Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación web para la automatización de la gestión de campañas de publicidad por email

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Miguel Ángel Real Mañó

Tutor: José Vicente Busquets Mataix

2018-2019

Resumen

El presente trabajo ilustra todo el proceso llevado a cabo para la creación de una aplicación web dedicada a la automatización de la gestión de campañas de publicidad por email, desde su diseño hasta su despliegue. El proceso se ha llevado a cabo siguiendo la metodología ágil SCRUM y con el uso del stack tecnológico MEAN (acrónimo para: MongoDB, ExpressJS, AngularJS, NodeJS).

Con el fin de conseguir dicho objetivo se ha utilizado un servicio externo para el envío masivo de emails, al cual se accede mediante llamadas a una API. Dicho servicio proporciona una serie de webhooks, los cuales notifican a nuestra aplicación cuando ocurren ciertos eventos (apertura de un correo, clic en un link, fallo en el envío…) Para recibir y gestionar dicha información se ha desarrollado una API RESTful en el lado del sevidor.

Palabras clave: aplicación web, campañas de publicidad por email,metodología ágil, SCRUM, stack tecnológico, API, webhooks, API RESTful.

Abstract

This project illustrates the whole process which has been followed to create a web application that allows the automation of the management of advertising campaigns by email, from design to deployment. The process has been developed through agile software Scrum and using MEAN stack (acronym for MongoDB, ExpressJS, AngularJS, NodeJS).

In order to achieve the goal, an external service has been used to send bulk emails, which is accessed through calls to an API. This service provides webhooks, which notify our application when certain events occur (opening an email, clicking on a link, failure to send an email...) To receive and manage this information a RESTful API has been developed on the server side.

Keywords: web application, advertising campaigns by email, stack, API, bulk email, webhooks, API RESTful.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

El envío de emails con fines comerciales se remonta a la década de los 90 [1], con la popularización de internet las empresas vieron una oportunidad para llegar a la gente de manera rápida y económica. Fue un primer intento de email marketing, donde los empresarios se dedicaban a enviar correos electrónicos de manera masiva sin reparar en estrategia alguna. No fue hasta el 2000 cuando aparecieron las primeras herramientas dedicadas al envío de email marketing.

No duró mucho la alegría, ya que con la llegada de las redes sociales también llegó lo que hoy día se conoce como *Social Media Marketing* (Marketing en Redes Sociales), lo que supuso un boom tremendo que se tradujo en un gran golpe para otras estrategias de marketing en las que se incluía el email marketing.

Por todos es sabido que el envío de publicidad por parte de las empresas por medio del email está a la orden del día, solamente hace falta echar un vistazo a la bandeja de entrada de nuestro cliente de correo. Entonces, ¿qué pasó?,¿cómo el email marketing dio la vuelta a una situación cuanto menos complicada? La respuesta estuvo en los smartphones. Con la llegada de estos dispositivos, allá por el 2007, se dio un vuelco a toda la manera en la que el usuario consumía internet. El correo resurgió de sus cenizas debido a la accesibilidad e inmediatez que le bridaban estos dispositivos.

Hoy día, dentro de los distintos medios de marketing que existen, el email marketing es uno de los que más tasa de captación y retención de clientes tienen. Estando en cabeza en los años 2016 y 2017 como medio de captación en modelos empresariales B2B (de empresa a empresa) y entre los más importantes para los modelos B2C (de empresa a cliente).

1.1. Motivación

La motivación a nivel personal para la elección y posterior realización de este proyecto ha sido la posibilidad de poder aprender *MEAN Stack* en el proceso. Además, el hecho de realizar una web desde cero hasta su despliegue suponía un reto que sabía con toda certeza que iba a aumentar mi valor como desarrollador de software.

Es debido a la popularidad y al auge del uso de software especializado en el envío de email marketing que se decidió realizar este proyecto en concreto. Una aplicación web que permita a los usuarios la importación de sus contactos, así como la creación, gestión, envío y análisis de sus campañas de email.

1.2. Objetivos

La meta del proyecto es la creación de una aplicación web destinada al envío de campañas de email, para ello la herramienta debe de cumplir los siguientes objetivos:

- Adaptarse a los diferentes dispositivos móviles.
- Contar con un área de usuarios registrados.
- Contar con un creador de plantillas de email.

- Mostrar información básica relativa a las campañas enviadas.
- Permitir al usuario importar y gestionar sus contactos.
- Permitir al usuario enviar y gestionar campañas de correo.
- Proporcionar al usuario un nivel mínimo de seguridad.
- Ser accesible desde cualquier navegador web moderno.
- Tener un diseño limpio y minimalista.

1.3. Metodología

El desarrollo de la aplicación a realizar se llevará a cabo mediante el uso de la metodología ágil SCRUM. Se ha decidido dividir el trabajo en *sprints* de 2-3 semanas de duración, a su vez cada sprint tendrá un peso en horas que comprende entre 50-68h.

1.4. Estructura de la memoria

En la sección actual se describe de manera breve como se ha estructurado la memoria:

- **Capítulo 1:** El primer capítulo corresponde a este mismo apartado, en el que se introduce el trabajo a realizar, la motivación del alumno y se definen los objetivos del mismo.

- **Capítulo 2:** Se analizan algunas aplicaciones que son competencia directa de la aplicación que se pretende crear, con el fin de explotar alguna ventaja competitiva. Finalmente se realiza una propuesta, extraída del estudio de la competencia realizado.

- **Capítulo 3:** En este capítulo se elabora una especificación de requisitos del sistema basada en el estándar ANSI IEEE 830-1998. En este apartado se pretenden recoger los requisitos funcionales y no funcionales que deberá cumplir nuestra aplicación.

- **Capítulo 4:** Se decide que tecnologías se usarán para el desarrollo de la aplicación y se definen las clases del sistema, sus atributos y como están relacionados entre sí. Además, se presentan una serie de prototipos de diseño que pretenden ser de apoyo para la fase de desarrollo.

- **Capítulo 5:** Para la realización del trabajo se ha seguido la metodología SCRUM, una de las metodologías ágiles más populares a fecha de hoy. En este capítulo se muestra como se ha organizado el proyecto y su descomposición en los diferentes *sprints* que lo conforman.

- **Capítulo 6:** Se comentan los puntos más relevantes que se han observado durante el desarrollo de la solución.

- **Capítulo 7:** Se detalla el proceso de implantación del proyecto. Es decir, como pasamos la solución a fase de producción

- **Capítulo 8:** Nos centramos en verificar que los requisitos previamente identificados en el capítulo 3 se han cumplido satisfactoriamente.

- **Capítulo 9:** Conclusión del trabajo y relación del mismo con los estudios cursados.
- **Capítulo 10:** Referencias bibliográficas.

2. Estado del arte

2.1. Análisis de mercado

El objetivo de este apartado es realizar un análisis de la competencia, con la finalidad de saber el estado en el que se encuentran los actuales y potenciales competidores, así como detectar sus fortalezas y vulnerabilidades.

2.1.1. SendBlaster

- Aplicación de escritorio, con la parte de análisis de campañas accesible a través de su web.
- En cuanto a funcionalidad, cubre los puntos básicos que se esperan de una aplicación de estas características: gestión de contactos, envío de campañas y análisis de las campañas enviadas.
- Permite exportar los informes de las campañas a aplicaciones de terceros, como pueden ser TrackRepots o Google Analytics.
- La mayoría de aplicaciones de este estilo suelen ser de pago mensual, mientras que SendBlaster es de pago único y está en torno a los 99€.

El punto más débil de SendBlaster[1] es que, a pesar de poder ver los informes por su web, todo se hace a través de la aplicación de escritorio. Por lo tanto, no es posible acceder mediante móvil o tableta. Además, puede que las funcionalidades se queden un poco cortas para un usuario medio.

2.1.2. MailChimp

- Diseño minimalista y simple en cada una de las secciones de la web. Otro punto fuerte sería que actualizan el contenido y la estructura de su página principal cada cierto tiempo, lo que puede tener cierto gancho.
- Cuenta con las funcionalidades básicas que se pueden esperar en este tipo de herramientas, como bien pueden ser el envío de campañas o la gestión de contactos, también cuenta con un apartado para el análisis de las campañas previamente enviadas.
- A las funcionalidades básicas se le suman al menos una veintena de funcionalidades más específicas que pueden aportar al usuario un punto extra de profesionalidad. Funcionalidades como: segmentación de campañas por tags, campañas de test A/B, emails transaccionales (email de bienvenida, email de cumpleaños…), auto respondedores…

[1] https://www.sendblaster.es/

- Los emails son adaptados para la mayoría de dispositivos móviles y clientes de correo.
- Tiene un creador de correos mediante plantillas muy completo.

El punto negativo de MailChimp[2] sería que sus funcionalidades están muy segregadas. Un usuario de nivel medio-bajo necesitaría de formación para usar correctamente la herramienta. Además, el hecho de que tenga tantas funcionalidades puede hacer que se le quede grande a un usuario estándar que solo busca enviar sus campañas de email.

2.1.3. SendiBlue

- Tiene un buen número de funcionalidades que amplían a las básicas: segmentación de clientes, programar plantillas para una fecha determinada, envío de SMS, emails transaccionales…
- Interfaz gráfica aceptable, no obstante, se queda por detrás comparada con la de MailChimp.
- Los emails son adaptados para la mayoría de dispositivos móviles y clientes de correo.
- Cuenta con un creador de correos mediante plantillas.

SendiBlue[3] se baraja como la alternativa perfecta a MailChimp, teniendo la mayoría de sus funcionalidades y siendo una opción más económica. El mayor contra que tiene, al igual que MailChimp, es que el público objetivo abarca a usuarios expertos en email marketing.

2.1.4. Software CRM

El software CRM (*Customer Relationship Management*) son todas aquellas aplicaciones cuya finalidad es aunar y automatizar todo el proceso de negocio entre empresa y cliente.

Esto suele traducirse en un cambio significativo en la manera en la que se trabaja. Por otro lado, tenemos toda la información relativa a los clientes digitalizada y centralizada.

¿Es un CRM realmente una alternativa? La respuesta es depende. Si somos una empresa y estamos pensando en obtener un software de gestión, entonces sin duda. Muchos CRM tienen integrado el servicio de mail marketing, evidentemente a un nivel más básico que páginas como MailChimp u otras que mostramos arriba. No obstante, este no es un factor negativo, ya que gran parte de empresas no saca todo el partido a las aplicaciones anteriormente comentadas y su uso es el de un usuario promedio, y aquí estamos cubiertos por lo que nos ofrece un CRM.

[2] https://mailchimp.com/
[3] https://es.sendinblue.com/

Aunque pueda considerarse una alternativa no podemos considerarlo competencia directa, ya que el número de personas que acaba adquiriendo este tipo de software cuando inicialmente busca una solución profesional de email marketing es muy bajo, por los siguientes motivos:

- Suponen un cambio grande en la manera de trabajar.
- Opción menos económica.
- Interfaz arcaica en la mayoría de los casos.
- Más limitada que otras herramientas de email marketing que se dediquen en exclusiva al sector.

2.2 Propuesta

Actualmente existen un gran número de aplicaciones dedicadas al envío de campañas de email, sin embargo, un alto porcentaje de ellas están enfocadas a usuarios expertos. Por otro lado, tenemos aplicaciones orientadas a usuarios noveles con un número muy limitado de funcionalidades (suelen ser las funcionalidades más básicas).

Nuestro objetivo es situarnos entre estas dos categorías, ofrecerles a los usuarios las funcionalidades básicas y además añadir otras más específicas que puedan ser de gran utilidad (segmentación de contactos, creador de plantillas, seguimiento de campañas…). Otra parte importante será mantener un diseño minimalista y limpio, altamente usable que permita que cualquier usuario novel pueda utilizar la aplicación de manera intuitiva.

3. Análisis de requisitos

3.1. Introducción

Este apartado es una Especificación de Requisitos Software (ERS) para un sistema de envío de campañas de publicidad por email.

Para la formulación de dicho ERS se ha seguido el estándar IEEE 830-1998 [3].

3.1.1. Propósito

El presente apartado tiene como objetivo definir las especificaciones funcionales, no funcionales y del sistema para la implementación de una herramienta web que permita a los usuarios la creación y gestión de contactos, así como el envío de campañas de email a todo el conjunto de los contactos o a un subconjunto del mismo.

La aplicación web podrá ser utilizada por cualquier persona que tenga una cuenta en la propia aplicación.

3.1.2. Ámbito del sistema

El nombre de la aplicación será mailinplane, un doble juego de palabras que bien puede significar "*mail in plane*" o "*mailin plane*" (haciendo alusión a *mailing*).

A grandes rasgos mailinplane será una aplicación web la cual será de utilidad en los siguientes procesos:

- Administrar contactos.
- Crear campañas de email de manera rápida y sencilla.
- Enviar campañas de email.
- Generar informes relativos a los envíos de las campañas.
- Generar emails adaptados a los diferentes dispositivos móviles.

No estarán contempladas las siguientes funcionalidades dentro de la aplicación:

- Envío automático de emails transaccionales.
- Encuestas y sondeos.
- Test A/B
- Editor de imágenes.

3.1.3. Definiciones, acrónimos y abreviaturas

Sistema: Nuestra aplicación de envío de campañas.

Usuario: Persona que usará el sistema.

Frontend: Parte del sitio web con la que interactúan los usuarios.

Hardware: Partes físicas y tangibles de un sistema informático

Freemium: Modelo de negocio en el que se otorga a los usuarios un servicio completamente funcional, y a este servicio se le añaden ciertas funcionalidades a las que solo tienen acceso los usuarios premium

JavaScript: Lenguaje de programación basado en el estándar ECMAScript

Bcrypt: Función hash de contraseñas basada en el cifrado blowfish, pensada para ser costosa computacionalmente con el fin de evitar ataques de fuerza bruta.

Mailgun: Servicio web orientado al envío de emails en grandes cantidades.

Principio de responsabilidad única: Primer principio de diseño de software de SOLID. Cada clase debería ser responsable de realizar una única actividad en el sistema.

Principio de abierto/cerrado: Segundo principio de diseño de software de SOLID. Ante un cambio se deberá extender funcionalidad y no modificar la existente.

Hash: Algoritmos que aplicados sobre una entrada (texto, contraseña, archivo…) generan una salida de longitud fija, de la cual no se puede inducir la entrada original. Son irreversibles.

SOLID: Cinco principios básicos de diseño de software cuya aplicación tiene relación directa con la creación de herramientas fácilmente mantenibles y ampliables con el tiempo.

API: Interfaz de Programación de Aplicaciones

BDD: Base de datos.

ERS: Especificación de Requisitos Software

JWT: JSON Web Tokens.

3.1.4. Referencias

Título del documento	Referencia
IEEE Std. 830 – 1998	IEEE

Tabla 1: Referencias de análisis de requisitos

3.1.5. Visión General

La especificación de requisitos va a seguir la siguiente estructura:

- **Introducción**: Se da una visión general del presente apartado (Análisis de requisitos) y del propio sistema.

- **Descripción general**: En esta sección se verán todos aquellos factores que afecten al producto y a sus requisitos, como bien pueden ser las dependencias y restricciones del mismo. También se verán a grandes rasgos las funcionalidades del sistema, pero nunca entrando al detalle.

- **Requisitos Específicos**: Esta sección contendrá tanto los requisitos funcionales como los requisitos no funcionales del sistema con un gran nivel de detalle.

3.2. Descripción general

3.2.1. Perspectiva del producto

La aplicación mailinplane estará diseñada con el propósito de ser accedida a través de un entorno web. Utilizará la API de mailgun para el envío masivo de emails y se aprovechará de algunas de sus características que nos permitirán: saber si el destinatario abre el email, si ha fallado el envío, los clics que se han hecho sobre los links del correo o si se ha producido rebote en el envío.

3.2.2. Funciones del producto

En este punto se presentan los diagramas de casos de uso asociados con nuestra aplicación. Se presentan los dos tipos de actores que podrán interactuar con el sistema: el actor "visitante" (Ilustración 1) y el actor "usuario registrado" (Ilustración 2).

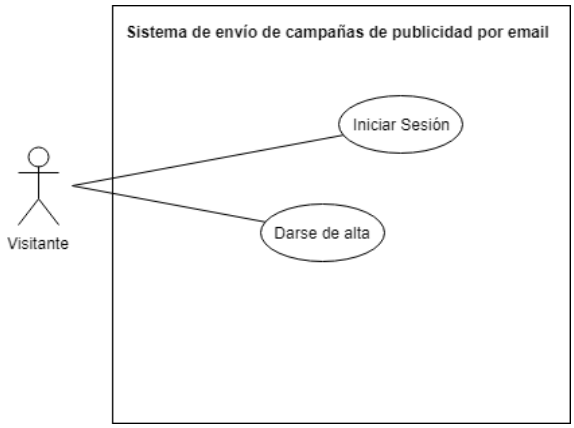


Ilustración 1: Diagrama de casos de uso del actor "Visitante"

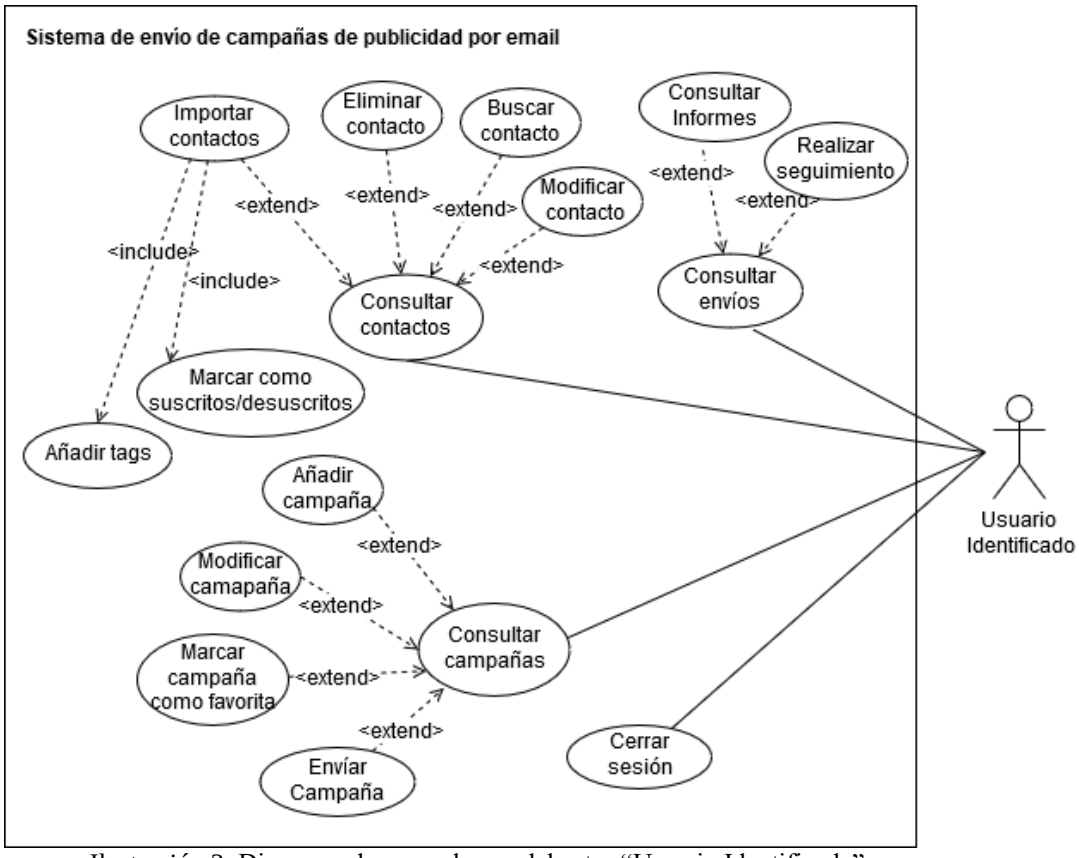


Ilustración 2: Diagrama de casos de uso del actor "Usuario Identificado"

3.2.3. Características de los usuarios

La herramienta web mailinplane contendrá 2 tipos de usuarios:

- **Visitante**: Usuario que ha accedido a la página principal de la aplicación y no ha iniciado sesión.

- **Usuario registrado**: Usuario que ha iniciado sesión correctamente en la aplicación.

3.2.4. Restricciones

Las restricciones de la aplicación serán las siguientes:

- La aplicación debe ser accesible por red.
- Es necesario tener internet para el uso de la aplicación.
- Estamos limitados al uso de HTML, CSS y JS en el *frontend*, ya que es prácticamente un estándar de programación en la parte del navegador.
- Deben cumplirse unos mínimos en cuanto a seguridad (*Hash* de contraseñas en BDD, uso de JWT…)

3.2.5. Suposiciones y dependencias

Haciendo uso de algunos de los principios SOLID como pueden ser el principio de responsabilidad única y el principio de abierto/cerrado vamos a tratar de hacer el código lo más mantenible posible y con el menor número de dependencias [7].

No obstante, sí existe una dependencia directa que en caso de fallar afectaría al correcto funcionamiento del sistema. En este caso sería mailgun, su mal funcionamiento se traduciría en la imposibilidad de enviar emails por parte del usuario y en el cese de la recolección de datos de interés de los correos previamente enviados.

3.2.6. Requisitos futuros

Dado que estamos ante un sistema abierto que ha de admitir posibles cambios en el futuro, se han planteado las posibles funcionalidades a abordar:

- Dar la posibilidad al usuario de diseñar el email a partir de unas plantillas por defecto (basadas en temáticas).
- Implementar un modelo de negocio *freemium*.
- Integración de pasarela de pago.
- Aumentar el número de funcionalidades del creador de plantillas de correo.

3.3. Requisitos específicos

3.3.1. Interfaces externas

3.3.1.1. Interfaces de usuario

La interfaz de mailinplane deberá ser diseñada pensando en el usuario, y por tanto tener un alto grado de usabilidad, para ello nos apoyaremos en los siguientes puntos:

- Diseño estético y minimalista.
- Diseño consistente
- Diseño intuitivo y amigable.

3.3.1.2. Interfaces hardware

Las interfaces hardware del sistema serían todos aquellos dispositivos con internet que tenga acceso a un navegador web moderno. En concreto el sistema debe estar preparado para ejecutarse de manera correcta en ordenadores, tabletas y móviles.

En el caso de los ordenadores de sobremesa también sería necesario contar con monitor, teclado y ratón.

3.3.1.3. Interfaces software

En necesario que los dispositivos que vayan a utilizar el sistema tengan instalados un sistema operativo, así como acceso a un navegador web moderno que ejecute JavaScript.

3.3.1.4. Interfaces de comunicaciones

No se han definido.

3.3.2. Funciones

Darse de alta	
Descripción	El sistema debe permitir a un visitante registrarse en la aplicación.
Referencia	RF01
Entradas	Email y contraseña
Proceso	Creación de un nuevo usuario en Base de Datos.
Salidas	• Operación realizada con éxito: Se le avisará al usuario de que debe validar su cuenta. • Validación del formulario incorrecta: Se le mostrará al usuario que campos debe corregir. • Email duplicado en el sistema. Mensaje de aviso. • Fallo del sistema al intentar realizar la operación: Mensaje de error.

Tabla 2: Requisito funcional: Darse de alta

Iniciar sesión	
Descripción	El sistema debe permitir a un visitante iniciar sesión en la aplicación.
Referencia	RF02
Entradas	Email y contraseña
Proceso	Comprobación de los datos de acceso.
Salidas	• Operación realizada con éxito: Se redirigirá al usuario al interior de la aplicación. • Validación del formulario incorrecta: Se le mostrará al usuario que campos debe corregir. • Email inexistente. Mensaje de aviso. • Contraseña incorrecta. Mensaje de aviso.

	• Fallo del sistema al intentar realizar la operación: Mensaje de error.

Tabla 3: Requisito funcional: Iniciar sesión

Cerrar sesión	
Descripción	El sistema debe permitir cerrar la sesión actual.
Referencia	RF03
Entradas	Ninguna.
Proceso	Eliminar token de sesión del navegador.
Salidas	Se redirigirá al usuario a la página principal.

Tabla 4: Requisito funcional: Cerrar Sesión

Consultar contactos	
Descripción	El sistema debe permitir listar todos los contactos de la Base de Datos.
Referencia	RF04
Entradas	Ninguna.
Proceso	Se listarán todos los contactos de la Base de Datos para el usuario identificado.
Salidas	Información de contactos.

Tabla 5: Requisito funcional: Consultar contactos

Buscar contacto	
Descripción	El sistema debe permitir buscar un contacto en concreto dentro de todo el conjunto de contactos existentes.
Referencia	RF05
Entradas	Texto a buscar.
Proceso	Se tecleará un texto y el sistema devolverá los contactos relacionados con ese texto.
Salidas	Información de contactos.

Tabla 6: Requisito funcional: Buscar contacto

Modificar contacto	
Descripción	El sistema debe permitir la edición de un contacto previamente creado.
Referencia	RF06
Entradas	Email, nombre, empresa, teléfono, edad, género, tags, contacto suscrito. Además de los campos predefinidos se podrán editar otros que corresponderán a las cabeceras creadas (si hubiese alguna) en la importación de contactos.
Proceso	El sistema comprobará que el campo pasa la validación, entonces editará los atributos del contacto en Base de Datos.
Salidas	• Operación realizada con éxito: Se le informa al usuario y posteriormente se le redirige a la sección de contactos. Quedaría por tanto realizada la edición. • Validación del formulario incorrecta: Mensaje de aviso. • Fallo del sistema al intentar realizar la operación: Mensaje de error.

Tabla 7: Requisito funcional: Modificar contacto

Eliminar contacto	
Descripción	El sistema debe permitir la eliminación de uno o más contactos.
Referencia	RF07
Entradas	Contactos a eliminar.
Proceso	El sistema le pedirá confirmación al usuario, si el usuario sigue adelante se procederá a la eliminación del contacto en Base de Datos.
Salidas	• Operación realizada con éxito: Se le avisará al usuario de que se han borrado los contactos solicitados. • Fallo del sistema al intentar realizar la operación: Mensaje de error.

Tabla 8: Requisito funcional: Eliminar contacto

Importar contactos	
Descripción	El sistema debe permitir cargar contactos ya sea mediante la subida de archivos .csv o copiando y pegando el texto relativo a los contactos de archivos con extensión: txt,xls,xlsx.
Referencia	RF08

Entradas	Archivo de contactos o directamente el texto con los contactos. El texto debe dividirse en filas y columnas.
Proceso	El sistema comprobará si existe o no fila de cabecera. Posteriormente filtrará aquellas filas que no tengan el atributo email y eliminará filas con emails duplicados. Esta información será mostrada al usuario en columnas y deberá relacionar cada columna con la cabecera que corresponda. En caso de no existir podrá crearla en el momento. Para terminar, se podrán añadir etiquetas y marcar como suscritos los contactos a importar.
Salidas	• Operación realizada con éxito: Se le informa al usuario y posteriormente se le redirige a la sección de contactos. Quedaría por tanto realizada la importación. • Email no enlazado: Mensaje de aviso. • Fallo del sistema al intentar realizar la operación: Mensaje de error.

Tabla 9: Requisito funcional: Importar contactos

Añadir etiquetas	
Descripción	El sistema debe permitir asociar etiquetas.
Referencia	RF09
Entradas	Etiquetas.
Proceso	El sistema asociará el valor de la entrada en el campo de base de datos llamado “tags”
Salidas	Ninguna.

Tabla 10: Requisito funcional: Añadir etiquetas

Marcar lista como suscrita/desuscrita	
Descripción	El sistema debe permitir al usuario marcar una lista de contactos como suscrita o no.
Referencia	RF10
Entradas	Valor del campo “suscritos”.
Proceso	El sistema asociará el valor de la entrada en el campo booleano de base de datos llamado “suscrito”
Salidas	Ninguna.

Tabla 11: Requisito funcional: Marcar lista como suscrita/desuscrita

Consultar campañas	
Descripción	El sistema debe permitir listar todas las campañas de la Base de Datos para un determinado usuario.
Referencia	RF11
Entradas	Ninguna.
Proceso	Se listarán todas las campañas del objeto “Campañas” de la Base de Datos para el usuario identificado.
Salidas	Campañas.

Tabla 12: Requisito funcional: Consultar campañas

Crear campaña	
Descripción	El sistema debe permitir la creación de campañas de email.
Referencia	RF12
Entradas	Nombre de la campaña Tipo de campaña: • Regular: Texto plano. • HTML: Por plantilla.
Proceso	Se creará en Base de Datos una nueva campaña con el nombre y el tipo indicado.
Salidas	• Operación realizada con éxito: Se redirigirá al usuario al modo edición de la campaña. La campaña estaría creada. • Fallo del sistema al intentar realizar la operación: Mensaje de error.

Tabla 13: Requisito funcional: Crear campaña

Modificar campaña	
Descripción	El sistema debe permitir editar los valores de los campos asociados a las campañas ya creadas.
Referencia	RF13
Entradas	Audiencia, nombre, asunto, texto que se previsualizará, email del remitente, nombre del remitente y contenido del mensaje.

Proceso	Se modificará en Base de Datos los atributos modificados. En caso de no haber modificado ningún atributo no se hará cambio alguno en Base de Datos.
Salidas	No hay salidas. Los avisos y errores se hacen en el momento del envío de la campaña.

Tabla 14: Requisito funcional: Editar campaña

Marcar campaña como favorita	
Descripción	El sistema debe poder dejar marcar campañas como favoritas con el fin de facilitar las futuras búsquedas de las mismas.
Referencia	RF14
Entradas	Campaña.
Proceso	En Base de Datos se cambiará el valor del atributo booleano “Fav” a *true*.
Salidas	Habrá *feedback* a nivel visual.

Tabla 15: Requisito funcional: Marcar campaña como favorita

Enviar campaña	
Descripción	El sistema debe permitir el envío de campañas de email.
Referencia	RF15
Entradas	Campaña.
Proceso	El sistema verificará que se cuenta con la información necesaria para el envío de la campaña. Una vez comprobado, nuestro servidor llamará a la API de mailgun con los datos que necesite para el envío de la campaña.
Salidas	• Operación realizada con éxito: Se le informa al usuario de que la campaña ha sido enviada y posteriormente se le redirige a la sección de campañas. • Validación incorrecta: Se le mostrará al usuario que campos debe corregir. • Fallo del sistema: Mensaje de error.

Tabla 16: Requisito funcional: Enviar campaña

Consultar envíos	
Descripción	El sistema debe permitir listar todos los envíos de campañas de la Base de Datos.
Referencia	RF16
Entradas	Ninguna.
Proceso	Se listarán todos los informes del objeto “CampañasEnviadas” de la Base de Datos para el usuario identificado.
Salidas	Información de campañas enviadas.

Tabla 17: Requisito funcional: Consultar envíos

Consultar informes	
Descripción	El sistema debe permitir acceder a un envío en concreto para ver información más detallada.
Referencia	RF17
Entradas	Campaña enviada.
Proceso	Se cargará de Base de Datos toda la información útil de la campaña enviada para la generación de los informes.
Salidas	Informes.

Tabla 18: Requisito funcional: Consultar informes

Realizar seguimiento de envío	
Descripción	El sistema debe permitir el seguimiento de un registro del listado de campañas.
Referencia	RF18
Entradas	Campaña enviada.
Proceso	En Base de Datos habrá un atributo booleano para indicar si un envío está en seguimiento. Le asignaremos el valor *true*. Los envíos en seguimiento saldrán en otra lista para su rápida localización.
Salidas	*Feedback* a nivel visual.

Tabla 19: Requisito funcional: Seguimiento de envío

3.3.3. Requisitos de rendimiento

- El tiempo de respuesta medio de todas las funcionalidades de la página debe ser menor a 3 segundos.
- El tiempo de respuesta medio asociado con la carga de la página debe ser menor a 6 segundos.

3.3.4. Restricciones de diseño

No se han definido restricciones de diseño.

3.3.5. Atributos del sistema

3.3.5.1. Seguridad

- Las contraseñas deben estar cifradas en base de datos utilizando bcrypt.
- La contraseña del usuario debe tener un mínimo de 8 caracteres.
- El sistema hará uso de la autenticación basada en *tokens* (JWT).
- La aplicación web deberá operar sobre el protocolo HTTPS.
- El usuario no ha de poder acceder a las funcionalidades del sistema sin haber verificado la cuenta a través del email de verificación.

3.3.5.2. Fiabilidad

El sistema ha de ser tolerante ante los posibles fallos que puedan producirse, avisando al usuario de lo ocurrido siempre que tenga impacto directo sobre el mismo.

3.3.5.3. Disponibilidad

El sistema estará operativo 7 días a la semana 24 horas al día.

3.3.5.4. Mantenibilidad

Mailinplane será un sistema abierto, lo que quiere decir que será susceptible de ampliar su funcionalidad. El sistema deberá diseñarse con el fin de ser fácilmente mantenible [6].

4. Diseño

4.1. Arquitectura del sistema

Nuestro sistema utiliza una arquitectura cliente/servidor de tres capas (Ilustración 3) donde distinguimos:

- **Cliente**: Es el encargado de realizar la petición de un recurso al servidor.
- **Servidor**: En este caso nuestro servidor contendrá una API. Es el encargado de gestionar la petición del cliente y comunicar con la base de datos si hiciese falta.
- **Base de Datos**: Contiene toda la información no volátil de la aplicación. Se comunica directamente con el servidor.

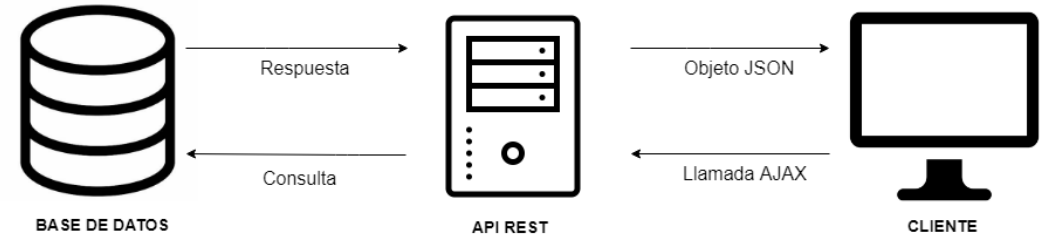


Ilustración 3: Arquitectura del sistema

Por otro lado, estamos trabajando con AngularJS, una librería de JavaScript que utiliza el patrón de diseño Modelo-Vista-VistaModelo, más conocido por sus siglas MVVM [4] (Ilustración 4).

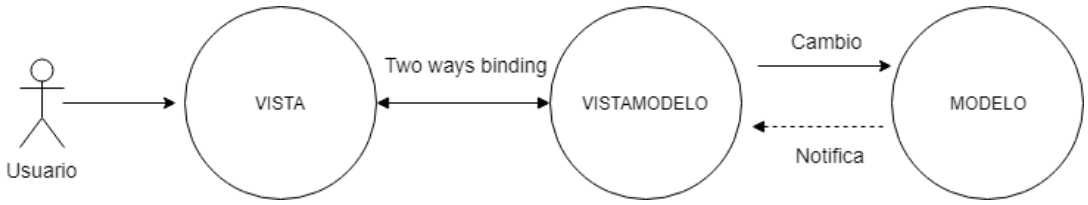


Ilustración 4: Patrón MVVM

La principal diferencia entre el patrón MVC y el patrón MVVM es la propiedad que este último posee llamada "*two ways binding*" (unión de dos sentidos). Gracias a esto, cualquier cambio producido directamente sobre la vista se refleja de manera inmediata sobre el modelo y viceversa.

En el patrón de diseño MVVM el controlador se sustituye por la capa vistaModelo, que además de contener la lógica de negocio es la responsable de mantener las relaciones entre la vista y el modelo.

Otra propiedad importante de este patrón es el desacople entre la capa de lógica de negocio y la capa de presentación, existiendo dependencia entre vista y vistaModelo pero no al contrario. Esto se traduce en la posible reutilización de un vistaModelo en varias vistas.

4.2. Diseño detallado

4.2.1. Capa de persistencia

Para describir la estructura del sistema se ha elaborado un diagrama de clases (Ilustración 5), el cual muestra las clases, atributos y relaciones entre los objetos del propio sistema.

- **User**: Representación de un usuario registrado en el sistema. Sus atributos son: email, contraseña y un atributo denominado "active" que indica si el usuario ha completado la verificación.

- **Contact:** Representación de un contacto**.** A la información básica de contacto se le suma: fecha de creación, fecha de última modificación y un atributo que indica si el contacto está suscrito a la lista del usuario.

- **TargetContact**: Representación de un contacto al que se le ha enviado una campaña de email. Contiene información relativa al estado del correo que se la ha enviado: si el contacto ha abierto el email, fecha de apertura, número de clics del contacto en los enlaces, si se ha dado de baja de la lista, fecha de baja, fallo en el envío…

- **Campaign:** Representación de una campaña de email. Está compuesta por: identificador, nombre, fecha de creación, fecha en la que se envió por última vez, número de veces que se ha enviado y el atributo "Fav" que indica si la campaña está marcada como favorita.

- **Setup:** Representación de la configuración asociada a una campaña. Compuesta por: asunto, texto que se va a previsualizar, email de remitente, nombre de remitente y contenido (texto o html, dependiendo del tipo de campaña).

- **CampaignSent:** Representación de un envío de campaña. Los atributos son: identificador, fecha de envío, email de quién envía la campaña y si la campaña está marcada para su seguimiento.

- **Audience:** Representación del segmento de usuarios al que se le va a enviar una campaña. La clase está formada por un identificador y el tipo de audiencia que la conforma. A la hora de realizar el envío de una campaña se utilizará la audiencia para seleccionar los "contactos objetivo".

- **CustomAudience:** Tipo específico de audiencia que añade información adicional a esta.

- **Tag:** Representa una etiqueta y solamente contiene un identificador y un nombre.

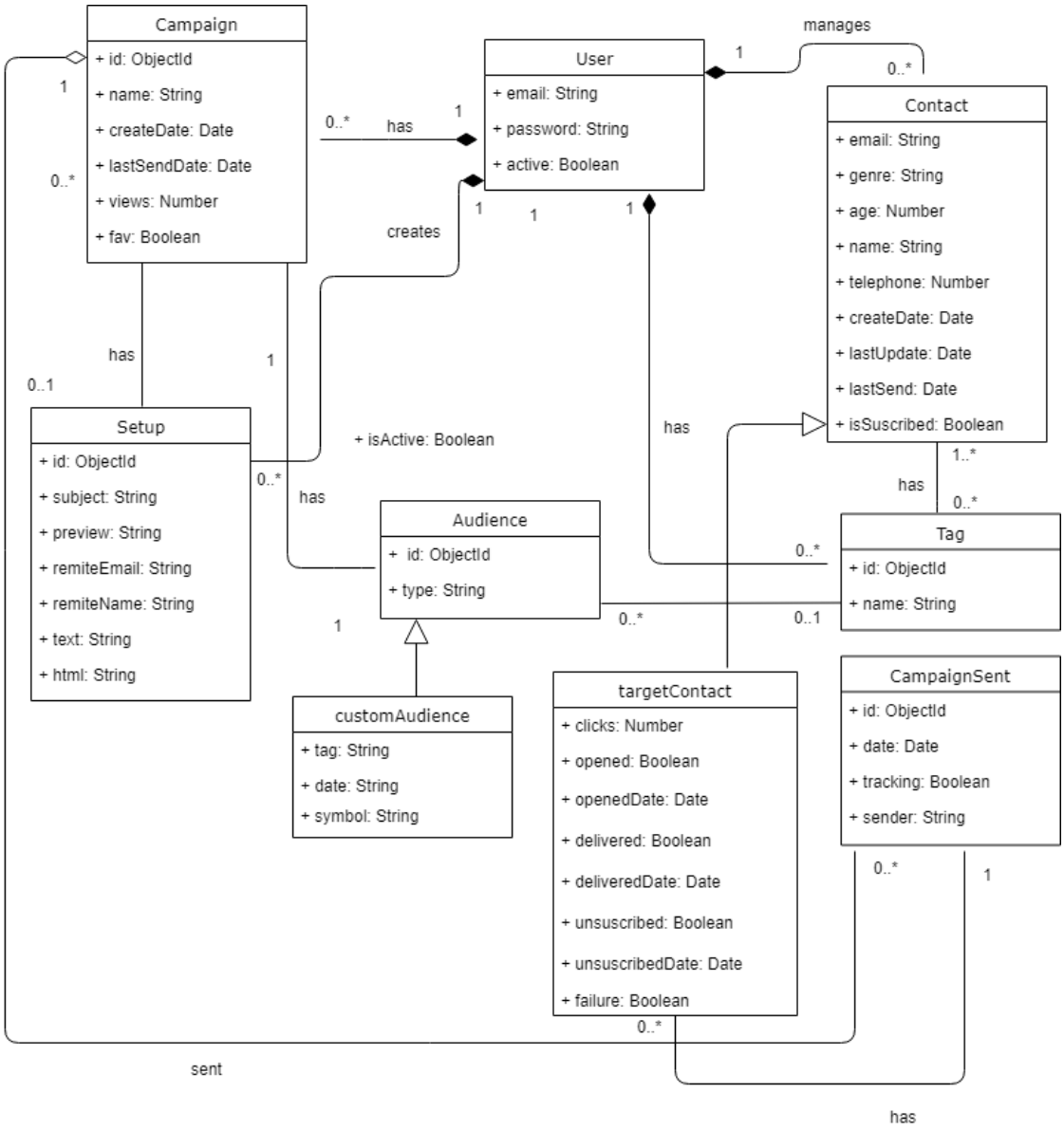


Ilustración 5: Diagrama de clases UML

4.2.2. Capa de presentación

La capa de presentación también es conocida como capa de usuario. En esta capa se organiza la información y se le presenta en pantalla al usuario.

Con el fin de tener una visión primeriza de cómo se va a organizar la información en pantalla y facilitar el posterior desarrollo de la aplicación, se han realizado diferentes prototipos con el uso del programa de escritorio balsamiq[4].

[4] https://balsamiq.com/

Destacar que en toda la aplicación se hace especial énfasis en cumplir los siguientes tres puntos:

1. **Diseño minimalista**: Paleta limitada de colores, espacio entre contenido, todos los elementos tienen que tener relación con la temática de la web...

2. **Delimitar colores de marca:** Algo ya muy estudiado por los profesionales del marketing son los colores de marca. La psicología del color. La gente relaciona colores con estados de ánimo, además se sabe que los colores asociados a una marca consolidan a la misma. Por lo que el objetivo será elegir una paleta adecuada que represente lo que queremos vender, además del correcto uso de la misma para intentar que los usuarios asocien los colores de nuestra aplicación con nuestra marca.

3. **Tipografía:** Al igual que el color la tipografía puede convertirse en un rasgo distintivo de la marca. Nosotros queremos transmitir un mensaje, la manera en la que se hace en el medio digital es la escritura. Psicológicamente la representación visual del mensaje influye sobre el lector, contextualizando el mensaje. Por lo tanto, hemos de elegir una tipografía que se adapte a nuestro producto.

4.2.2.1. Prototipos

El objetivo de la página principal (Ilustración 6) es convertir a los navegantes de paso en usuarios de nuestra aplicación, para ello utilizaremos las siguientes estrategias en esta vista:

- **Textos amigables:** Elegir una tipografía más legible no garantiza que el usuario vaya a leer el texto. Por eso, hemos de evitar una vez desarrollemos la página principal hacer uso de textos monótonos y poco atractivos.

- **Poner el foco en lo que queremos que haga el usuario:** Queremos que el usuario se registre y pruebe la aplicación, para ello se hará un uso reiterado de los botones de registro. Utilizando sinónimos, por ejemplo, “Únete gratis” y resaltando el color del botón frente al contenido que le rodea.

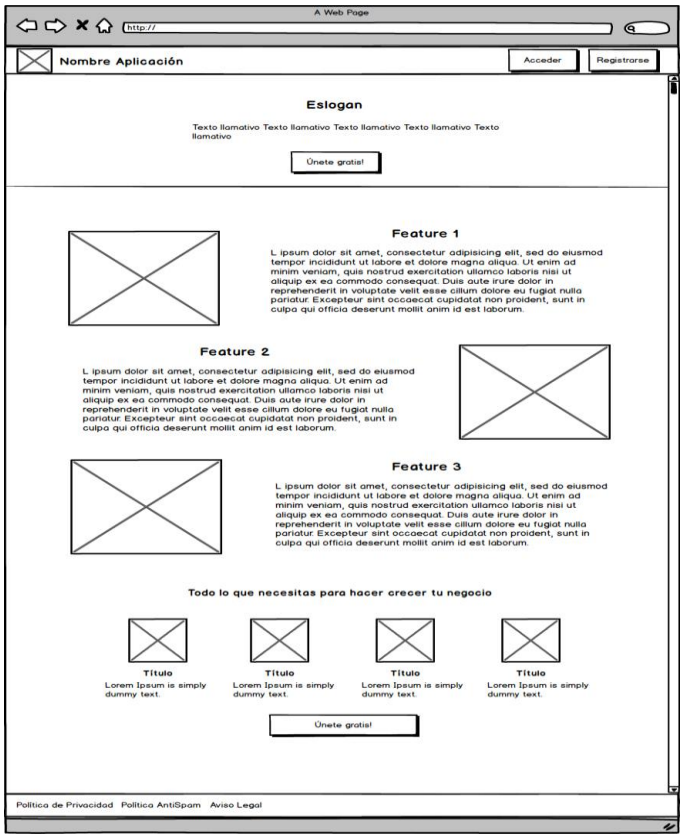


Ilustración 6: Página principal

El usuario podrá importar sus contactos de varias maneras.

- Copiando y pegando de un archivo de texto o bien de una hoja de cálculo (Ilustración 7)
- Subiendo un fichero .csv (Ilustración 8)

Hecho esto se cargarán las diferentes columnas del fichero y el usuario deberá vincular dichas columnas con el texto de cabecera correspondiente.

Antes de importar los contactos deberá dejar claro si pertenecen a un grupo de usuarios suscritos o dados de baja y añadir los tags que considere necesario. (Ilustración 9)

Los tags ayudarán posteriormente en la selección del segmento de audiencia al que se quisiera enviar la campaña de correo. También sirven para filtrar en la lista de contactos.

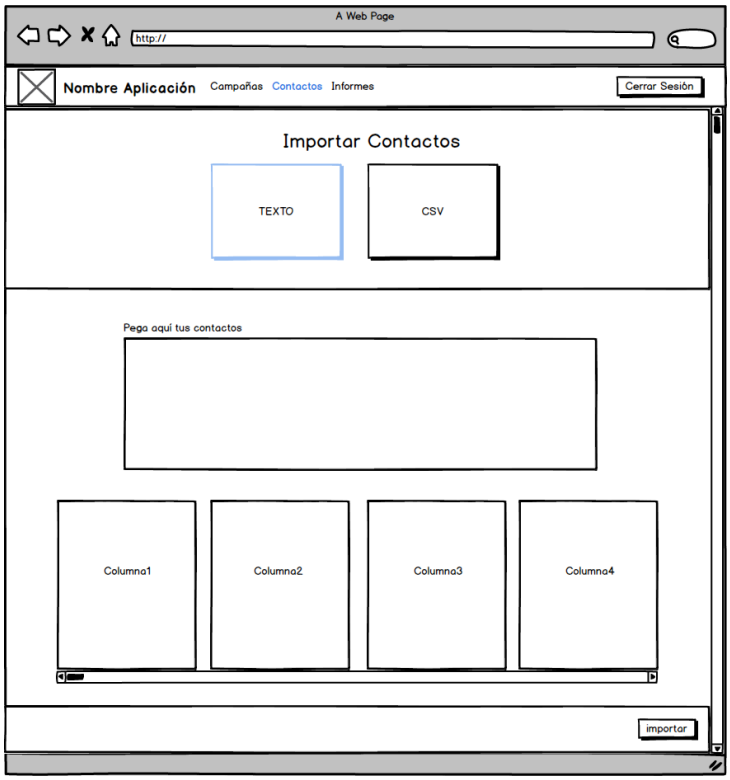


Ilustración 7: Importar contactos mediante texto

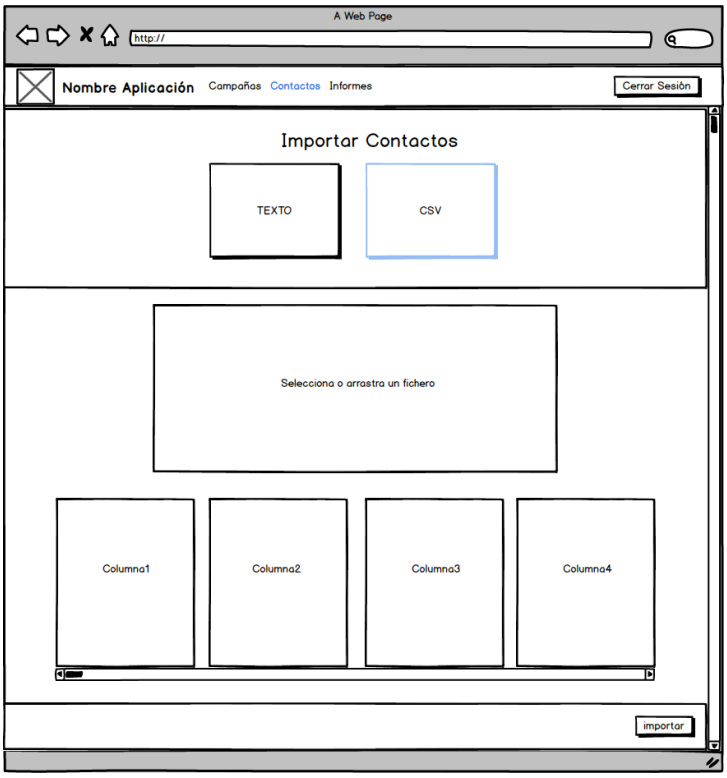


Ilustración 8: Importar contactos mediante archivo

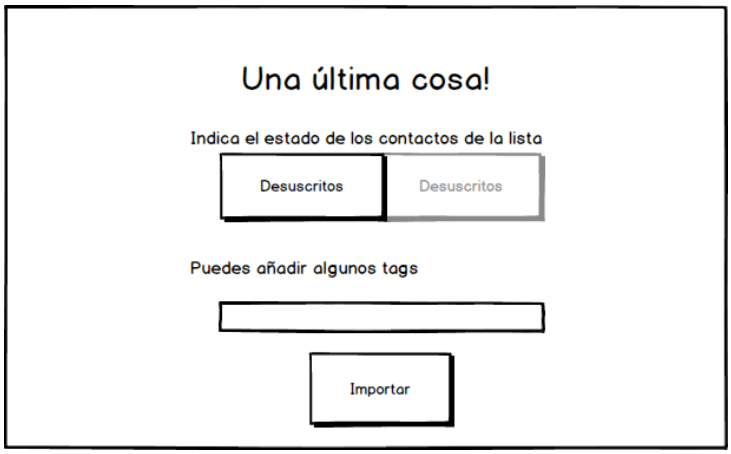
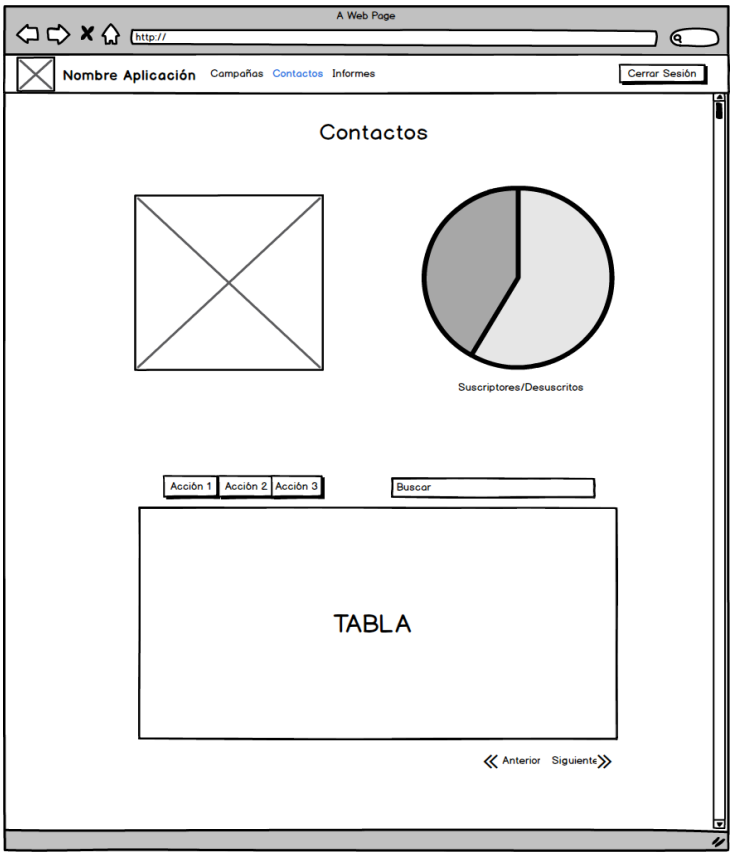


Ilustración 9: Importar contactos: confirmación

Podremos acceder a nuestros contactos a través del menú. Los contactos se mostrarán en una tabla (Ilustración 10), la cual tendrá la opción de búsqueda rápida. La tabla tendrá varias opciones básicas (añadir, editar y eliminar contacto). También será posible ocultar y mostrar columnas.

El gráfico de esta vista indicará el número de contactos suscritos en contraposición con los contactos que se han dado de baja de nuestra lista.



lustración 10: Mis contactos

Desde el apartado de campañas (Ilustración 11), accesible desde el menú, es posible acceder a una campaña creada previamente ya sea para reenviarla o para editarla. También es posible crear una nueva campaña.

Ilustración 11: Mis campañas

Al seleccionar “Crear Campaña”, se debe seleccionar el tipo de campaña que vamos a crear (Ilustración 12).

- **Texto plano:** Escribimos directamente el texto.
- **Plantilla**: Creación del email a partir de una herramienta “*drag and drop*”.

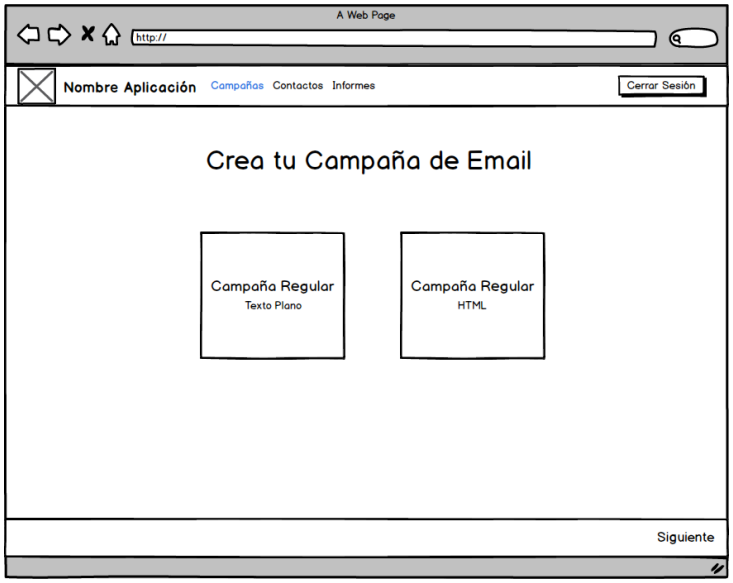


Ilustración 12: Creación de campaña: tipo de campaña

La primera parte a completar para enviar una campaña es la audiencia, dicho de otra manera, a qué contactos vamos a enviar nuestra campaña.

- **Todos los contactos** (Ilustración 13)
- ***Tags* (Etiquetas)**: Bien sean agregados por el usuario en la importación de los contactos o etiquetas predefinidas. (Ilustración 14)
- **Audiencia Personalizada**: Permite la selección de un tag creado por el usuario, un operador (mayor, menor, igual, mayor o igual…) y una fecha. La fecha corresponde a todos los envíos que se ha hecho de la campaña en cuestión sobre dicha etiqueta. (Ilustración 15)

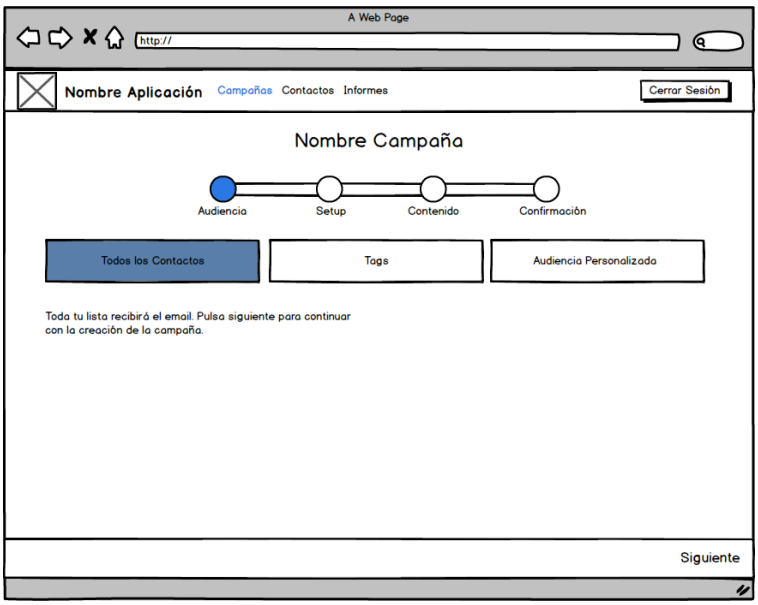


Ilustración 13: Creación de campaña: todos los contactos

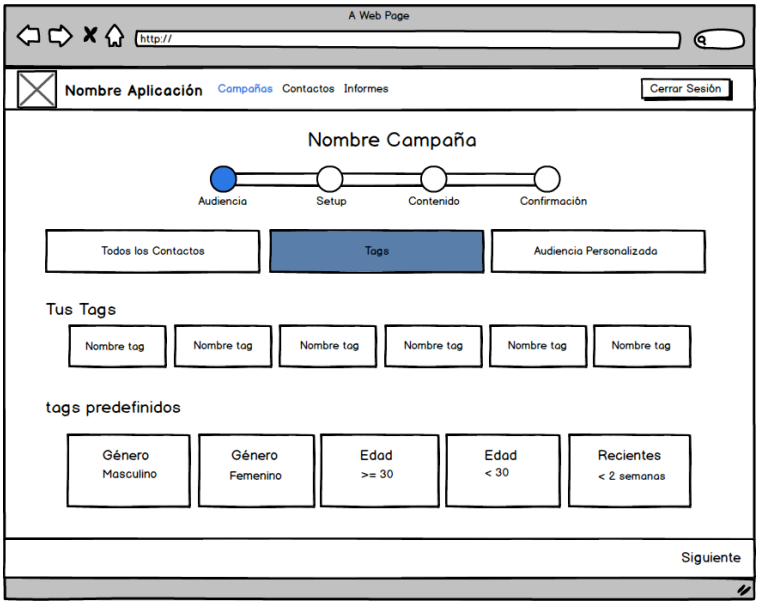


Ilustración 14: Creación de campaña: tags

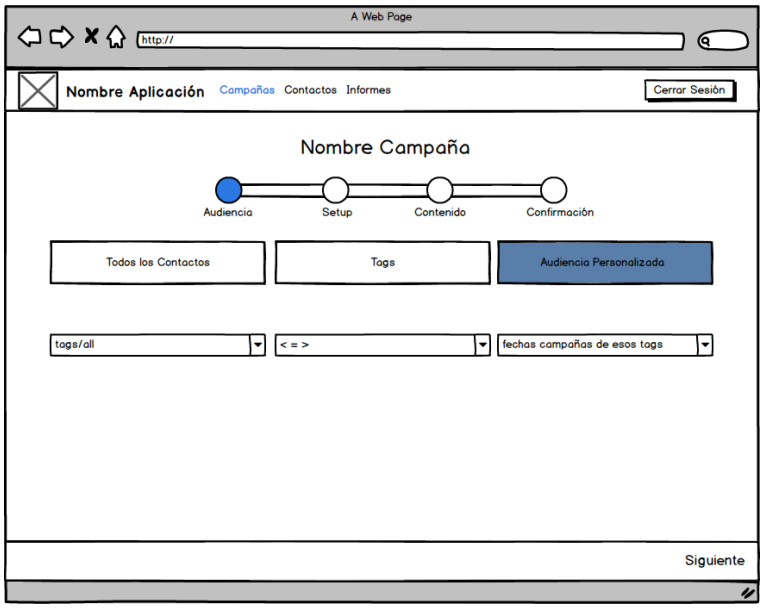


Ilustración 15: Creación de campaña: audiencia personalizada

El siguiente paso es completar la información necesaria para el envío de la campaña (Ilustración 16). Como bien puede ser: el email desde el que se envía la campaña, asunto, texto que se mostrará en la bandeja de entrada…

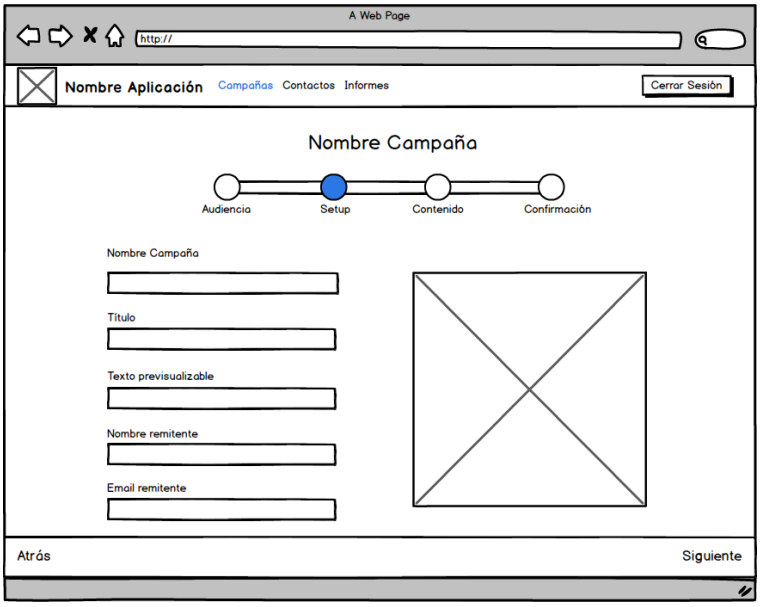


Ilustración 16: Creación de campaña: configuración

Antes de poder enviar la campaña tenemos que escribir el contenido que queremos enviar a nuestra audiencia (Ilustración 17) o bien crear el email mediante plantilla (Ilustración 18). Se podrá enviar un grupo de variables que en el momento de enviar el email se sustituirán el valor del contacto correspondiente (nombre, edad, email…).

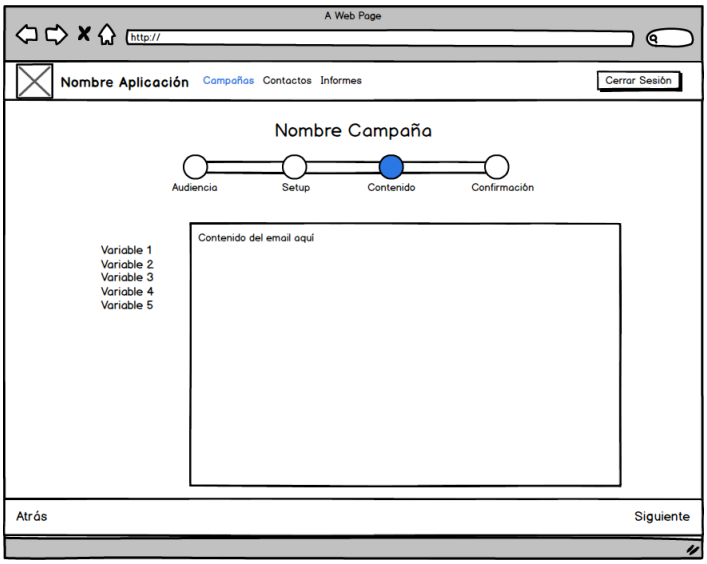


Ilustración 17: Creación de campaña: contenido mediante texto plano

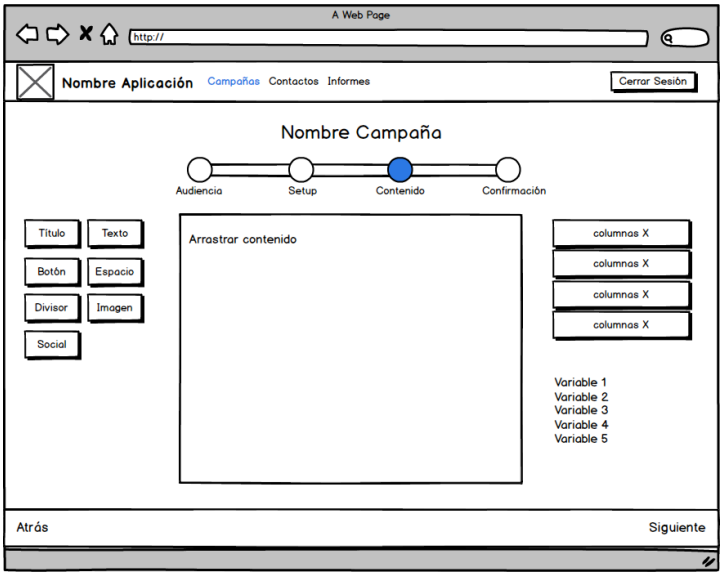


Ilustración 18: Creación de campaña: contenido mediante plantilla

El último paso es enviar la campaña (Ilustración 19). Si hay algún paso por completar no será posible enviar la campaña, en su lugar aparecerán los errores a corregir. Una vez enviada la campaña correctamente se redirige al usuario al apartado **"**Mis Campañas**".**

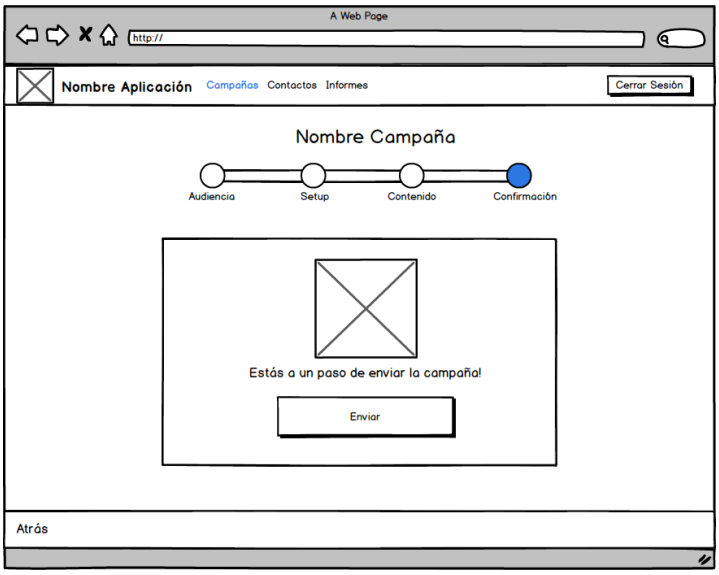


Ilustración 19: Enviar campaña

Tras realizar el envío de una campaña será posible acceder a su informe (Ilustración 20), el cual se encontrará dentro del apartado de informes.

Ilustración 20: Informes de campaña

4.3. Tecnología utilizada

Las tecnologías empleadas en el lado del cliente, también llamado *frontend*, han sido html5, CSS3 y JavaScript (Ilustración 21).

- **HTML5 (*HyperText Markup Language*, versión 5)**: Es un lenguaje de marcas utilizado para la estructuración y presentación del contenido de páginas web.

- **CSS3 (*Cascading Style Sheets*):** Lenguaje orientado a la definición de la parte visual de la web.

- **JavaScript**: Lenguaje de programación basado en el estándar ECMAScript. Se utiliza con el fin de otorgar funcionalidad a los componentes.

Ilustración 21: Logotipos de HTML, JavaScript y CSS3

Mean Stack es un conjunto de tecnologías basadas en JavaScript orientadas al desarrollo de aplicaciones web. Mean es el acrónimo de: MongoDB,ExpressJS, AngularJS, NodeJS.

AngularJS[5] (Ilustración 22) es una librería de JavaScript utilizada para el desarrollo web en el lado del cliente que nos brinda un *framework* para la creación de aplicaciones SPA (*Single-Page Applications*). La manera de trabajar es por componentes, maximizando así la reutilización de código.

Ilustración 22: Logotipo de AngularJS

NodeJS[6] (Ilustración 23) es un entorno en tiempo de ejecución de JavaScript lo que quiere decir que abarca todo lo necesario para ejecutar un programa escrito en este lenguaje. Cuenta con las siguientes características:

- Asíncrono
- Escalable
- Muy rápido
- Licencia MIT

En nuestro caso es especialmente útil su gestor de paquetes npm[7]: el repositorio de librerías *open source* (código abierto) más grande hasta el momento.

Ilustración 23: Logotipo de NodeJS

[5] https://angularjs.org/
[6] https://nodejs.org/es/
[7] https://www.npmjs.com/

Exspress[8] (Ilustración 24) es un *framework* minimalista desarrollado sobre NodeJS cuyo objetivo es ayudarnos en la creación de aplicaciones web.

Algunos de los puntos que hacen a Express el *framework* por excelencia para la creación de aplicaciones web en el lado de un servidor con NodeJS son:

- Nos proporciona herramientas para el enrutamiento de direcciones url.
- Creación de APIs RESTful robustas de manera fácil.
- Estructura MVC.
- Operaciones I/O muy rápidas.

Ilustración 24: Logotipo de Express.js

MongoDB[9] (Ilustración 25) es una base de datos NoSQL, lo que quiere decir que difiere del modelo clásico el cual estaba centrado en las relaciones entre tablas. En este nuevo modelo se almacenan directamente los datos en documentos JSON, lo cual lo hace ideal para lidiar con grandes cantidades de datos.

La desventaja más grande de este tipo de bases de datos es que no garantizan atomicidad a nivel de operación, como si lo hacen las bases de datos SQL. La atomicidad garantiza que en caso de error durante una operación se vuelva a un estado previo a la realización de la misma.

Ilustración 25: Logotipo de MongoDB

Gulp[10] (Ilustración 26) es lo que se conoce como un manejador de tareas. Una herramienta que requiere de NodeJS y permite automatizar tareas comunes de desarrollo que pueden llegar a ser tediosas.

En nuestro caso se ha utilizado con las siguientes finalidades:

- Como preprocesador para pasar código sass a css
- Preparar el entorno de producción: minificar, concatenar archivos, eliminar comentarios y crear la estructura de carpetas de nuestro proyecto en fase de producción.

[8] https://expressjs.com/es/
[9] https://www.mongodb.com/
[10] https://gulpjs.com/

Ilustración 26: Logotipo de Gulp

Sass[11] (Ilustración 27) en un lenguaje de hoja de estilos. Gracias a su sintaxis permite la simplificación de los archivos CSS. Existen dos sintaxis distintas asociadas a Sass, la más reciente se llama SCSS. Para transformar dicha sintaxis a CSS se utiliza un preprocesador, en nuestro caso hemos utilizado Gulp.

Ilustración 27: Logotipo de Sass

[11] https://sass-lang.com/

5. Organización del trabajo

Para la realización de la aplicación web se ha utilizado la metodología ágil SCRUM. En este apartado se va a mostrar la planificación realizada del trabajo. En total seis *sprints* de 2-3 semanas de duración con un peso total en horas de entre 50-68.

5.1. Sprint 1

En el primer sprint nos centraremos en la preparación del entorno de trabajo y la realización de la página principal, en la cual se incluye el registro y el inicio de sesión.

Las tareas que componen este sprint son las siguientes:

- Preparación del entorno de trabajo (3h).
- Elección del nombre de la aplicación (0.30h).
- Selección de los colores de marca (0.45h).
- Diseño de la página principal (12h).
- Creación del logotipo (4h).
- Diseño de la vista de inicio de sesión (5h).
- Validación de campos en el inicio de sesión (1.45h).
- Diseño de la vista registro (5h).
- Validación de campos en el registro (1.45h).
- Autenticación de usuarios (15h).
- Diseño de la cabecera del interior de la web (3h).

5.2. Sprint 2

En el segundo sprint se hace especial hincapié en todo lo referente a los contactos, desde la visualización hasta la importación de los mismos.

Las tareas que componen este sprint son las siguientes:

- Diseño de la vista contactos (15h).
- Creación del gráfico de la vista contactos (4h).
- Mostrar contactos (11h).
- Filtrar contactos (1h).
- Añadir un único contacto (5.30h).
- Editar contactos (8h).
- Diseño de la vista de importar contactos (8h).
- Subir archivos .csv (6h).
- Copiar y pegar texto de contactos (8h).
- Crear las redirecciones al interior de la aplicación (1.5h).

5.3. Sprint 3

En el tercer sprint nos enfocaremos en la creación de las campañas de email y en un primer envío básico, que corresponde al envío de texto plano (con variables).

Las tareas que componen este sprint son las siguientes:

- Elección del servicio para el envío de emails (2.30h).
- Diseño de las vistas de creación de campañas (16h).
- Redirección entre fases de la creación (1h).
- Filtrar contactos por segmentos de tags y fechas (8h).
- Sustituir variables del contenido del mensaje en el servidor (1.30h).
- Creación de email *responsive* (3.30h).
- Validación de campos de la campaña (4h).
- Envío de campaña a través de Gmail (6.30h).
- Añadir link para darse de baja de la lista (3h).
- Reutilización de campañas (6h).
- Diseño de la vista de consultar campañas (4h).
- Marcar campaña como favorita (2h).

5.4. Sprint 4

En el cuarto sprint nos conectaremos con el servicio externo para enviar mensajes, por lo que también tendremos que hacer un despliegue temprano de una API nuestra que permita recibir la información que nos envíe dicho servicio. Además, crearemos los informes para visualizar la información de los envíos de las campañas.

Las tareas que componen este sprint son las siguientes:

- Envío de campaña por el servicio externo (8h).
- Creación de la API (10h).
- Despliegue temprano de la API y la BDD (9h).
- Diseño de la vista de informes (12h).
- Funcionalidad de informes (8h).
- Diseño de la vista de consultar envíos (3h).
- Realizar seguimiento de envío (1h).

5.5. Sprint 5

El quinto sprint es el último antes de pasar la aplicación a fase de producción. En este sprint vamos a añadir la funcionalidad del creador de plantillas, con la que el usuario podrá crear emails arrastrando una serie de componentes que le facilitaremos (texto, imágenes, botones…) hacia un cuadro en blanco.

Las tareas que componen este sprint son las siguientes:

- Diseño de los distintos componentes del creador de plantillas (5h).
- Agregar columnas al cuadro del contenido (8.30h).
- Permitir arrastrar componentes y columnas al cuadro del contenido (4h).
- Funcionalidad de los distintos componentes (15h).
- Subir imágenes (5h).
- Permitir modificar ciertos atributos css (10h).
- Convertir el contenido a una estructura de email *responsive* (12h).
- Guardar y cargar plantilla (5h).

5.6. Sprint 6

En este último sprint nos vamos a centrar en el despliegue de la aplicación para su producción. Por lo tanto, tendremos que seleccionar dominio y hosting y llevar el código de nuestra aplicación minificado, concatenado y ofuscado para que esté lo más optimizada posible.

Las tareas que componen este sprint son las siguientes:

- Pruebas finales de funcionamiento de la aplicación (15h).
- Llevar la aplicación a producción (15h).
- Elección de hosting (1.30h).
- Elección de nombre de dominio (0.30h).
- Despliegue de la aplicación en la nube (15h).
- Obtención de un certificado SSL (4h).

6. Desarrollo de la solución propuesta

El objetivo de este apartado es explicar los puntos que han sido considerados de especial relevancia para llevar acabo el desarrollo de la solución.

6.1. Servicio externo de mensajería mailgun

Para el desarrollo de la aplicación se ha utilizado un servicio externo llamado mailgun[12] (Ilustración 28). Este servicio permite el envío masivo de emails, además también permite recoger datos de interés de los correos enviados (si se ha abierto, si la persona se ha dado de baja de la lista, si ha clicado en algún link…). Esto es posible gracias a que cuenta con distintos *webhooks*, los cuales podemos configurar para recibir los distintos datos de interés en los diferentes puntos de entrada de nuestra aplicación.

Ilustración 28: Logotipo de mailgun

Los *webhooks* se suele confundir frecuentemente con el concepto de API [2], no obstante, un *webhook* no es más que un método que se desencadena cuando sucede algo y que entonces y solo entonces hace una llamada HTTP a donde le hayamos indicado. Como ejemplo, cuando el usuario abre un email se desencadenaría un *webhook* y este nos avisaría de dicha apertura mediante una llamada HTTP a nuestra API.

Los motivos por los que se ha decido utilizar mailgun y no otro servicio similar son los siguientes:

- Cuenta con *webhooks* para el envío de los datos de interés.
- API para nodejs.
- Documentación muy detallada tanto de la API como de los *webhooks*.
- 10000 envíos de emails gratis al mes.

Una vez hecha la parte del envío deberemos realizar la API para la gestión de la información que nos enviarán los diferentes *webhooks* (Ilustración 29) y hacerla accesible a través de internet. Para hacerla accesible a través de internet se decidió hacer un despliegue temprano de dicha API y de la BDD en Amazon AWS, el cual se explica con más detalle en el siguiente apartado.

[12] https://www.mailgun.com/

```
//mailgun webhooks
app.post('/mailgun/open', mailgun.openEmail);
app.post('/mailgun/spam', mailgun.spamEmail);
app.post('/mailgun/clicked', mailgun.emailClicked);
app.post('/mailgun/delivered', mailgun.emailDelivered);
app.post('/mailgun/softBounce', mailgun.softBounce);
app.post('/mailgun/failure', mailgun.failure);
```

Ilustración 29: Puntos de entrada de la API

Por último, configuramos los *webhooks* a través de la propia web de mailgun, tal y como se muestra en la Ilustración 30 para la acción de clicar enlace.

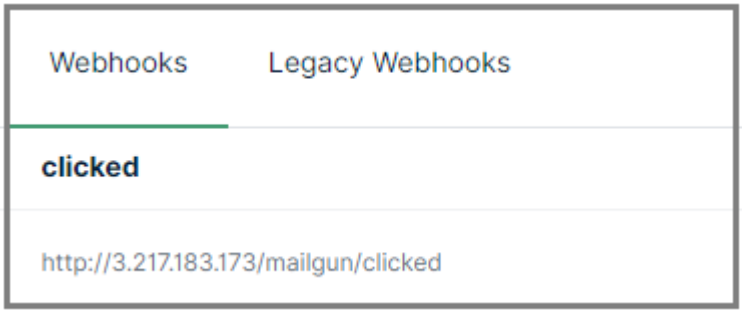


Ilustración 30: mailgun configuración

Nuestra API captaría los mensajes provenientes de mailgun y llamaría a su función correspondiente. Tal y como se muestra en la Ilustración 29 la función que se ejecutaría al clicar un enlace del email sería “emailClicked” (Ilustración 31). Tanto en el caso particular de clicar un enlace como en las otras funciones de nuestra API, básicamente lo que se hace es guardar dicha información en nuestra base de datos para su posterior análisis en la parte de informes de la web.

```
exports.emailClicked = async function(req, res) {
  let triggerContact = req.body['event-data']['recipient'],
    variables = req.body['event-data']['user-variables'],
    urlClicked = req.body['event-data']['url'];

  res.status(200).send({ success: true }); //return to webhook

  let dboUser = await User.findOne({ email: variables['user-email'] });

  var triggerContactObj = getTriggerContactObj(dboUser, variables, triggerContact);
  if (!triggerContactObj) { return };

  //si el link es unsuscribed
  if (urlClicked.includes('unsuscribe')) {
    triggerContactObj.unsuscribed = true;
    triggerContactObj.unsuscribedDate = moment();
  } else {
    triggerContactObj.clicks++;
    triggerContactObj.openedDate = moment();
  }

  saveSendedCampaign(dboUser);
};
```

Ilustración 31: Código de la función emailClicked

6.2. Autenticación mediante JSON Web Token

Otro de los puntos importantes en el desarrollo es la autenticación basada en tokens (JWT) [8]. Es la manera que tenemos de saber que quién hace una petición al servidor realmente tiene los privilegios necesarios para llevarla a cabo.

El servidor envía un *token* al usuario cuando inicia sesión y este se guarda en el almacenamiento local del navegador (Ilustración 32).

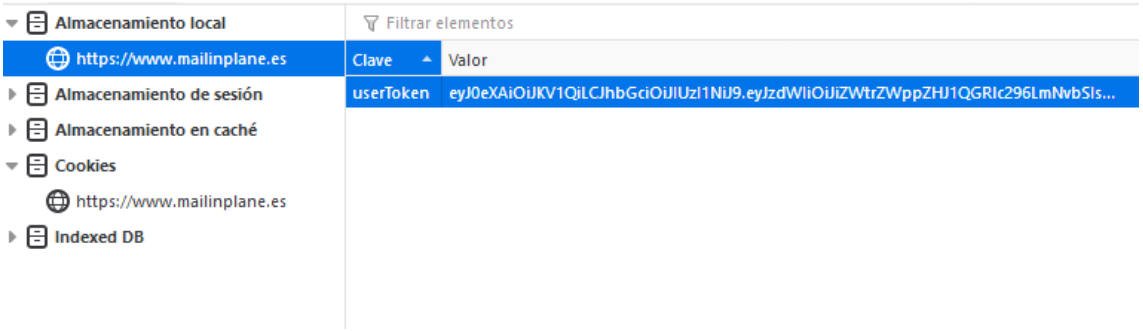


Ilustración 32: Token de usuario en Firefox

Este *token* contiene la información necesaria para poder identificar al usuario, dicha información llega encriptada desde el servidor. En el *frontend* se ha creado un interceptor para adjuntar el *token* en la cabecera de las peticiones http realizadas por el usuario (Ilustración 33).

```
(function() {
  var authInterceptor = function(authToken) {
    return {
      request: function(config) {
        var token = authToken.getToken();

        if (token) {
          config.headers.authorization = 'Bearer ' + token;
        }

        return config;
      },
      response: function(response) {
        return response;
      }
    };
  };

  app.factory('authInterceptor', ['authToken', authInterceptor]);

})();
```

Ilustración 33: Interceptor de peticiones http

En el servidor hemos hecho uso del módulo "jwt-simple[13]". Cuando nos llega una petición http del *frontend* que requiere autenticación se comprueba si el usuario realmente es quien dice ser. (Ilustración 34). Tanto la encriptación como la desencriptación del *token* se realiza con una clave que nosotros le indicamos (la variable "TOKEN_SIGN" contiene nuestra clave).

```
exports.isAuthorizated = async function(req, res, next) {
  if (!req.headers.authorization) {
    res.status(403).send({ message: 'UNAUTHORIZED', redirect: '/home' });
    return;
  };

  let token = req.headers.authorization.split(' ')[1];
  let payload = jwt.decode(token, TOKEN_SIGN);

  if (!payload.sub) {
    res.status(403).send({ message: 'UNAUTHORIZED', redirect: '/home' });
    return;
  }

  return dboUser;
}
```

Ilustración 34: JWT, comprobación de identidad

6.3. Cifrado de contraseñas con Bcrypt

Bcrypt es una función de *hashing* para contraseñas que está basada en Blowfish. Lo que distingue bcrypt de otras funciones hash es la velocidad, las funciones hash están hechas para ser rápidas, está cualidad se torna negativa en el ámbito web cuando se trata con contraseñas [5]. Con Bcrypt nos aseguramos que un ataque de fuerza bruta sea prácticamente imposible debido a su lentitud comparado con muchos de los otros algoritmos existentes.

En nuestra aplicación utilizamos el módulo llamado bcrypt[14], con el cual almacenamos el hash de la contraseña proporcionada por el usuario en el registro (Ilustración 35).

```
UserSchema.pre('save', function(next) {
  let user = this;

  // only hash the password if it has been modified (or is new)
  if (!user.isModified('password')) return next();

  //Hashear el password
  bcrypt.genSalt(SALT_WORK_FACTOR, function(err, salt) {
    if (err) return next(err);

    bcrypt.hash(user.password, salt, function(err, hash) {
      if (err) return next(err);

      user.password = hash;
      next();
    });
  });
});
```

Ilustración 35: Hash de contraseñas

[13] https://www.npmjs.com/package/jwt-simple
[14] https://www.npmjs.com/package/bcrypt

En la Ilustración 36 vemos cómo quedaría el campo contraseña tras aplicarle bcrypt sobre la cadena “12345678”.

Ilustración 36: Registro de BDD

6.4. Darse de baja de la lista de email

Algo importante a tener en cuenta, tal y como se recalca en el apartado 5 del capítulo 4 de la LOPD (Ley Orgánica de Protección de Datos), es el derecho del usuario que recibe nuestras campañas a darse de baja de las mismas.

Para cumplir con este punto nuestra aplicación añade en todos los correos un enlace para darse de baja de la lista de email (Ilustración 37).

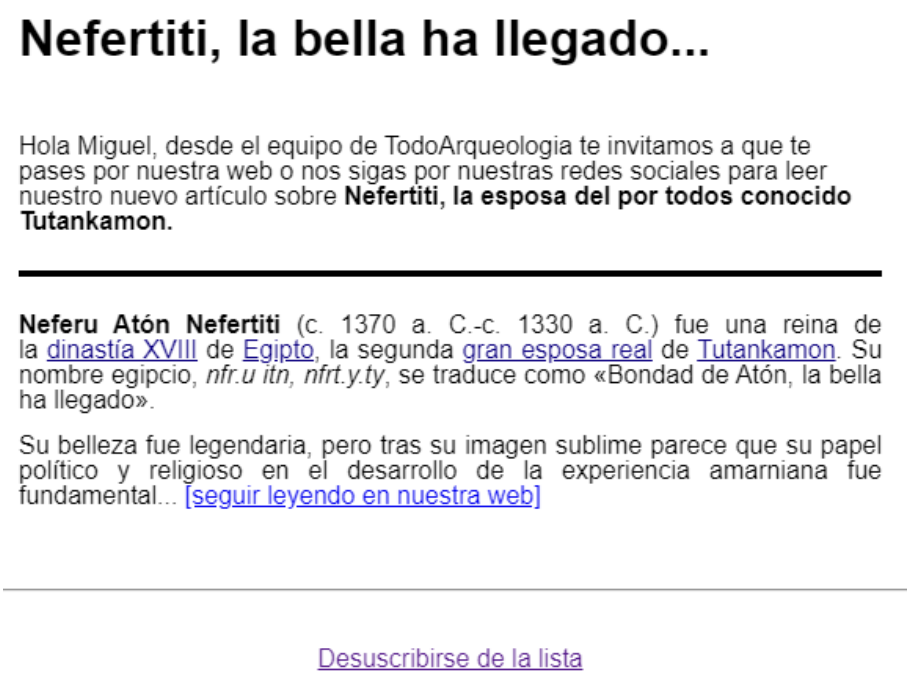


Ilustración 37: Correo enviado desde mailinplane

Si un usuario en cuestión decide darse de baja simplemente tendría que clicar el link que pone "Desuscribirse de la lista". Esta acción llama a un punto de entrada de nuestra API en el que se gestiona dicha baja (Ilustración 38). Tal y como se muestra en la ilustración de abajo, nuestro servidor desencadenará la función "unsuscribeContact" siempre que la petición a nuestra web tenga la estructura: "/contact/:email/unsuscribe/:token", donde ":email" será el correo electrónico del usuario que se va a dar de baja y ":token" la información encriptada del usuario, que previamente ha generado el servidor (apartado 6.2).

```
//Unsuscribe contact
app.get('/contact/:email/unsuscribe/:token', contacts.unsuscribeContact,
function(req, res, next) {
  res.sendFile(rootPath + '/server/views/unsuscribed.html');
});
```

Ilustración 38: Punto de entrada para la acción de darse de baja de la lista

El código relativo a "unsuscribeContact" se muestra en la Ilustración 39. Dicho código comprueba si el usuario ya se ha dado de baja previamente, si fuese así no haría nada, de lo contrario se marca en la base de datos la propiedad "suscribed" con el valor *false* y guarda la fecha en la que se produce dicho cambio en "desuscribedDate".

```
exports.unsuscribeContact = function(req, res, next) {
  var token = req.params.token;
  var contactEmail = req.params.email;

  var payload = jwt.decode(token, config.EMAIL_SECRET);
  var email = decodeURIComponent(payload.sub);

  if (!email) { return console.log('error') }

  User.findOne({ email: email }, function(err, dboUser) {
    if (err) { return console.log('error') }

    let isSuscribed = dboUser.contacts[contactEmail].suscribed;
    if (!isSuscribed) { next() }

    dboUser.contacts[contactEmail].suscribed = false;
    dboUser.contacts[contactEmail].desuscribedDate = moment().format("DD/MM/YYYY");
    dboUser.markModified('contacts');

    dboUser.save(function(err, user) {
      if (err) { return res.status(500) }
      next();
    });
  });
};
```

Ilustración 39: Código de la función unsuscribeContact

Tras completarse el proceso de baja a nivel interno se le muestra al usuario una pantalla en la que se le informa que el proceso se ha completado de manera satisfactoria (Ilustración 40).

Vaya! Sentimos que te vayas de nuestra lista.

Te has desuscrito correctamente de nuestra lista.

Ilustración 40: Pantalla de información de baja de la lista

Ahora en nuestra aplicación el estado del usuario será “desuscrito” (Ilustración 41). Esto quiere decir que la aplicación no mandará emails a este usuario en los posteriores envíos que se realicen.

Ilustración 41: Contacto dado de baja de la lista

7. Implantación

En el siguiente apartado nos centraremos en explicar los pasos realizados para llevar el proyecto a la fase de producción. Este proceso implica tanto la optimización del código como el despliegue de la solución final en la nube.

7.1. Optimización del código

Cuando hablamos de optimización del código nos referimos a la puesta a punto previa al despliegue de la aplicación web. En nuestro caso esta optimización comprende el proceso de minificación, ofuscación y concatenación del código.

- **Minificación**: Reducir el código a su mínima expresión. Esto se logra mediante la eliminación de espacios, saltos de línea y comentarios.

- **Ofuscación**: Alterar el código fuente para que sea difícil de entender. En nuestro caso mediante la sustitución del nombre de las variables locales y funciones privadas.

- **Concatenación**: Agrupar todo el código en el menor número de archivos posibles para minimizar el número de peticiones de recursos por parte del cliente.

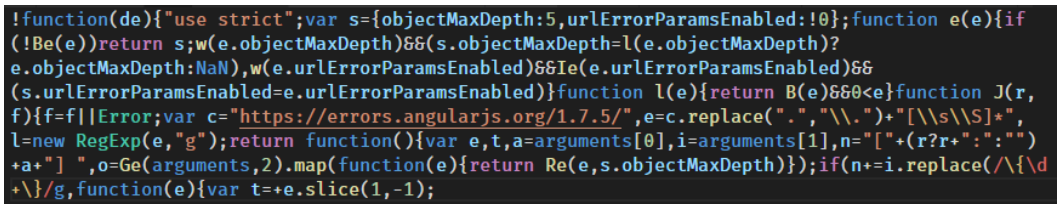
Para lograr dicha optimización se ha hecho uso de la herramienta Gulp junto con los módulos "gulp-uglify[15]" y "gulp-concat[16]". Gulp nos permite automatizar tareas comunes en el desarrollo de aplicaciones. Como se aprecia en la Ilustración 42 Gulp ejecutará para nuestros archivos JavaScript las tareas de concatenar, minificar, ofuscar y por último renombrar el archivo resultante.

```
gulp.task('js', function() {
  return gulp.src(JS_PATH)
    .pipe(gulpif(enviorment.isProduction(), concat('bundle.js')))
    .pipe(gulpif(enviorment.isProduction(), uglify()))
    .pipe(gulpif(enviorment.isProduction(), rename('bundle.min.js')))
    .pipe(gulp.dest('./dist'));
});
```

Ilustración 42: Gulp aplicado a archivos js

El resultado sería un archivo JavaScript llamado "bundle.min.js" con todo el código minificado y ofuscado tal y como se muestra en la Ilustración 43 (código original perteneciente a las primeras líneas de la librería de angularjs v1.7.5).

[15] https://www.npmjs.com/package/gulp-uglify
[16] https://www.npmjs.com/package/gulp-concat



```
!function(de){"use strict";var s={objectMaxDepth:5,urlErrorParamsEnabled:!0};function e(e){if
(!Be(e))return s;w(e.objectMaxDepth)&&(s.objectMaxDepth=l(e.objectMaxDepth)?
e.objectMaxDepth:NaN),w(e.urlErrorParamsEnabled)&&Ie(e.urlErrorParamsEnabled)&&
(s.urlErrorParamsEnabled=e.urlErrorParamsEnabled)}function l(e){return B(e)&&0<e}function J(r,
f){f=f||Error;var c="https://errors.angularjs.org/1.7.5/",e=c.replace(".","\\.")+"[\\s\\S]*",
l=new RegExp(e,"g");return function(){var e,t,a=arguments[0],i=arguments[1],n="["+(r?r+":":"")
+a+"] ",o=Ge(arguments,2).map(function(e){return Re(e,s.objectMaxDepth)});if(n+=i.replace(/\{\d
+\}/g,function(e){var t=+e.slice(1,-1);
```

Ilustración 43: Ejemplo de minificación y ofuscación

7.2. Despliegue en Amazon AWS

Para el despliegue de nuestra aplicación se ha utilizado *Amazon Web Services* (AWS)[17]. Amazon AWS se define a sí misma como una plataforma de servicios en la nube que cuenta con una gran variedad de servicios de infraestructura.

En nuestro caso los servicios utilizados han sido:

- **Route 53**: Nos permite registrar y configurar dominios.
- **Amazon EC2**: Amazon nos proporciona servidores virtuales en los cuales podemos ejecutar nuestras propias aplicaciones. También llamados instancias.
- **Buckets de S3**: Nos permite compartir archivos de nuestro ordenador a las diferentes instancias que tengamos habilitadas.

7.2.1. Registro de dominio

Para nuestra aplicación el dominio seleccionado ha sido mailinplane.es. Estamos ante un dominio nacional dado que la terminación corresponde a un país (España). No ha habido un criterio especial para elegir un dominio nacional ante uno internacional, para el caso de estudio se considera que ambos son igual de válidos.

7.2.2. Instancias

Para el proyecto contamos con dos instancias, una para la base de datos y otra para el servidor web. Ambas instancias se han creado bajo un sistema Ubuntu Server 16.04 LTS.

Los servidores virtuales sobre los cuales están montados el servidor web y la base de datos corresponden a la opción gratuita que facilita Amazon (*free tier*), por lo que sus especificaciones son estándar, válidas para aplicaciones pequeñas y medianas (según la propia Amazon).

[17] https://aws.amazon.com/es/

7.2.3. Puesta en marcha

Tras crear las instancias e instalar la base de datos y los programas necesarios para correr nuestro servidor web, llevamos nuestra solución optimizada a su instancia en Amazon AWS.

Para ello hay varias maneras de hacerlo, la solución que hemos utilizado ha sido mediante los *buckets*, que permiten subir archivos a la nube y acceder a ellos desde las instancias.

Una vez desplegado el servidor y puesto en marcha quedaría enlazar el dominio con la instancia donde se encuentra el servidor web. Para ello hay que agregar dicha relación en el dns de nuestro dominio.

Una vez terminado todo este proceso ya podemos acceder a nuestra web a través de su nombre de dominio como se muestra en la Ilustración 44.

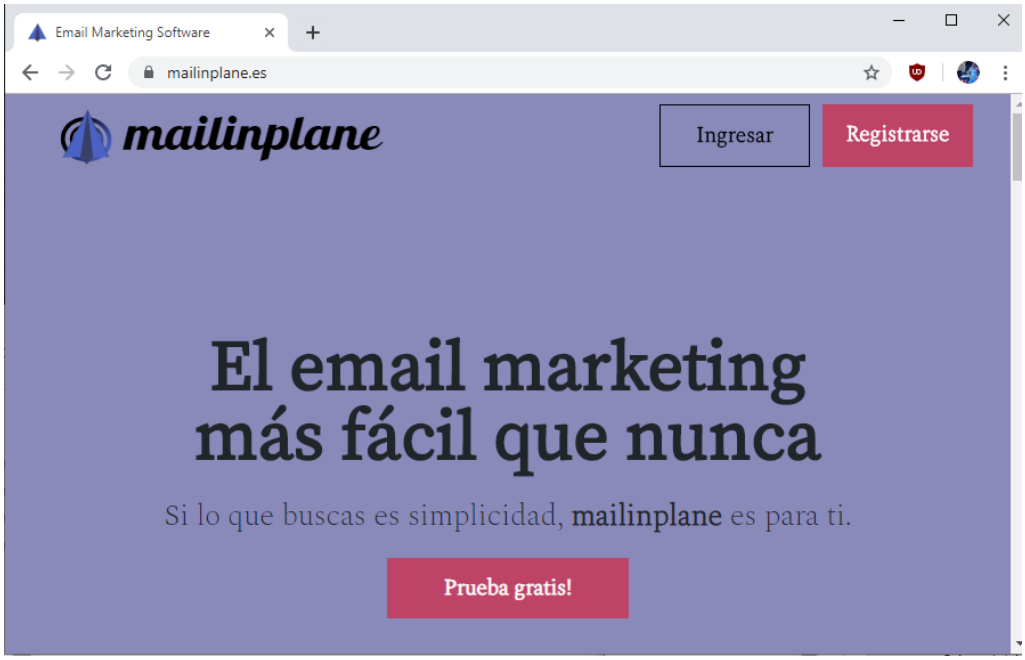


Ilustración 44: página principal de mailinplane.es

7.3. Certificado SSL

Un certificado SSL permite cifrar las conexiones entre cliente y servidor por lo que se reduce mucho la posibilidad de que alguien intercepte y extraiga información de dicho flujo de mensajes.

Es por tanto muy importante obtener un certificado SSL para nuestra página web. Amazon AWS nos lo pone bastante fácil, solo hace falta que lo solicitemos y en un par de días podremos enlazar el certificado a una de nuestras instancias.

Tal y como se muestra en la Ilustración 44 nuestra página cuenta con un certificado SSL. En la Ilustración 45 se puede ver con más detalle la información acerca de este certificado.

Ilustración 45: Certificado SSL

8. Pruebas

El objetivo de este punto es comprobar si realmente se han cumplido los requisitos funcionales y no funcionales planteados previamente en el análisis de requisitos (apartado 3 de este mismo documento).

8.1. Requisitos funcionales

Los requisitos funcionales han sido probados uno por uno validando sus casos de uso. En este apartado a través de pruebas de uso vamos a validar los requisitos más esenciales de la aplicación.

Todos los requisitos se han probado en las últimas versiones de los navegadores: Firefox, Google Chrome y Microsoft Edge.

8.1.1. Registro y autenticación

La vista mostrada en la Ilustración 46 corresponde con el requisito RF01. Como se muestra en la imagen para registrarse en la aplicación solo se ha de facilitar un email válido y una contraseña (mínimo 8 caracteres).

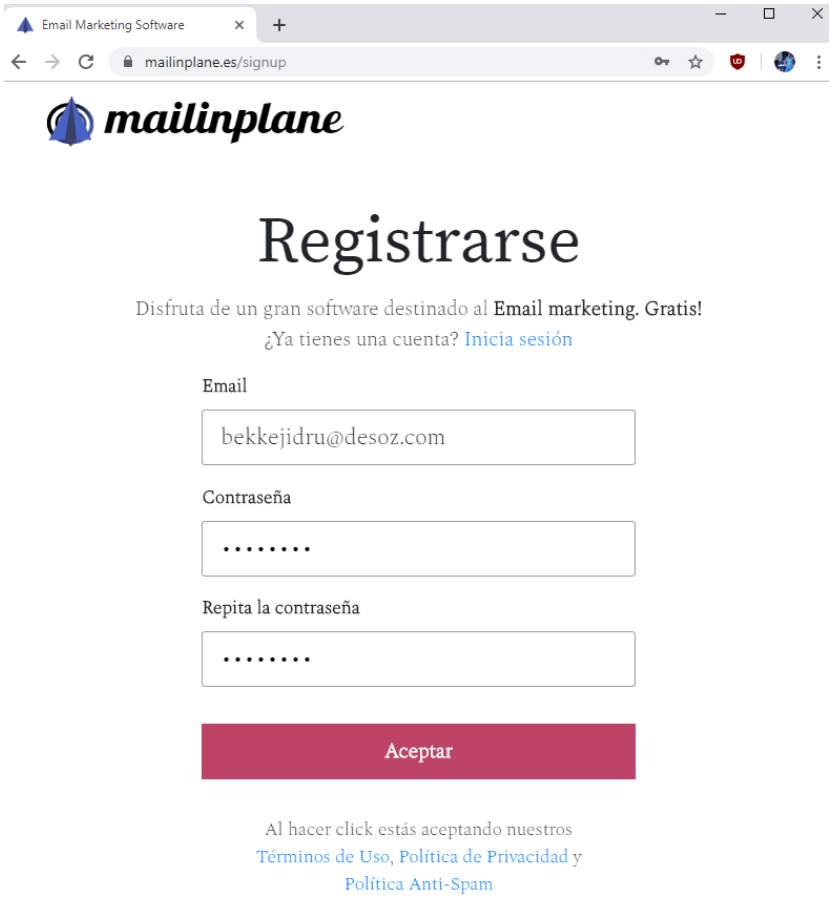


Ilustración 46: Vista de Registro

Una vez registrados la aplicación mandará un email de validación al correo facilitado en el paso previo. La Ilustración 47 muestra la pantalla que se le presentará al usuario hasta que valide la cuenta.

Ilustración 47:Vista de confirmación

La pantalla de inicio de sesión ha de permitir a los usuarios ya registrados entrar a la aplicación. En la Ilustración 48, el usuario con email “bekkejudru@desoz.com” registrado y validado previamente inicia sesión de manera satisfactoria (Ilustración 49), cumpliendo así el requisito RF02.

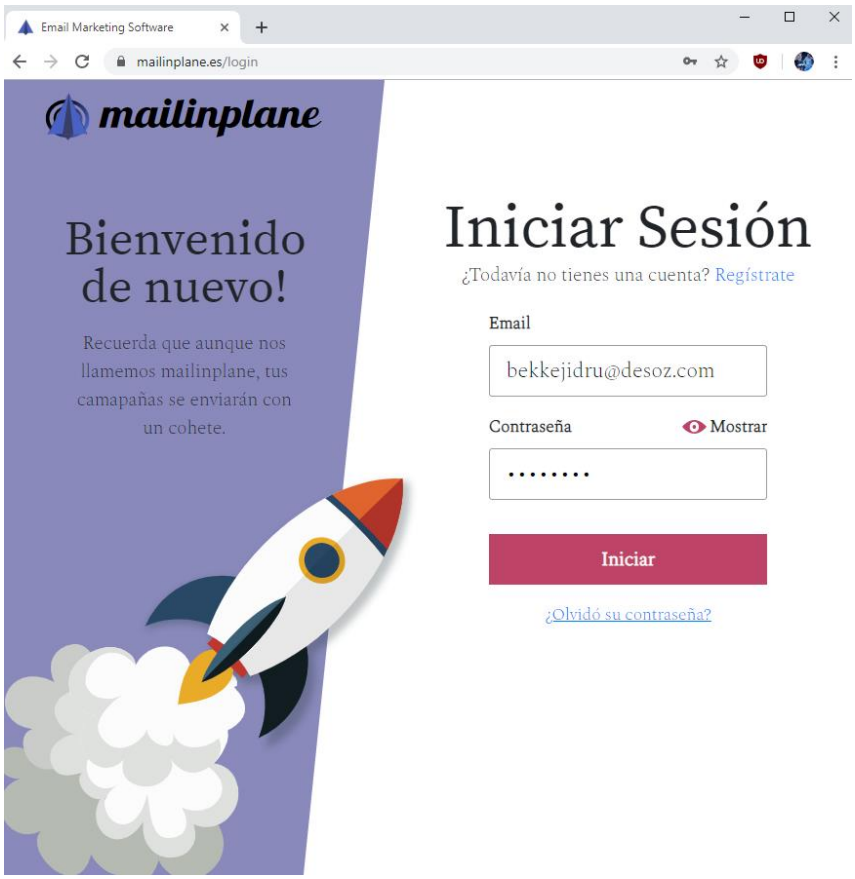


Ilustración 48: Vista de login

Para cumplir el requisito RF03 se ha colocado un botón en la cabecera. Al pulsar el botón se borrará el token de acceso del almacenamiento local del navegador y se redirigirá al usuario a la página principal de la aplicación.

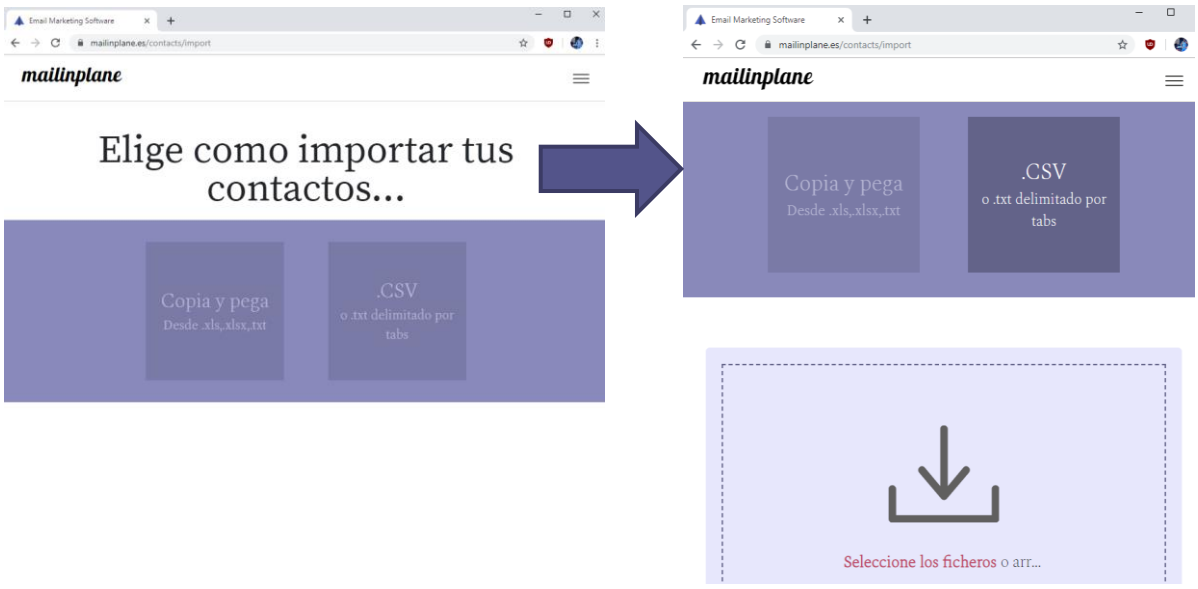
Ilustración 49: Vista de contactos

8.1.2. Contactos

En la Ilustración 50 se observa una tabla de contactos vacía.

Ilustración 50: Vista de lista de contactos

A continuación, pasamos a importar una lista de contactos (RF08). Para ello se han seguido una serie de pasos, tal y como se muestra en la Ilustración 51. En el último paso, a la hora de importar los contactos, la aplicación le mostrará al usuario una ventana modal para que añada etiquetas y marque el estado actual de los contactos de la lista (RF09, RF10).



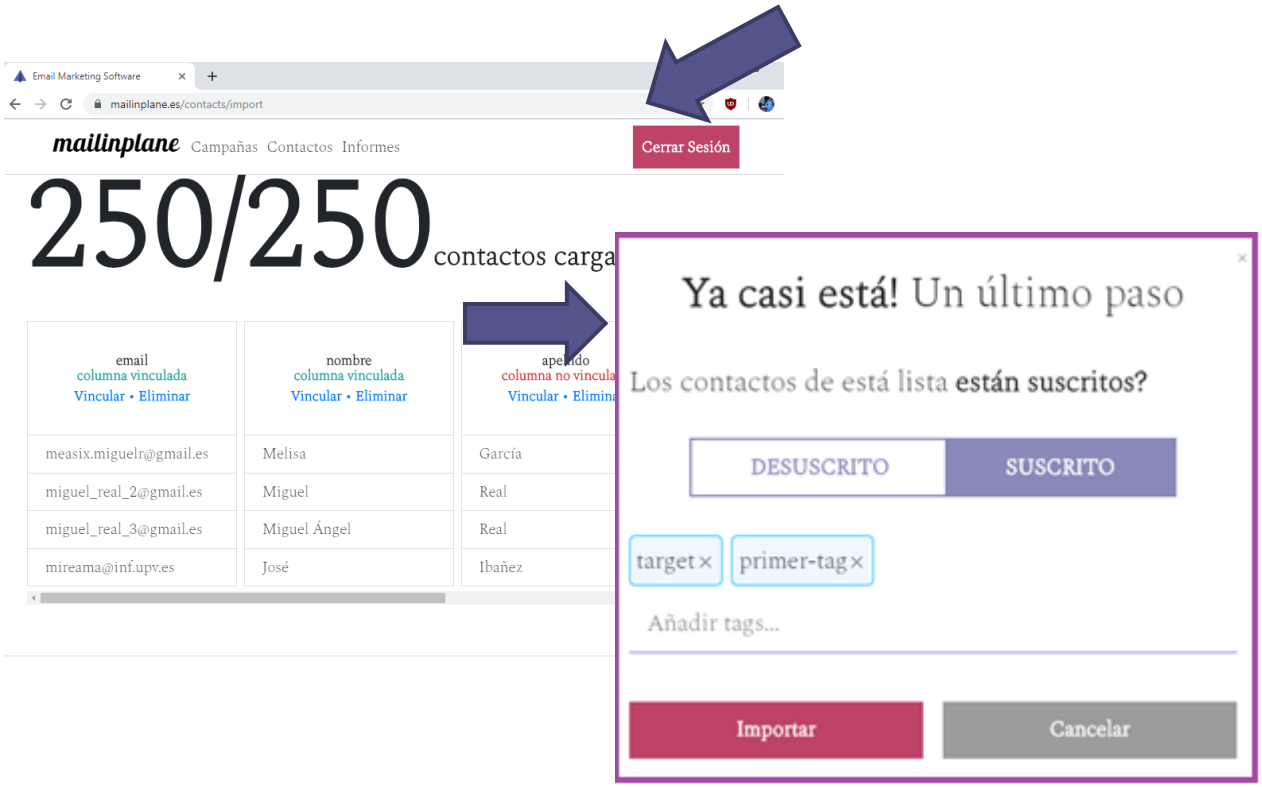


Ilustración 51: Vista de importar contactos

Una vez importados los contactos la tabla se llenará con información relativa a los mismos (Ilustración 52) cumpliendo así el requisito RF04.

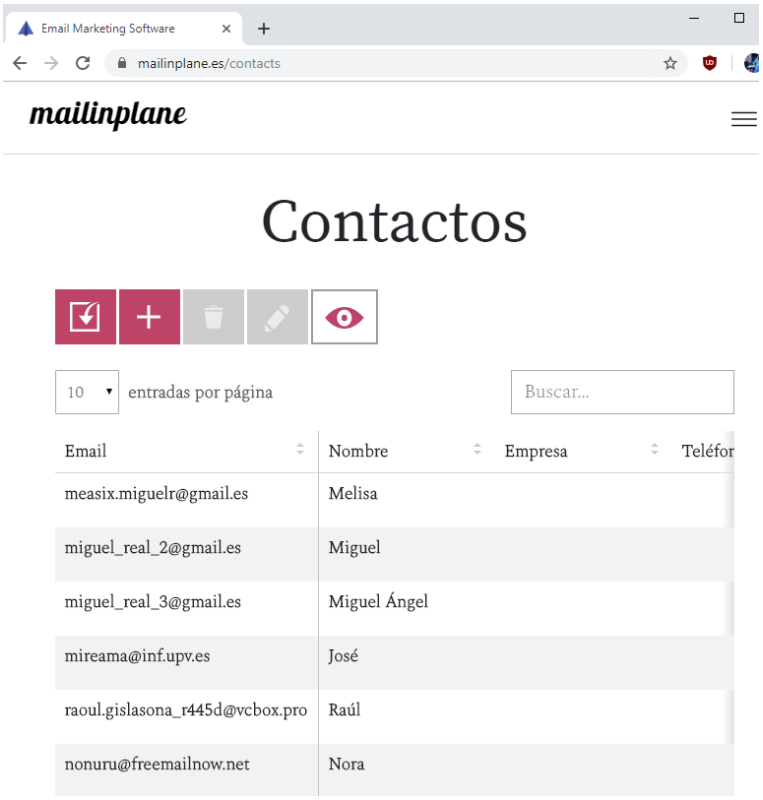


Ilustración 52: Vista de lista de contactos II

También nos es posible realizar búsquedas de contactos tal y como se muestra en la Ilustración 53 (RF05)

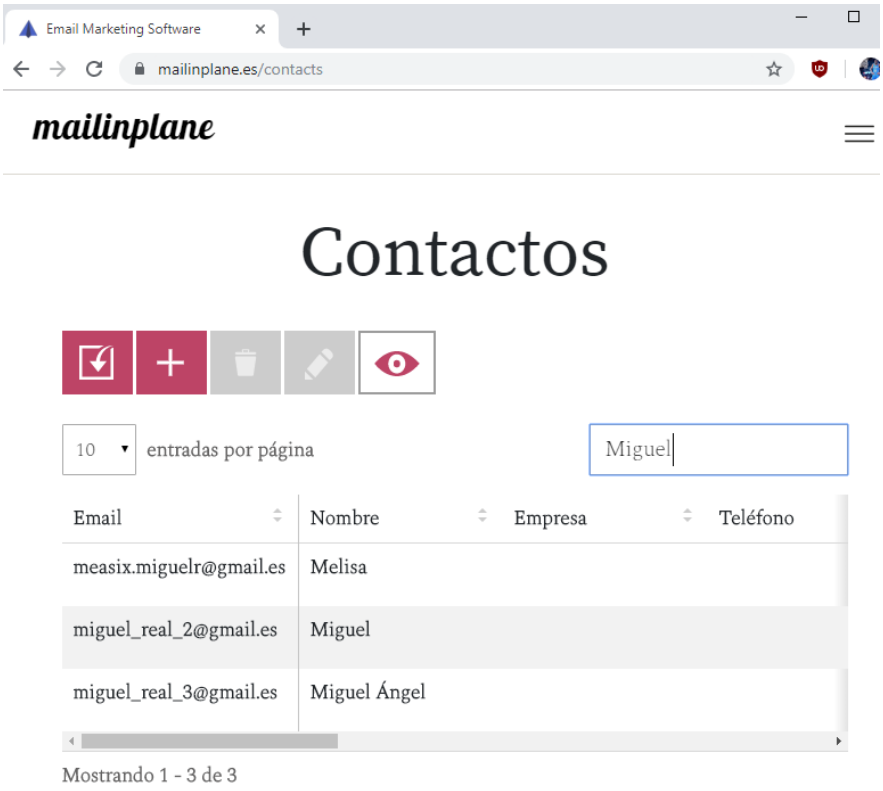


Ilustración 53: Vista de buscar contacto

Es posible acceder a los requisitos RF06 y RF07 desde la botonera de la tabla, no obstante, por motivos de relevancia se ha dejado fuera de la memoria la comprobación de estás dos operaciones básicas.

8.1.3. Campañas

Para hacer las pruebas pertinentes a las campañas crearemos una y la enviaremos a una serie de contactos. Un usuario recién registrado en la aplicación encontraría una vista similar a la de la Ilustración 54.

Ilustración 54: Vista de mis campañas

Clicando el botón de “Crear campaña” accedemos a una pantalla donde tendremos que elegir el tipo de campaña a crear (texto o plantilla) y el nombre (Ilustración 55). Posteriormente, iniciaremos un proceso guiado para la correcta creación de la campaña (correspondiente a RF12).

Ilustración 55: Vista de creación de campaña

El primer paso es seleccionar la audiencia a la que vamos a enviar la campaña. Para la prueba vamos a seleccionar la etiqueta “target” (Ilustración 56).

Ilustración 56: Vista de selección de audiencia

Lo siguiente es cumplimentar la información básica para el envío de la campaña (Ilustración 57).

Ilustración 57: Vista de configuración de campaña

Por último, crear el contenido (Ilustración 58). En el caso de prueba se crea el email mediante plantilla (arrastrando distintos elementos que facilita la aplicación).

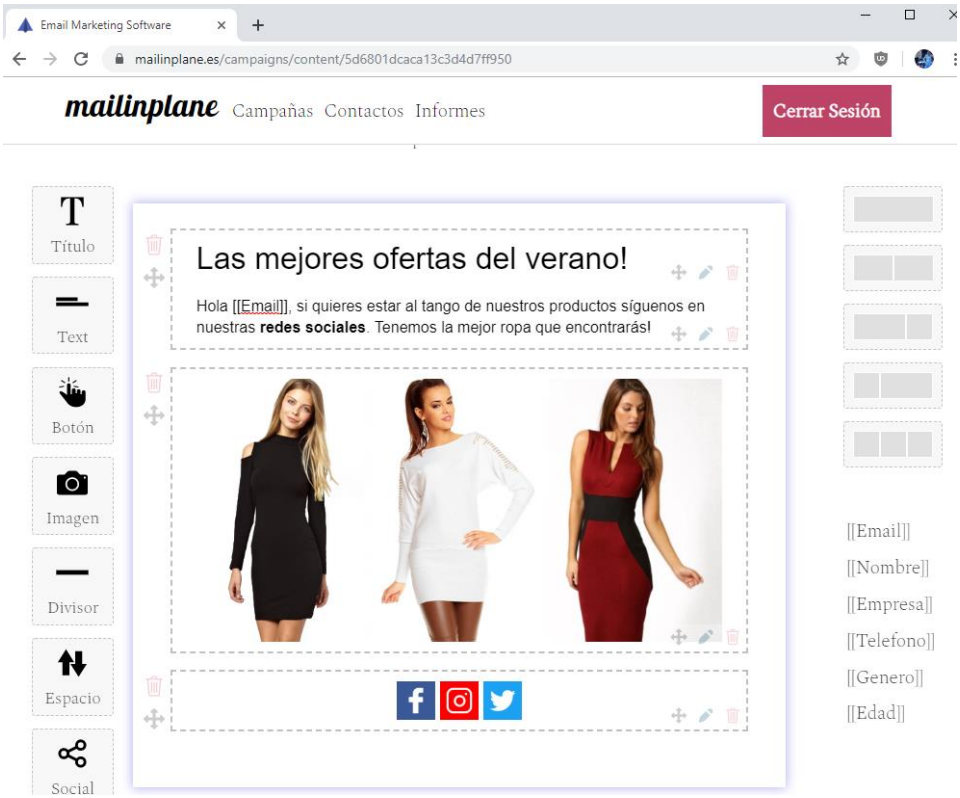


Ilustración 58: Vista de contenido de campaña

Una vez creada y configurada la campaña se nos permite enviarla tal y como se muestra en la Ilustración 59 (RF15).

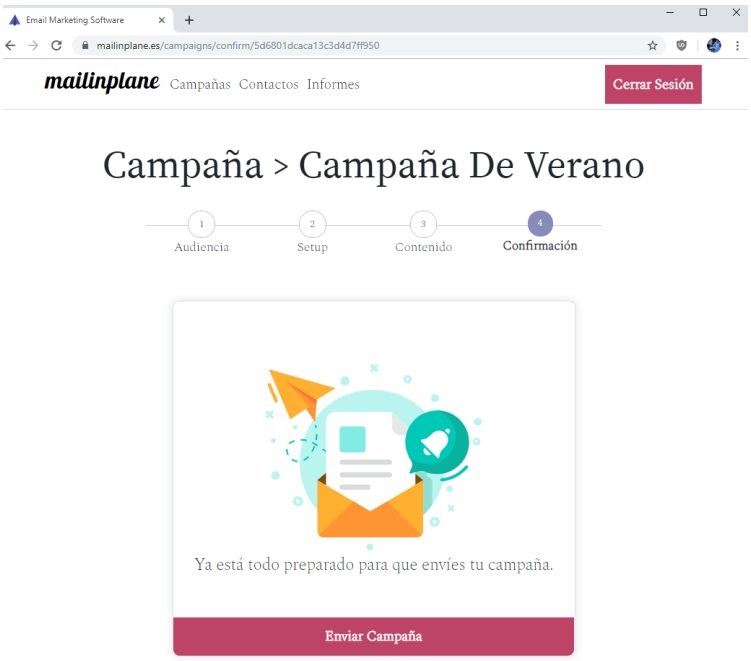


Ilustración 59: Vista de envío de campaña

La ilustración 60 corresponde con la apertura de la campaña recientemente enviada.

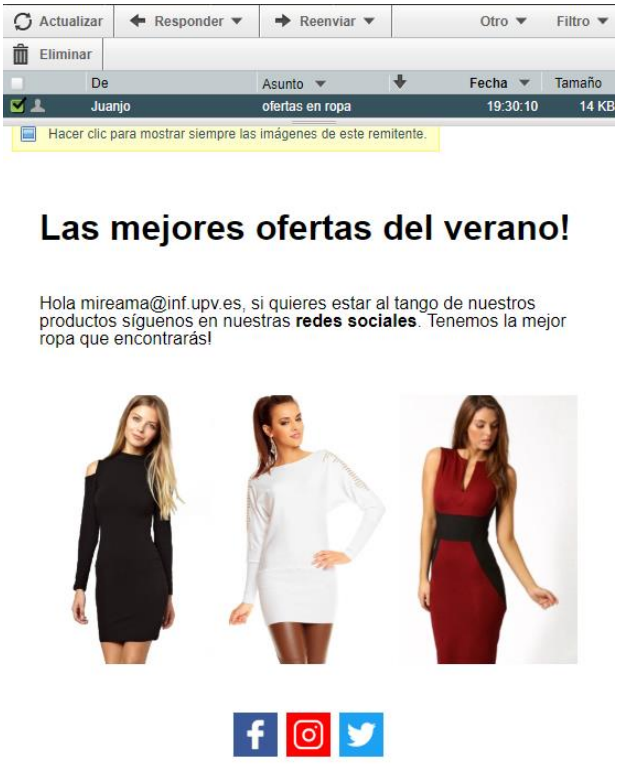


Ilustración 60: Correo enviado desde mailinplane

En la Ilustración 61 observamos las campañas creadas por el usuario (RF11), y en cada una de ellas en la parte superior derecha hay un icono de una estrella, el cual indica si tenemos o no marcada la campaña como favorita. Además, podemos editar cada campaña (RF14) pulsando sobre el texto “Ver campaña” (RF13).

Ilustración 61: Vista de mis campañas II

8.1.4. Informes

Si el envío de la campaña ha ido bien, podremos observar los informes relacionados a la misma a través de la sección de “Informes” de la cabecera (RF16). Tal y como se observa en la Ilustración 62, la campaña enviada aparece en la lista.

Ilustración 62: Vista de envíos

Si realizamos el seguimiento de la misma (RF18) nuestra campaña aparecerá además en la lista de campañas en seguimiento (Ilustración 63).

Ilustración 63: Vista de envíos en seguimiento

Si clicamos en nuestro envío observaremos una serie de informes básicos sobre los datos del mismo (Ilustración 64) dando por cumplido el requisito RF17.

Ilustración 64: Vista de informes

8.2. Requisitos no funcionales

8.2.1. Rendimiento

En este punto se han medido los diferentes tiempos de carga en las distintas pantallas de la aplicación. Para la medición del tiempo de carga se ha utilizado el monitor de red de Firefox.

Los datos correspondientes a la Tabla 20 corresponden a la pantalla principal, en la que se observa que la carga es más rápida en comparación con los datos obtenidos en la Tabla 21, los cuales corresponden al interior de la página. Esto se debe a que en el interior de la página se producen peticiones de recursos por parte del navegador a nuestro servidor, cosa que no ocurre en la página de inicio.

Tipo de conexión	Tiempo de carga (s)	Número de pruebas
Sin limitaciones (50Mb)	1,13	10
3G Buena	3,08	10
3G Regular	5,32	10

Tabla 20: Tiempos de carga en la página principal

Tipo de conexión	Tiempo de carga (s)	Número de pruebas
Sin limitaciones (50Mb)	2,42	10
3G Buena	4,6	10
3G Regular	5,70	10

Tabla 21: Tiempos de carga en el interior de la web

Podemos concluir que la carga inicial de la página no supera en ningún caso los 6 segundos que se habían marcado en el análisis de requisitos como límite.

Una vez hecha la carga inicial, el tiempo de respuesta de la aplicación se reduce considerablemente debido a que se trata de un SPA. Tanto es así que no existe funcionalidad alguna de nuestra aplicación que supere los 3 segundos de tiempo de respuesta, siendo el tiempo medio de 1 segundo.

8.2.2. Seguridad

En los puntos 6 y 7 de este documento se han tratado los puntos más relevantes en cuanto a seguridad, redactados previamente en el análisis de requisitos.

8.2.3. Fiabilidad

El sistema ha sido diseñado para ser tolerante a fallos y tal y como se muestra en las Ilustraciones 65-67 también ha sido diseñado para dejarle claro al usuario el estado de la aplicación en todo momento.

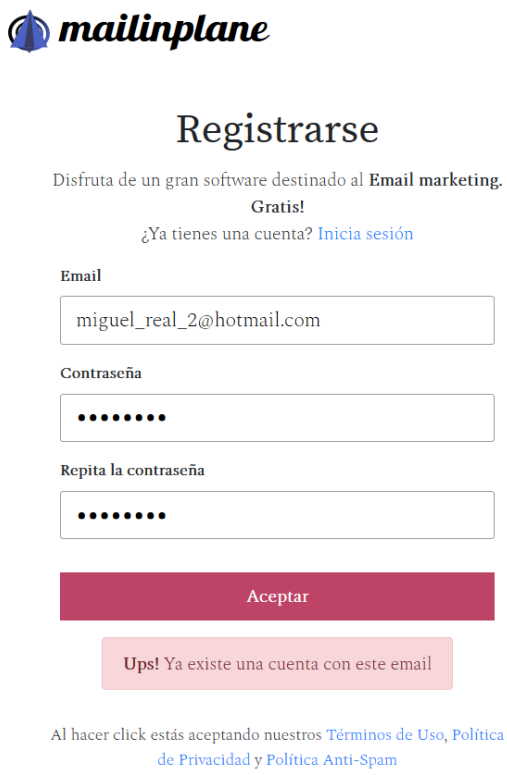


Ilustración 65: Email existente en la BDD

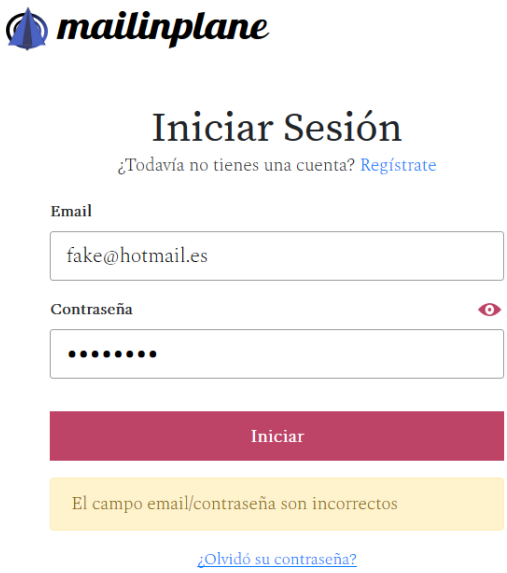


Ilustración 66: Email/contraseña incorrectos

Ilustración 67: Avisto de errores al enviar campaña

9. Conclusiones

Las aplicaciones orientadas al email marketing están asentadas en el mercado debido a la gran tasa de retención y captación de clientes que tienen.

En el desarrollo de una aplicación de estas características se me han presentado varios retos como son: el aprendizaje de un nuevo lenguaje, la utilización de una API externa, el uso de MongoDB y el despliegue de la aplicación en Amazon AWS.

No obstante, puedo decir que los resultados obtenidos satisfacen completamente los objetivos inicialmente propuestos. Se ha creado una aplicación web totalmente funcional capaz de gestionar contactos y enviar campañas de email, además de generar informes sobre dichas campañas para su posterior estudio.

Como punto a mencionar diré que ha sido un proyecto bastante ambicioso en el que se ha invertido más tiempo del recomendado, ya que a las horas invertidas en la creación de la web hay que sumarle las horas dedicadas al aprendizaje de *MEAN Stack*, a la investigación y a la elaboración de la memoria. Sin embargo, considero que el hecho de afrontar el desarrollo de una aplicación de tales características de principio a fin ha aumentado mi valor como desarrollador de software.

9.1. Relación del trabajo con los estudios cursados

Con relación a los estudios cursados, me ha sido de gran ayuda el hecho de haber realizado la rama de ingeniería del software, tanto para la parte de análisis de requisitos como para la planificación ágil utilizada (SCRUM en este caso). En concreto, las siguientes asignaturas me han sido de especial ayuda:

- Análisis y especificación de requisitos
- Diseño de software
- Gestión de proyectos
- Ingeniería del software
- Proyecto de ingeniería de software

10. Bibliografía

[1] Enríquez I. (3 mayo, 2017). "La historia del email marketing contada en 500 palabras". marketing4ecommerce. Recuperado el 10 de mayo de 2019, de: https://marketing4ecommerce.net/historia-del-email-marketing/

[2] Manzo J. (2 de agosto, 2017). "API vs Webhooks". Recuperado el 7 de junio de 2019, de: https://medium.com/@jdevmanzo/api-vs-webhooks-4745bffcfa65

[3] Méndez G. (2009). "Especificación de Requisitos según el estándar de IEEE 830". Recuperado el 6 de junio de 2019, de: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

[4] Prikaznov A. (20 febrero, 2016). "Angular JS Tutorial – MVC and MVVM Design Patterns". Recuperado el 15 de junio de 2019, de: https://dzone.com/articles/angularjs-tutorial-lesson-1

[5] Rietta F. (5 de febrero, 2016). "Use Bcrypt or Scrypt Instead of SHA* for Your Passwords, Please!". Recuperado el 15 de mayo, de: https://rietta.com/blog/bcrypt-not-sha-for-passwords/

[6] Robert C. Martin (2008). "Clean code: A Handbook of Agile Software Craftsmanship". Upper Saddle River, NJ: Prentice Hall.

[7] Robert C. Martin, Micah Martin (2006). "Agile Principles, Patterns, and Practices in C#". Upper Saddle River, NJ: Prentice Hall.

[8] Stecky-Efantis M. (16 de mayo, 2016). "5 Easy Steps to Understanding JSON Web Tokens (JWT)". Recuperado el 15 de mayo, de: https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec