



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo fin de Máster

Máster en Ingeniería Informática

Metodología Running Lean aplicada a un lector de noticias inteligente

**Realizado por
(ponente): Andrés M. Jiménez Ríos**

**Dirigido por
Alejandro Fernández-Montes González
Juan Manuel Cordero Valle**

**Departamento
Lenguaje de Sistemas Informáticos**

Sevilla, 29 de mayo de 2019

Resumen

La metodología LEAN está abriéndose paso a través de múltiples campos en la industria. Esta ofrece una serie de características que hacen de ella una herramienta indispensable en determinadas áreas. Entre estas destacan la rapidez de desarrollo, el enfoque orientado a pruebas y a presentar a los interesados los avances en determinadas fases.

En concreto, dentro del mundo del software, fue Ash Maurya el que aplicó dicha metodología de manera sistemática y casi programática. Con *Running LEAN* Maurya explora proyectos personales llevados a cabo siguiendo LEAN de una manera concreta.

Con este prisma, se quería desarrollar un lector de noticias. Sin embargo, previa a su construcción, era necesaria una fase de estudio de la idea. Que esta fuese probada *en la calle*, y con las características que los futuros clientes utilizarían. Así, LT-News saldría como el lector que la gente desea.

¿Por qué LT-News? Surge de la necesidad de todos de informarse, pero la enormidad de medios y el poquísimo tiempo nos hace excusarnos de la gran tarea de leer y formarse una opinión propia de lo que ocurre a nuestro alrededor. Esta aplicación se propone ser un lector RSS, pero no uno típico, no, sino uno que acabe conociendo tanto al usuario, que averigüe tus gustos, le muestre las noticias en las que esté realmente interesado.

Este añadido, aportará gran valor a la aplicación, aunque será únicamente lo que haga de LT-News un lector de noticias *inteligente*. Poseerá funciones de búsqueda de noticias y análisis de las mismas. En concreto, aportará cinco grandes características. En primer lugar, analizará indirectamente al usuario debido al uso que hará de la aplicación para extraer sus gustos. En segundo lugar, analizará las noticias extrayendo los temas que trata. En tercer lugar, relacionará noticias entre sí debido a la similitud de temas y recomendará noticias al usuario en base a sus gustos y temas preferidos. Finalmente, permitirá la búsqueda facetada y hará un análisis de prensa, haciendo un resumen de la misma en el periodo de tiempo que se deseé.

Agradecimientos

Sero te amavi, pulchritudo tam antiqua et tam nova, sero te amavi! et ecce intus eras et ego foris, et ibi te quaerebam, et in ista formosa, quae fecisti, deformis inruebam. mecum eras, et tecum non eram. ea me tenebant longe a te, quae si in te non essent, non essent. vocasti et clamasti et rupisti surditatem meam: coruscasti, splenduisti et fugasti caecitatem meam: fragrasti, et duxi spiritum, et anhelo tibi, gustavi et esurio et sitio, tetigisti me, et exarsi in pacem tuam.

Índice general

Índice general	III
Índice de cuadros	V
Índice de figuras	VII
1 Introducción	1
1.1 Justificación del proyecto	1
1.2 Objetivos	2
1.2.1 Objetivos funcionales	2
1.2.2 Objetivos técnicos	2
1.3 Alcance	3
1.4 Estado del arte	3
1.4.1 Feedly	3
1.4.2 Flipboard	5
1.4.3 Google News	6
1.4.4 Otras	8
2 Estudio	9
2.1 Metodología LEAN	9
2.1.1 <i>Running Lean</i>	10
2.1.2 Aplicación real	14
2.2 Primer estudio	15
2.2.1 Segmento de clientes	16
2.2.2 Problema	16
2.2.3 Propuesta de valor único	16
2.2.4 Principales hipótesis	16
2.3 Iteraciones	16
2.3.1 Primera iteración	17
2.3.2 Segunda iteración	17
2.3.3 Tercera iteración	19
2.3.4 Cuarta iteración	20
2.3.5 Quinta iteración	25
2.3.6 Quinta iteración	29
2.4 Conclusión estudio	30
3 Planificación	31
3.1 Metodología	31
3.1.1 SCRUM	32
3.1.2 Kanban	35

3.1.3	Aplicación	36
3.2	Planificación temporal	38
3.3	Roles del proyecto	39
3.4	Planificación económica	40
3.4.1	Costes directos	40
3.4.2	Costes indirectos	41
3.4.3	Presupuesto	42
4	Análisis	43
4.1	Modelo conceptual	43
4.2	Actores del sistema	44
4.3	Catálogo de requisitos	44
4.3.1	Requisitos funcionales	45
4.3.2	Requisitos no funcionales	50
4.4	Diagrama de secuencia	51
4.4.1	Perfil	51
4.4.2	Section	53
4.4.3	Feed	57
4.4.4	Item	60
5	Diseño	70
5.1	Patrones	70
5.1.1	Modelo-Vista-Controlador	70
5.1.2	Controlador-Servicio	70
5.1.3	Desacoplamiento Cliente-Servidor	71
5.2	Modelo de datos	72
6	Implementación	75
6.1	Tecnologías	75
6.2	Herramientas	79
6.2.1	Backend	79
6.2.2	Frontend	80
6.2.3	Servidor	82
6.2.4	Gestión	83
6.3	Estructura del proyecto	84
6.3.1	GitHub	85
6.3.2	Backend	87
6.3.3	Frontend	89
7	Pruebas	91
7.1	Pruebas unitarias	91
7.2	Pruebas integradas	91
7.3	Pruebas de aceptación	92
7.4	Estándares de código	93
8	Conclusión	95
8.1	Resultados	95
8.2	Mejoras futuras	95
8.3	Lecciones aprendidas	97
Referencias		98

Índice de cuadros

2.1	Aprendizaje entrevista de problema	13
2.2	Aprendizaje entrevista de solución	14
2.3	Aprendizaje entrevista del MVP	14
2.4	Iteraciones realizadas	16
3.1	Planificación de Sprints: duración	38
3.2	Planificación de Sprints: horas	38
3.3	Planificación temporal	39
3.4	Presupuesto económico	42
4.1	HE1 - Acceso aplicación	45
4.2	HE2 – Gestión de perfil	45
4.3	HE3 – Gestión de secciones	45
4.4	HE4 – Gestión de noticias	45
4.5	HE5 – Gestión de intereses	45
4.6	HU01 – Registro	45
4.7	HU02 – Acceso	46
4.8	HU03 – Ver perfil	46
4.9	HU04 – Editar perfil	46
4.10	HU05 – Ver secciones	46
4.11	HU06 – Editar secciones	46
4.12	HU07 – Crear secciones	46
4.13	HU08 – Eliminar secciones	47
4.14	HU09 – Ver feeds por sección	47
4.15	HU10 – Añadir feed por sección	47
4.16	HU11 – Eliminar feed por sección	47
4.17	HU12 – Ver noticias por sección	47
4.18	HU13 – Ver noticias por feed	47
4.19	HU14 – Ver noticias de actualidad	48
4.20	HU15 – Ver una noticia	48
4.21	HU16 – Ver noticias relacionadas	48
4.22	HU17 – Ver comentarios de noticias	48
4.23	HU18 – Comentar noticias	48
4.24	HU19 – Buscar noticias	48
4.25	HU20 – Marcar noticia como <i>Me gusta</i>	49
4.26	HU21 – Ver palabras claves por noticia	49
4.27	HU22 – Ver noticias por palabras claves	49
4.28	HU23 – Resumen de actualidad	49
4.29	HU24 – Guardar una noticia	49
4.30	HU25 – Ver intereses	49

4.31 HU26 – Ver noticias según mis intereses	50
4.32 RNF1 – Control de acceso	50
4.33 RNF2 – Recurrencia	50
4.34 RNF3 – Sencillez de uso	50
4.35 RNF4 – Guía de ayuda	50
4.36 RNF5 – Barra de navegación	50
4.37 RNF6 – Sesión iniciada	50
4.38 RNF7 – Interfaz responsiva	51
4.39 RNF8 – Traspaso de datos	51

Índice de figuras

1.1	Logo de Feedly	4
1.2	Página inicio de Feedly	4
1.3	Logo de Feedly	5
1.4	Página inicio de Flipboard	6
1.5	Logo de Google Noticias	7
1.6	Página inicio de Google News	7
2.1	Resumen Running Lean	10
2.2	Lean Canvas	11
2.3	Adecuación de nuestra idea al mercado	12
2.4	Prueba sistemática	13
2.5	LEAN Canvas inicial	15
2.6	LEAN Canvas primera iteración	17
2.7	LEAN Canvas segunda iteración	19
2.8	LEAN Canvas tercera iteración	20
2.9	Encuesta primera pregunta	21
2.10	Encuesta segunda pregunta	21
2.11	Encuesta tercera pregunta	22
2.12	Encuesta cuarta pregunta	22
2.13	Encuesta quinta pregunta	23
2.14	Encuesta sexta pregunta	23
2.15	Encuesta séptima pregunta	24
2.16	Encuesta octavo pregunta	24
2.17	LEAN Canvas cuarta iteración	25
2.18	Landing Page 1	26
2.19	Landing Page 2	26
2.20	Landing Page 3	27
2.21	Google Analytics 1	27
2.22	Google Analytics 2	28
2.23	Google Analytics 3	28
2.24	LEAN Canvas final	30
3.1	Aprendizaje en SCRUM	32
3.2	Eventos en SCRUM	33
3.3	Equipo en SCRUM	34
3.4	Ejemplo de tablero Kanban	35
3.5	Tablero Kanban usado	37
4.1	Diagrama de clases	43
4.2	Diagrama de secuencia de profile_view	51
4.3	Diagrama de secuencia de profile_edit	52

4.4	Diagrama de secuencia de feed_list	53
4.5	Diagrama de secuencia de section_view	54
4.6	Diagrama de secuencia de section_delete	55
4.7	Diagrama de secuencia de section_edit	56
4.8	Diagrama de secuencia de feed_view	57
4.9	Diagrama de secuencia de feed_create	58
4.10	Diagrama de secuencia de feed_delete	59
4.11	Diagrama de secuencia de item_list	60
4.12	Diagrama de secuencia de item_view	61
4.13	Diagrama de secuencia de item_view_like	62
4.14	Diagrama de secuencia de item_view_save	63
4.15	Diagrama de secuencia de item_view_web	64
4.16	Diagrama de secuencia de item_recommend	65
4.17	Diagrama de secuencia de item_query	66
4.18	Diagrama de secuencia de item_search	67
4.19	Diagrama de secuencia de item_summary	68
4.20	Diagrama de secuencia de item_saved	69
5.1	Modelo de datos	72
6.1	Muestra del <i>Web Scrapping</i>	75
6.2	Muestra de Recuperación de Información	76
6.3	Muestra de sistemas de recomendación	77
6.4	Muestra de extracción del perfil del usuario	78
6.5	<i>Git Flow</i>	85
6.6	Respositorios en GitHub	86
6.7	<i>Issues</i> en Github	86
6.8	<i>Labels</i> en Github	87
6.9	<i>Milestones</i> en Github	87
6.10	Estructura en PyCharm	88
6.11	Estructura en WebStorm	89
7.1	Muestra de logs de Travis	92
7.2	Respositorios en Codacy	93
7.3	Seguridad en Codacy	94

CAPÍTULO 1

Introducción

1.1– Justificación del proyecto

Cuando decidí hacer mi Trabajo Fin de Grado, tome uno de los temas que más me apasiona: el periodismo. La idea era hacer una aplicación que superase a todas las existentes -se entiende que solo desde el punto de vista conceptual. Finalmente lo conseguí, quedando un lector RSS inteligente.

La aplicación cumplía todas las características que un usuario pide a un lector de noticias. Estas son: añadir cualquier fuente que posea sindicación de contenido, gestionar los *feeds* y leer las noticias de estos al momento, llevando un control de lectura de las mismas. Añadía a lo anterior complementos interesantes como guardar noticias para leerlas más tarde o agrupar los *feeds* en secciones.

Además de lo anterior, poseía características adicionales que le daban el sobrenombre de: inteligente. Una de estas era la capacidad de relacionar noticias entre sí y mostrárselas al usuario mientras leía un noticia. Otra, la extracción del perfil del usuario y la recomendación de noticias en base al uso de la aplicación y al tipo de artículos leídos. La tercera característica era la búsqueda de noticias en base a los criterios que se desease.

Como se puede ver, no se encuentra actualmente un lector de noticias con estas características. ¿Por qué? El motivo es el coste de la infraestructura que ha de poseer la aplicación. Si pensamos en cualquier medio importante, veremos que su fichero RSS se actualiza de media cada diez minutos, publicando así una media de una quince noticias por hora. Si seguimos aritméticamente los cálculos, veremos que por día, sólo en España, se publican del orden de millones de artículos.

Por ello, las tecnologías, infraestructura y algoritmos que utilicé quedaron obsoletos al poco tiempo, siendo necesario profesionalizar el proyecto si se quería una aplicación que el público final pudiese utilizar. Además, faltaba en esta una parte importante: la monetización.

Sin embargo, después de lo estudiado a lo largo del Máster, muchos planteamientos seguidos en el Trabajo Fin de Grado están obsoletos. Gran parte de estos se aglutan a la hora de tomar requisitos de la aplicación. Como se puede ver, este era un proyecto de emprendimiento, de una necesidad real que percibía. Sin embargo, no había llegado a probarlo con los futuros usuarios, los que estarían dispuestos a pagar por el servicio ofertado. Para esto, se aplicará la metodología *Running Lean*, con la que se testará la idea con el público objetivo.

Como se puede dilucidar, el Trabajo Fin de Máster consistirá en estudiar y probar la idea de negocio bajo el marco LEAN. Una vez comprobada, se procederá a rehacer la aplicación utilizando tecnologías adaptadas a esta enorme ingesta de datos y utilizando un formato más profesional, del que más tarde hablaremos. Se desarrollarán sólo aquellas características que provengan del fruto del estudio, ya que se realizará un Producto Mínimo Viable. Además, la idea poseerá un cariz comercial, ya que se desea ganar dinero con el sistema.

1.2– Objetivos

Los objetivos se engloban en dos categorías: funcionales y técnicos. En la primera se agrupan todos los conocimientos que quiero adquirir en cuanto a metodología, conocimiento del dominio del problema y venta de producto. Segundo será el aprender un determinado número de tecnologías o infraestructuras.

1.2.1. Objetivos funcionales

El objetivo principal a adquirir en cuanto a conocimiento es la experiencia emprendedora. Esto no es más que convertir de una idea tecnológica atrayente y estimulante, a una pequeña empresa que sostenga y monetice dicha idea. Aunque pueda parecer algo nimio a simple vista, requiere todo un proceso de concepción de la idea, el cliente y el ecosistema de ventas. Gracias a las asignaturas del máster, veo posible implementar dicha idea hacia una idea, ya no puramente tecnológica, sino de negocio.

Los objetivos subyacentes a esta son el conocimiento de las metodologías que me permitan realizar este proceso. En concreto, me he basado en tres ideas actuales de emprendimiento que se pueden encontrar a lo largo del documento y que se reducen a tres autores:

- *Customer Development* de Steve Blank
- *Business Model* de Alex Osterwalder
- *Running Lean* de Ash Maurya

En estos se definen los procesos de creación de la *start-up*, diseño del producto, atracción de clientes y de mejora continua. Me han parecido enormemente interesante y acertados para la aplicación en mi idea de negocio, por eso decidí aplicarlo.

El segundo objetivo es el conocimiento más férreo del dominio del problema de la idea de negocio, es decir, el periodismo. Este campo está viviendo un proceso de cambio en cuanto a necesidad de información se refiere. Los consumidores de noticias ya no están interesados en conocer lo que ocurrió el día anterior, sino lo que pasa en cualquier momento en cualquier parte del mundo. Es por ello que la prensa ha tenido que pivotar para saciar el ansia de informarse de sus usuarios. Además de esto, han sabido atraer a usuarios más jóvenes, que prefieren conocer el mundo a través de las redes sociales, antes que la crónica de los periodistas.

En este proceso, además, se esconde una oscura pero cada vez más visible realidad: el mundo de la comunicación avanza varios pasos por detrás del ritmo *impuesto* por la sociedad digital. Este es el motivo por el que se pueden ver múltiples deficiencias en la prensa actual, que se explicarán a lo largo del documento. Es por ello también objetivo del presente estudio proponer mejoras al mismo.

1.2.2. Objetivos técnicos

Una vez se haya probado la idea, quedará desarrollarla. Para ello, se desarrollará la aplicación siguiendo las pautas marcadas por las grandes compañías, en cuanto a tecnologías, herramientas e infraestructuras se refiere. Teniendo en cuenta este preámbulo se entenderán correctamente los objetivos subyacentes.

El primer objetivo es el desarrollo de una aplicación utilizando la estrategia de desacople de capa de servidor y cliente. Esto no es más que ofrecer desde el *backend* una API consumible para cualquier cliente. Gracias a esto, se pueden diferenciar las tecnologías de cliente y servidor, haciéndolas independientes.

El segundo objetivo es desarrollar la capa de presentación utilizando un framework JavaScript. Esta estrategia es enormemente usada en el mundo del desarrollo web, ya que es posible realizar toda una aplicación *frontend* con AJAX, es decir, sin recargas globales de la página para extraer

la información. Toda la comunicación con el servidor será asíncrona, por lo que la fluidez que percibirá el usuario será total.

El tercero será el desarrollo de una capa de inteligencia artificial que permita conocer tanto al usuario que interacciona con la aplicación así como el contenido que se muestra. Con ello, se podrá recomendar y extraer datos de ambos: noticia y usuario. Esta capa, además de dar un valor añadido al software que se realice, lo hará enormemente interesante para un usuario con conocimiento del dominio del problema, ya que será posible extraer conclusiones de contexto.

Por último, y en la medida de lo posible, se incoará un proceso de integración y entrega continua. Para ello, se utilizarán herramientas que permitan agilizar, en la medida de lo posible, el tiempo desde que un desarrollador hace un cambio hasta que el usuario lo percibe en la web.

1.3– Alcance

Es objeto del presente trabajo es el estudio, análisis e implementación de un proceso completo de negocio que va desde la concepción de la idea hasta la venta de la misma. Para ello, se utilizará el método indicado por Ash Maurya con su idea del *Running Lean*, enormemente apoyada por emprendedores de todo el globo.

Dentro de la implementación, realizado siempre bajo el prima de Producto Mínimo Viable, como indica Ash, se pretende realizar una lector RSS con características adicionales que le den un valor añadido como son el análisis de perfiles de usuarios, el análisis de medios y el análisis de noticias. El hacer hincapié en el *análisis*, no es más que recordar lo que diferenciará el producto del resto. El usuario, en definitiva, lo notará debido a la capacidad de realizar búsquedas en la aplicación, así como la recomendación de noticias.

Dentro del marco actual, la aplicación estará pensada para los dispositivos actuales en toda la gama de resoluciones de pantallas. Además, al permitirse la posibilidad de añadir cualquier tipo de *feed*, se añadirá a la plataforma la posibilidad de añadir medios en múltiples idiomas.

1.4– Estado del arte

La sindicación de contenido vino a resolver un problema en 1995. Este no era otro que dar la posibilidad a un usuario de poder seguir desde una única plataforma todas las noticias de diferentes medios.

En cambio, si vemos como ha evolucionado el mundo digital, en cuanto al número de webs se refiere, podemos ver lo siguiente:

En solo 10 años, hemos pasado de tener 92 millones de sitios web en Internet a más de 1.000 millones. Internet ya no es solo un lugar dónde encontrar información o dónde encontrar ocio y entretenimiento, ni es solo una herramienta de trabajo, Internet es dinero y es cambio social.

Rana Negra (2018)

Ahora, entonces, podemos ver otro problema, no resuelto todavía. Ante la gran cantidad de medios actuales, el problema es escoger entre tanta cantidad de noticias, solo aquello que aporte valor y realmente nos interesa. Ahora que termina el segundo decenio del siglo, podemos ver cómo intenta solucionar esto cada aplicación.

1.4.1. Feedly

Aplicación nacida en 2008 con el objetivo de llegar a los usuarios que empezaban a entrar en el mundo de los smartphones. Vio su auge al desaparecer Google Reader. Uno de sus puntos fuertes que le hizo crecer, a la par del suceso anteriormente mencionado, es el tratamiento multi-idioma que se hace en la plataforma.



Figura 1.1: Logo de Feedly

Una de sus mejores ventajas es la sencillez de uso a la hora de añadir nuevas fuentes al *feed*: es momentáneo. Otra es la vista que tiene dentro del apartado *Today*. Aquí encontramos las noticias nuevas de los medios a los que estamos suscritos. Puede parecer una característica baladí o demasiado popular. Sin embargo, Feedly consigue hacerlo único siguiendo una operativa que le da un valor añadido con respecto al resto.

Primero, que este no muestra las noticias que *te has perdido* en orden cronológico puramente; entremezcla las secciones que tengas en pantallas y ahí sí muestra las noticias en orden de publicación. Esto es enormemente útil ya que, por ejemplo, si tienes *El País* o el *ABC* en la sección *Noticias*, lo normal es que tengas noticias cada diez minutos. Sin embargo, si posees otra sección, *Fútbol*, donde se recogen *feeds* con análisis de los partidos más importantes, lo normal es que tengas cinco noticias a la semana. Siguiendo dicho ejemplo, Feedly mostrará igual la noticia de hace cinco minutos de *El País* que a la de hace una semana de *Ecos del Balón*.

Abstrayendo del ejemplo, podemos ver que habrá secciones con noticias cada minuto y otras, cada semana. Con esta distribución, Feedly consigue que se le da igual importancia a ambas secciones mientras que te queden por leer noticias.

Figura 1.2: Página inicio de Feedly

Segundo, que no hay un *scroll infinito* a la hora de leer noticias. Esto hace aburrido y desmotivante al lector la lectura. Lo que hace son barridas, es decir, que va seleccionando un número determinado de noticias por cada sección. Una vez visto estas, pregunta al usuario si quiere seguir viendo más.

Este detalle junto a las múltiples integraciones que posee le hace enormemente usable y querido por el usuario, por lo que hace de esta aplicación, la principal a combatir en cuanto a presencia en el sector se refiere.

Puesto a poner pegas siempre se le achacan dos desventajas. La primera es que el usuario de esta aplicación suele ser una persona que ha de conocer, aunque sea levemente, el mundo de la sindicación de contenido. Esto hace que para un usuario con poco conocimiento de internet pueda parecer un mundo.

La otra característica a mejorar es la recomendación tanto de medios como de noticias. Actualmente no se ofrece ni por asomo esta funcionalidad y cada vez es más solicitada en blogs especializados, ejemplo del mismo el artículo de Paul Bonduelle (2018). Como se puede ver, empieza a tomar cuerpo, pero falta mucho camino a recorrer.

1.4.2. Flipboard

Aplicación nacida de la mano del iPad, la primera tableta del mercado, en 2010. Debido a su auge, se decidió desde la compañía crecer a los dispositivos móviles de Apple. Por ello, nació la aplicación para iPhone e iPod. Igual de triunfante fue tal decisión que, tras un *crowdfunding* en 2012, se decidió ir tras el mundo Android.

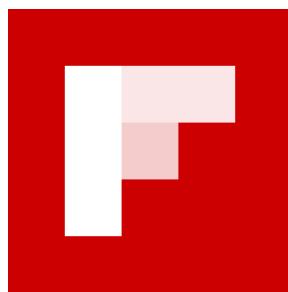


Figura 1.3: Logo de Feedly

Su enorme ventaja es la sencillez de uso total. Cualquier tipo de usuario puede configurarse su *feed* sin necesidad de ningún conocimiento técnico ni informativo adicional. Desde el registro en el sistema, se dan una serie de secciones base que elegir. Una vez añadidas e informado de los gustos, Flipboard se encargará de añadir o quitar medios en dichos feeds en cuanto a tus gustos se refiere.

Hace totalmente transparente al usuario a la tecnología que se encuentra de fondo. Este es el motivo por el que no es posible crear secciones ni seleccionar medios más concretos utilizando su enlace de sindicación, es decir, por la URL del RSS o Atom.

Al centrarse en el usuario, hace que la interfaz sea enormemente atractiva y el punto continuo de mejora. Al ser este su foco de atención, obvia puntos conflictivos y más técnicos. Uno de ellos es la vista previa de noticias. Es sabido que dentro del marco de sindicación de contenidos, son los medios los que deciden qué información añaden al *feed*. Algunos les interesa captar usuarios a la web, por lo que la descripción de la noticia es casi nula; a otros, que el usuario se informe desde cualquier medio, por lo que añaden toda la noticia. Sabiendo esto, es el lector el que decide como lidiar con esta casuística, que conllevará diferentes problemas. Flipboard lo *soluciona* llevando directamente al usuario al medio, sin previsualización de la noticia.

Otro hecho llamativo es el flujo de estados de las noticias por los usuarios. Cada artículo,

pasa por una serie de etapas: no-leído, leído, comentado o gustado, entre otros. Esto complica a la aplicación, dando un valor añadido al lector. Al igual que lo anterior, Flipboard se lo ahorra: a la hora de mostrar las noticias, se muestran hayan sido vistas o no, además que en formato *scroll continuo*.

El evitar solucionar estos problemas, no pone en tela de juicio el valor de la app. Simplemente, sabe cuál es su *target* y qué le interesa. Incluye, como característica disruptiva, la integración con las redes sociales, en concreto, con Twitter. Cada noticia se redirecciona con los tuits oficiales o de cuentas con cierto conocimiento de causa de lo que se está hablando.

Otro punto fuerte de Flipboard es el concepto de *Revista*. Esto no es otra cosa que la posibilidad de agrupar noticias que nos interesen y poder enseñarla al mundo. Tan es así, y tanto fama y utilidad le dan los usuarios, que la empresa dio el dato en 2016 que poseía 28 millones de revistas Wikipedia (2018).

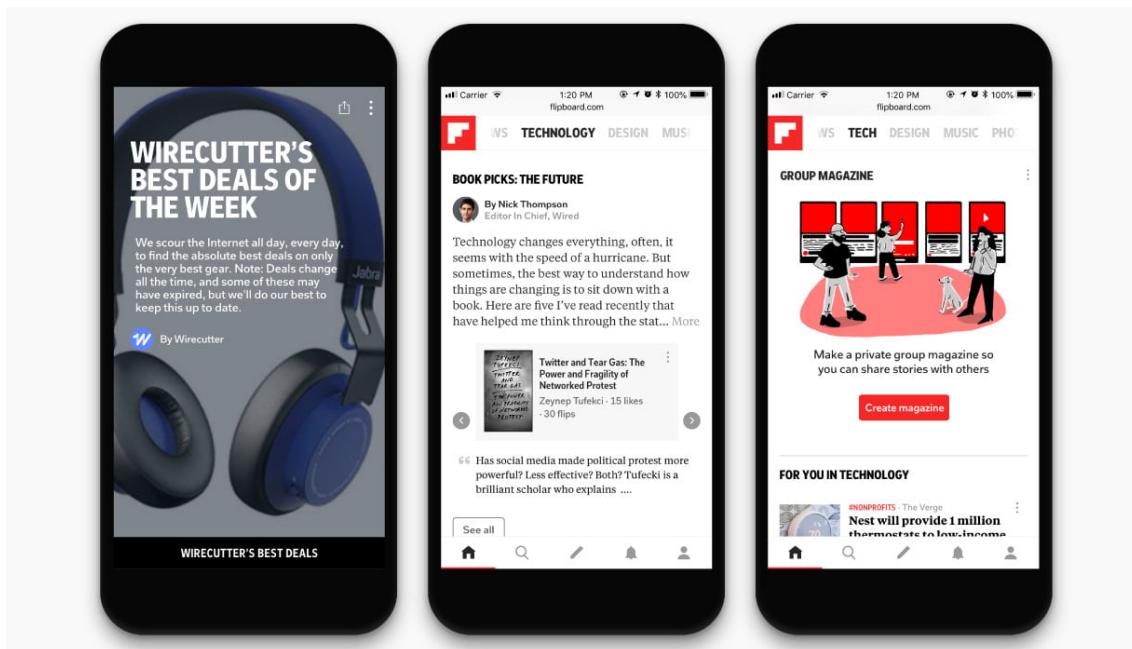


Figura 1.4: Página inicio de Flipboard

Como se ha dicho, desde el punto de vista técnico puede resultar un poco pobre, sobre todo a un usuario que posea conocimiento del tema. Además, hay poco uso de las recomendaciones de noticias o medios, y no se plantea en medio plazo el de incorporarlo. Sin embargo, su punto fuerte es y será la interfaz de usuario y la sencillez de uso.

1.4.3. Google News

Google Noticias fue lanzada en 2002. Desde entonces hasta hace unos años, posee las mismas características y, si se apura, la misma interfaz. Desde la conferencia de Google del pasado año Quentyn Kennemer (2018), se le dio a la aplicación un vuelco que la hizo orientarse por completo a la inteligencia artificial con recomendaciones de noticias entre sí aportando un contexto de la misma. Además, se implementó, tal y como hemos comentado con Feedly, una vista diaria.

Como se puede ver, se ha seguido un orden al mencionar los productos digitales. Este muestra la importancia que cada uno aporta al mundo de los lectores RSS. Es este es el más novedoso y el que posee un valor añadido superior.



Figura 1.5: Logo de Google Noticias

Además, como es sabido, Google tiende a usar un diseño común, hilo conductor en todos sus productos. Este era *Material Design*, y ahora *Material Theme*. Dicho esto, se podrá intuir el nivel del usabilidad del mismo, así como de su atracción visual al usuario.

Sin embargo, y este es el motivo de que aparezca en tercer lugar en el estado del arte, no se puede usar en España. Como se ha dicho, 2018 fue un cambio importante desde el punto de vista técnico. También ocurrió lo mismo en 2014, pero a menor escala. En este año, la gran G lanza Google Noticias y Tiempo. Aquí, solo usando la ubicación del usuario, se era capaz de darle unos cuantos titulares de noticias de rabiosa actualidad según su posición.

Los grandes medios españoles se vieron afectados por esta característica, ya que se daba igual importancia a un medio local que a los grandes periódicos. Esto, unido al hecho de la nueva legislación española sobre la propiedad intelectual, provoca el cierre de Google News Ana Pérez Barredo (2014).

A screenshot of the Google News homepage. At the top, there's a search bar and navigation links for Headlines, Local, For You, and U.S. Below that, a sidebar lists 'SECTIONS' like Top Stories, World, U.S., Business, Technology, Entertainment, Sports, Science, and Health, with 'Top Stories' currently selected. The main content area shows 'Top Stories' with an article about Sarah Huckabee Sanders. To the right of the main content, there's a sidebar with a 'Google N' section and a 'In the Ne...' section. At the bottom, there's another news item about Time Magazine.

Figura 1.6: Página inicio de Google News

Este hecho, sin embargo, no ha hecho que disminuya el éxito de la misma en otros países. Si a Feedly o Flipboard le interesa tener su plataforma en diferentes idiomas, qué menos Google. Tan es

así, que Google Noticias posee soporte para más de 40 idiomas. Esto es un paso más al tratamiento multi-idioma. Es adaptar la web y las lecturas de las noticias en diferentes abecedarios, tales como el árabe, hindi o chino.

No obstante, posee algunas deficiencias técnicas, que en realidad no son tal según su naturaleza. Esto se debe a que Google no le interesa hacer una web de noticias, sino un simple agregador de las mismas. La diferencia radica, por ejemplo, en que a Google no le interesa ser un lugar donde consultar todas las noticias a lo largo del tiempo, sino, más bien, donde poder ver los titulares del momento.

Dichas diferencias se pueden ver en la siguientes operativas que realiza Google News en comparación con cualquier otro lector. La primera es que las noticias podremos verlas con un retraso de un cuarto de hora, es decir, que se actualizarán cada 15 minutos, por tanto no se intenta estar en tiempo real de las noticias.

Otra diferencias radica en que no se intenta hacer un resumen de la noticia en sí, como preámbulo, sino que se redirecciona a la web original para leer la noticia entera. Con esto, se ahorra todos los problemas legislativos subyacentes. Por último, y resumen de todo, es la ya mencionada restricción del número de noticias a guardar. Solo se podrán consultar noticias en un periodo de un mes, sino, Google ya nos las guarda.

1.4.4. Otras

Se pueden encontrar otras soluciones al problema de la sindicación de contenido, pero son más simples que las ya comentadas. Algunas a destacar son Feedbin, Inoreader o Winds. Cada cual proponen algo diferente al usuario, aunque, poseen una gran deficiencia cada una que la imposibilitan de triunfar.

Primero en importancia encontramos Feedbin. Esta es una plataforma RSS que recopila noticias de todas las fuentes que seleccione el usuario, funcionando al estilo PaaS (*Platform as a Service*). Esto quiere decir que los clientes, tanto webs, como de escritorio o de aplicaciones, que encontraremos son independientes al mismo, por lo que la funcionalidad de cada uno diferirá en cada caso.

Este ofrece las características esperadas de cualquier lector RSS en 2019, con algún añadido, propios de la una tecnología puramente de servidor tales como Newsletter, Búsqueda o Podcasts. Lógicamente, con un sobrecoste adicional e inicial, ya que está casi por completo orientado al mundo iOS.

En segundo lugar encontramos Inoreader. Su enorme fama se debe a su increíble simplicidad para un usuario con previos conocimiento de sindicación de contenido. Posee muy pocas opciones y solo es accesible desde aplicaciones móviles y además es gratuito, por lo que le hace perfecto para usuarios que estén a caballo entre Feedly y Flipboard: ni la configuración de uno, ni la simpleza del otro.

En tercer lugar tenemos a Winds. Sin duda, si sigue creciendo será la aplicación del mercado en este sector, ya que con tecnologías potentes y novedosas, con una estrategia de código abierto y con una orientación clara a la inteligencia artificial, se mantiene a la vanguardia de la innovación. Actualmente solo está disponible funcionalmente para escritorio y empezando a través de una aplicación web. Sin embargo, este hecho, hace que casi ni se le mencione.

En definitiva, y sobre todo haciendo hincapié en esta tercera, a estas herramientas les hace falta un punto de cocción para ser consideradas en la primera línea en cuanto al dominio del problema se refiere. Sin embargo, o ya han llegado a un público concreto, como las dos primeras, y no se plantean seguir creciendo, o todavía les hace falta más trabajo y más llegada al público objetivo, como la última.

CAPÍTULO 2

Estudio

2.1– Metodología LEAN

A medida que va acabando el siglo XX y empezando el XXI, los empresarios van siendo conscientes de que los métodos tradicionales para construir empresas han quedado anticuados. Ya no sirven aquellas reuniones interminables con acreedores para que financien proyectos, entre otros motivos, porque no hay sustento para estas. Además, la filosofía de institución de sociedades cambia de una versión más monolítica a una más ágil: las start-up.

Uno de los padres del nuevo método de emprendimiento es Steve Blank, y él es el que se cae en la cuenta y reformula la solución. Antes, para construir un producto, era necesario contar con una empresa, a la que cualquier banco estaba dispuesto a ceder un crédito. Sin embargo, ahora, se empiezan a construir empresas para desarrollar productos, por lo que no se posee ningún aval a priori.

Va pasando el tiempo, Steve Blank va enseñando su manera de emprender y aparecen dos personajes que van poblando el mismo bosque aunque con diferentes árboles. Estos son Eric Ries y Alex Osterwalder. El primero aplica la metodología LEAN al emprendimiento. El segundo, convierte el plan de negocio de una empresa a un gráfico sencillo de implementar y mostrar. A la par de estos acontecimiento, Blank llega hasta un concepto revolucionario dentro del emprendimiento: el desarrollo de los clientes.

Pues es en todo este marco de trabajo donde nace la teoría-práctica de Ash Maurya y su *Running Lean*. Esta metodología no hace más que unir los tres conceptos anteriormente mencionados: *Customer Development*, *Bussiness Model Canvas* y *Lean Startup*. Sin embargo, consigue algo que ninguno de los tres si quiera empezó: llevarlo a la práctica de una manera clara, concreta y sencilla.

Ash hace en su libro un rápido plan de acción para llevar a cabo el método LEAN a la construcción de una empresa que triunfe siguiendo una serie de procesos. Estos, se pueden resumir en el gráfico 2.1 que aparece en su libro. Así como la figura muestra un resumen del proceso a seguir, el resumen del libro se puede encontrar en la siguiente frase del mismo, aparecida en el primer capítulo del libro.

Running Lean is a systematic process for iterating from Plan A to a plan that works, before running out of resources.

Mauyria (2011)

En definitiva, no es más que documentar el plan inicial que se posee de la idea, identificar los posibles riesgos e ir sistemáticamente evolucionando nuestro plan hasta dar con el que el cliente comprará. Todo esto, como dice la cita, sin llegar a agotar los recursos. ¿Para qué es necesaria todas estas fases? Para conocer cuáles son los intereses reales del cliente. Parece trivial, pero no es sencillo: lleva tanto tiempo porque el mismo cliente no sabe qué problema posee.

2.1.1. Running Lean

Una vez visto un enfoque general de la metodología LEAN, se verá cómo funciona operativamente el *Running Lean* de Ash Maurya. Él distingue tres etapas dentro de la metodología, por lo que se comentará brevemente cada una de las fases.

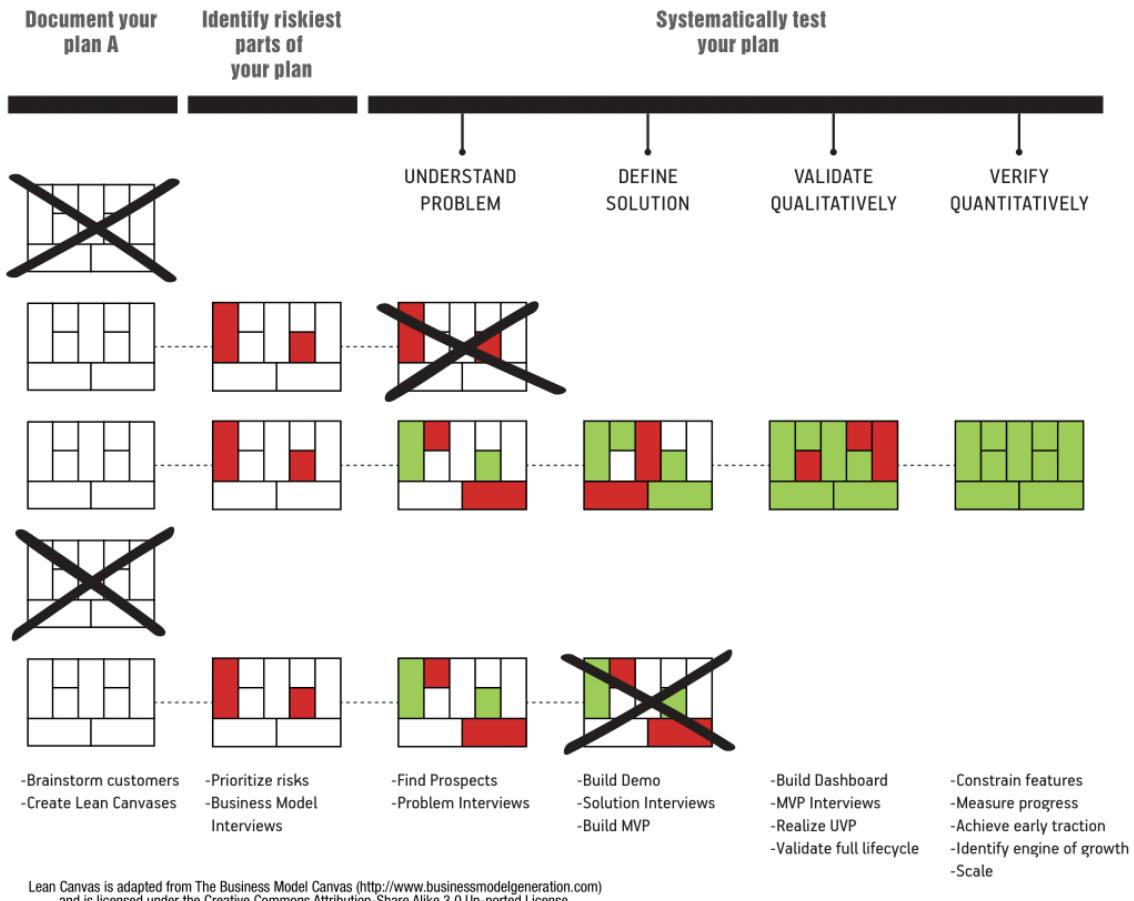


Figura 2.1: Resumen Running Lean

Documentar el plan A

La primera es la documentación de un plan inicial. Esto no es más que plasmar en un papel las ideas que se tienen sobre el producto. En vez de realizar un documento típico con todas sus partes y llenando cada una de las casillas que definen un producto tipo, Ash proporciona un *Lean Canvas*. Este es la evolución del *Bussiness Canvas* de Osterwalder. En este se llenan los atributos imprescindibles que definirán el producto con una condición: ha de ser un prototipo fácil de realizar y rápido en el tiempo.

Así, en un simple papel se puede realizar este y empezar a trabajar sobre el mismo. ¿No sirve? A tirarlo y a empezar de nuevo. Además que en este es posible ver de un simple vistazo las partes claves de la idea de negocio y por ello compartirlo para pedir opinión. Esto es fundamental, ya que la opinión del emprendedor con es imparcial. Este será una de las personas que más sepa del tema, y no será totalmente objetivo.

Este se rellena siguiendo un orden establecido, que ayudará a formar la cabeza de emprendedor para centrarse realmente en lo importante. La plantilla es la que se muestra a continuación.

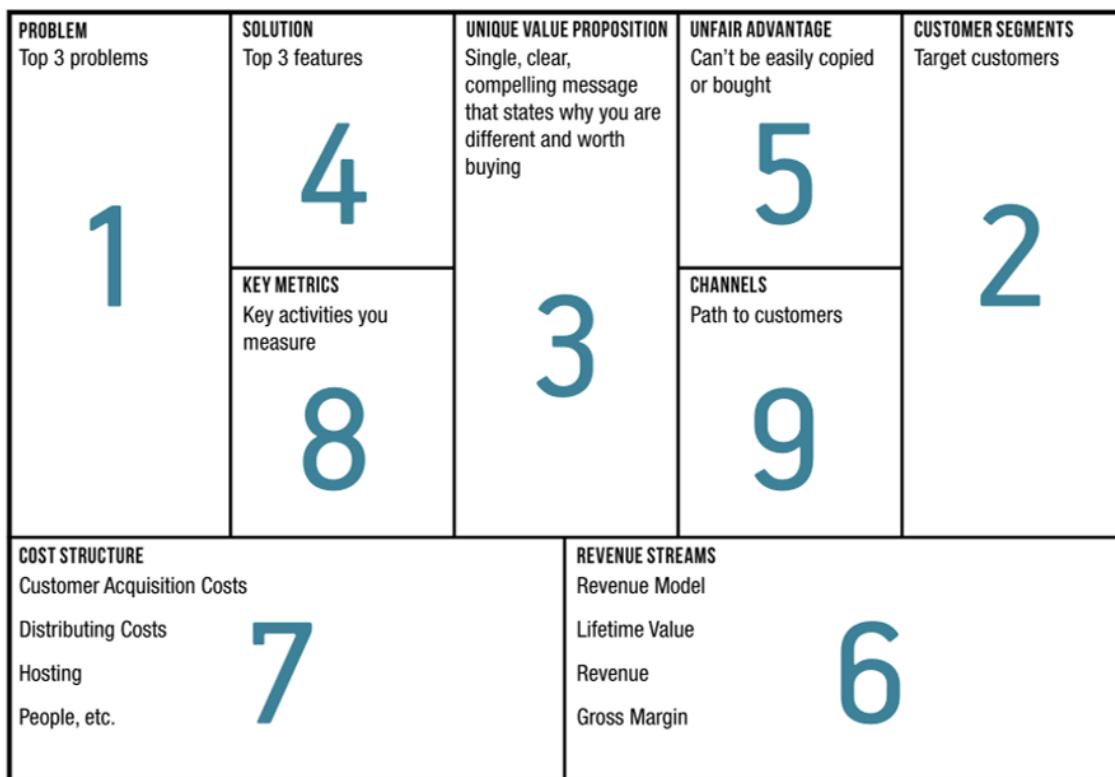


Figura 2.2: Lean Canvas

Las partes más importantes del *canvas* son los tres primeros puntos: problema, segmento de clientes y propuesta de valor única. Tan es así, que Ash afirma que:

Investors, and more important, customers, identify with their problems and don't care about your solution (yet). Entrepreneurs, on the other hand, are naturally wired to look for solutions. But chasing after solutions to problems no one cares enough about is a form of waste.

Mauyria (2011)

Es por tanto primordial encontrar el problema a solucionar, algo no trivial, porque debemos afinar en la necesidad real del cliente para así satisfacerla.

Identificar los riesgos del plan

Una vez se posee el primer esbozo de la idea, con una primera aproximación a las partes esenciales del producto y cómo será la venta de este en el mercado, se han de identificar unos riesgos iniciales. Todavía no se ha llegado a la madurez de la idea, pero es conveniente conocer ya los puntos flacos de esta.

Ash Maurya, de nuevo, ayuda con esto. Lo hace enseñando cuáles son las etapas por las que va a pasar el producto. Además, en cada una de estas, para que no se pierda el foco, muestra qué elementos se han de priorizar. Esta se puede ver en el siguiente gráfico.



Figura 2.3: Adecuación de nuestra idea al mercado

Para llegar a esos riesgos iniciales sin saber si quiera cuándo ni cómo se van a afrontar, Ash ilustra a los futuros emprendedores con las siguientes preguntas.

- **Problem/Solution Fit:** *Do I have a problem worth solving?*
 - *Is it something customers want? (must-have)*
 - *Will they pay for it? If not, who will? (viable)*
 - *Can it be solved? (feasible)*
- **Problem/Market Fit:** *Have I built something people want?*
- **Scale:** *How do I accelerate growth?*

Se centra sobre todo en el primer punto porque, como recomienda, hay que pivotar para llegar al problema concreto a solucionar. Esto ayudará cuando se pruebe el producto en la siguiente fase. Pero ya, contestando a las preguntas, es posible ver los riesgos a los que el emprendedor se verá sometido. Estos serán mayoritariamente de cinco tipos.

El primero hará referencia al problema. Este nos dirá cuánto *dolor* le quita al posible comprador. Esto es importante, porque podemos ver un problema real, que ocurra a un número alto de personas. Sin embargo, si este es visto como algo accesorio, nunca se conseguirá venderlo. Hay que ver, por tanto, si el problema merece la pena afrontarlo por el nivel de importancia que le de el usuario.

El segundo consiste en conocer el modo de llegar al público objetivo y a través de qué canales. Al igual que antes, si hay un buen producto, que soluciona un problema concurrente, pero que el público objetivo percibe como algo que le hace *sufrir*, no conseguirá llegar a ellos y, por tanto, no venderá.

El tercero tipo se relaciona con la estructura prevista de pérdidas y ganancias. Una vez que madurada la idea y encontrado el público objetivo y los canales para llegar a él, se hace necesario comprobar cómo se sostendrá económicaamente la idea. Por ello, es necesario analizar el margen de beneficio y la estructura de costes e ingresos.

El cuarto consistirá en el tamaño del mercado. Hay que comprobar cuál es el nicho de mercado y con quién hay de competir. Con esto, aseguraremos la audiencia y el objetivo a nivel de cuota de mercado.

El quinto y último a analizar será comprobar si técnicamente se puede implementar la solución propuesta. Solo con este criterio, se puede cambiar o alterar ligeramente la idea de negocio.

Probar sistemáticamente el plan

Ya tenemos el *canvas* y los posibles riesgos asociados. Ahora falta ir perfilándolo para llegar la madurez completa del producto. Las fases por las que pasará este testeo sistemático son las siguientes.

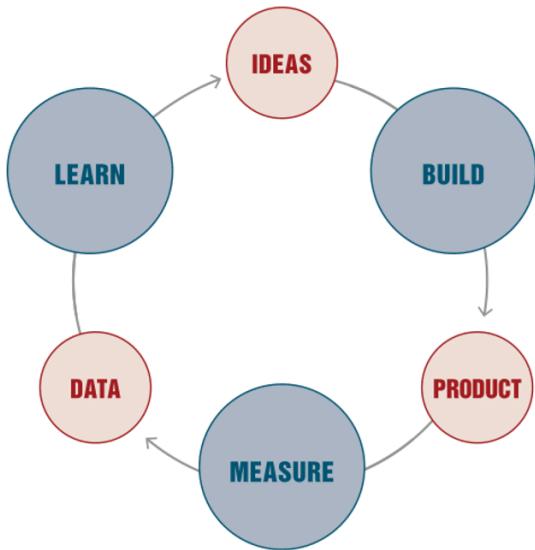


Figura 2.4: Prueba sistemática

Esta es la genialidad de Ash Maurya: el cómo convertir una metodología etérea en un algoritmo a seguir. Aquí unifica las ideas de Blank y Ries, porque es llevar la oficina a la calle. Esto no es otra cosa que preguntar y asentar ideas, preguntar y cambiar conceptos, preguntar y llegar a lo que la gente quiere.

Es por tanto primordial en esta fase salir a la calle y hacer entrevistas, muchas entrevistas. Estas llevarán al emprendedor a cambiar continuamente su *canvas* para acercarse más a la idea final.

Esta última parte contiene a su vez cuatro fases, que el emprendedor ha de ir quemando para llegar a la siguiente con algo cada vez más claro. En cada una de estas se ha de tener la mentalidad de aprender siempre algo.

Primero serán las entrevistas de problema. En estas se han de validar las hipótesis sobre el par: problema y segmento de cliente. De manera gráfica, Maurya propone *aprender* en esta fase lo siguiente.

Risk	Learn
Product risk: What are you solving?	How do customers rank the top three problems?
Market risk: Who is the competition?	How do customers solve these problems today?
Customer risk: Who has the pain?	Is this a viable customer segment?

Cuadro 2.1: Aprendizaje entrevista de problema

Una vez aprendido esto, se podrá saltar al siguiente punto. Esto se cristaliza en el hecho de poder identificar al *early adopter*, es decir, el primer usuario; si se tiene un problema que mereza la pena analizar; y si se puede averiguar cómo los usuarios potenciales resuelven a día de hoy este problema.

Una vez conseguido esto, se habrá llegado a las entrevistas de solución. Poseen la misma dinámica que la anterior, solo que ahora cambia lo que es necesario aprender. Además, hay de tener en cuenta que se está en la antesala de la salida a ventas del producto.

Risk	Learn
Customer risk: Who has the pain?	How do you identify early adopters?
Product risk: How will you solve it?	What is the minimum feature set needed to launch?
Market risk: What is the pricing model?	Will customers pay for a solution?

Cuadro 2.2: Aprendizaje entrevista de solución

Llegados a este punto, se tendrán unas características mínimas de el producto, un precio al que el cliente está dispuesto a pagarte y un negocio alrededor del producto.

Ya se poseen constancia de problema y solución, por lo que es necesario implementar la solución. Ahora, con lo que el emprendedor ha aprendido, no interesa enrolarse en hacer una aplicación 100 % funcional: hay que implementar solo aquello imprescindible para que sea vendida. Con esto, se hace un aseguramiento en tiempo y dinero. Hay que llegar a este Mínimo Producto Viable (MVP en adelante) para comprobar si el cliente si la solución implementada es acorde al problema real del cliente. Una vez hecho, se hace necesario volver a las entrevistas, pero sobre este MVP. Ahora, el emprendedor ha de aprender lo siguiente.

Risk	Learn
Product risk: What is compelling about the product?	Landing Page, Activation Flow, UVP
Customer risk: Do you have enough customers?	Channels
Market risk: Is the price right?	Price

Cuadro 2.3: Aprendizaje entrevista del MVP

Llegados a este punto, se poseerá una página explicativa del producto (*Landing Page*), el canal desde que el usuario llega a dicha página hasta que se interesa por el producto (*Activation Flow*) y un MVP que cumpla el UVP definido. Si en las entrevistas se aprecia que ya convence, solo queda lanzarlo al exterior y ver si se tienen los canales y precio adecuados. Esta será la última fase, continua a lo largo del tiempo: medir la adecuación entre el producto y el mercado.

Aquí se aprecia si convence el producto y cuál es el modo de llegar a que los usuarios paguen. Ya los *early adopter* habrán entrado en el flujo de clientes, por lo que se tienen que tirar de ellos para conseguir más usuarios. Además, ahora se recibirán continuas propuestas para mejorar el MVP que se harán necesarias analizarla su aplicabilidad.

2.1.2. Aplicación real

Como se puede observar, esta metodología es totalmente practicable a un proyecto real. De una parte, porque parte de la experiencia de varios autores ilustrados sobre el tema, llevando este a la práctica. De otra, porque posee una serie de pautas para aplicarlo en un entorno totalmente compartido.

Así, lo que se ha realizado en este proyecto, es la aplicación de la metodología LEAN, con su variante en el *Running LEAN* para proyectos de emprendimiento, a una idea propia. Esta, primero, era una idea puramente técnica, y con la ayuda y guías de este marco de trabajo, ha terminado por convertirse en una idea de negocio, al que poder dedicarle jornadas y jornadas de trabajo para llegar de la rentabilidad básica del mismo a un beneficio alto.

La correcta aplicación y su interiorización para poder llevarla a cabo ha ido de la mano de dos hechos. El primero es la realización de las asignaturas *Fundamentos de Innovación y Emprendimiento*, FEI, y *Taller de Innovación y Emprendimiento*, TEI. A través de estas asignaturas del máster, hemos estudiado y llevado a la práctica todos estos conceptos.

En FEI y TEI, en concreto, teníamos que llevar un proyecto a consecución siguiendo las claves que propone Maurya en su libro Maurya (2011). Yo, para poder entenderlo con todas sus variantes, he utilizado este proyecto, mi TFM, como idea de negocio. Así, he ido viendo problemas buscando

su resolución, he recibido feedback de parte de los profesores y alumnos, he aprendido de los proyectos ajenos. Como se ve, podría ser la fiesta del saber en cuanto a emprendimiento se refiere.

El segundo hecho importante para llevar este Estudio a término ha sido la lectura y estudio del libro tantas veces citado a lo largo de este apartado. *Running Lean* me ha enseñado que lo aprendido en las asignatura de Emprendimiento del Máster tienen bastante de fundamento real e innovador. Además, estudiado el libro he comprobado cómo llevar las teorías de Maurya hasta límites insospechados.

Así, a lo largo de los siguientes puntos, iré desgranando los pasos que se indican en *Running Lean* y cómo los he llevado a la práctica en mi proyecto. Además, se indicarán una serie de conclusiones sacadas de la realización del mismo.

Como se puede intuir viendo lo anterior, no tendrá sentido aplicar totalmente la metodología ya que la consecución del TFM no es otra que el aprendizaje de tecnologías y conocimientos, y no tanto la venta de productos. Esto, aunque no descartado en un marco futuro, no se realizará en las iteraciones LEAN. Por tanto, se quedará el Estudio hasta el conocimiento certero de la problema y su posterior solución.

2.2– Primer estudio

LT-News, como se ha dicho, es un lector de noticias inteligentes, capaz de detectar los gustos de los usuarios en base a su utilización de la aplicación, de relacionar noticias entre sí y de detectar las noticias más importantes del día.

El objetivo principal del proyecto es filtrar las noticias de los periódicos y quedarse solo con aquellas realmente interesantes para el usuario. El objetivo secundario y subyacente al anterior no es otro que el de no perder el tiempo.

Con estas premisas de la idea de negocio, se ha construido un canvas, que representa cada una de sus partes, como se puede apreciar en el siguiente cuadro.

-Gran cantidad de medios a elegir. -Gran cantidad de noticias a leer. -Fakes news y sesgo de los medios. -Perderse novedades de tus intereses.	- Recomendación de noticias. -Extracción del perfil de usuario y recomendar noticias. -Análisis diario de prensa. -Tiempo pasado en aplicación. -Número de usuarios. -Suscripciones mensuales.	-Algoritmo de inteligencia artificial capaz de extraer el perfil del usuario y de las noticias y relacionarlos entre sí. -Ahorrar tiempo y a la vez que estar al día.	-Análisis de textos de las noticias. -Webscrapping de las noticias en los medios. -Web. -Aplicaciones.	-Usuarios lectores de noticias. -Jóvenes y mediana edad. -Cultura e interés actualidad. -Personas asociadas a la tecnología.
-Coste de servidores.		-Aplicación premium: 3€/mes.		

Figura 2.5: LEAN Canvas inicial

Con esta aproximación a todo el entramado de la idea, se van a hacer especial hincapié en los puntos principales: clientes, problemas y propuesta de valor único. En base a esto, se realizarán las hipótesis de problemas para poder contrastarlos.

2.2.1. Segmento de clientes

El cliente objetivo es el lector actual de noticias, joven o de mediana edad con cultura e interés sobre temas de actualidad y con manejo de las nuevas tecnologías.

2.2.2. Problema

Se encuentran principalmente tres problemas con los que se enfrenta el usuario. Primero la gran cantidad de medios a elegir, por tanto, de noticias a leer. La segunda es que las noticias a leer no son objetivas, es decir, el usuario encuentra frecuentemente noticias falsas y otras afectadas por el sesgo de los medios. La tercera es que, entre tanta cantidad de noticias, se pierden las noticias que realmente interesan.

2.2.3. Propuesta de valor único

La UPV es el algoritmo de inteligencia artificial capaz de extraer el perfil del usuario y de las noticias y de relacionarlas entre sí. Dicho de otro modo, ahorrar tiempo a la vez que estar al día.

2.2.4. Principales hipótesis

- A los lectores de noticias les preocupa la gran cantidad de noticias.
 - Es la hipótesis más arriesgada porque si la respuesta es que no, hay que pivotar completamente.
- A los lectores de noticias les preocupa que las noticias no sean objetivas.
 - Es una hipótesis que requiere ser probada para seguir validando el producto, aunque es parte de una futura funcionalidad secundaria.
- A los lectores de noticias les interesa estar al día de sus temas favoritos.
 - Es otra hipótesis secundaria para comprobar.

2.3– Iteraciones

El resumen de las iteraciones realizadas sobre el producto se encuentran en esta tabla.

Iteración	Hipótesis	Fechas	Tests	Resultado
Problema 1	Primera hipótesis	S/29-09	2 entrevistas	No aplica
Problema 2	Primera hipótesis	S/06-10	4 entrevistas	Se confirma
Problema 3	Segunda hipótesis	S/13-10	4 entrevistas	Se confirma
Problema 4	Tercera hipótesis	S/20-10	143 encuestas	No se confirma
Solución 1	Soluciones	S/27-10	4 entrevistas	Se confirma

Cuadro 2.4: Iteraciones realizadas

El principal pivote realizado es el de simplificar el Producto Mínimo Viable. Esto es debido a la realidad de que un problema no afecta de igual manera a los *early adopters*. Por tanto, aunque el seguimiento de los temas favoritos sea importante, no se incluirá en la primera versión. Todos los demás problemas se han confirmado al cien por cien.

Con respecto a las soluciones, se han confirmado, aunque los usuarios piden más características. Dado que en la primera versión ya se ha demostrado un MVP aceptable por los usuarios, se tomarán estas funcionalidades pedidas para futuras versiones.

En relación a los *early adopters*, he cambiado su percepción. Antes los consideraba como personas cultas interesadas por lo que ocurre en la actualidad y que siguen una serie de temas. Ahora, veo que puede ser cualquiera que ya use un lector RSS o aplicación de un periódico. En concreto, me centro en los que usen Feedly.

2.3.1. Primera iteración

La primera iteración consistió en confirmar la primera hipótesis, es decir, confirmar el primer problema. Para ello, se hicieron dos entrevistas. De ellas no se sacaron ninguna conclusión, simplemente se redefinió lo que se consideraba por *early adopter*. Hasta este momento se les tenía por personas que eran jóvenes, por interés por la cultura y con gran uso de smartphone y ordenadores. Fue entonces cuando comprobé que estos eran más bien personas que ya usan un lector RSS o, sin más, una aplicación de información en su móvil.

Son estos mis *early adopters* por un motivo: a estos usuarios les interesa estar al día de diferentes medios y ya han buscado una solución para estar al día, que no es otra que utilizar un lector RSS. Además, si llevan tiempo utilizándolo, se habrán dado cuenta del principal problema de este: la enorme cantidad de noticias a leer.

-Gran cantidad de medios a elegir.	- Recomendación de noticias.	-Algoritmo de inteligencia artificial capaz de extraer el perfil del usuario y las noticias y relacionarlos entre sí.	-Análisis de textos de las noticias.	- Usuarios activos de lectores de RSS.
-Gran cantidad de noticias a leer.	-Extracción del perfil de usuario y recomendar noticias.	-Análisis diario de prensa.	-Webscrapping de las noticias en los medios.	- Usuarios que suelen usar Feedly.
-Fakes news y sesgo de los medios.	-Análisis diario de prensa.	-Tiempo pasado en aplicación.	-Ahorrar tiempo y a la vez que estar al día.	-Web. -Aplicaciones.
-Perderse novedades de tus intereses.	-Número de usuarios.	-Coste de servidores.		
-Aplicación premium: 3€/mes.				

Figura 2.6: LEAN Canvas primera iteración

2.3.2. Segunda iteración

Viendo lo anterior, se decidió volver a preguntar lo mismo, pero a los verdaderos usuarios potenciales. Como se dijo, fue importante confirmar la primera hipótesis, ya que de esta partía todo el problema, y, por ende, todo el proyecto. Es por ello que se realizaron cuatro entrevistas siguiendo el siguiente esquema:

1. Welcome
 - a) Presentación personal
 - b) Qué quiero de esta entrevista
2. Collect Demographics

- a) Nombre
- b) Email
- c) Edad
- d) Código Postal
- e) Uso de apps de noticias
- f) Veces a la semana que lee el periódico

3. Tell a Story

- a) Contexto de la cantidad de noticias en la actualidad.
- b) Leer en papel vs Leer en formato digital: actualidad vs concentración.

4. Problem Ranking

- a) ¿Qué problemas tienes a la hora de estar al día de todas las noticias que te interesan?
Dime en orden.
- b) ¿Piensas que las noticias que lees están dadas desde un punto de vista o pueden ser falsas? ¿Te preocupan?
- c) ¿Alguna vez has visto una noticia de un tema que te interesaba y te ha dado pena no haberte enterado antes?
- d) ¿Usas alguna aplicación de noticias?

5. Explore Customer Overview

- a) Ver su opinión sobre la manera de leer noticias

6. Wrapping Up

- a) Te gustaría una aplicación donde: ¿se recojan todas las noticias, te las relaciona y te las recomienda?

7. Document Results

- a) Pasarlo a un Google Forms

Sobre esto se ha conseguido confirmar la primera hipótesis de manera rotunda, aunque la segunda y la tercera no ha quedado del todo claras. Es claro que el problema principal existe y la gente que ya lee en sindicadores de contenido ha intentado plantear soluciones en su día a día para solventarlo.

Sobre las entrevistas y el feedback recibido en clase, sí que se han modificado los canales. Antes eran un poco genéricos; ahora, intentan llegar más a los *early adopter*. Por tanto, se va a intentar que publiquen la aplicación en blogs de aplicaciones, se van a mencionar en blogs y foros especializados en el tema y se va a intentar hacer SEO de la aplicación con los tags: aplicación, noticias, inteligente. Algunas de estas podrían ser:

- El Androide Libre - <https://elandroidelibre.elespanol.com>
- Xataka - <https://www.xatakandroid.com/>
- Android4All - <https://andro4all.com/>

-Gran cantidad de medios a elegir. -Gran cantidad de noticias a leer. -Fakes news y sesgo de los medios.	- Recomendación de noticias. -Extracción del perfil de usuario y recomendar noticias. -Análisis diario de prensa.	-Algoritmo de inteligencia artificial capaz de extraer el perfil del usuario y de las noticias y relacionarlos entre sí.	-Análisis de textos de las noticias. -Webscrapping de las noticias en los medios.	-Usuarios activos de lectores de RSS. -Usuarios que suelen usar Feedly.
-Perderse novedades de tus intereses.	-Tiempo pasado en aplicación. -Número de usuarios. -Suscripciones mensuales.	-Ahorrar tiempo y a la vez que estar al día.	-Blogs sobre aplicaciones. -Foros especializados. -SEO de las keywords.	
-Coste de servidores.		-Aplicación premium: 3€/mes.		

Figura 2.7: LEAN Canvas segunda iteración

2.3.3. Tercera iteración

La tercera iteración consistió en confirmar, de manera sistemática y similar a la anterior, las dos hipótesis restantes. Para ello, se volvieron a hacer cuatro entrevistas. De estas salió validada la segunda hipótesis y algo clara la tercera, es decir, la mitad lo afirmaron como problema y la otra mitad, no les parecía relevante.

Aquí también salieron algunas características adicionales para añadir al producto que se considerarán en el futuro, ya que no están dentro del producto mínimo viable y son las siguientes:

- Sacar de una noticia su relación con temas de fondo o contexto en el que se encuentra. Para esto es necesario llegar a contenido de calidad.
 - Una aproximación podría ser relacionar temas de las noticias con Wikipedia o fuentes relevantes y más profundas.
 - Para móviles Android, hacer que la aplicación tenga Widgets, ya que algunos usuarios estaban interesados.
 - Sobre los resultados de las entrevistas, hemos cambiado la formulación de los dos primeros problemas.

<p>-Enorme cantidad de noticias a leer.</p> <p>-Asegurar objetividad de las noticias.</p> <p>-Perderse novedades de tus intereses.</p>	<ul style="list-style-type: none"> - Recomendación de noticias. - Extracción del perfil de usuario y recomendar noticias. - Análisis diario de prensa. 	<ul style="list-style-type: none"> -Algoritmo de inteligencia artificial capaz de extraer el perfil del usuario y de las noticias y relacionarlos entre sí. 	<ul style="list-style-type: none"> -Análisis de textos de las noticias. -Webscrapping de las noticias en los medios. 	<ul style="list-style-type: none"> -Usuarios activos de lectores de RSS. -Usuarios que suelen usar Feedly.
	<ul style="list-style-type: none"> -Tiempo pasado en aplicación. -Número de usuarios. -Suscripciones mensuales. 	<ul style="list-style-type: none"> -Ahorrar tiempo y a la vez que estar al día. 	<ul style="list-style-type: none"> -Blogs sobre aplicaciones. -Foros especializados. -SEO de las keywords. 	
<p>-Coste de servidores.</p>		<p>-Aplicación premium: 3€/mes.</p>		

Figura 2.8: LEAN Canvas tercera iteración

2.3.4. Cuarta iteración

Como se ha dicho, después de haber validado las dos primeras hipótesis como Producto Mínimo Viable, quedaba la tercera. Dado que las entrevistas individuales en este caso resultaban inefficientes porque no había ninguna opinión que esclareciera el asunto, se decidió realizar una encuesta y pasarla por diferentes canales. Gracias a este medio se consiguieron tres cosas, en este orden de importancia:

- confirmar la opinión sobre la tercera hipótesis,
- conocer el comportamiento por los usuarios de algunas características del sistemas,
- saber qué funcionalidad echan los potenciales interesados en falta.

Se les preguntó, sacado del guión de entrevistas, datos demográficos y confirmación de las tres hipótesis. No fueron solamente preguntas exclusivas, ya que no me darían ningún dato. Se les incluyó un lugar donde se solicitaban la expresión de opiniones al respecto o modus operandi actuales. Clave fueron las preguntas sobre cómo intentan resolver alguno de sus problemas. Así se puede ver cuánto le interesa realmente eso que ha respondido.

El resumen fue el siguiente (la encuesta está en <https://goo.gl/forms/UJHY6o6lV3JxmM712>).

Edad

143 respuestas

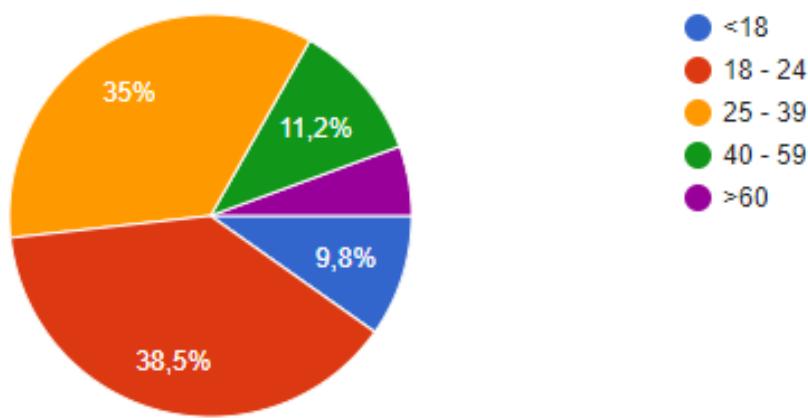


Figura 2.9: Encuesta primera pregunta

¿Cuantas horas a la semana dedicas a leer noticias?

143 respuestas

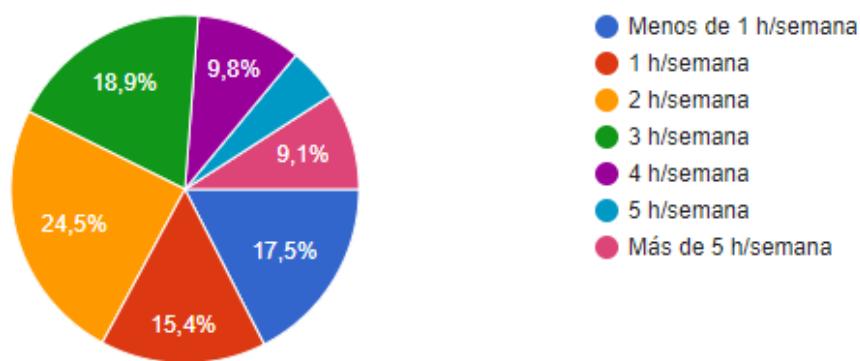


Figura 2.10: Encuesta segunda pregunta

¿Qué aplicación sueles usar para leer noticias?

138 respuestas

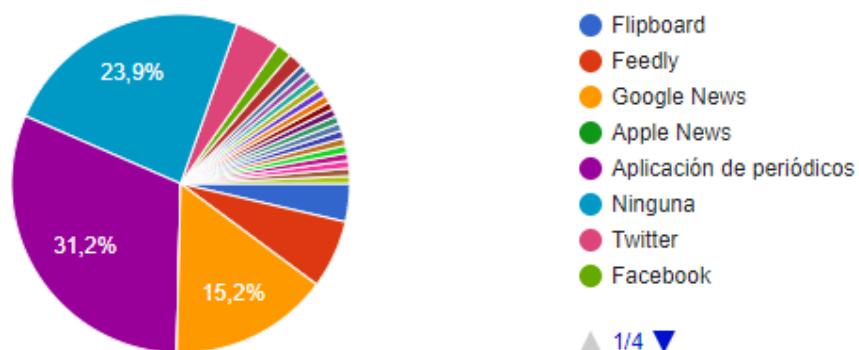


Figura 2.11: Encuesta tercera pregunta

¿Intentas estar al día de lo que pasa en el mundo?

138 respuestas

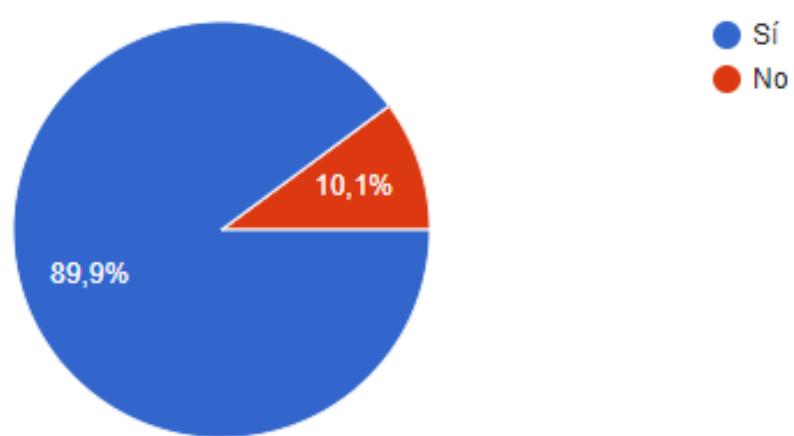


Figura 2.12: Encuesta cuarta pregunta

¿Te preocupa que las noticias que lees sean falsas?

138 respuestas

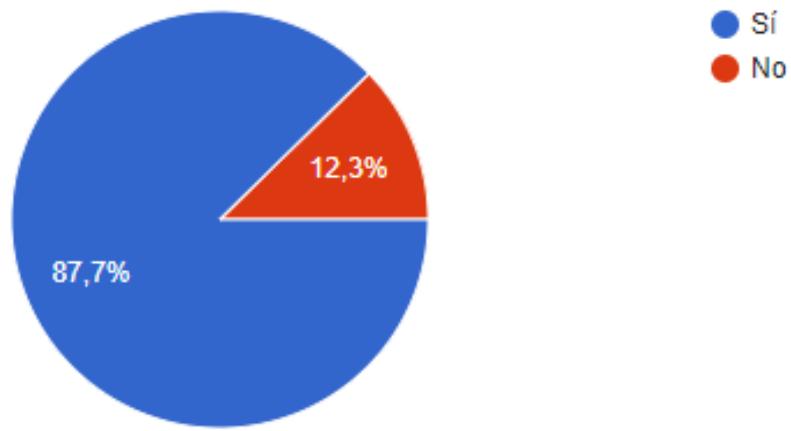


Figura 2.13: Encuesta quinta pregunta

¿Ves que muchas veces lees noticias que no sabes el contexto o la cronología de los hechos?

138 respuestas

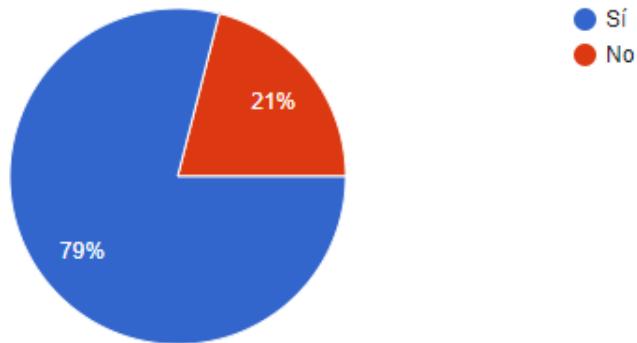


Figura 2.14: Encuesta sexta pregunta

¿Intentas estar a la última sobre los temas o noticias que te interesan?

138 respuestas

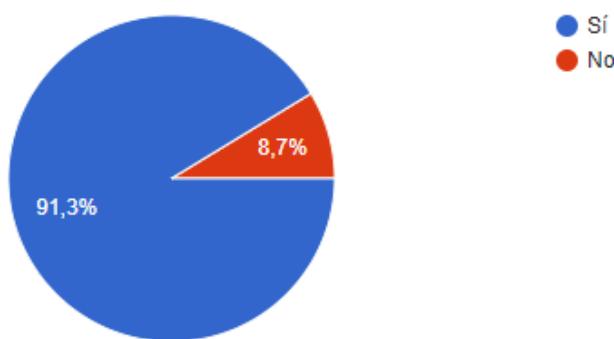


Figura 2.15: Encuesta séptima pregunta

¿Te preocupa que muchas veces te enteras de estos temas tarde o por otros, en vez de por tí mismo?

138 respuestas

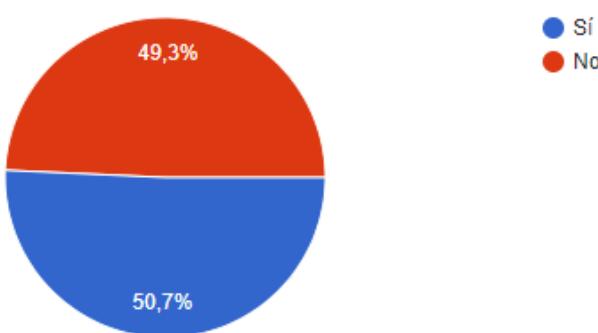


Figura 2.16: Encuesta octavo pregunta

Como se pudo ver, no quedó confirmado la tercera hipótesis, por lo que se quitará del Producto Mínimo Viable y se añadirá como primera característica futura, ya que interesa a la mitad de los usuarios. Por otra parte, se reafirmaron las dos primeras hipótesis. Además, salieron de ahí diferentes opiniones, así como futuras características a considerar en un futuro. Algunos son:

- Hacer una aplicación que consuma poco espacio.
- Hacer cómoda la lectura del cuerpo de la noticia.
- Evitar los problemas de conexión a internet.

Con esto, nos queda el siguiente Canvas:

-Enorme cantidad de noticias a leer. -Asegurar objetividad de las noticias.	- Recomendación de noticias. -Extracción del perfil de usuario y recomendar noticias. -Análisis diario de prensa.	-Algoritmo de inteligencia artificial capaz de extraer el perfil del usuario y de las noticias y relacionarlos entre sí. -Ahorrar tiempo y a la vez que estar al día.	-Análisis de textos de las noticias. -Webscrapping de las noticias en los medios.	-Usuarios activos de lectores de RSS. -Usuarios que suelen usar Feedly.
	-Tiempo pasado en aplicación. -Número de usuarios. -Suscripciones mensuales.		-Blogs sobre aplicaciones. -Foros especializados. -SEO de las keywords.	
-Coste de servidores.		-Aplicación premium: 3€/mes.		

Figura 2.17: LEAN Canvas cuarta iteración

2.3.5. Quinta iteración

Por último, una vez confirmados los problemas, se hicieron dos entrevistas de solución a usuarios potenciales de la aplicación, dispuestos a probar todas las versiones de este. Estos son realmente *early adopters*, dispuestos a pagar en cuanto salga al mercado si ven un producto fiable.

Para esto, se procederá a enseñar una demo. Esta se hará primero con la *Landing Page*. En esta se ha intentado mostrar las futuras características del proyecto, así como recoger datos de usuarios. Esta se encuentra en varios idiomas, con idea de empezar a distribuirla por personas que habla inglesa o incluso francesa. El motivo de esto último se debe a mi cercanía con personas que residen y pertenecen a estos países.

Con lo dicho, la página resultante quedó de la siguiente manera:

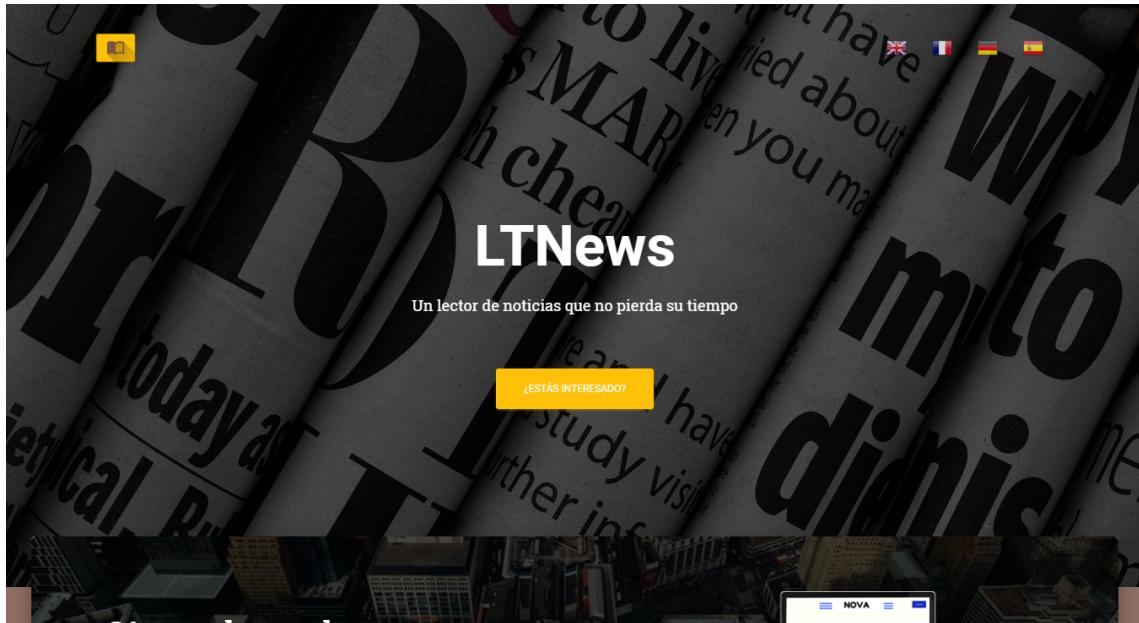


Figura 2.18: Landing Page 1



Figura 2.19: Landing Page 2

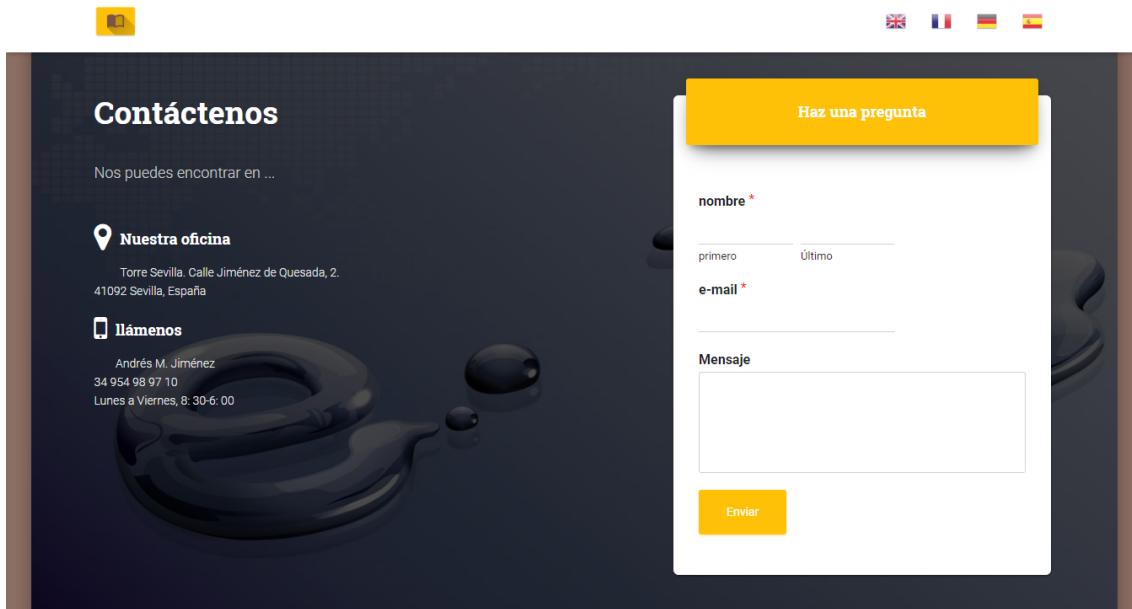


Figura 2.20: Landing Page 3

Por otra parte, se le dio tráfico, para saber el comportamiento de los posibles usuarios. Este fue analizado utilizando la herramienta de Google Analytics.

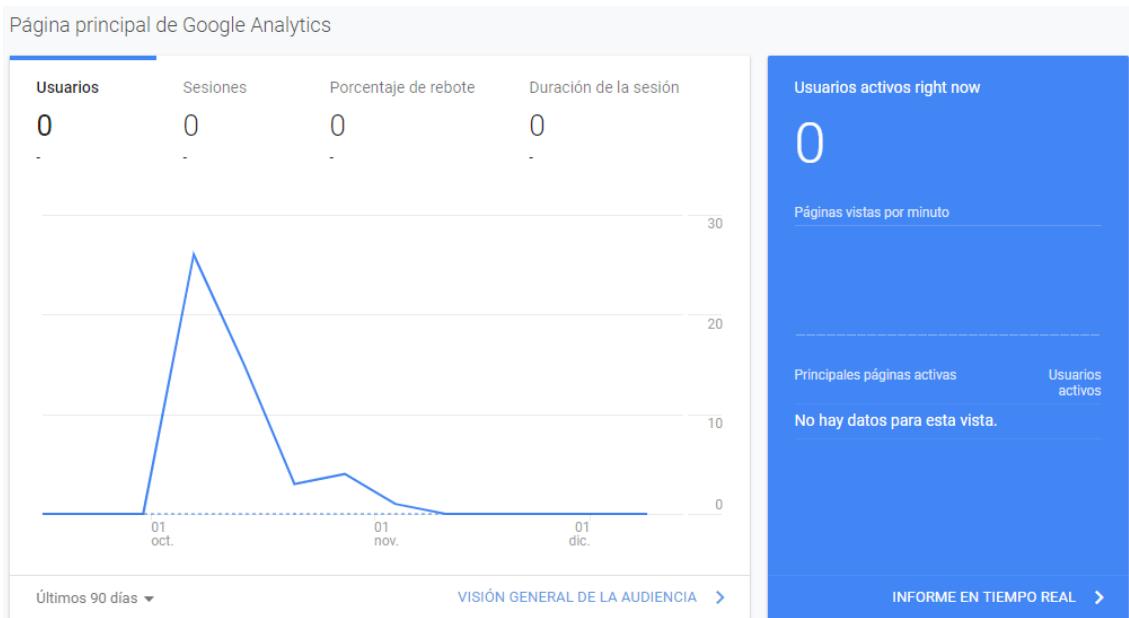


Figura 2.21: Google Analytics 1

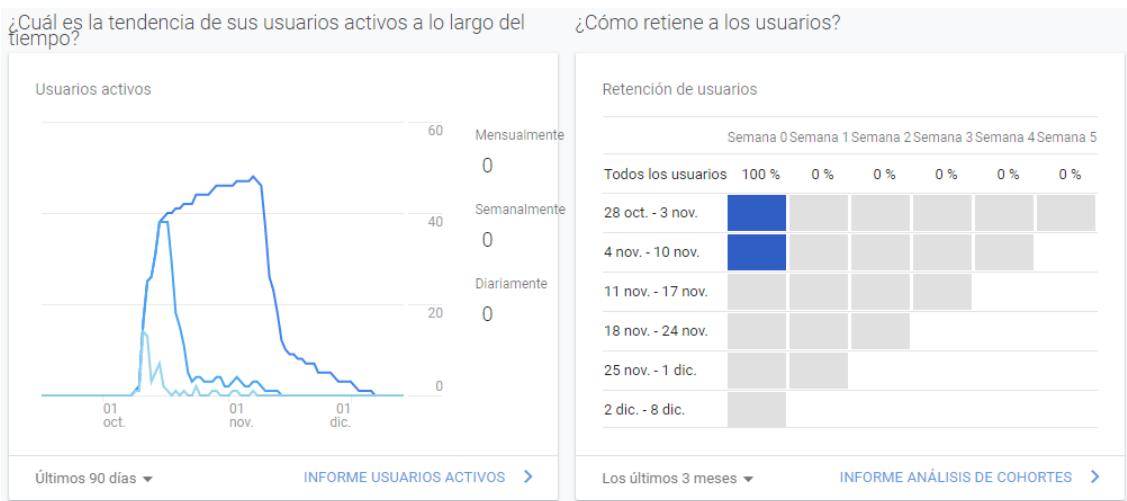


Figura 2.22: Google Analytics 2

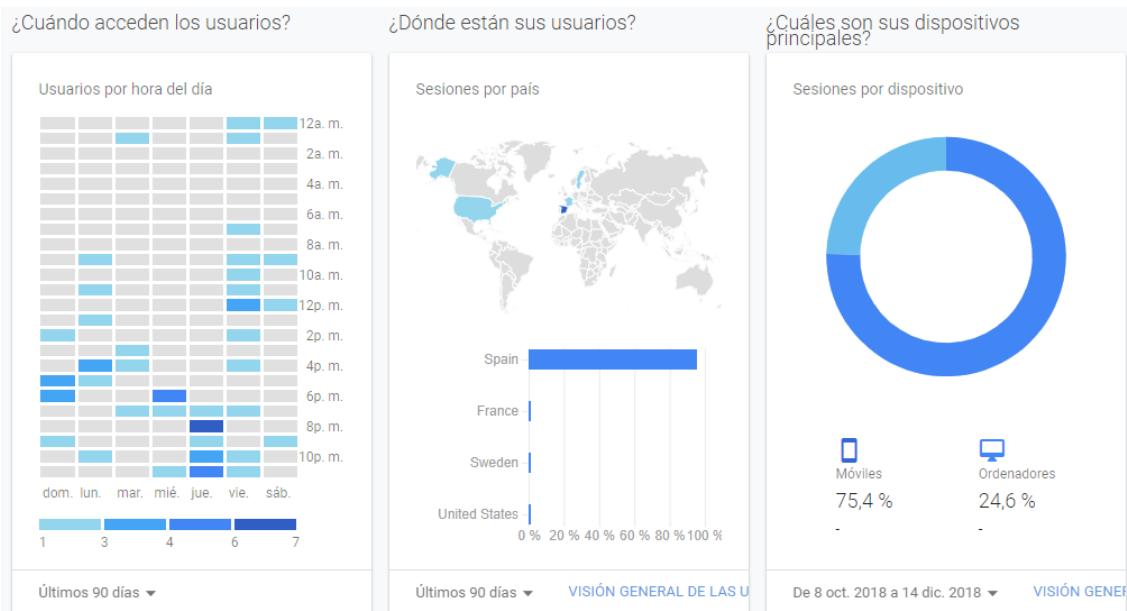


Figura 2.23: Google Analytics 3

Con esto, se hicieron las entrevistas de solución. La estructura de la entrevista fue la siguiente:

1. Welcome
 - a) Presentación personal
 - b) Qué quiero de esta entrevista
2. Collect Demographics
 - a) Nombre
 - b) Email
 - c) Edad
 - d) Código Postal

- e) Uso de apps de noticias
 - f) Veces a la semana que lee el periódico
3. Tell a Story
 - a) Contexto de la cantidad de noticias en la actualidad
 - b) Leer en papel vs Leer en formato digital: actualidad vs concentración
 4. Demo
 - a) Le enseño la Landing Page
 - b) Hablamos de cómo voy a solucionar estos problemas y le pregunta su opinión
 5. Test Pricing
 - a) ¿Qué te parece el plan de pagos?
 6. Wrapping Up
 - a) ¿Me podrías dar tu email para seguir contándote cómo sigue el proyecto?
 - b) ¿Conoces a alguien que pueda estar interesado en el proyecto?
 7. Document Results
 - a) Pasarlo a un Google Form

En sendas entrevistas, se volvieron a confirmar la solución, explicando lo que se iba a realizar y cómo se implementaría. Pareció correcto, estando los *early adopters* expectantes a futuras versiones.

2.3.6. Quinta iteración

Finalmente, validado ya el canvas, tanto a nivel de solución como de problemas, tanto cualitativa como cuantitativamente, este ha variado en algunos aspectos que vamos a considerar. Todo lo demás, en definitiva, se ha explicado al principio del apartado.

El mayor cambio ha consistido es la variación de concepto del usuario potencial, esto gracias a las entrevistas realizadas. Otro gran cambio ha sido la formulación de los problemas del usuario, que se han adaptado a un lenguaje más cercano a este, así como el evitar generalidades. Del fruto de estas directrices, ha quedado un MVP más conciso y acercable a los usuarios.

Con respecto a las soluciones, se han mantenido como estaban inicialmente, ya que han sido validadas por los usuarios. Esto se debe a que previamente a estos experimentos, se buscó las mejores soluciones posibles atendiendo a los problemas que se conocían a priori.

-Enorme cantidad de noticias a leer. -Asegurar objetividad de las noticias.	- Recomendación de noticias. -Extracción del perfil de usuario y recomendar noticias. -Análisis diario de prensa.	-Algoritmo de inteligencia artificial capaz de extraer el perfil del usuario y de las noticias y relacionarlos entre sí. -Ahorrar tiempo y a la vez que estar al día.	-Análisis de textos de las noticias. -Webscrapping de las noticias en los medios.	-Usuarios activos de lectores de RSS. -Usuarios que suelen usar Feedly.
	-Tiempo pasado en aplicación. -Número de usuarios. -Suscripciones mensuales.		-Blogs sobre aplicaciones. -Foros especializados. -SEO de las keywords.	
-Coste de servidores.		-Aplicación premium: 3€/mes.		

Figura 2.24: LEAN Canvas final

2.4– Conclusión estudio

La mayor conclusión no ha sido otra que el centrarse durante todo el proceso creativo y sistemático en el posible usuario. Esto ha sido posible gracias a entrevistas personales, cercanía con los posibles usuarios, adaptación del lenguaje técnico, así como otras técnicas sacadas por Ash Maurya en *Running Lean*.

De otra manera, personalmente, hubiese adoptado por otra técnica. Esta hubiese sido construir el producto como entendía e intentar después venderlo. Ante opiniones de usuarios, hubiese desarrollado estas *sin ton ni son*, solo por el simple hecho de vender y de hacer que estuviesen contentos.

Así, en cambio, sin haber construido nada, sin invertir tiempo ni cabeza en la implementación, ya vislumbro que es lo que comprará un posible usuario. Esto hace el trabajo más eficiente, así como más humilde. Humilde por el hecho de no canonizar mis ideas preconcebidas, sino pasarlas por el crisol de la opinión de los demás.

CAPÍTULO 3

Planificación

3.1– Metodología

Todo proyecto que esté englobado dentro de la ingeniería del software posee una metodología que proporciona un marco de trabajo estructurado, planificado y controlado para el proceso de desarrollo. Sin embargo, no hay una única metodología a aplicar, sino más bien múltiples y variadas. Lo importante no es tanto escoger la *perfecta*, que no existe, sino ver aquella o aquellas que mejor se ajusten a la naturaleza del proyecto en cuestión.

Como se ha podido observar anteriormente, hace falta una metodología que agilice el desarrollo dada la tipología del proyecto. Es por tanto imprescindible el uso de una metodología *agile* o ágil. Esta se orienta a la construcción y entrega. Para conseguir esto, reduce el tiempo de desarrollo. Dado que el producto no se puede reducir conceptualmente, habrá que construir por fases para conseguir esto.

Dicho lo anterior, se puede apreciar que los procesos de especificación, diseño y la misma implementación son concurrentes. Utiliza, por ello, un ciclo de vida evolutivo, en el que el ciclo requisitos-desarrollo-evaluación termina en una versión del proyecto. Esta puede ser probada por el cliente, dando mayor oportunidad a feedback sobre el entendimiento de los requisitos.

La metodología ha sido elegida por la naturaleza del proyecto, como se ha dicho. Además, al sector al que va dirigido es muy concreto y, a la vez, diferente. Así, no se conocían a priori todos los requisitos, porque no se tenían unos clientes concretos a los que preguntar que abarcasen la representación del conjunto. Esto se ha podido ver con propiedad cuando se ha hablado de las entrevistas con *early adopters*, en el punto 2.3.

Así, los requisitos, se han ido conociendo a lo largo del desarrollo del trabajo. Se ha intentado que esto fuera mínimo siguiendo la metodología *Running Lean*, aunque los métodos preventivos nunca previenen al completo sobre posibles problemas. Así, como se decía, sin todos los requerimientos por parte del usuario, no es posible comenzar un ciclo de vida clásico, ya que en este es necesario conocer, idealmente, todos los requisitos del proyecto.

Por otra parte, se necesitaba tener, en poco tiempo, una versión testeable de la aplicación, para obtener un flujo de información más claro sobre lo que se desea realmente del aplicativo. Este es el segundo argumento a favor de la metodología ágil, ya que trabaja continuamente con versiones de la aplicación, al acabar el ciclo antes mencionado, y estas pueden ser fácilmente enseñables a futuros clientes para recibir opiniones.

Sin embargo, como se ha dicho, *ninguna metodología es perfecta*. *Agile* presenta algunos aspectos negativos, como son la difícil planificación por los requisitos cambiantes o la falta de cobertura en todas las áreas del proceso, entre otras. No obstante, se asumen por la naturaleza del proyecto, en favor de los motivos mencionados más arriba.

Aunque se ha hablado continuamente de la metodología ágil y su aplicación al proyecto, esto

no es tan directo. Es necesario descender a marcos de trabajo que especifiquen y complementen dichas formas de trabajo. Las elegidas para este proyecto serán SCRUM y Kanban, ambas complementarias, que se explicarán en los siguientes puntos.

3.1.1. SCRUM

Dentro de la metodología anteriormente comentada, es necesario descender a los detalles del día a día. De esta manera surge SCRUM como un marco de trabajo que permite el trabajo conjunto entre miembros de un mismo equipo. Su etimología proviene del rugby, y al igual que en él, a la hora de entrenar, se ha de basarse en la experiencia, organizarse para resolver los problemas y reflexionar sobre los resultados obtenidos para mejorar.

SCRUM es un marco que provee a cualquier equipo de trabajo de unas pautas para poder aplicarse. Este describe un conjunto de funciones, herramientas y reuniones que hacen trabajar de manera coordinada a los miembros del equipo. Dicho lo anterior, se puede ver que SCRUM es un marco de trabajo, mientras que la metodología ágil es un ideal. Por ello no se puede trabajar directamente con *agile*, sino que es necesario recurrir a operativas que implementan, o al menos, desarrollan dicha mentalidad.

Este marco de trabajo se basa en el aprendizaje y adaptación al medio. Esta heurística hace al equipo humilde, ya que reconoce que no posee el pleno conocimiento al inicio, y que irá evolucionando a lo largo de la realización del proyecto. Hace, de manera soterrada, que el equipo de trabajo se vaya adaptando a los cambios tanto de las condiciones como del cliente. Esto se consigue presentando a los interesados continuas versiones del producto o servicio que se va desarrollando, además de un orden por prioridad de las peticiones del cliente.

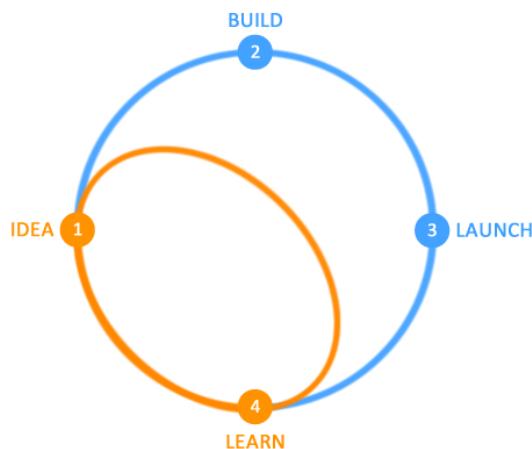


Figura 3.1: Aprendizaje en SCRUM

Como se puede apreciar en la imagen, el aprendizaje de SCRUM sigue la estructura: IDEA-CONSTRUCCIÓN-DESPLIEGUE-APRENDIZAJE. Es foco lo tiene el aprendizaje, y no tanto el desarrollo. Sin embargo, no es este un marco rígido. SCRUM se adapta sin problema a cualquier tipo de proyecto, estableciendo únicamente algunas pautas a seguir. Lo que de verdad importa, y a esto va dirigido este marco, es hacer que la comunicación sea clara, la transparencia sea real y la dedicación sea continua.

A continuación se desgranarán cada una de las partes, componentes y fases que presenta SCRUM. Esta clasificación y explicación ha sido fruto del estudio del artículo de Claire Drumond (2019).

Artefactos de SCRUM

Los artefactos son aquellos desarrollos que se realizan para solucionar el problema. En SCRUM son tres: *Product Backlog*, *Sprint Backlog* e *Increment*.

El *Product Backlog* es la lista de tareas que debe realizar el equipo para desarrollar con éxito el entregable para ser presentado al cliente. Esta es una lista con fluctuaciones en la que se encuentran requisitos, mejoras y correcciones a realizar sin indicar a priori una planificación de la misma. Cada tarea posee una prioridad que irá cambiando dependiendo de múltiples factores: cliente, entorno, sistema, etc.

El *Sprint Backlog* es una lista de elementos a realizar en un periodo concreto propuesto por el equipo. Esta se alimenta del *Product Backlog* y se encuentra cerrada durante el sprint. De hecho, antes de iniciar este periodo de tiempo, en la reunión de planificación del sprint, el equipo elige aquellas tareas del producto en las que trabajará.

El *Increment* es el producto desarrollado al final del sprint. Este está estrechamente relacionado con el concepto de finalizado. En este sólo se incluirán los elementos del *Sprint Backlog* que se hayan finalizado durante el sprint. Será el equipo de desarrollo el que indique qué entiende él mismo sobre la finalización de una tarea. Este prototipo construido, lejos del producto final, podrá ser testeable por el cliente y así proponer cambios.

Dentro de los artefactos, se suelen incluir también los gráficos de análisis de SCRUM, como *Burn-down Chart* o *Burn-up Chart*. Estos se podrán realizar tanto sobre el *Product Backlog* como por el *Sprint Backlog*. Estos son indicadores de cómo se ajusta la planificación a la realidad en base a las tareas finalizadas.

Eventos de SCRUM

Los eventos son el conjunto de protocolos, reuniones y acontecimientos secuenciales que los equipos de SCRUM realizan de manera periódica. Se encuentran seis eventos principales, que se detallarán a continuación y se encuentran explicitados en la siguiente imagen.

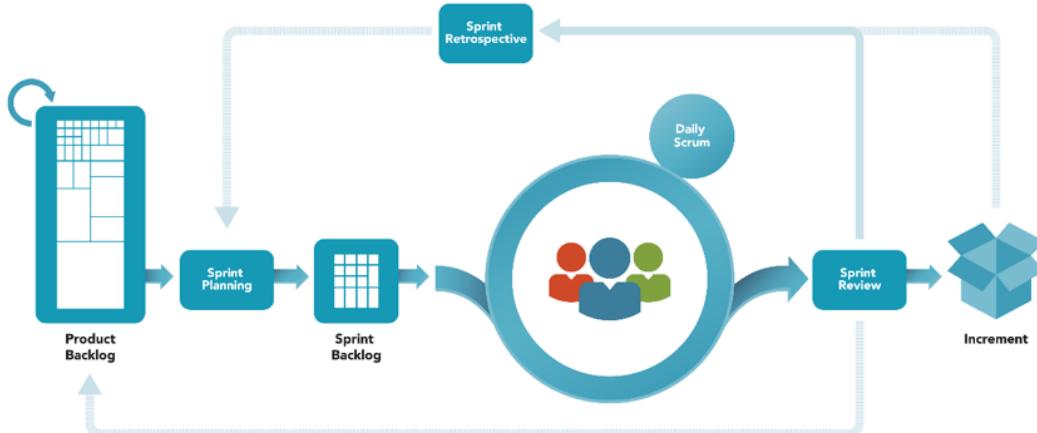


Figura 3.2: Eventos en SCRUM

El primero es la Organización del *Backlog*. Su función es dirigir el producto final hacia una visión comercial y tecnológica que busque el éxito del proyecto. El mantenimiento del *Product Backlog* se realiza con los comentarios de los usuarios y del equipo de desarrollo para priorizar los elementos de la misma. Gracias a esta tarea, cada elemento de la lista estará a punto para poder trabajar en él en cualquier momento.

El Sprint Planning es la reunión en la que el equipo de desarrollo planifica el trabajo que va a realizar durante el Sprint. En esta se establece un objetivo. En base a este se añaden tareas pertinentes del *Backlog* del producto al *Sprint Backlog*. Al final de esta reunión, cada miembro debe de tener claro cuál va ser el incremento a realizar al final del sprint y la manera de lograrlo.

El Sprint es el periodo de tiempo real en el que equipo SCRUM trabaja para llevar a término el incremento. El intervalo de tiempo será elegido por el equipo; autores recomiendan entre una semana y un mes, dependiendo de la tipología del proyecto. Este será el punto de referencia para todo el marco de SCRUM.

El Daily Meeting es la reunión diaria en la que todos los miembros del equipo indican el estado de sus tareas e inquietudes que posean sobre la consecución de las mismas. La duración ha de ser breve, en torno a un cuarto de hora. Debido a este motivo, se insiste en pequeños detalles como la puntualidad al comenzar o que sea en pie.

El Sprint Review es la reunión al final del sprint. En esta, el equipo comprueba el incremento realizado y lo muestra a las partes interesadas con el objetivo que obtener feedback sobre las tareas que ha ido realizando. La duración variará dependiendo del número de semanas del sprint. Se recomiendan entre una y cuatro horas.

Por último, el Sprint Retrospective. Es la reunión del equipo y para el equipo, donde se indican aquellos aspectos que han funcionado o fallado del sprint. Este se enfoca en todos los aspectos que rodean al grupo, desde el desarrollo hasta las personas. Su duración variará dependiendo de la duración del sprint, pero suele estar entre una y tres horas.

Funciones en SCRUM

El equipo de SCRUM debe poseer entre sus miembros tres funciones principales: *Product Owner*, *Scrum Master* y Equipo de desarrollo. Además, como los equipos ágiles suelen ser multidisciplinares, estos se podrán estructurar a su vez en otros grupos, de manera transversal. Los tres roles SCRUM se concretarán a continuación.



Figura 3.3: Equipo en SCRUM

El Product Owner es la persona que con mayor profundidad conoce el producto. Su misión es entender los requisitos del mercado, de los clientes y de la empresa. Estos se encargan de gestionar el *Product Backlog*, deciden si se lanzan los incrementos y lideran la mayoría de las reuniones. Este no ha de ser el Jefe de Proyecto, ya que ambos poseen misiones diferentes, que podrían perderse en la unión.

El SCRUM Master es la persona con mayor conocimiento de la metodología en el equipo. Su misión es proporcionar formación al equipo, buscar formas de afinar en su práctica y asegurar su buen funcionamiento. Este posee un nivel de detalle más amplio sobre el trabajo que cualquier miembro del equipo. Por ello, puede ayudar al grupo a mejorar su transparencia y flujo de entrega.

Por último, se encuentra el Equipo de desarrollo. Son los que realizan el trabajo de construcción en el día a día. Poseen el conocimiento de las prácticas a seguir, todas orientadas a un desarrollo sostenible. Estos han de tener una relación cercana, se deben encontrar en el mismo lugar y han de ser pocos. Autores expertos recomiendan entre cinco y siete.

Cada uno de los miembros del equipo, posee diferentes cualidades. Es tarea de todos conocerlas y optimizarlas, evitando cuellos de botellas. Es por ello importante sus funciones de auto-organización y trabajo en equipo. Gracias a este doble aspecto, estiman la duración de sus tareas en base al histórico de sus tareas realizadas.

3.1.2. Kanban

Kanban, al igual que SCRUM, es un marco de trabajo dentro de la metodología *agile*. Este marco pone su foco en un tablero donde aparecen las tareas que está realizando el equipo y el estado de las mismas. Esto da, de un simple vistazo, gran parte de la información del estado del proyecto. Además, debido a su simplicidad, puede ser aplicable a cualquier tipo de industria.

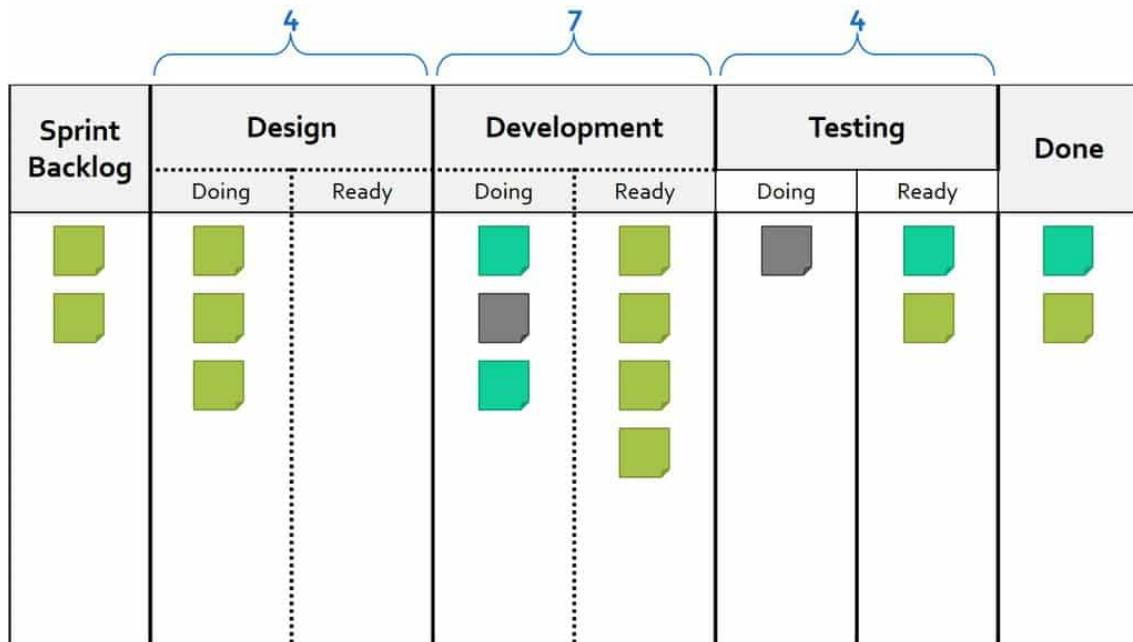


Figura 3.4: Ejemplo de tablero Kanban

Gracias a este doble aspecto de simplicidad y diversidad, es enormemente usado dentro del marco de SCRUM. Simplemente añade algunas consideraciones al uso del tablero. En concreto, se especifican cuatro acciones y seis prácticas, todas orientadas a realizar las tareas pendientes de manera eficaz.

Principios en Kanban

David Anderson es considerado como el líder del pensamiento Lean y Kanban. Este formuló los cuatro principios para llevar a cabo, de manera correcta y eficaz, el proceso evolutivo e incremental.

El primer principio: empezar con lo que se hace ahora. Como se ha dicho, Kanban no requiere ningún tipo de configuración, pudiendo ser aplicado sobre cualquier flujo de trabajo. Así, para empezar a usarlo, no es necesario realizar cambios radicales.

El segundo principio: comprometerse a buscar e implementar cambios incrementales y evolutivos. Este marco de trabajo está diseñado para enfrentarse a cualquier resistencia al cambio. Todo está orientado a cambios continuos, evolutivos e incrementales del proceso actual. Se evitan cambios drásticos en aras a pequeños virajes en el rumbo.

El tercer principio: respetar los procesos, las responsabilidades y los cargos actuales. Kanban reconoce que los anteriores roles y procedimientos pueden tener valor en el entorno del proyecto, por eso quiere conservarlos. Como se ve, se incita al cambio, aunque no se prescribe de manera alocada.

El cuarto principio: animar el liderazgo en todos los niveles. Recuerda a cada miembro del equipo que la consecución de los fines no es exclusiva a las capas directivas. Es por ello que promueve pequeños actos día a día de mejora continua en todos los aspectos del equipo. Este se pondrá en valor cuando se consiga un rendimiento óptimo.

Prácticas en Kanban

Una vez visto los puntos claves de la mentalidad de Kanban, se han de realizar las tareas pertinentes para implementar con éxito este marco de trabajo. En concreto, David Anderson indica seis prácticas para conseguir dicho fin.

La primera práctica: visualizar el flujo de trabajo. Se han de definir dos aspectos importantes. El primero es el estado por el que pasan las tareas, desde que se definen hasta que finalizan. El segundo es la información que poseerá cada tarea en el tablero.

La segunda práctica: eliminar las interrupciones. Uno de los perjuicios más importantes que se intentan solventar con Kanban es el continuo cambio de enfoque en los miembros del equipo. Así, en cada columna del tablero, se indicará un número máximo de tareas que pueden estar a la vez. Este es el *Work In Progress* (WIP). Su finalidad: solo se puede empezar con una tarea cuando alguna pendiente se termine.

La tercera práctica: gestionar el flujo. La idea de seguir Kanban no es otra que crear un proceso continuo e ininterrumpido de las tareas. Cada uno de los estadios del flujo son las columnas por las que pasan las tareas. Teniendo esto claro, habrá que analizar la velocidad y continuidad del movimiento.

La cuarta práctica: hacer las políticas explícitas. Hay una máxima en el mundo Lean: no puede mejorar algo que no se entiende. Así, todo proceso dentro del marco deberá estar definido, publicado y promovido.

La quinta práctica: circuitos de retroalimentación. Si se quiere que el cambio tenga éxito y sea duradero, se han de tener en cuenta las reuniones regulares para la transferencia de conocimiento. Ejemplos de estas son las reuniones diarias para ver el estado de las tareas o la reunión de revisión de entregas.

La sexta práctica: mejorar colaborando. Finalmente, si se quiere que este cambio exceda a la jurisdicción propia del proyecto que se está ejecutando y percute en la empresa, es necesario compartir ideas entre los miembros del equipo. En concreto, se ha de buscar una visión conjunta de las teorías sobre la manera de trabajar, el flujo de las tareas o procesos internos.

3.1.3. Aplicación

Una vez analizadas las metodología de trabajo, se indicarán cómo se han llevado a la práctica durante la ejecución y estudio del Trabajo Fin de Máster. Se ha utilizado un derivado entre SCRUM y Kanban, haciendo especial hincapié en el primero.

Parte importante de SCRUM es el equipo, gracias al cual se optimizan flexibilidad, creatividad y productividad. Normalmente, se trabajan con los tres roles anteriormente mencionados. Dado que solo hay un miembro que lleva a cabo todo el proceso, *Product Owner*, *Scrum Master* y *Development Team* se unen en una persona.

De la misma manera que en la definición del equipo, ha habido modificaciones en los eventos de SCRUM. La primera es el Sprint, que en nuestro caso son dos semanas. Estas son dos semanas en tiempo acumulado, no real. Esto se debe a que la dedicación al TFM ha sido fluctuante a lo largo del tiempo, además de centradas en fines de semana y periodos vacacionales.

Las reuniones, como se puede ya deducir, son de poco tiempo, aunque con pautas bien establecidas, como se puede ver a continuación.

- *Daily Scrum*: antes de ponerse a trabajar, se establece en el Tablero Kanban las tareas en estados *Ready* e *In Progress*.
- *Sprint Review*: se tiene con el profesor-tutor del proyecto de manera formal y con un tiempo establecido (sobre una hora), y con los potenciales clientes, de manera informal y sin tiempo previamente establecido, aunque nunca más de media hora.
- *Sprint Retrospective*: se sacan las gráficas para ver resultados, se apuntan las cosas a mejorar y se toman decisiones concretas, como, por ejemplo, mejorar una tarea, cambiar una funcionalidad fruto del feedback recibido. Su duración es de un cuarto de hora y se une con la siguiente.
- *Sprint Planning Meeting*: se hacen inmediatamente después de la anterior reunión estableciendo en el Tablero los requisitos a cumplir, así como lo que haya que mejorar del anterior Sprint. Igual que anteriormente, su duración no excede de los quince minutos.

Como se ha dicho anteriormente, el punto neurálgico de SCRUM en este proyecto es el Tablero Kanban. Este es la referencia visual del estado de las tareas y, por ende, del proyecto. Es usado por los interesados del proyecto y, por ello, se basa en su buen uso el éxito del proyecto, en cuanto a comunicación y seguimiento se refiere.

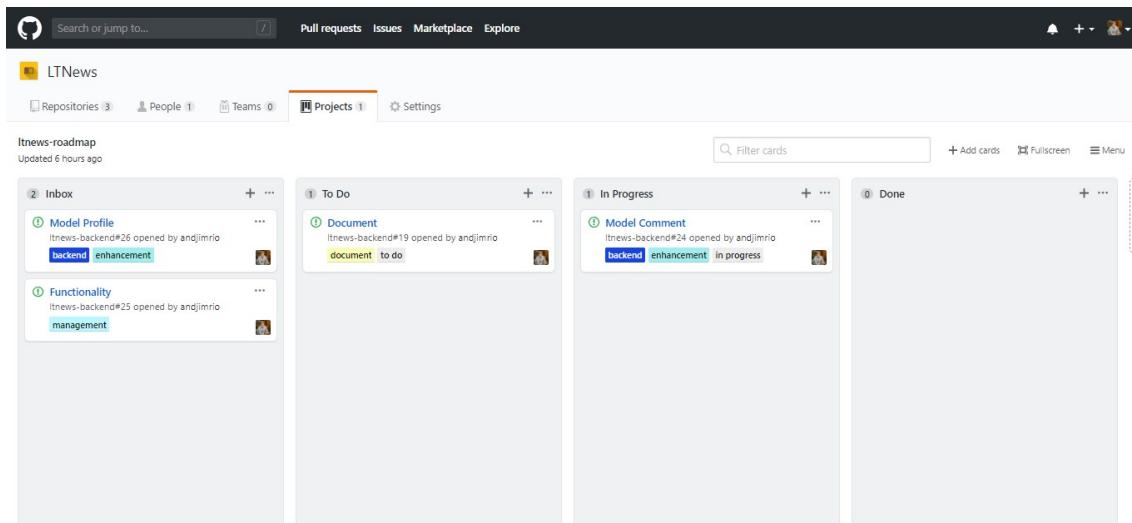


Figura 3.5: Tablero Kanban usado

El Tablero se usa para saber en qué se está trabajando y qué falta por hacer. En este se establecen las tareas a hacer, así como una aproximación temporal a cada una. El tiempo estimado se mide en una escala tomada según la secuencia de Fibonacci, donde 1 representa una tarea muy sencilla de una media hora de duración y 20 (nuestro máximo), una tarea excesivamente larga, de unas diez horas. Se evitará siempre tener tareas de más de 8 puntos, aunque, en algunos casos, no es viable.

En el Tablero usado se poseen los siguientes flujos, en cuanto a las columnas se refiere.

- *Backlog*: al principio del Sprint, aquí aparecen todas las tareas a realizar, así como aquellas que se dejen para futuros Sprints.
- *Ready*: aparecen las tareas del *Backlog* que hay que completar en dicha semana, es decir, en la mitad del sprint.
- *In progress*: son las tareas que se están realizando. Como limitación, el tiempo estimado de todas las tareas que aparecen en esta columna, no podrá superar los 8 puntos, si hay más de una tarea.
- *Done*: son las tareas hechas, esto es, desarrolladas y probadas.

Para el desarrollo de esta doble metodología, se ha hecho uso de la plataforma GitHub, que ofrece a desarrolladores un tablero kanban con las características previamente mencionadas. Las tareas serán las *issues* de todos los repositorios anejos al mismo. En concreto, las tareas de los repositorios que vuelcan a este tablero son los códigos de Back-End, Front-End y documentación asociada.

3.2– Planificación temporal

En este apartado se encuentra tanto la planificación en tiempo como la duración real, una vez finalizado el desarrollo. Como se ha dicho, la planificación utilizada ha sido utilizando la metodología ágil.

A continuación, se recoge la planificación por sprints con sus fechas de inicio y fin, así como su duración real en el tiempo y una variación del error.

ID	Nombre	Fecha inicio	Fecha fin	Semanas
S0	Fase previa	08/04/2018	06/06/2018	8
S1	Investigación	06/06/2018	23/09/2018	16
S2	Funcionalidad completa	23/09/2018	31/03/2019	27
S3	Virtualización de los servicios	31/03/2019	05/05/2019	5
S4	Capa Inteligencia Artificial	05/05/2019	26/06/2019	7

Cuadro 3.1: Planificación de Sprints: duración

ID	Nombre	Estimación	Duración	Variación
S0	Fase previa	70:00:00	66:57:53	4,34 %
S1	Investigación	30:00:00	60:26:29	-101,47 %
S2	Funcionalidad completa	100:00:00	70:26:20	29,56 %
S3	Virtualización de los servicios	30:00:00	46:58:15	-56,57 %
S4	Capa Inteligencia Artificial	70:00:00	55:57:54	20,05 %
300:00:00		300:46:55		-0,26 %

Cuadro 3.2: Planificación de Sprints: horas

Como se puede apreciar, ha habido un desajuste global en los sprints. Esto se ha debido a numerosas consideraciones. En primer lugar, se estimaba un fase de investigación de productos y tecnologías menor. Sin embargo, debido a la cantidad de estas y a su desconocimiento, ha implicado mayor tiempo. Otro punto a considerar ha sido la virtualización de los servicios con Docker: se estimaba un menor tiempo, no siendo así finalmente, por la complejidad de un sistema Full-Stack.

En general, estos dos acontecimientos han cambiado el rumbo del proyecto y de algunas de sus prioridades: se estimaba tener en la mitad de tiempo el Producto Mínimo viable para centrarse en la capa de Inteligencia Artificial, no siendo así finalmente.

A su vez, los sprints se han de desglosar en fases concretas en cuanto al desarrollo del producto se refiere. Por ello, la planificación más coherente con respecto a cualquier producto software es la siguiente.

Fase	Subfase	Estimación	Duración	Variación
1. Planificación		10:00:00	7:33:31	24,41 %
2. Estudio		30:00:00	50:00:00	-66,67 %
	2.1. Estudio de LEAN	10:00:00	17:00:00	-70,00 %
	2.2. Realización de modelos	5:00:00	8:00:00	-60,00 %
	2.3. Entrevistas	10:00:00	15:00:00	-50,00 %
	2.4. Estudio de los datos	5:00:00	10:00:00	-100,00 %
3. Análisis		50:00:00	56:50:13	-13,67 %
	3.1. Documentación	40:00:00	51:50:13	-29,59 %
	3.2. Elicitación de requisitos	5:00:00	3:00:00	40,00 %
	3.3. Modelo conceptual	5:00:00	2:00:00	60,00 %
4. Diseño		20:00:00	10:00:00	50,00 %
	4.1. Detalles técnicos	10:00:00	5:00:00	50,00 %
	4.2. Modelo de datos	10:00:00	5:00:00	50,00 %
5. Formación		30:00:00	23:32:03	21,55 %
	5.1. Django Rest Framework	5:00:00	8:14:13	-64,74 %
	5.2. VueJS	10:00:00	5:53:01	41,16 %
	5.3. ElasticSearch	10:00:00	2:21:12	76,47 %
	5.4. Docker	5:00:00	7:03:37	-41,21 %
6. Implementación		100:00:00	109:59:4	-9,99 %
	6.1. Back-End	25:00:00	34:44:07	-38,94 %
	6.2. Front-End	50:00:00	57:53:31	-15,78 %
	6.3. AI-Layer	25:00:00	17:22:03	30,53 %
7. Pruebas		25:00:00	5:47:21	76,84 %
8. Despliegue		25:00:00	27:04:01	-8,27 %
9. Presentación		10:00:00	10:00:00	0,00 %
TOTAL		300:00:00	300:46:51	-0,26 %

Cuadro 3.3: Planificación temporal

A nivel de tareas se puede apreciar con mayor precisión lo anterior. En concreto, se ha excedido en tiempo el estudio de la idea bajo el marco LEAN, y el desarrollo de las capas de Frontend y Backend, dejando sin recursos temporales a la capa de IA.

3.3– Roles del proyecto

Tal y como dijimos antes, en nuestro proyecto se unifican los roles de SCRUM en una única persona. Sin embargo, esto no quita que este perfil tenga que asumir a lo largo del proyecto una serie de roles, en función del tipo de tareas a realizar. Sin embargo, para simplificar dichos roles, se han unificado en la responsabilidad de Jefe de Proyecto.

Por ello, las responsabilidad adquiridas y sus misiones serán las que aparecen a continuación.

- Jefe de proyecto: encargado de la gestión y planificación del proyecto, de reunirse con el cliente y de tomar decisiones sobre el equipo.
- Analista: encargado de recolectar los requisitos y necesidades del proyecto, así como la de diseñar el sistema de información.
- Desarrollador: se encarga de implementar los requisitos desarrollando para ello el software.

- Tester: su misión es la asegurar la calidad de los productos obtenidos, en concreto, se encarga de elaborar y ejecutar las pruebas.

3.4– Planificación económica

Para la planificación económica se han considerado los gastos asociados a la ejecución del proyecto, sin ingresos relacionados directamente. Aunque el proyecto buscará en un futuro tener beneficios, no entra este aspecto dentro del ámbito del Trabajo Fin de Máster.

Habrá que considerar, por tanto, dos tipos de costes, dependiendo si hacen referencia a las personas físicas, o a materiales e infraestructuras necesarias para llevar a cabo el proyecto.

3.4.1. Costes directos

Dentro de este tipo de costes se consideran exclusivamente los costes de personal. Tal y como se ha dicho en el apartado 3.3, para este es necesario una serie de roles, cada uno de los cuales posee un salario y dedicación diferentes.

Así, dentro de estos costes, se van a tener en cuenta los sueldos de los distintos tipos de trabajadores durante el desarrollo del proyecto. Para conseguir esto, se utilizará el Boletín Oficial del Estado publicado en abril de 2009, referente a salarios mínimos, como se puede apreciar en el Trabajo e Inmigración (2009). En el caso actual, conlleva el uso y búsqueda de los datos del sector de las TIC.

Para todos los casos, se considerará que un año posee 250 días hábiles, que suponen 2.000 horas. Así, analizando cada rol en concreto, se obtienen los siguientes datos.

- Jefe de Proyecto: su sueldo mínimo es de 36.600€ que a la hora hace 18,3€
- Analista: su sueldo mínimo es de 21.555,66€ que a la hora hace 10,78€
- Desarrollador: su sueldo mínimo es de 15.442,56€ que a la hora hace 7,72€
- Tester: su sueldo mínimo es de 11.773,02€ que a la hora hace 5,89€

Además, a estos costes, se han de aplicar los impuestos que paga la empresa para cualquier trabajador en España. Todos son porcentajes que se aplican a la base de cotización. Estos son los siguientes:

- Contingencias comunes
 - Estos dan cobertura a bajas temporales o prestaciones de jubilación, entre otras.
 - Supone el 23,6 %
- Fondo de Garantía Social
 - Es un organismo encargado de dar garantía a los trabajadores de su salario o indemnizaciones.
 - Supone el 0,2 %
- Formación Profesional
 - Asegura la formación al trabajador.
 - Supone el 0,7 %
- Seguridad Social

- Dependerá del tipo de contrato y de su duración. Consideramos duración determinada y a tiempo parcial.
- Supone 7,7 %

Teniendo en cuenta todos los porcentajes, queda para aplicar a la base de cotización una suma de los anteriores porcentajes: 32,2 %.

3.4.2. Costes indirectos

En este tipo de costes, se incluyen los materiales utilizados para realizar el proyecto, que en este caso se dividen entre hardware y software. Sobre cada material, independientemente de su tipología, se aplicará una base de amortización. Esto es así ya que no se adquiere un material concreto exclusivamente para un proyecto, sino que su coste deberá aparecer reflejado en los distintos proyectos en los que se utilice hasta que termine su vida útil.

Hardware

Dentro de este apartado se consideran dos dispositivos: un ordenador portátil y un móvil. Ambos han servido para desarrollar el proyecto y para sus respectivas pruebas. El ordenador utilizado ha sido un *ASUS A55L*, valorado en 789,99€. Se considera que la vida útil de cualquier portátil es de cuatro años. Así, cada mes utilizándolo, supondrá uno gasto de 16,46€. Este ha sido utilizado durante todo el proyecto. El smartphone que se utilizará para comprobar la interfaz gráfica será un *Xiaomi Mi A2 Lite*, valorado en 189,99€. Considerando en este caso una vida útil de dos años, supondrá un gasto del mes utilizado por 7,92€. Este ha sido utilizado durante la mitad del proyecto: los meses finales para visualizar la interfaz gráfica móvil.

Software

Dentro de este, se engloban las licencias de los programas utilizados, así como los costes del servidor. Las licencias, como el hardware, requieren también de una amortización. Así, las utilizadas para llevar a cabo el proyecto y su coste han sido los siguientes:

- Windows 10 Home
 - 135€ en pago único, amortizable en 8 años.
 - 1,41€/mes.
- Office 365 Personal
 - Posee pago mensual.
 - 7,00€/mes.
- PyCharm
 - 199€ por año utilizado.
 - 16,58€/mes.
- WebStorm
 - Al igual que PyCharm, 199€ por año utilizado.
 - 16,58€/mes.

Para el servidor, hemos adoptado el plan más básico de DigitalOcean. Esto hace unos 5\$/mes, que en el cambio a euros son 4,45€/mes.

3.4.3. Presupuesto

Con el conocimiento de los gastos, se procederá a dar un presupuesto del proyecto. Antes, se consideran que es necesario un colchón para pagos no previstos. Este será de un 15 % del subtotal. Además de esto, hemos de tener en cuenta el I.V.A. Por ello, la cantidad resultante será la siguiente.

	Tiempo	Coste unitario	Coste total
Personal	horas	euros	3.755,47€
Jefe de proyecto	25	24,19	604,82€
Analista	100	14,25	1.425,12€
Desarrollador	150	10,21	1.530,88€
Tester	25	7,79	194,66€
Hardware	meses	euros	309,58€
ASUS A555L	14	16,46	230,41€
Xiaomi Mi A2 Lite	10	7,92	79,16€
Software	meses	euros	471,61€
Windows 10 Home	14	1,41	19,69€
Office 365 Professional	14	7,00	98,00€
PyCharm	10	16,58	165,83€
WebStorm	10	16,58	165,83€
Digital Ocean	5	4,45	22,25€
Subtotal		4.536,65€	
Contingencias		680,50€	
Total		5.217,15€	

Cuadro 3.4: Presupuesto económico

Por tanto, el coste total del proyecto asciende a 5.217,15€, contando todos costes y un margen de contingencia.

CAPÍTULO 4

Análisis

4.1– Modelo conceptual

El modelo conceptual consiste en las entidades principales del sistema y sus relaciones. Por ello, se explicará a partir de la imagen 4.1, que podrá verse a continuación.

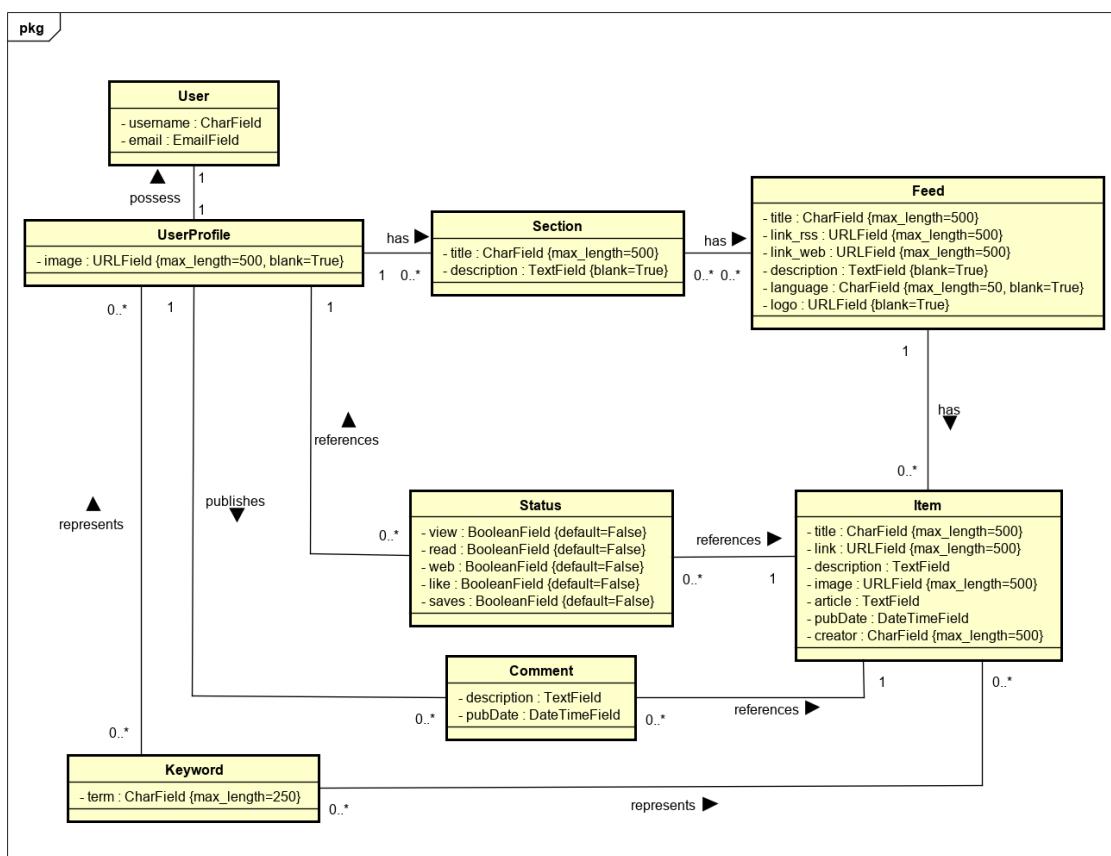


Figura 4.1: Diagrama de clases

Los elementos centrales del sistema son *Profile* e *Item*. Esto se puede apreciar a simple vista, debido a la cantidad de las relaciones con las diferentes entidades.

En primer lugar, *Profile* gestiona más relaciones que atributos posee. Esto es debido a que

la lógica y datos del usuario se encontrarán en *User*. Así, entre ellos habrá una relación directa y bidireccional, en concreto, una relación *One To One*. El perfil de un usuario, así, poseerá secciones asociadas, tendrá estados a noticias, publicará comentarios en noticias y estará representado por temas que le interesa. Mientras, los datos estarán propiamente en dicho usuario. Aunque esto será transparente al mismo, el sistema diferenciará esta lógica.

En segundo lugar, como punto neurálgico de la aplicación, se encuentran las noticias, *Item* en el modelo. El usuario, por lo comentado anteriormente, posee cuatro relaciones indirectas hacia este, y que representarán las diferentes funcionalidades de la aplicación.

- *Profile-Section-Feed-Item*. Este camino quiere representar la condición natural de noticia con respecto al usuario. Las noticias están agrupadas en feeds de sindicación de contenido. Estos, a su vez, asociados a secciones. Por último, es el usuario el que crea dicha secciones. Con esto, se pueden agrupar las noticias en base a dos criterios: secciones o feeds asociados. Esta será la navegabilidad principal de un usuario para leer noticias.
- *Profile-Status-Item*. En segundo lugar, se guardará la interacción del usuario con dicha noticia. Con esto se consigue guardar los estados por los que pasa la noticia para el usuario y poder valorar cuánto gusta a un usuario una determinada noticia. Así, se mantiene guardado si un usuario ha visto ya una noticia, para así no volver a mostrársela; si la ha leído, le ha gustado o ha accedido al link original. Así sabremos que ese tipo de noticias le interesa o si la quiere guardar para leerla más tarde.
- *Profile-Comment-Item*. Se da también la posibilidad a los usuarios de comentar las noticias para poder dar su opinión. Por ello, es necesario mantener dicha entidad como clase de asociación.
- *Profile-Keyword-Item*. Por último lugar, el sistema analiza cada noticia y extrae las palabras claves de la misma. Con estas *keywords* se definen las noticias y es posible relacionarlas entre sí. Además, al usuario también se le asignan palabras claves, que definirán sus gustos. Es por ello que se usa esta entidad y su relación *Many To Many*, para albergar el fruto del análisis de ambas entidades.

4.2– Actores del sistema

El sistema posee dos actores, consistente en la diferenciación entre usuarios autenticados o anónimos. Los primeros son aquellos que han accedido a la aplicación, están registrados en el sistema y se han logado con sus credenciales. Los segundos son los que el sistema no puede identificar únicamente, ya que no están registrados, o si lo están, no se han autenticado.

La diferencia radicará en el acceso a contenido que requiera de poseer un usuario asociado. No obstante, se ha intentado, en la medida de lo posible dar el mayor acceso posible al usuario anónimo en la aplicación.

En resumen, los actores serán: **Anónimo** y **Usuario**.

4.3– Catálogo de requisitos

Dado que se ha usando una metodología ágil, como se indica en el punto 3.1, se representan los requisitos como historias de usuario, ya que suponen una forma clara y sencilla de representarlos. En este apartado se englobarán los requisitos funcionales del sistema, dentro de las historias épicas correspondientes, y los no funcionales.

4.3.1. Requisitos funcionales

Historias épicas

HE1 - Acceso aplicación	
Usuario	Anónimo
Descripción	El sistema deberá proveer el registro de cualquiera, así como el acceso a los usuarios ya registrados.

Cuadro 4.1: HE1 - Acceso aplicación

HE2 – Gestión de perfil	
Usuario	Usuario
Descripción	El sistema deberá almacenar toda la información perteneciente al usuario registrado, así como la posibilidad de que este la cambie.

Cuadro 4.2: HE2 – Gestión de perfil

HE3 – Gestión de secciones	
Usuario	Usuario
Descripción	El sistema deberá almacenar todas las secciones que el usuario cree, así como dejarle la posibilidad de que este la cambie. De igual manera con los feeds asociados a dicha sección.

Cuadro 4.3: HE3 – Gestión de secciones

HE4 – Gestión de noticias	
Usuario	Usuario
Descripción	El sistema deberá almacenar todas las noticias actualizadas de los feeds del usuario, así como dejarle interaccionar con ellas.

Cuadro 4.4: HE4 – Gestión de noticias

HE5 – Gestión de intereses	
Usuario	Usuario
Descripción	El sistema deberá almacenar todas las preferencias del usuario para recomendarle noticias.

Cuadro 4.5: HE5 – Gestión de intereses

Historias de usuario

HU01 – Registro	
Usuario	Anónimo
Historia épica	HE1 – Acceso Aplicación
Descripción	Como usuario anónimo no registrado previamente quiero poder registrarme en el sistema para acceder a él.

Cuadro 4.6: HU01 – Registro

HU02 – Acceso

Usuario	Anónimo
Historia épica	HE1 – Acceso Aplicación
Descripción	Como usuario anónimo registrado previamente quiero poder acceder al sistema para interactuar con la aplicación.

Cuadro 4.7: HU02 – Acceso

HU03 – Ver perfil

Usuario	Usuario
Historia épica	HE2 – Gestión de perfil
Descripción	Como usuario quiero poder ver la información que el sistema guarda de mi para saber qué datos posee la aplicación sobre mi identidad.

Cuadro 4.8: HU03 – Ver perfil

HU04 – Editar perfil

Usuario	Usuario
Historia épica	HE2 – Gestión de perfil
Descripción	Como usuario quiero poder editar la información que el sistema guarda de mi para que los datos sean más acordes a mi identidad.

Cuadro 4.9: HU04 – Editar perfil

HU05 – Ver secciones

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero ver las secciones que previamente he creado para saber qué secciones poseo actualmente.

Cuadro 4.10: HU05 – Ver secciones

HU06 – Editar secciones

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero editar las secciones que previamente he creado para modificarlas.

Cuadro 4.11: HU06 – Editar secciones

HU07 – Crear secciones

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero crear secciones para poder organizar mejor los feeds que leo.

Cuadro 4.12: HU07 – Crear secciones

HU08 – Eliminar secciones

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero eliminar secciones para poder quitar las secciones que ya no me sirven.

Cuadro 4.13: HU08 – Eliminar secciones

HU09 – Ver feeds por sección

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero ver los feeds asociados a cada sección para saber a qué periódicos sigo.

Cuadro 4.14: HU09 – Ver feeds por sección

HU10 – Añadir feed por sección

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero añadir feeds a cada sección para seguir a más periódicos.

Cuadro 4.15: HU10 – Añadir feed por sección

HU11 – Eliminar feed por sección

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero eliminar feeds asociados a una sección para quitar los periódicos que ya no me interesen.

Cuadro 4.16: HU11 – Eliminar feed por sección

HU12 – Ver noticias por sección

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero ver todas las noticias de los feeds asociados a cada sección para tener en una vista las últimas noticias de dichos medios.

Cuadro 4.17: HU12 – Ver noticias por sección

HU13 – Ver noticias por feed

Usuario	Usuario
Historia épica	HE3 – Gestión de secciones
Descripción	Como usuario quiero ver todas las noticias de un feed para tener en una vista las últimas noticias de dicho feed.

Cuadro 4.18: HU13 – Ver noticias por feed

HU14 – Ver noticias de actualidad	
Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero ver las noticias más actuales de todos los periódicos que sigo para tener en una vista las últimas noticias.

Cuadro 4.19: HU14 – Ver noticias de actualidad

HU15 – Ver una noticia	
Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero ver una noticia para no tener que ir a la página del periódico cada vez.

Cuadro 4.20: HU15 – Ver una noticia

HU16 – Ver noticias relacionadas	
Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero ver dentro de la vista de una noticia, otras relacionadas para saber qué dicen otros diarios de la misma o el mismo periódico, pero en otro día.

Cuadro 4.21: HU16 – Ver noticias relacionadas

HU17 – Ver comentarios de noticias	
Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero ver dentro de la vista de una noticia, los comentarios para saber qué dicen otros usuarios de la misma.

Cuadro 4.22: HU17 – Ver comentarios de noticias

HU18 – Comentar noticias	
Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero comentar noticias para expresar mi opinión de la misma.

Cuadro 4.23: HU18 – Comentar noticias

HU19 – Buscar noticias	
Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero buscar noticias para poder comparar qué se dice sobre un tema.

Cuadro 4.24: HU19 – Buscar noticias

HU20 – Marcar noticia como Me gusta

Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero marcar una noticia como <i>Me gusta</i> para transmitir a los demás usuarios que dicha noticia es acorde a mis gustos.

Cuadro 4.25: HU20 – Marcar noticia como *Me gusta*

HU21 – Ver palabras claves por noticia

Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero ver las palabras que definen a una noticia para saber de qué hablará sin necesidad de leerla.

Cuadro 4.26: HU21 – Ver palabras claves por noticia

HU22 – Ver noticias por palabras claves

Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero ver las noticias que poseen dicha palabra clave para saber qué se dice sobre un tema.

Cuadro 4.27: HU22 – Ver noticias por palabras claves

HU23 – Resumen de actualidad

Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero ver de manera resumida las noticias de la última hora, o último día, semana o mes, para saber qué dicen hoy los feeds que sigo sin necesidad de leer todas las noticias.

Cuadro 4.28: HU23 – Resumen de actualidad

HU24 – Guardar una noticia

Usuario	Usuario
Historia épica	HE4 – Gestión de noticias
Descripción	Como usuario quiero guardar una noticia para poder leerla más tarde.

Cuadro 4.29: HU24 – Guardar una noticia

HU25 – Ver intereses

Usuario	Usuario
Historia épica	HE5 – Gestión de intereses
Descripción	Como usuario quiero saber cuáles son mis intereses, según el sistema, en base a mis lecturas para conocerme más y buscar sobre los mismos.

Cuadro 4.30: HU25 – Ver intereses

HU26 – Ver noticias según mis intereses	
Usuario	Usuario
Historia épica	HE5 – Gestión de intereses
Descripción	Como usuario quiero que el sistema me muestre noticias recomendadas para mí, para saber qué se dice sobre lo que me interesa.

Cuadro 4.31: HU26 – Ver noticias según mis intereses

4.3.2. Requisitos no funcionales

RNF1 – Control de acceso	
Descripción	El sistema deberá restringir el acceso a determinadas partes solo a aquellos usuarios registrados.

Cuadro 4.32: RNF1 – Control de acceso

RNF2 – Recurrencia	
Descripción	El sistema deberá permitir, como mínimo, 15 usuarios simultáneos en el sistema sin afectar a la rapidez de cada uno.

Cuadro 4.33: RNF2 – Recurrencia

RNF3 – Sencillez de uso	
Descripción	El sistema deberá proporcionar una interfaz clara con fácil navegabilidad. En concreto, el usuario debe acceder a cualquier vista en menos de 5 clics o menos.

Cuadro 4.34: RNF3 – Sencillez de uso

RNF4 – Guía de ayuda	
Descripción	El sistema deberá proporcionar en una página todas las funcionalidades de la página explicadas.

Cuadro 4.35: RNF4 – Guía de ayuda

RNF5 – Barra de navegación	
Descripción	El sistema deberá proporcionar al usuario en cada vista un menú de navegación con las opciones más importantes.

Cuadro 4.36: RNF5 – Barra de navegación

RNF6 – Sesión iniciada	
Descripción	El sistema deberá guardar en una cookie la sesión actual. Así, aunque cierre el navegador, tendrá iniciada su sesión. Esta se quitará cuando el usuario cierre sesión.

Cuadro 4.37: RNF6 – Sesión iniciada

RNF7 – Interfaz responsiva

Descripción	El sistema deberá proporcionar una interfaz acorde al tamaño de pantalla del dispositivo, en concreto, aptos para móviles, tabletas, ordenadores portátiles y ordenadores de sobremesa.
--------------------	---

Cuadro 4.38: RNF7 – Interfaz responsiva

RNF8 – Traspaso de datos

Descripción	El sistema deberá evitar el traspaso de información de manera no controlada fuera de la página bajo cualquier circunstancia.
--------------------	--

Cuadro 4.39: RNF8 – Traspaso de datos

4.4– Diagrama de secuencia

Una vez definido el catálogo de requisitos, queda establecer cómo será dicha comunicación entre el actor y el sistema para cada requerimiento establecido. Esto se consigue de manera conceptual con los diagramas de secuencia.

Además, pensando en el futuro desarrollo, estos se agruparán en las diferentes entidades del sistema, como se ha visto en el punto 4.1. Para la comunicación, se usarán nombres descriptivos y en inglés, en un modo cercano al lenguaje máquina.

4.4.1. Perfil

En primer lugar, para que el usuario vea su perfil, éste ha de pedirlo, así el sistema se lo devolverá.

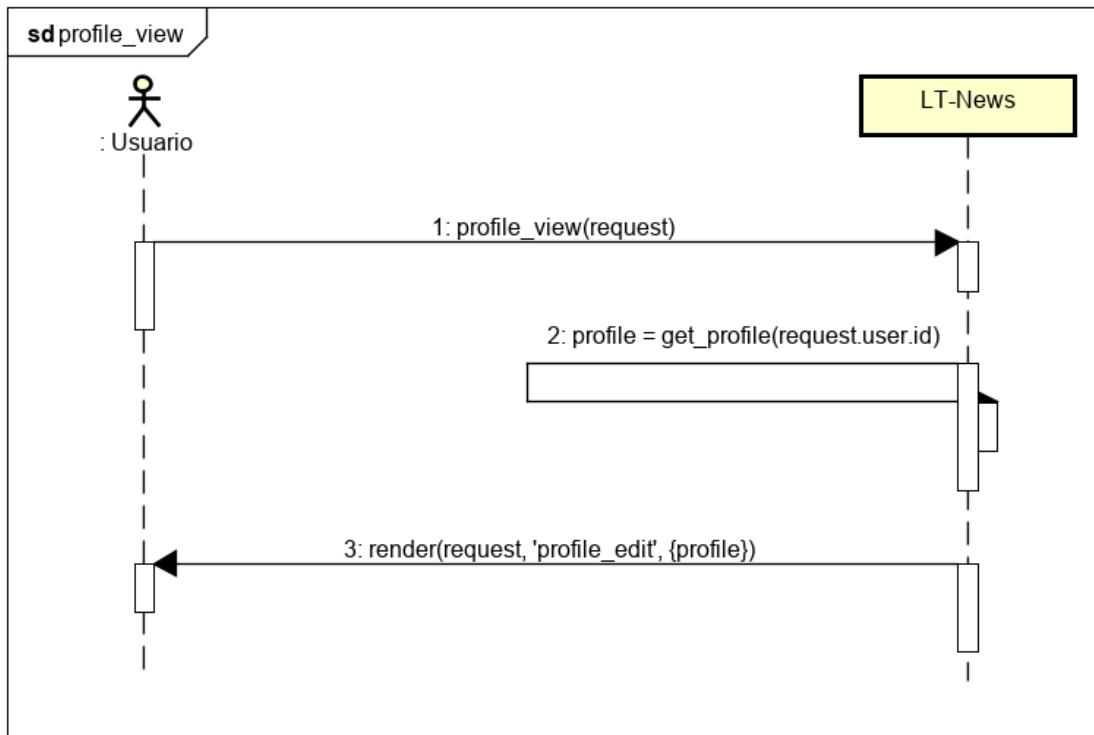


Figura 4.2: Diagrama de secuencia de profile_view

Si quiere modificar el perfil, el usuario debe pedirle al sistema un formulario base, donde estén sus datos para así, enviarlo una vez modificado y completo.

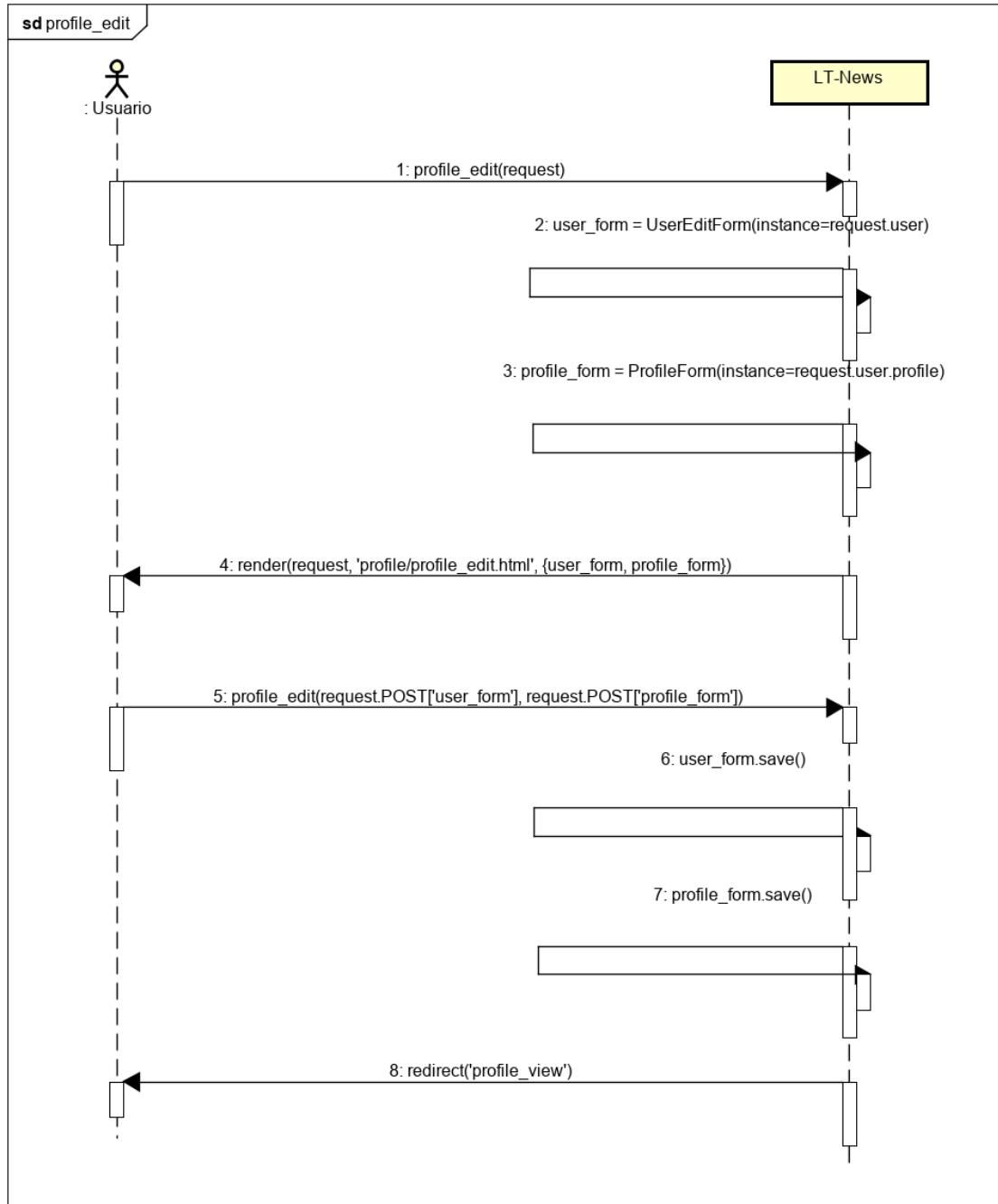


Figura 4.3: Diagrama de secuencia de profile_edit

4.4.2. Section

El sistema proporcionará al usuario una vista donde poder ver todas sus secciones.

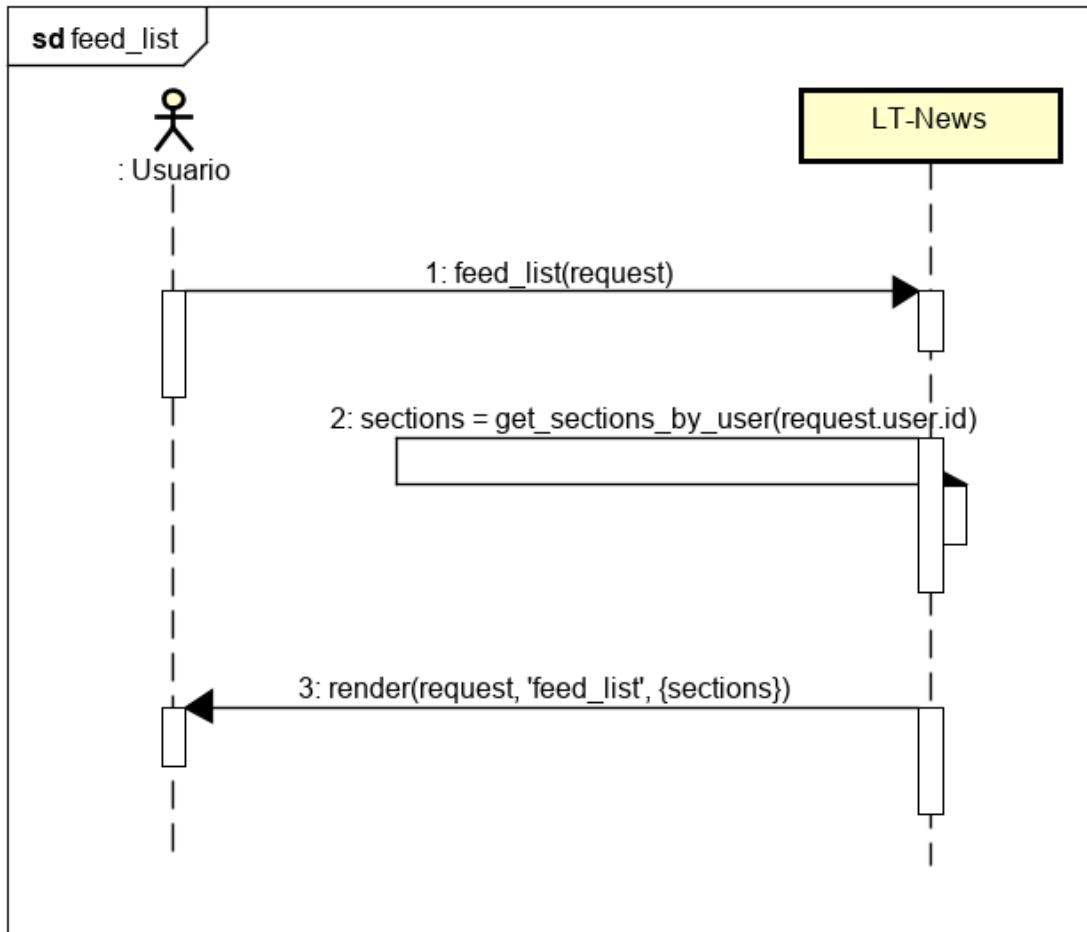


Figura 4.4: Diagrama de secuencia de feed_list

El usuario puede ver alguna de sus secciones en concreto, pidiéndolo expresamente al sistema.

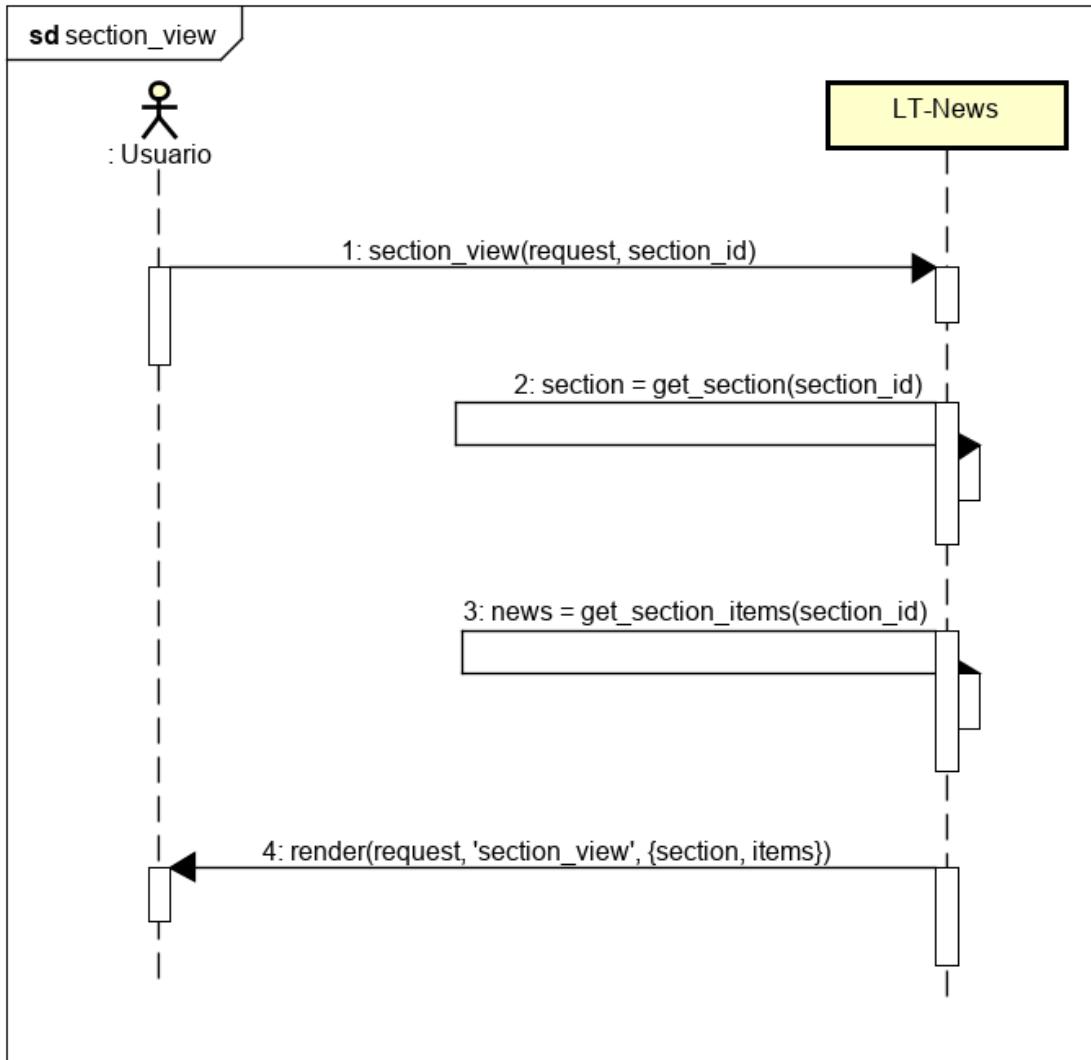


Figura 4.5: Diagrama de secuencia de section_view

Podrá eliminarlas también.

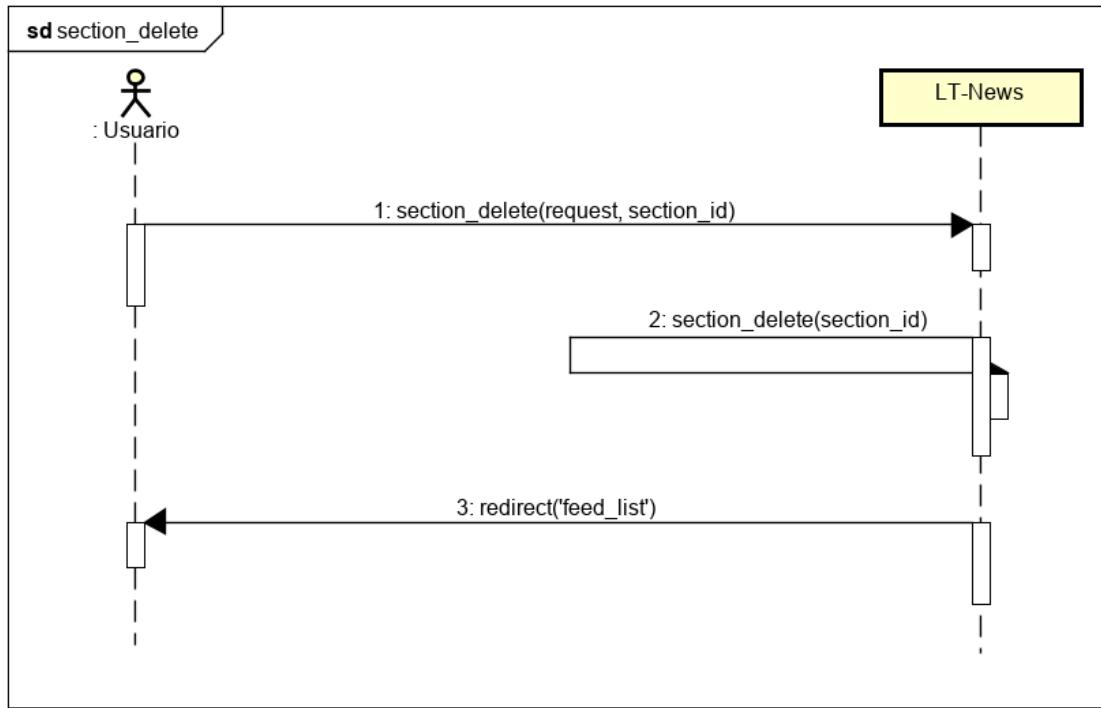


Figura 4.6: Diagrama de secuencia de section_delete

De la misma manera, podrá editarlas.

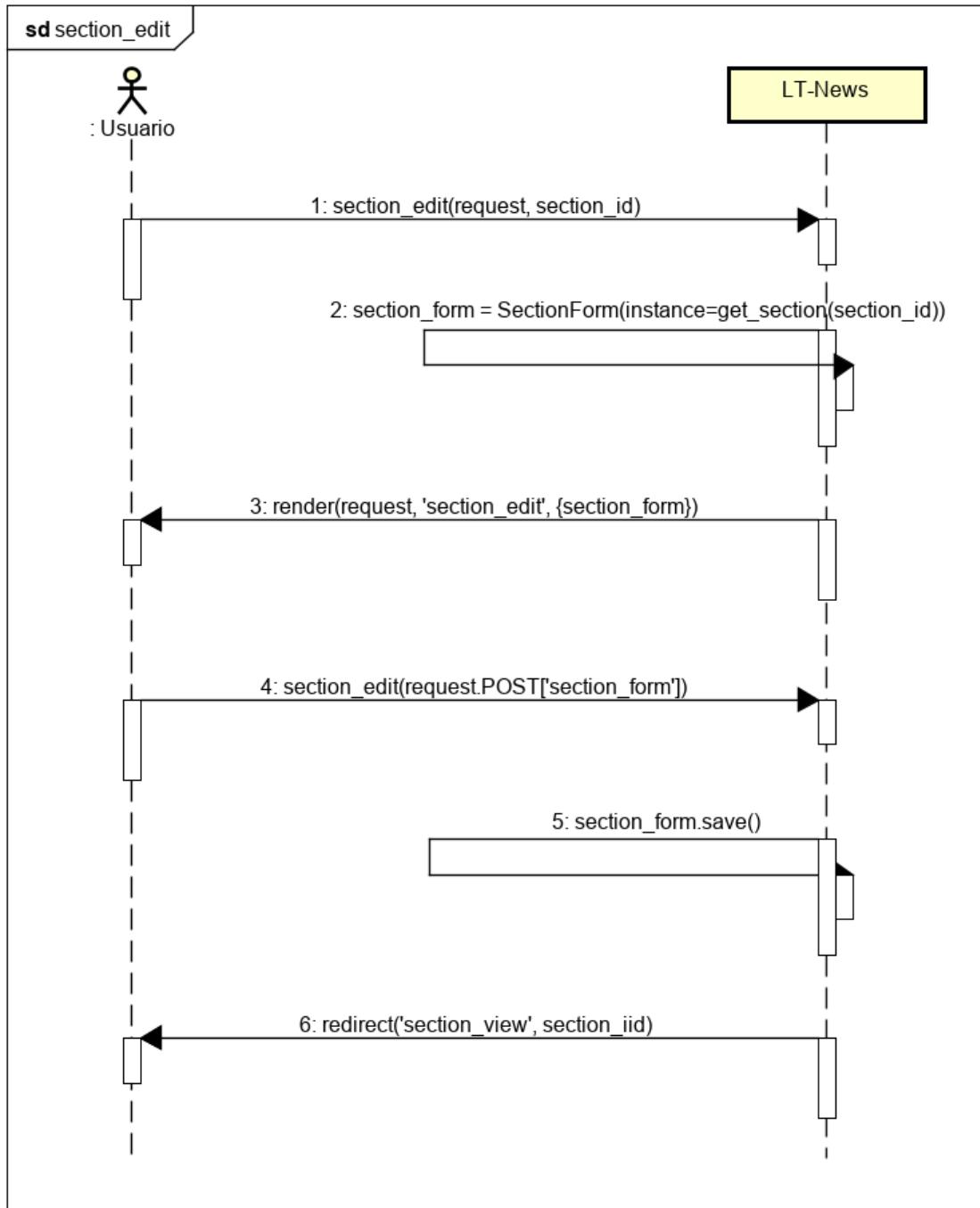


Figura 4.7: Diagrama de secuencia de section_edit

4.4.3. Feed

El usuario puede ver un feed en concreto o crearlo.

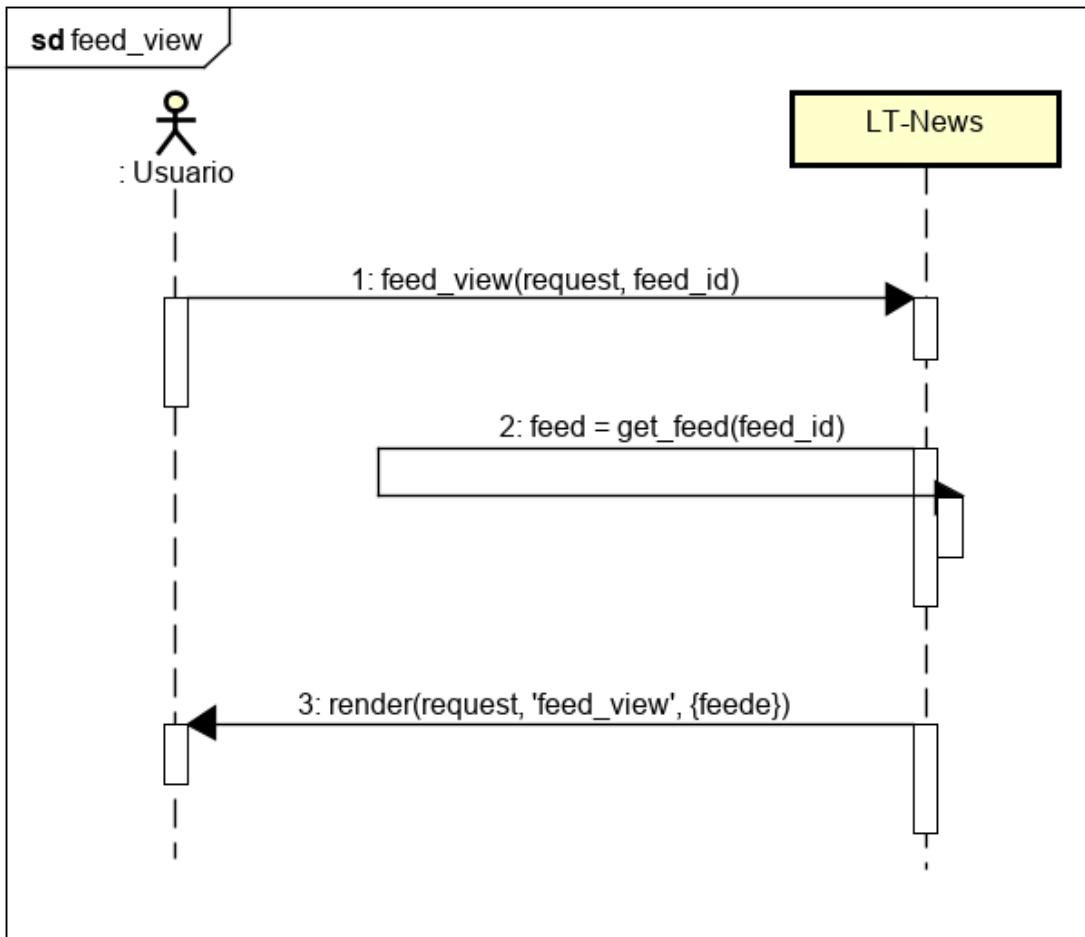


Figura 4.8: Diagrama de secuencia de feed_view

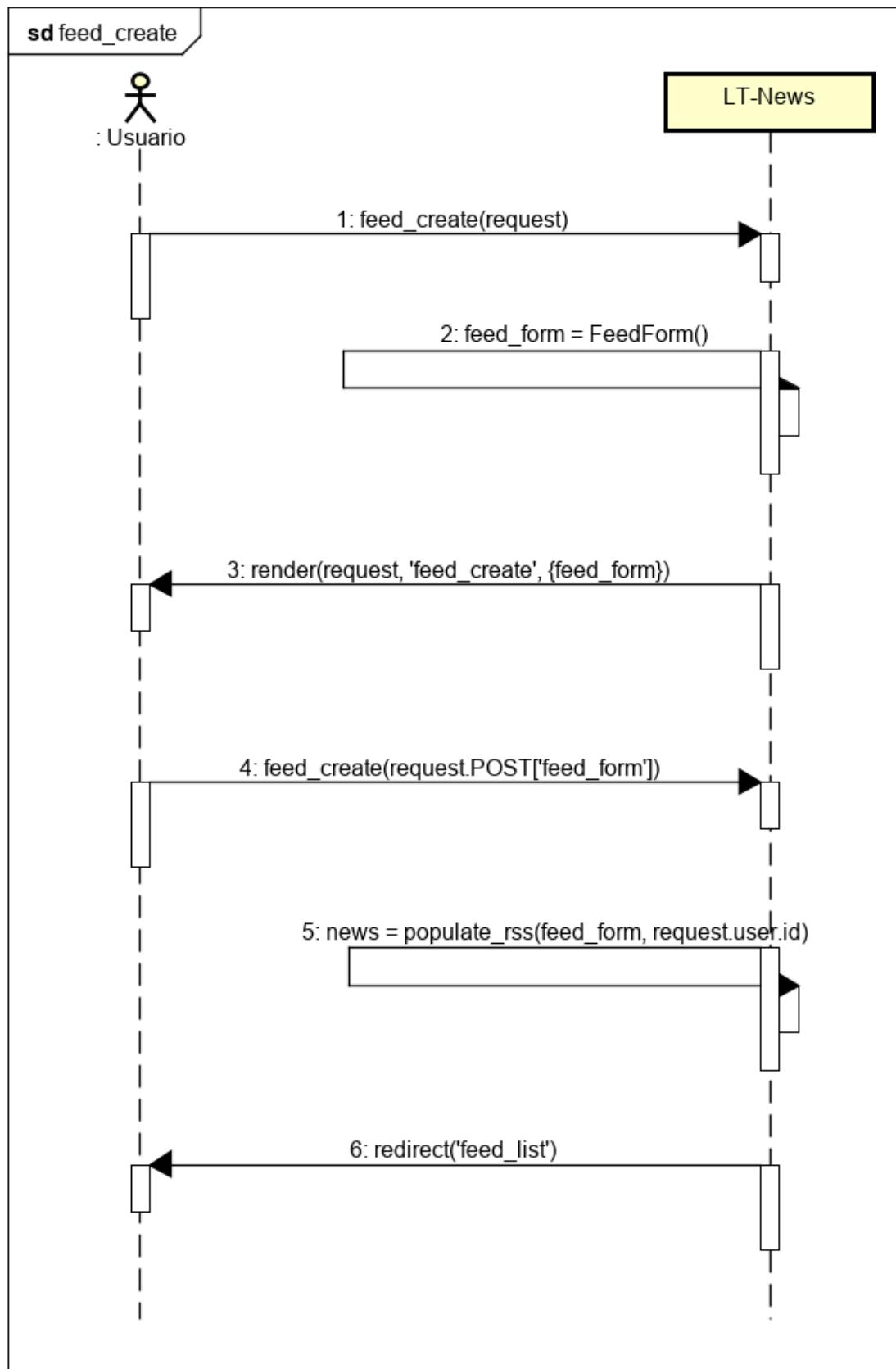


Figura 4.9: Diagrama de secuencia de feed_create

De la misma manera, eliminarlo.

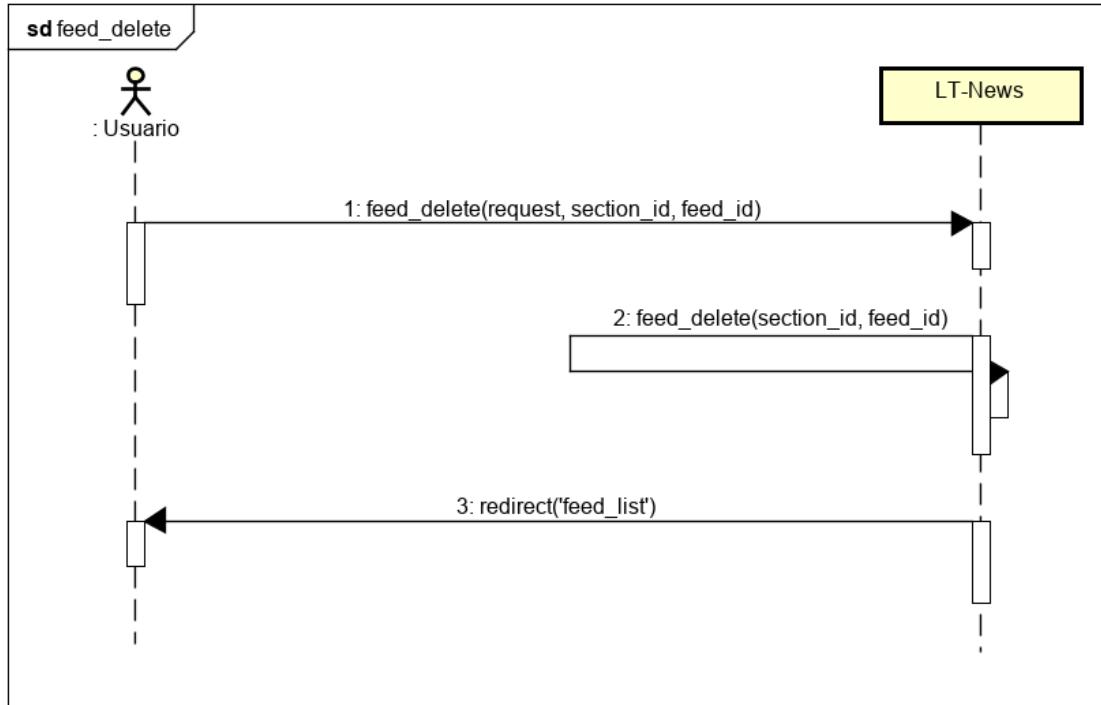


Figura 4.10: Diagrama de secuencia de feed_delete

4.4.4. Item

El usuario puede pedirle al sistema que le de las últimas noticias o ver una noticia en concreto.

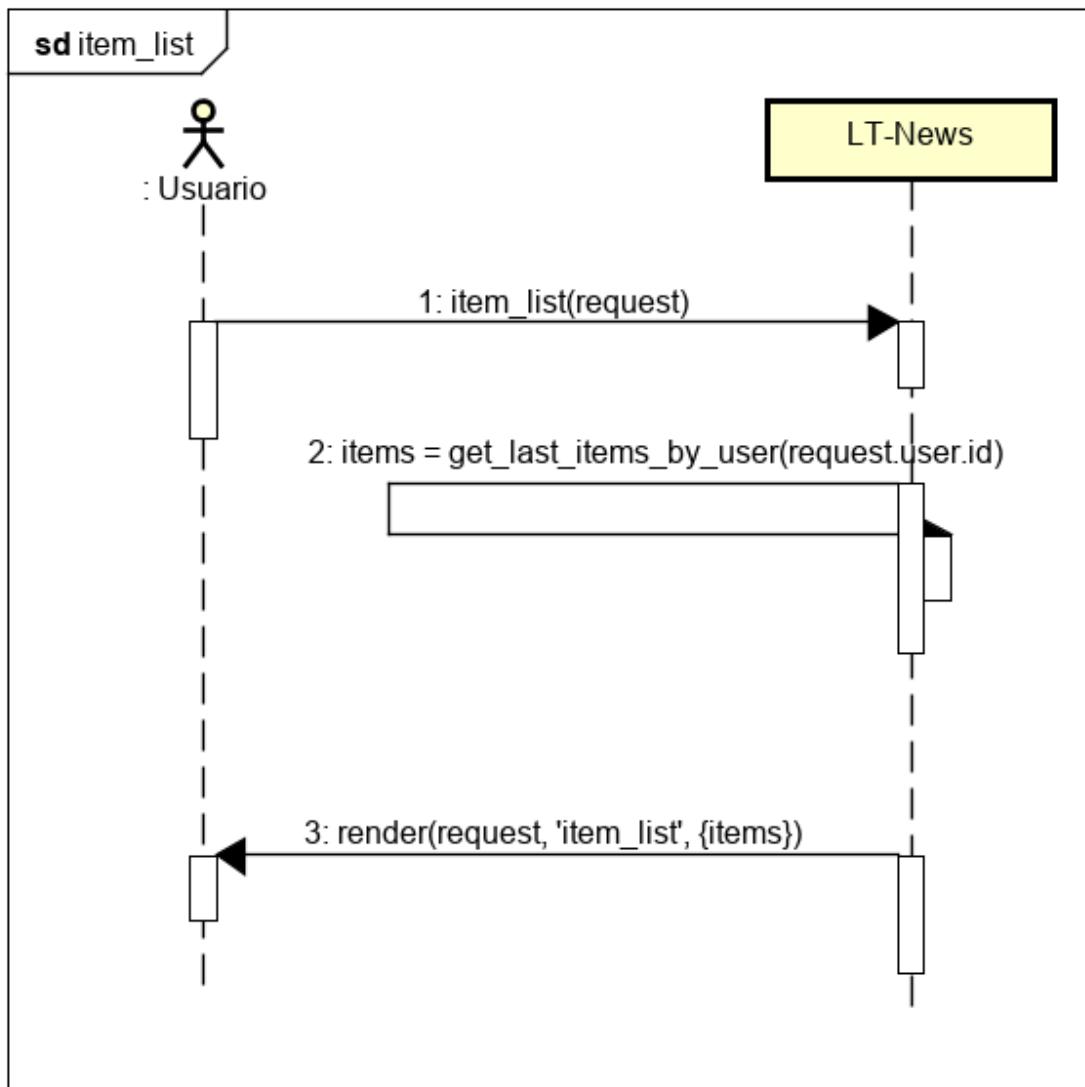


Figura 4.11: Diagrama de secuencia de item_list

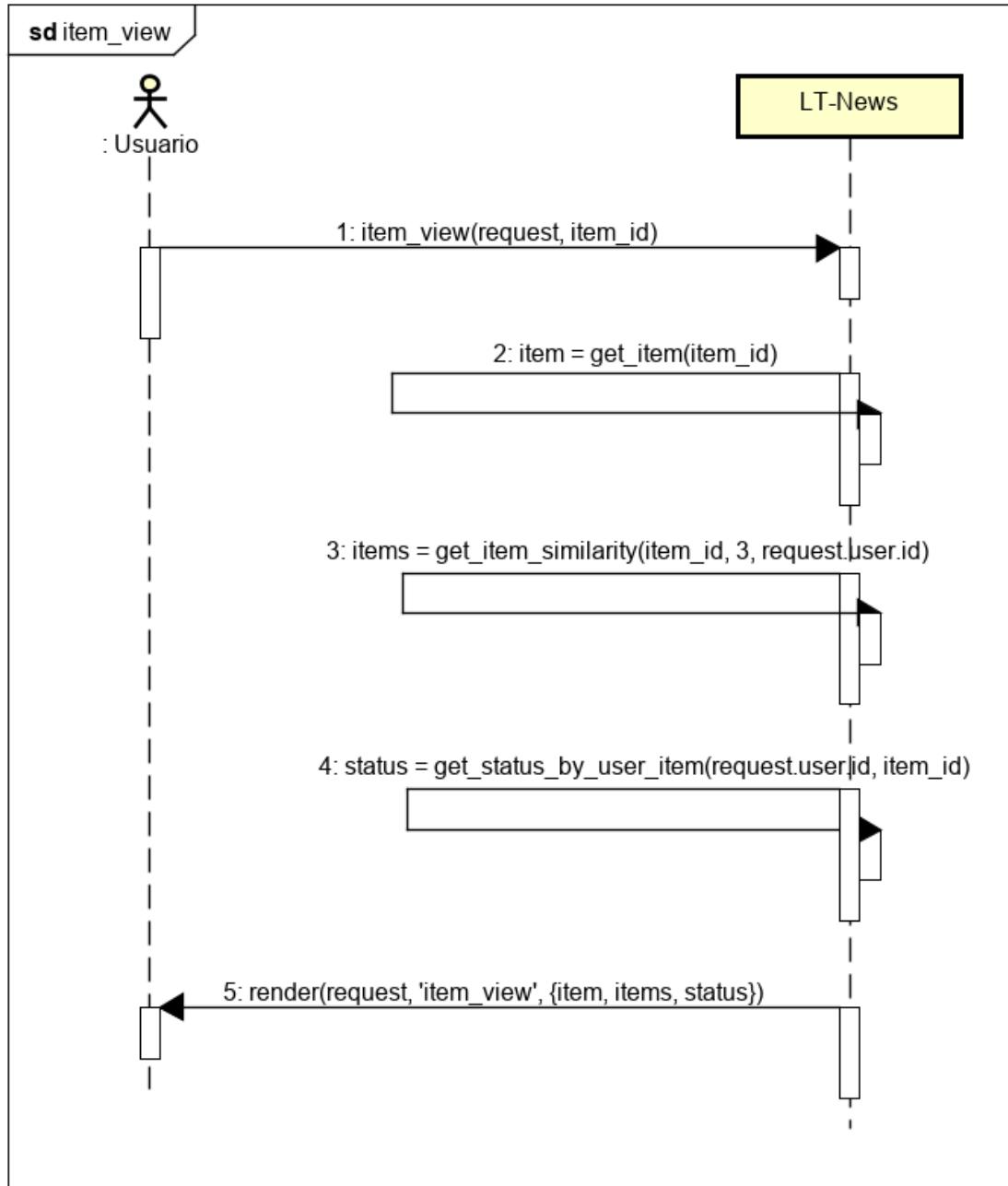


Figura 4.12: Diagrama de secuencia de item_view

Así como marcar una noticia como *Me gusta* o para guardarla.

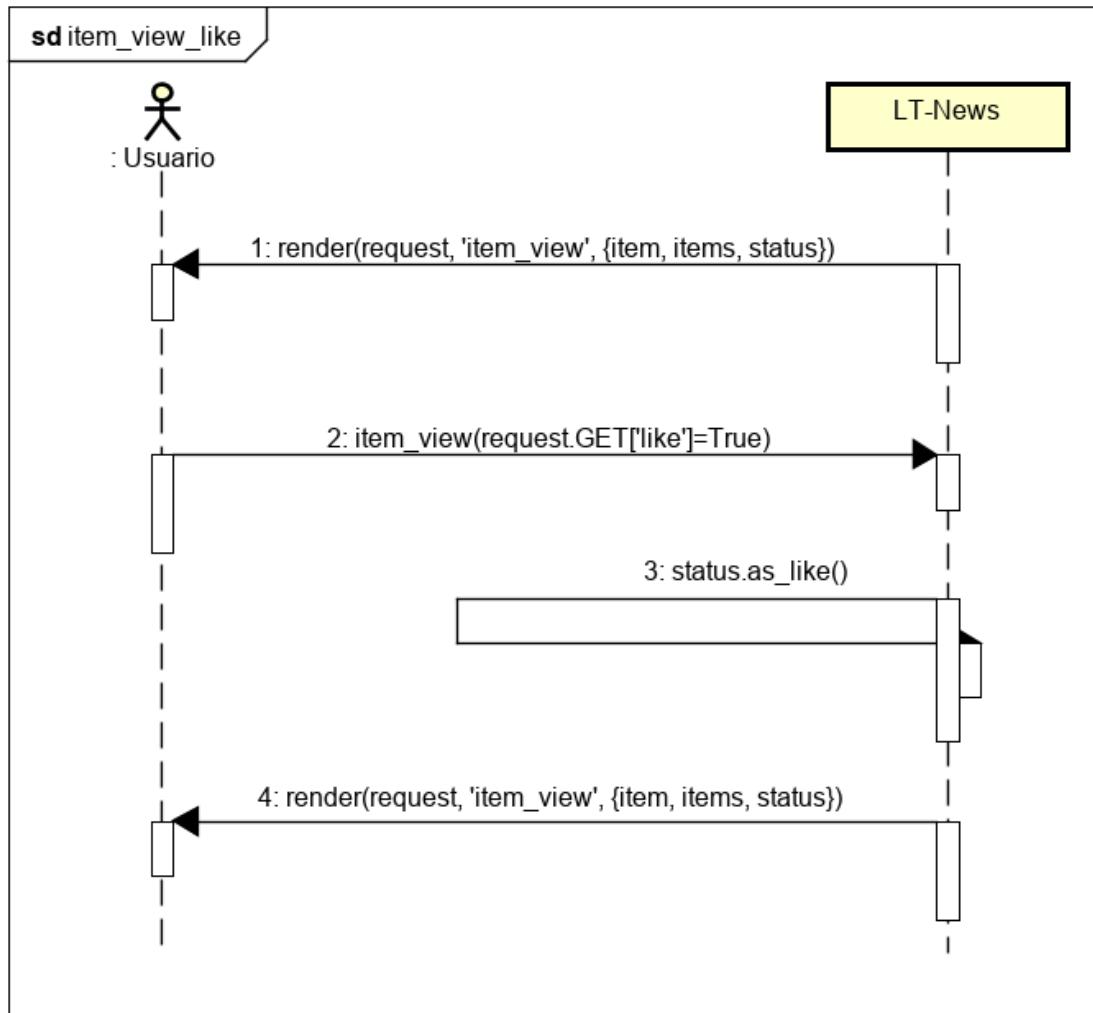


Figura 4.13: Diagrama de secuencia de item_view_like

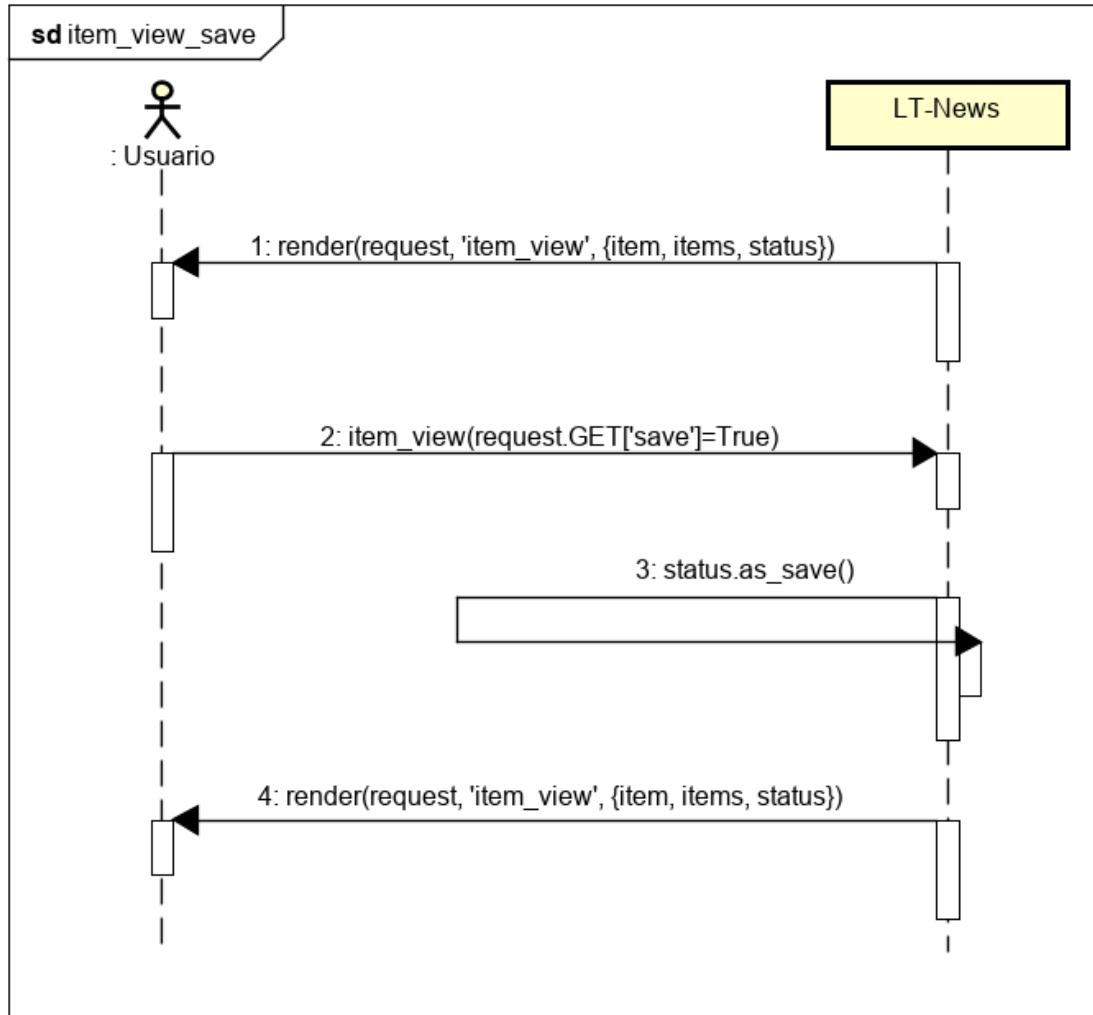


Figura 4.14: Diagrama de secuencia de item_view_save

También se pueden ir a la fuente, la noticia de origen, en su página web.

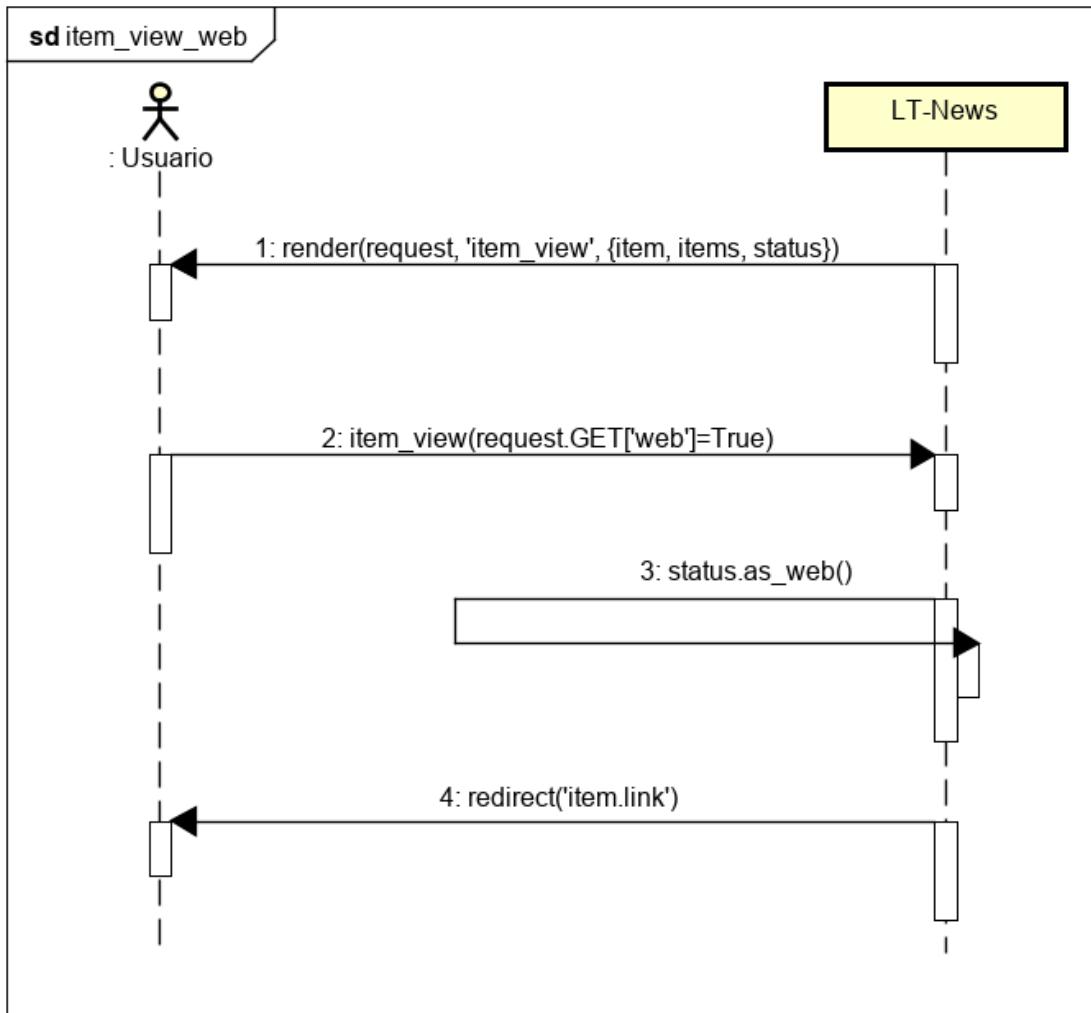


Figura 4.15: Diagrama de secuencia de item_view_web

Por otra parte, así funcionará el flujo de información para recomendar noticias.

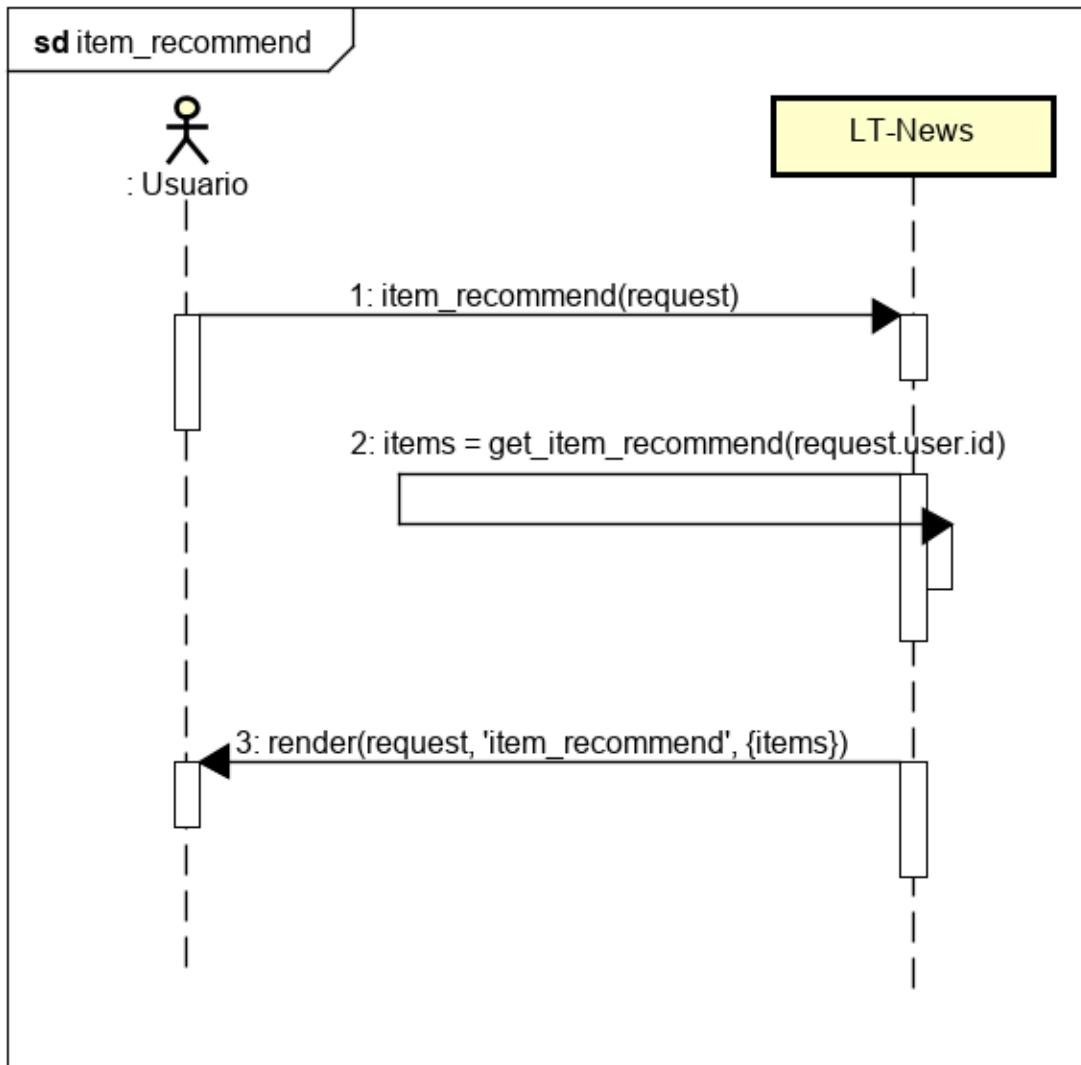


Figura 4.16: Diagrama de secuencia de item_recommend

También el sistema permitirá buscar por las palabras clave.

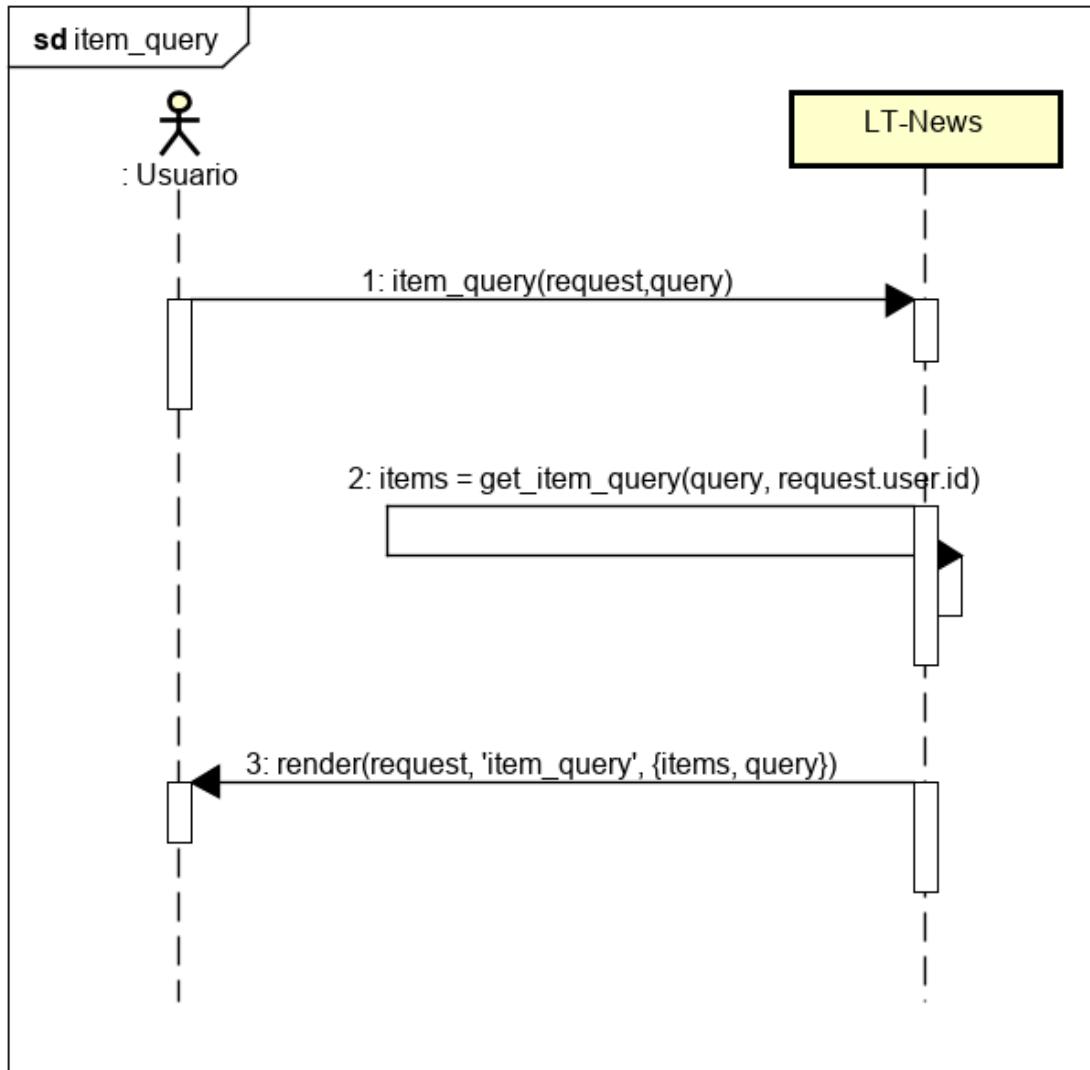


Figura 4.17: Diagrama de secuencia de item_query

Además, es posible buscar noticias en base a diferentes criterios.

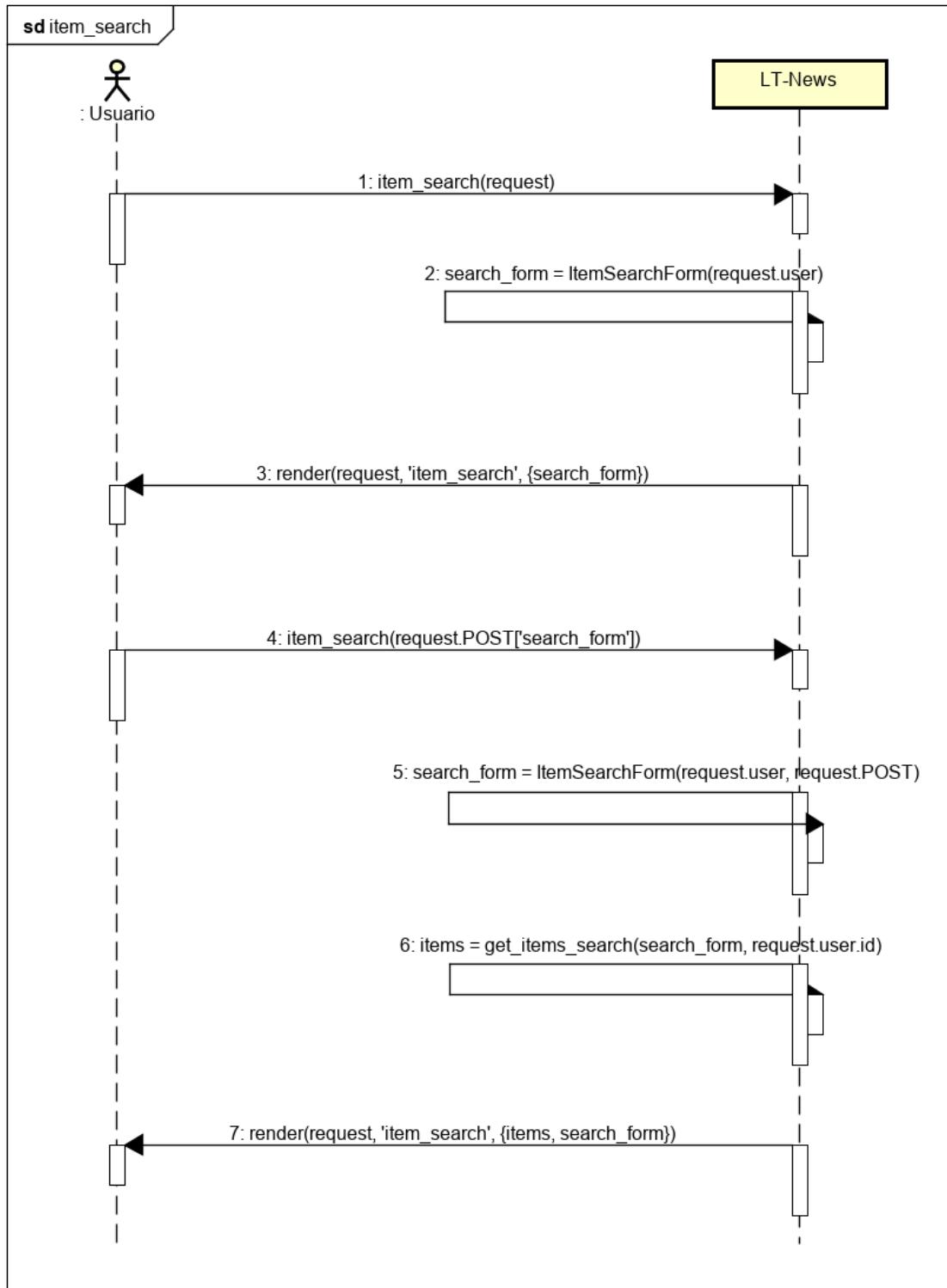


Figura 4.18: Diagrama de secuencia de item_search

El sistema también permitirá ver el resumen de noticias diario.

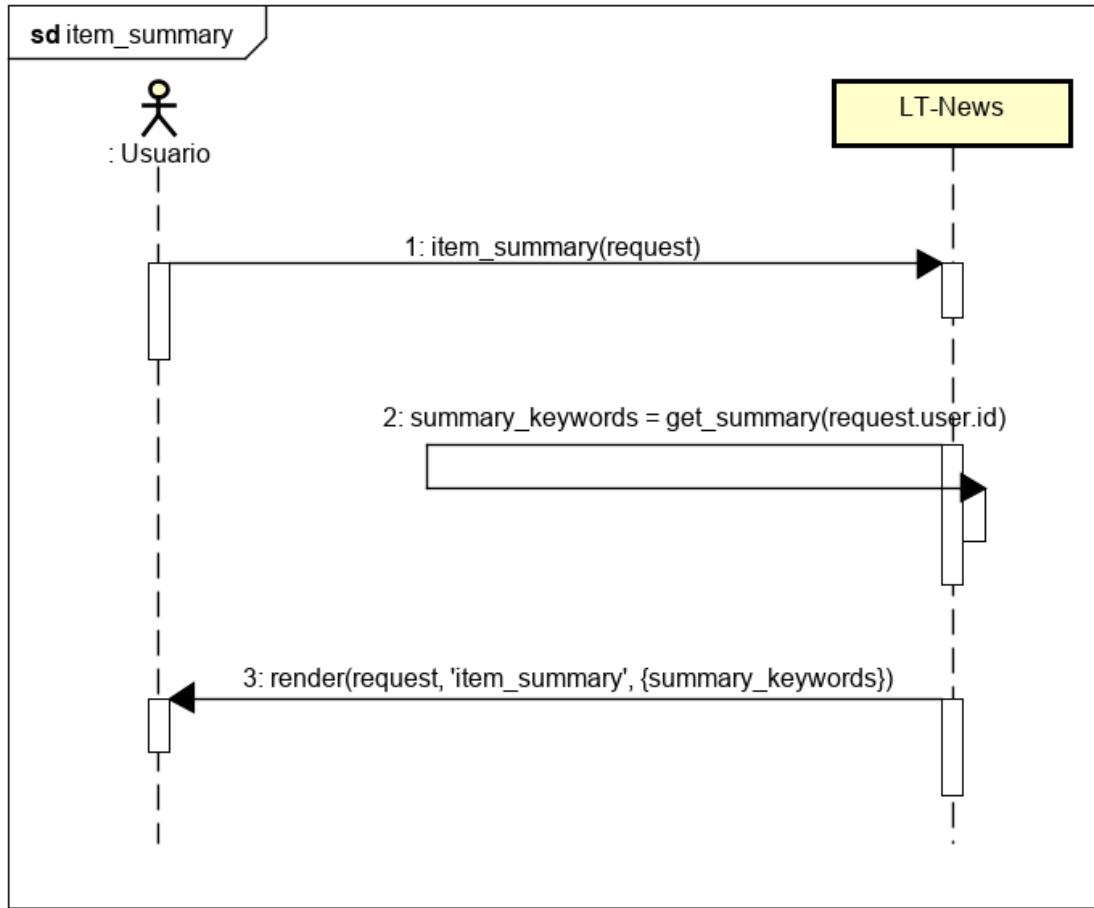


Figura 4.19: Diagrama de secuencia de item_summary

Por último, este es el flujo de comunicación para ver las noticias guardadas.

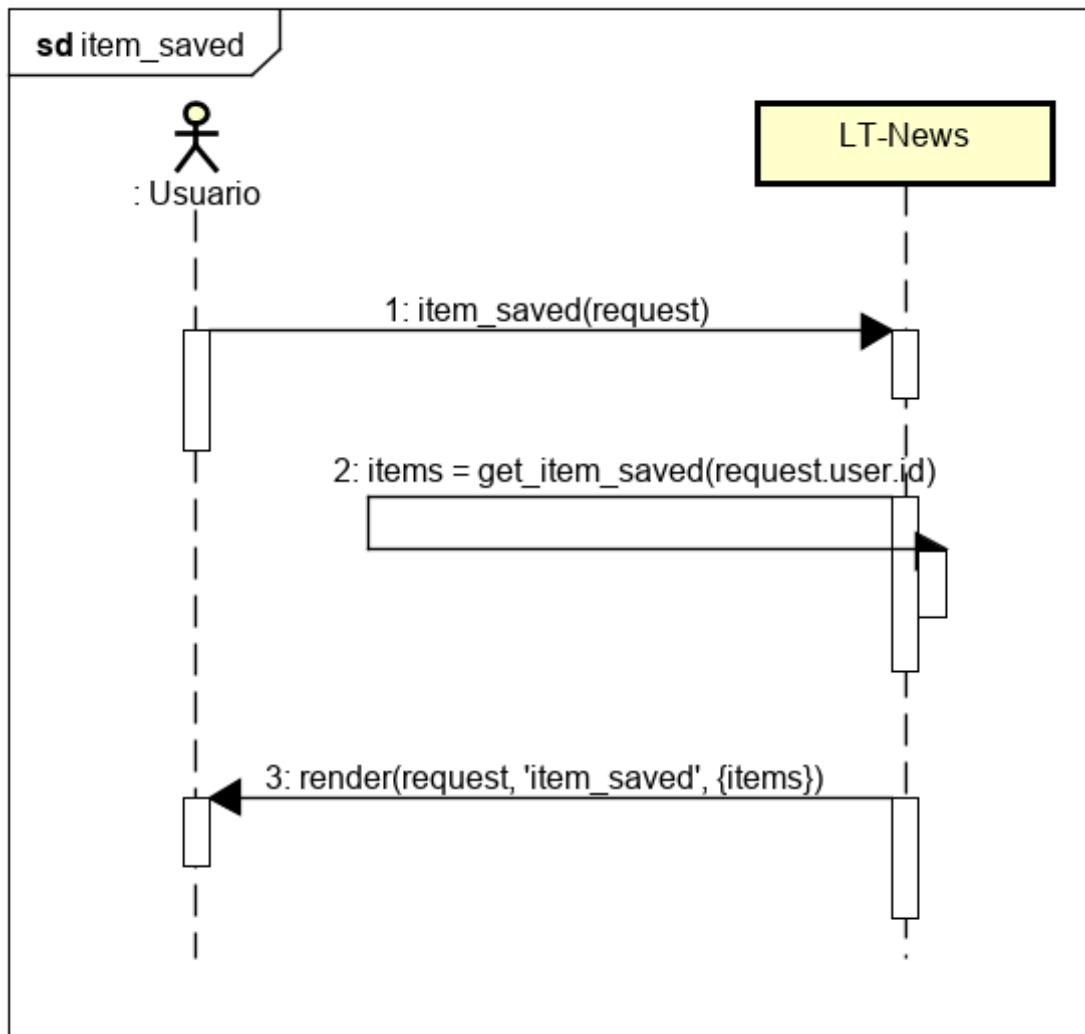


Figura 4.20: Diagrama de secuencia de item_saved

CAPÍTULO 5

Diseño

5.1– Patrones

En este apartado se mencionarán todos los patrones que se han utilizado para llevar a cabo este proyecto. Estos son soluciones concretas a problemas reales, abstrayendo tecnología y herramientas. Están enfocadas a la arquitectura del desarrollo, por lo que es imprescindible que se posean desde el principio.

Como se ha dicho, son soluciones a problemas ya estudiados y analizados desde el comienzo del software. Por ello, su utilización dará un valor añadido al código que lo hará más mantenible y reutilizable.

A continuación se encuentran los patrones utilizados. Están descritos de manera global, tal y como se puede encontrar en cualquier documentación. Esto quiere decir que no se descenderá a detalles de implementación.

5.1.1. Modelo-Vista-Controlador

El Modelo-Vista-Controlador, MVC en adelante, es un patrón de diseño que estructura el código en tres componentes principales: datos, lógica de negocio e interfaz de usuario. Para ello, propone la construcción del software diferenciando bien estos tres elementos.

- **Modelo:** su misión es la representación de la información con la que se opera que incluirá tanto los datos como la lógica para operar con ellos.
- **Vista:** se encarga de presentar los datos de forma adecuada para interactuar con los mismo desde la interfaz de usuario.
- **Controlador:** tiene como fin el hacer de intermediario entre los dos otros componentes antes mencionados.

Este patrón es enormemente eficaz para la reutilización del código entre distintos módulos del sistema y la separación del software por su misión. Esto maximiza la cohesión y reduce el acoplamiento. Además, hace que el código sea más comprensible y facilita la labor de mantenimiento. Es por estos motivos por los que se ha escogido dicho patrón.

5.1.2. Controlador-Servicio

El Controlador, como se ha dicho, es el punto neurálgico de la aplicación. Este implementará la lógica de la aplicación. Por ello, es este el que accede al modelo de datos. Con estos, transformados y adaptados, construimos la plantilla de datos, que será transferida a la Vista.

Dada la importancia de este hecho, será necesario mantener ordenado el controlador. Si no es así, se puede convertir en un *totum revolutum*: en un mismo método se puede encontrar una consulta a base de datos, una manipulación de dicho datos y su entrega posterior a una plantilla.

Una manera de simplificar el controlador es delegar alguna de sus funciones en otros componentes. Esto no es más que crear nuevos elementos que tengan una misión y se encarguen exclusivamente de la misión para la que han sido creados. Esta es una práctica cada vez más común dentro del marco del desarrollo global. Botón de muestra es la creación de arquitecturas basadas en microservicios: componentes creados para una misión *ad hoc*.

Así, el servicio tendrá una doble misión. La primera es la interacción con la base de datos. Se convertirá así para el controlador en una interfaz programática o API. Será transparente para el Controlador todo lo concerniente a la gestión de los datos: conexión y desconexión con la base de datos, consultas y actualizaciones de datos, concurrencias e hilos. No es que el servicio se encargue de todo esto, sino que es él el que interactúa con el cliente de base de datos correspondiente para ofrecer dichas funcionalidades.

La segunda misión del servicio es ofrecer unos datos acordes para trabajar con ellos. Para ello, se suelen transformar los resultados en objetos. A esta técnica se la llama: *Data Transfer Object* o DTO. Esto facilita la interacción con los datos, ya que son tratados como objetos del propio lenguaje.

En aras a lo cual, el Controlador usa al Servicio para toda la gestión con la base de datos. Con esta delegación, solo es necesaria la manipulación de los datos y la entrega a la Vista. Además, el construir una API interna aporta al código mayor versatilidad y cohesión, ya que se encontrarán agrupados los ficheros concernientes a la relación con la base de datos.

5.1.3. Desacoplamiento Cliente-Servidor

Ha sido un tema bastante estudiado la interacción entre lenguajes de programación de servidor en el cliente. Desde el inicio de las aplicaciones web, este problema se solucionó integrando herramientas al servidor que permitiesen añadir en tiempo de ejecución código al cliente. De esta filosofía, por ejemplo, nace PHP, lenguaje que se integra completamente dentro del HTML. Al igual ocurre con Java y sus JPA o con Python y sus Jinja Template.

Sin embargo, a medida que evoluciona el frontend, este se va aportando valor a sí mismo. Se empiezan a utilizar herramientas de servidor, pero para la construcción del mismo. Dicho de otro modo, es ahora en la compilación donde se consigue un código de cliente coherente y funcional. Es este el motivo, por el que se empiezan a construir aplicaciones de cliente de manera hermética a cualquier servidor.

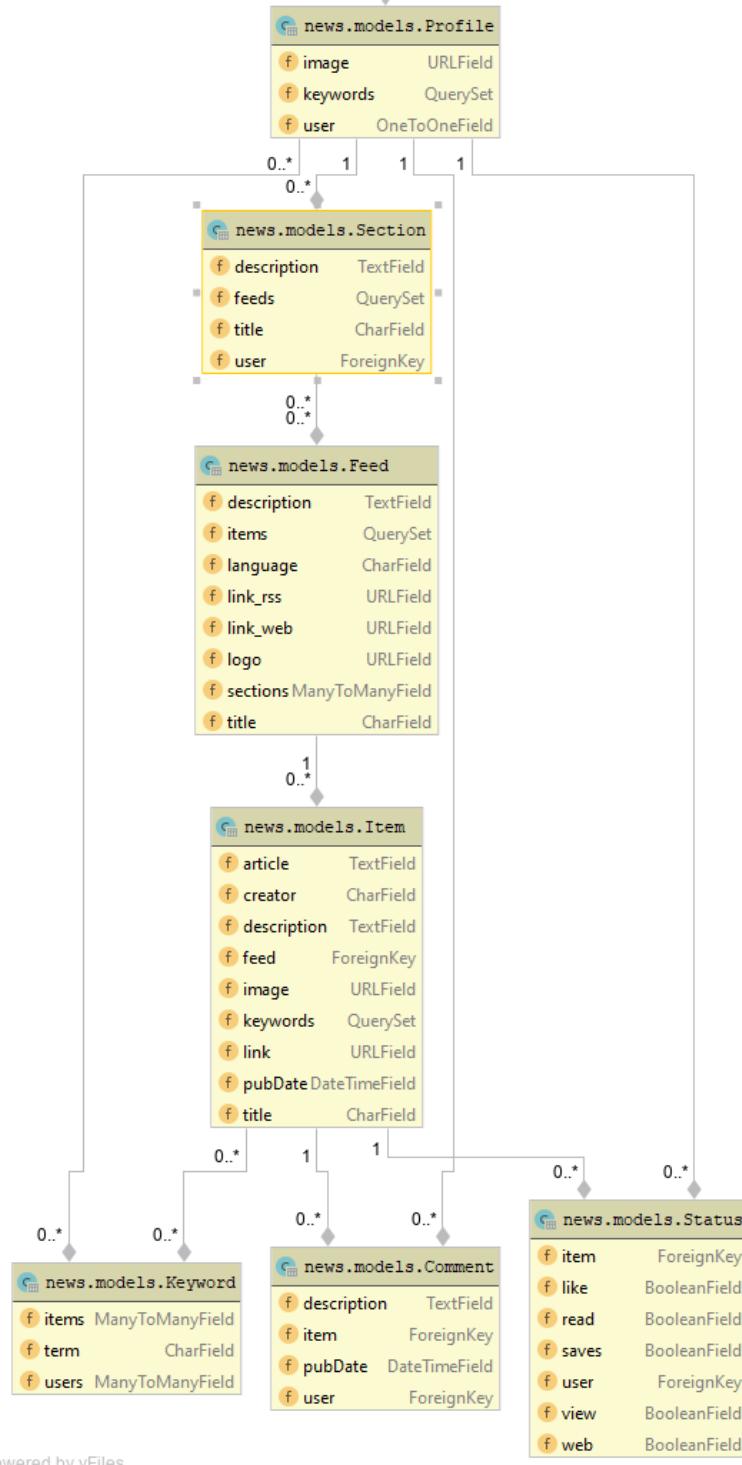
La herramienta de servidor por excelencia, como se ha llamado, es Node. Esta consigue construir aplicaciones de cliente independientes trabajando exclusivamente con las tecnologías de cliente: HTML, CSS y JavaScript. Sin embargo, a estas se les da un valor añadido al poseer este doble aspecto: tecnologías de servidor en tiempo de compilación y tecnologías de cliente en tiempo de ejecución. Por ello, todo el código que se va construyendo se transforma y empaqueta (que no compila realmente) para que sea ejecutado por el navegador.

Como se ve, este hecho, cada vez más recurrente por otra parte, hace más clara esta división entre cliente y servidor. En primer motivo, dada su naturaleza; en segundo, dada su finalidad. Aunque desde el principio la naturaleza de ambos fuera diferentes, había puntos de unión que podían confundir. Ahora, que están totalmente diferenciados, es necesario respetar su esencia.

Con vistas al respeto a la naturaleza, se observa un cambio diametral: su finalidad. Mientras el servidor ofrece una API, es el cliente quien la consume. Como se puede observar, ahora puede haber más de un cliente asociado a un servidor. Además, todos poseerán el mismo punto de entrada para acceder a los datos y lógica de la aplicación. Esto hará más mantenible el código, así como aportará más frescura al mismo.

5.2– Modelo de datos

El modelo de datos utilizado para la base de datos se ha basado en el definido en el punto anterior: 4.1 Modelo conceptual. Este diagrama ha sido generado en base a las relaciones que se poseen. Dado que anteriormente se aplican las entidades y sus relaciones, ahora será conveniente detenerse más en las propiedades y detalles técnicos.



Powered by yFiles

Figura 5.1: Modelo de datos

A continuación, se irá mencionando cada una de las clases del modelo así como sus atributos.

- User: es, como dijimos anteriormente, la entidad que gestionará a todos los usuarios. Algunos de sus atributos más relevantes son los siguientes.
 - Atributos
 - * username: nombre de usuario único en el sistema.
 - * email: correo electrónico.
 - * password: hash de la contraseña.
 - * first_name: nombre.
 - * last_name: apellidos.
 - * last_login: fecha de última conexión.
 - * is_superuser: booleano que es verdadero si el usuario es administrador.
 - * is_active: booleano que es cierto cuando el usuario esté activo.
 - * date_joined: fecha el que se registró.
 - Relaciones
 - * profile: relación OneToOne con Profile.
- Profile: en esta entidad se guarda toda la información relevante del perfil del usuario.
 - Atributos
 - * image: link de la imagen del usuario.
 - Relaciones
 - * sections: relación OneToMany con Section.
 - * comments: relación OneToMany con Comment.
 - * statuses: relación OneToMany con Status.
 - * keywords: relación ManyToMany con Keyword.
- Section: representa las secciones del usuario. Esta es una agrupación de feeds RSS. Por ello, cada sección tendrá asociado un usuario y uno o varios periódicos; estos, además, podrán estar en otras secciones.
 - Atributos
 - * title: título.
 - * description: descripción.
 - Relaciones
 - * user: relación ForeignKey con Profile.
 - * feeds: relación ManyToMany con Feed.
- Feed: esta entidad representa a todos los feeds o periódicos online. Esta poseerá varias noticias y, como dijimos anteriormente, podrá estar en varias secciones.
 - Atributos
 - * title: nombre del feed.
 - * link_rss: enlace al feed RSS.
 - * link_web: enlace al link de la web del feed.
 - * description: descripción del feed.
 - * language: idioma del feed.
 - * logo: link del logo.

- Relaciones
 - * sections: relación ManyToMany con Section.
 - * items: relación OneToMany con Item.
- Item: representa a cada noticia. Esta, como ya hemos visto, posee numerosas relaciones, dado que es el núcleo de nuestro sistema.
 - Atributos
 - * title: título de la noticia.
 - * link: enlace.
 - * description: pequeña descripción que provee el feed RSS.
 - * image: imagen de la noticia.
 - * article: cuerpo de la noticia.
 - * pubDate: fecha de publicación.
 - * creator: autor.
 - Relaciones
 - * feed: relación ForeignKey con Feed.
 - * comments: relación OneToMany con Comment.
 - * statuses: relación OneToMany con Status.
 - * keywords: relación ManyToMany con Keyword.
- Status: entidad que representa que estado posee una noticia para un usuario concreto. Todos sus atributos son booleanos; por defecto, se inicializarán a falso.
 - Atributos
 - * view: el usuario ha visto la noticia.
 - * read: el usuario ha leído la noticia o, por lo menos, ha entrado en ella.
 - * web: el usuario ha accedido a la web original de la noticia.
 - * like: el usuario ha marcado esta noticia como ‘Me gusta’.
 - * saves: el usuario ha guardado la noticia.
 - Relaciones
 - * user: relación ForeignKey con Profile.
 - * item: relación ForeignKey con Item.
- Comment: representa un comentario de un usuario a una noticia.
 - Atributos
 - * description: cuerpo del comentario.
 - * pubDate: fecha en la que se publicó.
 - Relaciones
 - * user: relación ForeignKey con Profile.
 - * item: relación ForeignKey con Item.
- Keyword: entidad que representa a una palabra clave y si esta caracteriza a un usuario, a una noticia o a ambos.
 - Atributos
 - * term: término.
 - Relaciones
 - * users: relación ManyToMany con Profile.
 - * items: relación ManyToMany con Item.

CAPÍTULO 6

Implementación

6.1– Tecnologías

A continuación se definirán las tecnologías usadas para llevar a cabo el Producto Mínimo Viable.

Web Scrapping

El aspecto fundamental del producto es la recolección de noticias de periódicos. Esta, como se ha ido diciendo a lo largo del documento, se realiza a través de RSS. Sin esta característica, no tendría utilidad la aplicación. Sin embargo, con esto solo, el aplicativo se vería inútil. Normalmente, cada medio solo publica un titular y una pequeña descripción vía RSS. Esto les sirve de gancho al usuario para que pase del lector de sindicación de contenido a la web de su medio.

Como se ha dicho, otro pilar fundamental para el buen funcionamiento del producto es el análisis de noticias. Por tanto, si no se posee más que un pequeño texto acompañado de cada noticia, poco podrá aportar este. Es por ello imprescindible conseguir el texto original del artículo para poder procesarlo correctamente. Dado que los medios no poseen esta característica, hemos de realizarla a través del *Web Scrapping*.

El *Web Scrapping* es una técnica que permite extraer información de cualquier sitio en Internet de forma automática. La información en la web se encuentra en diferentes formatos dependiendo de su misión para con el receptor. Resumidamente se encuentran dos tipos de datos: estructurados, como pueden ser ficheros XML o APIs, y no estructurados, como los ficheros HTML. Estos últimos abundan ya que, en un principio, van destinados al usuario final directamente, con la idea de ser leídos o vistos sin más.

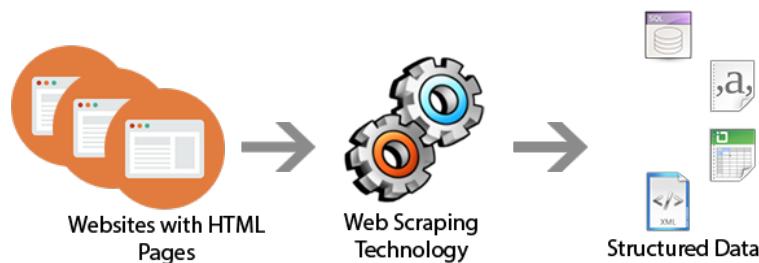


Figura 6.1: Muestra del *Web Scrapping*

Por lo dicho, esta técnica posee diferentes implementaciones, dependiendo del nivel de auto-

matización que se desee. Un primer nivel sería el de extraer el fichero HTML y trabajar con él, utilizando, por ejemplo, expresiones regulares; otro sería utilizar parsers o minería de datos, para extraer el contenido principal; y otro, el de analizar una página en concreto, para saber cómo se estructura el contenido realmente interesante y conociendo dicho patrón, aplicarlo correctamente.

Como última consideración, se ha informar que antes de trabajar con la información extraída del *Web Scrapping*, hay que saber que esta tiene implicaciones legales. Estas, en la mayoría de los casos, se encuentran en un vacío legal. Esto es debido a lo delicado del tema, en cuanto a jurisdicción, responsabilidad y autoridad de dichas fuentes.

En la práctica, si los datos se usan para un uso personal, no hay problema añadido. La cosa cambia si es para un tratamiento comercial. De hecho, actualmente, hay varios juicios en curso por un uso fraudulento de esta técnica en Australia o Estados Unidos, como se puede ver en el artículo publicado por Ainhoa Lafuente (2019).

Recuperación de información

Una vez extraída la información de los medios, es imprescindible tratarla correctamente. No basta el guardarla en una base de datos sin más, sino que se hace necesario usar medios especializados para ello para albergar las. Si se usan, por ello, técnicas de Búsqueda y Recuperación de Información (ISR, por sus siglas en inglés, en adelante) es posible tratar cada noticia como un documento y así extraer metadatos de cada una.

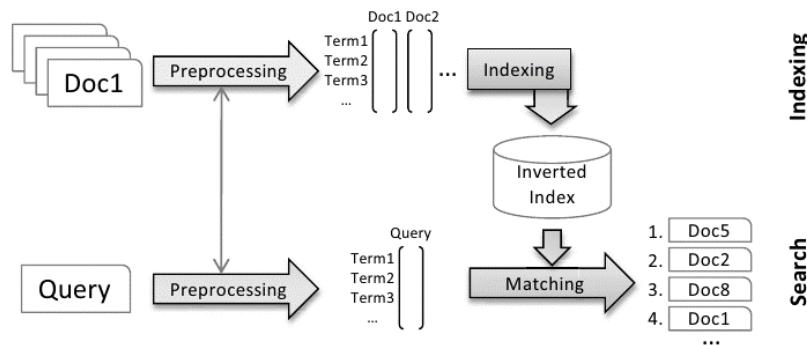


Figura 6.2: Muestra de Recuperación de Información

Esta técnica, por tanto, permite recuperar documentos en base a búsqueda de sus metadatos guardados en un índice. Esta información no suele estar estructurada al uso y su misión es la de buscar información relevante en base a diferentes filtros. Hay diferentes técnicas, dependiendo del grado de exactitud que se requiera, en base a la naturaleza de la consulta. Algunas de las más importantes son las siguientes.

- **Sistemas de recuperación de lógica difusa:** las búsquedas sobre la información son de tipo fuzzy, es decir, frases y oraciones normales en lenguaje humano, que el sistema transforma en sentencias lógicas. Así, la información que se consigue, se basa en proposiciones, teniendo en cuenta si los términos buscados aparecen o no.
- **Técnicas de ponderación de términos:** dado que hay términos de búsqueda que tienen más relevancia que otros, es lógico que se pondere dichos términos sobre el resto. Así, los documentos más relevantes, serán los que contengan el mayor número de dichos términos y que más se repitan a lo largo de su cuerpo.
- **Técnicas de clustering:** es un modelo probabilístico sobre la frecuencia de los términos de búsqueda. Se clasifican los documentos por valores que, finalmente, representan los pesos. Sobre esto, se hacen grupos por cercanía en significación.

- **Técnicas de retroalimentación por relevancia:** una vez hecha una búsqueda utilizando una de las técnicas anteriores, se vuelve a buscar pero con términos relacionados con los primeros documentos resultantes. Así, se consigue aumentar el número de documentos en la búsqueda.
- **Técnicas de stemming:** usa la técnica de Stemming para la búsqueda, es decir, trunca los términos de búsqueda quitando posibles prefijos o sufijos. En fin, evita ambigüedades léxicas. Así, se hacen búsqueda por significado, aunque como aproximación.

Como se puede observar, estas técnicas se pueden combinar entre sí dentro de una implementación concreta, aunque siempre el funcionamiento es el presentado en la imagen 6.2: se procesan los documentos, se indexan y se busca sobre el índice.

Como se ha dicho al inicio del apartado, interesa que este índice creado trate correctamente la información guardada. Esta es, por tanto, la mayor ventaja de usar un ISR: se pueden elegir técnicas y tratamientos concretos por cada campo, así como por cada índice de documentos. En el ejemplo de guardar noticias se puede decir que extraiga las palabras claves, que de más importancia al cuerpo de la noticia, que obvie mayúsculas y tildes en el índice y para el análisis, que no tenga en cuenta artículos y preposiciones.

Esto dará una versatilidad a esta *base de datos no relacional* mucho mayor, ya que estará especializada en el dominio del problema concreto. Mientras tanto, en una base de datos clásica, la información se guarda siempre de manera homogénea y genérica para cualquier dato que se quiera albergar.

Sistemas de recomendación

Una vez obtenidas las noticias y analizadas y correspondientemente albergadas en el sistema, habrá que dar uso a los metadatos generados por el ISR. Esta característica, como se ha dicho a la hora de explicar el producto, será el valor diferencial a cualquier lector de noticias: la capacidad de extraer el perfil del usuario y recomendar noticias en base a sus intereses. Esto, por tanto, se consigue con un sistema de recomendación especializado.

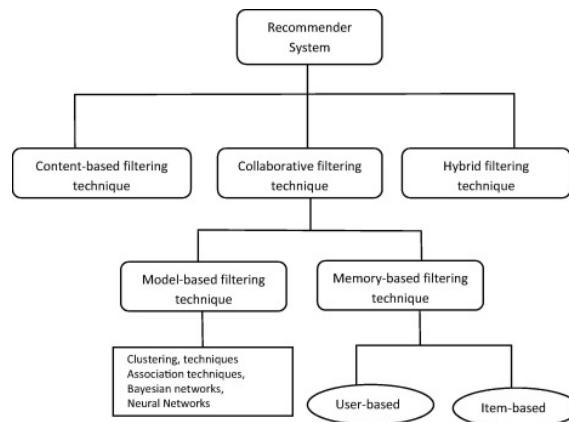


Figura 6.3: Muestra de sistemas de recomendación

Un sistema de recomendación es la técnica de filtrado de información en base a los intereses de los usuarios. En definitiva, se encarga de comparar el perfil del usuario con los ítems, en concreto para recomendarle aquellos que no conoce y podrían gustarle. Para realizar esta técnica, existe la capacidad de elección entre diversas implementaciones. Como se puede apreciar, es importante analizar tanto a usuarios como ítems a recomendar.

Analizar el perfil del usuario significa, en definitiva, extraer características de su interacción con dichos ítems. El método de extracción de sus características puede ser implícito o explícito, dependiendo de la voluntariedad del usuario al indicar al sistema sus gustos. Dentro del primero se encuentran aquellos sistemas que solicitan al usuario que valore un determinado producto, o, directamente, seleccione sus intereses. Dentro del segundo, a aquellos en los que, en base a visitas a un ítem, al tipo de interacción con el mismo, al tiempo que está viéndolo, u otras técnicas, se saca información del usuario de manera indirecta.

Ambos tienen aspectos positivos y negativos. En el primero, el usuario dice al sistema expresamente qué le gusta o disgusta, mientras el segundo, la aplicación ha de suponerlo. Esto último puede llevar a numerosos fallos de interpretación frente a la certeza que proporciona el primero. Por otro lado, el método implícito extrae dicha información sin necesidad de la intervención del usuario, cosa que el primero no puede. Esto puede hacer que en el método explícito, los interesados nunca completen sus preferencias, por el poco uso de la aplicación, lo que conllevaría a fallos en la recomendación.

Como se podrá intuir, la mejor solución depende de la naturaleza de la aplicación y el enfoque que ponga esta en el usuario objetivo. No obstante, una buena solución podrá ser un aproximación a ambas, de forma híbrida. Con esto, se poseerá las ventajas de ambas de manera sencilla. En la siguiente imagen se puede apreciar un motor de recomendación.

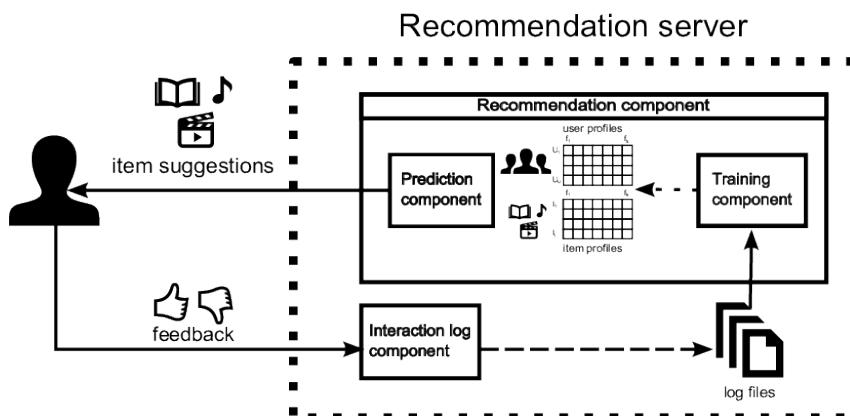


Figura 6.4: Muestra de extracción del perfil del usuario

Una vez obtenido el perfil del usuario, se ha de poder recomendarle ítems. En este apartado se aprecian dos enfoques mayoritarios (y un tercero, de la unión de ambos) como se puede apreciar en la imagen 6.3.

La primera implementación son las recomendaciones basadas en contenido. Esta técnica consiste en analizar los ítems del sistema en base a una o varias características. Así, sabiendo los ítems con los que el usuario ha interactuado, es posible recomendarle otros que posean características similares. El principal problema de este sistema de recomendación es la falta de serendipia, es decir, al usuario siempre se le muestra el mismo tipo de productos. El motivo: es la información que se tiene y sobre esta se recomienda exclusivamente. Esto conllevará un hartazgo de las recomendaciones, resultando finalmente inútiles.

La segunda implementación es el filtrado colaborativo. Esta consiste en mostrar productos que han resultado atractivos para otros usuarios con gustos similares y que todavía no ha visto. Para esto, se han de calcular similitud entre los usuarios, y no sobre los ítems, como vimos en las recomendaciones en base al contenido. El principal problema del este motor es que funcionará bien si se poseen diferentes y numerosos tipos de usuario y si se tiene de los interesados en cuestión de bastante información para tener un perfil completo. Esto es un problema, sobre todo para aplicaciones que se empiezan, lógicamente, sin afluencia.

Aquí, igual que anteriormente con el perfil del usuario, es posible combinar ambas técnicas haciendo un filtrado híbrido. Esto hará que se contrarresten los defectos de ambas, centrándose en sus virtudes. Con respecto a la falta de serendipia, el filtro colaborativo lo supera haciendo similitud entre usuarios: un interesado puede verse sorprendido al ver que se le recomienda algo que puede ser nuevo para él. Con respecto a la comunidad necesaria inicialmente, las recomendaciones basadas en contenido lo contrarrestan por la similitud de ítems: habiendo un solo usuario, funcionarán bien las sugerencias de productos.

6.2– Herramientas

Para el desarrollo del producto y, concretamente, la implementación de las tecnologías previamente mencionadas, se han usado las siguientes herramientas.

6.2.1. Backend

Para el desarrollo del servidor se han usado Python con el framework web Django, ambos en sus últimas versiones. Además, claro está, de librerías que permitan desarrollar los requisitos de la aplicación.

Python

Python es un lenguaje interpretado y multiparadigma. Esto último se debe a que soporta tanto programación orientada a objetos como imperativa, además de programación funcional. Actualmente está administrado por la fundación *Python Software Foundation* y posee una licencia de código abierto denominada *Python Software Foundation License*. Esta nos permite hacer modificaciones de su código fuente y la creación todo trabajo derivado de este, sin necesidad de que sea también código abierto.

Se ha decidido utilizar este lenguaje de programación en el lado del servidor debido a la cantidad y facilidad de uso de herramientas que implementan las tecnologías previamente mencionadas. Es la referencia en temas de *Web Scrapping* e Inteligencia Artificial, como se puede apreciar en el artículo de Ian Buckley (2018).

Django

Django es un framework de desarrollo web, también open source, que ofrece un marco inicial desarrollado sobre el patrón de diseño Modelo-Vista-Controlador. Siendo exactos, ya no se habla de *Model-View-Controller*, sino de Modelo-Plantilla-Vista. Sin embargo, hacen referencia a los mismos conceptos. El Modelo es la capa de accesos a los datos; la Plantilla, es la capa de presentación; la Vista posee la lógica de negocio de la aplicación y, por tanto, hace de puente entre ambos. Vemos, por tanto, que esencialmente representan lo mismo.

Su misión es la facilitar la creación de aplicaciones web complejas. Su valor diferencial es su acento en la conectividad, reutilización y la extensión de todos y cada uno de sus componentes. Tiene, además, como principal característica, desde el punto de vista del desarrollo, el que todo esté hecho en Python. Esto es conocido dentro de la comunidad como desarrollo *Pythonic* y ofrece un ventaja lógica: el mismo lenguaje de ejecución y compilación.

Se ha decidido utilizar este framework por la facilidad de uso, y la potencia que posee sin configuración previa. Django posee integrado, desde el primer momento, aspectos avanzados de seguridad (evita inyecciones SQL, validaciones de los formularios, autenticación en cada página, entre otros), gestión en cada momento de las sesiones, así como la transparencia en multitud de aspectos: la base de datos utilizada, la localización de los archivos estáticos, el despliegue...

Librerías

Los módulos son gestionados por Python a través del CLI *pip*. Estos se añaden automáticamente al software a través de esta herramienta y se albergan donde elija el usuario: carpeta de desarrollo o en el sistema. Además, es posible guardar una traza de las versiones usadas para replicar la arquitectura en otro sistema en un fichero de configuración: *requirements.txt*.

Las librerías que han sido necesarias para cumplir los requisitos son las indicadas a continuación:

- **django-rest-framework**: es la herramienta que permite ofrecer una API desde Django. Esta convierte las vistas del framework en un endpoint para publicitar contenido, accesible a través de los verbos HTTP.
- **django-rest-auth**: ofrece, sobre la API implementada, una capa de autenticación usando el modelo de usuarios de Django. Provee diferentes técnicas de registro, dando la posibilidad de usar redes sociales para ello, además de hacer posible el acceso a la información completa de los usuarios.
- **django-cron**: crea operaciones que se ejecutan sobre el CLI propio de Django para poder ser ejecutadas de manera recurrente. Estos métodos interaccionarán con todos los módulos de framework, haciéndolo útil para implementar operaciones *cron*.
- **gunicorn**: es un servidor HTTP que soporta WSGI. Este ofrece a través de peticiones HTTP las aplicaciones de Django.
- **beautifulsoup**: es una herramienta que permite tratar con archivos HTML de manera programática para extraer información.
- **feedparser**: permite descargar y parsear todo tipo de Feed RSS. Además, es compatible con muchos de los formatos de sindicación: desde RSS 0.9 hasta RSS 2.0; desde Atom 0.3, hasta Atom 1.0.
- **newspaper3k**: es una herramienta de obtención de noticias. Dada una dirección URI, analiza la web y calcula la localización del contenido principal y los metadatos asociados. Además, soporta la mayoría de lenguajes principales y posee herramientas de Procesamiento de Lenguaje Natural en algunos idiomas.
- **python-dateutil**: facilita el tratamiento con fechas en Python y su conversión a diferentes formatos.

Para elegir cada una, se ha realizado un pequeño estudio entre sus posibles alternativas. El resumen de esta investigación arroja la siguiente conclusión: son módulos enormemente usados en su mayoría, por lo que cada uno de ellos se ha convertido en la referencia en su tema. Es por ello que rara vez hay alguna alternativa de la misma embargadura que haga dudar al desarrollador a su elección.

6.2.2. Frontend

Para la construcción de la interfaz de usuario se ha escogido JavaScript y el framework de desarrollo VueJS.

JavaScript

Desde el inicio del desarrollo web hasta hoy en día solo han existido tres lenguajes en el mundo del frontend: HTML, CSS y JS. El primero para la estructuración de la información, el segundo

para el diseño y el tercero para la interacción del usuario y como puente entre ambos. Es sobre este tercero donde mayores avances ha habido a lo largo de los años.

Ni que decir tiene que JavaScript se ha convertido en el lenguaje de referencia en el desarrollo web si se quiere una interfaz desacoplada, asíncrona y elegante, como revela la encuesta mundial de The State of JavaScript (2019). Son estos los motivos por los que se ha escogido como lenguaje de desarrollo del lado del cliente.

De hecho, si se quiere realizar una *Single Page Application* (SPA en adelante), no hay otra opción plausible. Con dicha técnica es posible tener un simple fichero HTML e ir interactuando con él a través de JavaScript exclusivamente. Esto da al usuario una visión limpia y rápida de la aplicación en cuestión, con navegabilidad directa y sin interrupciones.

Se ha utilizado este sobre otros lenguajes de alto nivel basados en JavaScript como TypeScript o Flow por su sencillez y recomendaciones de las herramientas utilizadas.

VueJS

Una vez decidido que se pretende realizar un frontend desacoplado del lado del servidor, solo queda la elección del framework a utilizar. Realmente, esta decisión en nuestros días se reduce a tres posibles alternativas: React, Angular y VueJS. Esta afirmación se basa en los datos y dirección de la comunidad, como se puede observar en el ya citado The State of JavaScript (2019).

Si se observa, aparecen, junto a estos tres, dos framework en cuanto a utilización actual: VanillaJS y AngularJS. Sin embargo, estos dos van quedando en desuso en cuanto a framework frontend de una aplicación web como tal. VanillaJS utiliza JavaScript en su estado puro, por lo que, realmente, no se le puede considerar un framework; consiste en una librería con el *core* de JavaScript y se usa cuando se quiere máxima velocidad y mínimo espacio. AngularJS, por su parte, es el predecesor de Angular (también conocido como Angular 2). Cambia por completo su funcionamiento y arquitectura, siendo esta primera versión una librería y la segunda y posteriores, un framework. Google, autor del mismo, ha dejado de actualizar AngularJS, por lo que no ha de tenerse en cuenta para desarrollos actuales.

Visto lo anterior, comparemos los tres frameworks mencionados para ilustrar el motivo de elegir VueJS sobre los demás. Dichos datos y afirmaciones son frutos del artículo de estudio del tema realizado por Miguh Ruiz (2018).

- **React:** es la librería de Facebook centrada en la creación de vistas. Su gran virtud son los patrones de eventos, ya que estos permiten actualizar en tiempo real las vistas con los datos. Realmente no es un framework, por lo que es posible utilizarlo anexo a Angular o VueJS. Sin embargo, debido a su potencial, puede ser usado exclusivamente para realizar la capa de presentación.
- **Angular:** es el framework de Google diseñado sobre su primera versión, anteriormente mencionado, y hecho para ser totalmente independiente y funcional. Usando HTML, CSS y TypeScript consigue crear vistas, componentes independientes y rutas, todas relacionados entre sí. Trabaja sobre el concepto *Single Page Application* (SPA) utilizado para ello Webpack. Este empaquetador, una vez desarrollada la aplicación, agrupa el código en un fichero HTML (*index.html*), un fichero CSS y varios ficheros JS que interaccionarán entre sí.
- **VueJS:** es el framework mantenido por la comunidad que une las ventajas de los desarrollos anteriores, a destacar: el ciclo de vida de los componentes, propio de React, y las directivas de Angular. La curva de aprendizaje es más reducida que Angular y el código resultante poseerá menos ficheros que este, ya que aglutina HTML, CSS y JS en un fichero de extensión *.vue*. Además, es totalmente versátil: se puede utilizar como librería para un desarrollo pequeño hasta como framework en un aplicación SPA o SSR completa.

Son estos últimos motivos los que han llevado a optar por VueJS. Además, el ecosistema que ofrece Vue, como se verá en el siguiente punto, es único.

Librerías

Los módulos usados están relacionados con el ecosistema que ofrece Vue así como por la comunidad de JavaScript. Estos se gestionan a través de *npm* o *yarn* y al igual que ocurría con *pip*, condensa la información en un fichero para replicar la infraestructura: *package.json*.

- **vue-router**: gestiona la navegabilidad entre las vistas, compartiendo datos entre sí y haciéndolas accesibles desde la URL del navegador.
- **vuex**: administra los estados de determinados objetos a lo largo de toda la aplicación. Ofrece de manera general un almacén de estados que pueden ser mutados y consultados desde cualquier módulo.
- **vuetify**: implementa un ecosistema basado en Material Design, el diseño propuesto por Google, usado componentes Vue.
- **axios**: gestiona las peticiones HTTP en JavaScript ofreciendo una API completa y simple.
- **moment**: convierte cualquier campo de tipo fecha al formato deseado por el usuario.

Todas estas librerías son usadas por la gran mayoría de la comunidad y no poseen alternativas claras y robustas: los desarrolladores prefieren mejorarlas a enfrentarse a ser competencia directa.

6.2.3. Servidor

En este apartado se mencionarán las tecnologías usadas en la parte del servidor, de manera conjunta a las de Cliente y Servidor previamente mencionadas.

PostgreSQL

Es imprescindible usar en este tipo de proyectos una base de datos consistente. Además, dada la naturaleza de la aplicación en cuestión, esta ha de ser Relacional. De todas las posibilidades que ofrece este gran mundo, Django recomienda usar PostgreSQL debido a su naturaleza y fuerte interconexión con su framework.

Postgres es un sistema gestor de bases de datos de tipo relacional, como se ha dicho. Posee una gran escalabilidad, al usar una infraestructura propia de paralelización: multiprocesos en vez de multihilos. Esto implica que desacopla los posibles errores propios de la gestión de hilos, como son las condiciones de carrera. Es de código abierto y con un gran uso dentro de proyectos *open source*.

Elasticsearch

Para implementar la Búsqueda y Recuperación de Información, descrito en el punto 6.1, referente a las tecnologías, es necesario usar un Índice.

Elasticsearch es un servidor de búsqueda, distribuido y accesible a través de una interfaz de tipo RESTful. Está desarrollado en Java, basándose concretamente en Lucene, y es de código abierto. Propone una sintaxis basada en JSON para las búsquedas.

Se ha decidido usar sobre índices como Algolia o Solr por dos motivos principales. El primero se debe a su coste, ya que Elasticsearch es gratuito en implicaciones *on premise*, es decir, con puesta en marcha y mantenimiento propio. El segundo motivo es el lugar dentro de la comunidad: se está poniendo a la cabeza en término de uso y preferencias de los desarrolladores.

Nginx

Una vez albergado y desarrollada la aplicación, es necesario hacerla accesible a través de Internet de manera segura y consistente. Para ello, hace falta un servidor web o proxy inverso. Las dos grandes alternativas en este punto son Nginx y Apache.

Nginx es un tipo de servidor web ligero y con alto rendimiento. Es open source y permite, además, integración con servidores SMTP, así como soporte de HTTP, SSL, HTTP2, IPv6 y demás protocolos.

Se ha decidido usar este por el gran rendimiento que ofrece sobre el Servidor HTTP Apache. Por otra parte, ofrece una gran comunicación y facilidad en la configuración en aplicaciones Full Stack.

Docker

Para hacer toda esta arquitectura independiente a cualquier sistema y con la configuración similar a entornos de desarrollo y producción es necesaria la virtualización de los servicios. Esto ha marcado un nuevo hito en el mundo del software: la abstracción entre sistemas operativos y la concentración de la configuración en un único fichero. Esto es Docker.

Docker automatiza el despliegue de cualquier software en contenedores. Estos poseerán toda la configuración que necesita cualquier desarrollo para funcionar. Se le considera la evolución lógica de la máquina virtual: se convierten en instancias Linux ligeras y aisladas, configurables desde ficheros de texto plano.

Actualmente, no hay una alternativa clara a Docker, debido a su potencia y al hecho de ser una tecnología *open source*. De hecho, tecnologías de orquestación de contenedores, tales como Swarm o Kubernetes, se basan en esta.

6.2.4. Gestión

Finalmente, se hará mención a herramientas usadas para la administración. En cuanto a la gestión del proyecto se refiere, se hablará de dos grandes ámbitos: desarrollo y seguimiento del proyecto.

Desarrollo

Para el desarrollo del proyecto han sido necesarias las herramientas que se mencionarán adelante. La preferencia general de dicha elección se basa en dos premisas. La primera es la orientación de la comunidad del desarrollo y la segunda es la disponibilidad real de dichas herramientas. Además de utilizar tecnologías gratuitas, se ha hecho uso de versiones de prueba de herramientas de pago ofrecidas por la universidad.

En primer lugar son necesarios los programas para el desarrollo de las diferentes partes de la aplicación. Dado que cada módulo se desarrolla en un lenguaje concreto, se ha apostado por usar IDEs personalizados. Así, y debido a su potencial, se han usado todos aquellos necesarios por lenguajes creados por JetBrains. En el caso de no existir, se ha decidido acudir a Visual Studio Code.

Se ha usado también una herramienta para la realización de diagramas llamada AstahUML. Esto permite analizar lo que se necesita antes de implementarlo. Este programa posee soporte para todo tipo de diagramas: como son el de estados o de clases. Se ha usado esta gracias a su gratuidad para estudiantes.

Para mantener una trazabilidad y respaldo del código, se han usado tecnologías de control de versiones. En concreto, la elegida ha sido Git, debido a su posición en la comunidad del desarrollo y a su funcionalidad y robustez. La plataforma remota donde se alojará dicho código sobre Git es GitHub. Ha sido elegida por ser la principal usada por la comunidad.

En cuanto a la comunicación y despliegue por el servidor, se ha elegido como VPS: Digital Ocean. Posee un enorme potencial y un periodo de prueba más que suficiente para la distribución de la aplicación. Además, para la comunicación con este, se ha usado WinSCP y PuTTY, como herramientas SSH principalmente usadas en Windows.

Seguimiento

Una vez llegado y concluidas las decisiones convenientes para desarrollar el producto, es necesario seguir una serie de pautas, marcarse unos hitos y tener la posibilidad de comprobar el avance del proyecto. Así, han de usarse una serie de herramientas para medir la evolución y seguir con acierto.

Se han usado grosso modo dos aplicaciones para ella: Toggl para la gestión del tiempo y Waffle para la gestión de las tareas. La primera permite comprobar las horas que lleva la consecución de cada tarea. Esta permite etiquetar cada una, por lo que se podrá comprobar el tiempo dedicado a cada tipo de tarea (gestión, documentación, desarrollo, despliegue o investigación, en el caso que se presenta), así como el total dedicado al proyecto. Con la segunda se consigue agrupar las tareas en hitos y asociarlos a funcionalidades dentro del código. Además, ambas están relacionadas, por lo que simplifica dicha gestión, siendo esta la principal motivación para su uso.

Waffle es la implementación concreta del tablero canvas mencionado en el punto 2.1. Si se aprecia la imagen 3.5, vemos que es GitHub la plataforma mencionada para el tablero. Lo ocurrido se puede apreciar en el artículo de Waffle (2019). Esta herramienta deja de mantenerse por falta de recursos y recomienda su migración a GitHub.

6.3– Estructura del proyecto

Como se ha ido diciendo a lo largo del documento, el proyecto utiliza Git. Así, se ha estructurado el trabajo en ramas según la metodología Git Flow. Esta estructura cualquier proyecto software con cinco ramas principales:

- **master**: el código en esta rama es la versión estable y funcional que se encuentra funcionando en producción.
- **hostfix**: cualquier fallo detectado en producción se arregla aquí y se lleva a *master* una vez solucionado: no antes.
- **release**: sobre los cambios agrupados del desarrollo, que se prueban integralmente antes de ser subidos a *master*.
- **development**: agrupa los cambios en las diferentes características que se van trabajando.
- **feature**: cada una de las funcionalidades a añadir en el sistema.

La explicación dada de cada una de las ramas e interacción entre ellas se pueden ver gráficamente en la siguiente imagen.

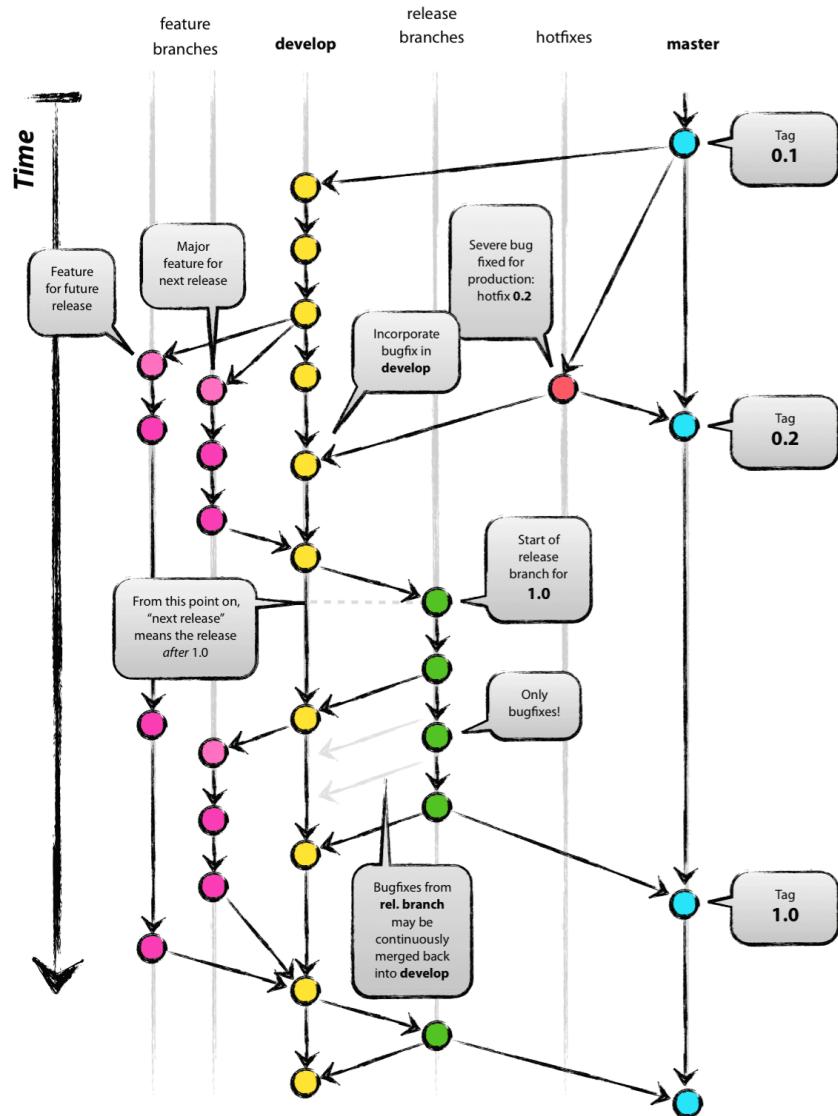


Figura 6.5: Git Flow

Teniendo esto en cuenta, se mostrará cómo se estructura cada repositorio de manera general en GitHub, para luego descender, de manera individual, a cada uno.

6.3.1. GitHub

De manera general, se poseen estos repositorios de proyectos en la plataforma de código abierto GitHub. Cada uno de los cuales alberga un desarrollo concreto.

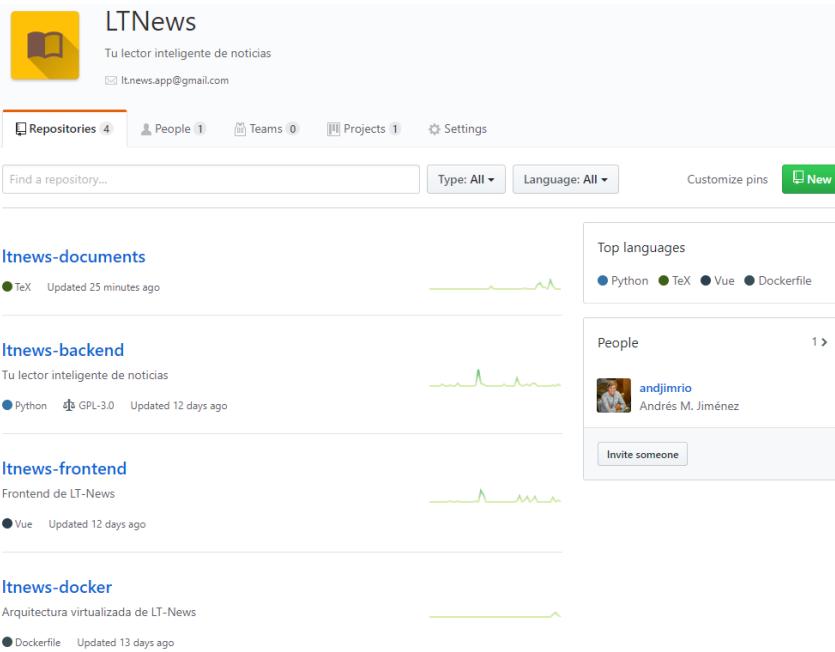


Figura 6.6: Respositorios en GitHub

Los repositorios que muestra la imagen son los siguientes:

- **Itnews-documents**: es el proyecto LaTeX donde se desarrollan los documentos a presentar, tanto la presente memoria como la presentación del trabajo.
- **Itnews-docker**: alberga la arquitectura de la aplicación en docker y se relaciona con los demás repositorios de código.
- **Itnews-backend**: es el proyecto de Django para el servidor de la aplicación.
- **Itnews-frontend**: es la capa de presentación de la aplicación realizada en VueJS.

Gracias a la integración que realiza GitHub, se ha añadido metacontenido a los commits que se van realizando. En concreto, se han utilizado tres entidades: *issues*, *labels* y *milestones*.

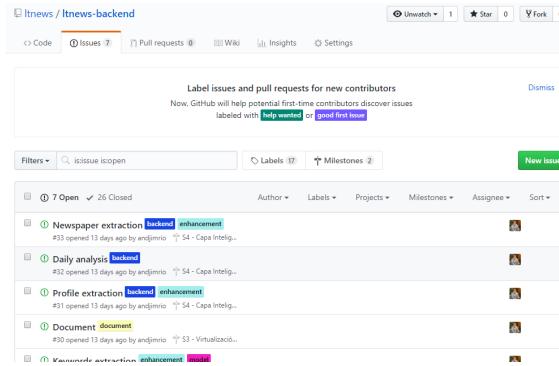
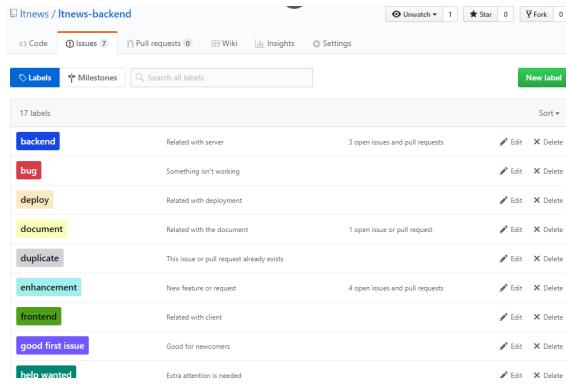
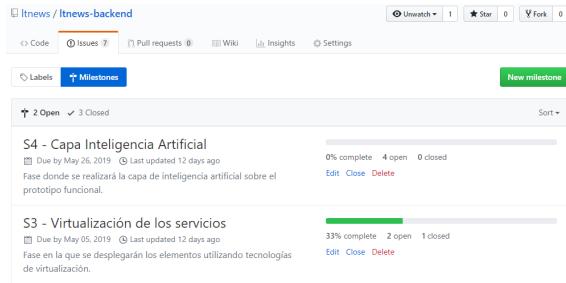


Figura 6.7: Issues en Github

Del primero, los *issues*, comentar que GitHub permite usar esta entidad como una herramienta integrada de gestión de incidencias. Como aparece en la siguiente imagen, cada *issue* viene relacionado con: proyectos, hitos, encargados o etiquetas. Además, poseen una traza de comentarios y commits relacionados. Es este el motivo de su uso como gestor de tareas del proyecto.

Figura 6.8: *Labels* en Github

A la vez, como se ha dicho, se pueden utilizar diferentes etiquetas para relacionar las incidencias. Estas son las que aparecen en la imagen 6.8. La idea general es poder clasificar cada tarea según su entidad correspondiente: plataforma relacionada o tipo de tarea. De esta manera, es fácilmente clasificable cada una de las tareas.

Figura 6.9: *Milestones* en Github

En la última entidad, se agrupan también las tareas o *issues* en hitos, llamadas *milestones*. Esto sirve para indicar las fases del proyecto y la fecha de finalización de cada sprint.

A un nivel más operativo, que un repositorio esté en GitHub conlleva a que estén presentes los siguientes archivos:

- .gitignore: es el archivo que contiene los elementos que deberá ignorar git en su proceso de control de versiones.
- LICENCE: es la licencia del proyecto, que en el presente caso es MIT Licence para todos los proyectos de 6.6: esta es una licencia de software permisiva, que posee pocas limitaciones en la reutilización.
- README.md: contiene la información imprescindible del proyecto que aparecerá reflejada en GitHub.

6.3.2. Backend

En cuanto al código del servidor, se ha estructurado el código siguiendo las pautas recomendadas tanto por Python, como por Django. Así, la estructuración en carpetas es la que aparece en la imagen 6.10. Estará se irá desgranando para explicar su lógica.

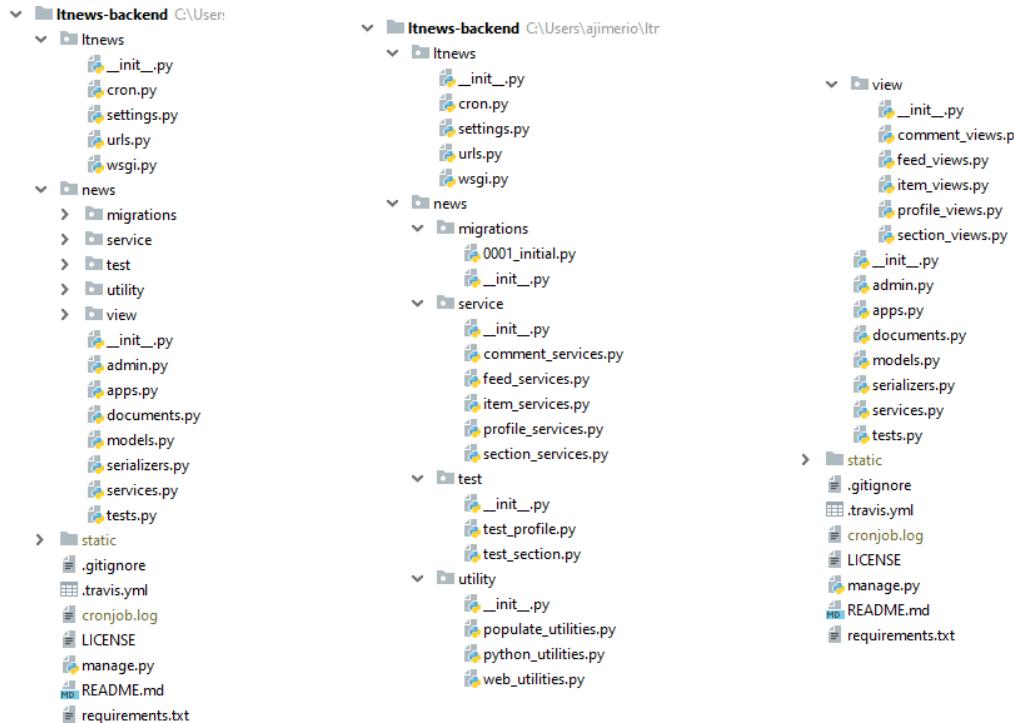


Figura 6.10: Estructura en PyCharm

- **Itnews**: contiene la lógica de funcionamiento de Django.
 - cron.py: archivo que contiene las tareas recurrentes de la aplicación, entre ellas, la que actualiza las noticias y la que calcula el perfil del usuario.
 - settings.py: archivo que contiene las propiedades principales de nuestra aplicación.
 - urls.py: archivo con los accesos a las direcciones accesibles a nuestro proyecto; relaciona controlador con url.
 - wsgi.py: archivo que tiene la información y propiedades del despliegue del servidor Django.
- **news**: contiene la lógica de la aplicación implementada.
 - **migrations**: contiene las migraciones hechas del modelo de Django.
 - **service**: contiene los servicios de las entidades por cada entidad del modelo y son intermediarios entre el modelo y el controlador, ya que extraen la información de la base de datos ya ordenada y útil de utilizar en los controladores.
 - **test**: contiene los tests agrupados en las entidades del sistema.
 - **utility**: contiene las clases de utilidad que han sido necesarias desarrollar la el correcto funcionamiento del sistema.
 - **view**: contiene los controladores por entidad, en concreto, la implementación de cada API publicada.
 - admin.py: archivo en el que se declara el nombre de las entidades del sistema.
 - apps.py: archivo con nombre y características de la aplicación.
 - documents.py: archivo que contiene la comunicación de Elasticsearch con la aplicación así como la implementación de los documentos.
 - models.py: archivo que contiene las entidades del modelo.

- serializers.py: archivo que contiene la implementación de serialización de cada entidad del modelo a través de la API.
- services.py: archivo con los servicios genéricos que usan todas las entidades.
- tests.py: archivos que contiene las pruebas generales del sistema.
- **static**: alberga los ficheros estáticos generados para el panel de administración.
- .travis.yml: los comandos que se ejecutarán en la herramienta de pruebas de integración continua Travis.
- manage.py: es el archivo principal de Django, el archivo de ejecución principal.
- requirements.txt: contiene todos los componentes que usa el proyecto de Python, así como sus versiones.

6.3.3. Frontend

El código del cliente se estructura siguiendo las directrices del CLI de Vue. Estas no son más que una implementación concreta y ordenada de las pautas de Node y Webpack. Así, la organización se muestra en la imagen 6.11.

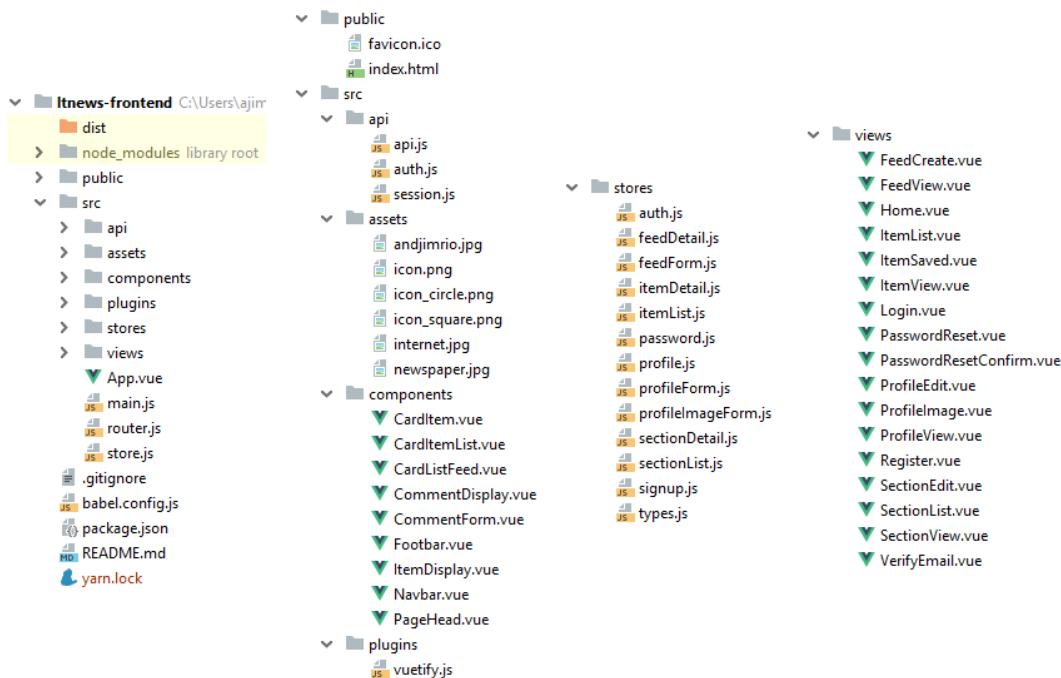


Figura 6.11: Estructura en WebStorm

- **dist**: contiene la construcción de la aplicación en formato SPA realizada por *npm* o *yarn*.
- **node_modules**: contiene los paquetes necesarios por *npm* para la construcción y ejecución de la aplicación.
- **public**: contiene los ficheros globales de la aplicación tales como el índice o el favicon.
- **src**: contiene la lógica general de la aplicación siguiendo la estructuración de Webpack.
 - **api**: contiene las llamadas generales a la API y la configuración de estas por *axios*.

- **assets**: contiene las imágenes y ficheros estáticos usados por la aplicación.
- **components**: contiene los componentes de la aplicación: estos funcionan de manera independiente y pueden ser usados en cualquier vista.
- **plugins**: contiene la configuración de algunos módulos usados por la aplicación, como en el caso de *Vuetify*.
- **stores**: contiene la configuración de estados de las diferentes entidades creadas; don gestionadas por *vuex*.
- **views**: contiene las vistas que posee la aplicación con la lógica implementada en cada una.
 - App.vue: archivo con la arquitectura visual que presenta la aplicación.
 - main.js: archivo con la configuración general de la aplicación Vue.
 - router.js: archivo con la navegabilidad entre las diferentes vistas así como su relación con las URLs.
 - store.js: archivo con la relación de estados y mutaciones especificados en la carpeta **stores**.
- babel.config.js: archivo con la configuración de Babel.
- package.json: archivo con el fichero de configuración de la aplicación con la información general y de librerías necesarias y uso específico de estas.
- yarn.lock: archivo con la configuración generada por las herramientas de node antes mencionadas.

CAPÍTULO 7

Pruebas

7.1– Pruebas unitarias

Los test unitarios prueban una funcionalidad en toda su completitud y tipología de casuísticas. Debido a la complejidad y gran tiempo que conlleva la realización de las pruebas, se han reducido estas a una parte representativa, tanto en número como en naturaleza. Este es el motivo por el que se han realizado exclusivamente en el lado del servidor, es decir, en el backend.

Para desarrollarlas, por tanto, se ha utilizado el framework de pruebas que proporciona Django. Con este, solamente haciendo uso de la función *Test*, se pueden ejecutar todas las pruebas a la vez. Se han comprobado principalmente los usos principales sobre una parte de la aplicación.

En total se han realizado en torno a 20 pruebas unitarias de dos entidades exclusivamente: *Section* y *Profile*. Estas comprueban toda la lógica que se ofrece por la API, haciendo uso de una nueva base de datos para ello. Además, contienen tanto las pruebas positivas y negativas. Esto quiere decir que se comprueban el correcto funcionamiento de estas entidades, así como los posibles errores que podrían acontecer.

7.2– Pruebas integradas

Como se ha dicho, los tests unitarios comprueban una funcionalidad concreta del sistema. Aunque hubiera pruebas sobre cada característica de la aplicación, esto no implicaría la comprobación de la aplicación en su conjunto. Es por ello importante el análisis de la interacción de los distintos componentes del sistema. Estas son las pruebas integradas.

Estas se realizan en la aplicación utilizando un sistema externo: Travis. Este, con un fichero de configuración, visto en el apartado 6.3, compone la arquitectura del sistema y realiza todo el conjunto de pruebas de manera global. Así, se consigue integrar toda la funcionalidad en una misma ejecución.

Travis analiza cada commit realizado sobre los proyectos que analiza. Así, cada vez que detecta un cambio en el repositorio de código, construye el sistema con las indicaciones que le han sido dadas y realiza las pruebas con esa versión del código. De esta manera, se consigue tener un reporte en tiempo real de qué versiones del software son coherentes con lo esperado y cuáles propician fallos en la aplicación.

La siguiente imagen muestra los diferentes logs de las últimas ejecuciones de Travis. Se puede ver que hay algunos en error. Esto es debido al cambio de composición de la aplicación: pasa de el montaje de cada componente individualmente a una servicios interconectados y virtualizados con Docker.

ltnews / ltnews-backend		build failing		
Current	Branches	Build History	Pull Requests	
X development	draft #28: add advanced search to API	⌚ #53 failed ⌚ 1 min 34 sec ⌚ 13 days ago	⌚ 1 min 34 sec ⌚ 13 days ago	(C)
X development	draft #28: add simple search to API	⌚ #52 failed ⌚ 1 min 37 sec ⌚ 13 days ago	⌚ 1 min 37 sec ⌚ 13 days ago	(C)
X master	fix #27: add docker environment	⌚ #51 failed ⌚ 1 min 32 sec ⌚ 13 days ago	⌚ 1 min 32 sec ⌚ 13 days ago	(C)
X master	CRON fix #27: add docker configuration and up	⌚ #50 failed ⌚ 1 min 46 sec ⌚ 14 days ago	⌚ 1 min 46 sec ⌚ 14 days ago	(C)
X master	fix #27: add docker configuration and update the	⌚ #49 failed ⌚ 1 min 34 sec ⌚ 14 days ago	⌚ 1 min 34 sec ⌚ 14 days ago	(C)
✓ v0.2	new version of dependencies	⌚ #48 passed ⌚ 1 min 29 sec ⌚ 27 days ago	⌚ 1 min 29 sec ⌚ 27 days ago	(C)
/ master	new version of dependencies	⌚ #47 passed ⌚ 1 min 12 sec	⌚ 1 min 12 sec	(C)

Figura 7.1: Muestra de logs de Travis

Al igual que ocurría con las pruebas unitarias, se ha realizado un esquema de pruebas integradas más como concepto que como aplicación real al proyecto. Este es el hecho por el que hay pocas pruebas funcionales del mismo y que no se haya continuado con el cambio de arquitectura.

7.3– Pruebas de aceptación

Se ha hablado anteriormente de la figura del *early adopter* como una pieza fundamental dentro del estudio y comprobación de la idea. Estas personas selectas, debido a su rol, se han encargado también de ir testando paulatinamente la aplicación. No de manera fría y accidentada, buscando fallos o faltas de usabilidad. Más bien han intentado usar el sistema desde un primer momento en aras a convertirla en su futura referencia en cuanto a lector de noticias se refiere.

Dicha utilización ha sido regular, aunque diferente en cada uno de ellos. En algunos consistía en un uso esporádico cada varios días. En otros se ha convertido casi en una costumbre semanal o mensual. Esto ha servido de acicate para la mejora de la calidad de los resultados y el acabado desde el punto de vista del diseño.

Estos, además, daban su opinión de cambios acontecidos en la aplicación. Gracias a esto, se ha podido ir modificando las diferentes características del sistema y se ha tenido en un periodo de tiempo corto el feedback de estos perfiles. Como botón de muestra de su aportación encontramos: el diseño de los colores, la latencia de determinadas vistas o la sugerencia del resumen de noticias.

Esto ha proporcionado al sistema de un flujo de comunicación directa con los clientes. Esto está basado en la metodología *Running Lean*, como se indicaba en el punto 2.1. Aporta un gran valor al aplicativo, ya que se hace partícipe al usuario de las decisiones.

En un futuro se estandarizará dicho proceso y se ampliará de a los *early adopters* a cualquier usuario. Se utilizará como herramienta de feedback imprescindible para cualquier desarrollo que se realice. Así, se poseerá el flujo completo de comunicación. Este comenzará desde el lanzamiento de una nueva característica y culminará con el estudio de la recepción de la misma. De esta manera, será posible controlar la gestión del cambio.

7.4– Estándares de código

Uno de los aspectos más importantes a la hora de llevar a cabo un proyecto dentro del área de la ingeniería del software es la calidad. De hecho, gran parte del éxito de una aplicación consiste en la perfección con la que se realiza en cada uno de sus aspectos. No solo es importante cumplir los requisitos, con pruebas que nos aseguren su correcto funcionamiento, hay que llegar a más allá.

En mayor o menor medida, la calidad en el software hace al código más mantenible y orientado hacia una buena lógica. Muchos problemas que se presentan al desarrollador son antiguos y han sido solucionados con creces. Se hace necesario, por tanto, usar dichos patrones, arquitecturas y metodología que recomiendan la comunidad para conseguir dicho perfeccionamiento.

Es necesario también cumplir alguna serie de estándares o llevar el estilo del código a un determinado modelo. Esto nos asegura ir en el buen camino dentro de un framework o lenguaje concreto. Esto, además, nos llevará a tener en cuenta cuestiones de seguridad, rendimiento o tamaño, que de otra manera no consideraríamos.

STATUS	PROJECT	LAST COMMIT	ISSUES
<input checked="" type="checkbox"/>	Itnews-backend GitHub / Public	Andrés M. Jiménez 3 months ago Add new IP to settings	0 NEW 0 FIXED 1 TOTAL
<input checked="" type="checkbox"/>	Itnews-frontend GitHub / Public	Andrés M. Jiménez 4 months ago draft #9: profile view	0 NEW 0 FIXED 16 TOTAL

Figura 7.2: Repositorios en Codacy

Este ha sido uno de los aspectos enfocados en la calidad que más se ha tenido en cuenta a la hora de realizar el proyecto. Para ello, se han utilizado dos herramientas principales: Optimize que provee los IDE de Jetbrains y Codacy, como sistema externo.

Del primero decir que, al igual que cualquier IDE, en PyCharm se nos muestran errores de código en tiempo de compilación, posibles errores en tiempo de ejecución, así como advertencias a la hora de usar una función obsoleta. Hasta aquí, no apreciamos ninguna novedad. Sin embargo, con esta herramienta, podremos ver qué recomienda Python para nombrar las variables, cómo realizar de manera correcta un try/except o cualquier función por defecto del software, así como muchísimas más características.

Otra herramienta utilizada es Codacy. Esta analiza cualquier repositorio Git por cada commit que detecte, mostrando tanto errores sintácticos, como problemas de seguridad que poseamos. Así, se han intentado mejorar los repositorios del proyecto hasta llegar a la máxima certificación.

Como se ha dicho, Codacy comprueba varios aspectos. Uno de ellos es la seguridad. En este nos dice si nuestro sistema posee precauciones sobre determinado tipo de ataques. Como se puede observar en la siguiente imagen, todo está correcto, debido a la potencia de los frameworks utilizados. En concreto, Django ofrece soporte sobre gran diversidad de vulnerabilidades.

Security monitor master ▾

Warnings

All ▾ More ▾

- ✓ Auth >
- ✓ Command Injection >
- ✓ Cryptography >
- ✓ File Access >
- ✓ HTTP >
- ✓ Input validation >
- ✓ Insecure modules/libraries >
- ✓ Other >
- ✓ SQL Injection >
- ✓ SSL >
- ✓ XSS >

Figura 7.3: Seguridad en Codacy

CAPÍTULO 8

Conclusión

8.1– Resultados

Como se ha podido comprobar a lo largo de la memoria, la metodología LEAN, en concreto, *Running Lean*, ha surtido efecto a la hora de enfocar el desarrollo de la aplicación. Creo que esto es debido a la antropología usada por dicha metodología. El centro del desarrollo ya no es el producto sino el cliente. Esto ha permitido testear desde el inicio la idea *sobre papel*, sin necesidad de construir nada. Esto conlleva la reducción de costes y mayor implicación por parte del equipo de desarrollo: *ponerse en los zapatos* del cliente.

Sin embargo, esto no es una acción puntual de un momento, como puede ser el estudio o análisis de la idea. Debe ser un proceso que vaya desde el inicio de la concreción de la idea hasta la búsqueda del feedback continuo ante cualquier cambio en la aplicación. Es por tanto un actitud a asimilar, no una serie de pasos a realizar. Creo que esto es lo más complicado de entender a la hora de aplicar la metodología. En mi caso personal, creo que lo he aprehendido gracias a la lectura detenida del libro de Ash Maurya.

Un resultado negativo a tener en cuenta es la valoración de los comentarios de los *early adopter*. En el caso de LT-News, me ha sido difícil diferenciar algunas características esenciales del Mínimo Producto Viable de las prescindibles, en base a aportaciones de los futuros usuarios. En aras de contentar a todos, creo que he caído en manos un servilismo por parte de dicho sector. Afortunadamente, esta actitud ha sido descubierta por mis tutores y he sabido podar correctamente en la fase de Estudio.

Resumidamente, creo que es un gran aporte la utilización de *Running Lean* para el estudio de las ideas de aplicación. Aporta conocimiento y enfoque diferentes al equipo de desarrollo. Sin embargo, creo que es imprescindible poseer a una persona que tutele el proceso para poder realizarlo correctamente y con objetividad.

8.2– Mejoras futuras

La aplicación construida es un Mínimo Producto Viable, que, de momento, no busca una monetización y unos rendimientos económicos. Analizando el proceso seguido y hacia donde puede llegar la aplicación distingo tres fases para realizar en un futurable. Antes de ir desgranando una a una, se hará un informe de la situación actual del proyecto, para poder encarrilar correctamente dichos cambios.

El sistema se ha intentado realizar siguiendo las pautas y herramientas que propone la comunidad del software, utilizando últimas versiones de dichas herramientas y una arquitectura novedosa y profesional. Por tanto, se han cimentado las bases sobre futuros desarrollos.

A corto plazo, en menos de un año, la aplicación habrá adquirido un pequeño volumen de usuarios y una cantidad, no tan pequeña, de noticias. En este punto, pienso que sería conveniente realizar los siguientes cambios:

- Empezar a integrar la información de las noticias con las redes sociales. Dado que este podría ser un requisito que se convierta en un mar sin orillas, se propone realizar tres acciones.
 - Integrar las cuentas de Twitter de los periódicos añadidos y relacionar noticias con tweets para analizar y usar el movimiento de comentarios que genera, así como añadir noticias de actualidad.
 - Añadir en cada noticia los botones de las principales redes sociales para poder distribuir fácilmente las noticias y hacer auto-publicidad de la plataforma.
 - Permitir el registro con las cuentas de las redes sociales, así como su integración posterior. De esta manera, publicar contenidos automáticamente en estas como respuesta a una noticia, por ejemplo.
- Otro aspecto importante debe ser la adición de la publicidad en la plataforma. Esta se deberá añadir intentando que esté integrada con el contenido de la aplicación sin ser molesta por su exceso a los usuarios. Además, se relacionará con Google Analytics para mostrar qué contenidos proveen mayor beneficios.
- Mostrar en la noticias información de contexto de la misma. Dado que esto puede ser muy amplio, se podría empezar integrando Wikipedia y mostrando en la misma noticia aquellos términos interesantes con la información que provea la enciclopedia.
- En este momento, será casi semanal el cambio que sufrirá el software. Para ahorrar tiempo, se proponer la realización de una plataforma de DevOps que facilite el desarrollo y recorte el *time to market*.

A medio plazo, en torno a tres años, y si la aplicación va siendo popular, aumentará el número de usuarios considerablemente. Por ello, en este momento, se proponen los siguientes cambios:

- En primer lugar, añadir un nuevo actor del sistema: analista. Este será, normalmente, un periodista de un medio o un redactor de un blog con un volumen medio de gente. Poseerá un acceso a un cuadro de mando de temas que le podrán hacer diferencial así como poder estudiar a sus competidores. El dashboard se caracterizará por las siguientes funciones.
 - Este se basará en las noticias que se vayan leyendo y el perfil de aquellos que interactúan con ella.
 - Estarán las noticias más leídas del tema en cuestión así como los medios de los que proceden estas.
 - Además, habrá una evolución en el tiempo tanto de la publicación de estas como de su popularidad por parte de los lectores.
 - Poseerá un sobrecoste añadido debido a su gran aportación.
- El flujo de feedback continuo ya poseerá un gran número de actores. Es por ello que ya no valdrán canales como el correo para la resolución de problemas o propuesta de nuevas características. En este momento, habrá que añadir un gestor de incidencias anexo a la aplicación, que permita un proceso centralizado, ordenado y visible.
- Al crecer tanto el número de usuarios, será necesario pensar en hacer la web más accesible a ellos. Es por ello que en dicho momento se planteará la posible realización de una aplicación híbrida. Esta dará enormes ventajas, entre ellas la sencillez de uso y una utilización mejor de las notificaciones.

- Dado que en este momento se estará popularizando la aplicación, será necesario ir marcando la diferencia con respecto a la competencia. Es por ello que se empezarán a estudiar técnicas avanzadas de NPL, como pueden ser el autoresumen y la traducción.
- Dado que van a ser cada vez mayores las características del portal sobre la competencia, se propone empezar en este momento un plan premium. Así, se podrá diferenciar las funcionalidades y dar aquellas mejores a los usuarios que paguen. Aun esto, es importante cuidar a los clientes que vienen gratuitamente a nuestra aplicación.

Por último, en un plazo aun mayor, de entre cinco a diez años, la aplicación será una de las referentes en el mercado, con una comunidad de usuarios en tres continentes y acogiendo a millones de usuarios. Será en este momento cuando se potenciarán tres características.

- En primer lugar, desarrollar un dashboard por periódico o medio que esté en la aplicación. Albergará los mismo datos que el cuadro de mando por temas con una característica más: temáticas más recurrentes. Además, se centrará en mayor medida en los tipos de lector. Este se ofrecerá a analistas de la aplicación a un coste mayor, dada su embergadura. Estará pensado en editores jefes de periódicos físicos interesados en aumentar su audiencia.
- En este punto, y como guinda al pastel del plan freemium, se mejorará el dashboard personal de cada usuario. Con esta medida, se hará ver a los lectores que son importantes y se potenciará el uso de la IA.
- Por último, se ofrecerá una API con algunos datos de la aplicación que se vean convenientes. Esta poseerá un coste que se estudiará en el momento. La complejidad fundamental de esto serán las implicaciones legales.

8.3– Lecciones aprendidas

Al margen de los resultados y conclusiones obtenidas a la aplicación de la metodología LEAN me gustaría concluir con algunos hechos que me han llevado a aprender con la puesta en práctica del presente Trabajo.

En primer lugar, indicar que la realización de este TFM me ha hecho madurar como ingeniero. Esto se debe a que el centro del trabajo no residía en el producto, sino en la metodología que se ha llevado a cabo para realizar el mismo. Como he ido diciendo en los puntos que venían al caso: me he dado cuenta de que importa más el cliente que el producto, no solo en teoría, sino en la praxis.

En un segundo punto, me gustaría hablar del reto que ha supuesto para mí, desde un punto de vista más tecnológico, la realización de la aplicación. Quería realizar una sistema que pudiese usarse en un entorno productivo real y creo que lo he conseguido. Para llegar a esto, he tenido que empaparme de muchas herramientas y tecnologías novedosas, donde había más ganas que referencias en la red. Esto me ha hecho madurar mucho como informático así como aprender a trabajar con gran parte de incertidumbre tecnológica.

Por último, creo que este tipo de proyectos necesita de un pequeño equipo para llevarse a cabo. Trabajar individualmente es bonito y agiliza ciertas tareas, pero en general, es arduo. No tanto por la acumulación de trabajo, sino por la falta de personas y lo que estas aportan: moral, sentimientos o puntos de vista. Veo cada vez más imprescindible en el mundo del software el calor humano que ofrece un equipo.

Referencias

- Ainhoa Lafuente. (2019). *Qué es el web scraping*. Descargado 2019.04.20, de <https://aukera.es/blog/web-scraping/>
- Ana Pérez Barredo. (2014). *España se queda sin google news*. Descargado 2019.02.24, de https://elpais.com/politica/2014/12/16/actualidad/1418718308_671454.html
- Claire Drumond. (2019). *¿qué es scrum?* Descargado 2019.03.24, de <https://es.atlassian.com/agile/scrum>
- de Trabajo e Inmigración, M. (2009). Resolución de 18 de marzo de 2009, de la dirección general de trabajo, por la que se registra y publica el xvi convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública. *Boletín Oficial del Estado*, 3, 28–32. Descargado de <https://www.boe.es/boe/dias/2009/04/04/pdfs/BOE-A-2009-5688.pdf>
- Ian Buckley. (2018). *6 reasons why python is the programming language of the future*. Descargado 2019.04.21, de <https://www.makeuseof.com/tag/python-language-future/>
- Mauyria, A. (2011). *Running lean*. O'Reilly.
- Miguh Ruiz. (2018). *Los 5 frameworks de javascript para frontend más usados en 2018*. Descargado 2019.04.23, de <https://openwebinars.net/blog/los-5-frameworks-de-javascript-para-frontend-mas-usados-en-2018/>
- Paul Bonduelle. (2018). *The data science behind recommendations in feedly*. Descargado 2019.02.24, de <https://blog.feedly.com/data-science-behind-recommendations-in-feedly/>
- Quentyn Kennemer. (2018). *What's new in google news at google i/o 2018*. Descargado 2019.02.24, de <https://www.androidcentral.com/whats-new-google-news-google-io-2018>
- Rana Negra. (2018). *¿cuántos sitios web hay en internet?* Descargado 2019.02.17, de <https://www.rananegra.es/blog/cuantos-sitios-web-hay-internet>
- The State of JavaScript. (2019). *The state of javascript 2018*. Descargado 2019.04.21, de <https://2018.stateofjs.com/introduction/>
- Waffle. (2019). *Farewell from waffle*. Descargado 2019.04.26, de <https://blog.waffle.io/farewell-from-waffle-%EF%B8%8F-794da4a72851>
- Wikipedia. (2018). *Flipboard*. Descargado 2019.02.24, de <https://es.wikipedia.org/wiki/Flipboard>