Aplicación multicliente para la creación de portales cautivos y gestión de los datos capturados

Fabián Baena Tarriño

Resumen—Dado el uso creciente de tecnologías web mediante portales cautivos para ofrecer y proporcionar acceso a internet gracias a dispositivos físicos preparados para ello, nace la idea de este proyecto. Se basa en la mejora de las herramientas utilizadas por parte de la empresa Please Networks S.L. para generar estos portales cautivos y, además, una interfaz gráfica de usuario que aporta una visión global, mediante el uso de representaciones gráficas, de los datos generados por los dispositivos que permiten la conexión de los usuarios a la red a través del uso de los portales cautivos. El proyecto consta de dos partes que trataran de dar solución a la problemática de la empresa. Por una parte, se constituye de una aplicación que permitirá gestionar de manera global los diferentes portales cautivos, permitiendo su creación de manera simple y rápida, y apostando por herramientas y técnicas modernas de desarrollo del software. Por otro lado, se utilizará parte de la tecnología de la empresa que se comunicará con una aplicación, capaz de tratar los datos que se deseen, y mostrarlos de manera centralizada y totalmente personalizable a través de herramientas visuales. Este proyecto supone una importante y sustancial mejora en cuanto a nivel de detalle y calidad de software se refiere.

Palabras clave—Portales cautivos, Wi-Fi, dashboard, metodología ágil, interfaz, personalización, reutilización de código, laravel, angularJS, framework, Scrum, comunicación, PHP, API

Abstract—Given the increasing use of web technologies through captive portals in order to offer and provide access to internet thanks to physical devices prepared for it, the idea for this project was born. It is based on the improvement of the tools used by Please Networks S.L. company in order to generate these captive portals and, furthermore, a graphical user interface that provides a global view, using graphical representations, of the data generated by the devices that allow connecting the network thanks to the use of captive portals. The project consists of two parts which try to solve the problems of the company. On the one hand, it is an application that will allow global management of the different captive portals, allowing its creation in a simple and fast way, and betting on modern tools and techniques of software development. On the other hand, it will use part of the technology of the company which will communicate with an application, able to treat the data that is wanted, and display them in a centralized and fully customizable through different kind of visual tools. This project constitues an important and substantial improvement in level of detail and quality as das as software is concerned.

Index Terms— Captive portals, Wi-Fi, dashboard, agile methodology, interface, personalization, code reuse, laravel, angularJS, framework, Scrum, communication, PHP, API

1 Introducción

La gran influencia de internet dentro del día a día de las personas, impulsada por el uso cada vez más generalizado de los dispositivos móviles inteligentes, suscita una nueva necesidad de estar conectado a la red permanentemente.

Es común, de hecho, encontrar redes Wi-Fi libres en la mayoría de establecimientos públicos desde los cuales se permite el acceso a la red de manera gratuita a cambio de aceptar las condiciones del acceso y, dependiendo del tipo de establecimiento, información proporcionada por el propio usuario.

Este tipo de accesos son posibles gracias al uso de portales cautivos que se muestran al usuario al conectarse a la red impidiendo el acceso completo a internet hasta completar el flujo del mismo. Las empresas no solo utilizan estos portales como una toma de información personal de los clientes que la utilizan sino también como herramienta de marketing y comunicación comercial.

Una particularidad de los dispositivos capacitados para mostrar estos portales es el uso de la intensidad de la señal que reciben los clientes como medida de monitorización de los mismos dentro del punto de venta físico, la cual es una característica muy interesante dentro del mundo comercial.

La motivación de este proyecto nace de la necesidad que tiene la empresa Please Networks S.L. la cual trabaja desarrollando portales cautivos a diferentes tipos de

- *E-mail de contacto: fabianbae007@gmail.com*
- *Mención realizada: Ingeniería del Software*
- *Trabajo tutorizado por: Gemma Sánchez Albaladejo (CVC)*
- *Curso 2016/17*

clientes. El crecimiento reciente de la empresa hace imperiosa la realización de una aplicación que sea capaz de gestionar los diferentes portales cautivos que poseen, añadiendo flexibilidad a cambios, robustez y modularización de los diferentes elementos comunes entre ellos.

Asimismo, el interés en crear portales cada vez más complejos y con un nivel de detalle mucho mayor para el que inicialmente fueron creados, añade, aún, mayor importancia a la realización de una aplicación más completa, con más rigor y siguiendo unos estándares en lo que a desarrollo de software se refiere.

Junto con la aplicación de creación de portales, el tratamiento de los datos captados por el dispositivo que actúa como punto de acceso a la red es otra de las tareas pendientes que tiene la empresa, la cual busca una manera fácil y sencilla de entender y mostrar los datos que se generan.

Dados estos puntos, aplicando las técnicas y conocimientos adquiridos durante el transcurso del grado en Ingeniería Informática que he realizado, decidí proponer y dar solución a toda la problemática encontrada por la empresa. Todo ello, consiste en la realización de una aplicación que aporte una solución única a la gestión de los portales cautivos existentes y por haber, y la utilización de la tecnología interna que posee la empresa para la creación de un *dashboard* (vista desde la que se representan datos e indicadores en formato gráfico) desde el que obtener toda la información captada por los dispositivos de manera centralizada y personalizada.

El documento se ha organizado primero introduciendo el problema planteado en el proyecto, siguiendo por el estado del arte y los objetivos del mismo. Posteriormente se especifican los objetivos del proyecto seguido de la metodología propuesta para su desarrollo y consiguiente resolución. A continuación, se expone el desarrollo seguido y los resultados obtenidos del mismo. Finalmente, se exponen las líneas futuras y las conclusiones del proyecto, así como los agradecimientos y la bibliografía empleada.

2 ESTADO DEL ARTE

Para realizar el proyecto, se ha procedido a hacer un análisis exhaustivo del estado del arte para las dos partes de las que trata el mismo.

2.1 Portales cautivos

Para la parte de la creación y gestión de los portales cautivos, lo primero en lo que debemos fijarnos es en la propia plataforma que utilizaba la empresa [1]. La plataforma se encontraba en un estado antiguo en cuanto a estructura de código, diseño y lógica. Por ello, solo se ha utilizado para tener una idea de como deben ser los portales que la aplicación tendrá. Se han tenido en cuenta todos los puntos que se deseaban mejorar y tratarlos para que la aplicación final pudiera tenerlos.

Existen, de hecho, varias alternativas en cuanto a sistemas de código abierto como podrían ser *EasyHotspot* [2] y *WiFiDog* [3] los cuales ofrecen un servicio con el que poder crear tu propio portal cautivo. Estos portales son muy simples y aportan una solución fácil con el que poder mostrar algún tipo de mensaje o un sistema de autenticación muy sencillo. Se a podido extraer información sobre su funcionamiento para entender como es el proceso de interacción entre la red y el usuario que intenta acceder a ella. Se puede ver un ejemplo de portal en la figura 1.

Figura 1 - Ejemplo portal cautivo Easy Hotspot

Otros casos donde se ha extraido información sobre el funcionamiento de los dispositivos a los que se conecta el usuario serían diversos estudios sobre portales cautivos: [4] y [5]. No dejan de ser alternativas muy simples de lo que es un portal cautivo, cuando lo que se buscaba con este trabajo es ir un paso más allá. Por lo tanto, sirven para entender su funcionamiento y, de esta manera, aprovechar todas las ventajas que ofrecen y dar un servicio de calidad y óptimo para el usuario.

2.2 Dashboard

Para la parte de la gestión de los datos, como pasaba en el anterior caso, debemos fijarnos primero en qué están utilizando para entender que se esperaba conseguir [6]. Se utilizaba una plataforma para visualizar los datos de *tracking* (monitorización) que se capturaban, pero no se tenía acceso a como era el funcionamiento de la aplicación. Los datos mostrados son fijos, al ser un sistema cerrado no

se puede ajustar en la totalidad a las necesidades de cada cliente y, por este motivo, se desea dejar de utilizarlo. Así, de esta manera, se deseaba tener un *dashboard* personalizado con el que poder manejar estos datos y poder mostrar gráficos que tuvieron valor para el cliente final. Se puede ver un ejemplo de *dashboard* en la figura 2.

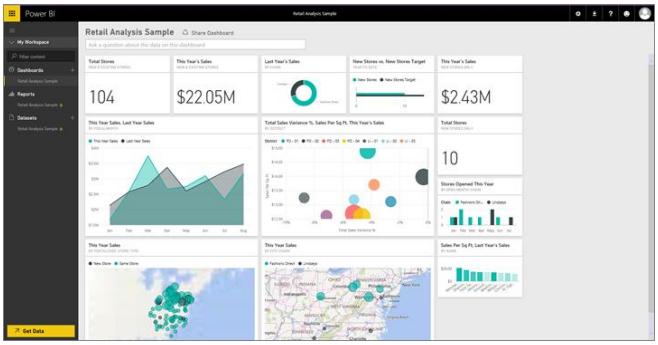


Figura 2 – Ejemplo dashboard

Para poder realizar este *dashboard* era necesario conocer cómo hacer el diseño del mismo, entendiendo en qué se basan y cómo deberían ser. Existen, de hecho, diferentes tipos de estudios [7] donde poder obtener información sobre cómo realizar y desarrollar un *dashboard* o qué elementos debería llevar. También se debía conocer o entender el funcionamiento del mismo [8]. Saber cómo se procede a la creación y desarrollo del *dashboard*, qué elementos tiene, cómo se comunican y qué tipo de *dashboard* se adecuaba más a sus necesidades.

Después de tratar de entender los diferentes tipos de herramientas de desarrollo de los portales cautivos y entender el funcionamiento de los mismos, así como para el caso de los *dashboard* personalizables, se ha procedido a planificar el desarrollo del proyecto. Se debe destacar que no toda la información era accesible puesto que muchos sistemas eran sistemas cerrados desde los cuales se hacía imposible la entrada a los mismos. A pesar de ello, sí se ha conseguido encontrar el nivel deseado en cuanto a entendimiento de los funcionamientos básicos que permitieron dar con una solución al problema planteado.

3 OBJETIVOS

Los objetivos del proyecto se han especificado y priorizado durante la fase inicial del mismo. A pesar de ello, al ser un proyecto de larga duración y teniendo en cuenta el tipo de proyecto que es, era de esperar que pudieran cambiar, variar en su priorización y replantearse a lo largo del desarrollo. Estos son, finalmente, los objetivos del proyecto.

3.1 Críticos

- Desarrollar una aplicación capaz de gestionar los diferentes portales cautivos. Este es el principal objetivo del proyecto.
- La aplicación debe permitir crear un nuevo portal cautivo, incluyendo las vistas del portal y la lógica que permite dar conexión al usuario. Se debe tener en cuenta los diferentes tipos de dispositivos que alojan estos portales y determinar el proceso de conexión correctamente.
- La aplicación debe gestionar las diferentes direcciones de los múltiples portales existentes en ella. Los portales tendrán una serie de direcciones desde las que se llamarán.
- La aplicación debe determinar el tipo de dispositivo, de manera automática, al que se está tratando de conectar y gestionar los parámetros pertinentes para ofrecer la posterior conexión al usuario.
- Desarrollar una aplicación capaz de gestionar los datos generados con los dispositivos, que ofrecen los portales cautivos, procediendo a su unificación y tratado para ofrecer datos entendibles para el usuario que se mostrarán de manera gráfica.
- La aplicación permitirá convivir diferentes clientes que se conectarán con diferentes tipos de base de datos para acceder a ellos.
- La aplicación permitirá mostrar los datos en forma de gráficas dependiendo de los parámetros especificados.

3.2 Prioritarios

- La aplicación debe ofrecer la gestión de los datos generados por el usuario, así como la validación y almacenaje de los mismos.
- La aplicación permitirá conectarse a diferentes bases de datos desde las que podrá leer y guardar los datos necesarios para establecer un seguimiento de los usuarios conectados. Así mismo, deberá gestionar los datos exclusivos al funcionamiento básico de cada portal cautivo de manera independiente.
- La aplicación será capaz de reutilizar y reaprovechar el código común para cada uno de los portales cautivos. Todo ello, utilizando herencia tanto en vistas, como controladores y modelos.
- La aplicación permitirá ser fácilmente modificable y permitir adaptar los portales de manera simple y sencilla.

- La aplicación deberá incorporar el inicio de sesión de los usuarios en las diferentes redes sociales, especialmente Facebook y Twitter.
- La aplicación que muestra los gráficos de los datos deberá incluir un filtrado mediante fechas que podrá especificar el usuario.

3.3 Secundarios

- El sistema debe ser capaz de aceptar y usar diferentes módulos de terceros capaces de la interacción entre la aplicación y diferentes estructuras ajenas, como por ejemplo la integración de códigos Qr.
- La aplicación debe estar capacitada para permitir la localización de la misma, es decir, debe ser multilenguaje.

El proyecto ha sido capacitado con todos estos objetivos, obteniendo, de esta manera, el nivel deseado y esperado al final del desarrollo del mismo.

4 METODOLOGÍA

Para proceder a la realización del proyecto se ha utilizado una metodología de trabajo ágil como puede ser SCRUM [15]. Esta metodología se basa en la partición del tiempo total necesario para desarrollar el proyecto en *sprints* de dos semanas para, de esta manera, obtener un mayor seguimiento del desarrollo del proyecto y, a su vez, obtener información del mismo.

Gracias a este tipo de metodología, se consigue obtener una versión funcional después de cada iteración, ayudando a obtener un código de mayor calidad. También, poder tener una gran capacidad de reacción ante cualquier cambio o problema surgido durante el desarrollo, reduciendo, así, los riesgos que puedan ocurrir. Además, se pueden hacer predicciones de tiempos dado que, al obtener resultados en cada iteración, conoces la situación en la que está el proyecto y lo que queda por hacer. Como el trabajo se debe fraccionar al inicio de los *sprints* para su uso en formato tarea, en cada iteración, se consigue, además, tener un gran conocimiento y profundización de lo que se está trabajando en cada momento, lo cual ayuda a implicarse en las fases de desarrollo y obtener más conocimiento del estado: qué se está trabajando, como está cada parte, etc.

Por lo tanto, se considera muy oportuna, la utilización de este tipo de metodologías para un proyecto de estas características puesto que se pueden aprovechar todos los beneficios antes comentados.

Para desarrollar todos los elementos de los cuales se compone el proyecto se han utilizado una serie de herramientas y programas que han ayudado a la resolución del mismo. Para desarrollar el código se ha utilizado *JetBrains PhpStorm*, un *IDE* para *php* muy completo, compuestos por diferentes tipos de utilidades que hacen una tarea mucho más asequible la programación web. Para la parte integrada en la plataforma de la empresa, al programar en *AngularJS* [14] se ha utilizado *JetBrains WebStorm*, similar al anterior comentado, pero más enfocado a la parte *front-end* de la aplicación.

Para el desarrollo de las bases de datos se ha utilizado *Sequel Pro*, para la base de datos *MySQL*, y *Postico*, para la base de datos *PostgreSQL*. En el primer caso, ha sido necesario crear una base de datos para guardar los datos referentes a los portales cautivos como, por ejemplo, los datos que introduce el usuario. En el segundo caso, ha sido necesario la conexión a una base de datos externa desde donde se recogen los datos referentes a la monitorización de los usuarios de los portales cautivos, ya que éstos se guardan automáticamente en este tipo de base de datos.

Para la creación de la documentación se ha utilizado *Microsoft Office Word* y *Microsoft Project*. Además de diferentes tipos de herramientas online gratuitas para el desarrollo de los diagramas.

Todos los programas eran los que utilizaba la empresa para desarrollar los portales y acceder a las diferentes bases de datos, ese es el motivo principal de su utilización dentro de este proyecto.

5 DESARROLLO

Durante el desarrollo del proyecto se hizo, primeramente, una fase de planificación y, posteriormente, se empezó a desarrollar la aplicación.

5.1 Planificación

La planificación se realizó de la siguiente manera:

Nombre de la tarea	Duración	Inicio	Final
Reunión inicial tutor	3 días	06/02/2017	08/02/2017
Consultar información	10 días	06/02/2017	17/02/2017
Planteamiento inicial	10 días	09/02/2017	22/02/2017
Reunión con la empresa	1 día	13/02/2017	13/02/2017
Toma de requisitos	5 días	14/02/2017	20/02/2017
Informe inicial	10 días	21/02/2017	06/03/2017
Diseño inicial de la aplicación	8 días	21/02/2017	02/03/2017
Diseño inicial	5 días	27/02/2017	03/03/2017

base de datos			
Diagramas UML para casos de uso	10 días	06/03/2017	17/03/2017
Diseño modelo Entidad-Relación	5 días	06/03/2017	10/03/2017
Mejora en el diseño de BD	5 días	13/03/2017	17/03/2017
Aprendizaje framework	10 días	06/03/2017	17/03/2017
Informe de seguimiento I	10 días	20/03/2017	03/04/2017
Inicio desarrollo de la aplicación	10 días	20/03/2017	03/04/2017
Desarrollo arquitectura de la aplicación	15 días	03/04/2017	24/04/2017
Inicio de pruebas de test	10 día	03/04/2017	17/04/2017
Reunión de seguimiento I	6 días	04/04/2017	11/04/2017
Gestión de riesgos y replanificación	10 días	10/04/2017	21/04/2017
Creación diagramas de secuencia	10 días	10/04/2017	21/04/2017
Continuar desarrollo de aplicación I	10 días	24/04/2017	08/05/2017
Testing de la aplicación I	10 días	24/04/2017	05/05/2017
Continuar desarrollo de aplicación II	10 días	08/05/2017	22/05/2017
Testing de la aplicación II	10 días	08/05/2017	19/05/2017
Informe de seguimiento II	10 días	08/05/2017	19/05/2017
Reunión de seguimiento II	1 día	22/05/2017	22/05/2017
Continuar desarrollo de aplicación III	10 días	22/05/2017	05/06/2017
Testing de la aplicación III	10 días	22/05/2017	02/06/2017
Inicio propuesta informe final	10 días	22/05/2017	02/06/2017
Finalización desarrollo	5 días	05/06/2017	12/06/2017

5.2 Desarrollo

Una vez dados los requisitos del sistema, se ha procedido a un análisis de las posibles herramientas para su desarrollo. Como el trabajo se ha realizado en dos partes bien diferenciadas se han tratado cada una de esas partes de manera independiente. A pesar de ello, se ha buscado encontrar una solución única entre las herramientas a utilizar.

Empezando por la aplicación desde la que se crearán y gestionarán los diferentes portales cautivos: se ha tenido en cuenta como punto principal de desarrollo el hecho del reaprovechamiento de código y de la facilidad de adaptación y adecuación de cada uno de los portales cautivos que tendrá la aplicación.

Para la parte de la gestión y visualización de los datos: se ha tenido en cuenta la fácil integración con la plataforma desde la que se permitirá la visualización de los gráficos.

Teniendo en cuenta estos puntos de partida y los lenguajes que se deben usar, puesto que al ser aplicaciones web sólo pueden desarrollarse en ciertos tipos de lenguajes, se ha procedido a analizar los diferentes *frameworks* (conjunto estandarizado de conceptos, prácticas y criterios que se utilizan en el proceso de desarrollo de software) existentes [9].

De entre todos los *frameworks* existentes se ha destacado a *symfony* [10] y *laravel* [11]. Ambos son sistemas que permiten el desarrollo de aplicaciones web en *php* por lo que cualquiera de ellos sería útil para la resolución del problema. A pesar de esto, dado los requisitos necesarios, *laravel* fue la opción más adecuada, puesto que se adapta más a ellos y a las necesidades del proyecto. *Laravel* destaca por su sencillez a la hora de programar en él, utiliza herramientas como *Artisan*, un cliente de consola que nos permite ejecutar comandos propios, *Composer*, un gestor de dependencias y paquetes de *php*, o *Migrations*, herramienta que permite el control de versiones de nuestra base de datos, que hacen de él una potente herramienta para programar [12].

Uno de los puntos más importantes a la hora de tener en cuenta *Laravel* es su sistema de *blades*, se trata de un motor de plantillas con las que manejar las vistas. De esta manera, es posible tener plantillas de las vistas más utilizadas, heredar de ellas, incluirlas, etc. Por lo que con esta herramienta se ha dado solución a nuestro problema con el reaprovechamiento del código. Utilizando estas vistas, se ha podido tener todos los elementos comunes de los portales en plantillas que, posteriormente, utilizar para crear los diferentes portales permitiendo, también, su personalización si fuera necesario. También se debe considerar el uso de las rutas para la redirección y correcto funcionamiento de las páginas. *Laravel* utiliza un sistema de rutas con las que gestionar las diferentes páginas de la aplicación, pudiendo implementar de manera simple el modelo *MVC* (modelo, vista, controlador). Podemos ver un ejemplo de cómo funciona en la figura 3. Desde el cliente se pide una solicitud para mostrar la página. Esta solicitud llega al servidor y, automáticamente, se direcciona la petición al controlador mediante el sistema de rutas. El controlador va

a buscar los datos e información que necesite al modelo, el cual accederá a la base de datos. El controlador, con la información del modelo invocará la vista correspondiente con la información de los datos. Esta vista es la que el usuario verá debido a su petición.

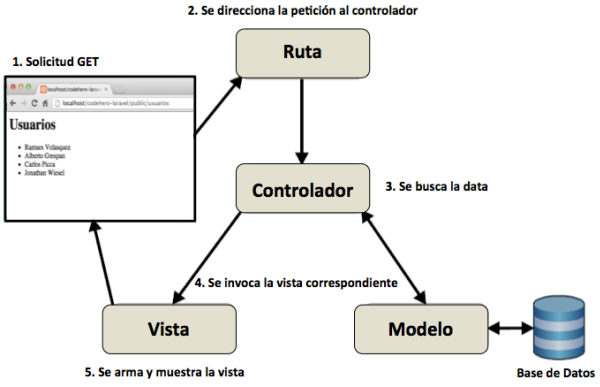


Figura 3 – Modelo MVC en Laravel

Además, el sistema de rutas juntamente con el uso de *middlewares* (son funciones que nos permiten agregar filtros a cada petición HTTP realizada por un usuario en la aplicación) son capaces de implementar el sistema de localización y multilenguaje de la aplicación, pudiendo adaptar cada petición que se haga con su correspondiente vista en el lenguaje deseado por el usuario.

En lo respectivo a la arquitectura de la aplicación, ésta se ha adaptado de manera sencilla a nuestras necesidades aportando, a la vez, una estructura sólida y ordenada de toda la aplicación. *Laravel* proporciona un directorio donde se almacenan todos los paquetes y módulos de terceros, manteniendo un control y orden y pudiendo acceder a ellos fácilmente.

Para la parte visual de la aplicación se han usado tecnologías como HTML5 y CSS, utilizando *bootstrap* [13] (otro *framework* para la creación de contenido web) para obtener un diseño *responsive* (adaptable a todos los tipos de dispositivos y tamaños de pantalla desde los que se acceda a la aplicación).

Bootstrap cuenta con una gran variedad de elementos y diseños con los que alimentar las vistas de nuestros portales obteniendo, así, un sistema moderno, elegante, sencillo y responsivo. En definitiva, aporta todos los elementos que se buscaban para poder realizar este proyecto y ayudar a estandarizarlo.

La plataforma desde la que se ha implementado el *dashboard* con los gráficos y demás elementos visuales está desarrollada con el framework *AngularJS* [14]. Se trata de un framework que utiliza el lenguaje *JavaScript*, el cual, como en el caso de *laravel*, nos proporciona herramientas con las que desarrollar nuestra aplicación y dotarla de mayor calidad siguiendo unos estándares concisos.

Todos estos elementos han permitido desarrollar, por un lado, la aplicación para crear los portales cautivos, y por otro lado la programación necesaria para imprimir los datos en el *dashboard*.

6 Resultados

Después de haber acabado el desarrollo del proyecto se ha conseguido completar todos los objetivos establecidos al inicio del mismo.

La arquitectura final de la aplicación para crear los

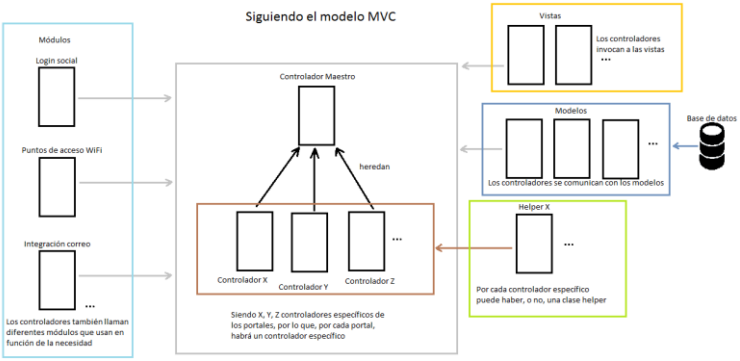


Figura 4 – Arquitectura aplicación

Como se puede ver en la figura 4, la aplicación está compuesta siguiendo el modelo *MVC* (modelo, vista, controlador). Existe un controlador maestro desde el cual heredan todos los demás controladores (de los cuales habrá uno por cada portal cautivo).

Este controlador posee todas las funciones que se utilizarán en la mayoría de portales, las cuales se llamarán a través de los ficheros de rutas (también habrá uno por cada portal). Estas funciones interaccionan con los modelos que, a su vez, se comunicarán con la base de datos para acceder a los datos propios del portal y almacenar los generados por los usuarios. Además, el controlador invocará las vistas cuando sea necesario y se comunicará con los diferentes módulos que necesite la aplicación, como puede ser el módulo que detecta qué tipo de dispositivo es el que está mostrando el portal y analizar los parámetros necesarios para proceder a dar conexión.

Dependiendo de la complejidad de cada portal puede existir una clase de soporte al controlador llamada *helper* que se encargará de ejecutar las funciones más pesadas de carga computacional dejando más limpio el controlador, el cual se comunicará con estas clases.

Las vistas de los portales cautivos también heredarán de una vista maestra que tendrá los elementos que

todos los portales tienen, como pueden ser las inclusiones de estilos. Además, existen unas vistas específicas las cuales son elementos que todo portal puede incluir, como puede ser un botón de inicio de sesión, un selector de idioma, etc.

Todo el código ha sido testeado gracias a testeo unitario, donde se comprueba el correcto funcionamiento de cada una de las funciones, y testeo de integración, donde se comprueba que la integración de cada una de las funciones sea correcta.

El uso de este tipo de arquitectura es el que hace posible el haber completado con éxito el principal objetivo de este proyecto. Cada uno de los elementos que lo componen han sido analizados y focalizados en cada uno de los objetivos del proyecto, aportando, de esta manera, un resultado de la calidad y rendimiento esperado.

La figura 5 muestra la primera página de un portal cautivo generado por la aplicación. Esta página es la primera que ven los usuarios que se conectan a la red mediante los dispositivos que permiten el acceso y contienen el portal.

Figura 5 – Ejemplo índice portal cautivo

En este punto, la aplicación ha captado los parámetros del dispositivo y del usuario, identificando el dispositivo al que se conecta y el dispositivo desde el que se está conectando. La aplicación ha guardado los datos pertinentes para proceder, posteriormente, a la conexión del usuario. Desde la vista de la figura 5 se puede cambiar el idioma si se desea, permitir el acceso mediante login con redes sociales o a partir de el e-mail.

Cada portal tiene diferentes tipos de elementos que permiten la personalización de lo que se desea mostrar. Cada elemento, además, puede ser fácilmente modificable en función de parámetros y estilos que se puden personalizar. El número de vistas también es personalizable, pero el funcionamiento del portal es siempre el mismo.

Siguiendo con la otra parte del proyecto, los *dashboards* de los datos capturados por los diferentes dispositivos, se ha desarrollado una aplicación *laravel* que actúa como *Api*. Esta *Api* consta de un controlador por cada tipo de *dashboard* (uno por cliente), junto con los modelos y rutas necesarios para cada uno de ellos. En este caso, al no utilizar ninguna vista al ser una *Api*, la plataforma que implementará los gráficos hará una llamada a una dirección de la *Api*, la cual a partir de las rutas identificará la función de qué controlador debe llamar. Esta función, se encargará de recibir los parámetros necesarios y se comunicará con los modelos para acceder a los datos de la base de datos pertinente. Se procederá, entonces, al tratamiento de estos datos para devolverlo, en el formato adecuado, a la plataforma.

La plataforma, una vez tenga los datos, procederá a imprimir la información en las vistas dependiendo de los parámetros que especifique el usuario, el cual puede filtrar por fechas para ver los datos. En la plataforma, se accederá a partir del menú del cliente a la opción de los *dashboards*. A partir de este punto, se mostrará una lista con el resumen de los datos para cada una de las tiendas desde las que se esté capturando los datos. Posteriormente, se accede a una vista para cada tienda, desde la que se mostrará el resumen de los datos en formato gráfico con la información deseada.

En la figura 6 se puede ver un claro ejemplo de una llamada desde la plataforma:

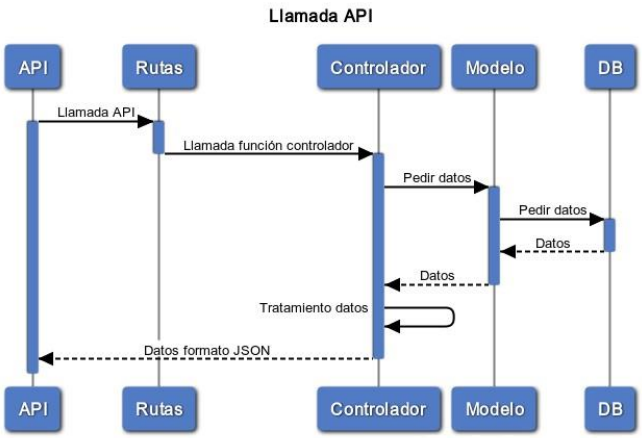


Figura 6 – Ejemplo de llamada desde la plataforma

En la figura 7 se puede ver un ejemplo de los datos en gráficos mostrados por la plataforma a partir de los datos de monitorización siguiendo el flujo anteriormente comentado.

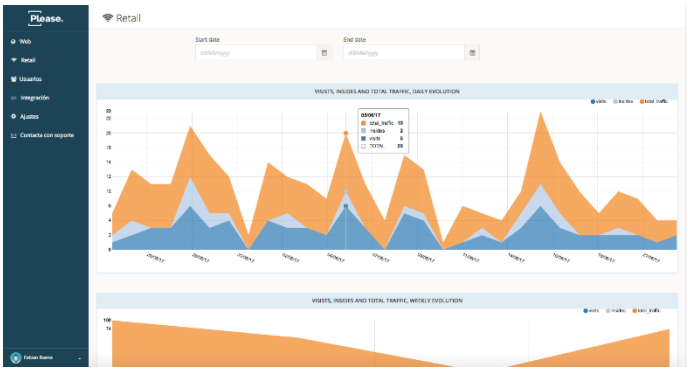


Figura 7 – Gráficos mostrados en la plataforma

7 Líneas futuras

Dado el estado actual del proyecto y la finalidad para la que fue creada, es correcto afirmar que las aplicaciones tendrán una gran utilidad dentro de la empresa para la cual han sido desarrolladas. De hecho, la aplicación para gestionar los portales cautivos se está utilizando actualmente y ha reemplazado por completo la anterior plataforma.

La aplicación de creación de portales cautivos está en un nivel muy avanzado de desarrollo, por lo que se encuentra en producción. Como puntos a mejorar se podría acabar de afinar la parte de los test, incluyendo otro tipo de técnicas para su máximo testeo, ya que una de las ventajas de esta aplicación es, también, una de las debilidades. Englobar todos los portales de manera general hace que cambiar un elemento desactualizado en cada uno de ellos implique cambiar solo una función o una clase, por contrapartida, un error en una función o clase se replicaría dentro de cada portal.

Además, la inclusión de más portales a la aplicación hará, seguramente, añadir más elementos, funciones y modificaciones a la aplicación. Por lo tanto, es una herramienta que continuará en continua evolución.

La aplicación para realizar los *dashboards* es, también, un proyecto destinado a la ampliación y mejora. En este caso, es un aplicación la cual será muy personalizable dependiendo de cada uno de los clientes que requieran de su uso. En la plataforma donde se muestran los datos y los gráficos es donde hay mayor aspecto de mejora. Uno de los puntos futuros que se trabajará es el poder generar informes en *pdf* de los datos impresos en ella, así como la exportación de los gráficos y los datos en formato *csv*.

Por lo tanto, existe una gran visión de futuro para el proyecto que se ha desarrollado.

8 Conclusiones

Como se ha comentado en la metodología seguida, la metodología de trabajo ágil ha permitido obtener un mayor seguimiento del desarrollo del proyecto.

En cada finalización de cada *sprints* se ha obtenido versiones funcionales de lo que se estaba trabajando, aportando mayor valor en la resolución del proyecto en sí. Además, esto ha ayudado a obtener conclusiones sobre en qué estado se encontraba el proyecto, permitiendo, de esta manera, tener una mejor capacidad de reacción ante imprevistos que han podido surgir. El proceso de iteraciones hace que tengas que plantear en cada *sprint* qué se está realizando, si era lo que se deseaba, qué queda por hacer, etc. Así, de esta manera, el proyecto nunca se ha visto comprometido, siempre que fuera necesario ha sido posible activar un plan de contingencia ante un posible riesgo. Por lo tanto, los requisitos se han ido adaptando a las necesidades del proyecto a medida que éste avanzaba.

Por contrapartida, la metodología de trabajo SCRUM incluye muchas más ventajas de uso pero que, por el tipo de proyecto a desarrollar, no ha sido posible de aplicar. Por lo tanto, podríamos hablar de las desventajas ocasionadas por aplicar este tipo de metodología dentro de este proyecto. La metodología SCRUM está pensada para trabajar con un equipo reducido de personas, pero no para realizar el trabajo una única persona [15]. Utiliza la repartición de tareas y roles entre los participantes por lo que, al ser una única persona la que desarrolla el proyecto estos puntos dentro de la metodología se pierden. También se hacen reuniones cada día y al inicio y final de los *sprints* que, como ya he comentado anteriormente, no ha sido posible aplicar. A pesar de esto, considero acertada la elección de la metodología porque las ventajas respecto a las metodologías tradicionales, como por ejemplo la metodología en cascada, siguen siendo mayores dadas las características del trabajo, como por ejemplo la duración.

Laravel ha proporcionado las herramientas necesarias para alcanzar los objetivos propuestos. El uso de esas tecnologías ha permitido conseguir el nivel de detalle y de calidad, en cuanto a software se refiere, deseado.

El planteamiento inicial planificado durante la primera fase de desarrollo del proyecto se siguió con éxito exceptuando algunos puntos que no se tuvieron en cuenta al inicio:

- Teniendo en cuenta el tipo de aplicación a desarrollar se decidió eliminar el diagrama de clases de la planificación inicial y sustituirlo por diagramas de secuencia ya que han aportado más valor al desarrollo de la aplicación.

Dentro del proceso de elaboración de aplicaciones web el diagrama de clases aporta mucho menos valor al proceso de diseño de la misma, por ello, se eliminó de la planificación para sustituirlo por diagramas de secuencia, que sí aportan más valor al diseño y explicación de la aplicación.

- A pesar de ello, se consideró, dadas las condiciones iniciales y las herramientas que se han utilizado para el desarrollo del proyecto, desplazar este diseño de diagramas de secuencia al siguiente *sprint*, con respecto al *sprint* inicialmente planteado. El inicio del desarrollo implica una fase de aprendizaje y consolidación de los conocimientos principales del funcionamiento de los diferentes *frameworks* que se han utilizado, lo cual se tuvo en cuenta posteriormente a la planificación incial.

Una de los puntos a tratar dentro de la gestión de riesgos y planificación fue la duración de las dos partes en las que se divide el proyecto. Puesto que la idea principal de este proyecto es la creación de la aplicación para poder crear portales cautivos y poder capturar los datos de las personas que los utilicen se dio una mayor importancia y, por ende, mayor parte del tiempo planificado para su desarrollo. Todo ello, formó parte de la mayor variación dentro del desarrollo del proyecto.

Llegados a este punto, se debe valorar lo conseguido, lo cual cumple con todos los objetivos y requisitos que debía tener, además de ciertos elementos que se han ido añadiendo a medida que avanzaba el proyecto. De esta manera, se ha obtenido una versión muy completa y enriquecida con lo que se podría decir que el proyecto ha finalizado con el éxito esperado.

Agradecimientos

Quisiera agradecer, en primer lugar, a mi tutor dentro de la empresa, Alex López, el cual me ha estado ayudando desde el primer día hasta el último en el desarrollo del proyecto.

En segundo lugar, a mi tutora dentro de la universidad, Gemma Sánchez, por su apoyo y dedicación durante estos meses.

A mi familia, que han estado pendientes desde el primer día apoyándome en todo momento, pero, en especial, a mis padres, los cuales son la razón principal de que haya podido llegar a este punto.

Por último, agradecer a mis compañeros de trabajo y amigos, que han estado pendientes de mí y me han ayudado cuando me ha hecho falta y, en definitiva, a todo el mundo que se ha preocupado por mí.

Bibliografía

[1] BeeCells - Guest WiFi becomes smart. Beecells.com. Retrieved 25 February 2017, from http://beecells.com/Referència 2

[2] EasyHotspot. Easyhotspot.inov.asia. Retrieved 25 February 2017, from http://easyhotspot.inov.asia/

[3] WiFiDog. Dev.wifidog.org. Retrieved 25 February 2017, from http://dev.wifidog.org/

[4] Moral Gómez, A., & Megías Jiménez, D. (2013). Desarrollo de un portal cautivo y herramientas de administración en un entorno wi-fi abierto.

[5] ZAMPIELLO, Geoffrey, et al. Scalable captive portal redirect. U.S. Patent Application No 11/370,811, 7 Mar. 2006.

[6] Data Visualizations & Interactive Dashboards. (2017). Bimeanalytics.com. Retrieved 25 February 2017, from https://www.bimeanalytics.com/

[7] Few, S. (2006). Information dashboard design.

[8] Loeb, B. (2010). Configurable control panel and/or dashboard display

[9] Sierra, F., Acosta, J., Ariza, J., & Salas, M. (2017). Estudio y análisis de los framework en php basados en el modelo vista controlador para el desarrollo de software orientado a la web. Retrieved 5 April 2017, from http://publicaciones.unisimonbolivar.edu.co/rdigital/ojs/index.php/identic/article/view/1517

[10] Symfony (2017). symfony.es. Retrieved 5 April 2017, from http://symfony.es/

[11] Otwell, T. (2017). Laravel - The PHP Framework For Web Artisans. Laravel.com. Retrieved 5 April 2017, from https://laravel.com/

[12] Otwell, T. (2017). Documentation - Laravel - The PHP Framework For Web Artisans. Laravel.com. Retrieved 5 April 2017, from https://laravel.com/docs/5.4

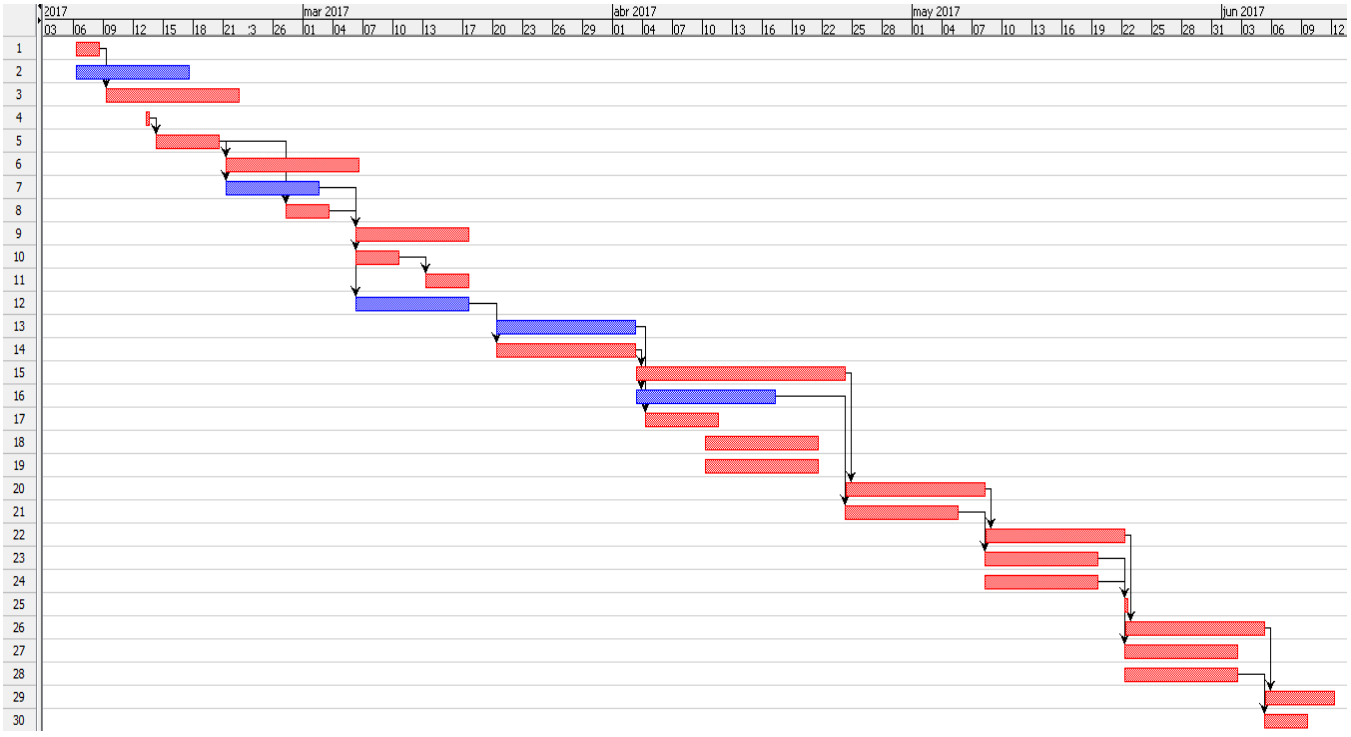
[13] Mark Otto, a. (2017). Bootstrap · The world's most popular mobile-first and responsive front-end framework. Getbootstrap.com. Retrieved 5 April 2017, from http://getbootstrap.com/

[14] AngularJS — Superheroic JavaScript MVW Framework. (2017). Angularjs.org. Retrieved 11 June 2017, from https://angularjs.org/

[15] James, M., & Walter, L. (2017). SCRUM Reference Card. Retrieved 16 May 2017, from https://www.collab.net/sites/default/files/uploads/CollabNet_scrumreferencecard.pdf

APÉNDICE

A1. Diagrama de Gantt



A2. Resultados

Ilustración 1 - Índice portal cautivo en versión escritorio

Ilustración 2- Índice portal cautivo en versión móvil

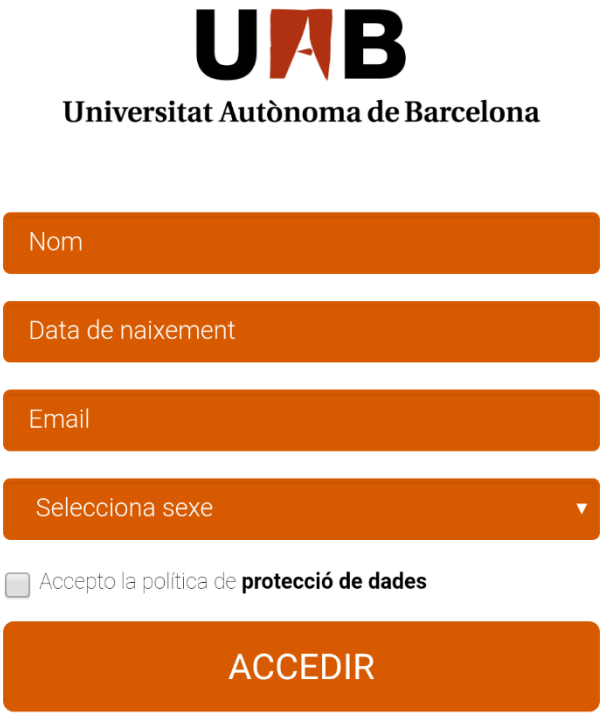


Ilustración 3 - Formulario portal cautivo en versión móvil

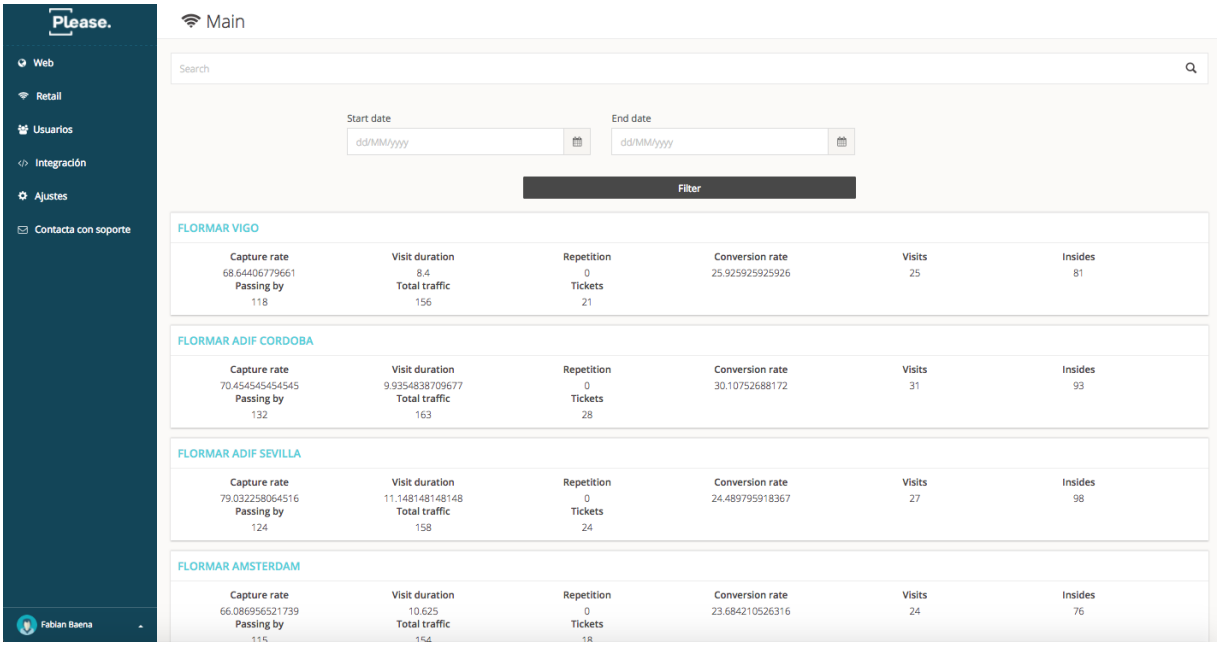


Ilustración 4 - Resúmen tiendas dashboard

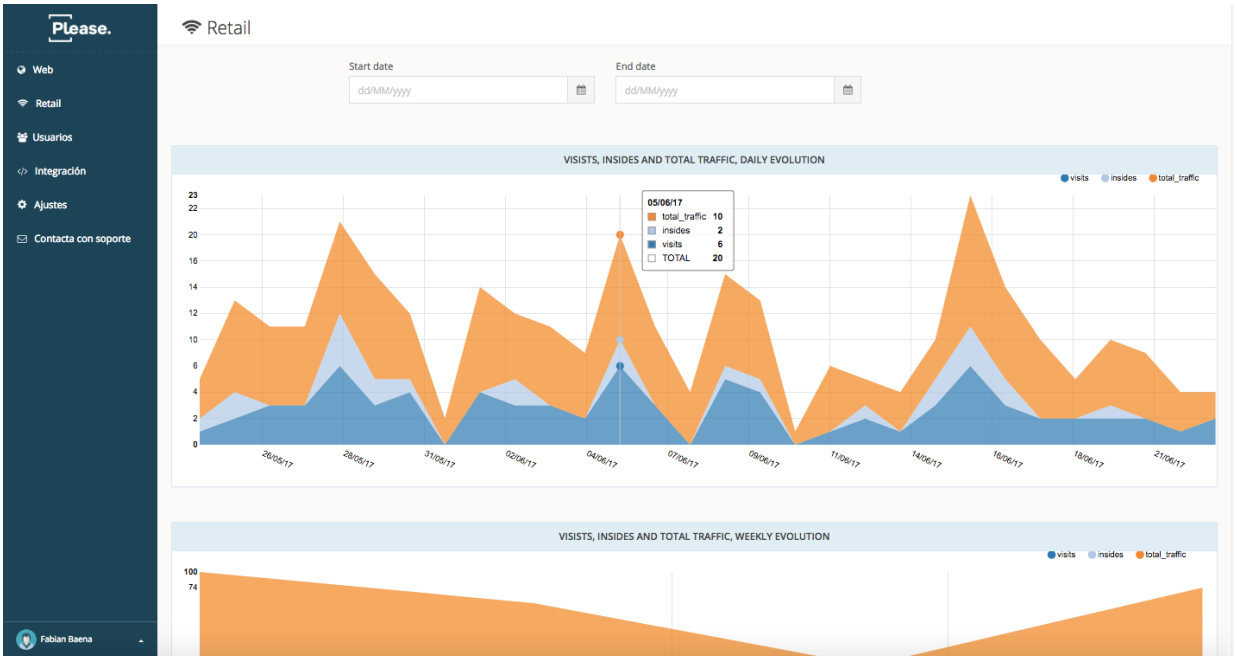


Ilustración 5 - Gráficos tienda dashboard 1

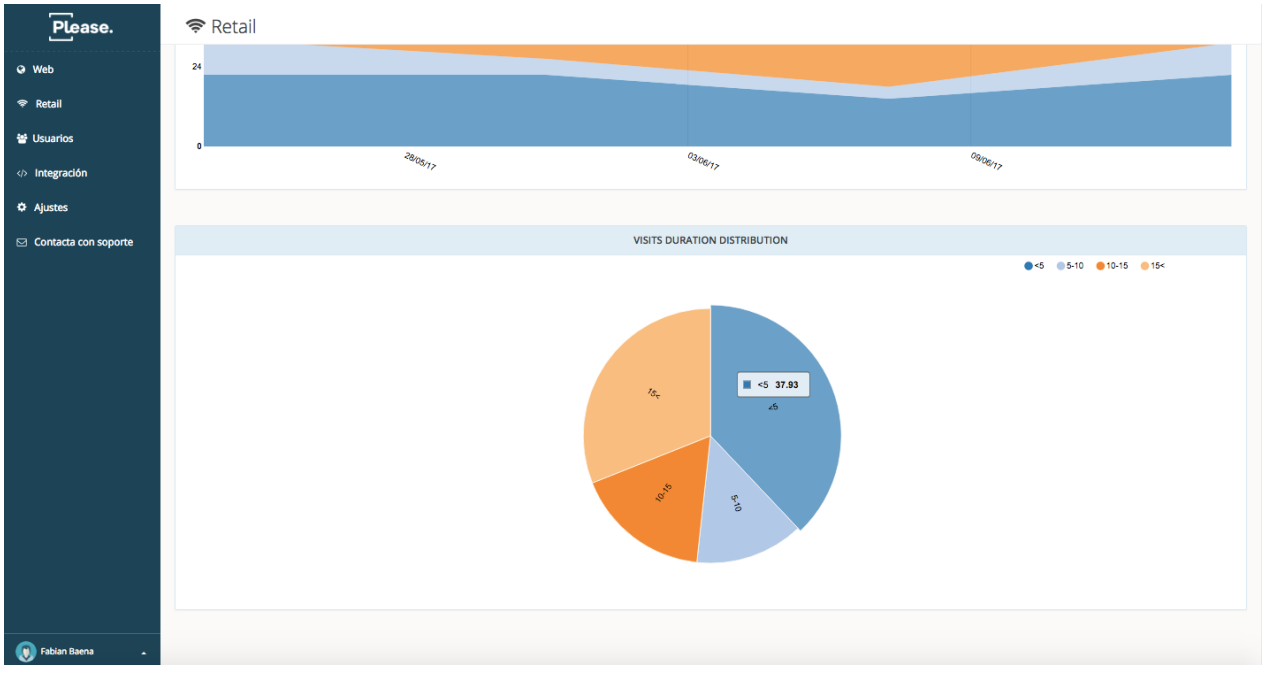


Ilustración 6 - Gráficos tienda dashboard 2