardar.
Postcondición	Se actualiza la pregunta en el sistema con los valores que ha pasado el usuario.	
Excepciones	Paso	Acción
	4	Si no se ha introducido un valor para un campo requerido, se indica mediante un error y se cancela envío de la actualización.

Cuadro 3.19: Descripción 4.4.3 - Modificar pregunta

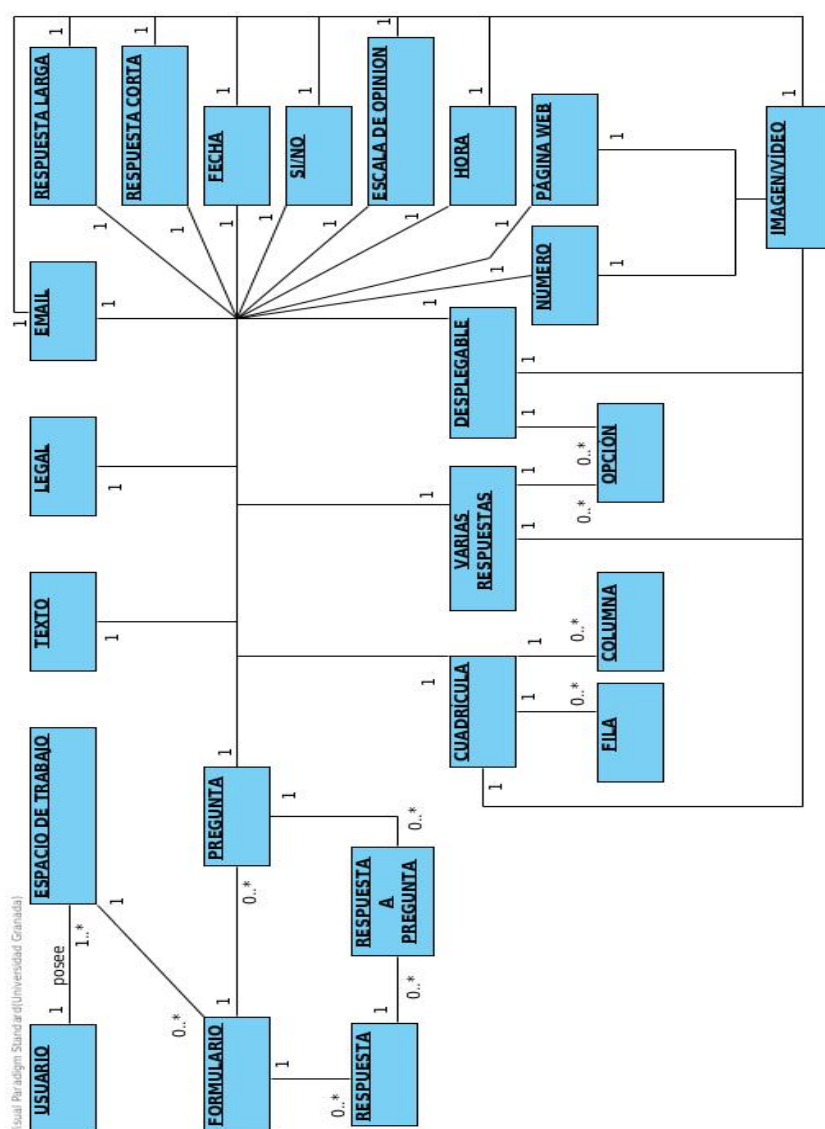
CU. 4.4.4	Cambiar posición de una pregunta	
Actores	Usuario	
Descripción	Un usuario puede cambiar la posición de una pregunta en el formulario actual	
Secuencia normal	Paso	Acción
	1	El usuario hace clic sobre la pregunta que quiere cambiar de posición.
	2	Sin dejar de hacer clic situa la pregunta en su nueva posición.
	3	El usuario suelta la pregunta dejando de hacer clic.
Postcondición	Se actualizan las posiciones de las preguntas del formulario actual.	

Cuadro 3.20: Descripción 4.4.4 - Cambiar posición de una pregunta

3.2. Modelo conceptual

Tras haber realizado la especificación de los requisitos y haber determinado cuales son las funciones necesarias en nuestro sistema para cumplir tales requisitos, pasamos a ver un diagrama del modelo conceptual, el cuál nos muestra los conceptos presentes en el dominio del problema. Posteriormente se explicará de forma detallada las relaciones entre estos conceptos.

Figura 3.3: Modelo conceptual del sistema.



En este diagrama tenemos múltiples objetos. Cada uno de ellos tiene una función específica dentro del sistema, así que pasemos a ver cuál es:

1. **Usuario:** este objeto del diagrama respresentará a cada uno de los usuarios del sistema, y nos permitirá gestionar sus atributos y relaciones con el resto de objeto. Además de por supuesto permitir la identificación y autorización en la web.
2. **Espacio de trabajo:** los espacios de trabajos son entidades que están asociadas a un único usuario y que permiten gestionar un conjunto de formularios.
3. **Formulario:** los formularios serán el centro de nuestra aplicación y almacenarán tanto una serie de preguntas como las respuestas al mismo.
4. **Pregunta:** una pregunta es un objeto que forma parte de un formulario concreto, y que almacena la información común a todos los tipos de preguntas presentes en la aplicación.
5. **Respuesta:** una respuesta mantendrá los datos acerca de un envío de un encuestado concreto para un formulario específico.
6. **Respuesta a una pregunta:** este objeto mantendrá la respuesta a una pregunta de un formulario para un envío concreto por parte del encuestado.
7. **Texto:** este objeto aunque es tomado como un tipo de pregunta, no lo es verdaderamente, sin embargo debido a que comparte ciertos atributos con las preguntas, será tomado como tal para facilitar el desarrollo.
8. **Legal:** será un tipo de pregunta que permite mostrar un texto legal por pantalla y que solo acepta dos opciones: aceptar o no aceptar el texto legal.
9. **Email:** este tipo representará a una pregunta que solo permite como entrada un texto de tipo email. Su objetivo de ser es facilitar la validación de la entrada de la pregunta para determinar si el texto introducido es un email.
10. **Respuesta larga:** este tipo representa a una pregunta a la que únicamente se puede responder con texto largo, es decir con una etiqueta HTML *textarea*. A la hora de generar la pregunta en el formulario, se generará dicha etiqueta HTML tras determinar que la pregunta es de este tipo.
11. **Respuesta corta:** este tipo representa a una pregunta a la que únicamente se puede responder con breve texto, es decir con una etiqueta

HTML *input* del tipo *text*. A la hora de generar la pregunta en el formulario, se generará dicha etiqueta HTML tras determinar que la pregunta es de este tipo.

12. **Fecha:** al igual que los dos tipos anteriores, las preguntas de este tipo generan una entrada en la que se deberá introducir una fecha.
13. **Escala de opinión:** las preguntas de este tipo mostrarán por pantalla un rango de valores representando una escala en la que solo se podrá seleccionar un único valor.
14. **Hora:** las preguntas de este tipo generan una entrada en las que únicamente se podrá introducir una hora.
15. **Página web:** las preguntas de este tipo generan una entrada en las que únicamente se podrá introducir una dirección URL.
16. **Número:** las preguntas de este tipo generan una entrada en las que únicamente se podrá introducir un número entre un rango de valores de forma opcional.
17. **Desplegable:** las preguntas de este tipo generan una entrada que estará formada por una etiqueta HTML del tipo *select*, por lo que únicamente se podrá seleccionar una opción.
18. **Varias respuestas:** este tipo de pregunta permite elegir entre varias respuestas en una misma pregunta, y dependiendo del caso, se podrán seleccionar varias opciones o no.
19. **Opción:** este objeto, representa a una opción de los dos tipos de pregunta anteriores: *Desplegable* y *Varias respuestas*. Sirve para almacenar el valor de la opción y la asociación con la pregunta a la que pertenece.
20. **Si/No:** este tipo es verdaderamente un subtipo de *Varias respuestas* pero que genera automáticamente en la interfaz solo dos opciones: si o no.
21. **Cuadrícula:** se trata del tipo más complejo de la aplicación ya que representa una matriz en la que por cada dupla fila-columna, se permite seleccionar una opción.
22. **Fila:** representa a una fila del tipo de pregunta *Cuadrícula*, y almacena el valor que se le ha dado y la relación entre la fila la pregunta a la que pertenece.
23. **Columna:** representa a una columna del tipo de pregunta *Cuadrícula*, y almacena el valor que se le ha dado y la relación entre la fila la

pregunta a la que pertenece. Además establece cuál será el valor de la respuesta para esa dupla fila-columna.

24. Image

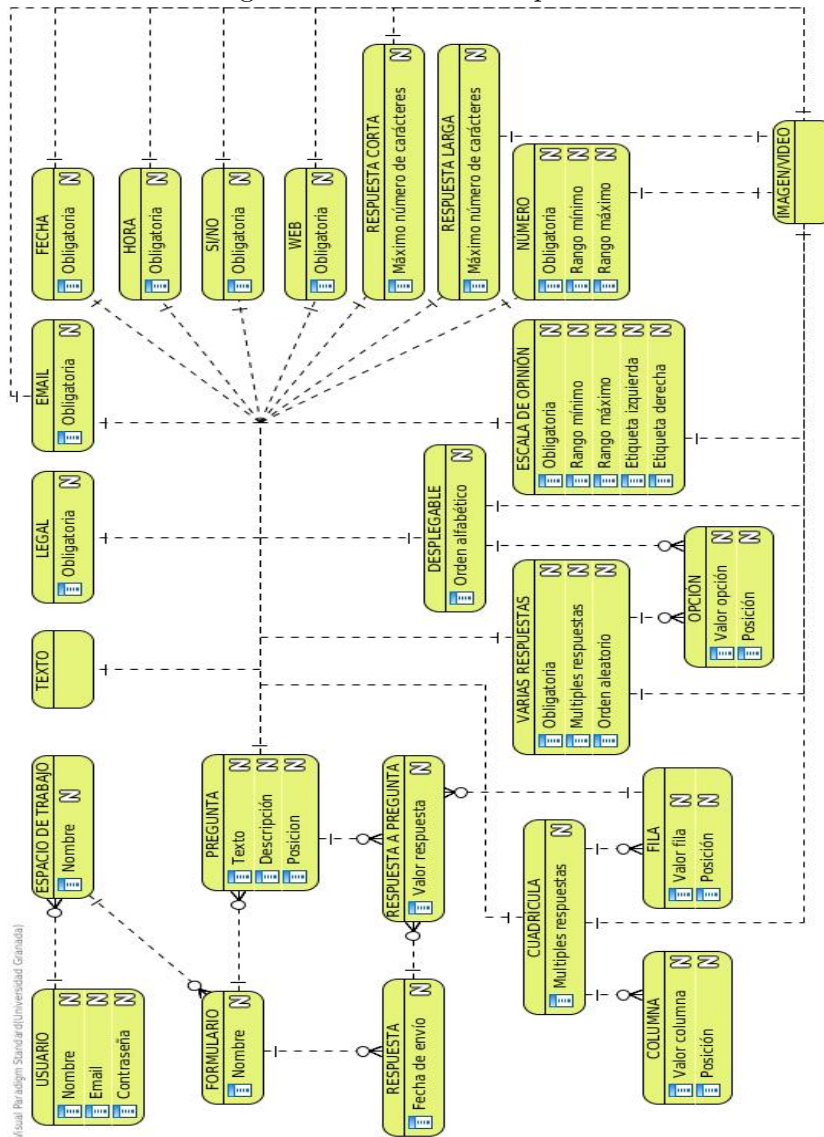
Vídeo: representa al vídeo o imagen que pueden tener asociado ciertos tipos de pregunta.

Tras describir los objetos voy a pasar a comentar cada una de las relaciones del diagrama:

- Un **usuario** puede tener de uno a muchos **espacios de trabajo**. Además como mínimo debe tener uno por defecto.
- Un **espacio de trabajo** puede contener de cero a muchos **formularios**.
- Un **formulario** puede estar formado de cero a muchas **preguntas**. Estas preguntas, tal y como se ve en el diagrama disponen de una posición en la lista de preguntas que componen el formulario, y dicha posición puede ser modificada por el usuario. Esta posición determinará el orden en el que le aparecen las preguntas al encuestado.
- Una **pregunta** está asociada con un tipo específico de pregunta. Esta entidad que representa al tipo de pregunta mantiene la configuración o valores de los atributos que definen a la pregunta dentro del formulario.
- Los tipos de pregunta **Desplegable** y **Varias respuestas** pueden tener de cero a muchas **opciones**. Estas opciones están definidas por el valor de la respuesta que representan.
- El tipo de pregunta **Cuadrícula** puede tener de cero a muchas **filas** y de cero a muchas **columnas** que representarán las respuestas de la cuadrícula.
- Como se puede ver en el diagrama conceptual, ciertos tipos de pregunta pueden tener asociado bien un vídeo o una imagen.
- Un **formulario** puede haber recibido de cero a muchas **Respuestas**.
- Una **respuesta** a un formulario está compuesta de las repuestas a cada una de las preguntas del formulario.
- Una **pregunta** puede tener muchas respuestas de distintas **respuestas al formulario**.
- Una **respuesta a una pregunta** puede estar asociada a un fila del tipo de pregunta **Cuadrícula**.

Y ya para acabar el apartado, se muestra con más detalle los tributos necesarios para cada objeto según la especificación de requisitos:

Figura 3.4: Modelo conceptual del sistema.



Capítulo 4

Diseño

Tras deliberar acerca de qué tecnologías debía usar para llevar a cabo la implementación del proyecto, me decidí a utilizar para el back-end el lenguaje de programación PHP junto con un framework para el mismo lenguaje denominado Laravel y que se basa en el conocido framework Symfony. Además para la gestión de la base de datos he utilizado MySQL/MariaDB debido a que se integra de forma sencilla con el framework a utilizar.

Por otro lado, para el front-end uso HTML, CSS y Javascript. Más concretamente, he hecho uso de dos frameworks para el CSS: Bootstrap en su versión 3.3.7 para el panel de gestión del usuario y MaterializeCSS para la visualización de los formularios por los encuestados. Cabe recalcar que aunque el panel de gestión lo haya diseñado con Bootstrap, este no está pensado para ser responsive, pues bajo mi punto de vista resulta mucho más cómodo y lógico diseñar gestionar los formularios desde en ordenador de escritorio. No obstante la visualización de los formularios si es responsive. Por último, para facilitarme el trabajo con Javascript, he utilizado la librería JQuery, puesto que además es la que usan por defecto tanto Bootstrap como MaterializeCSS. Todo ello se ejecuta sobre el servidor Ngnix.

El porqué he decidido utilizar estas tecnologías se debe a qué ya había trabajado con ellas durante mi formación en el grado y ello me iba a facilitar el desarrollo del proyecto.

Dicho esto, comenzaré a comentar los aspectos de diseño de la aplicación desarrollada.

4.1. Algunos conceptos antes de comenzar

Antes de comenzar a explicar los detalles de diseño de la aplicación es importante comentar algunos conceptos relacionados con el framework Laravel que he utilizado para desarrollar la misma.

1. Laravel sigue el patrón MVC. Los objetos de la clase *Model* (de la que se hablará más adelante) representan al modelo en este patrón. Por

otro lado, tenemos la clase *Controller*. Todas las clases que hereden de esta, podrán ser usadas como controladores en la aplicación, y estas clases podrán usarse para gestionar las peticiones al sistema. Ya por último tenemos las vistas, que en la Laravel esta representada por la fachada *View*, que nos permite retornar las vistas junto con los datos que le pasemos.

2. Para establecer las URLs que están disponibles en nuestra aplicación, disponemos de un fichero denominado **web.php**. En este fichero establecemos las URLs junto con el método HTTP al que responden. Además, podemos especificar la acción a realizar tras la solicitud, bien mediante una función *callback* o con una función de una clase: es importante recalcar que esta clase debe heredar de la clase **Controller**.
3. Laravel dispone de una clase para obtener los datos recibidos de una solicitud HTTP. Esta clase se denomina **Request** y engloba todos los datos relacionados con la solicitud (inputs, método HTTP utilizado, etc.). Los objetos de esta clase se pasan por defecto a todas las funciones especificadas en el fichero comentado anteriormente y que tengan como parámetro un objeto de dicha clase.
4. Para comunicarnos con nuestra base de datos Laravel incluye un ORM denominado Eloquent que implementa el patrón arquitectónico *Active-Record*. Para implementar este patrón Eloquent hace uso de una clase llamada *Model*, de la cual heredarán los objetos que interacciones con la base de datos. Los modelos sirven para establecer las relaciones con otros modelos, determinando así como será nuestra base de datos (número de tablas y relaciones entre ellas mediante claves externas).
5. Para los objetos que hereden de la clase *Model* disponemos de una serie de eventos que se disparan cuando hay una comunicación con la base de datos (creado, creandose, eliminandose, eliminado...). Para poder usar estos eventos, debemos hacer uso de un *Listener*. Estos objetos están asociados a un evento concreto, y reaccionan cuando se disparan. En ciertas partes de la aplicación he hecho uso de estos eventos para poder llevar a cabo la lógica del mismo.
6. Eloquent posee un mecanismo denominado *Polymorphic relations*. Este mecanismo permite que un mismo modelo pueda pertenecer a distintos modelos. Imaginemos el caso en el que tenemos dos tipos de pregunta que pueden tener muchas opciones. En un esquema de base de datos normal tendríamos una tabla para cada tipo de pregunta y dos tablas para las opciones: una para el primer tipo y otra para el segundo, en las que cada una tendría un clave externa hacia la tabla de su tipo. Con este mecanismo podemos tener una sola tabla para ambos tipos de pregunta, ya que en lugar de tener una clave externa,

tenemos dos columnas que indican la tabla a la que pertenece la tupla y la id de la tupla en la otra tabla. Podemos crear así relaciones de herencia de forma mucho más sencilla.

7. Disponemos de una clase fachada denominada *Auth* que proporciona una interfaz para obtener los datos del actual usuario identificado a través de su modelo. Esta fachada puede ser usada como un *middleware* de forma automática por los controladores que deseemos, consiguiendo de esta forma que en estos controladores sea necesario que el usuario esté identificado para llevar a cabo alguna acción.

4.2. URLs de la aplicación

Antes de comenzar a desarrollar la aplicación, hay que establecer las URLs a las que responderá nuestra web de acuerdo con los requisitos establecidos. En la lista que acompaña esta sección, iré explicando una a una la direcciones disponibles y a qué métodos HTTP responden.

1. **URL:** /
Método: GET
Descripción: devuelve la vista principal de la web.
2. **URL:** /workspaces
Método: POST
Descripción: crea un nuevo espacio de trabajo con los datos del formulario y devuelve la vista del nuevo espacio de trabajo que se ha creado.
3. **URL:** /workspaces/:id
Método: GET
Parámetros URL:
 - **Requeridos:** id=[integer]**Descripción:** devuelve la vista para gestionar el espacio de trabajo cuyo ID coincide con la que se ha pasado como parámetro.
4. **URL:** /workspaces/:id
Método: PUT
Parámetros URL:
 - **Requeridos:** id=[integer]**Descripción:** actualiza los atributos del espacio de trabajo.
5. **URL:** /workspaces/:id
Método: DELETE
Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: elimina de la base de datos el espacio de trabajo cuyo ID coincide con la que se ha pasado como parámetro y redirige al usuario al espacio de trabajo por defecto.

6. **URL:** /forms

Método: POST

Descripción: crea un nuevo formulario y redirige al usuario a la página de diseño del formulario.

7. **URL:** /forms/id/build

Método: GET

Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: devuelve la vista de la página de diseño del formulario cuyo ID coincide con la que se ha pasado como parámetro.

8. **URL:** /forms/id/share

Método: GET

Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: devuelve la vista de la página de para compartir el formulario cuyo ID coincide con la que se ha pasado como parámetro.

9. **URL:** /forms/id/analyze

Método: GET

Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: devuelve la vista de la página de análisis del formulario cuyo ID coincide con la que se ha pasado como parámetro.

10. **URL:** /forms/id/analyze/global_stats

Método: GET

Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: esta URL devuelve texto HTML con gráficas de las estadísticas de todas las respuestas del formulario cuyo ID coincide con la que se ha pasado como argumento.

11. **URL:** /forms/id/analyze/single_stats

Método: GET

Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: esta URL devuelve texto HTML con la página donde se visualiza de forma individual las respuestas del formulario cuyo ID coincide con la que se ha pasado como argumento.

12. **URL:** /forms/id/analyze/response_stats

Método: GET

Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: devuelve de forma individual una respuesta del formulario cuyo ID coincide con la que se ha pasado como argumento.

13. **URL:** /forms/id

Método: PUT

Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: actualiza los atributos de un formulario.

14. **URL:** /forms/id

Método: DELETE

Parámetros URL:

- **Requeridos:** id=[integer]

Descripción: elimina el formulario de la base de datos que coincide con la ID pasada como argumento.

15. **URL:** /questions

Método: POST

Descripción: crea una nueva pregunta para un formulario determinado.

16. **URL:** /questions/create/type

Método: GET

Parámetros URL:

- **Requeridos:** type=[string]

Descripción: devuelve código HTML con la vista para crear un pregunta del tipo pasado como argumento.

17. **URL:** /questions/id/edit

Método: GET

Parámetros URL:

- **Requeridos:** id=[string]

Descripción: devuelve código HTML con la vista para crear un pregunta con la ID pasada como argumento.

18. **URL:** /questions/id

Método: PUT

Parámetros URL:

- **Requeridos:** id=[string]

Descripción: actualiza los atributos de la pregunta cuyo ID coincide con la que se ha pasado como argumento.

19. **URL:** /questions/update/order

Método: PUT **Descripción:** actualiza el orden de las preguntas para un determinado formulario.

20. **URL:** /questions/id

Método: DELETE

Parámetros URL:

- **Requeridos:** id=[string]

Descripción: elimina la pregunta cuyo ID coincide con la pasada como argumento.

21. **URL:** /view/form/id

Método: GET

Parámetros URL:

- **Requeridos:** id=[string]

Descripción: devuelve la vista pública para rellenar el formulario que coincide con la ID pasada como argumento.

22. **URL:** /submit/form

Método: POST

Parámetros URL:

- **Requeridos:** id=[string]

Descripción: envía las respuestas de un formulario para que sean almacenadas en la base de datos.

23. **URL:** /form/submitted

Método: GET

Parámetros URL:

- **Requeridos:** id=[string]

Descripción: devuelve una vista cuando se ha enviado un formulario.

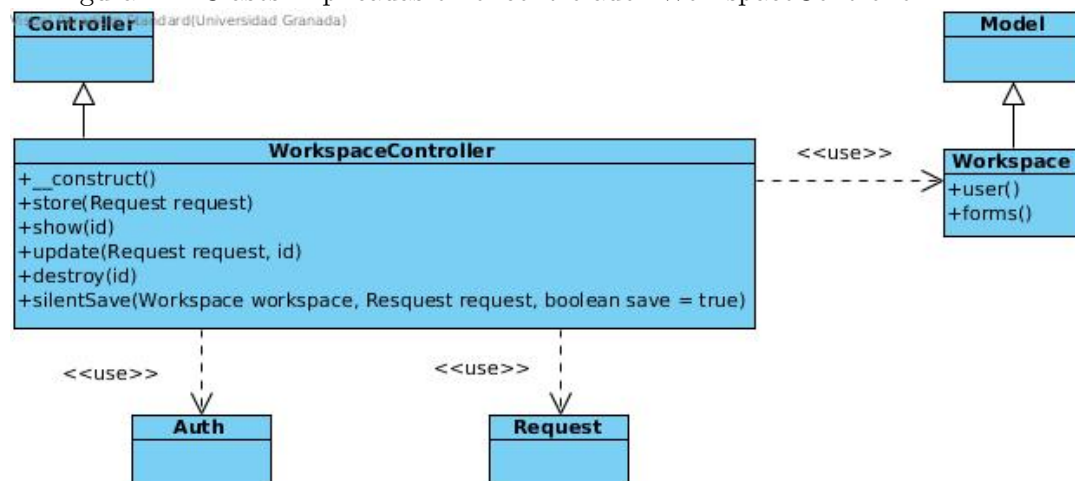
4.3. Diagramas de clase

A continuación se muestran los diagramas de clase de la aplicación. En ellos se muestran las clases que intervienen y las relaciones de estas para llevar a cabo la lógica de la misma. Para facilitar la comprensión y debido al tamaño de los diagramas, he dividido estos según el controlador que dirige el flujo de datos. Por tanto, en cada diagrama aparecerán las clases con las que interacciona el controlador y como estas se comunican con otras. Además explicaré algunas de las funciones más relevantes de cada clase.

WorkspaceController

El siguiente diagrama de clases muestra las clases implicadas en la lógica del controlador **WorkspaceController**:

Figura 4.1: Clases implicadas en el controlador WorkspaceController



La clase **WorkspaceController** se encarga de recibir y gestionar todas las operaciones relacionadas con los espacios de trabajo del usuario. Las funciones que la componen son:

1. **store:** esta función se encarga de crear un nuevo espacio de trabajo con la información recibida en la solicitud HTTP. Para ello y tal y

como se ha comentado antes, se hace uso de un objeto de la clase `Request`.

2. **show:** esta función recibe como parámetro de la URL el ID de un espacio de trabajo concreto, con el cual obtendremos el espacio de trabajo que el usuario desea visualizar. Antes de redirigir al usuario a la vista de dicho espacio de trabajo, se comprueba que el usuario es el dueño del espacio de trabajo. En caso de que no fuera así, se envía como respuesta un error 403.
3. **update:** se actualizan los atributos del espacio de trabajo cuyo ID se ha pasado como argumento.
4. **destroy:** se elimina el espacio de trabajo cuyo ID se ha pasado como argumento y se redirige al usuario al espacio de trabajo asignado por defecto.
5. **silentSave:** se trata de una función básica auxiliar para asignar a un objeto del modelo *Workspace* los valores del parámetro *request*, tras lo cual se guarda el modelo en la base de datos.

Por otro lado, tenemos la clase *Workspace*. Esta clase es una clase de Eloquent, ya que extiende de la clase *Model*, lo cual significa que establece las relaciones con otros modelos y nos sirve para interactuar con la base de datos. Tenemos las siguientes funciones:

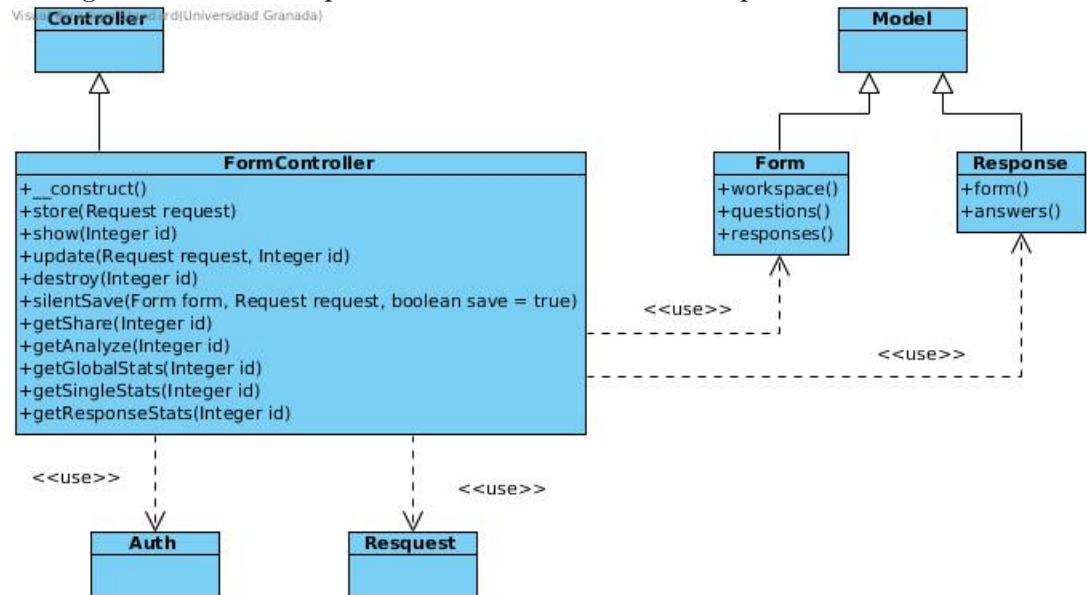
1. **user:** devuelve el modelo del usuario al que pertenece el espacio de trabajo que ha llamado a la función.
2. **forms:** devuelve todos los modelos de los formularios asociados al espacio de trabajo.

Las funciones de la clase *Workspace*, siguen el estandar del ORM Eloquent para poder obtener las relaciones entre las distintas tablas de nuestra base de datos.

FormController

El siguiente diagrama de clases muestra las clases implicadas en la lógica del controlador **FormController**:

Figura 4.2: Clases implicadas en el controlador WorkspaceController



La clase **FormController** se encarga de gestionar las peticiones relacionadas con los formularios y sus funciones son:

1. **store:** esta función se encarga de crear un nuevo formulario con la información recibida en la solicitud HTTP. Para ello se hace uso de un objeto de la clase **Request** que contiene toda la información de la solicitud.
2. **show:** esta función recibe como parámetro de la URL un ID con el cual obtendremos el formulario que el usuario desea visualizar. Antes de redirigir al usuario a la vista de dicho formulario, se comprueba que el usuario es el dueño del espacio de trabajo con el cual está asociado el formulario. En caso de que no fuera así, se envía como respuesta un error 403.
3. **update:** se actualizan los atributos del formulario cuyo ID se ha pasado como argumento.
4. **destroy:** se elimina el formulario cuyo ID se ha pasado como argumento junto con toda la información con la que estaba relacionada.

5. **silentSave:** se trata de una función básica auxiliar para asignar a un objeto del modelo *Form* los valores del parámetro *request*, tras lo cual se guarda el modelo en la base de datos.
6. **getShare:** se comprueba si el usuario tiene acceso al formulario y se le redirige a la vista de compartir formulario.
7. **getAnalyze:** se comprueba si el usuario tiene acceso al formulario y se le redirige a la vista de analizar formulario.
8. **getGlobalStats:** se realizan calculos sobre las respuestas obtenidas hasta el momento del formulario que coincide con el ID pasado como argumento. Una vez hecho los calculos, se devuelve una vista con dichas estadísticas.
9. **getSingleStats:** se devuelve una vista en la que se podrá ver de forma individual las respuesta del formulario que coincide con el ID pasado como argumento.
10. **getResponseStats:** devuelve una respuesta concreta a un formulario. Está función es usada por la URL a la que responde la función anterior, ya que el envío por parte de esta función se realiza ante una solicitud hecha por AJAX. Básicamente se obtiene la siguiente respuesta a mostrar, y se pasa su contenido a la vista a enviar como respuesta.

Respecto a la clase *Form*, tenemos como funciones:

1. **workspace:** devuelve el modelo del espacio de trabajo al que pertenece el formulario.
2. **questions:** devuelve todos los modelos de las preguntas que contiene el formulario.
3. **responses:** devuelve todos los modelos de las respuestas recibidas para el formulario.

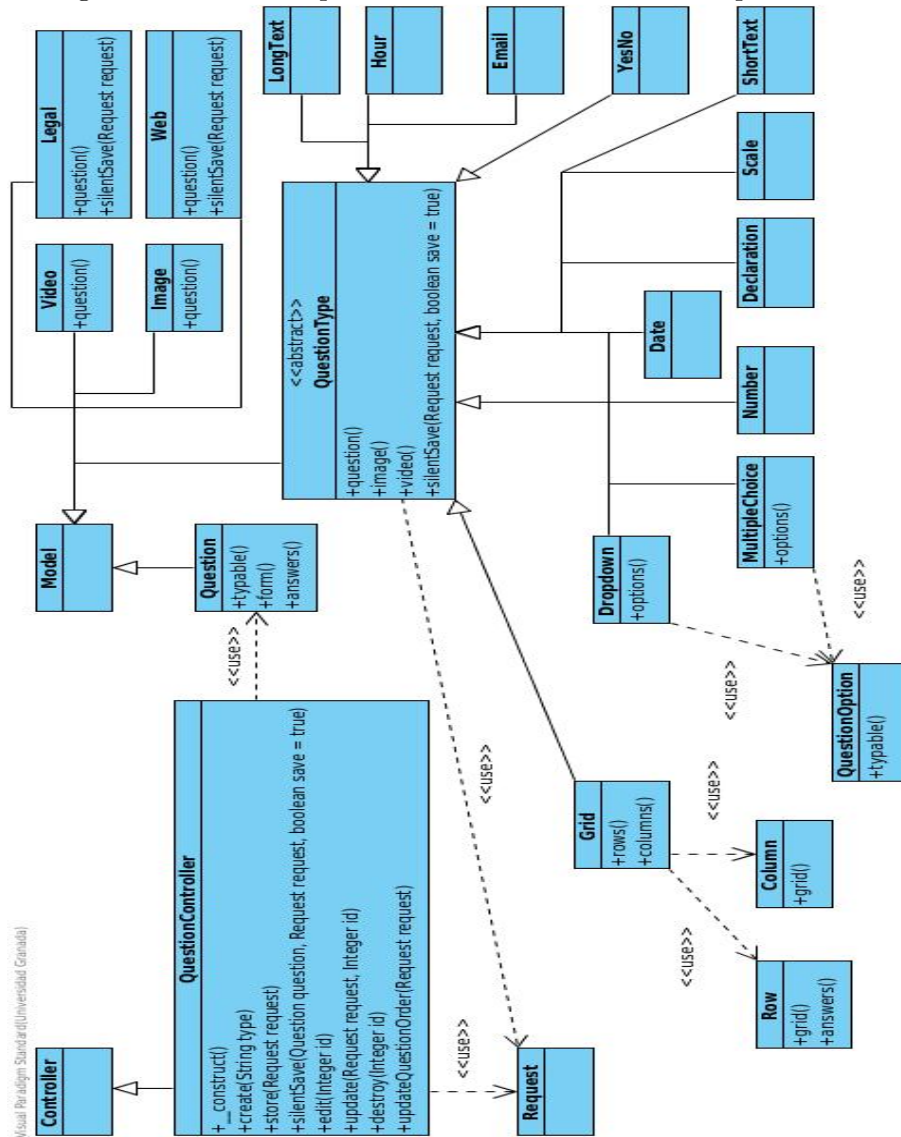
Para finalizar este subapartado, tenemos la clase *Response*:

1. **form:** devuelve el formulario al que pertenece la respuesta.
2. **answers:** devuelve las respuestas a las preguntas del formulario al que pertenece.

QuestionController

El siguiente diagrama de clases muestra las clases implicadas en la lógica del controlador **QuestionController**:

Figura 4.3: Clases implicadas en el controlador WorkspaceController



En la clase `QuestionController` encontramos las funciones encargadas de gestionar las preguntas de un formulario:

1. **create:** devuelve la vista para crear una pregunta según el tipo de pregunta que se le haya pasado como parámetro en la URL.
2. **store:** esta función se encarga de crear una nueva pregunta con la información recibida en la solicitud HTTP. Para ello se hace uso de un objeto de la clase `Request`.
3. **silentSave:** función auxiliar para asignar a un objeto del modelo del tipo de pregunta específica los valores del parámetro *request*, tras lo cual se guarda el modelo en la base de datos.
4. **edit:** devuelve la vista para editar una pregunta según el ID de esta.
5. **update:** se actualizan los atributos de la pregunta cuyo ID se ha pasado como argumento.
6. **destroy:** se elimina la pregunta con el ID que se ha pasado como argumento.
7. **updateQuestionOrder:** para cada pregunta del formulario, se actualiza la posición dentro de este. Para ello, se ha recibido un array en el que la clave es el id de la pregunta, y el valor su nueva posición. Para cada pregunta del array se le asigna una nueva posición y se guarda en la base de datos.

A continuación, voy a pasar a describir las distintas clases con las que el controlador tiene relación. Todas heredan de la clase *Model* de Eloquent. Para comenzar, describiré la clase *Question*. Esta clase representa a una pregunta de un formulario concreto. Carece de tipo y podría decirse que actúa como superclase de los distintos tipos de pregunta. Sin embargo esto no es así, ya que su función es establecer la relación entre el formulario y las preguntas que este posee utilizando únicamente dos tablas: la tabla que contiene las tuplas de formularios, y la tabla que contiene las tuplas de preguntas. Cada tupla de una pregunta, estará relacionada posteriormente con un tipo concreto de pregunta siguiendo el mecanismo de *Polymorphic relations* explicado anteriormente. Tras esto, pasemos a ver cada una de las funciones de esta clase:

1. **typable:** esta función sirve para devolver el modelo que representa la tupla que contiene los datos del tipo concreto de pregunta. Estos modelos que representan los tipos de pregunta los explicaré a continuación.
2. **form:** devuelve el modelo del formulario al que pertenece la pregunta.

3. **answers:** devuelve todos los modelos de las respuestas que ha recibido esta pregunta.

Por otro lado tenemos la clase *QuestionType*. Esta clase actúa como superclase para los tipos específicos de pregunta y alberga las funciones comunes a todos estos tipos. Se trata de una clase abstracta, ya que define la función *silentSave*, la cual recibe como parámetro un objeto de la clase *Request* para poder establecer los atributos específicos a un tipo de pregunta. Las funciones que define son:

1. **question:** devuelve el modelo de la pregunta a la que pertenece el tipo de pregunta.
2. **image:** devuelve el modelo de la imagen que tiene asociada el tipo de pregunta.
3. **video:** devuelve el modelo del video que tiene asociado el tipo de pregunta.
4. **silentSave:** tal y como se ha comentado, esta función es definida por cada uno de los tipos de pregunta para establecer la lógica de almacenamiento de la pregunta.

Las clases *Video*, *Image*, *Web* y *Legal* definen una única función:

1. **question:** que devuelve el modelo de la pregunta a la que pertenece.

Las clases *Web* y *Legal* no heredan de la clase *QuestionType* debido a que según la especificación de requerimientos, estas clases no pueden tener una imagen o un video asociados.

Por otro lado, las clases *Dropdown* y *MultipleChoice*, definen una sola función:

1. **options:** devuelve todos los modelos de las opciones asociadas al tipo de pregunta

La clase *Option* por su parte define la función:

1. **typable:** devuelve el modelo del tipo de pregunta a la que pertenece esta opción. Este modelo puede ser de dos tipos únicamente: *Dropdown* y *MultipleChoice*.

Ya para finalizar, tenemos la clase *Grid*. Esta clase representa al tipo de pregunta *Cuadrícula*, lo que supone que debe estar relacionado con las filas y columnas que posee. Para poder obtener los modelos de dichas filas y columnas, hace uso de las funciones *rows* y *columns*. Por su parte, la clase *Row* define las siguientes funciones:

1. **grid:** devuelve el modelo del tipo de pregunta Cuadrícula al que está asociado.
2. **answers:** devuelve todas las respuestas que ha recibido la fila. Estas respuestas tienen como valor el que estableciera la columna correspondiente

Por último la clase *Column*, define al igual que la clase *Row* una función para obtener la pregunta a la que pertenece.

Tras haber explicado las relaciones existente en este diagrama, voy a mostrar otro diagrama con las misma clases, pero indicando la relación de los modelos que representan los tipos específicos de pregunta con los eventos que disparan, ya que son una parte muy importante del funcionamiento de la aplicación.

Iré explicando uno por uno los eventos que aparecen en el diagrama, qué modelos los disparan y cuales son los *listeners* que tiene asociados. A todos estos *listeners* se les pasa como argumento un objeto que del evento

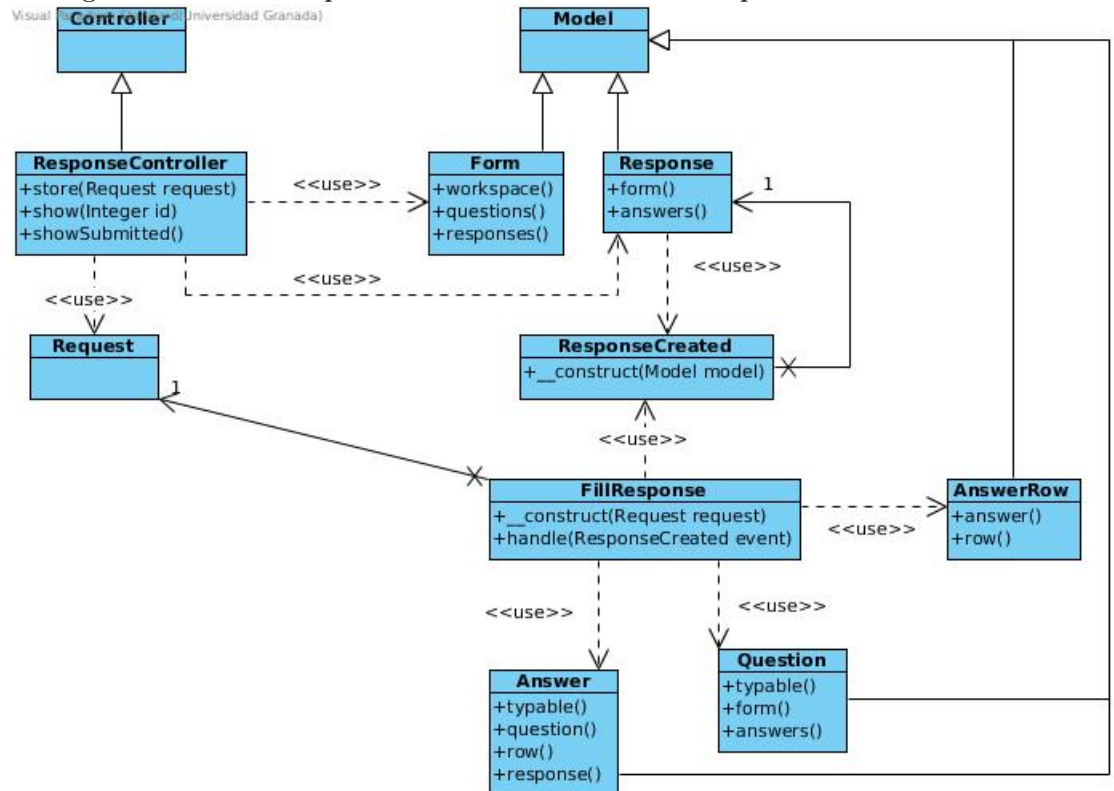
correspondiente.

1. **DeleteQuestion:** este evento se dispara cuando un modelo va a ser eliminado. Es disparado por todos los modelos que aparecen en el diagrama de clases, ya que todos estos pueden tener o bien una imagen o un video asociado. El *listener* que escucha a este evento es **DeleteFile**. Este se encarga de eliminar el objeto de video o imagen que tenga asociado el modelo.
2. **SaveQuestion:** este evento se dispara cuando un modelo va a ser guardado en la base de datos, ya sea mediante una actualización o la creación del mismo. Es disparado por todos los modelos que aparecen en el diagrama de clases, ya que todos estos pueden tener o bien una imagen o un video asociado. El *listener* que escucha a este evento es **SaveFile**. Este se encarga de guardar o actualizar el objeto de video o imagen que tenga asociado el modelo.
3. **DeleteOptionsQuestion:** este evento se dispara cuando un modelo de las clases MultipleChoice o Dropdown va a ser eliminado. Los *listeners* que escuchan a este evento son **DeleteFile** y **DeleteOptions**. Este último se encarga de eliminar los modelos de la clase **QuestionOption** que tenga asociado el modelo.
4. **DeleteGridQuestion:** este evento se dispara cuando un modelo de la clase Grid va a ser eliminado. Los *listeners* que escuchan a este evento son **DeleteFile**, **DeleteRows** y **DeleteColumns**. Estos últimos se encargan de eliminar los modelos de las clases **Row** y **Column** respectivamente que tenga asociado el modelo.

ResponseController

Para terminar este apartado, tenemos el diagrama de clases muestra las clases implicadas en la lógica del controlador **ResponseController**:

Figura 4.5: Clases implicadas en el controlador WorkspaceController



En este diagrama tenemos la clase **ResponseController**. Esta clase se encarga de gestionar todo lo relacionado con la generación y almacenado de los formularios que son públicos para los encuestados. Es por ello que no hace uso de la fachada *Auth* anteriormente descrita, ya que no es necesario estar identificado en el sistema para relalizar un formulario. Las funciones de esta clase son:

1. **show**: muestra por pantalla el formulario cuyo ID coincide con el que se ha pasado como parámetro en URL, para que pueda ser respondido por el encuestado.
2. **store**: se encarga de almacenar en la base de datos una respuesta a un determinado formulario. Y redirige al usuario a una nueva pantalla.
3. **showSubmitted**: muestra una pantalla indicando que la respuesta al formulario se ha almacenado con éxito.

Por otro lado tenemos la clase *Form* (que ya ha sido explicadas en apartados anteriores) y la clase *Response*, que tiene los métodos:

1. **form:** devuelve el modelo del formulario asociado a esta respuesta.
2. **answers:** devuelve los modelos de todas las respuestas de las preguntas del formulario asociado a esta respuesta.

Para finalizar voy explicar el evento que dispara la clase *Response* al ser creada. Este evento es ***ResponseCreated***. Cuando una respuesta ha sido creada, este evento obtiene como parámetro el modelo de esta respuesta. Este modelo será usado por la clase ***FillResponse*** que es el *listener* del evento. Básicamente, esta última clase crea y relaciona una respuesta con la pregunta a la que responde. En el caso particular del tipo de pregunta *Cuadrícula*, es necesario establecer una relación entre una respuesta y la fila a la que responde, ya que para cada fila de la cuadrícula puede haber una o más respuestas. Para conseguir esto, he utilizado una tabla pivote entre las respuestas y las filas representada por el modelo *AnswerRow* que almacena el id de ambas tuplas para establecer la relación entre ellas.

Capítulo 5

Implementación

En este apartado voy a describir con pseudocódigo los algoritmos de mayor complejidad de la aplicación.

5.1. Algoritmo para crear/actualizar el tipo Cudrícula

Este algoritmo se encuentra en el método *silentSave* de la clase *Grid*. Su función es crear y almacenar en la base de datos una pregunta del tipo *Cuadrícula*, junto con sus atributos y modelos asociados. Estos modelos son de las clases *Row* y *Column*. Basicamente su objetivo es actualizar, eliminar o crear las filas y columnas asociadas que tenga el modelo. A continuación de muestra el pseudocódigo:

```
1 //Creacion/actualizacion de filas
2 Si hay alguna fila para crear o actualizar
3     Obtengo las filas asociadas actuales //Puede que no haya ninguna
4
5     //Compruebo si alguna fila actualmente asociada se ha eliminado
6     Para cada fila en las filas actuales
7         Si en el array de entrada no se encuentra la fila actual
8             Elimino la fila y por tanto la asociacion
9
10    var n = numero de filas de la entrada
11    Desde i = 0 hasta n
12        Si la fila con posicion i no existe entre las asociadas
13            Creo un nuevo objeto fila
14
15            Establezco los datos de la fila a partir de la entrada
16            Guardo y asocio la fila con la pregunta
17
18 En caso contrario
19     Elimino todas las filas que pueda tener asociada el modelo
20
21 //Creacion/actualizacion de columnas
22 Si hay alguna columna para crear o actualizar
23     Obtengo las columnas asociadas actuales //Puede que no haya
        ninguna
```



```
24
25 //Compruebo si alguna columna actualmente asociada se ha eliminado
26 Para cada columna en las columnas actuales
27     Si en el array de entrada no se encuentra la columna actual
28         Elimino la columna y por tanto la asociacion
29
30     var n = numero de columna de la entrada
31     Desde i = 0 hasta n
32     Si la columna con posicion i no existe entre las asociadas
33         Creo un nuevo objeto columna
34
35     Establezco los datos de la columna a partir de la entrada
36     Guardo y asocio la columna con la pregunta
37
38 En caso contrario
39     Elimino todas las columnas que pueda tener asociada el modelo
```

5.2. Algoritmo para almacenar una respuesta

Este algoritmo se encuentra en el método *handle* de la clase *listeter Fill-Response*. Su objetivo es: una vez se ha recibido una respuesta a un formulario por parte de un usuario encuestado, se almacenan todas las respuestas a cada una de las preguntas que ha respondido el encuestado. Veamos cuál es el algoritmo que sigue:

```
1 //La funcion toma como entrada el evento al que escucha, del cual
  obtiene el modelo
2 //Toma tambien a traves de un atributo de clase un objeto Request
3 Si se ha recibido al menos una respuesta a una pregunta
4     Para cada respuesta a una pregunta
5         Si el tipo de pregunta es Cuadrícula
6             Para cada fila que ha sido respondida
7                 Si son varias respuestas en una misma fila
8                     Para cada respuesta de la fila
9                         Creo la respuesta
10                        Asocio la respuesta con la fila
11                        Guardo la respuesta en la base de datos
12                 Si solo hay una respuesta en la fila
13                     Creo la respuesta
14                     Asocio la respuesta con la fila
15                     Guardo la respuesta en la base de datos
16         Si es cualquier otro tipo de pregunta
17             Si hay varias respuestas a una misma pregunta
18                 Para cada respuesta a la pregunta
19                     Creo la respuesta
20                     Guardo la respuesta en la base de datos
21             Si solo hay una respuesta a la pregunta
22                 Creo la respuesta
23                 Guardo la respuesta en la base de datos
```

5.3. Algoritmo para obtener las estadísticas globales de un formulario

Este algoritmo se encuentra en el método *getGlobalStats* de la clase *FormController* y su función es devolver una vista al usuario con las estadísticas para ciertos tipos de preguntas. Para ello antes debe generar un objeto en el que mapear para cada pregunta los resultados que ha obtenido. En lugar de entrar en detalles sobre como el algoritmo genera el analisis de los resultados, voy a mostrar como se mapean para cada tipo de pregunta estos resultados. Comenzaremos por los tipos de pregunta *MultipleChoice* y *Dropdown*. Cuando obtenemos los resultados para uno de estos tipos de pregunta el item del array que obtenemos tiene este formato:

```

1 id_pregunta => array (
2   'labels' => array(
3     0 => 'Opcion 1',
4     1 => 'Opcion 2',
5   ),
6   'count_options' => array (
7     0 => 2,
8     1 => 1,
9   ),
10 )

```

En el formato mostrado, la clave *labels* encapsula cada una de las opciones disponibles actualmente para la pregunta. En cuanto a la clave *count_options*, esta encapsula para cada opción las veces que ha sido escogida cada opción en el mismo orden en la que aparecen en la clave anterior. De esta forma podemos representar mediante una gráfica cuales han sido las opciones escogidas.

Por otro lado, tenemos el tipo de pregunta *Grid*, para el cuál se obtiene lo resultados en el siguiente formato:

```

1 id_pregunta => array (
2   'labels' => array(
3     0 => 'Fila 1',
4     1 => 'Fila 2',
5   ),
6   'columns' => array (
7     'Columna 1' => array (
8       0 => 1,
9       1 => 3,
10    ),
11    'Columna 2' => array (
12      0 => 2,
13      1 => 4,
14    ),
15  ),
16 )

```

En este caso, al igual que en el formato anterior, tenemos una clave

labels en la que almacenamos las filas de la cuadrícula para que se refleje en la gráfica sobre qué fila estamos mostrando los datos. Por otro lado tenemos la clave *columns*, que contiene un array en el que se especifica para cada fila cuantas veces se ha seleccionado dicha columna.

El siguiente tipo de pregunta del cuál podemos mostrar gráficas es el tipo *Scale*. Este tipo sigue el formato que se muestra a continuación:

```
1 id_pregunta => array (
2   'labels' => array(
3     0 => '0',
4     1 => '1',
5     2 => '2'
6   ),
7   'values' => array (
8     0 => 1,
9     1 => 5,
10    2 => 3,
11  ),
12 )
```

La clave *labels* encapsula cada una de las opciones de la escala, mientras que la clave *values* la cantidad de veces que ha sido seleccionada cada opción.

A continuación tenemos el tipo de pregunta *Number*, que sigue el formato:

```
1 id_pregunta => array (
2   'labels' => array (
3     0 => '2',
4     1 => '34',
5   ),
6   'counts' => array (
7     0 => 1,
8     1 => 3,
9   ),
10 ),
```

En este formato al igual que el anterior, la clave *labels* encapsula cada una de los números que han sido introducidos, y la clave *counts* el número de veces que aparece cada número.

Por último tenemos los tipos de pregunta *Legal* y *YesNo*, que siguen el formato:

```
1 id_pregunta => array (
2   'labels' => array (
3     0 => 'No',
4     1 => 'Si',
5   ),
6   'count_options' => array (
7     0 => 0,
8     1 => 2,
9   ),
10 ),
```

En este formato, la clave *labels* encapsula siempre las dos únicas opciones que están disponibles en estos tipos de pregunta, y la clave *counts* el número de veces que ha sido seleccionada cada una de las preguntas.

Con toda esta información es posible generar las gráficas para poder analizar los resultados de una encuesta.

5.4. Configuración del entorno

Por último, y para acabar este apartado, voy a explicar paso a paso como he configurado el entorno de desarrollo. Para comenzar, he utilizado el siguiente software:

- Ubuntu 16.04.2. La configuración de los paquetes utilizados puede que no siga la misma estructura de directorios y ficheros de configuración en otras distribuciones linux.
- Distintos paquetes de PHP que detallaré más adelante en este apartado.
- Base de datos MySQL/MariaDB.
- Servidor Nginx del repositorio oficial de Ubuntu.
- Gestor de dependencias Bower. Para ello necesitaremos instalar NodeJS junto con el gestor de dependencias NPM.
- Algunos otros paquetes como OpenJDK.

Comencemos con la configuración del entorno:

1. Lo primero que debemos hacer es actualizar nuestro sistema, para ello utilizamos el siguiente comando:

```
1 sudo apt update
2 sudo apt upgrade -y
```

2. Una vez hemos actualizado los paquetes de nuestro sistema, vamos a pasar a instalar todos los paquetes necesarios para nuestro proyecto. Para ello usaremos el comando:

```
1 apt-get install -y nginx nano openjdk-8-jdk php7.0-cgi php7.0-cli
  php7.0-common php7.0-curl php7.0-fpm php7.0-json php7.0-
  mbstring php7.0-mysql php7.0-openssl php7.0-readline php7.0-
  xml git nodejs npm curl zip unzip php7.0-zip
```

3. A continuación, debemos instalar el gestor de paquetes Composer y hacer que sea accesible desde cualquier directorio:

```
1 curl -sS https://getcomposer.org/installer | sudo php -- --  
install-dir=/usr/local/bin --filename=composer
```

4. Lo mismo hacemos con el gestor de paquetes Bower:

```
1 sudo npm install -g bower  
2 sudo ln -s /usr/bin/nodejs /usr/bin/node
```

5. El siguiente paso será configurar algunos de los paquetes de PHP usando estos comandos. Donde pone *nombre_usuario* debemos poner el nombre de usuario de nuestro sistema.

```
1 #File: /etc/php/7.0/fpm/php.ini  
2 cd /etc/php/7.0/fpm/  
3 sudo sed -i "s/;cgi.fix_pathinfo=1/cgi.fix_pathinfo=0/g" php.ini  
4 sudo sed -i "s/upload_max_filesize = 2M/upload_max_filesize = 200  
M/g" php.ini  
5 sudo sed -i "s/post_max_size = 8M/post_max_size = 200M/g" php.ini  
6 #File: /etc/php/7.0/fpm/pool.d/www.conf  
7 cd /etc/php/7.0/fpm/pool.d/  
8 sudo sed -i "s/user = www-data/user = nombre_usuario/g" www.conf  
9 sudo sed -i "s/group = www-data/group = nombre_usuario/g" www.  
conf  
10 sudo sed -i "s/listen.owner = www-data/listen.owner =  
nombre_usuario/g" www.conf  
11 sudo sed -i "s/listen.group = www-data/listen.group =  
nombre_usuario/g" www.conf
```

6. Ahora vamos a configurar nuestro servidor Nginx para que podamos acceder a nuestro proyecto Laravel. Este puede estar situado en cualquier directorio de nuestro sistema, sin embargo yo lo voy a situar donde los sitúa por defecto Nginx: en el directorio */var/www/html*. En este directorio pondré el proyecto dentro de una carpeta con nombre *tfg*. Un punto muy importante, es que debemos ser dueños de este directorio y darle permisos:

```
1 sudo chown -R pablo:pablo /var/www/html/tfg
```

Tras ello, vamos a ver como debemos configurar Nginx. Lo que debemos hacer es situarnos en el directorio de configuración de Nginx y crear un nuevo fichero que establezca la configuración de nuestra web:

```
1 cd /etc/nginx/sites-enabled/  
2 sudo touch tfg.com
```

En este fichero */var/www/html/tfg.com* debemos establecer la siguiente configuración:

```

1 server {
2     listen 80 default_server;
3     listen [::]:80 default_server;
4     root /var/www/html/tfg/public;
5     index index.php index.html index.htm index.nginx-debian.html;
6     server_name tfg.com;
7
8     location / {
9         try_files $uri $uri/ /index.php?$query_string;
10
11     location ~ \.php$ {
12         try_files $uri /index.php =404;
13         fastcgi_split_path_info ^(.+\.php)(/.+)$;
14         fastcgi_pass unix:/run/php/php7.0-fpm.sock;
15         fastcgi_index index.php;
16         fastcgi_param SCRIPT_FILENAME $document_root\
17             $fastcgi_script_name;
18         include fastcgi_params;
19     }
20 }

```

Puesto que ya tenemos otro servidor por defecto en Nginx: */etc/nginx/sites-enabled/default*, lo eliminamos para evitar problemas. Con todo esto ya tenemos nuestro servidor configurado, ahora solo queda ejecutar estos comandos:

```

1 systemctl restart nginx
2 systemctl restart php7.0-fpm.service

```

También tenemos que configurar algunas parámetros de su configuración general:

```

1 cd /etc/nginx/
2 sudo sed -i "s/user www-data;/user ubuntu;/g" nginx.conf
3 sudo sed -i 's/worker_processes 4;/worker_processes 1;/g' nginx.conf
4 sudo sed -i 's/worker_connections 768;/worker_connections 1024;/g' nginx.conf
5 sudo sed -i "11i\ client_max_body_size 100m;" nginx.conf
6 sudo sed -i "11i\ fastcgi_buffers 16 16k;" nginx.conf
7 sudo sed -i "11i\ fastcgi_buffer_size 16k;" nginx.conf
8 sudo sed -i "11i\ proxy_busy_buffers_size 256k;" nginx.conf
9 sudo sed -i "11i\ proxy_buffer_size 128k;" nginx.conf
10 sudo sed -i "11i\ proxy_buffers 4 256k;" nginx.conf
11 sudo sed -i "11i\ gzip_vary on;" nginx.conf
12 sudo sed -i "11i\ gzip_proxied any;" nginx.conf
13 sudo sed -i "11i\ gzip_comp_level 6;" nginx.conf
14 sudo sed -i "11i\ gzip_buffers 16 8k;" nginx.conf
15 sudo sed -i "11i\ gzip_http_version 1.1;" nginx.conf
16 sudo sed -i "11i\ gzip_types text/plain text/css
    application/json application/x-javascript text/xml
    application/xml application/xml+rss text/javascript;" nginx.conf
17 sudo sed -i "11i\ " nginx.conf

```

7. Ahora pasaremos a configurar la aplicación. Lo primero que debemos hacer es descargar todas las dependencias, para ello nos situamos en el directorio `/var/www/html/tfg` y ejecutamos:

```
1 composer install
2 bower install
```

Una vez hemos descargado las dependencias, tenemos que configurar la conexión con la base de datos. Laravel incluye un fichero denominado *.env.example* que contiene los parámetros de configuración de la aplicación. Primero debemos renombrar este fichero a *.env*. Una vez lo hemos hecho, entramos y establecemos la configuración de nuestra base de datos en los siguientes parámetros:

```
1 DB_CONNECTION=mysql
2 DB_HOST=127.0.0.1
3 DB_PORT=3306
4 DB_DATABASE=database
5 DB_USERNAME=user
6 DB_PASSWORD=password
```

Tras ello, guardamos y ejecutamos el siguiente comando:

```
1 php artisan key:generate
```

8. Por último, creamos todas las tablas de la aplicación:

```
1 php artisan migrate --seed
```

Con todo esto ya podemos acceder a nuestra aplicación escribiendo en la barra de direcciones de navegador *localhost*. Adjuntaré un script para realizar todo este proceso de forma automática sobre Ubuntu 16.04.2. Para ejecutar este script es necesario hacerlo como superusuario.

Capítulo 6

Manual

En este apartado explicaré brevemente como funciona la aplicación a través de distintas capturas de pantalla de la misma.

Para comenzar, tenemos la página inicial. En ella únicamente podemos hacer dos cosas: pasar a la página de identificación o a la página de registro:

Figura 6.1: Página de inicio de la aplicación.

LOGIN REGISTRO

TFG Pablo

Si pulsamos sobre el enlace que nos lleva a la página para identificarnos, nos llevará a esta página:

Figura 6.2: Página para identificarse.

TFG Pablo Login Registro

Login

Correo

Contraseña

☐ Recuérdame

[Acceder](#) [¿Hás olvidado tu contraseña?](#)

En ella podremos ingresar nuestro correo electrónico y nuestra contraseña para acceder al sistema.

Por otra parte, si hacemos clic en el enlace que nos lleva a la página de registro nos encontraremos con la siguiente página:

Figura 6.3: Página para registrarse.

TFG Pablo Login Registro

Registro

Nombre de usuario

Correo

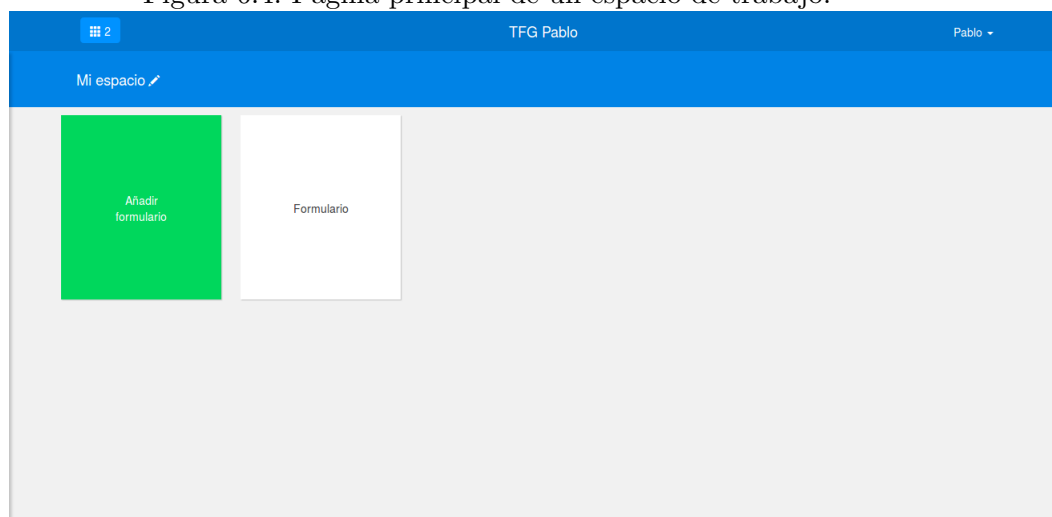
Contraseña

Confirmar contraseña

[Registrarse](#)

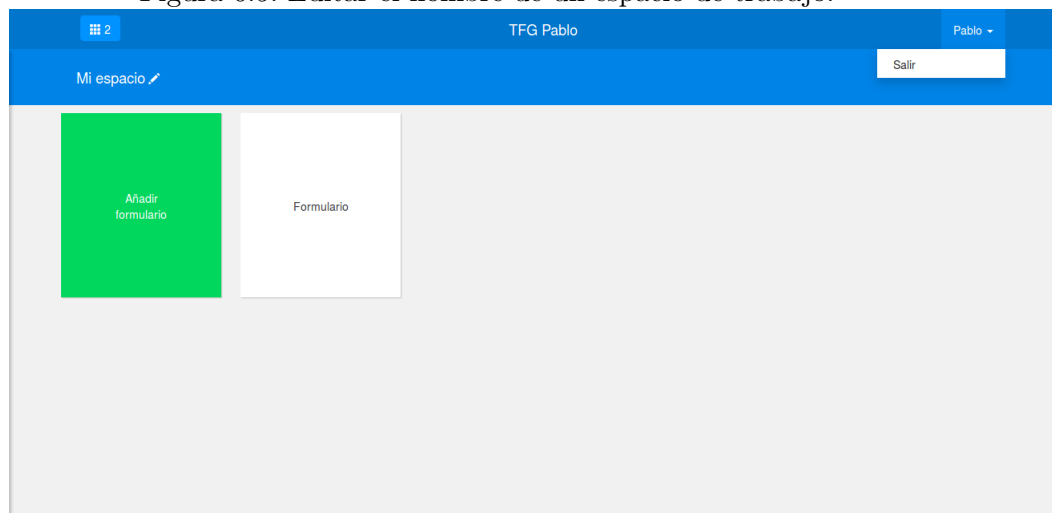
Una vez hayamos iniciado sesión, se nos redirige a nuestro espacio de trabajo por defecto, en el que podemos gestionar los formularios que tiene asociados (eliminarlos, crear uno nuevo y acceder a ellos):

Figura 6.4: Página principal de un espacio de trabajo.



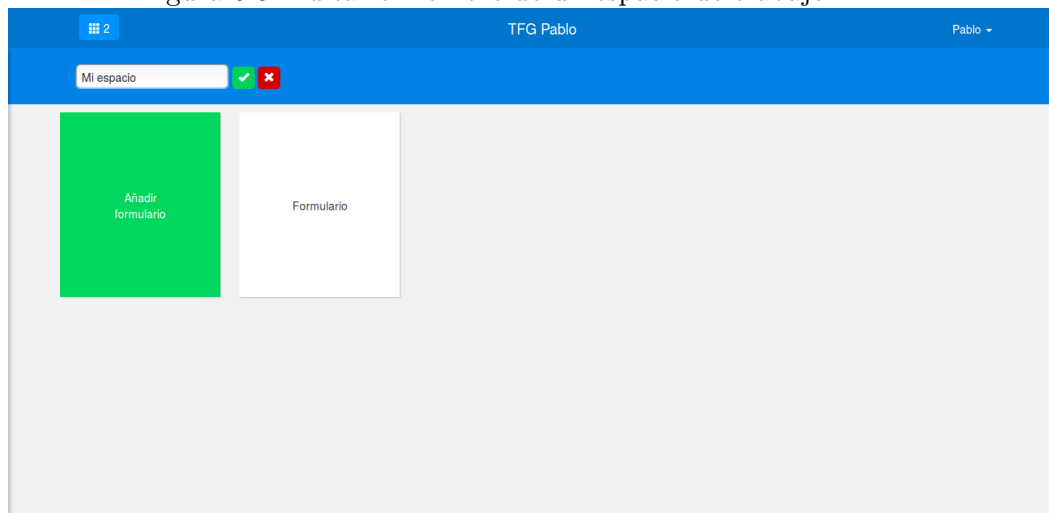
Desde esta página o cualquier otra, podemos hacer clic en el nombre de nuestro usuario y cerrar la sesión en el sistema:

Figura 6.5: Editar el nombre de un espacio de trabajo.



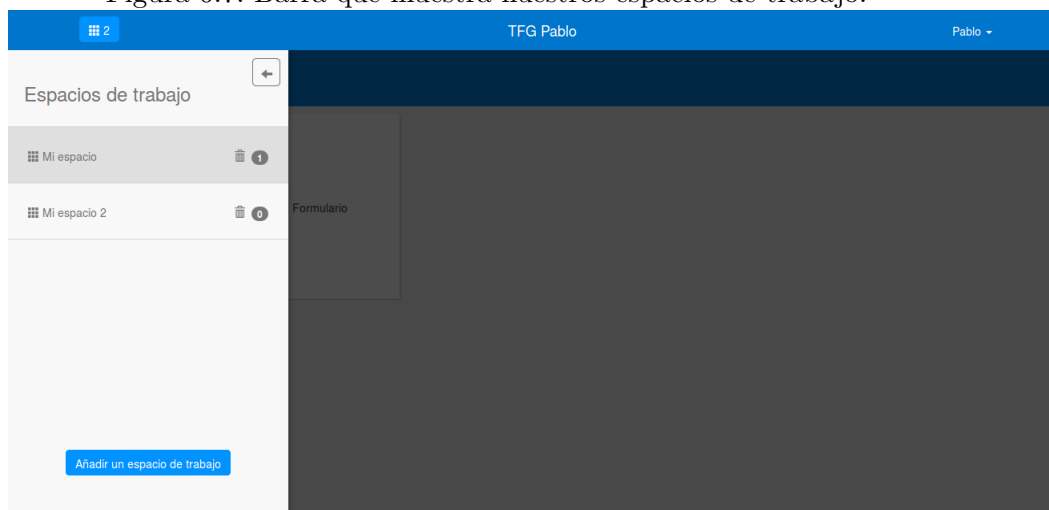
Además, desde la página que muestra un espacio de trabajo, si hacemos clic sobre el nombre de este, nos aparecerá una entrada para editarlo:

Figura 6.6: Editar el nombre de un espacio de trabajo.



Una vez estamos identificados en el sistema, desde cualquiera de sus páginas podemos ver los espacios de trabajo que tenemos disponible haciendo clic sobre el icono que aparece en la parte izquierda barra superior. Este icono muestra indica los espacios de trabajo que tenemos actualmente asociados a nuestro usuario, y además muestra cada uno de ellos:

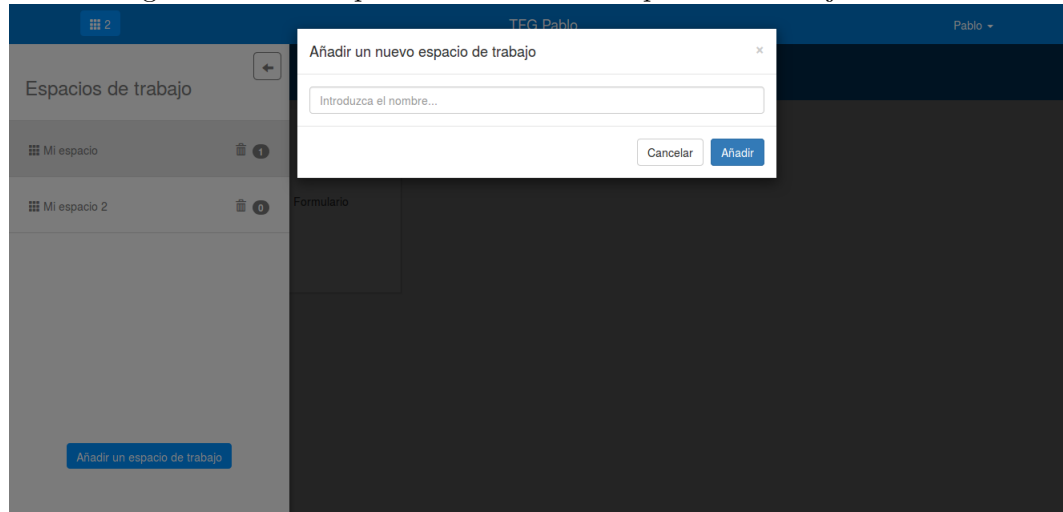
Figura 6.7: Barra que muestra nuestros espacios de trabajo.



Desde esta barra lateral podemos acceder a uno de nuestros espacios de trabajo además de eliminar el que deseemos, siempre y cuando este no sea

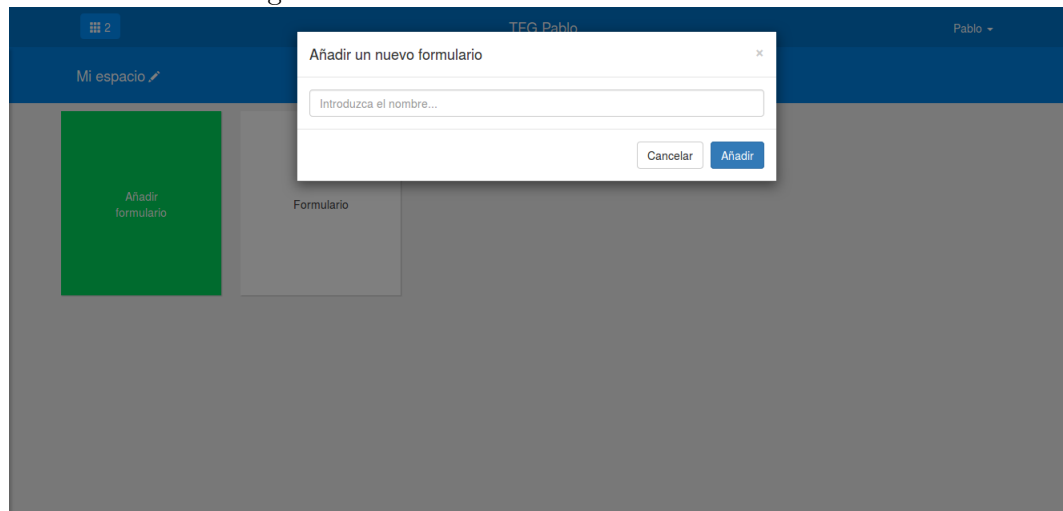
nuestro espacio de trabajo por defecto. En la parte inferior de la barra lateral tenemos un botón que nos permite crear un nuevo espacio de trabajo:

Figura 6.8: Modal para crear un nuevo espacio de trabajo.



Si hacemos clic en el botón que pone *Añadir formulario*, aparece un modal con el que podemos crear un nuevo formulario asociado a este espacio de trabajo. Una vez creado, se nos redirige a él:

Figura 6.9: Crear un nuevo formulario.



Ya por último, si hacemos clic en uno de los formularios que aparece en la página de un determinado espacio de trabajo, se nos redirigirá a la página para contruir el formulario:

Figura 6.10: Página para diseñar nuestros formularios.

The screenshot shows the 'Formulario' (Form) design page in the TFG Pablo application. The interface has a blue header with the user's name 'Pablo' and a 'Ver formulario' button. Below the header, there are tabs for 'Construir' (Build), 'Compartir' (Share), and 'Analizar' (Analyze). On the left, there is a sidebar with various question types: Respuesta corta (Short answer), Respuesta larga (Long answer), Elección múltiple (Multiple choice), Desplegable (Dropdown), Cuadrícula (Grid), Escala de opinión (Opinion scale), Texto (Text), Número (Number), Email, Página web (Website), Fecha (Date), Hora (Time), Si/No (Yes/No), and Legal. The main area on the right shows a list of questions added to the form, including 'Respuesta corta', 'Respuesta larga', 'Elección', and 'Cuadrícula'. At the bottom of this list is a button that says 'Añade nuevas preguntas' (Add new questions).

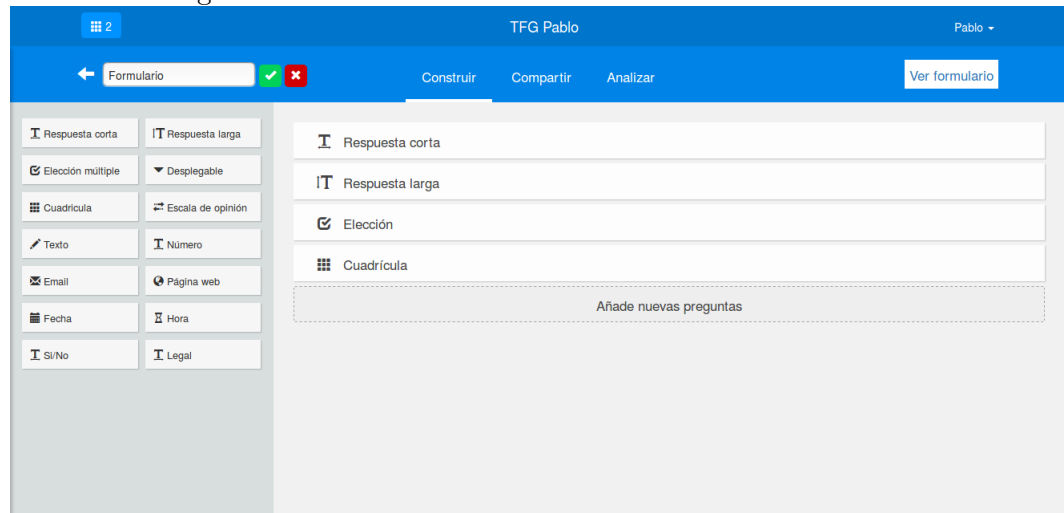
En esta página podemos crear tantas preguntas como deseemos de los distintos tipos disponibles, además de cambiar el orden en el que están mantenido pulsando sobre una de ellas. Si hacemos clic sobre un tipo o sobre una pregunta, nos aparecerá un modal sobre el que podremos crear o editar respectivamente una pregunta:

Figura 6.11: Modal para crear un nuevo espacio de trabajo.

The screenshot shows the 'Elección múltiple' (Multiple choice) modal for creating a new question. The modal has a title bar with a close button. It contains several fields: 'Pregunta' (Question) with a text input, 'Descripción' (Description) with a text input, and 'Opciones' (Options) with a green button labeled 'Añadir una opción' (Add an option). Below the options, there are two radio button groups: 'Imagen/video' (Image/video) with options 'Si' (Yes) and 'No' (No), and 'Obligatoria' (Mandatory) with options 'Si' (Yes) and 'No' (No). At the bottom right of the modal are two buttons: 'Cancelar' (Cancel) and 'Guardar' (Save).

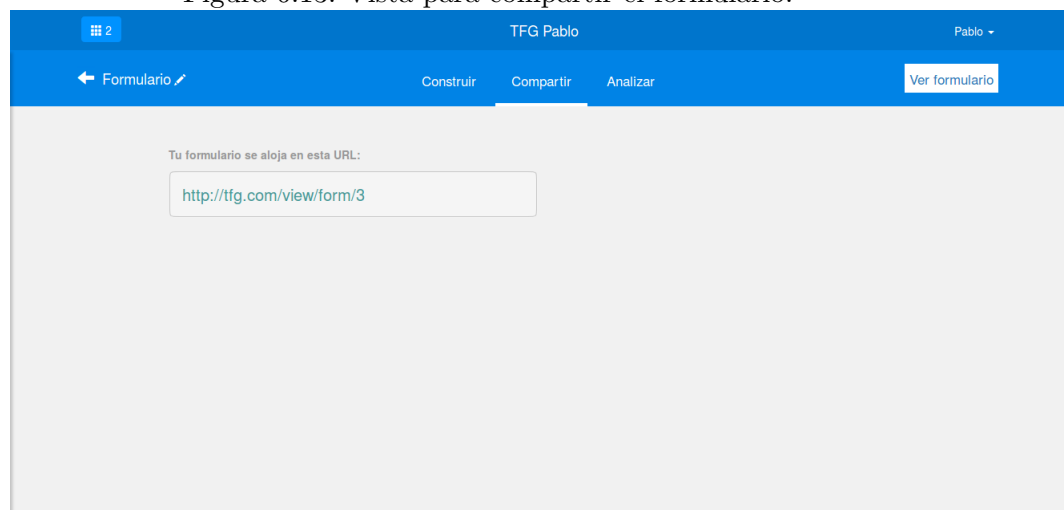
Al igual que ocurre con los espacio de trabajo, podemos cambiar el nombre de un formulario haciendo clic sobre su nombre:

Figura 6.12: Cambiar el nombre de un formulario.

The screenshot shows the 'Formulario' (Form) view in the TFG Pablo application. The top navigation bar is blue and contains the text 'TFG Pablo' and a user profile 'Pablo'. Below this, a secondary bar shows the current view 'Formulario' with a back arrow, a green checkmark, and a red X icon. The main navigation tabs are 'Construir', 'Compartir', and 'Analizar', with 'Construir' being the active tab. On the right of this bar is a 'Ver formulario' button. The left sidebar is a grey panel with a grid of question type icons: 'Respuesta corta', 'Respuesta larga', 'Elección múltiple', 'Desplegable', 'Cuadrícula', 'Escala de opinión', 'Texto', 'Número', 'Email', 'Página web', 'Fecha', 'Hora', 'Si/No', and 'Legal'. The main content area is white and shows a list of questions being edited: 'Respuesta corta', 'Respuesta larga', 'Elección', and 'Cuadrícula'. Below the list is a dashed box with the text 'Añade nuevas preguntas'.

Como se puede observar, en la vista para gestionar el formulario, dispone de una barra adicional en la que podemos navegar entre las distintas opciones de un formulario. Por ejemplo, si hacemos clic en el enlace *Compartir*, se nos muestra una pantalla con la URL de nuestro formulario público con la cual podremos realizar nuestras encuestas:

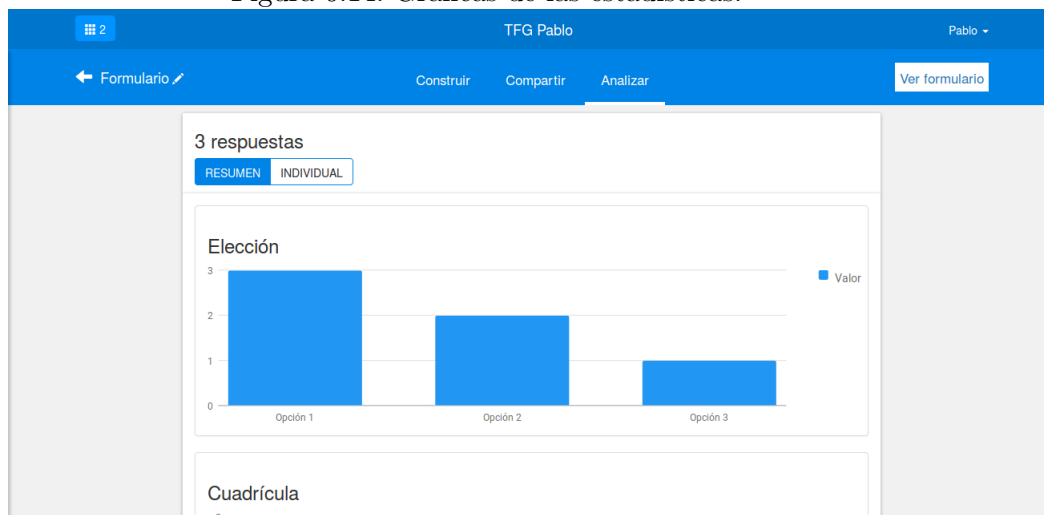
Figura 6.13: Vista para compartir el formulario.

The screenshot shows the 'Compartir' (Share) view in the TFG Pablo application. The top navigation bar is the same as in Figure 6.12, but the 'Compartir' tab is now active. The main content area is white and displays the message 'Tu formulario se aloja en esta URL:' followed by a text box containing the URL 'http://tfg.com/view/form/3'.

La última de las opciones de un formulario es la de ver los análisis de sus resultados. Para ello hacemos clic en el enlace de *Analizar*. Inicialmente

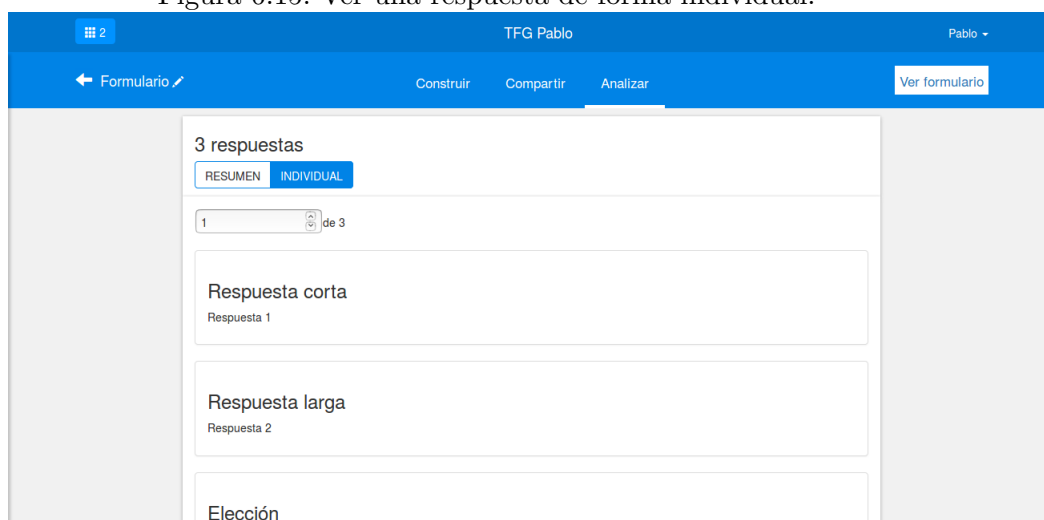
nos aparecerá una vista con las gráficas de las estadísticas globales de las respuestas para el formulario:

Figura 6.14: Gráficas de las estadísticas.



La otra opción que tenemos es la de ver las respuestas de forma individual:

Figura 6.15: Ver una respuesta de forma individual.



Y ya para finalizar este apartado, las encuestas tienen el siguiente aspecto y su diseño es *responsive*:

Figura 6.16: Formulario público.

The image shows a public survey form with a blue border. The form is divided into two main sections: 'Elección' and 'Cuadrícula'. The 'Elección' section contains three radio button options: 'Opción 1', 'Opción 2', and 'Opción 3'. The 'Cuadrícula' section contains a table with two columns, 'Columna 1' and 'Columna 2', and two rows, 'Fila 1' and 'Fila 2'. Each cell in the table contains a checkbox. At the bottom right of the form is a button labeled 'ENVIAR'.

	Columna 1	Columna 2
Fila 1	<input type="checkbox"/>	<input type="checkbox"/>
Fila 2	<input type="checkbox"/>	<input type="checkbox"/>

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

Como conclusión, podemos sacar varios aspectos. El primero de ellos es que durante el desarrollo se ha intentado seguir unas buenas prácticas de programación y diseño para hacer un software con una mínima calidad. Para ello me he servido de los patrones que aplica el framework del back-end.

La aplicación contiene una funcionalidad muy clara e intuitiva, que busca focalizar el objetivo principal: diseñar encuestas. Sin embargo, me hubiese gustado que fuese mucho más personalizable para el usuario en ciertos aspectos tanto de funcionalidad como de interfaz. Es en este último punto, donde se ha buscado que la aplicación fuese lo más sencilla y visualmente atractiva posible para el usuario, siguiendo las pautas habituales de este tipo de aplicaciones en lo que a la experiencia de usuario se refiere.

Y ya para acabar y desde un punto de vista personal, la elección de las tecnologías que he tomado para desarrollar la aplicación no solo se ha debido a que estoy familiarizado con ellas, sino que pienso que la tendencia en las empresas es la de usar habitualmente muchas de ellas, lo que me ayudará a estar más preparado llegado el momento.

7.2. Trabajos futuros

Respecto a las mejoras futuras que puede tener el proyecto, estas son bastantes. Por un lado, podemos crear una parte para los administradores del sistema, pudiendo gestionar de esta forma todo el contenido de la web incluyendo a los usuarios, sus espacio de trabajo, etc. En cuanto a la funcionalidad, tal y como he dicho en apartado anterior, se podría dar la posibilidad de que los formularios fuesen más personalizables adaptándolos de esta forma a los usuarios objetivos de la encuesta. Además se podría dar

la posibilidad de añadir múltiples usuarios a un mismo espacio de trabajo para que los usuarios pudieran trabajar de forma conjunta sobre un mismo formulario. Ya para terminar este apartado, también se podría integrar el sistema con otras plataformas a la hora de compartir nuestros formularios, como por ejemplo en redes sociales.

