ESCUELA POLITÉCNICA

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

Diseño e implementación de sistema de obtención de opinión de usuarios

Oscar Valle Diaz. Septiembre, 2018

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

Diseño e implementación de sistema de obtención de opinión de usuarios

Autor: Oscar Valle Diaz

Tutor: Antonio Gordillo Guerrero

Tribunal Calificador

Presidente: Pablo García Rodríguez

Secretario: Marino Linaje Trigueros

Vocal: Horacio González Velasco

A todos los que han inspirado mi pasión por la tecnología.

A mis profesores y compañeros, colaborando siempre.

A mi familia, por poner los cimientos.

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Actualmente, empresas e instituciones tienen dificultades para obtener y registrar de forma digital la opinión de los usuarios de sus servicios físicos/reales '*just in the moment'* (justo en el momento).

Hemos analizado, diseñado y desarrollado una solución mínima viable a través del uso conjunto de tecnologías web, para gestionar y analizar los datos, y componentes IOT (Internet de las cosas), para desarrollar dispositivos con conectividad Wifi y GSM en los que los usuarios puedan responder distintas cuestiones. Todo ello utilizando tecnologías muy óptimas en cuanto a costes tanto de recursos máquina, electrónica o tiempos de desarrollo.

Las metodologías de desarrollo seguidas nos han ayudado en la correcta implementación de la plataforma web y el rápido desarrollo de los prototipos.

La finalidad del sistema cumple con lo esperado y es útil para el usuario según las pruebas realizadas sobre la plataforma web que se encuentra puesta en marcha en un entorno de producción y con los diferentes dispositivos.

SUMMARY

Nowadays, companies and institutions have some difficulties to getting and recording the users' opinion of their physical/real services in digital form just in the moment.

We have analyzed, designed and developed a viable minimum solution through the joint use of web technologies to manage and analyze data, and IOT (Internet Of Things) components to develop devices with Wifi and GSM connectivity in which users can answer different questions. All of this using very optimal technologies in terms of cost, machine resources, electronics or development times.

The development methodologies followed have helped us in the correct implementation of the web platform and the rapid development of the prototypes.

The purpose of the system complies with expectations and it is useful for the user according to the tests carried out on the web platform, which it is set up in a production environment, and with the different devices.

1. INTRODUCCIÓN

CAPÍTULO 1 INTRODUCCIÓN

Las encuestas de opinión se realizan desde hace siglos. El primer estudio de opinión o encuesta política [1] fue realizada en 1824 por el periódico *The Harrisburg Pennsylvanian*, sin valor científico, que pretendía medir la preferencia de voto entre dos aspirantes a la Presidencia de los Estados Unidos [2].

Años más tarde, en 1916, el *Literary Digest Magazine* realizó mediciones de carácter nacional y con ellas consiguió predecir la victoria del Presidente Woodrow Wilson [2]. El sondeo consistió en el envío de tarjetas postales que debían ser respondidas y devueltas por correo postal. Con ello consiguieron predecir esta y las siguientes cuatro elecciones presidenciales.

En 1960, la metodología que era utilizada en estas encuestas de opinión evolucionó a método de investigación, enriqueciendo así el ámbito de las ciencias sociales y siendo una más de las herramientas en su labor como ciencia del estudio de la actividad humana en la sociedad [3].

En la actualidad, podemos ver diversos ejemplos de este tipo de estudios de opinión o encuestas, como las registradas por el Instituto Nacional de Estadística [4] o en el Centro de Investigaciones Sociológicas [5], que anualmente publican diferentes resultados de ámbitos políticos y regionales.

Según la RAE (Real Academia Española) el término '*Encuesta*' se define como [6]:

1. f. Conjunto de preguntas tipificadas dirigidas a una muestra representativa de grupos sociales, para averiguar estados de opinión o conocer otras cuestiones que les afectan.

2. f. Indagación o pesquisa.

Esta segunda definición, '*Indagación o pesquisa*', es más abierta y nos puede dar una visión más amplia. La propia RAE define '*Indagar*' como [7] "*Intentar averiguar algo discurriendo o con preguntas.*".

¿Quién necesita, y en muchos casos se ve en la obligación, de averiguar algo con preguntas de forma constante actualmente?

Las empresas y organizaciones que quieren cumplir con normativas como la ISO 9001 (Sistemas de gestión de la calidad) [8], en la que deben medir de forma constante diferentes parámetros relevantes para sus tareas. En algunos casos no es necesario hacer preguntas pues hay datos objetivos, técnicos y claros, pero en otros muchos no es posible y se necesitan encuestas para obtener estos datos.

Algunas de las referencias bibliográficas más destacadas son los libros "*Cómo medir la satisfacción del cliente según la ISO 9001:2000. Segunda edición*" [9] publicado en 2003 por *Fundación Confemetal* y "*Gestión de la calidad (ISO 9001/2008) en el comercio*" [10] publicado en 2010 por *EQUIPO VÉRTICE*. Ambas obras aportan el mismo enfoque y comentan la relevancia de poder medir la opinión o satisfacción del cliente mediante encuestas y encuestas de satisfacción para poder cumplir algunos de los objetivos que plantea la ISO 9001.

La relación entre las encuestas y la estadística es tan íntima que en algunos casos son denominadas como encuestas estadísticas. La estadística, y en especial la encuesta, a nivel empresarial se puso de moda como herramienta en los años cuarenta del siglo XX por su uso en estudios de mercado en Estados Unidos, después de la segunda guerra mundial [11].

Como podemos observar existen diferentes formas de nombrar a la encuesta, pero ya sea para uso político, social o empresarial solo se diferencian por su objetivo y tipología.

Es posible clasificar las encuestas por sus tipos de preguntas: abiertas, cerradas, semicerradas o semiabiertas, por su modalidad: encuesta personal, telefónica, postal, etc [11]. Pero una forma algo más completa de clasificarlas es considerando su dimensión temporal dividiéndolas básicamente en diseño transversal o diseño longitudinal. [12]

El diseño transversal describe la realidad en un momento determinado, y el diseño longitudinal recoge información a lo largo del tiempo. El primero es más fácil, sencillo y económico de forma general. Y el segundo es más costoso en tiempo y dinero, pero tiene una alta validez interna. [12]

Vidal Díaz de Rada, doctor en Ciencias Políticas y Sociología por la Universidad de Deusto, y profesor de Técnicas de Investigación Social en la Universidad Pública de Navarra publicó en el 2000 una nota de investigación sobre la utilización de nuevas tecnologías para la encuesta, cuya conclusión concuerda con otros estudios (trabajo de Nicholls y colaboradores): "*Los «nuevos» procedimientos de recogida de información aumentan notablemente la calidad de los datos recogidos*" [13]

El mismo autor comenta en otro artículo en 2010 [14] una conclusión similar, pero poniendo énfasis en el sesgo que genera hacer una encuesta por Internet al presentarse un perfil más joven de encuestados. Concluye que lo mejor es la consideración conjunta de encuestas presenciales y vía web, ya que la unión de datos de ambas se acerca mejor a la realidad que se intenta medir. [14]

Por todo ello este Trabajo Fin de Grado pone una especial atención en el uso de nuevas tecnologías, que ahora a través del hardware y sistemas de comunicación más económicos, nos permiten hacer encuestas presenciales de forma eficaz y eficiente.

1.1 Justificación y motivación del proyecto

Este Trabajo Fin de Grado persigue el estudio de un problema real y la búsqueda e implementación completa de una solución mínima viable, es decir, que pueda ser usada.

Nuestro problema base es: la dificultad de obtener y registrar de forma digital la opinión de usuarios de un servicio físico/real cualquiera '*just in the moment'*, justo en el momento.

Con las herramientas web actuales como Google Forms o TypeForm es bastante sencillo hacer consultas a posteriori sobre un tema o experiencia vivida. Por dar un ejemplo general, es posible y bastante fácil enviar a los asistentes de un congreso un cuestionario por email al día siguiente y realizar cualquier pregunta sobre su satisfacción o preferencias. Pero es más

complejo hacerlo de forma digital justo en el momento, justo después de la comida, o durante el acceso o salida del recinto, etc.

Para obtener datos en el momento, lo habitual es tener a personas en puntos específicos o recorriendo el lugar y preguntando, muchas veces apuntando a mano la respuesta en algún medio físico como libretas o cuadernos.

Obviando que en muchos casos es más eficaz el preguntar en el momento pues algunos usuarios no tienen forma de contacto digital, no querrán proporcionarla o simplemente es más cómodo para ellos.

En la actualidad un cuestionario en diferido sea por email, teléfono, carta u otros medios necesita del conocimiento de datos personales protegidos por la conocida como LOPD actualmente en vigor RGPD (Régimen General de Protección de Datos) [15] haciendo el proceso de consulta diferida más complejo.

Por ello se ha realizado esta propuesta de estudio, análisis, diseño e implementación de un sistema completo (tanto software como hardware funcional) que intente resolver este problema de la forma más acertada posible.

Es interesante destacar que esta propuesta puede obtener la opinión sobre unas cuestiones de forma no enlazada con datos personales y por tanto no necesita cumplir con el RGPD [16]. Excluyendo los casos en los que alguna de las cuestiones sea la petición de datos personales directos.

Por lo comentado anteriormente la propuesta también goza de la tranquilidad de no tener que justificar las famosas cookies o seguir las obligaciones sobre seguridad actuales, pues el usuario final (externo/encuestado) en nuestra propuesta no está navegando ni usando un dispositivos propios, está usando un dispositivo de la organización especialmente diseñado para recoger la opinión de los usuarios sin entrar en conflicto con ninguno de sus derechos.

1.2 Propuesta

Se propone el estudio y desarrollo de un sistema que permita al usuario de un servicio cualquiera, responder diferentes cuestiones justo en el momento; y a los gestores analizar esos datos. Todo vía web y soportado sobre hardware para la recogida de datos, implementado con tecnologías actuales, es decir cuyo soporte siga vigente y su uso este demostrado, y usar en la medida de lo posible tecnologías abiertas u *open source*.

Se priorizará el uso de tecnologías eficientes en cuanto a recursos máquina y especialmente en tiempo de desarrollo o curva de productividad.

Se plantea una clara separación entre la solución software o plataforma web, y la solución hardware o dispositivos/prototipos. Conectando ambas a través del desarrollo de una interfaz o API, incluida en la plataforma web.

La solución software propuesta se estudiará y desarrollará siguiendo el Proceso Unificado de Desarrollo de Software [16] dirigido desde los requisitos básicos del sistema y centrado en la arquitectura principal que dará soporte a la plataforma web.

No se prestará demasiada atención a las últimas connotaciones que este marco de desarrollo de software propone, como las iteraciones o ciclos, por el alcance del proyecto.

La solución hardware propuesta se estudiará y desarrollará siguiendo el Modelo de Prototipos [17], dentro del ámbito de la Ingeniería del Software. Como sus propios principios definen es necesario construir los prototipos en poco tiempo, usando las tecnologías adecuadas y sin utilizar demasiados recursos para así poder probar su funcionamiento de forma real lo antes posible.

La plataforma web propuesta se centrará en la configuración, guardado y visualización de datos para su correcto uso y análisis por los gestores o usuarios internos.

Los dispositivos y prototipos propuestos se centrarán en facilitar la inserción de datos por usuarios externos o clientes, es decir por ser el soporte físico o

interfaz real sobre la que interactuaran los usuarios para responder las cuestiones definidas.

1.3 Alcance

El estudio y desarrollo del sistema conlleva la realización de muchas fases y tareas diferentes. Por un lado, la plataforma web debe seguir una metodología algo más estricta que los prototipos, y su alcance es mayor pues los prototipos buscan ser lo más sencillos posible y la plataforma necesita tener varias funcionalidades para que sea útil para el usuario interno o gestor, y también debe poder administrase por usuarios administradores.

La plataforma web por si misma puede abarcar una gran cantidad de horas de desarrollo y documentación, por ello nos centraremos en las funcionalidades básicas necesarias, como más adelante recogerá la lista de requisitos, dejando en segundo lugar las mejoras visuales y las optimizaciones profundas. También por motivos de la extensión del presente proyecto no vamos a adentrarnos en altos niveles de seguridad, pero si se buscará garantizar la separación de datos entre grupos de usuarios y la no suplantación entre los mismos.

Los dispositivos se trabajarán en dos ámbitos, los dispositivos completos existentes en el mercado sobre los cuales el sistema se puede usar (Laptops, tablets, etc) y los prototipos propios. En el caso de los prototipos se buscará hacer un desarrollo rápido en dos versiones una con conectividad Wifi y otro con conectividad GSM. Sería posible hacer diferentes versiones de ambos, e incluso con otros tipos de conectividad, pero solo estudiaremos y fabricaremos una, la que pueda cumplir con los requisitos que se analicen y pueda implementarse rápidamente y a bajo coste.

Los dispositivos existentes en el mercado compatibles con el sistema disponen de pantalla y pueden mostrar una relación de preguntas que van pasando una tras otras, en el caso de los prototipos esta funcionalidad no se plantea, sólo mostraran una pregunta y sus diferentes opciones.

En cuanto a la documentación se crearán todos los documentos básicos para poder entender y mantener el sistema, algunos de estos documentos estarán incluidos directamente como apartados de este proyecto.

1.4 Notas sobre el documento

Por la extensión y las diferentes áreas del proyecto, software/hardware y análisis/implementación, se van a utilizar un número relativamente grande de apartados diferentes. El capítulo más extenso será el capítulo 5, Implementación y Desarrollo, en él se contarán todos los detalles del proceso, desde el estudio y análisis de requisitos hasta la implementación tanto de la plataforma web como de los prototipos.

Este quinto apartado contiene también secciones específicas sobre la visión de los diferentes tipos de usuarios administradores, gestores y externos. Estos subapartados nos ayudaran a tener una idea más clara de la realidad del uso del sistema.

Los capítulos 1, 2 y 3 son introductorios, el cuarto comenta las metodologías usadas para el estudio y desarrollo del sistema, y los capítulos 6 y 7 presentan los resultados y la conclusión.

2. OBJETIVOS

CAPÍTULO 2 OBJETIVOS

Dividimos el capítulo en objetivo general y objetivos específicos para tratar mejor los objetivos del software y del hardware de forma diferenciada.

2.1 Objetivo General

Estudiar y desarrollar un sistema que permita al usuario de un servicio físico cualquiera, responder diferentes cuestiones '*just in the moment*', justo en el momento; y a los gestores analizar esos datos. Todo vía web y soportado sobre hardware.

Como detalle ampliado:

El objetivo de la plataforma web es dar acceso a datos y configuraciones a los gestores, y poder recibir los datos desde diferentes dispositivos existentes en el mercado y de prototipos propios.

El objetivo de los prototipos propios a diseñar y fabricar es el ahorro de costes para el uso del sistema en algunos casos, tanto en su versión Wifi como GSM.

2.2 Objetivos Específicos

a) Estudiar el ámbito del problema propuesto.

b) Generar una lista de requisitos y análisis para el desarrollo de la solución software.

c) Diseñar un sistema software completo de obtención de opinión de usuarios desde los tres puntos de vista principales: a*dministrador, gestor, externo*.

d) Generar una lista de requerimientos mínimos de los dispositivos.

e) Evaluar las diferentes soluciones hardware de forma analítica respecto a los requerimientos mínimos de los dispositivos y diseñar los prototipos.

f) Desarrollar la solución software (plataforma web) y hardware propuesta (dispositivos propios/prototipos).

g) Documentar las soluciones para su fácil mantenimiento.

h) Puesta en marcha: Llevar a un entorno de producción la solución software y fabricar los distintos prototipos hasta llegar a una versión válida para su uso real.

3. ANTECEDENTES

CAPÍTULO 3 ANTECEDENTES

Para poder tomar las decisiones correctas en el desarrollo de cualquier proyecto siempre es necesario ver los antecedentes, ver los procesos y avances anteriores de otros equipos de investigación, pensadores, compañías o instituciones.

En este apartado vamos a plasmar la información más relevante sobre nuestro tema de estudio. En definitiva, nuestro tema de estudio tiene dos antecedentes claros: la evolución de las encuestas de opinión como tal y la tecnología que lleva haciendo que su realización avance y cambie a lo largo de la historia.

Hemos determinado que mirar muy atrás en el tiempo no aportaba más valor que el ya mostrado en la introducción pues ya conocemos la importancia de las encuestas de opinión y desde cuando se llevan realizando para diferentes cuestiones.

Al igual que la imprenta y la mejora en el transporte de la correspondencia fueron elementos relevantes para la realización de cuestionarios masivos, el gran punto de inflexión sobre la realización de cuestionarios fue la gran acogida de Internet. Desde ese momento se han creado muchas herramientas diferentes para la realización de encuestas en línea, y para el análisis de los datos vertidos de las mismas.

Por la extensión del proyecto no es posible adentrarnos demasiado en el análisis de datos, pero la forma en la que se realizan encuestas “en línea” amplia en gran medida la cantidad de datos que se pueden recolectar, en el pasado lo único que se obtenía de una encuesta eran sus respuestas, ahora es posible obtener datos como la hora o localización en el momento de la respuesta.

Nuestra propuesta y este proyecto intenta dar un paso más en la realización de cuestionarios gracias a otra revolución, la IOT (Internet de las cosas) pues podemos apaciguar las desventajas comentadas en la introducción como el sesgo que genera hacer una encuesta por Internet al presentarse un perfil más joven de encuestados [14], y aprovecharnos del gran poder de

recolección de datos y accesibilidad que tienen los sistemas conectados a Internet.

3.1 El mundo lo usa

Comencemos por ver la relevancia que ha tenido la tecnología para hacer más accesible la realización de cuestionarios.

En el artículo “Eficacia de las encuestas por Internet” [14] se comentan cinco grandes ventajas de los cuestionarios en línea: gran rapidez, mayor calidad de la información recogida por la posibilidad de introducir elementos audiovisuales, menor coste, ofrece al entrevistado la posibilidad de reflexionar y no se producen sesgos por su influencia.

En el párrafo anterior vemos varios motivos de su acogida, pero desde el punto de vista de la ingeniería del software, la usabilidad y la accesibilidad podríamos añadir uno más: la facilidad de hacerlo ahora. En el pasado era bastante complejo realizar todos los pasos necesarios para preparar, enviar o realizar presencialmente los cuestionarios y recolectar los datos. En estos últimos años, desde la expansión de Internet, las herramientas creadas por desarrolladores a lo largo del mundo aportan mucho a su uso.

Podemos observar como compañías especializadas en realizas encuestas como SurveyMonkey tienen valoraciones superiores a los dos mil millones de dólares y se encuentran listas para salir a bolsa [18]. Con ello podemos ver la evolución del uso y de las plataformas que ofrecen estos servicios. En 2015 publicaron que contaban con más de 25 millones de usuarios, 10 millones más que en 2018 y recibían 90 millones de respuestas al mes [19].

Con estos datos podemos observar el crecimiento del uso de plataformas para realizar cuestionarios en línea, pero también la gran inversión en tecnología que han hecho este tipo de compañías.

En sus inicios hace 20 años sólo realizaban encuestas sencillas en línea, más adelante sumaron herramientas de análisis de datos, las adaptaron a la revolución del smartphone y después se asociaron con otras plataformas para la integración entre aplicaciones [19].

3.2 IOT y plataformas web

En el apartado anterior podríamos haber agregado mucha más información tanto científica/teórica como práctica o del mercado pues podemos encontrar diversidad de ejemplos de ambos casos, pero sobre el uso de cuestionarios sobre dispositivos físicos en diferentes entornos como es objetivo de este proyecto es bastante más complejo encontrar información.

En este caso vamos a comentarlo desde el punto de vista del IOT (Internet de las cosas). Los antecedentes en la evolución del IOT también son claros y podemos dar ejemplo de diversas tecnologías que mejoran continuamente y del paradigma que supone en si mismo esta evolución [20].

Para nuestro objetivo lo más relevante es ver la evolución de costes, por la accesibilidad que aporta el IOT, y la mejora de la conectividad.

La ventaja de aportar IOT a un cuestionario es que estos sistemas han reducido su coste y lo reducen en la operativa por la menor intervención humana en el proceso de recolección presencial de datos.

Por otro lado, también podemos encontrar diversidad y una evolución constante en la conectividad de estos dispositivos. La siguiente imagen [21] muestra la gran cantidad de sistemas de conectividad con los que podemos trabajar desde el IOT y que se van agregando año a año.

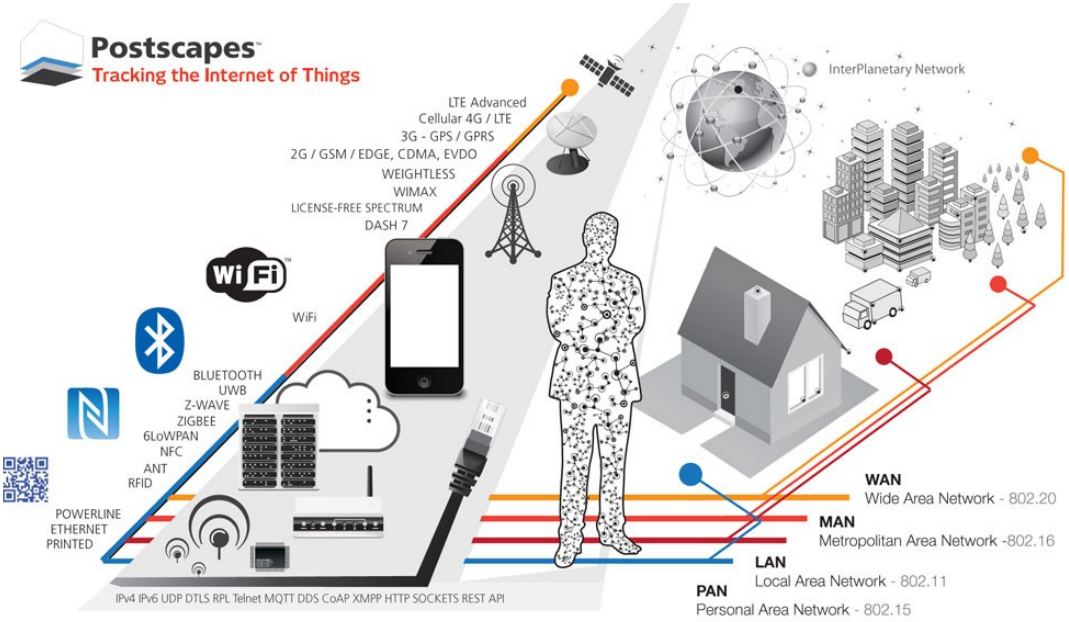

Ilustración 1: Sistemas de conectividad

También podríamos destacar la evolución de las plataformas web para la administración de los dispositivos y datos. En ese caso es posible extenderse en gran medida, pero sólo viendo la evolución de las tecnologías de desarrollo web y la evolución de los sistemas de computación en la nube o Infraestructure-as-a-Service (IaaS) como Azure, AWS o Google Cloud podemos apreciar cómo cambia constantemente todo lo referente a estas materias y la conclusión acaba siendo elegir lo que más se adapte a lo que necesitas hacer en ese momento o" trabaja con lo que conoces si cumple con los objetivos".

En los últimos años hemos observado que el IOT y la nube/cloud (sistemas IaaS, PaaS, etc) van de la mano y se facilitan mucho las cosas al trabajar en conjunto. Los grandes sistemas cloud ya se han adaptado al movimiento de pequeños datos masivos de la IOT y han creado APIs y librerías compatibles con los diferentes dispositivos IOT para su uso.

4. METODOLOGÍA

CAPÍTULO 4 METODOLOGÍA

Como en futuros apartados en este vamos a hacer una división entre la metodología que implica al desarrollo de la plataforma y al de los dispositivos.

En el desarrollo de este proyecto se han seguido dos líneas diferentes. Para la plataforma seguimos las fases del Proceso Unificado de Desarrollo de Software [16] y para los dispositivos propios el Modelo de Prototipos o desarrollo rápido de prototipos [17].

Por otro lado, con la expansión del IOT nos encontramos en un momento en el que el número de decisiones a tomar respecto a cada elemento, arquitectura o sistema es muy variado, esto genera dificultades a la hora de exponer algunas decisiones de diseño pues estudiar todo el entorno del IOT es muy amplio, incluso para nuestro acotado objetivo de envío de información simple desde paneles propios.

4.1 El desarrollo de la plataforma

A la hora de desarrollar una plataforma web, o aplicación web, siempre nos encontraremos con dos grandes problemas a resolver en el contexto de la toma de decisiones. El primero es el “cómo” y el segundo es el “con qué”.

En el ámbito del “cómo” encontramos diversas formas de gestionar y seguir un desarrollo. Hay modelos iterativos clásicos según estándares, metodologías ágiles, modelos de desarrollo rápido o metodologías de programación extremas (XP) por nombrar algunos. En nuestro caso ya hemos comentado que usaremos un modelo clásico principalmente porque parece ser el más concreto para explicar cada fase del desarrollo y poder mostrar en este documento como ha sido el proceso del mismo.

En el ámbito del “con qué” tenemos los mismo, una gran diversidad de herramientas, lenguajes, framework, servidores, servicios, etc. En este caso y gracias al conocimiento vertido de la ingeniería de software moderna sabemos que lo correcto es delegar la toma de decisiones sobre estos elementos a la fase de diseño del software, aunque no es nombrada en

algunas metodologías como tal, pero en todas ellas siempre se mira el conjunto a futuro para decidir sobre las herramientas software a utilizar.

A nivel de usuario, como veremos más adelante, la plataforma puede dividirse en grandes apartados, principalmente se compondrá de: acceso de usuarios, dashboard estadístico principal, visualización de datos con filtros, control de los dispositivos, control de los cuestionarios, API interna para recibir los datos desde los dispositivos y un área de administración sólo para administradores de los conjuntos de usuarios.

Esta visión general de lo que los requisitos recogerán para desarrollar la plataforma nos muestra que lo más relevante para la plataforma son las funcionalidades.

En la historia de la web hemos pasado de la web 1.0 o estática en la que el usuario sólo podía ver información a la web 2.0 en la que el usuario puede interactuar y existe un diseño centrado en él. Ahora se maneja el término de web 3.0 de forma difusa, la llaman web semántica y aún se está definiendo, pero aunque el futuro de la web conlleve tecnologías como la inteligencia artificial o de contenido semi-estructurado el camino es el mismo que desde el inicio de la web: facilitar el acceso a la información al usuario.

Nosotros facilitaremos el acceso a la información a través de gráficas sencillas y estructuras claras del sistema. Y que la solución pueda ser ampliamente usada buscando unos costes de desarrollo y consumos de recursos bajos.

4.2 El desarrollo de dispositivos

Con los dispositivos propios nos encontramos con algo similar a la plataforma. El “cómo” es menos diverso pues para el desarrollo de dispositivos IOT no hay tantos diferentes modelos de desarrollo o vías de gestión. Pero en cuanto al “con qué” nos encontramos con el mismo problema, la gran diversidad de elementos, en este caso de placas, componentes, carcasas y otros elementos que componen cualquier dispositivo IOT.

En este contexto volvemos a tener que tomar decisiones tecnológicas enfocadas a negocio ya que no encontramos elementos con los que la solución sea directa e incuestionable o científicamente más correcta como se da en la algorítmica, optimización u otras áreas.

Los objetivos de tener una interfaz física accesible de nuestro sistema para enviar respuestas a preguntas específicas ya se han comentado anteriormente y con sistemas costosos no se cumplen, por ello es importante enfocar nuestro desarrollo IOT en el ratio utilidad/costes.

Para conseguir dispositivos IOT de bajo coste es necesario tener muy en cuenta una serie de posibles requisitos como el uso de tecnologías móviles actuales para no depender de redes de uso cercano o medio. Si se enfoca mal el análisis llevara a un diseño del prototipo costoso, por tanto, es muy importante plantear qué grado de conectividad es necesario realmente (LAN, Wifi, GSM, 3G, 4G), qué nivel de interacción con el usuario es suficiente y qué factores obligan a correr código en el dispositivo en lugar de en el servidor web.

Todos los aspectos anteriores serán tomados en cuenta en siguientes apartados del documento pues un añadido no muy necesario descompensaría el coste del dispositivo. Esta descompensación es debida a que las necesidades de potencia de procesamiento o los sistemas de conectividad pueden multiplicar en varias veces el coste de un componente.

En muchas ocasiones un prototipo nace completa o parcialmente desde otro anterior, la mayoría de los microcontroladores y placas a día de hoy funcionan de forma similar y es posible reaprovechar el conocimiento fácilmente. Aunque es habitual obviar esto en el mundo empresarial en el ámbito científico es algo normal y por ello siempre existe ese tipo de apartados llamado antecedentes.

En nuestro caso no vamos a reaprovechar nada de otros prototipos, no hemos encontrado trabajos al respecto que se alineen con la sencillez y optimización de recursos que en este proyecto deseamos plasmar. Pero buscaremos la selección de tecnologías de tal manera que podamos variar

aspectos como la conectividad reaprovechando el prototipo base, tanto el máximo de su código como hardware.

5. IMPLEMENTACIÓN Y DESARROLLO

CAPÍTULO 5 IMPLEMENTACIÓN Y DESARROLLO

Este capítulo tomara la senda de las metodologías de desarrollo que hemos seleccionado para la implementación y el desarrollo de la plataforma y los dispositivos propios siguiendo el Proceso Unificado de Desarrollo de Software [16] y el Modelo de Prototipos [17].

También hemos incluido pequeños apartados que comentan aspectos cercanos o relevantes al proyecto como los dispositivos de proveedores qué, aunque no los desarrollaremos nosotros, pueden trabajar con nuestra plataforma como dispositivos físicos en el cliente y deben ser estudiados y analizados.

5.1 Planificación y gestión general

Es una buena práctica hacer una estimación de tiempos si es posible y tenemos el conocimiento o la experiencia para poder acertar al menos mínimamente. En nuestro caso gracias al haber partido en diferentes secciones el proyecto y poder ajustarnos a los tiempos que estimemos, por contar con la libertad de poder trabajar hasta el punto que creamos interesante, podemos asegurar que la siguiente estimación es totalmente fiable y acertada ya que casi más que una estimación es una declaración de intenciones.

En la siguiente tabla podemos ver tareas de redacción de elementos usados en el proceso de desarrollo como por ejemplo la redacción de listas de requisitos. En los diferentes apartados del trabajo y en varios de los puntos siguientes se han adaptado esas redacciones ampliado su información para que encajen con el formato de un trabajo fin de grado y sea más fácilmente entendible para los no expertos en este tipo de fases de desarrollo.

La tabla contiene la estimación y las tareas necesarias para el estudio, análisis e implementación de la plataforma, los prototipos y los agregados necesarios para completar el trabajo fin de grado con éxito. La planificación como tal se guía por ir completando las tareas desde la primera que vemos en la tabla a la última durante los primeros ocho meses de 2018.

Tabla 1: Planificación y estimación de horas

Fase	**Tarea**	**Duración**	**Resp.**
PLATAFORMA		**horas**	
Definición			
	Redacción de: La Definición del alcance y objetivos de la plataforma.	4	JP
Análisis y Requisitos			
	Redacción de: La Lista de requisitos de la plataforma.	6	JP
	Gestión general del desarrollo. (Planificación, Entorno de desarrollo, Gestión de requisitos).	12	JP
Diseño			
	Redacción de: Tecnologías de la plataforma.	6	JP
	Redacción de: Diseño y arquitectura general de la plataforma.	10	AS
Desarrollo			
	Configuración del entorno de desarrollo.	6	AS
	Documentación de: Entorno de desarrollo y su configuración.	4	AS
	Desarrollo requisitos de administradores.	24	P
	Desarrollo requisitos de usuarios logueados y externos.	60	P
	Implementación de las mejoras para cumplir con los requisitos no funcionales.	16	P

Lanzamiento / Publicación			
	Configuración del entorno de producción.	8	AS
	Documentación de: Entorno de producción y su configuración.	4	AS
	Subidas al entorno de producción.	4	AS
	Prueba del entorno.	10	AS
	Documentación de: Subidas e integración en producción y sus pruebas.	4	AS
	Documentación de: Manual de usuario.	6	P
Mantenimiento			
	Redacción de: La política de mantenimiento.	4	JP
DISPOSITIVO			
Análisis Requisitos			
	Redacción de: Requisitos mínimos de los dispositivos.	4	JP
	Redacción de: Dispositivos de proveedores y Dispositivos propios.	4	JP
	Gestión general del desarrollo (Planificación, Entorno de desarrollo, Requisitos, Materiales).	6	JP
Diseño rápido			
	Redacción de: Implementación del prototipo según la de requisitos, el prototipo y circuito.	14	AS
Desarrollo			

	Configuración del entorno de desarrollo y librerías.	6	AS
	Desarrollo de prototipo Wifi.	15	P
Fabricación y prueba			
	Fabricación del prototipo.	9	JP
	Pruebas y resultado.	6	P
TFG			
	Tareas de gestión y entregas.	No cont	TFG
	Redacción general del proyecto (sumando a los apartados anteriormente redactados).	55	TFG
	Preparación de la presentación.	20	TFG

Como se puede observar en la anterior tabla tenemos las horas de dedicación estimadas divididas por responsable o perfil. Para poder ver que rol hay que tomar para realizar las diferentes tareas los puestos o perfiles han sido definidos de la siguiente forma:

Tabla 2: Puestos y perfiles de un equipo de desarrollo de software

	Puesto	**Tareas**
JP	Jefe de proyecto	Project Management, Presupuestos, Gestiones ,Environment.
AS	Arquitecto de Software	Configuration & Change Management,Requisitos, Análisis y diseño alto nivel. Testing. Despliegue con proveedores.
P	Programador	Apoyo al análisis y diseño, Implementación, Testing, Despliegue.
TFG		*Horas añadidas para seguir el procedimiento, formato y apartados necesarios del trabajo.

Para seguir la línea de realismo con respecto al mundo del desarrollo de software y dispositivos nos es interesante mostrar un presupuesto aproximado con precios estimados del desarrollo de software/hardware en España en 2018 por horas (consultado con especialistas del sector) para este tipo de proyectos. Con ello obtendríamos un presupuesto similar al siguiente:

Tabla 3: Presupuesto aproximado por perfil/hora y total

Puesto	**Precio hora**	**Horas totales**	**Coste total**
Jefe de proyecto	65,00 €	55	3.575,00€
Arquitecto de Software	50,00 €	70	3.500,00€
Programador	38,00 €	125	4.750,00€
TFG		75	
	Totales	325	**11.825,00€**

Se han incluido las horas propias del ajuste del proyecto al formato y estilo de un trabajo fin de grado, la redacción de sus apartados específicos, etc para separarlo de las horas reales de desarrollo y para facilitar así la visión de la proporción de horas que supone el desarrollo de un proyecto como el nuestro.

5.1.1 Plataforma

La plataforma busca cumplir con los objetivos específicos que definen la solución software que trabajará junto con la solución hardware, es decir, con los dispositivos para poder visualizar datos, configurar los dispositivos, etc.

Se estudiará y desarrollará la plataforma siguiendo el Proceso Unificado de Desarrollo de Software [16] como veremos en el apartado de plataforma web. Se ha elegido de base el uso de una plataforma web y no por ejemplo un sistema local o en red privada porque la propia definición del proyecto y el uso de prototipos IOT (Internet de las Cosas) nos mueve hacía ello y los sistemas locales no nos aportan ninguna ventaja en nuestro caso de estudio.

5.1.2 Dispositivos

La solución hardware, los dispositivos, se estudiará y desarrollará siguiendo el Modelo de Prototipos [17] buscando un desarrollo rápido de los mismos.

Los dispositivos se dividen en: dispositivo de proveedores como tablets en las que es posible poner visualmente las opciones y que los usuarios las pulsen en la pantalla táctil, o nuestros dispositivos propios que dispondrán de botones mecánicos que pueden ser pulsados y transmitirán las opciones escogidas como veremos en los puntos referentes a nuestros prototipos.

Se sale del modelo de desarrollo el estudio de sus diferencias, ventajas e inconvenientes. Pero es necesario para este proyecto el diferenciar sobre qué interfaz física pueden interactuar nuestros usuarios.

5. 2 Plataforma web

Como podemos ver en la planificación, estimamos la plataforma web y hemos asignado a ella un número mayor de horas que a los dispositivos. La plataforma web necesita una lista de funcionalidades mayor para ser realmente útil y al contar con muchos diferentes tipos de herramientas y tecnologías a nuestra disposición entre las que escoger y ser el propio enfoque de plataforma web tan amplio se entiende justificado este mayor número de horas.

5.2.1 La plataforma web

Como se recoge en los objetivos la plataforma web debe partir de un diseño software completo para la obtención de opinión de usuarios desde los tres puntos de vista principales: a*dministrador, gestor o externo,* tanto a través de los dispositivos de proveedores como propios.

Comenzamos con la lista de requisitos que nace del análisis producido durante las fases anteriores de introducción al problema, alcance, objetivos específicos, etc.

5.2.2 Lista de requisitos de la plataforma

En las siguientes tablas se muestran las listas completas de requerimientos que necesita cumplir la plataforma para poder ser usada, cumplir sus objetivos y combinarse con los dispositivos que veremos en futuros apartados.

Como conocimiento de negocio o del problema debemos saber antes de definir los requisitos que un cuestionario se compone de al menos una pregunta o página de cuestionario, en ocasiones incluyen una página de bienvenida y una de despedida y/o agradecimiento. También para nuestro caso específico el contexto del problema se basa en la muestra de estos cuestionarios a través de dispositivos físicos.

Los requisitos se van a clasificar en requisitos funcionales y requisitos no funcionales como es habitual en este tipo de desarrollos:

Tabla 4: Tabla de requisitos funcionales de la plataforma web

Requisito	**Título**	**Descripción**
RF-01	Registro y autenticación	El sistema permitirá registrar usuarios y permitirles entrar bajo autorización/logueo.
RF-02	Dispositivos, cuestionarios, páginas de cuestionarios y de agradecimiento	El sistema permitirá gestionar (crear, editar y borrar) dispositivos, cuestionarios, páginas de cuestionario y agradecimiento y su relación a usuarios logueados.
RF-03	Usuarios asociados a cliente	El sistema permitirá gestionar (crear, editar y borrar) clientes y asociar cada usuario a un cliente a los usuarios administradores.
RF-04	Test de cuestionarios	El sistema permitirá probar los cuestionarios en modo test, sin registrar respuestas a usuarios logueados.
RF-05	Visión externa de cuestionarios	El sistema permitirá a usuarios externos/sin logueo ver y responder a los cuestionarios.

		Y registrar la respuesta.
RF-06	Visión de datos gráficos	El sistema permitirá a los usuarios logueados ver gráficas de sus datos y los resultados se podrán filtrar por dispositivo, cuestionario y página de cuestionario.
RF-07	Opciones visuales predefinidas	El sistema permitirá gestionar (crear, editar y borrar) opciones visuales a los usuarios administradores, a los usuarios no administradores sólo permitirá utilizarlas en sus cuestionarios.
RF-08	Los elementos enlazados al cliente	El sistema enlazará los dispositivos, cuestionarios, las páginas de cuestionario y de agradecimiento al cliente del usuario. Un usuario sin cliente asociado no debe crear ningún elemento.
RF-09	Recepción de respuestas sin interface visual ni logueo	El sistema permitirá a usuarios externos/sin logueo responder a los cuestionarios desde dispositivos físicos sin interface.

Tabla 5: Tabla de requisitos no funcionales de la plataforma web

Requisito	**Título**	**Descripción**
RNF-01	Seguridad web	El sistema dispondrá del sistema necesario para verificación segura de usuarios, y diferenciar los usuarios administradores, no administradores y externos.
RNF-02	Facilidad de uso	El sistema debe facilitar su uso mediante interfaces sencillas y lógicas de uso simples.
RNF-03	Eficiencia. Tiempo	El sistema debe ser suficientemente rápido

	de respuesta	para que el usuario no deba esperar en la visualización de datos más de 5 segundos ni en la posibilidad de responder a los cuestionarios más de 3 segundos.
RNF-04	Mantenibilidad y crecimiento.	El sistema debe contribuir a su fácil mantenimiento, ser fácilmente evolucionable y modificable.

Este último requisito no funcional 04 es el que anotamos como más relevante excluyendo el de seguridad. Entendemos que dar importancia al mantenimiento futuro y crecimiento de un sistema como el que queremos implementar es algo muy relevante para la utilidad futura del mismo.

Defendemos esta tesis por dos razones, la primera porque en la actualidad la gran velocidad del cambio en interfaces y sistemas web hace que los parámetros como la usabilidad o lo que es un buen tiempo de respuesta sean subjetivos y variables, la segunda es porque al trabajar con dispositivos físicos y prototipos nos encontramos ante una gran incertidumbre por sus necesidades especiales actuales o futuras.

El requisito funcional 09 la recepción de respuestas sin interface visual ni logueo, es el que nos lleva a poder responder a las cuestiones desde los prototipos físicos que desarrollaremos más adelante.

5.2.3 Tecnologías de la plataforma

A continuación del análisis en nuestro modelo de desarrollo nos encontramos con la fase de diseño, es la hora de preocuparnos por cómo hacer lo definido anteriormente.

El problema que nos encontramos para empezar a definir el diseño del sistema es que las diferentes tecnologías que tenemos a nuestra disposición nos crearán diferentes variaciones de este diseño, vemos como mejor idea primero definir el stack tecnológico o grupo de tecnologías relacionadas que

usaremos desde los requisitos antes de diseñar de forma abstracta el sistema y luego tener que reconvertirlo a nuestro estilo final de desarrollo.

Justificamos esta priorización sólo en casos como este, las plataformas web, donde nos encontramos con una cantidad inmensa de opciones, diseños, patrones, tecnologías como tal, estilos de desarrollo, etc.

Esta preselección tecnológica se basa en criterios como los siguientes:

- Uso de tecnologías suficientemente modernas pero maduras, en crecimiento y documentadas: Si queremos que nuestro sistema sea fácilmente mantenible y pueda crecer no podemos utilizar tecnologías desfasadas y en declive, necesitamos elegir tecnologías actuales en crecimiento para asegurarnos que su uso futuro sea viable en términos de capital humano y en termino general de evolución tecnológica.
- Uso de tecnologías libres o económicas: por el propio objetivo del sistema se debe priorizar el uso de tecnologías libres pues siempre suponen una solución más económica que las herramientas privativas para el desarrollo. Exceptuando los casos en las que por motivos de futuro consumo de recursos máquina o coste del desarrollo sea útil elegir un software que no sea libre.
- Evitar las dependencias y el Vendor lock-in: evitar en todo lo posible el bloqueo por proveedor, es decir, que podamos migrar nuestra aplicación web sin grandes costes adicionales.

Hasta hace tres o cuatro años la solución más adecuada para nuevos desarrollos de este estilo sería escoger el stack clásico de tecnologías open source LAMP (Linux, Apache, Mysql, PHP) que no vamos a utilizar porque creemos que en la actualidad, en 2018, podemos encontrar mejores soluciones.

Con sus ventajas e inconvenientes estos últimos años se popularizó el uso del stack MEAN (MongoDB, ExpressJS, Angular, NodeJS). La anterior lista de tecnologías LAMP tenía la ventaja de ser ya madura, pero limitada y costosa de implementar por su baja automatización de tareas de desarrollo. Con MEAN nos encontramos una gran cantidad de ventajas, más óptima en

cuanto a recursos, soporta gran cantidad de datos y enfocado a REST (arquitectura de software base de la transferencia de datos representacional). Pero realmente no nos aporta ninguna ventaja demasiado relevante en este desarrollo, aunque sí cumple con todas las necesidades que podemos tener, pero los tiempos de desarrollo por la variedad de nuestras necesidades es posible que se dispare sin el uso de gran cantidad de frameworks y rígidas arquitecturas.

En la actualidad nos encontramos con un nuevo stack, aún sin nombre específico, que cumple con todo lo que hemos comentado anteriormente. No sólo es sumamente eficiente desde el punto de vista del coste de desarrollo al tener una integración total de todos los componentes, sino que es open souce con licencia MIT (uso comercial, modificación distribución, uso privado), suficientemente maduro por su bagaje anterior y muy bien documentado gracias al apoyo de una gran corporación y evita de base el Vendor lock-in por el propio compromiso de la compañía. Este stack parte de ASP.NET Core.

ASP.NET Core nace del acercamiento de Microsoft al mundo Open Source. Microsoft crea en 2016 este nuevo framework que utiliza C# (desde hace tiempo open source hasta su compilador), Entity Framework Core (ORM también open source), MVC Core (Subframework open source) y Razor Core (Sintaxis) entre otros.

Microsoft desde el inicio abre este sistema a los desarrolladores y plantea de frente su mayor connotación negativa, el uso privativo. Este framework puede trabajar con una gran integración directa por sus librerías con bases de datos como MySQL, Postgres, SQL Server o MariaDB entre otras.

El Vendor lock-in sería el siguiente problema que solucionaron para la comunidad de desarrolladores. Todo este conjunto o stack de tecnologías y frameworks pueden ejecutarse con la instalación de los RunTime específicos para cada sistema operativo sobre diferentes versiones Linux, como Ubuntu, CentOS o Debian entre otros, incluso sobre macOS Sierra en adelante, y tambien Windows 7 y Server 2008 en adelante.

Por otro lado, también podemos usar contenedores Docker, preparar todo el sistema dentro de uno de estos paquetes y movernos libremente entre proveedores sin perjuicio alguno, esta solución puede gestionarse desde su propio IDE haciendo muy fácil la preparación de los contenedores.

Con todo esto sumado a las versiones también libres de su propio IDE de desarrollo Visual Studio, que ahora se puede ejecutar en Windows, Linux y Mac de forma gratuita, nos encontramos con una nueva vía de desarrollo que cumple con todo lo que necesitamos, poco explorada en Europa pero que cada día tiene más adeptos en Estados Unidos.

Nos hemos decantado por utilizar este stack o conjunto tecnologías de desarrollo web porque cumple con todos nuestros requisitos, cuenta con una integración total de los diferentes frameworks que facilitara mucho el desarrollo desde cero de una plataforma, y la calidad del IDE. El propio lenguaje C#, o Razor Core, como sustituto de gran cantidad de código JavaScript, nos evita riesgos y futuros problemas al mismo tiempo que aumenta nuestra eficiencia en el coste del desarrollo.

Por otro lado, no menos relevante tenemos el entorno de pre-producción, aspecto que el propio IDE o los RunTime nos soluciona en local de forma prácticamente automática y el entorno de producción queda abierto para escoger la mejor solución sea bajo servidores propios de cualquier tipo, o bajo Amazon Web Services, Google Cloud, Azure u otro similar.

5.2.4 Diseño y arquitectura general de la plataforma

Actualmente la arquitectura más utilizada para este tipo de soluciones, para dar cobertura a una plataforma y una entrada de datos externa es MVC (Modelo-Vista-Controlador) y las API REST.

MVC es un patrón de arquitectura que separa la lógica de datos y de negocio. A la hora de diseñar la arquitectura de un nuevo sistema es de gran ayuda fijarse en los diferentes patrones existentes como MVC, MVVM (Modelo–Vista–Modelo de Vista) o cliente/servidor en dos capas entre otras.

Sabiendo sus ventajas e inconvenientes es también una buena idea pensar quÉ herramientas o técnicas necesitamos o nos aportan alguna ventaja en nuestra aplicación para con toda esa información elegir un patrón, aunque luego se tenga que adaptar en pequeña medida a nuestras necesidades.

En nuestro caso creemos que un ORM (Mapeo Objeto-Relacional) nos aporta una gran eficiencia en el desarrollo. Por la naturaleza del problema la lógica de datos cental no debería variar en el tiempo, sí otros muchos aspectos ya sean visuales de la plataforma, de eficiencia o de conectividad. Esto lo definimos porque los dispositivos, cuestionarios, páginas de cuestionario o agradecimiento, clientes, usuarios no tienen muchas opciones para cambiar pues sus datos básicos son concretos del contexto del problema.

Por el uso de un ORM decidimos basarnos en el patrón MVC, patrón de 3 capas que uniendo su capa de modelo de datos con la capa artificial de datos generada por el ORM nos permite un uso muy eficaz de la arquitectura definida. En la siguiente imagen podemos ver un ejemplo de las capas:

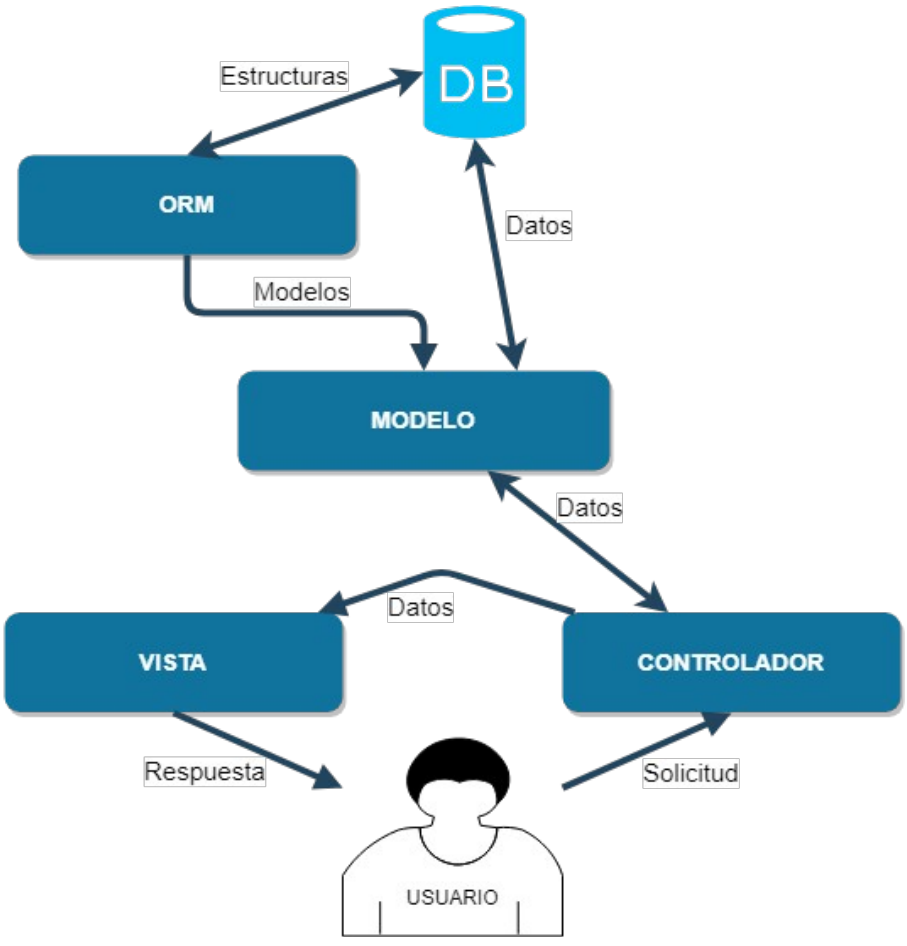

Ilustración 2: Arquitectura aplicación web MVC + ORM

Continuando con la fase de diseño vamos a definir el modelo de datos. Gracias al uso de un ORM nos encontramos con una lista de objetos generados desde clases que son a su vez los objetos de la capa Modelo de la arquitectura y las tablas que tenemos en la base de datos.

Modelo de datos

En los siguientes diagramas podemos ver todos los objetos principales de esta capa y base de datos tal como serán creados. Comenzamos con la relación entre los usuarios y los clientes.

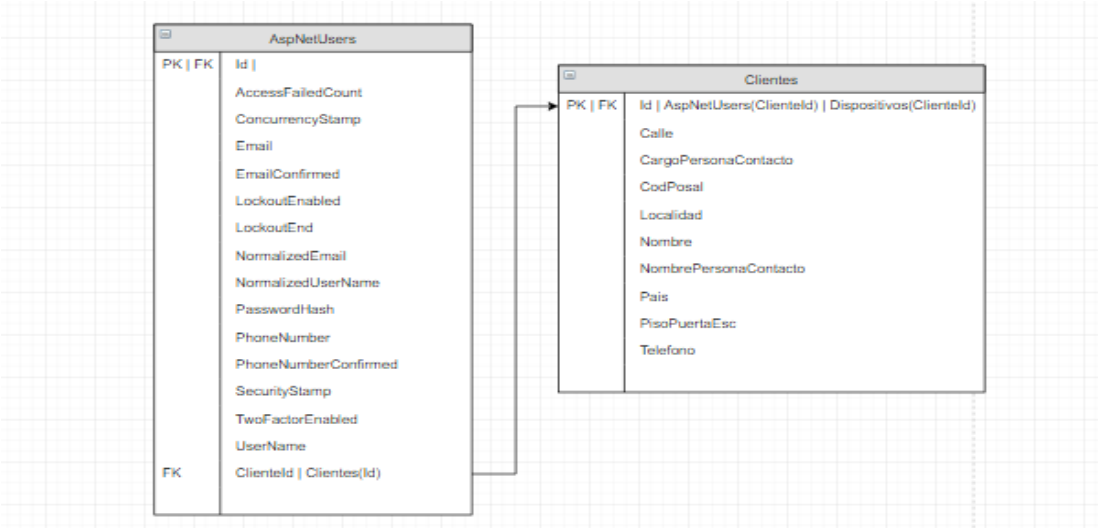

Ilustración 3: Modelo base de datos Usuario-Cliente

Algunas tablas auxiliares poco relevantes en la siguiente imagen.

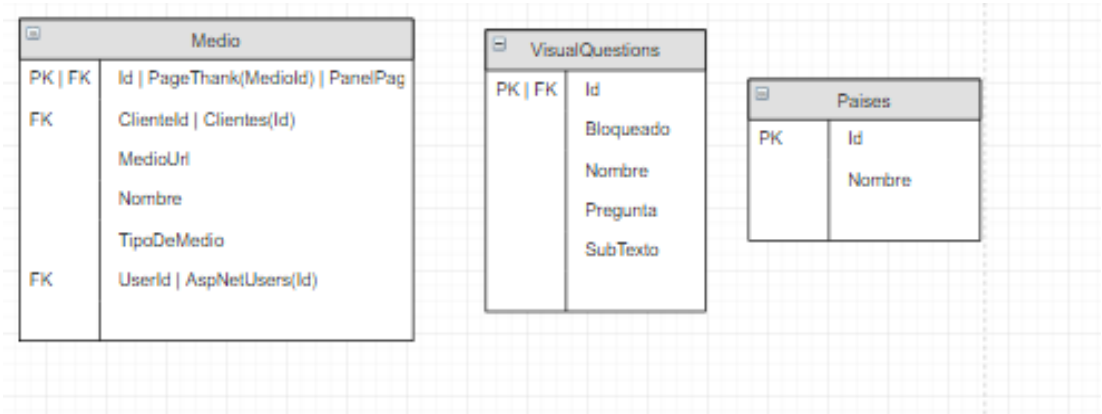

Ilustración 4: Modelo base de datos Auxiliares

La próxima imagen nos muestra las tablas y el enlace entre ellas de los dispositivos, paneles (nombrados a nivel de usuario como cuestionarios) y pageThank (páginas de agradecimiento).

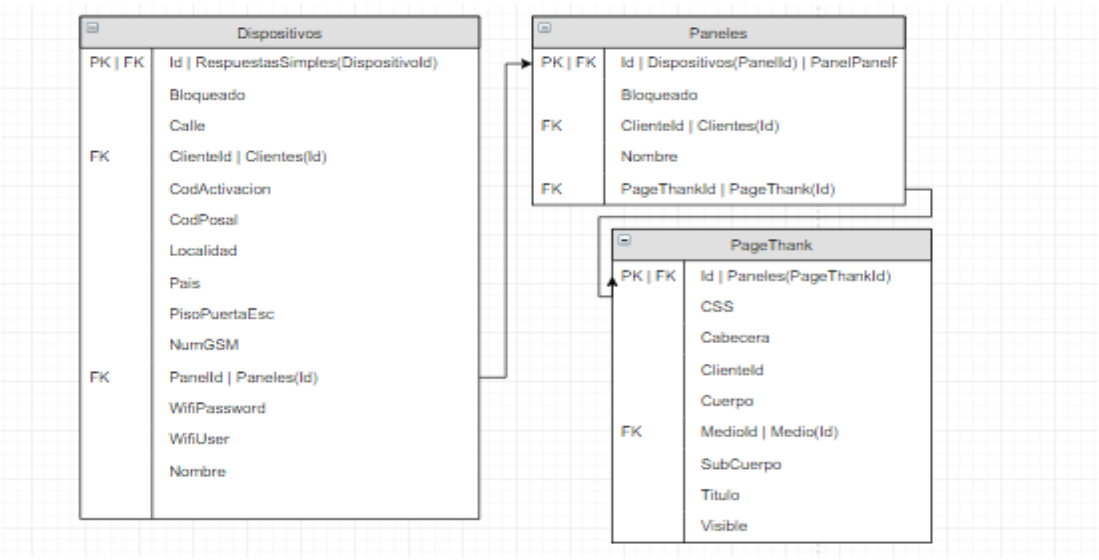

Ilustración 5: Modelo base de datos Dispositivos-Paneles-PageThank

Seguimos con la siguiente relación entre paneles y panelPages (cuestionarios y sus páginas), y estás panelPages con las opciones. Todo ello relacionado con las tablas intermedias de la izquierda de la imagen.

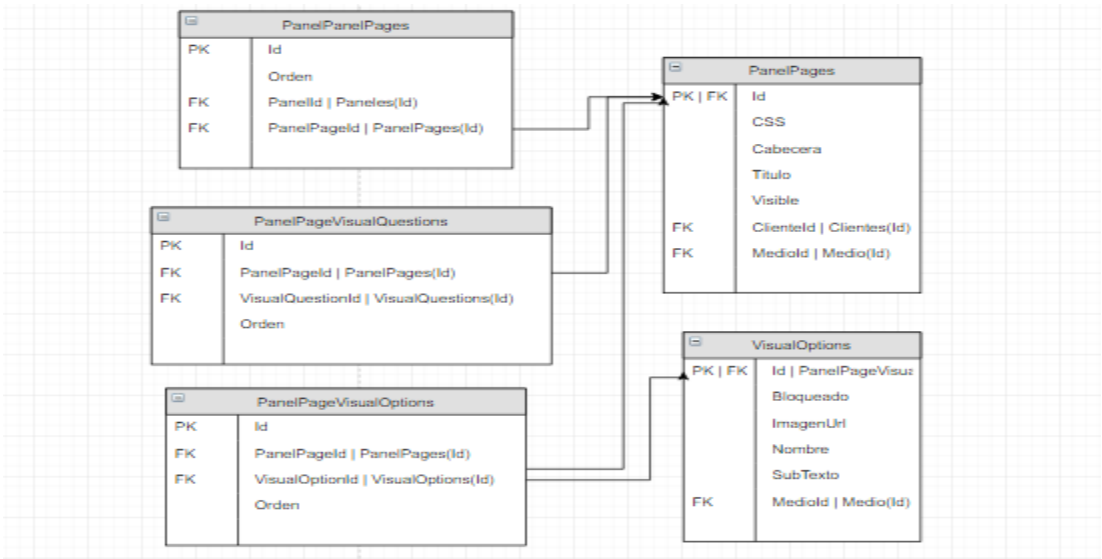

Ilustración 6: Modelo base de datos Paneles-PanelPage/PanelPage-Opciones

Y por último tenemos la tabla que guardará las respuestas a los cuestionarios, no hemos podido poner todos los enlaces con las demás por motivos de tabla, pero por el nombre de los ID queda claro que está relacionada con todo lo posible para poder filtrar estas respuestas en un futuro de diferentes maneras, sea por el dispositivo desde la que se creó, el cuestionario que responde, la página específica de cuestionario, etc.

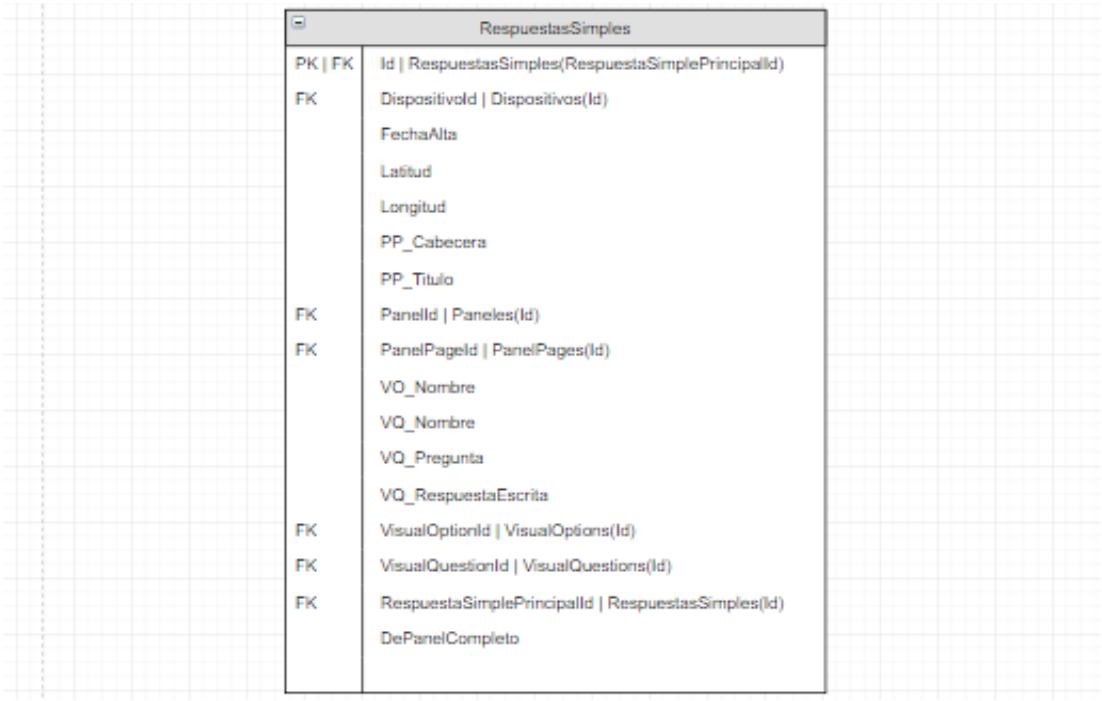

Ilustración 7: Modelo base de datos RespuestaSimple

El modelo de datos queda completo, seguimos con el diseño de los controladores y vistas. Este diseño parte del uso de una serie de frameworks o componentes que lo enfocan de una manera determinada y aquí es donde encontramos un gran salto de productividad en el desarrollo al contar con las herramientas:

- Entity Framework Core Scaffolding: que nos va a permitir generar desde el modelo los controladores de forma automática con su CRUD (Creación, Edición, Actualización, Borrado) para tener una base desde la que partir en su desarrollo y a su vez la creación automática de

vistas, también de forma básica, pero nos ahorrará el tener que ir creando los diferentes archivos, enlaces con las vistas, etc.

- LINQ: nos permite trabajar con el contexto del modelo desde los controladores. Es un lenguaje que simula al SQL pero en código C# y con el que podemos obtener datos del modelo filtrados y en diferentes formatos directamente sin necesitar hacer consultas de forma clásica. Esto nos ayuda a potenciar la ventaja de usar un ORM y de la autogeneración ya que las diferentes listas de datos que se necesitan salen directamente usando este interfaz para el modelo hasta la base de datos.

- Razor Core: esta sintaxis especial es un generador de páginas dinámico. Su uso está contenido en la capa de Vista y es el motor que nos ayudara a generar las vistas para el usuario, es capaz de ejecutar código C# a nivel de vista y generarlas a través de muchos diferentes componentes y elementos.

En este contexto el diseño es claro, los controladores están directamente ligados al modelo de forma lineal, es decir, un controlador por modelo, sin falta de ninguno de ellos, al ser autogenerados. Las vistas están ligadas a los modelos por la misma razón, y en cada controlador gestionaremos las vistas ligadas a ese modelo. En nuestro caso tendremos vistas desconectadas del modelo como la página de inicio o parciales como los menús, pero al poder seguir un estilo de diseño como el anterior ahorraremos una gran cantidad de recurso tiempo de desarrollo.

5.2.5 Notas sobre el desarrollo

Como anteriormente hemos decidido usar el stack .NET Core hemos escogido usar como base de datos SQL Server Express, que es gratuita por su fácil integración en este sistema, no es open source pero por motivos prácticos es para nosotros más rápido el desarrollo sobre esta base de datos por la preinstalación con el IDE.

El desarrollo se ha llevado siguiendo lo planteado en los requisitos con su traslación a casos de uso antes de la implementación, a continuación ponemos dos de ellos como ejemplo:

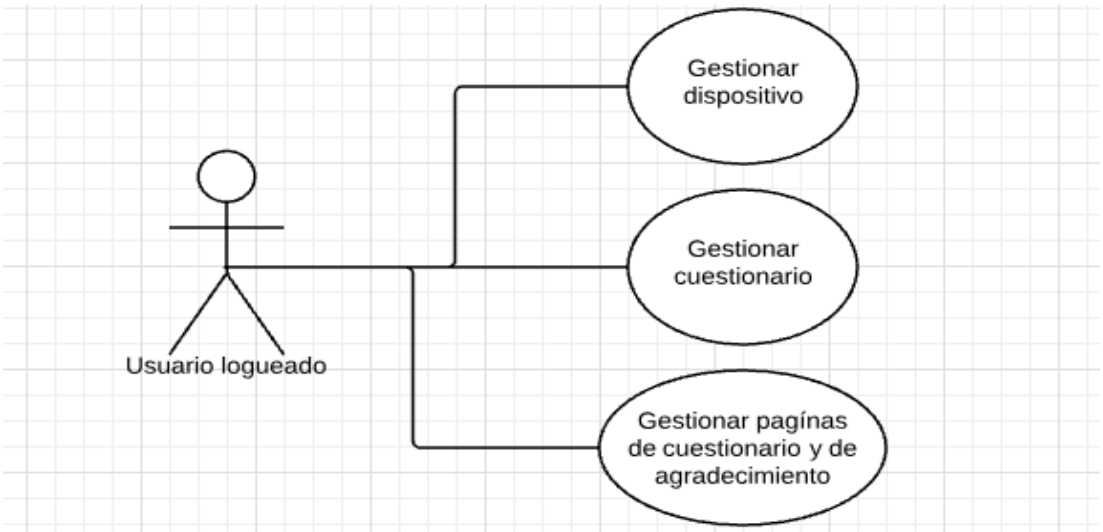

Ilustración 8: Caso de uso usuario logueado resumen

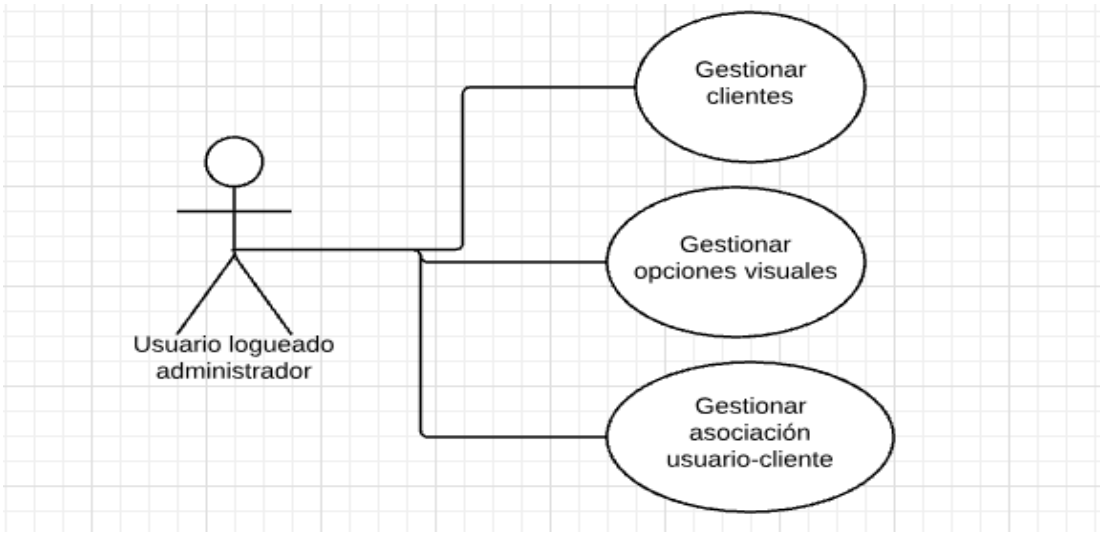

Ilustración 9: Caso de uso usuario logueado administrador adicional resumen

El anterior nos muestra el caso de uso resumido para un usuario logueado y el siguiente lo que además de lo anterior tiene un usuario logueado administrador.

Comenzamos con el planteamiento de la arquitectura observando las herramientas que nos aporta nuestro camino de desarrollo.

Utilizaremos el framework incluido en ASP.NET core para la autenticación llamado Identity y después agregaremos el modelo de datos comentado anteriormente, generaremos automáticamente todos los componentes posible como los controladores o vistas base para cerrar el desarrollo rematando los aspectos no automatizables como las gráficas, las opciones especiales de algunas vistas de creación o edición como la de los cuestionarios o las páginas de cuestionario que deben seleccionar listas de otros elementos, etc.

Con todo ello desarrollaremos una plataforma completa y funcional, en siguientes apartados como el Manual de Usuario veremos un recorrido por ella.

A continuación, mostramos la estructura interna del proyecto con todos los componentes:

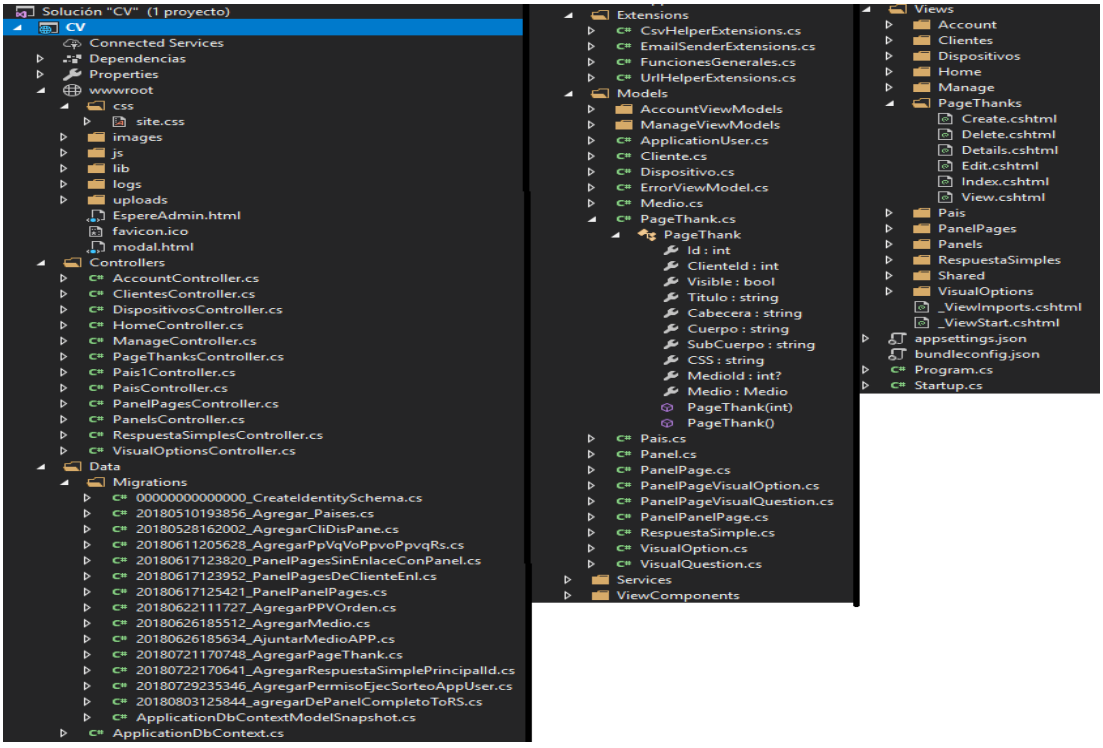

Ilustración 10: Componentes desarrollo plataforma web

Podemos observar como carpetas las capas más relevantes de la arquitectura del sistema como Data para contener lo referente al ORM,

Controllers para los controladores Models para los modelos o View para las vistas.

Hay agregados en la carpeta wwwroot para añadir complementos como Boostrap o css propios, entre otros.

También tenemos una carpeta llamada Extensiones con clases que contienen funciones genéricas para su uso distribuido por la plataforma.

Las siguientes imágenes muestran como son en C# y ASP.NET Core los principales elementos: modelo, controlador y vista:

PageThank.cs*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace CV.Models
{
    public class PageThank
    {
        public int Id { get; set; }
        public int ClienteId { get; set; }
        [DefaultValue(1)]
        public bool Visible { get; set; }
        [StringLength(255)]
        [Display(Name = "Título de la página")]
        [Required(ErrorMessage = "Debe escribir un título para la página")]
        [DefaultValue("")]
        public string Titulo { get; set; }
        [StringLength(511)]
        [Display(Name = "Texto principal")]
        public string Cabecera { get; set; }
        [StringLength(1023)]
        [Display(Name = "Texto secundario")]
        public string Cuerpo { get; set; }
        [StringLength(1023)]
        [Display(Name = "Texto inferior pequeño")]
        public string SubCuerpo { get; set; }
        [StringLength(8191)]
        public string CSS { get; set; }
        public int? MedioId { get; set; }
        public Medio Medio { get; set; }

        public PageThank(int _ClienteId)
        {
            ClienteId = _ClienteId;
            Visible = true;
        }

        public PageThank()
        {
            Visible = true;
        }
    }
```

PageThanksController.cs

```
using ...

namespace CV.Controllers
{
    public class PageThanksController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly ILogger _logger;
        private readonly UserManager<ApplicationUser> _userManager;

        public PageThanksController(ApplicationDbContext context, ILogger<ApplicationDbContext> logger, UserManager<ApplicationUser> userManager)...

        // GET: PageThanks
        [Authorize]
        public async Task<IActionResult> Index()...

        // GET: PageThanks/Details/5
        [Authorize]
        public async Task<IActionResult> Details(int? id)...

        // GET: PageThanks/Create
        [Authorize]
        public IActionResult Create()...

        // POST: PageThanks/Create ...
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize]
        public async Task<IActionResult> Create([Bind("Id,ClienteId,Visible,Titulo,Cabecera,Cuerpo,SubCuerpo,CSS,MedioId")] PageThank pageThank)...

        // GET: PageThanks/Edit/5
        [Authorize]
        public async Task<IActionResult> Edit(int? id)...

        // POST: PageThanks/Edit/5 ...
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize]
        public async Task<IActionResult> Edit(int id, [Bind("Id,ClienteId,Visible,Titulo,Cabecera,Cuerpo,SubCuerpo,CSS,MedioId")] PageThank pageThank)...

        // GET: PageThanks/Delete/5
        [Authorize]
        public async Task<IActionResult> Delete(int? id)...

        // POST: PageThanks/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        [Authorize]
        public async Task<IActionResult> DeleteConfirmed(int id)...
```

Ilustración 12: Controlador en código

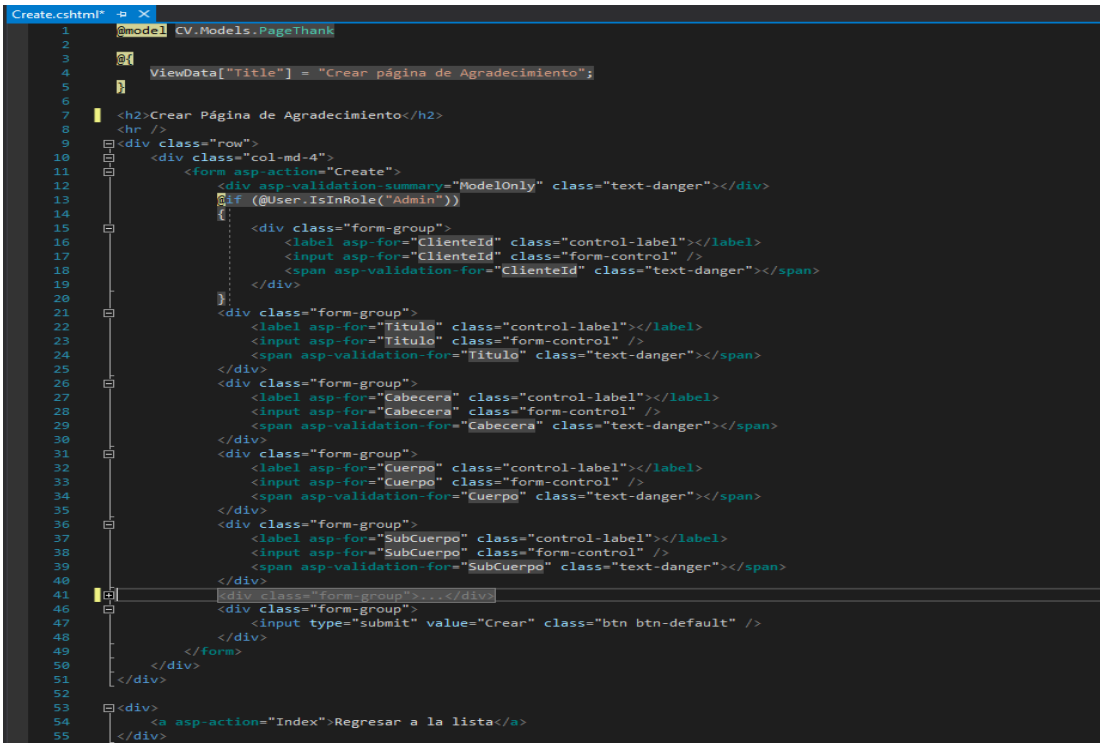

Ilustración 13: Vista en código

Un punto relevante sobre el proyecto es la API que vamos a crear para que los dispositivos físicos las puedan usar sin necesidad de interfaces gráficas. Se ha integrado en el controlador de RespuestaSimple que viene del modelo que se encarga de guardar las respuestas a los cuestionarios.

El siguiente código nos muestra la API, sigue REST y es un get con entrada anónima, es decir, sin necesidad de la autenticación de usuario de la plataforma. Aunque el dispositivo que la utilice debe informar de su código de activación y una clave para ver si es admisible.

```
// GET: /ApiR?r=1&codAct=ADF455GH66&clave=223123FG1
[AllowAnonymous]
[HttpGet("ApiR")]
public async Task<IActionResult> ApiR(int r,string codAct="",string clave ="")
        {
            if (verificarClave(clave))
            {
                Dispositivo dispositivo = await _context.Dispositivos
                    .FirstOrDefaultAsync(m => m.CodActivacion == codAct);
```

```
            if (dispositivo != null)
            {
                if (await guardarResId(_context, r, dispositivo.Id, 0))
                {
                    return Ok();
                }
            }
        }
        return NotFound();
    }
```

En el anexo Manual del programador se ha incluido mucha más información al respecto de los factores más relevantes de la programación de la plataforma.

5.2.6 Entorno de desarrollo y su configuración

El entorno de desarrollo por motivos prácticos lo vamos a montar sobre Windows 10, como hemos comentado anteriormente es posible también hacerlo sobre Linux o Mac.

Podríamos instalar un servidor de bases de datos SQL Server Express, un servidor web IIS Express y Visual Studio 2017 Community o Code, crear la base de datos y compilar la solución en una carpeta de aplicación del servidor web. Pero el propio Visual Studio puede preparar el entorno, desde la versión 2013 cuenta con todo lo necesario para ejecutar aplicaciones web y bases de datos SQL Server locales sin necesidad de gestionar toda la instalación de un entorno de desarrollo clásico.

Todo esto nos libera de tener que invertir recursos máquina y tiempo para preparar nuestro entorno de desarrollo local. Si necesitáramos un entorno en red sí deberíamos instalar servidores de bases de datos y web, como es habitual.

Es importante configurar como Development la variable de entorno ASPNETCORE_ENVIROMENT, para poder ver los errores. Esta variable

cambia en el entorno de producción ella misma para no mostrar errores y mantener la seguridad.

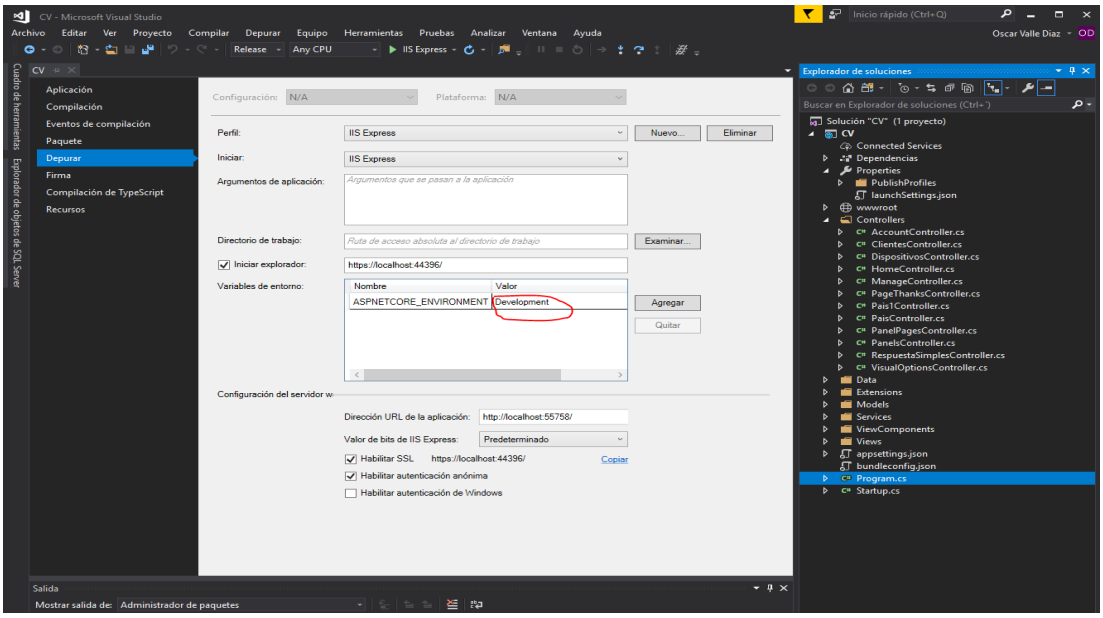

Ilustración 14: Preparación del entorno de desarrollo en el IDE

Con esto en cuenta ya sólo necesitamos lanzar la aplicación en modo debug una vez y el entorno será configurado. La base de datos se puede crear directamente en local, pero también se puede usar una remota, es decir, es posible probar con los datos de producción en este entorno de desarrollo fácilmente.

5.2.7 Entorno de producción y su configuración

Para escoger el entorno de producción primero tenemos que escoger si lo hacemos sobre máquinas propias o sobre algún proveedor, en nuestro caso usaremos un proveedor de servicios.

El segundo aspecto sería la infraestructura necesaria, de forma general disponemos de dos caminos, utilizar un servidor limpio e instalar el servidor web, servidor de bases de datos, etc o utilizar los servicios integrados cloud de los proveedores como AWS, Google Cloud o Azure.

Las ventajas que nos da el usar servicios cloud como Azure App Service son inmensas, no sólo es gratuito para un uso limitado de recursos que no superaremos, sino que se integra perfectamente con nuestro IDE. Ya comentamos que podemos crear contenedores Docker e integrarlo en cualquiera de los otros grandes proveedores, pero en nuestro caso no aporta demasiado hacerlo de esta forma ya que Azure por ahora nos aporta suficiente para tener un entorno de producción válido.

Para configurar en entorno de desarrollo sobre Azure, debemos crear una cuenta en el servicio y crear en el un hospedaje Azure App Service dándole un nombre, con ello en la web portal de Azure nos aparecerán los datos de conexión a ese contenedor de aplicación y los incluiremos en la pestaña publicar para integrarlo con nuestro IDE como se ve en la siguiente imagen.

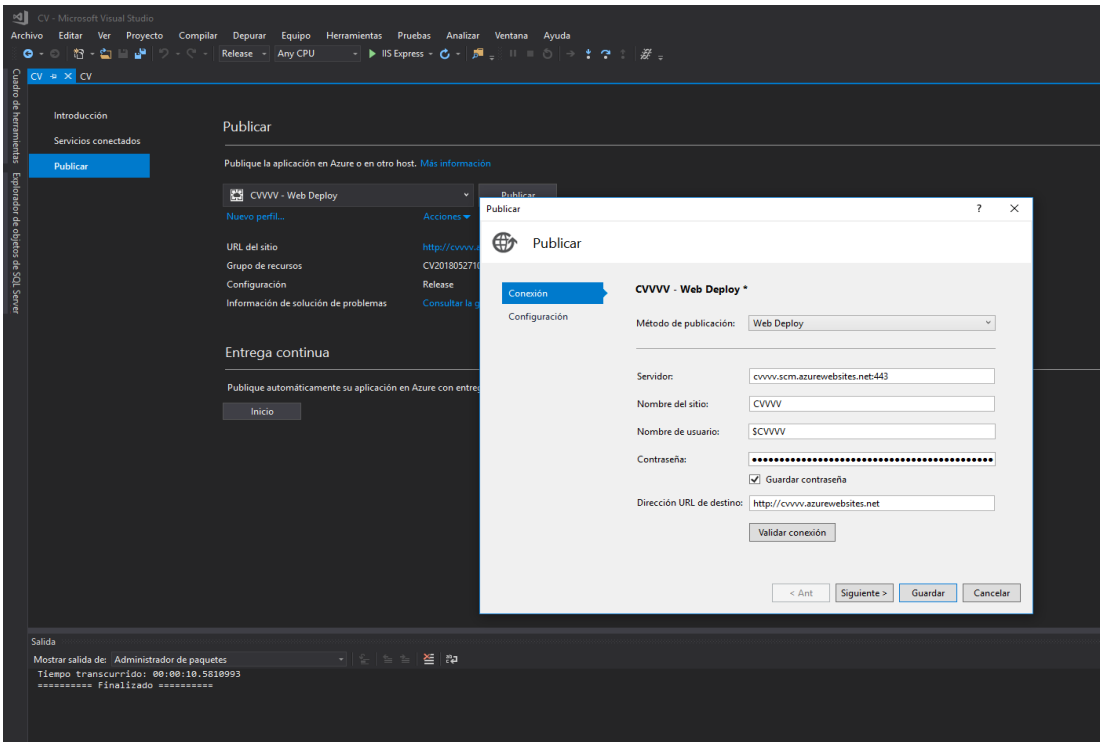

Ilustración 15: Preparación del entorno de producción desde el IDE

De esta forma hemos enlazado el proyecto que estamos desarrollando en nuestro IDE con el contenedor de aplicación de Azure App Service, ahora solo necesitamos publicar la aplicación, pulsado en publicar sobre el proyecto de Visual Studio 2017. Cualquier actualización también es una publicación. La base de datos al estar creada con Entity Framework

mediante los modelos de datos y contener en el proyecto las migraciones se crea de forma automática al subir al servicio el proyecto.

5.2.8 Integración continua y pruebas

Visual Studio 2017 puede utilizar muchos diferentes sistemas de control de versiones como Git o el más usado junto con el IDE llamado TFS (Team Fundation Server).

Con un buen control de ramas y la integración comentada en el punto anterior de Visual Studio con Azure App Service podemos publicar en producción de forma directa las actualizaciones. Hemos decidido no indagar más en este aspecto pues con esa integración y la capacidad de revertir cambios tanto en base de datos con las migraciones del ORM como en la propia plataforma mediante nuevas publicaciones en el servicio lo damos por suficiente.

En el entorno de las pruebas partimos con una gran ventaja por el modelo de datos cerrado del que disponemos, no es posible compilar la solución si no se cumplen todas las políticas del modelo de datos, estás políticas cubren desde los tipos de datos hasta la obligatoriedad del uso de campos, con LINQ y Razor Core evitamos que esto pase en las consultas o actualizaciones de base de datos, pues LINQ no lo permite, y en las capas visuales pues Razor al trabajar con el modelo de datos evita mediante sus componentes la inclusión o uso de datos no válidos.

Las pruebas realizadas sobre la plataforma han iterado siguiendo los requisitos. Al evitarnos de base problemas con el modelo de datos lo más relevante será validar que los diferentes tipos de usuarios podían hacer lo pedido.

5.2.9 Política de mantenimiento

Durante toda la definición del proyecto hemos dado mucha importancia al mantenimiento y crecimiento del sistema. Esta ha sido una de las razones

principales por las que escoger unas u otras tecnologías y usar unas u otras herramientas.

La fase de mantenimiento de un software es en la mayoría de los casos difícil de estimar y bastante más grande de lo deseable si caemos en algunos riesgos.

Nuestra política de mantenimiento evolutivo es seguir siempre que sea posible la misma línea de desarrollo ampliando las funcionabilidades desde modelos de datos y generando los controladores y vistas necesarios. En general el conjunto elegido o stack ASP.NET Core y la arquitectura escogida nos permiten hacer todo lo que podamos imaginar con respecto a la plataforma y por tanto no vemos necesario que en un futuro sea necesario modificar estos pilares que la forman.

Por otro lado, el mantenimiento preventivo debe ser constante y las mejoras en seguridad son necesarias especialmente en lo referente a la autenticación de usuarios.

Durante el mantenimiento correctivo siempre debemos seguir buenas prácticas como:

- Que las mejoras estéticas deben trabajarse exclusivamente desde el contexto de las vistas.
- Se deben nombrar los métodos de los controladores como verbos.
- Es mejor escribir html a usar helpers en las vistas.

5.3 Visión de Usuario

En los siguientes puntos vamos a comentar la visión de usuario, ya sea como administrador, gestor o externo.

Un usuario administrador es el encargado de controlar el sistema, la relación de usuarios con clientes y gestionar opciones visuales.

Un usuario gestor es el usuario general de la aplicación, gestiona los dispositivos, cuestionarios y páginas de estos. Se agrupan por cliente y su ámbito de visión de elementos es lo que su cliente asociado les permite.

Un usuario externo es el que no necesita loguearse y solo pueden responder a las encuentras de las que le han dado un link o que muestran los dispositivos físicos.

Los usuarios que necesitan loguearse, o sea no externos, comienzan viendo la siguiente pantalla de login general.

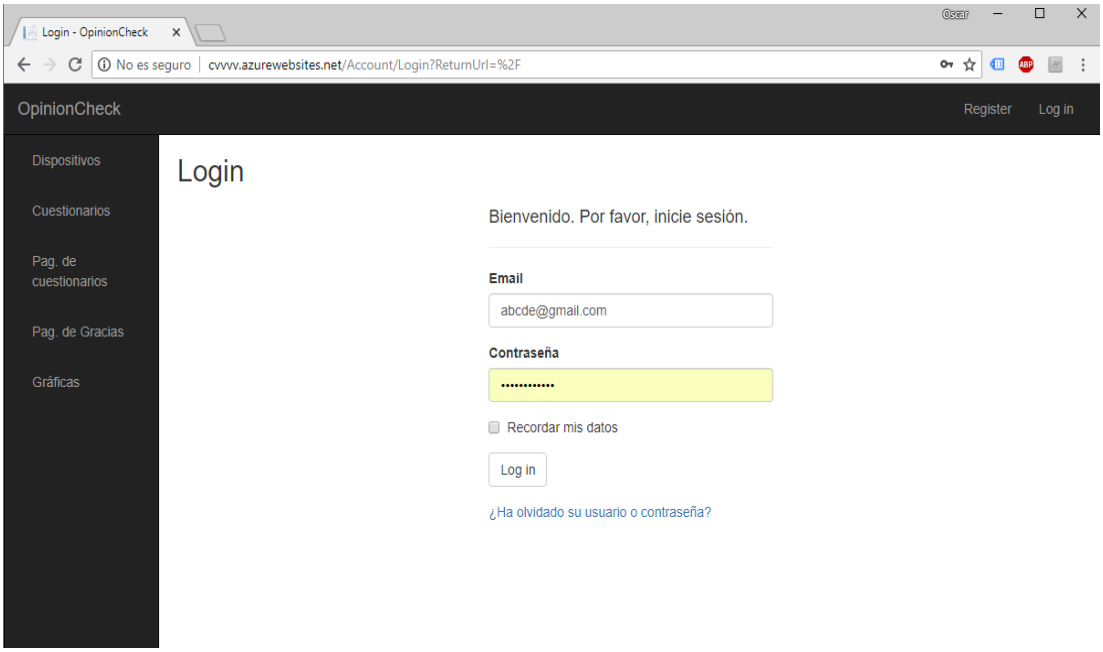

Ilustración 16: Visión usuarios login

5.3.1 Visión como administrador

Al iniciar sesión un usuario administrador dispone de todas las opciones de un usuario gestor, pero ademas puede ver en el menú lateral izquierdo debajo de las opciones generales de la aplicación las de administración.

Las opciones generales del menú son: Dispositivos, Cuestionarios, Pag de cuestionarios, Pag de Gracias, y Gráficas.

Las adicionales para administradores: Respuestas (para ver errores), Opciones Visuales, Clientes y Usuarios.

En el manual de usuario, incluido como anexo, podemos encontrar cada uno de estas opciones más definidas, pero básicamente hacen lo pedido en los requisitos de la plataforma.

El usuario administrador al autenticarse ve la siguiente pantalla:

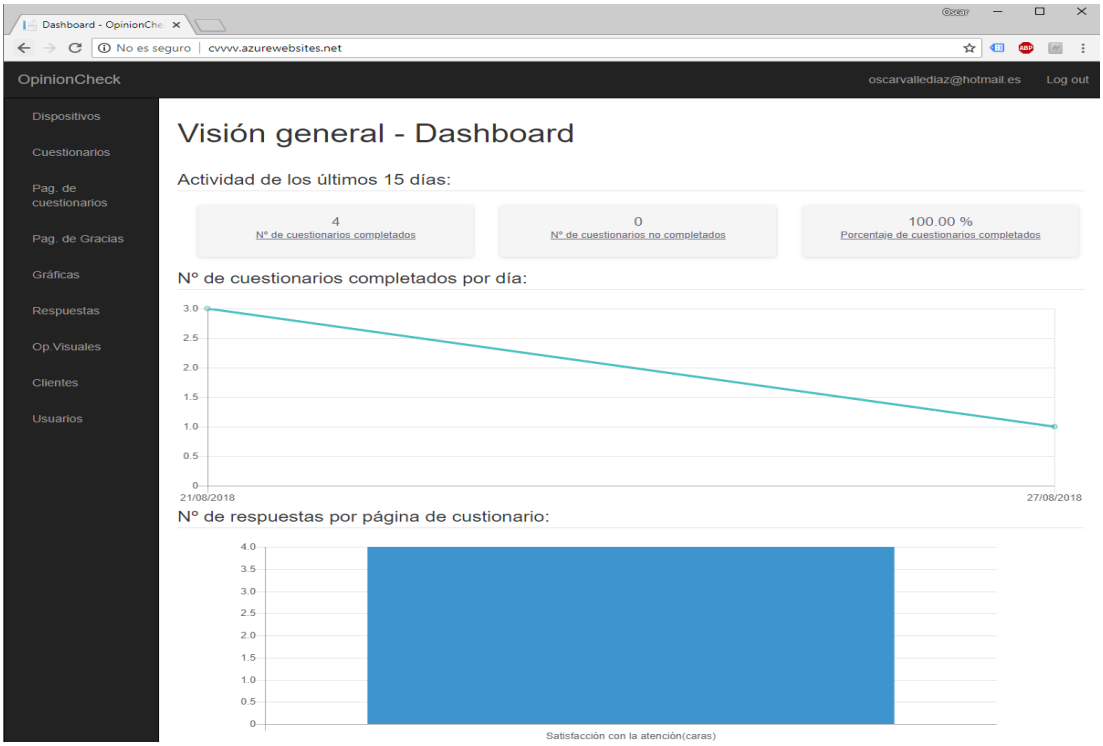

Ilustración 17: Visión usuarios administradores

La pantalla es el dashboard general con datos de su cliente enlazado. Un usuario administrador puede usar el sistema como cualquier usuario, pero tiene esas otras cuatro opciones propias disponibles.

5.3.2 Visión como gestor

Al iniciar sesión un usuario no administrador ve algo similar al usuario administrador, pero sin las opciones de administrador.

Este usuario puede ser administrador si un administrador desde su panel lo marca como administrador. El ser administrador es un Rol de usuario que puede darse o quitarse desde el panel de Usuarios de los administradores marcando un check, esto se comenta con más detalle en el anexo con el manual de usuario

En la siguiente imagen vemos el dashboard de un usuario gestor:

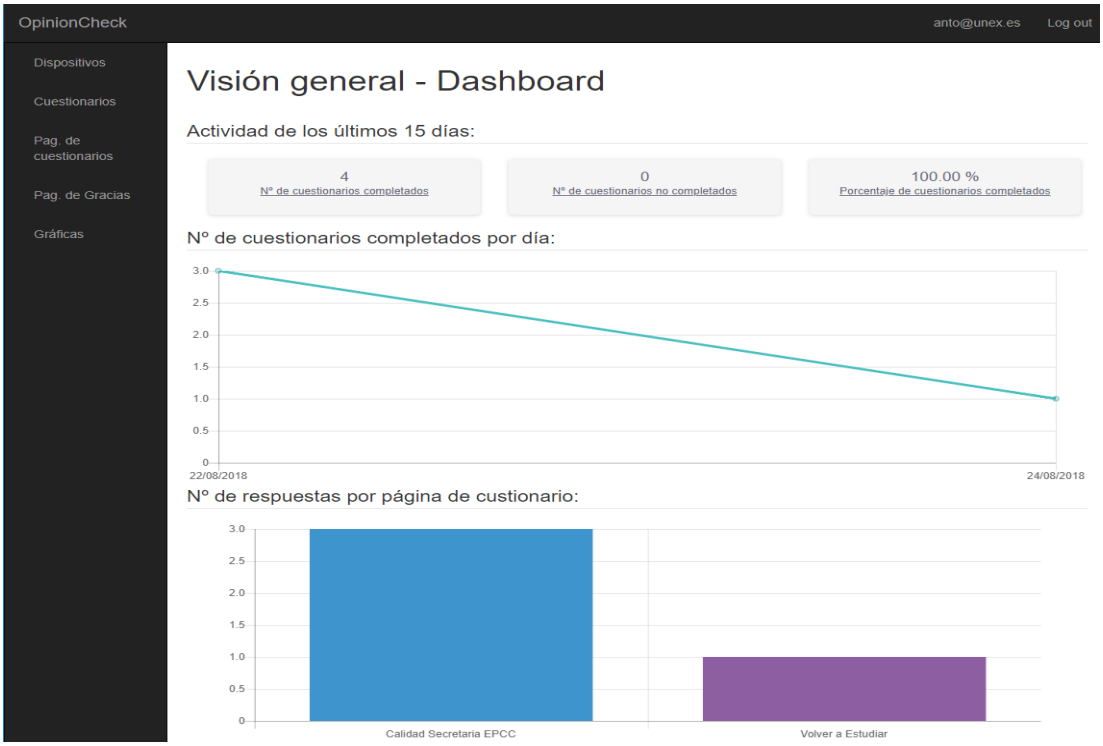

Ilustración 18: Visión usuarios no administradores

La siguiente imagen muestra la lista de dispositivos. El estilo de la página es similar al de cuestionarios, páginas de cuestionario o gracias.

Ilustración 19: Visión usuarios no administradores dispositivos

Estos dispositivos se pueden probar vía aplicación web pulsando en probar, y veríamos la siguiente pantalla, que es el modo de visualización de un

cuestionario para ser respondido. Responderlo en modo prueba no se registran datos.

Ilustración 20: Visión usuarios no administradores test

La siguiente imagen es una muestra de la pestaña de gráficas donde podemos filtrar los datos.

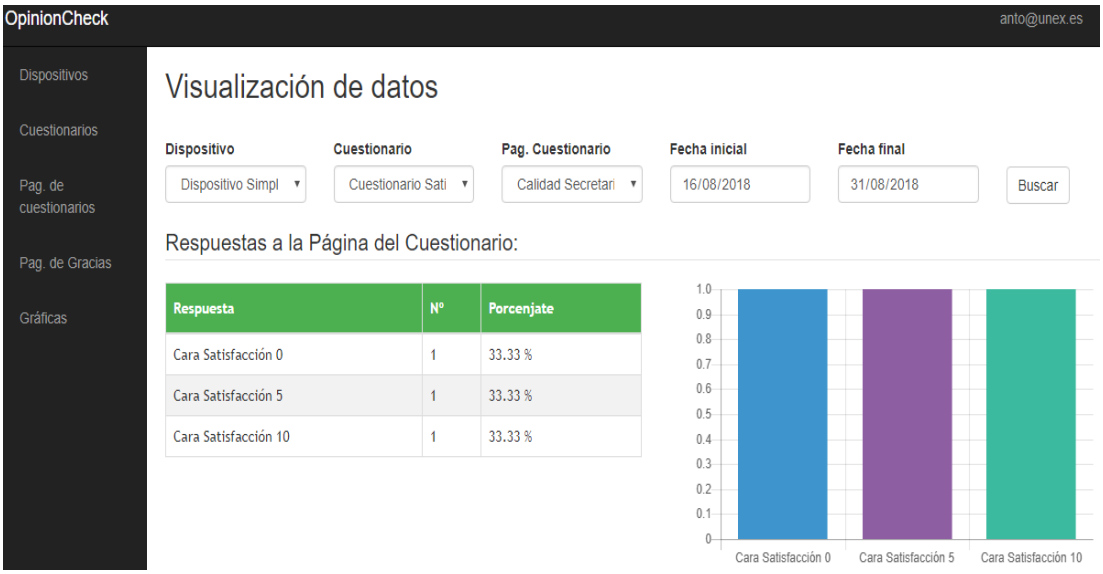

Ilustración 21: Visión usuarios no administradores gráficas

Esto son sólo algunos ejemplos de diferentes páginas.

5.3.3 Visión como externo

El usuario externo puede responder cuestionarios sin loguearse de dos formas. O bien mediante el acceso mediante un link pasado, o mediante la presentación en los dispositivos físicos de proveedores como tablets. La segunda es a través de nuestros prototipos físicos que lanzan las respuestas a través de la API.

5.3.3.1 Páginas visuales de cuestionario

Un usuario externo se acercarse a una tablet que muestra nuestra la página inicial del cuestionario elegido por el gestor verá lo siguiente:

Ilustración 22: Visión usuarios externos en dispositivo físico con pantalla

Cuando acaba de responder al cuestionario llega a una página de agradecimiento y se reinicia el cuestionario para el siguiente usuario.

5.3.3.2 API para integración con dispositivos

En el caso de que el usuario tenga que contestar una pregunta desde uno de nuestros prototipos u otro sistema sin interface visual en el futuro la respuesta llegara al servidor a través de la API REST creada para tal uso.

La interface de la misma es como el siguiente:

http://dominio.net/ApiR?r=1&codArt=ADF455&clave=223RFT1

Los dispositivos cuentan con un código de activación, la clave es una validación simple para que cualquiera que tenga el código de activación sólo con ello no pueda usar la API y r es el parámetro respuesta valor numérico de 1 a X siendo X el número de respuestas posibles definidas en la página del cuestionario, estás respuestas van en orden al igual que se muestran de forma visual de izquierda a derecha, aquí tendríamos la correlación de valor 1 a X para el parámetro r.

5.4 Dispositivos

El objetivo de este apartado y el interés de tener una interfaz física accesible de nuestro sistema vienen motivados por la siguiente cita celebre de Henry Ford (1863-1947):

> *"El verdadero progreso es el que pone la tecnología al alcance de todos."* [22]

Por ello la propuesta de este proyecto incluye la realización de prototipos propios que hagan accesible el sistema desarrollado. Como podremos ver en los siguientes puntos la principal razón de crear nuestros propios prototipos es que para algunos casos es más fácil y económico el uso de los mismos, pues los dispositivos completos de proveedores con los que puede ser usado el sistema tienen un coste superior y un ámbito mucho más amplio.

Es importante entender claramente qué es un dispositivo, para ello nos podemos ayudar de la definición, que ampara nuestro ámbito de estudio, del Oxford Dictionaries:

> *Pieza o conjunto de piezas o elementos preparados para realizar una función determinada y que generalmente forman parte de un conjunto más complejo.* [23]

Exactamente ese es el objetivo de este apartado, el estudiar y crear elementos que cumplan con la función de acercar al usuario la interfaz y le permitan de la forma más simple posible responder a cuestiones determinadas.

En la introducción de este proyecto comentábamos que poder realizar encuestas y que se puedan responder preguntas de forma presencial era bastante valioso para obtener datos exactos y de un público más homogéneo. En los siguientes puntos desarrollaremos de forma técnica nuestra solución para poder hacerlo.

Se estudiarán los requisitos mínimos para que los dispositivos puedan ser usados fácilmente como interfaz física del sistema software y se desarrollarán dos prototipos propios, con el objetivo de hacer más accesible y económico el uso del sistema.

El alcance y objetivo de los dispositivos comentados en los siguientes puntos es que sean usados por usuarios externos para responder preguntas. La gestión se realizará desde la plataforma web sin repercusión de ello en este apartado.

5.4.1 Requisitos mínimos de los dispositivos

En las siguientes tablas se muestran las listas de requisitos mínimos, también llamados requerimientos, que necesitan cumplir los dispositivos para poder ser usados con el sistema software y ser útiles para el usuario externo. Esta lista de requisitos ha sido realizada gracias al análisis de la plataforma web desarrollada y al objetivo de servir de interfaz física para el usuario externo, es decir, los dispositivos son las herramientas recolectoras de datos del sistema.

Los requisitos mínimos se van a clasificar en requisitos funcionales y requisitos no funcionales para facilitar su entendimiento.

Tabla 6: Tabla de requisitos funcionales de los dispositivos

Requisito	**Título**	**Descripción**
RF-01	Muestra de una pregunta	El dispositivo debe poder mostrar una pregunta.
RF-02	Muestra de diferentes opciones	El dispositivo debe mostrar diferentes opciones cerradas (únicas).
RF-03	Selección única	El dispositivo sólo permitirá la selección de una única opción cada vez.
RF-04	Envío único	El dispositivo sólo permitirá enviar una opción controlando dobles pulsaciones, pulsaciones continuadas, etc.
RF-05	Reinicio instantáneo	Tras el envío de una respuesta el dispositivo estará listo para enviar otra.
RF-06	Configuración interna básica	El dispositivo debe registrar las respuesta bajo un código propio del mismo.
RF-07	Uso conectado a red y a batería	El dispositivo debe poder conectarse a la red eléctrica o a una batería para su uso desconectado.
RF-08	Conectividad inalámbrica	El dispositivo debe poder conectarse a la Internet mediante Wifi o GSM.

Tabla 7: Tabla de requisitos no funcionales de los dispositivos

Requisito	**Título**	**Descripción**
RNF-01	Seguridad cedida a la plataforma web	Toda la seguridad deber estar cedida a la plataforma web, el dispositivo no validara nada de forma interna.
RNF-02	Facilidad de uso	Se deben simplificar los elementos con los que interactuara el usuario lo máximo.

		posible.
RNF-03	Eficiencia. Tiempo de respuesta.	El dispositivo debe actuar lo más rápido posible para enviar el dato y estar listo para recibir una nueva respuesta inmediatamente.
RNF-04	Rendimiento con consumo mínimo.	El dispositivo debe consumir lo mínimo posible para desempeñar sus funciones.

5.4.2 Dispositivos de proveedores

En este apartado se van a estudiar dispositivos completos que se pueden utilizar con el sistema y que se pueden comprar directamente a algún proveedor. También deben cumplir con los requisitos mínimos de forma general para que tenga sentido su uso.

Estos dispositivos de proveedores tienen una ventaja respecto a los prototipos que vamos a desarrollar en el siguiente punto.

En nuestros prototipos sólo se podrá responder a una sola cuestión, a una pregunta simple y con respuestas cerradas directas, pues hacerlo multipregunta aumenta su coste de fabricación en dos o más veces al necesitar de una pantalla.

En los siguientes dispositivos se puede seguir una secuencia de preguntas con diferentes respuestas, pero son más caros y plantean una instalación, mantenimiento y preparación mayor que nuestros propios prototipos. Esta ventaja e inconvenientes no se incluirán en la siguiente relación al cumplir con ello todos los dispositivos de proveedores.

Para hacer más amigable este apartado de dispositivos de proveedores vamos a mostrar los datos analizados en forma de tabla con sus ventajas e inconvenientes para ser usados con nuestro sistema.

La siguiente tabla contiene lo comentado:

Tabla 8: Tabla de ventajas e inconvenientes de distintos tipos de dispositivos

Tipo de dispositivo	**Ventajas**	**Inconvenientes**
Tablet/Laptops táctiles	Portabilidad. Uso táctil amigable.	
Equipos sobremesa con pantalla táctiles	Uso táctil amigable.	Portabilidad, configuración e instalación compleja por la conectividad.
Laptops no táctiles	Portabilidad.	Uso obligado de periféricos.
Equipos sobremesa sin pantalla táctiles		Portabilidad, configuración e instalación compleja por la conectividad, uso obligado de periféricos.

No es necesario hacer un exhaustivo estudio de costes pues para usar alguno de estos dispositivos de forma eficaz necesitaremos una pantalla de mínimo diez pulgadas y actualmente, a mediados de 2018, no es posible encontrar dispositivos con estas características con una calidad media por menos de 100€.

También en algunos casos sería necesario agregarle con hardware externo a los mismos periféricos para su uso, soportes, protectores de teclado para que el usuario externo no pueda trastocar el equipo, sistemas de conectividad adicional Wifi o GSM, etc, que incrementaran aún más su coste.

En la siguiente imagen podemos ver cómo quedaría nuestro sistema de forma visual en una tablet con un soporte de pie. Para ello solo sería necesario mostrar la página del dispositivo configurada sobre la plataforma web, ella muestra las preguntas y válida lo necesario.

Ilustración 23: Tablet con soporte de pie visualizando cuestionario en app web

5.4.3 Dispositivos propios

Como hemos comentado en apartados anteriores tenemos la capacidad de crear dispositivos más simples, enfocados a realizar una sola pregunta, por un coste mucho menor al de cualquiera de los dispositivos completos de

proveedores y que a la larga darán menos problemas y tendrán un mantenimiento más sencillo gracias a que están compuestos por muchos menos componentes y por componentes más simples.

Nuestros dispositivos tienen una ventaja adicional. Al ser creados sobre el ámbito específico del problema que buscamos resolver y ser propios es más fácil adaptarlos a futuros cambios que el sistema deba hacer y que tengan que ser reflejados en los dispositivos.

También es posible personalizar de una forma muy profunda nuestros dispositivos para adecuarlo a la imagen de una corporación, para adecuar su imagen a la pregunta que se esté realizando, en ese caso para mejorar la usabilidad, y para poder hacerlos más accesibles a usuarios con requisitos especiales como personas con deficiencias o minusvalías.

En los siguientes apartados mostraremos el estudio, diseño y fabricación de nuestro prototipo Wifi, y buscando el máximo aprovechamiento de recursos desde este primer prototipo el desarrollo del segundo prototipo, el prototipo GSM.

El estudio de costes se adjuntará como anexo tanto del primer prototipo como del segundo.

Al tener muy claros los objetivos y requisitos mínimos no ha sido necesario iterar con diferentes soluciones para encontrar un prototipo que pudiera cumplir su misión. La primera solución aportada es válida y completa.

5.4.3.1 Dispositivo Prototipo Wifi

Este primer prototipo busca la solución más simple que cumpla con los requisitos básicos (en el modo conectividad inalámbrica de corto-medio alcance con Wifi). Se ha acotado al uso de tres pulsadores por razones de coste pues no es necesario hacer un gasto superior para probar el sistema. Es suficiente con tener 3 posibles respuestas para una pregunta, aunque como se verá en los diagramas y el código es sumamente sencillo ampliar el número de pulsadores.

Como seguimos una metodología de modelo de prototipos, buscando el desarrollo rápido de prototipos, la implementación de los requisitos se realizará de una forma directa atajándolos por el camino más rápido para construir el sistema.

Implementación del prototipo según la lista de requisitos:

RF-01 Muestra de una pregunta

Por motivos de costes no utilizaremos una pantalla para mostrar una sola pregunta, la sobreimprimiremos en el dispositivo.

Si ha de cambiar en un futuro solo será necesario modificar el cuestionario del dispositivo en la plataforma web, para que ahora se guarden los nuevos datos, y cambiar la pregunta impresa del dispositivo por otra de forma manual.

RF-02 Muestra de diferentes opciones

Como se ha comentado tendremos 3 pulsadores. Para que se vea claramente lo que esta respondido el usuario tendrán sobreimpreso o pegado la respuesta o una iconografía de la respuesta. También es posible que se sobreimprima o pegue una ayuda textual alrededor de los mismos para ayudar a usuarios que no comprendan el dispositivo.

En la ilustración siguiente el cartel superior sería la pregunta impresa y los pulsadores están cubiertos por las respuestas internamente valoradas como 0, 5 y 10 como si de una puntuación se tratase, pero mostrándoselo al usuario de esta forma más visual.

Ejemplo gráfico de lo que sería el concepto externo del prototipo con pregunta simple y tres respuestas cerradas:

Ilustración 24: Ejemplo gráfico del Prototipo Wifi

RF-03 Selección única

El código se implementará de forma que no sea posible la selección múltiple. Se enviarán respuesta de una en una, una bandera o flag se encargará de bloquear el envío de más de una respuesta por pulsación, bloqueando que hasta que no se levanten todos los pulsadores no se pueda volver a enviar.

RF-04 Envío único

En el código implementará un bloqueo temporal para que no se envíen respuesta de forma errónea por dobles pulsaciones rápidas y otros sucesos mecánicos.

RF-05 Reinicio instantáneo

El hardware seleccionado será una placa cuya velocidad nos permita enviar una respuesta y momentos después estar lista para volver a recibir una pulsación y repetir el ciclo sin interrupciones.

RF-06 Configuración interna básica

El código contendrá variables para almacenar la ruta y códigos necesarios para enviarlos a través de la API de la plataforma web que representaran al dispositivo.

RF-07 Uso conectado a red y a batería

El hardware seleccionado será compatible con estás dos formas de alimentación.

RF-08 Conectividad inalámbrica

El hardware seleccionado para este prototipo dispondrá de un sistema de conectividad inalámbrica por Wifi. Y el código implementara lo necesario para conectarse a la red configurada en el mismo, en las variables *WIFI_SSID, WIFI_PASSWORD.*

Los requisitos no funcionales se implementarán de igual forma en todo el prototipo, al estar definidas en si mismas no vamos a entrar en más detalles.

El prototipo:

En la búsqueda de soluciones abiertas y más ajustadas a lo necesario para la implementación del hardware seleccionado ha sido una placa Wemos Mini D1 Wifi con 4MB de memoria Flash y basada en el microcontrolador ESP-8266EX. [24]

Se ha elegido este hardware por ser el más pequeño, económico y con disponibilidad que cumplía con los requisitos mínimos y está muy enfocado en la construcción de prototipos. A la par se encontraban Arduino Uno [25] pero es demasiado grande y con un consumo mayor y Arduino Nano [26] que también es algo más grande. pero con un coste mayor que la placa Wemos seleccionada.

Para componer el circuito se van a utilizar también cables sencillos de cobre, resistencias de 1K Ohmio y tres pulsadores momentáneos de 22mm tipo NO (normalmente abierto). Es aconsejable el uso de estos NO pues al estar normalmente abiertos la placa sólo consume energía al estar pulsados. El siguiente circuito es válido tanto para normalmente abiertos como cerrados.

A continuación, se muestras los circuitos realizados con la herramienta Fritzing [27] que permite ver los circuitos de una forma fácil y clara.

El circuito en formato protoboard del prototipo:

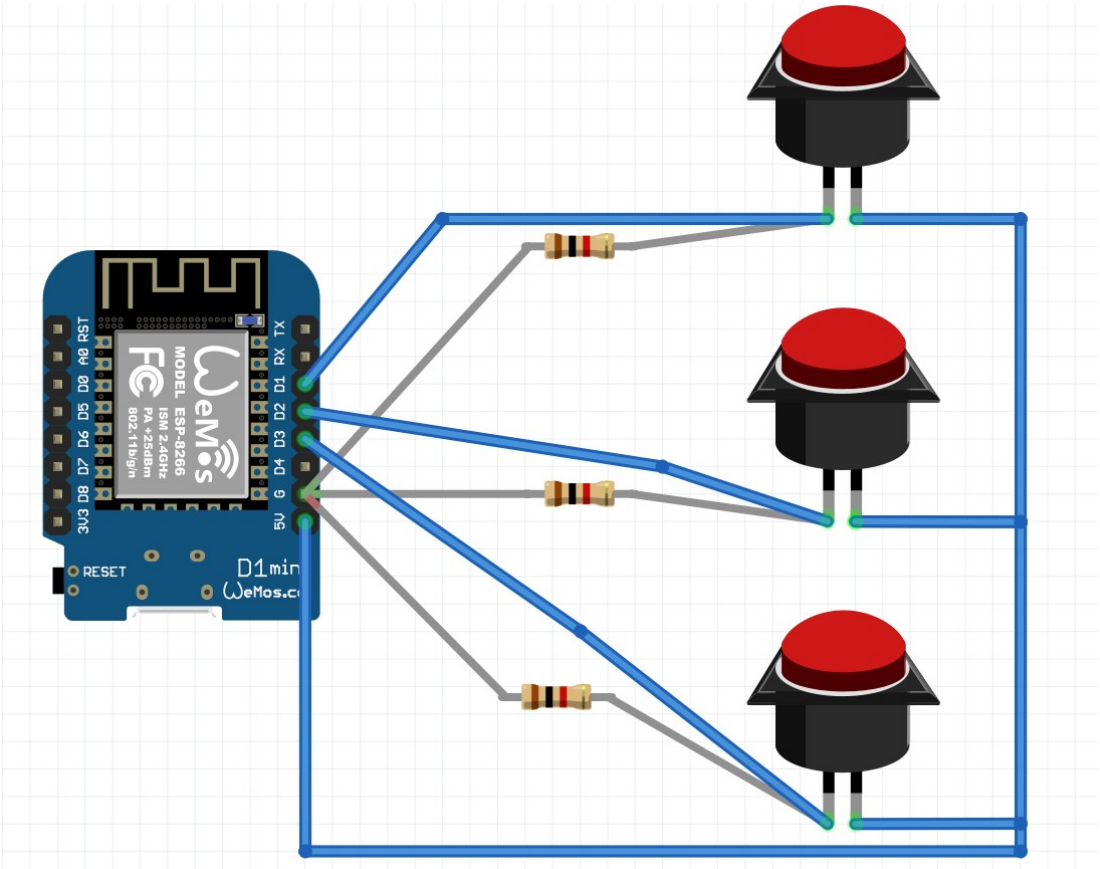

Ilustración 25: Ejemplo del circuito del Prototipo Wifi formato Protoboard Visual

El circuito en formato esquemático del prototipo:

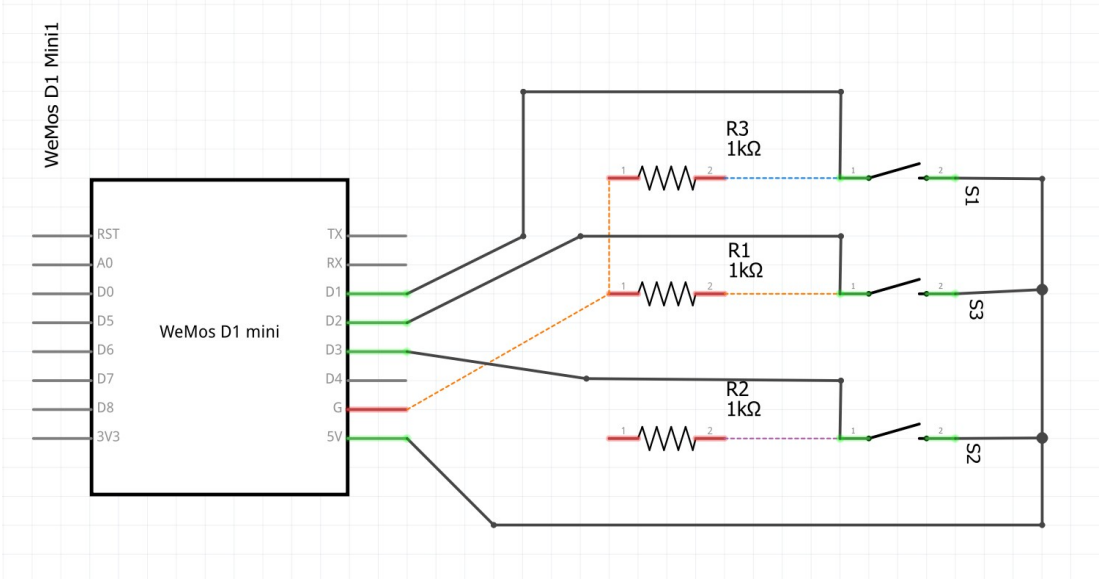

Ilustración 26: Ejemplo del circuito del Prototipo Wifi formato Esquemático

El circuito y los pulsadores se han incluido en una carcasa de controles con 3 controles como la que se ve en la siguiente imagen.

Ilustración 27: Carcasa Caja de Controles para 3 pulsadores

A continuación se muestra el código implementado sobre el editor oficial de Arduino para Windows.

El código:

```
/* Nombre: 3ButtonsOpinionControlForESP8266WiFi
 * Version: 1.0.0
 * Fecha: 18-04-2018
  * Descripción: Controlador de 3 pulsadores para la recogida de opinión sobre API desarrollada en el TFG
 *    "Diseño e implementación de sistema de obtención de opinión de usuarios"(Oscar Valle Diaz, 2018).
 *    Conectado por Wifi.
 * Probado en placa: Wemos D1 Mini
 * Autor: Oscar Valle Diaz
 */

#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

//Constantes
#define WIFI_SSID "VODAFONE Ejemplo SSID 33"
#define WIFI_PASSWORD "pssVodafoneEjemp"

const int buttonPin1 = D1;
const int buttonPin2 = D2;
const int buttonPin3 = D4;
const int ledPin =  LED_BUILTIN;

//Variables
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int flag=0;

String apiGet = "http://cvvvv.azurewebsites.net/ApiR?"; // "r=1&codAct=ADF455&clave=223RFT1";

void setup() {
  Serial.begin(115200);
```

```
  conectarWifi();

  //Inializaciones
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(buttonPin3, INPUT);
}

void loop() {
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = !digitalRead(buttonPin2); //Ejemplo NC(normalmente cerrado), invertir valor
  buttonState3 = digitalRead(buttonPin3);

  //Se controla la pulsación
  if (buttonState1 == HIGH or buttonState2 == HIGH or buttonState3 == HIGH) {
    if ( flag == 0){
      //Orden de izquierda a derecha en el panel físico
      if (buttonState1 == HIGH){
            enviarRespuesta(0);
      }else if (buttonState2 == HIGH){
            enviarRespuesta(1);
      }else if (buttonState3 == HIGH){
            enviarRespuesta(2);
      }
    }
  }else if(flag == 1){
    //En el momento en el que suelta el pulsador se vuelve al estado original
     digitalWrite(ledPin, LOW);
     flag=0;
     Serial.println("Sueltos Todo Los Botones");
  }


  if(WiFi.status() != WL_CONNECTED){
    conectarWifi();
  }
}
```

```
void enviarRespuesta(int idVO){
    digitalWrite(ledPin, HIGH);
    enviarPorHttp(idVO);
    delay(1000); //Evita doble pulsación por rebote del botón
    Serial.println("..ENVIADO " + String(idVO) +" ..");
    flag=1;
}

void enviarPorHttp(int idVO){
  if (WiFi.status() == WL_CONNECTED) {
   HTTPClient httpCli;
   httpCli.begin(String(apiGet + "r=" + idVO + "&codAct=ADF455GH66&clave=223123FG1"));
   int httpRespCod = httpCli.GET();
   httpCli.end();
  }
}

void conectarWifi(){
    //Módulo WiFi modo station, desconectar cualquier red que estubiera conectada
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(2000);
       /* Conexión de la WiFi */
         WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
         Serial.print("Conectando....");
         while (WiFi.status() != WL_CONNECTED) {
           Serial.print(".");
           delay(500);
         }
         Serial.println();
         Serial.println("Conectado como: " + WiFi.localIP());
}
```

Notas sobre el prototipo fabricado:

El prototipo fabricado funciona correctamente, cumpliendo todos los requisitos y registrando las respuestas individuales en la aplicación web.

La siguiente imagen muestra el prototipo real por dentro con todas las conexiones del circuito mostrado anteriormente en este apartado:

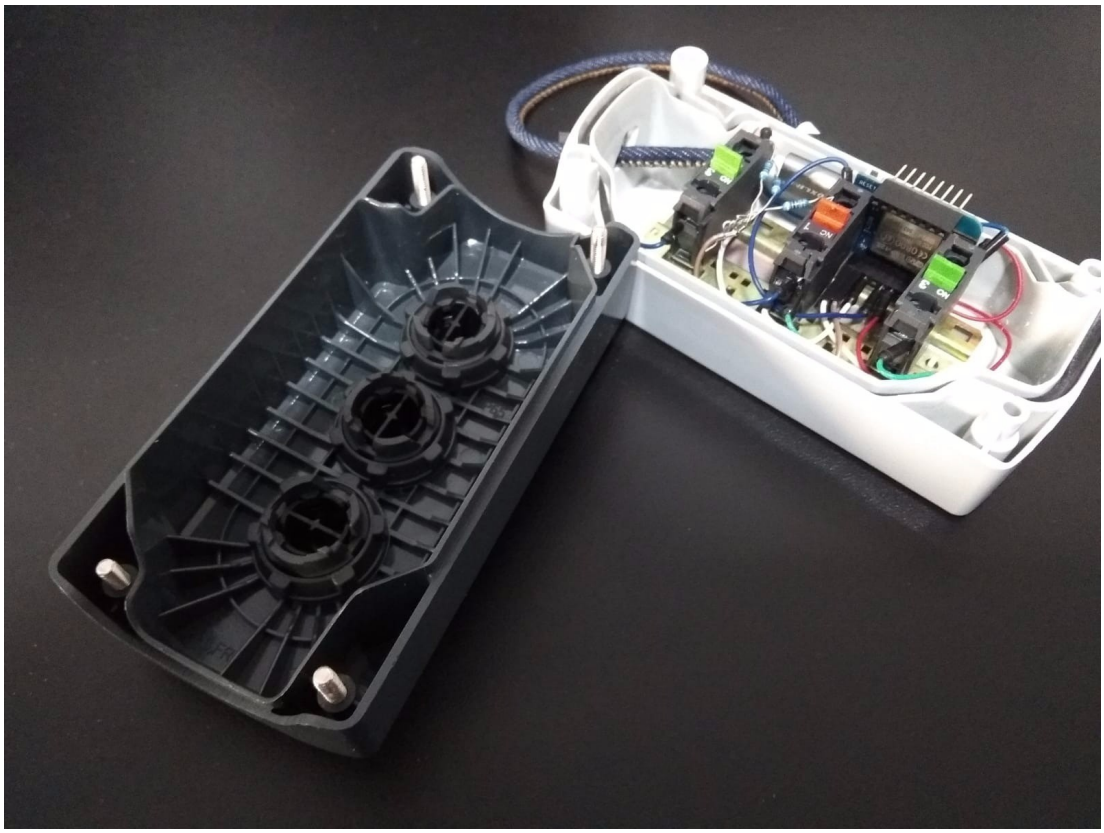

Ilustración 28: Prototipo Wifi fabricado interior

Al tener tantos componentes es un espacio tan ajustado es difícil diferenciarlos. En la tapa con tornillos se encuentran las cabezas de los tres pulsadores, en esta imagen se ven por detrás y sus mecanismos son las tres piezas negras con puntas verdes a los lados y una con punta roja en el centro.

También se pueden diferenciar el cableado, las resistencias y la placa encajada entre dos mecanismos de los pulsadores.

En la siguiente imagen podemos verlo desde otro ángulo:

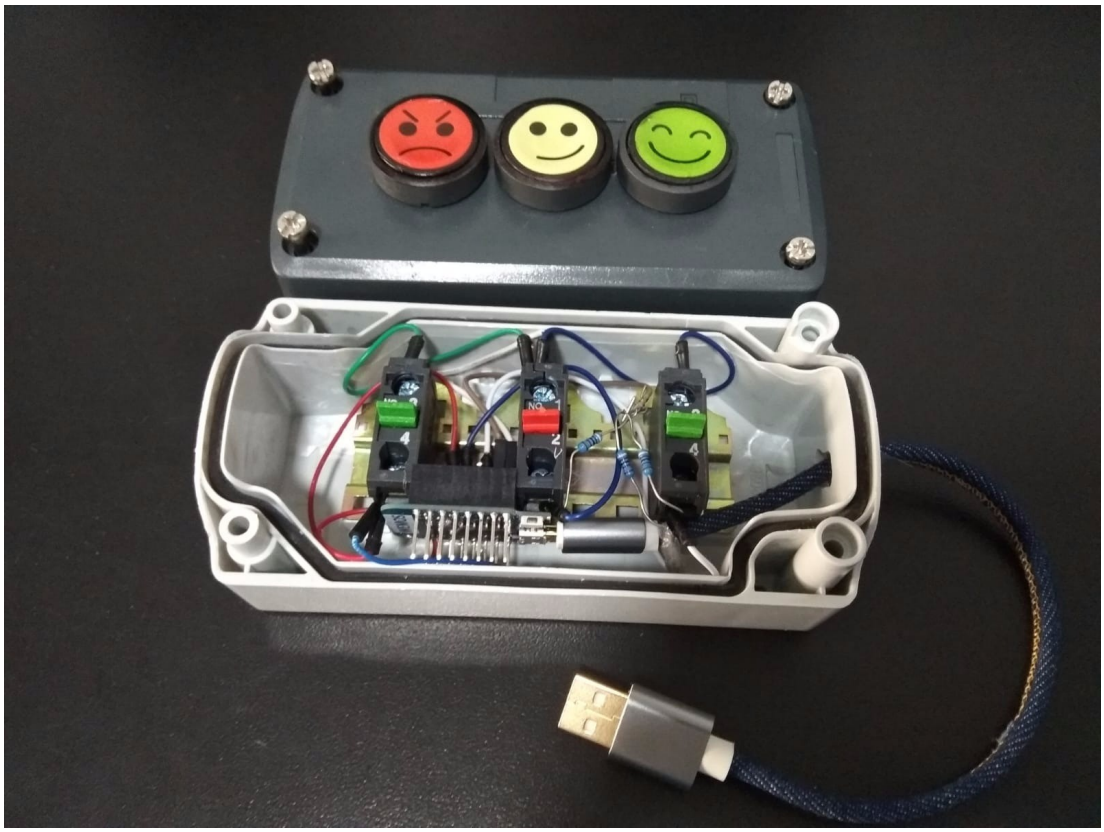

Ilustración 29: Prototipo Wifi fabricado interior ángulo delantero

Ahora es posible apreciar el cable USB conectado a la placa y el lado exterior de los pulsadores con sus respuestas en formato visual.

La siguiente imagen se puede ver claramente la placa Wemos Mini D1 usada.

Al ser solo tres botones no ha sido necesario usar los pin que en la imagen vemos arriba.

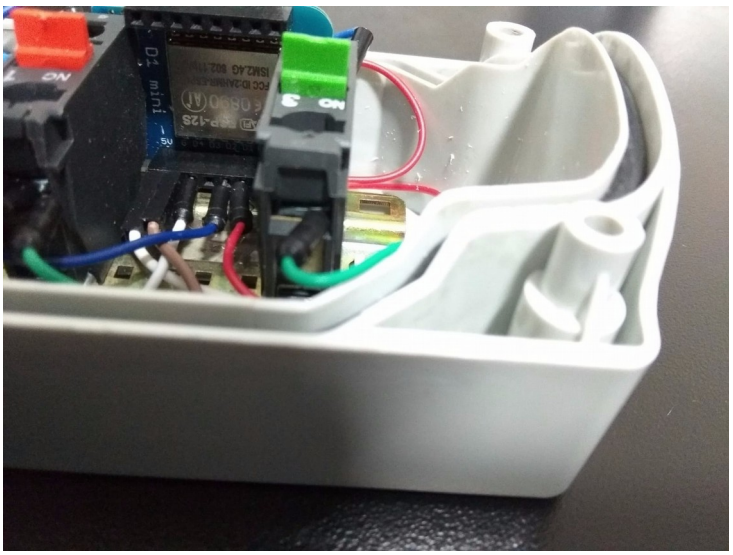

Ilustración 30: Prototipo Wifi fabricado interior placa

Una muestra del prototipo cerrado y listo para usarse a falta de la pregunta impresa sobre el chasis agregado sería la de la siguiente imagen:

Ilustración 31: Prototipo Wifi fabricado exterior

Y con la pregunta impresa queda como en la siguiente imagen:

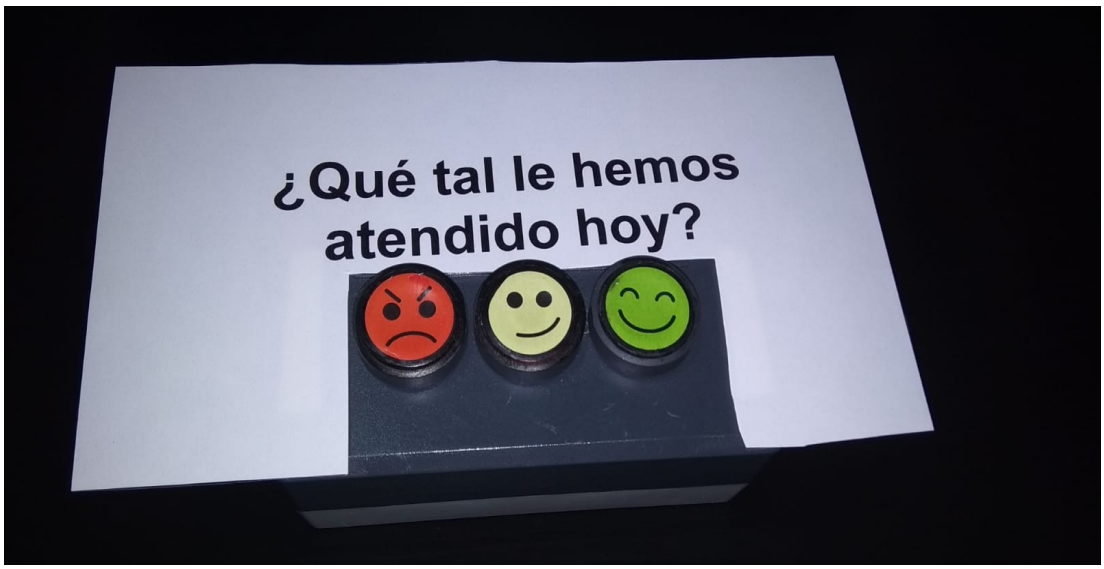

Ilustración 32: Prototipo Wifi fabricado exterior con pregunta

El coste medio aproximado de los materiales y componentes necesarios para la fabricación del prototipo ha sido de 20,30€ con IVA incluido. Se puede comprobar en el anexo 1 presupuesto de prototipos.

5.4.3.1 Dispositivo Prototipo GSM

Gracias a nuestro prototipo anterior y su diseño podemos reutilizar gran parte del código y el hardware para crear este nuevo dispositivo. Esto ahorra costes de desarrollo, mantenimiento y si deseáramos fabricar un gran conjunto de ellos para hacer pruebas con usuarios en diferentes sitios también facilitaría el proceso productivo.

En este caso el dispositivo prototipado para trabajar con conectividad GSM mediante una tarjeta SIM sólo nos genera una variación sobre el desarrollo del requisito funcional 08.

RF-08 Conectividad inalámbrica

El hardware seleccionado para este prototipo dispondrá de un sistema de conectividad inalámbrica por GSM/GRPS. Y el código implementara lo necesario para conectarse a la red general de la compañía de red de la tarjeta insertada.

Los requisitos no funcionales se implementarán de igual forma en todo el prototipo, al estar definidas en si mismas no vamos a entrar en más detalles tampoco para este prototipo.

El prototipo:

En la búsqueda de soluciones abiertas, y más ajustadas a lo necesario, para la implementación del hardware seleccionado hemos encontrado un módulo pequeño e inferior a otros en coste, el Módulo GSM/GPRS SIM800L [28]

Este módulo puede trabajar con cualquier microcontrolador que soporte interfaz serial TTL de 3.3V y 5V, es Quad-Band (850/900/1800/1900 MHz) pudiendo transmitir SMS, voz, y datos.

Con un tamaño de tan solo 40 mm * 28 mm * 3 mm es ideal para incluirlo dentro de la carcasa seleccionada y adaptada en el prototipo anterior. Sólo

necesitamos conectar una MicroSim válida para trabajar en esas bandas con el operador adecuado (dependiendo de la cobertura necesaria)

Por su tipo de interfaz la conexión es directa desde sus pines RX y TX. Gracias a su antena externa, su reducido tamaño y sus limitadas características u opciones disponibles para este tipo de placas, aunque es totalmente adecuada y completa para nuestras necesidades, la placa es capaz de consumir menos energía que otras similares más grandes y potentes del mercado.

A continuación, se muestran los circuitos, evolucionados desde el prototipo anterior, realizados con la herramienta Fritzing [27] en los dos formatos:

El circuito en formato protoboard del prototipo GSM:

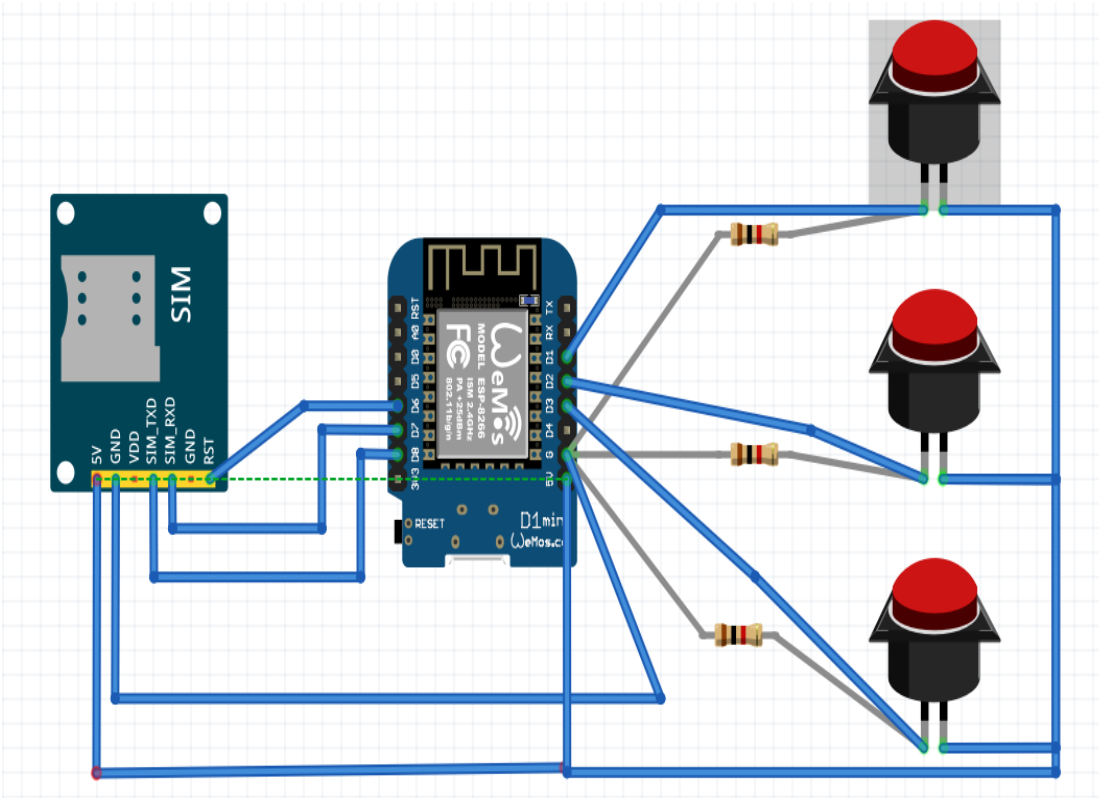

Ilustración 33: Ejemplo del circuito Prototipo GSM formato Protoboard Visual

El circuito en formato esquemático del prototipo GSM:

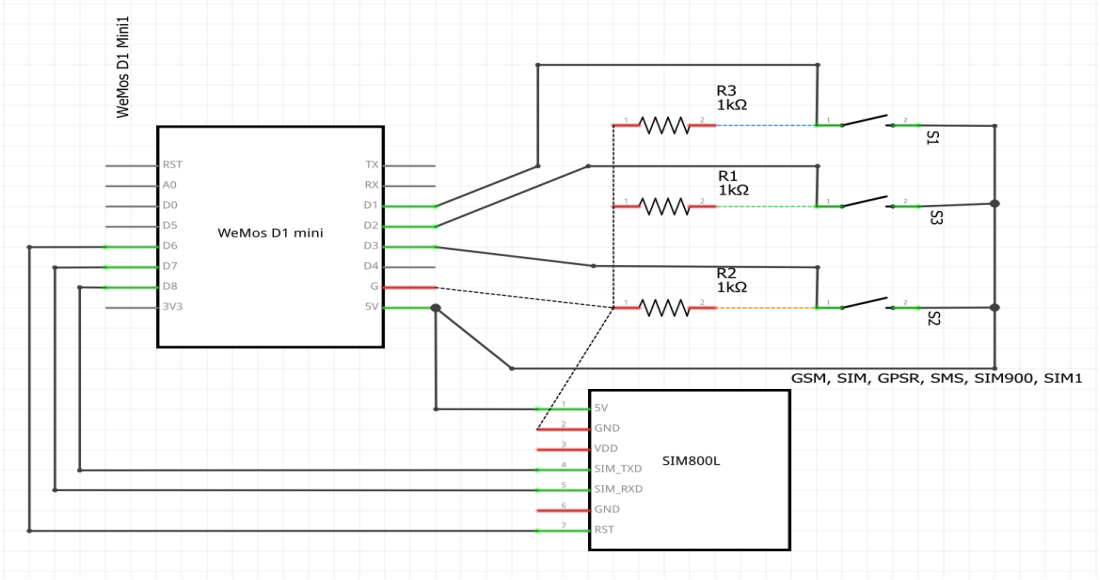

Ilustración 34: Ejemplo del circuito del Prototipo GSM formato Esquemático

El circuito y los pulsadores se han incluido en una carcasa de controles con 3 controles, igual a la del anterior protitipo.

A continuación se muestra el código implementado sobre el editor oficial de Arduino en Windows. Con tarjeta SIM sin clave, para facilitar la configuración y con la librería https://github.com/carrascoacd/ArduinoSIM800L.

El código:

```
/* Nombre: 3ButtonsOpinionControlForESP8266GSM
 * Version: 1.0.0
 * Fecha: 20-06-2018
  * Descripción: Controlador de 3 pulsadores para la recogida de opinión sobre API desarrollada en el TFG
 *          "Diseño e implementación de sistema de obtención de opinión de usuarios"(Oscar Valle Diaz, 2018).
 *          Conectado por GSM.
 * Probado en placa: Wemos D1 Mini
  * Autor: Oscar Valle Diaz
 */

#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <Http.h>
```

```
unsigned int RX_PIN = 7;
unsigned int TX_PIN = 8;
unsigned int RST_PIN = 6;
HTTP http(9600, RX_PIN, TX_PIN, RST_PIN);

const int buttonPin1 = D1;
const int buttonPin2 = D2;
const int buttonPin3 = D4;
const int ledPin =  LED_BUILTIN;

//Variables
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
int flag=0;

String apiGet = "http://cvvvv.azurewebsites.net/ApiR?"; //
"r=1&codAct=ADF455&clave=223RFT1";

void setup() {
  Serial.begin(9600);
  while(!Serial);

  Serial.println("¡Empezamos!");

  //Inializaciones
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(buttonPin3, INPUT);
}

void loop() {
  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = !digitalRead(buttonPin2); //Ejemplo NC(normalmente cerrado), invertir valor
  buttonState3 = digitalRead(buttonPin3);
  /*
     Serial.println("..D1...." + String(buttonState1) +" ..");
     Serial.println("..D2....." + String(buttonState2) +" ..");
     Serial.println("..D3....." + String(buttonState3) +" ..");
```

```
*/

 //Se controla la pulsación
 if (buttonState1 == HIGH or buttonState2 == HIGH or buttonState3 == HIGH) {
   if ( flag == 0){
     //Orden de izquierda a derecha en el panel físico
     if (buttonState1 == HIGH){
          enviarRespuesta(0);
     }else if (buttonState2 == HIGH){
          enviarRespuesta(1);
     }else if (buttonState3 == HIGH){
          enviarRespuesta(2);
     }
   }
 }else if(flag == 1){
   //En el momento en el que suelta el pulsador se vuelve al estado original
    digitalWrite(ledPin, LOW);
    flag=0;
    Serial.println("Sueltos Todo Los Botones");
 }
}

void enviarRespuesta(int idVO){
     digitalWrite(ledPin, HIGH);
     enviarPorHttp(idVO);
     delay(1000); //Evita doble pulsación por rebote del botón
     Serial.println("..ENVIADO " + String(idVO) +" ..");
     flag=1;
}

void enviarPorHttp(int idVO){
     http.configureBearer("orange.es"); //TODO poner operador
     http.connect();

     char response[256];
                    Result result = http.get(String(apiGet + "r=" + idVO +
"&codAct=ADF455GH66&clave=223123FG1"), response);

     Serial.println(response);
```

```
    http.disconnect();
}
```

Notas sobre el prototipo fabricado:

Este segundo prototipo fabricado funciona también cumpliendo todos los requisitos y registrando las respuestas individuales en la aplicación web. La conectividad es algo más lenta que en el caso anterior pero suficientemente rápida para no generar ningún problema.

La siguiente imagen muestra el prototipo real GSM por dentro con el lugar en el que se sitúa la placa, sin algunos pines conectados como los de su alimentación o los RX/TX para que sea visible:

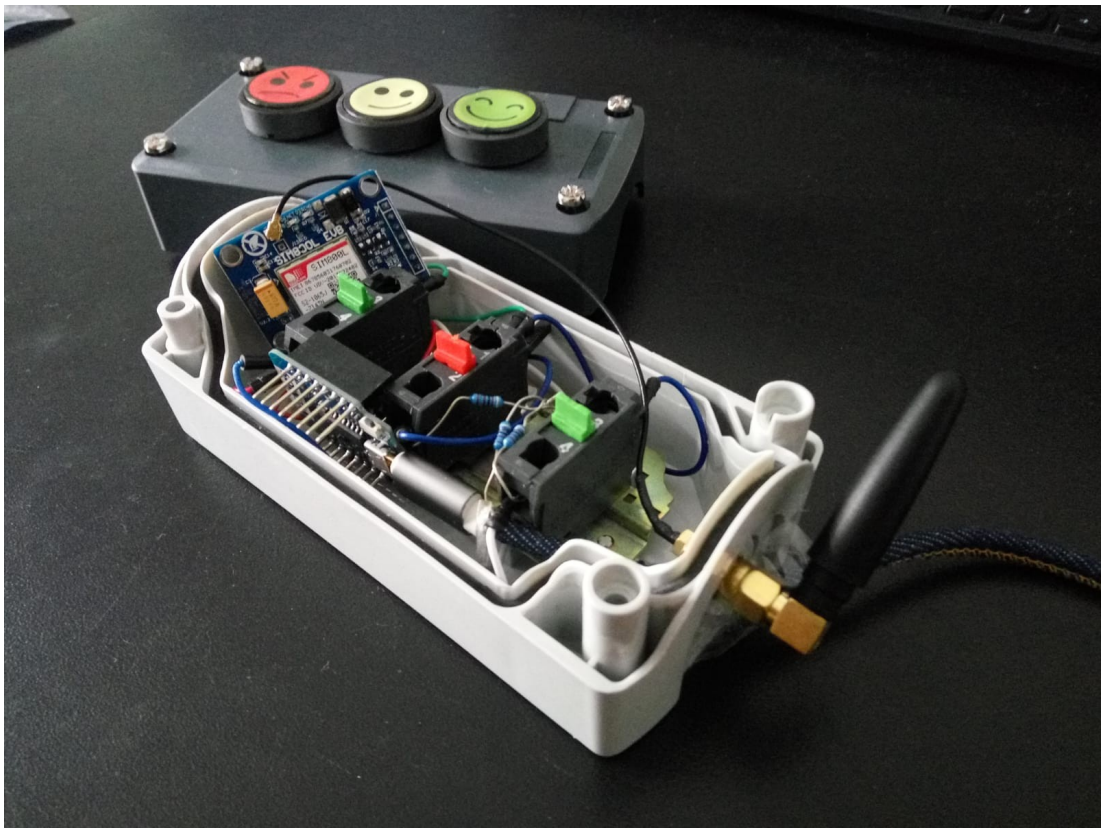

Ilustración 35: Prototipo GSM fabricado interior

El módulo SIM800L es el situado al final de la carcasa, está un poco elevado para que sea visible y se puede ver un cable negro conectándolo directamente a la antena. En la tapa con tornillos se encuentran las cabezas

de los tres pulsadores como en el caso anterior al igual que sus mecanismos resistencias y demás.

Es posible apreciar que la antena queda sujeta a la carcasa mediante arandelas interiores y exteriores, ajustadas para que no sea posible sacar la parte interior de la antena, por debajo podemos ver el cable USB que le proporcionara corriente.

En la siguiente imagen podemos verlo desde otro ángulo, y más de cerca:

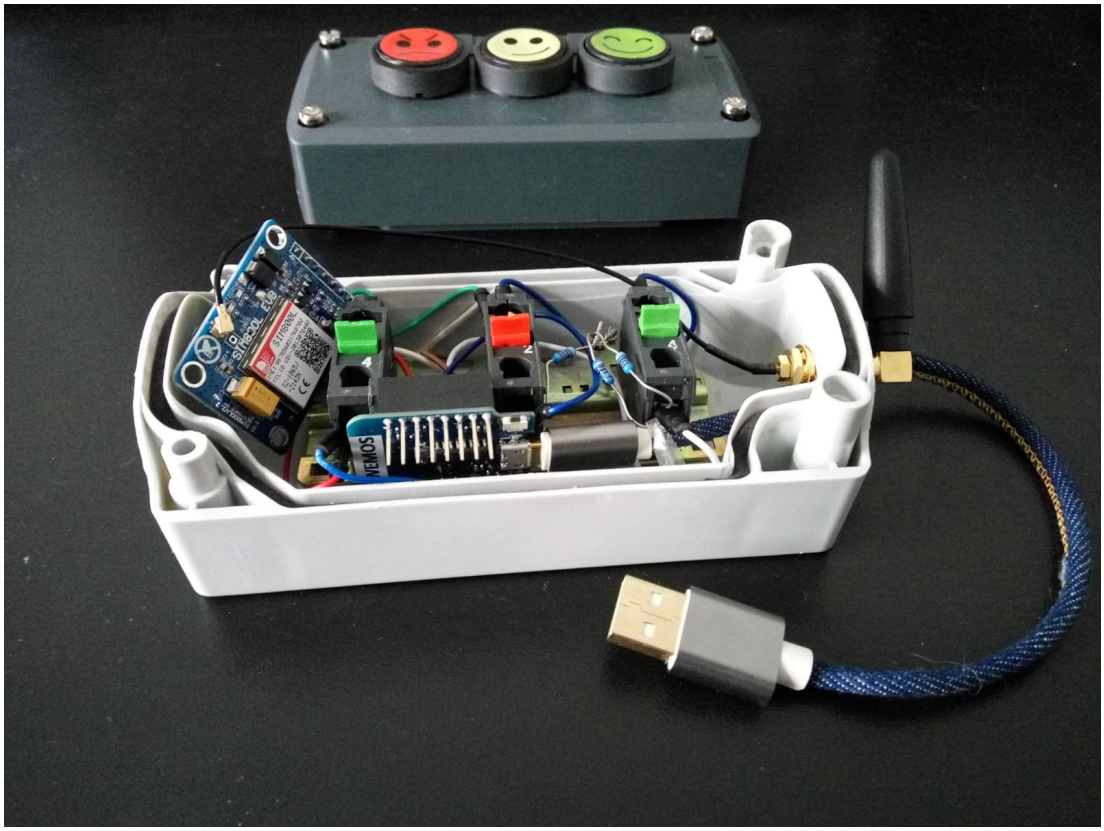

Ilustración 36: Prototipo GSM fabricado interior ángulo delantero

En la imagen anterior es posible apreciar mejor el cable USB conectado a la placa microcontroladora Wemos, la misma que fue usada en el prototipo anterior.

En esta disposición también conseguimos poder ejecutar las dos versiones del código del prototipo (la actual GSM o la anterior Wifi) El microcontrolador no utilizara el módulo GSM y sí su propia conexión Wifi si así lo hemos implementado. Cambiar a ejecutar una u otra versión del prototipo es tan

fácil como conectar a un PC al puerto USB y actualizar el código de la placa controladora, que vemos por detrás en las imágenes.

La siguiente imagen muestra el prototipo GSM/GRPS cerrado, con su antena y listo para usarse a falta de la pregunta impresa sobre el chasis agregado.

Ilustración 37: Prototipo GSM fabricado exterior

El coste medio de los materiales y componentes necesarios para la fabricación de este segundo prototipo ha sido de 31,08€ con IVA incluido. El resultado es la suma del anterior prototipo más el coste adicional del módulo SIM800L con conexiones y antena. Se puede comprobar en el anexo 1 presupuesto de prototipos.

Con la pregunta impresa queda como en la siguiente imagen:

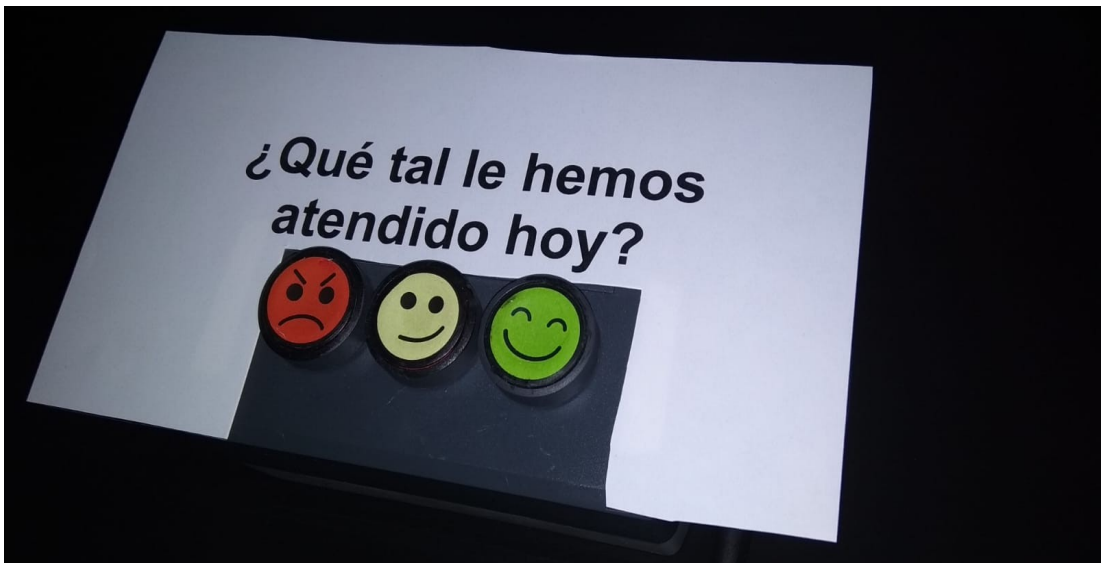

Ilustración 38: Prototipo GSM fabricado exterior con pregunta

6. RESULTADOS Y DISCUSIÓN

CAPÍTULO 6 RESULTADOS Y DISCUSIÓN

El planteamiento del trabajo mantiene dos indicadores de forma general desde el comienzo. El primero es el estudio y análisis del problema planteado. El segundo es el diseño e implementación de una solución valida y funcional.

El resultado de lo primero se puede ver representado en esta misma memoria de proyecto y todo lo agregado al conocimiento del problema y su contexto.

El resultado de lo segundo lo podemos encontrar en la plataforma web desarrollada y en los dispositivos propios creados.

El objetivo general de estudiar y desarrollar un sistema que permita al usuario de un servicio físico cualquiera, responder diferentes cuestiones '*just in the moment*', justo en el momento; y a los gestores analizar esos datos. Todo vía web y soportado sobre hardware se ha cumplido.

La plataforma web ha sido llevada hasta un entorno de producción real y funciona de forma correcta acompañada de unos dispositivos propios muy adaptables a futuros cambios y económicos, con precios medios que oscilan entre 20,30€ y 31,08€ actualmente.

Gracias al estudio de las tecnologías disponibles tanto para la plataforma como para los prototipos físicos hemos conseguido crear todos los elementos necesarios con tecnologías en crecimiento, soportadas por grandes compañías o grandes comunidades de desarrolladores y open source.

Las metodologías de desarrollo elegidas y seguidas nos han ayudado en la correcta implementación de la plataforma web y el rápido desarrollo de los prototipos.

Para validar la utilidad del sistema hemos realizado una prueba con una pequeña muestra de personas. En ella hemos visto que no tienen problemas o dificultades en el uso de la plataforma, al ver datos en las gráficas o en el uso de los dispositivos para responder a las cuestiones.

Con todo lo anterior damos por concluida la interpretación de los resultados.

7. CONCLUSIONES

CAPÍTULO 7 CONCLUSIONES

Planteamos que el desarrollo del proyecto ha sido un éxito. Hemos cumplido con el objetivo de estudiar y desarrollar un sistema para responder diferentes cuestiones '*just in the moment*' de forma óptima.

Entendemos que la extensión de la solución planteada en este trabajo fin de grado es amplia, pero gracias a ello podemos cumplir con los objetivos relativos a la flexibilidad del sistema y su facilidad de uso. Incluyendo dos formas de uso, tanto para dispositivos de proveedores como propios.

El plantear desde el inicio la necesidad de conocimiento agregado sobre el contexto del problema nos ha sido de gran ayuda para centrar el camino del proyecto y con ello el alcance definido se ha podido completar.

Después del análisis y dentro de los entornos de desarrollo planteados hemos conseguido utilizar conjuntos de tecnologías muy óptimos en cuanto a costes tanto de recursos máquina, electrónica o tiempos de desarrollo.

La finalidad del desarrollo cumple con lo esperado y es útil para el usuario según las pruebas realizadas sobre la plataforma web que se encuentra puesta en marcha en un entorno de producción y con los diferentes dispositivos.

ANEXOS

ANEXOS

Anexo 1. Presupuestos de prototipos

En este anexo vamos a realizar una estimación del precio de los materiales y componentes necesarios para la fabricación de los dos prototipos.

El prototipo Wifi se compone de:

- Caja de Control con 3 pulsadores momentáneos – BeMatik entre 10€ y 15€ /unidad dependiendo momento de compra, proveedor y cantidades.
- Wemos D1 Mini ESP8266 entre 4€ y 8€ /unidad dependiendo momento de compra, proveedor, si tiene pines soldados y cantidades.
- Tres resistencias, cables de puente, cable USB si no se incluye con el Wemos D1 y pegatinas para preguntas (caras y/o texto) entre 1,20€ y 2,40€ dependiendo momento de compra, proveedor, calidades y cantidades.

En total la aproximación ronda en el intervalo de 15,20€ a 25,40€, dando un precio medio aproximado de 20,30€ al no tener que sumar gastos de envío por incluirlos de forma gratuita la mayoría de los proveedores. Precios con IVA incluido.

El prototipo GSM se compone de:

- Todos los componentes anteriores con precio aproximado entre 15,20€ y 25,40€
- Módulo pequeño SIM800L GSM/GPRS TTL con antena entre 8,40€ y 12,55€ /unidad dependiendo momento de compra, proveedor, si tiene pines soldados y cantidades.
- Cables de puente adicionales entre 0,20€ y 0,40€ dependiendo momento de compra, proveedor y cantidades totales.

En total la aproximación para el segundo prototipo ronda en el intervalo de 23,80€ a 38,35€, dando un precio medio aproximado de 31,08€ al no tener que sumar gastos de envío por incluirlos de forma gratuita la mayoría de los proveedores. Precios con IVA incluido.

También sería necesario el uso de una tarjeta SIM de algún proveedor de telecomunicaciones, pero actualmente en 2018 podemos encontrar varios de ellos con tarjetas SIM gratuitas con coste por uso de datos. Nuestros prototipos tienen un consumo mínimo de datos por tanto los pocos céntimos que pueden cobrarnos por el servicio vamos a despreciarlo.

Tiendas de referencia y consultadas para el cálculo a 16 de mayo de 2018:

https://www.amazon.es

https://www.cablematic.es

https://es.aliexpress.com

https://www.banggood.com

Anexo 2. Manual de usuario

Veremos todas las opciones para los usuarios autenticados. Los no autenticados o usuarios externos tiene dos vías de uso: la visual o el uso desde la API mediante dispositivos físicos validados.

Las opciones para usuarios gestores son: Dispositivos, Cuestionarios, Pag de cuestionarios, Pag de Gracias, y Gráficas. La adicionales para administradores: Respuestas, Opciones Visuales, Clientes y Usuarios.

Los usuarios comienzan en una página de login como la siguiente:

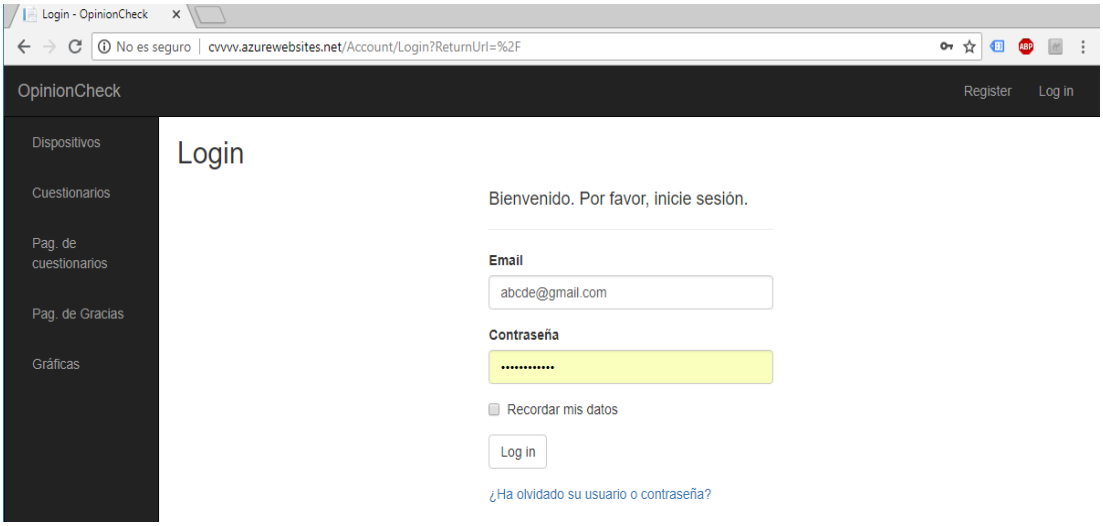

Ilustración 39: Manual-Usuario Visión usuarios login

Al entrar ven lo siguiente si son gestores, o sea no administradores.

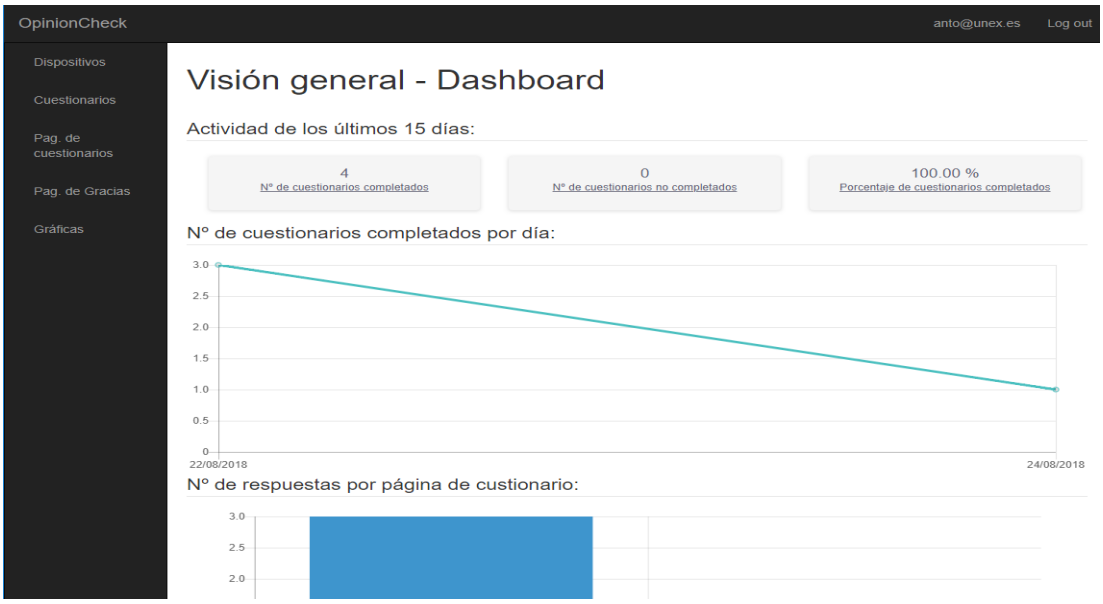

Ilustración 40: Manual-Usuario Visión usuarios no administradores

O si son administradores lo siguiente.

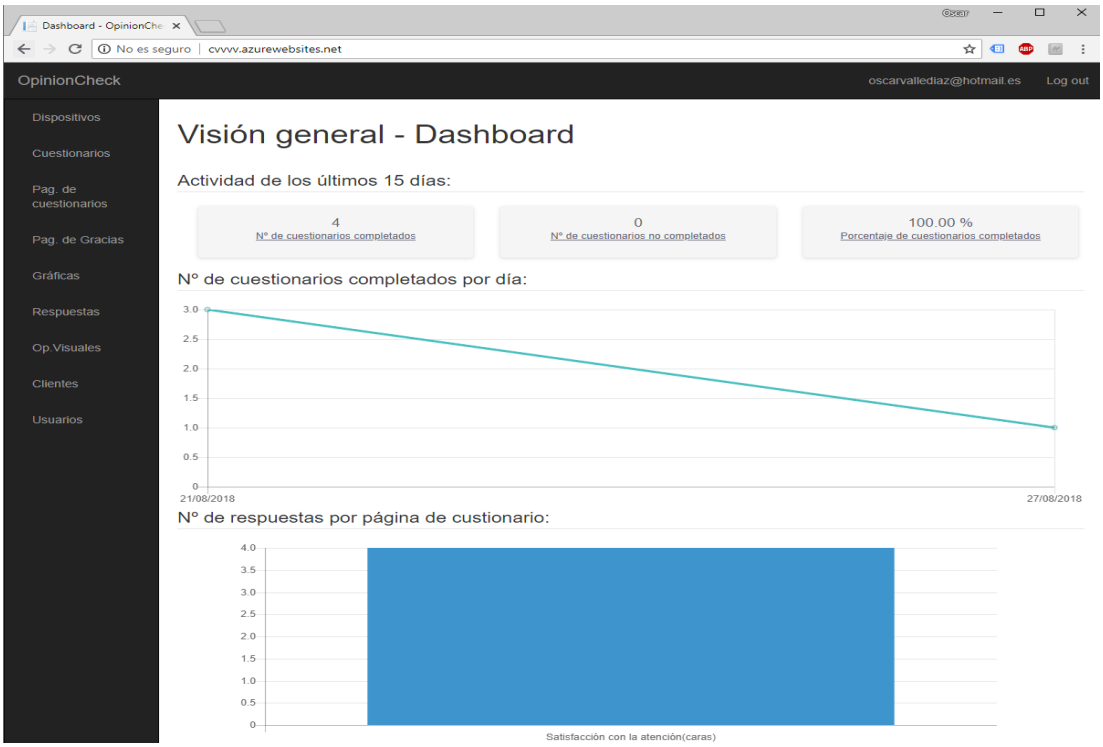

Ilustración 41: Manual-Usuario Visión usuarios administradores

Esto es una visión general del número de cuestionarios respondidos en los últimos 15 días.

A continuación, vamos a ver opción a opción comenzando por las de los usuarios gestores.

Dispositivos

Ilustración 42: Manual- Usuario Lista de dispositivos

Como podemos observar, la anterior imagen muestra la lista de dispositivos, a la cual se accede desde la opción Dispositivos del menú lateral izquierdo.

Nos encontramos con 4 opciones: Crear uno nuevo, editar uno existente, verlo en detalle, borrarlo o probarlo.

Dispositivos - Crear uno nuevo

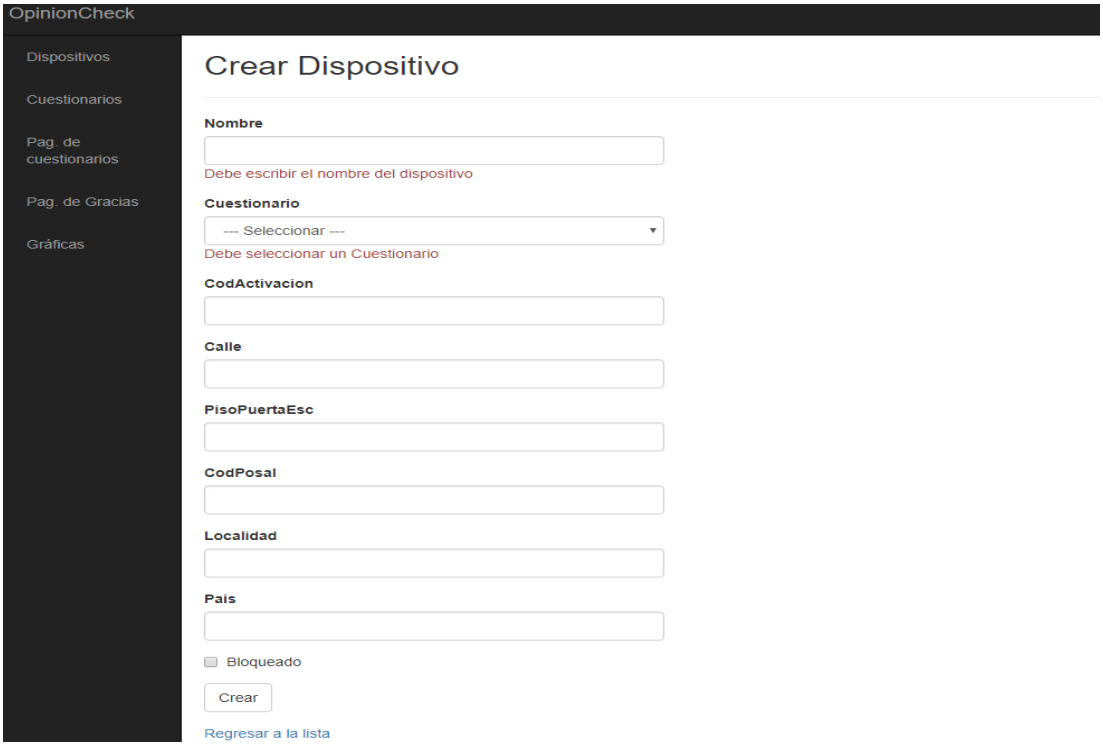

Ilustración 43: Manual-Usuario Dispositivos crear nuevo

Para crear un nuevo dispositivo debemos informar como mínimo del nombre y el cuestionario que mostrará este dispositivo. Como podemos ver el manual empieza comentando el elemento más fácil de entender, pero realmente necesitamos primero crear los elementos pequeños como las páginas de cuestionario o gracias, después crear los cuestionarios que las agrupan y por último dar de alta el dispositivo que lo mostrara o enlazara.

Dispositivos – Editar existente

Misma pantalla que la anterior, pero nos muestra los datos cargados del dispositivo.

Dispositivos – Ver en detalle

En el detalle del dispositivo podemos ver la url que utilizará si lo queremos incorporar de forma visual, podemos verlo y probarlo en modo test o descargar los datos de los cuestionarios respondidos en él en formato CSV.

Ilustración 44: Manual-Usuario Ver dispositivo en detalle

Dispositivos - Borrar

Elimina el dispositivo, pidiendo confirmación.

Dispositivos – Probar

Cuando probamos el formato visual vertido por el dispositivo vemos la primera página del cuestionario para poder responderla.

Esta prueba en modo test muestra abajo del todo de la pantalla el mensaje de aviso “MODO PRUEBA (no se guardarán las respuestas seleccionadas” para informarnos de que no se guardarán respuestas. Si copiamos la URL que podemos ver en el detalle del dispositivo entonces estamos accediendo sin modo prueba y se guardarán los datos, al igual que si lo hacemos desde la API, la API siempre guarda los datos si la entrada es correcta.

El cuestionario se autoreinicia después de mostrar la página final de agradecimiento. La siguiente imagen nos muestra la prueba.

Ilustración 45: Manual-Usuario Probar dispositivo

Cuestionarios

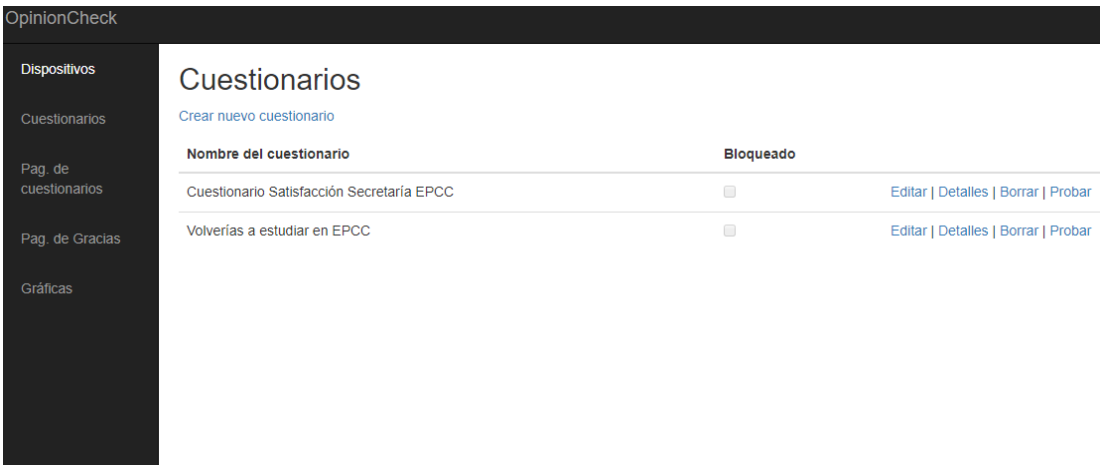

Ilustración 46: Manual- Usuario Lista de cuestionarios

Como podemos observar la anterior imagen muestra la lista de cuestionarios la cual se accede desde la opción Cuestionarios del menú lateral izquierdo.

Nos encontramos con 4 opciones: Crear uno nuevo, editar uno existente, verlo en detalle, borrarlo o probarlo.

Cuestionarios - Crear uno nuevo

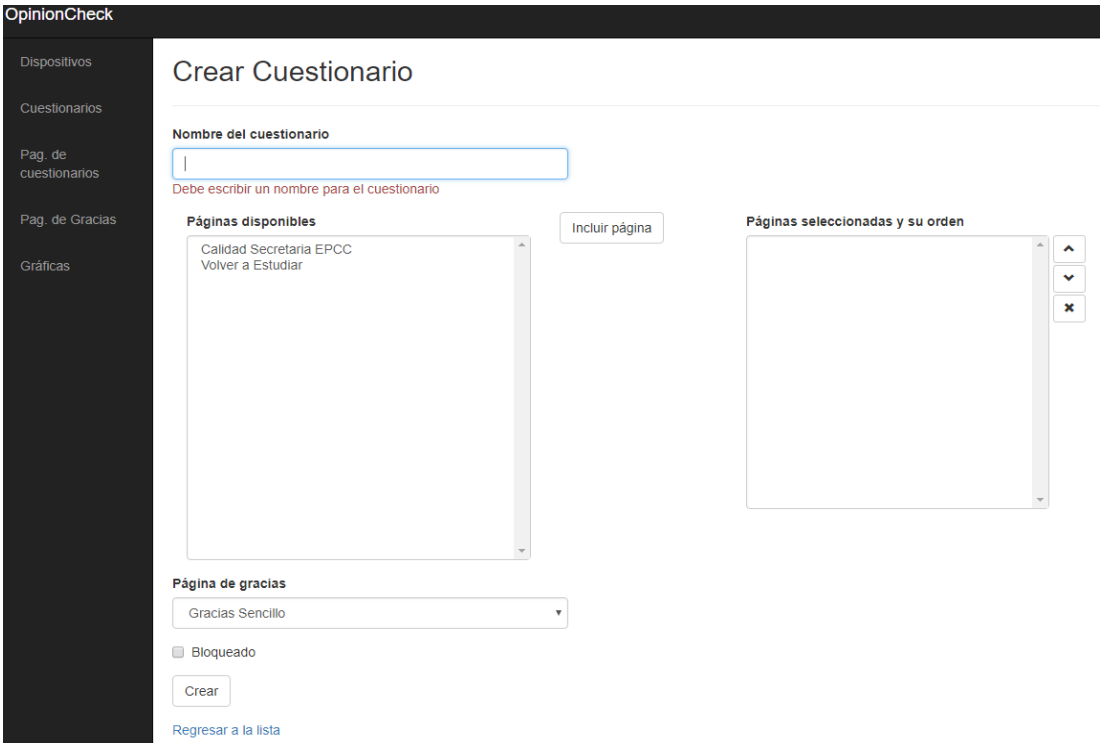

Ilustración 47: Manual-Usuario Cuestionario crear nuevo

Para crear un nuevo cuestionario debemos informar como mínimo del nombre y la página de agradecimiento asociada, para que sea útil debemos pasar de disponibles a seleccionadas las páginas de cuestionario.

Cuestionarios – Editar existente

Misma pantalla que la anterior, pero nos muestra los datos cargados del cuestionario para ser editado.

Cuestionarios – Ver en detalle

En el detalle del cuestionario vemos los datos básicos como el nombre o si está bloqueado, pero sin más funciones.

Cuestionarios - Borrar

Elimina el cuestionario, pidiendo confirmación.

Cuestionarios – Probar

Cuando probamos el formato visual de un cuestionario vemos la primera página del cuestionario para poder responderla.

Esta prueba en modo test muestra abajo del todo de la pantalla el mensaje de aviso “MODO PRUEBA (no se guardarán las respuestas seleccionadas” para informarnos de que no se guardarán respuestas. Es este caso se bloquea por defecto siempre para que no se guarden los datos nunca, ya que no se debe usar un cuestionario no asociado a un dispositivo.

La siguiente imagen nos muestra la prueba.

Ilustración 48: Manual-Usuario Probar cuestionario

Pag. De Cuestionarios

Como podemos observar en la siguiente imagen vemos la lista de páginas de cuestionario la cual se accede desde la opción Pag.de cuestinarios del menú lateral izquierdo.

Ilustración 49: Manual- Usuario Lista de Pag. De Cuestionarios

Nos encontramos con 4 opciones: Crear una nueva, editar una existente, verlo en detalle, borrarla o probarla.

Pag. De Cuestionarios - Crear una nueva

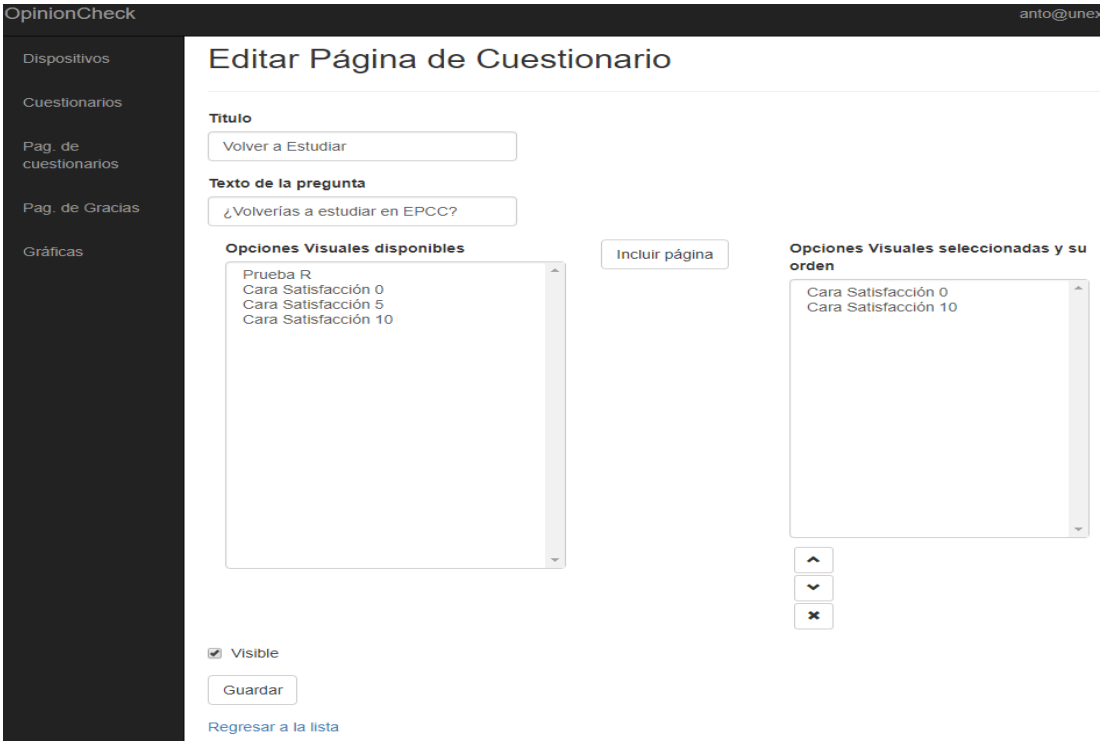

Ilustración 50: Manual-Usuario Pag. De Cuestionario crear nueva

Para crear una nueva página de cuestionario debemos informar como mínimo del título, el texto de la pregunta, y para que sea útil debemos pasar de disponibles a seleccionadas las opciones visuales.

Pag. De Cuestionarios – Editar existente

Misma pantalla que la anterior, pero nos muestra los datos cargados de la página de cuestionario.

Pag. De Cuestionarios – Ver en detalle

En el detalle de la página de cuestionario vemos los datos básicos como el título, el texto de la pregunta o si es visible, pero sin más funciones.

Pag. De Cuestionarios - Borrar

Elimina la página de cuestionario, pidiendo confirmación.

Pag. De Cuestionarios – Probar

Cuando probamos el formato visual de una página de cuestionario la vemos para poder responderla en modo test. Se muestra abajo del todo de la pantalla el mensaje de aviso “MODO PRUEBA (no se guardarán las respuestas seleccionadas”. Es este caso se bloquea por defecto para que no se guarden los datos ya que no se debe usar una página de cuestionario de forma independiente.

La siguiente imagen nos muestra la prueba.

Ilustración 51: Manual-Usuario Probar Pad. De Cuestionario

Pag. De Agradecimiento

Como podemos observar en la siguiente imagen vemos la lista de páginas de agradecimiento la cual se accede desde la opción Pag.de Gracias del menú lateral izquierdo.

Nos encontramos con 4 opciones: Crear una nueva, editar uno existente, verlo en detalle, borrarla o probarla.

Ilustración 52: Manual- Usuario Lista de Pag. De Agradecimiento

Pag. De Agradecimiento - Crear una nueva

Ilustración 53: Manual-Usuario Pag. De Agradecimiento crear nueva

Para crear una nueva página de agradecimiento debemos informar como mínimo del título.

Pag. De Agradecimiento – Editar existente

Misma pantalla que la anterior, pero nos muestra los datos cargados de la página de agradecimientos para ser editados.

Pag. De Agradecimiento – Ver en detalle

En el detalle de la página de agradecimiento vemos los datos básicos como el título, el texto principal, secundario, interior o si es visible, pero sin más funciones.

Pag. De Agradecimiento - Borrar

Elimina la página de agradecimiento, pidiendo confirmación.

Pag. De Agradecimiento – Probar

Cuando probamos el formato visual de una página de agradecimiento vemos la siguiente pantalla.

Gracias por sus respuestas

Estamos encantados de recibir su respuesta, nos ayudara a mejorar.

Ilustración 54: Manual-Usuario Probar Pad. De Agradecimiento

Graficas

Cualquier usuario puede consultar las respuestas a las páginas de cuestionarios completados con éxito y filtrar entre fechas, por el dispositivo en el que se grabó a que cuestionario pertenece y la página de cuestionario a ver.

En la siguiente imagen vemos la opción de Gráficas:

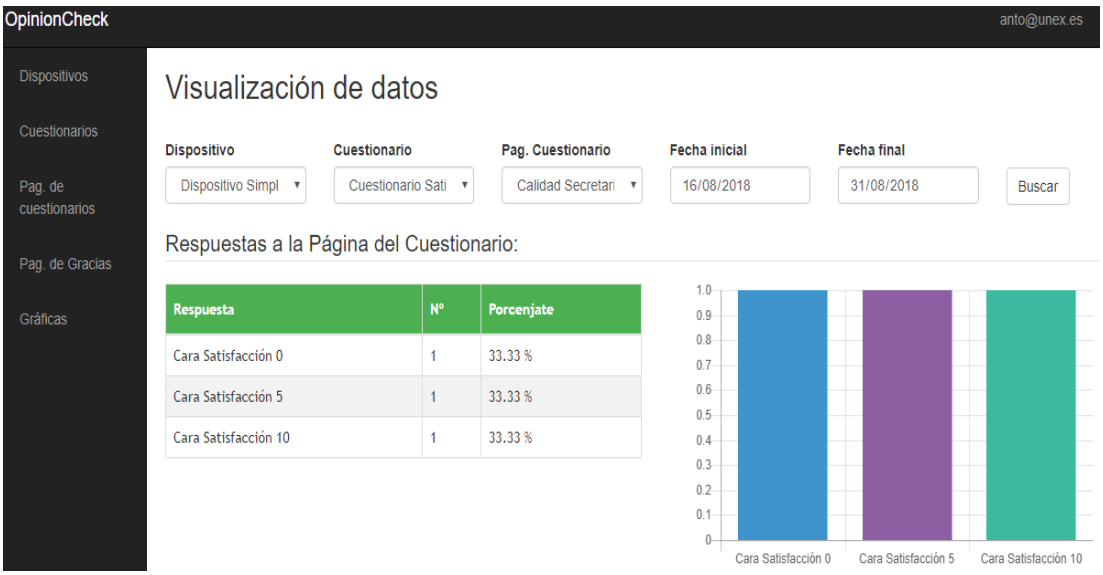

Ilustración 55: Manual-Usuario Gráficas

Respuestas

Sólo los usuarios administradores pueden ver las últimas respuestas captadas por el sistema o crearlas manualmente para hacer pruebas o comprobar errores.

Es una funcionalidad que se ha agregado sólo para eso, para probar sin tener que ir directamente a base de datos. Podemos crear nuevas respuestas, editar las existentes o borrarlas.

En la siguiente imagen vemos la lista de respuestas con las opciones comentadas, se accede desde la opción Respuestas del menú lateral izquierdo.

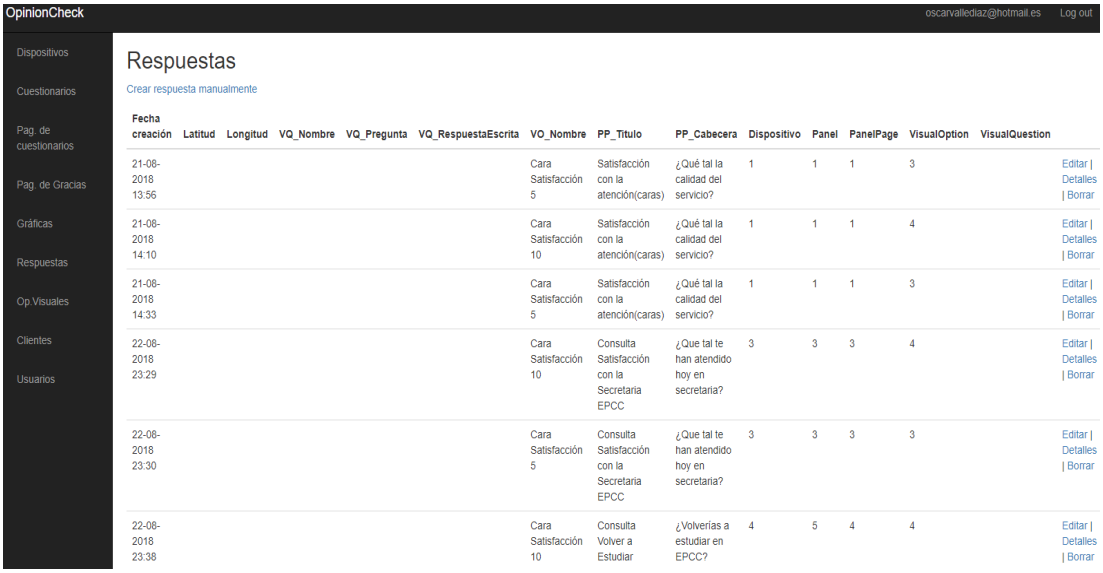

Ilustración 56: Manual-Usuario Respuestas

Opciones Visuales

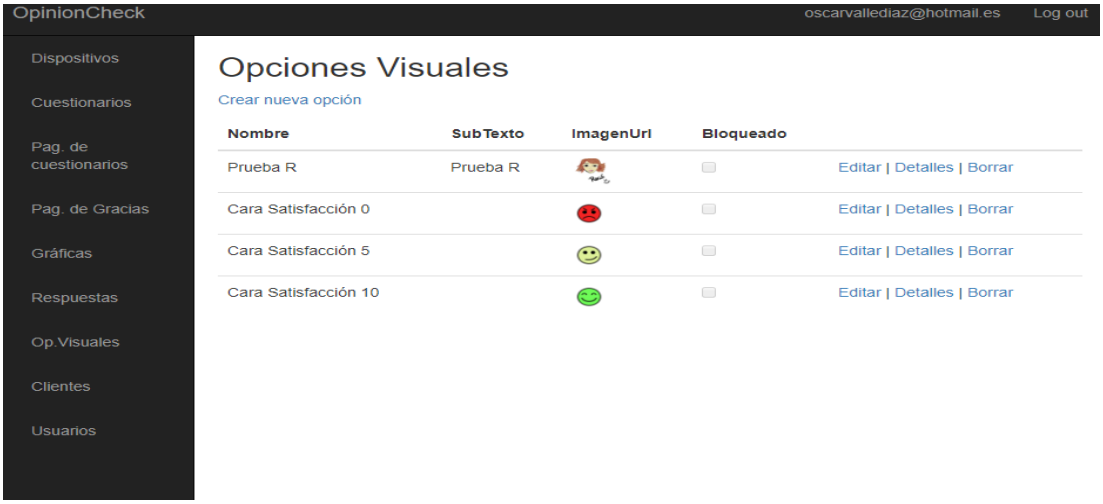

Ilustración 57: Manual- Usuario Lista de opciones visuales

Como podemos observar la anterior imagen muestra la lista de opciones visuales que un administrador ha creado en el sistema. A la página se accede desde la opción Op. Visuales del menú lateral izquierdo.

Nos encontramos con 4 opciones: Crear una nueva, editar una existente, verlo en detalle o borrarla.

Opciones Visuales - Crear una nueva

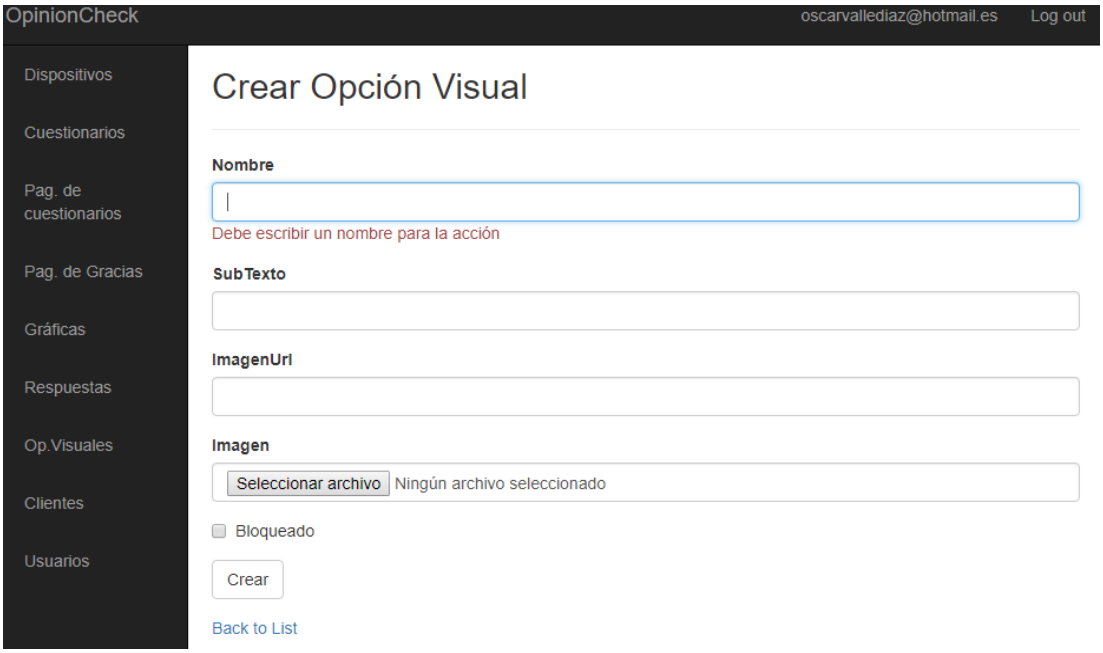

Ilustración 58: Manual-Usuario Opción visual crear nueva

Para crear una nueva opción visual debemos informar como mínimo del nombre y subir la imagen asociada.

Opciones Visuales – Editar existente

Misma pantalla que la anterior, pero nos muestra los datos cargados.

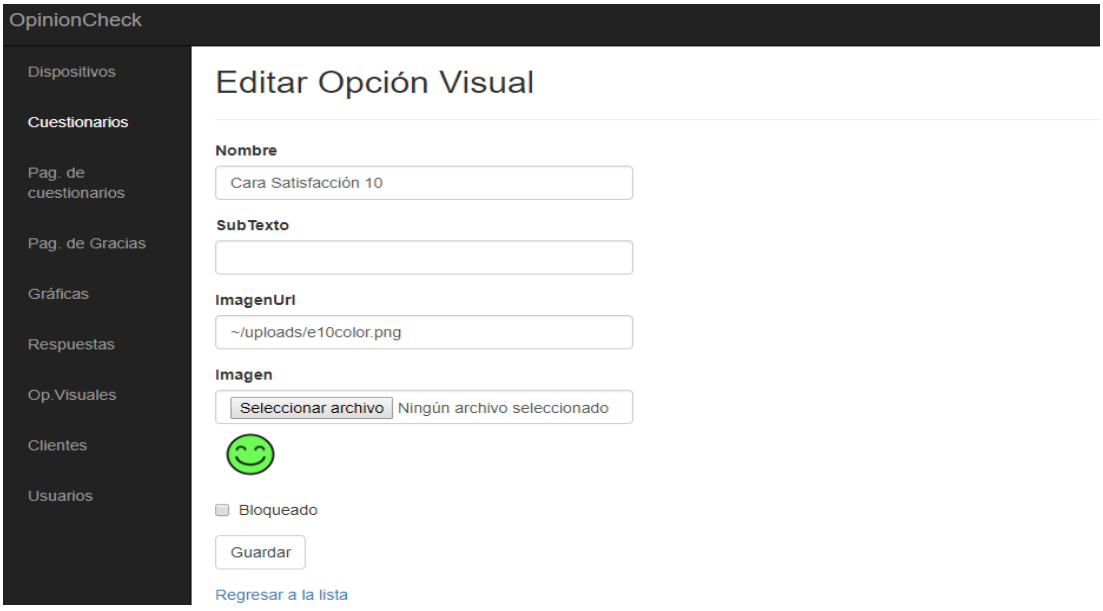

Ilustración 59: Manual-Usuario Editar opción visual

Opciones Visuales – Ver en detalle

En el detalle de una opción visual simplemente se nos muestran sus datos e imagen asociada.

Ilustración 60: Manual-Usuario Ver opción visual detalle

Opciones Visuales - Borrar

Elimina la opción visual, pidiendo confirmación.

Clientes

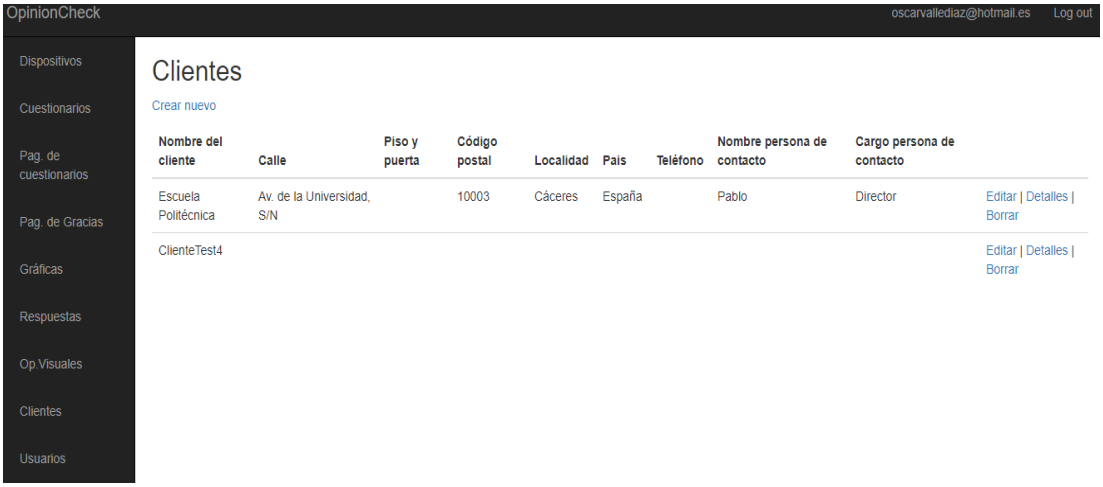

Ilustración 61: Manual- Usuario Lista de clientes

Como podemos observar la anterior imagen muestra la lista de clientes que un administrador ha creado en el sistema. A la página se accede desde la opción Clientes del menú lateral izquierdo.

Nos encontramos con 4 opciones: Crear uno nuevo, editar uno existente, verlo en detalle o borrarlo.

Clientes - Crear uno nuevo

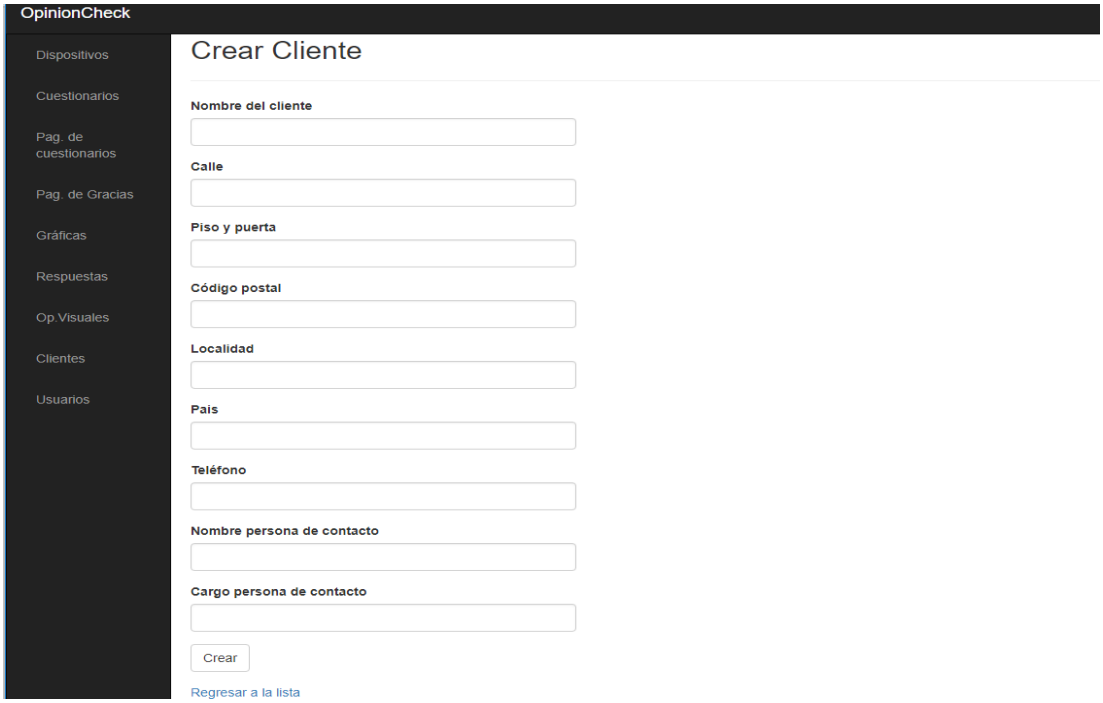

Ilustración 62: Manual-Usuario Cliente crear nuevo

Para crear un nuevo cliente debemos informar como mínimo del nombre.

Clientes – Editar existente

Misma pantalla que la anterior, pero nos muestra los datos cargados.

Clientes – Ver en detalle

En el detalle de un cliente simplemente se nos muestran sus datos sin ninguna función adicional.

Clientes - Borrar

Elimina el cliente, pidiendo confirmación.

Usuarios

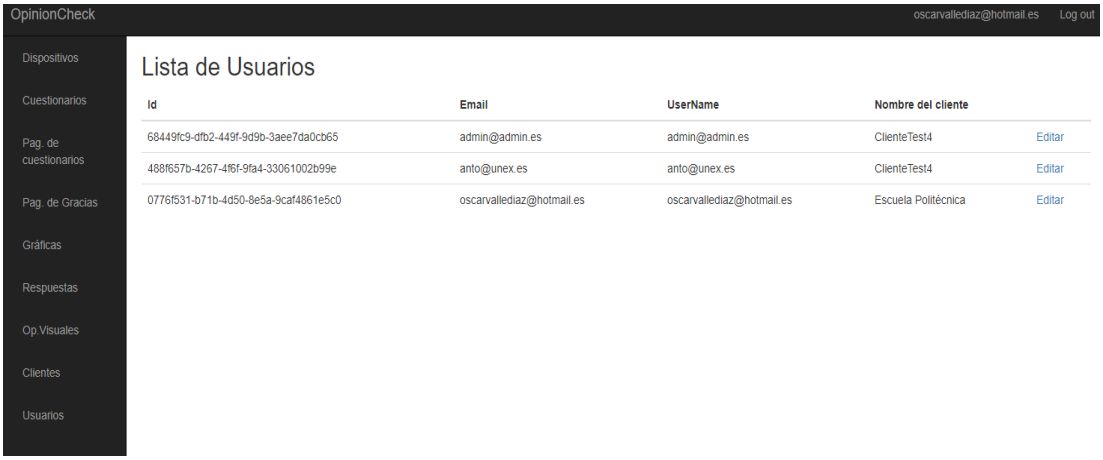

Ilustración 63: Manual- Usuario Lista de usuarios

Como podemos observar la anterior imagen muestra la lista de usuarios registrados. La página se accede desde la opción Usuarios del menú lateral.

Nos encontramos con una sola opción, editar un usuario. Aunque por otro lado desde la url ~/Account/Register podemos registrar nuevos usuarios.

Usurios - Registrar

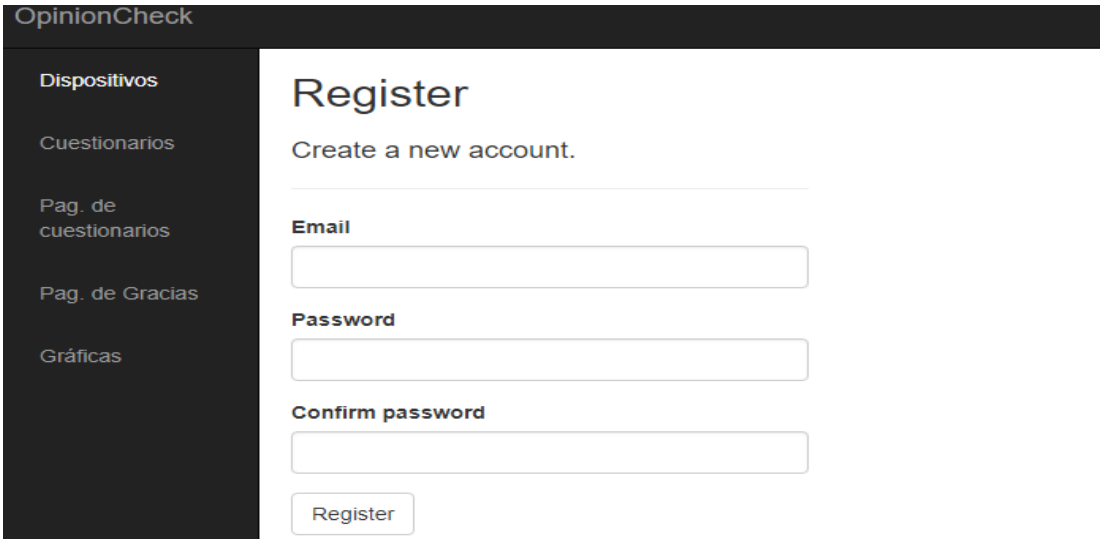

Ilustración 64: Manual-Usuario Usuarios registrar

Usuarios – Editar existente

En la página podemos seleccionar que cliente tiene asociado el usuario y si es administrador o no, si tiene ese rol.

Ilustración 65: Manual-Usuario Editar usuario

Anexo 3. Manual del programador

La arquitectura de la aplicación al basarse en MVC y usar un ORM genera muchas clases, carpetas y archivos diferentes. Pese a esa gran cantidad de elementos es una ventaja el encontrar separada de forma lógica cada capa y contexto del desarrollo.

Vamos a dividir en manual en cuatro áreas para comentarlas de forma separada y en el orden en el que deberían editarse y/o crearse: el modelo, la autogeneración, los controladores y las vistas.

El modelo

El modelo es la representación en clases de la lógica de datos. Al crear estás clases la herramienta Entity Framework Core como ORM genera migraciones que a su vez generan actualizaciones en la estructura de la base de datos. Por tanto, conseguimos desde el modelo crear toda la estructura de la base de datos.

En nuestro proyecto el modelo no son sólo clases planas para estructurar datos, sino que las clases que componen el modelo cuentan con una serie de iniciadores como vemos en el siguiente código:

```
public class PageThank
{
    public int Id { get; set; }
    public int ClienteId { get; set; }
    [DefaultValue(1)]
    public bool Visible { get; set; }
    [StringLength(255)]
    [Display(Name = "Título de la página")]
    [Required(ErrorMessage = "Debe escribir un título para la página")]
    [DefaultValue("")]
    public string Titulo { get; set; }
    [StringLength(511)]
    [Display(Name = "Texto principal")]
    public string Cabecera { get; set; }
    [StringLength(1023)]
```

```
        [Display(Name = "Texto secundario")]
        public string Cuerpo { get; set; }
        [StringLength(1023)]
        [Display(Name = "Texto inferior pequeño")]
        public string SubCuerpo { get; set; }
        [StringLength(8191)]
        public string CSS { get; set; }
        public int? MedioId { get; set; }
        public Medio Medio { get; set; }

        public PageThank(int _ClienteId)
        {
            ClienteId = _ClienteId;
            Visible = true;
        }

        public PageThank()
        {
            Visible = true;
        }
    }
```

Es posible ver varios de ellos como DefaultValue, que nos indica un valor por defecto, StringLength que limita en base de datos el tamaño máximo de los datos de tipo String, Required y su mensaje que trabaja en base de datos como not null y en las vistas generadas como campo obligatorio y mensaje de salida o display como mensaje de cabecera de los elementos para la autogeneración.

Tenemos tantos modelos como entidades, esto ya se ha explicado en el análisis cuando mostramos los diagramas generados de la base de datos, por puntualizar los más importantes son: Cliente, Dispositivo, Panel, PanelPage, PageThank, VisualOption, VisualQuestion, RespuestaSimple. Para enlazar algunos de ellos por necesidades como la de incluir en un PanelPage (página de cuestionario) una serie de VisualOption tenemos PanelPageVisualOption, PanelPageVisualQuestion o PanelPanelPage para enlazar varios PanelPage a un Panel.

Por ejemplo, el código de este último, el PanelPanelPage es el siguiente:

```
public class PanelPanelPage
{
    public int Id { get; set; }
    public int PanelId { get; set; }
    public Panel Panel { get; set; }
    public int? PanelPageId { get; set; }
    public PanelPage PanelPage { get; set; }
    public int Orden { get; set; }

    public PanelPanelPage()
    {
    }
}
```

En este caso podemos ver que la clase tiene enlace con las dos clases que comentamos, esto genera en base de datos una tabla intermedia entre Panel y PanelPage.

Es interesante comentar una buena práctica, con poner “public int PanelId {get;set;}” el sistema por sí mismo ya es capaz de enlazar con el Panel, pero al incluir una referencia implícita como “public Panel panel {get;set;}” a la hora de usarlo con LINQ en los controladores nos va a ser mucho más simple pues hace que la instancia de la clase cuando la llevamos a memoria contenga la instancia de la otra clase relacionada de esa forma, si solo tenemos el Id tendremos que ir a buscar por él para obtener el objeto en cuestión.

La autogeneración

Con los modelos creados ya podemos crear los controladores y las vistas, pero en nuestro proyecto no vamos a hacerlo manualmente, al tener esos modelos con tantas especificaciones podemos autogenerarlos y ahorrar mucho tiempo.

Para generar un controlador y sus vistas asociadas, si no queremos vistas como tal también podemos hacerlo en forma de API REST, vamos a pulsa con el botón derecho en la carpeta Controllers de nuestro proyecto abierto con Visual Studio 2017 como se ve en la siguiente imagen:

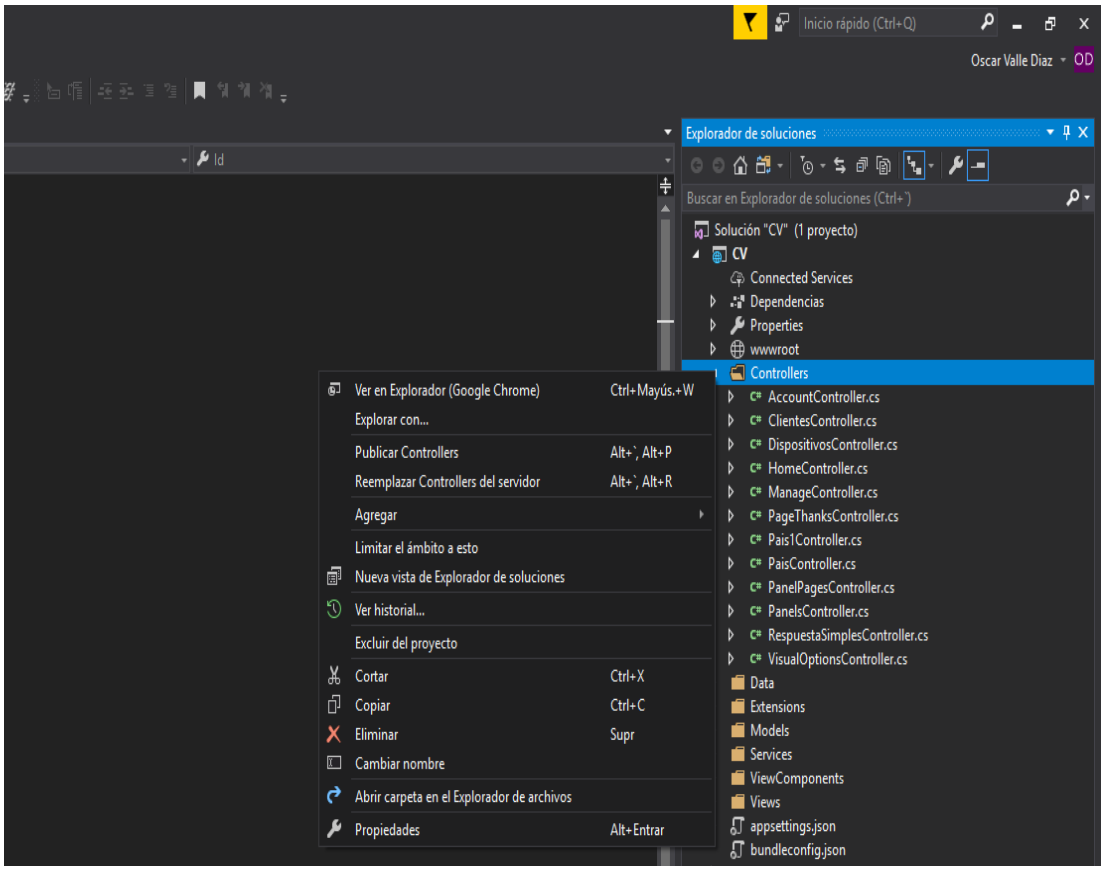

Ilustración 66: Manual-Programador Autogenerar controlador y vistas 1

Después pulsamos en agregar y en controlador como podemos ver :

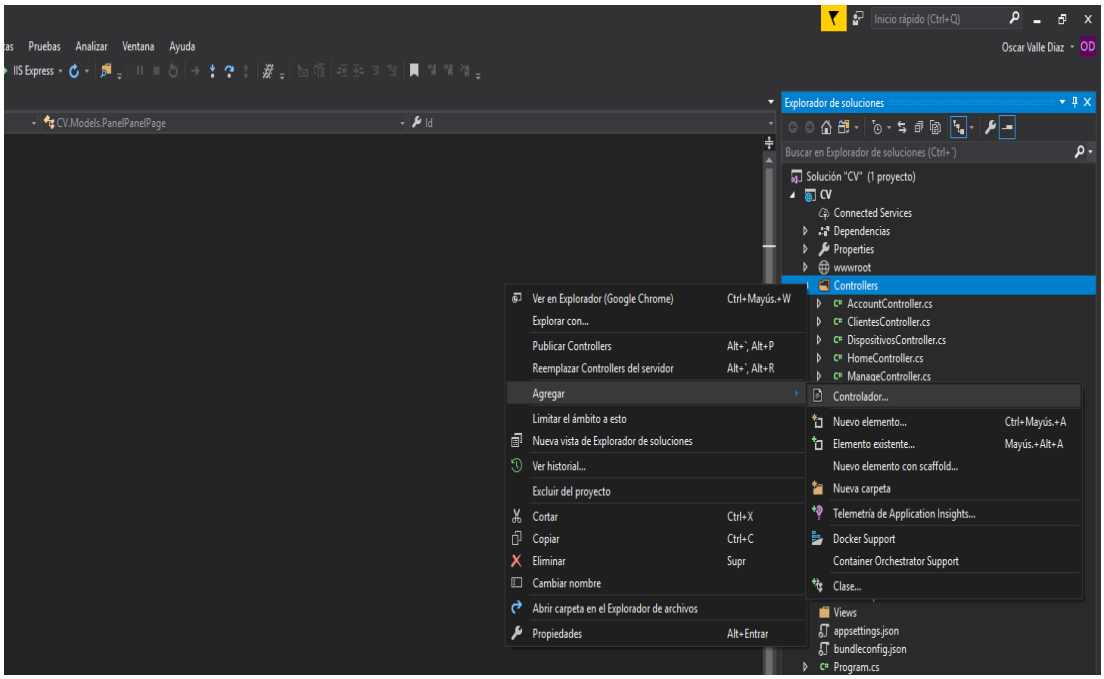

Ilustración 67: Manual-Programador Autogenerar controlador y vistas 2

Lo siguiente a seleccionar para crear el controlador con las vistas asociadas es la opción Controlador de MVC con vistas que usan Entity Framework:

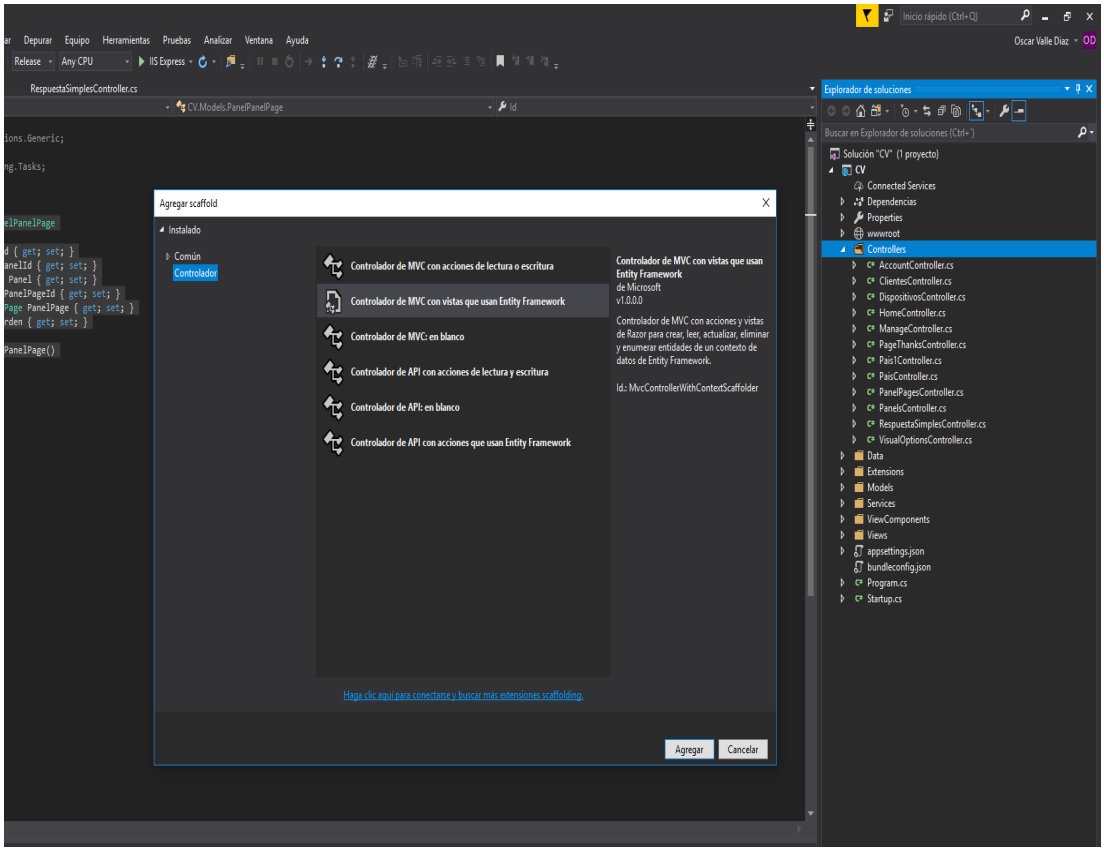

Ilustración 68: Manual-Programador Autogenerar controlador y vistas 3

En el paso anterior podíamos haber creado un controlador de API REST que nos escribiría automáticamente toda la API del modelo que seleccionemos en el siguiente paso.

En este caso vamos a seleccionar la clase modelo Dispositivo (CV.Models) contenido en el ApplicationDbContext (CV.Data) y con las opciones marcadas ya por defecto de Generar vistas y Usar página de diseño. En la siguiente imagen podemos ver esa pantalla.

Al pulsar en agregar conseguimos que se nos genere en esa carpeta Controllers el controlador con todos los métodos necesarios para un CRUD del modelo y las vistas enlazadas a ese controlador en la carpeta Views y dentro de una subcarpeta con su propio nombre.

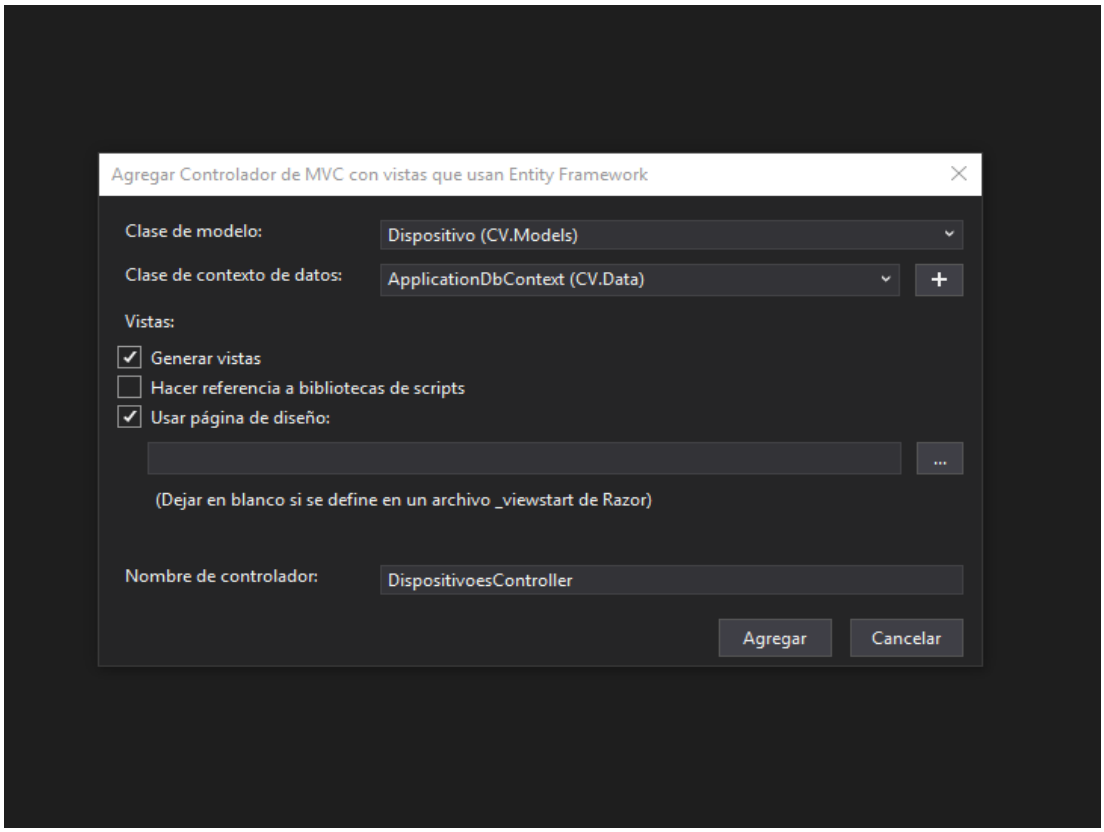

Ilustración 69: Manual-Programador Autogenerar controlador y vistas 4

Los controladores

Con los controladores autogenerados podemos editarlos para agregar llamadas especificas o adaptar alguna de sus características, como por ejemplo: en muchos de los controladores del proyecto se filtran datos si el usuario logueado es administrador, se hace como se puede ver en el siguiente código que es el método que sirve la vista Index de los Dispositivos, o sea una lista de los dispositivos.

```
// GET: Dispositivos
public async Task<IActionResult> Index()
{
ApplicationUser usuario = await _userManager.GetUserAsync(HttpContext.User);
 var roles = await _userManager.GetRolesAsync(usuario);

 ViewBag.isAdminHiddenTag = User.IsInRole("Admin");
  if (User.IsInRole("Admin"))
  {
      return View(await _context.Dispositivos.ToListAsync());
  }
```

```
        else
        {
          return View(await _context.Dispositivos.Where(x => x.ClienteId ==
usuario.ClienteId).ToListAsync());
        }
  }
```

Las vistas.

Las vistas básicas o de CRUD como una lista de elementos llamada Index, Create, Edit, Delete o Details han sido generadas, pero es posible que necesitemos crear una nueva. En el proyecto se ha realizado por ejemplo con View en PageThanksController, el código en el controlador es el siguiente:

```
        // GET: PageThanks/View/5
        [AllowAnonymous]
        public async Task<IActionResult> View(int? id, string urlRedireccion)
        {
            if (id == null)
            {
                return NotFound();
            }

            var pageThank = await _context.PageThank
                .Include(p => p.Medio)
                .SingleOrDefaultAsync(m => m.Id == id);
            if (pageThank == null)
            {
                return NotFound();
            }

            ViewBag.UrlRedireccion = urlRedireccion;

            return View(pageThank);
        }
```

Y la vista View.cshtml creada en la carpeta de vistas de ese modelo PageThank contiene el siguiente código:

```
@model CV.Models.PageThank
@{
    ViewData["Title"] = "Gracias";
}
<head>
    <meta http-equiv="Refresh" content="5;url=@ViewBag.UrlRedireccion" />
</head>

<style>
..
</style>

<div class="container rowCenter">
    <div class="row">
        <div class="col-md-12" style="padding-bottom: 1em;">
            <h1>@Html.DisplayFor(model => model.Cabecera)</h1>
        </div>
        <div class="col-md-12" style="padding-bottom: 1em;">
            <p>@Html.DisplayFor(model => model.Cuerpo)</p>
        </div>
    </div>
</div>
```

Podemos ver el paso del modelo en @model, el título que se le dará a la página cuando se genere desde el servidor al cliente “Gracias” y un container centrado con dos textos extraídos del modelo, el texto de cabecera y el del cuerpo.

REFERENCIAS BIBLIOGRÁFICAS

[1] GALLUP, G H. (1939) *The Gallup International Public Opinion Polls.* Greenwood Press, New York: Random House.

[2] JORGE ÁRIAS LÓPEZ Ciencias Sociales de Extremadura *la web de la Asociación de Ciencias Sociales de Extremadura (ACISE) ¿QUÉ SON LOS ESTUDIOS DE OPINIÓN?*.

https://sociologiaext.wordpress.com/2007/09/25/%C2%BFque-son-los-estudios-de-opinion-un-extracto-elaborado-por-jorge-arias-lopez/ [Consulta: 11 Abril 2018]

[3] RAE. *Ciencias sociales.*

http://dle.rae.es/srv/fetch?id=9AwuYaT [Consulta: 12 Abril 2018]

[4] INE *Instituto Nacional de Estadística.*

http://www.ine.es/ [Consulta: 12 Abril 2018]

[5] CIS *Centro de Investigaciones Sociológicas.*

http://www.cis.es/cis/opencms/ES/index.html [Consulta: 12 Abril 2018]

[6] RAE. *Encuesta.*

http://dle.rae.es/srv/search?m=30&w=encuesta [Consulta: 12 Abril 2018]

[7] RAE. *Indagar*.

http://dle.rae.es/?id=LLj8FxH [Consulta: 12 Abril 2018]

[8] ISO (2015) *Sistemas de gestión de la calidad Norma ISO 9001 Familia ISO 9000.*

https://www.iso.org/obp/ui/#iso:std:iso:9000:ed-4:v1:es [Consulta: 12 Abril 2018]

[9] G.VAVRA, T (2003) *Cómo medir la satisfacción del cliente según la ISO 9001:2000.* Segunda edición. Madrid: FC Editorial, Fundación Confemetal.

https://books.google.es/books?id=HGy1eJxZVJkC&lpg=PA14&ots=6eFZKR0Rgk&dq=encuestas%20iso

%209001&lr&hl=es&pg=PA149#v=onepage&q=encuestas%20iso%209001&f=false [Consulta: 12 Abril 2018]

[10] EQUIPO VÉRTICE. (2010) *Gestión de la calidad (ISO 9001/2008) en el comercio.* Málaga: Publicaciones Vértice.

https://books.google.es/books?id=ZJ9dhKW6xYUC&lpg=PA279&dq=encuestas%20iso%209001&hl=es&pg=PA279#v=onepage&q=encuestas%20iso%209001&f=false [Consulta: 15 Abril 2018]

[11] PROFESORADO UNIVERSIDAD ALCALÁ. *Temario. Tema 5 La encuesta estadística. Asignatura Sociología Económica*. Madrid: Universidad de Alcalá.

http://www3.uah.es/vicente_marban/ASIGNATURAS/SOCIOLOGIA%20ECONOMICA/TEMA%205/tema%205.pdf [Consulta: 15 Abril 2018]

[12] DÍAZ DE RADA,V. (2007) "Tipos de encuestas considerando la dimensión temporal." en *Papers: revista de sociología*. 2007 Núm.: 86 Educació i ocupació.

https://www.raco.cat/index.php/papers/article/view/81389 [Consulta: 16 Abril 2018]

[13] DÍAZ DE RADA, V. (2000) "Utilización de nuevas tecnologías para el proceso de 'recogida de datos' en la investigación social mediante encuesta." (2000) en *Reis: Revista Española de Investigaciones Sociológicas* No. 91 (Jul. - Sep., 2000), pp. 137-166.

https://www.jstor.org/stable/40184278 [Consulta: 17 Abril 2018]

[14] DÍAZ DE RADA, V. (2010) "Eficacia de las encuestas por Internet: un estudio preliminar." en *Revista Española de Sociología (RES)* Núm. 13 (2010).

https://recyt.fecyt.es/index.php/res/article/view/65165/39481 [Consulta: 18 Abril 2018]

[15] Parlamente Europeo. Régimen General de Protección de Datos (RGPD) Reglamento Europeo.

https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32016R0679 [Consulta: 25 Abril 2018]

[16] GIL CASTRO, R. (2004) "Estructura básica del proceso unificado de desarrollo de software". Universidad Icesi 2004.

https://www.researchgate.net/profile/Robin_Castro_Gil/publication/38319768_Estructura_basica_del_proceso_unificado_de_desarrollo_de_software/links/55f7247408ae07629dc03a74.pdf [Consulta: 25 Abril 2018]

[17] WIKIPEDIA. *Modelo de Prototipos.*

https://es.wikipedia.org/wiki/Modelo_de_prototipos [Consulta: 29 Abril 2018]

[18] INC. *Nearly 20 Years Later, SurveyMonkey Is Going Public.*

https://www.inc.com/business-insider/surveymonkey-files-to-go-public-ipo-svmk-inc.html [Consulta: 20 Junio 2018]

[19] WIKIPEDIA USA. *SurveyMonkey.*

https://en.wikipedia.org/wiki/SurveyMonkey [Consulta: 21 Junio 2018]

[20] GUBBI, J (2013) "Internet of Things (IoT): A vision, architectural elements, and future directions" en FUTURE GENERATION COMPUTER SYSTEM . Volume 29, Issue 7, September 2013, Pages 1645-1660.

https://www.sciencedirect.com/science/article/pii/S0167739X13000241 [Consulta: 21 Junio 2018]

[21] I-SCOOP. *Connectivity and networks for the Internet of Things data deluge*.

https://www.i-scoop.eu/internet-of-things-guide/connectivity-networks-fog-computing-internet-of-things/ [Consulta: 21 Junio 2018]

[22] WIKIQUOTE *Colección de citas, Henry Ford.*

https://es.wikiquote.org/wiki/Henry_Ford [Consulta: 2 Mayo 2018]

[23] OXFORD DICTIONARIES. *Dispositivo.*

https://es.oxforddictionaries.com/definicion/dispositivo [Consulta: 2 Mayo 2018]

[24] WEMOS. *Wemos D1 Mini board (4MB flash based on ESP-8266EX).*

https://wiki.wemos.cc/products:d1:d1_mini [Consulta: 3 Mayo 2018]

[25] ARDUINO *Arduino Uno.*

https://store.arduino.cc/arduino-uno-rev3 [Consulta: 3 Mayo 2018]

[26] ARDUINO *Arduino Nano.*

https://store.arduino.cc/arduino-nano

[27] FRITZING. *Open Source hardware software.*

http://fritzing.org/home/ [Consulta: 4 Mayo 2018]

[28] AVELECTRONICS *Módulo GSM/GPRS SIM800L.*

http://avelectronics.co/productos-2/modulo-gsmgprs-sim800l/ [Consulta: 2 Julio 2018]