



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

Desarrollo de una herramienta software orientada a la tercera edad para aprender a usar un ordenador

Autor:

Víctor Morais Llahí

Grado en Ingeniería Informática

Especialidad en Ingeniería del software

Directora:

Cristina Gómez Seoane

Departamento de Ingeniería de Servicios y Sistema de Información ESSI

2021

Agradecimientos

Este proyecto no habría sido posible sin la ayuda de varias personas a las que quiero expresar mi gratitud y darle el reconocimiento que se merecen.

Ante todo, a mis padres, por educarme y transmitirme los valores que, en parte, me han hecho la persona que soy hoy, por su constante apoyo sean cuales sean mis decisiones y por estar siempre ahí.

A mi abuela, por su amor y su cariño y por ser la verdadera motivación por la cual realicé este proyecto.

A mi tutora, Cristina Gómez Seoane, por su guía, consejo y corrección a lo largo de estos últimos cuatro meses y por haberme mostrado tan bien en el mundo de la ingeniería del software como para acabar especializándome en ella.

A todos los compañeros que han aportado alegría y diversión a mis años universitarios y a todos aquellos docentes que me han enseñado bien.

Y, por último, pero no menos importante, al resto de los involucrados en este proyecto: encuestados, voluntarios y todas aquellas personas que me han prestado ayuda y consejo.

Resumen

Más allá de la gente que enferma día a día, de los muertos y de las fatales consecuencias socioeconómicas que sufrimos en todo el planeta a causa de la grave pandemia de COVID-19 que asola nuestras vidas, existen otras repercusiones que hemos ido descubriendo durante este último año. Hechos que siempre han estado presentes o, al menos, desde hace mucho tiempo. Uno de ellos es la terrible brecha digital entre aquellos que nacimos en plena revolución tecnológica, y aquellos que han tenido que adaptarse a ella.

Pero, ¿qué sucede con aquellos que no lo consiguen? ¿Han de conformarse? ¿Resignarse a la exclusión social todo porque la situación de emergencia sanitaria haga avanzar a marchas forzadas la transición digital?

Este proyecto tiene como principal objetivo evitar eso y, para ello, se pretende desarrollar una plataforma web que permita a la gente de avanzada edad aprender los conceptos básicos de un ordenador y los servicios que puede ofrecerle el uso de Internet mediante un sistema de lecciones sencillas y una dinámica basada en pistas. Dar los primeros pasos hacia una autosuficiencia tecnológica. En definitiva, la inclusión digital.

Resum

Més enllà de la gent que emmalalteix dia a dia, dels morts i de les fatals conseqüències socioeconòmiques que sofrim a tot el planeta a causa de la greu pandèmia de COVID-19 que assola les nostres vides, existeixen altres repercussions que hem anat descobrint durant aquest últim any. Fets que sempre han estat presents o, almenys, des de fa molt temps. Un d'ells és la terrible bretxa digital entre aquells que vam néixer en plena revolució tecnològica, i aquells que han hagut d'adaptar-se a ella.

Però, què succeeix amb aquells que no ho aconsegueixen? Han de conformar-se? Resignar-se a l'exclusió social tot perquè la situació d'emergència sanitària faci avançar a marxes forçades la transició digital?

Aquest projecte té com a principal objectiu evitar això desenvolupant una plataforma web que permeti a la gent d'avançada edat aprendre els conceptes bàsics d'un ordinador i els serveis que pot oferir-li l'ús d'Internet mitjançant un sistema de lliçons senzilles i una dinàmica basada en pistes. Fer els primers passos cap a una autosuficiència tecnològica. En definitiva, la inclusió digital.

Abstract

Beyond the people who get sick every day, the dead and the fatal socioeconomic consequences that we suffer in our planet due to the serious COVID-19 pandemic that devastate our lives, there are other repercussions that we have been discovering during the last year. Facts that have always been present or, at least, for a long time. One of them is the terrible digital divide between those of us who were born in the middle of the technological revolution, and those who have had to adapt to it.

But what happens to those who cannot adapt? Do they have to conform? Resigning oneself to social exclusion all because the health emergency situation forces the digital transition to advance at full speed?

The main objective of this project is to avoid that and, for this, it is intended to develop a web platform that allows elderly people to learn the basic concepts of a computer and services that the use of the Internet can offer you through a system of simple lessons and a dynamic based on clues. Take the first steps towards technological self-sufficiency. In short, digital inclusion.

Contenido

1. INTRODUCCIÓN	8
1.1 DESCRIPCIÓN DEL PROBLEMA	8
1.2 MOTIVACIONES	10
1.3 CONTENIDO DE LA MEMORIA	10
2. CONTEXTUALIZACIÓN	11
2.1 MARCO DEL PROYECTO	11
2.2 PARTES INTERESADAS	11
2.3 ESTADO DEL ARTE	13
2.4 CONCEPTOS PREVIOS	18
2.4.1 LA SENSACIÓN DE SER CAPAZ	18
2.4.2 PSICOLOGÍA APLICADA AL DISEÑO DE LA UI	21
3. ALCANCE	25
3.1 OBJETIVOS	25
3.2 OBSTÁCULOS Y RIESGOS	26
4. METODOLOGÍA Y RIGOR	28
4.1 METODOLOGÍA DE TRABAJO	28
4.2 HERRAMIENTAS Y RECURSOS PARA EL DESARROLLO	29
4.2.1 HERRAMIENTAS HARDWARE	29
4.2.2 HERRAMIENTAS SOFTWARE	29
4.2.3 HERRAMIENTAS DE SEGUIMIENTO Y CONTROL DE VERSIONES	30
4.3 RIGOR DE VALIDACIÓN	31
5. PLANIFICACIÓN TEMPORAL	31
5.1 DESCRIPCIÓN DE TAREAS	32
5.1.1 GESTIÓN DEL PROYECTO (60 HORAS)	32
5.1.2 DISEÑO Y ARQUITECTURA (150 HORAS)	32
5.1.3 IMPLEMENTACIÓN (175 HORAS)	33
5.1.4 REVISIÓN Y REFACTORIZACIÓN (52 HORAS)	35
5.1.5 DOCUMENTACIÓN Y COMUNICACIÓN	35
5.2 ESTIMACIÓN	36
5.3 DIAGRAMA DE GANTT	37
5.4 GESTIÓN DE RIESGOS	38
5.4.1 INEXPERIENCIA EN LAS TECNOLOGÍAS UTILIZADAS	38
5.4.2 FALTA DE VOLUNTARIOS	39

5.4.3 EL PAPEL DE OTRAS ASIGNATURAS Y UNA DATA DE ENTREGA FIJA	39
6. GESTIÓN ECONÓMICA	40
6.1 IDENTIFICACIÓN DE LOS COSTES	40
6.1.1 RECURSOS HUMANOS	40
6.1.2 RECURSOS HARDWARE	41
6.1.3 RECURSOS SOFTWARE	42
6.1.4 GASTOS GENERALES	42
6.1.5 GASTOS IMPREVISTOS Y DE CONTINGENCIA	43
6.2 CONTROL DE GESTIÓN	44
7. ANÁLISIS DE SOSTENIBILIDAD	45
7.1 DIMENSIÓN MEDIOAMBIENTAL	45
7.2 DIMENSIÓN ECONÓMICA	45
7.3 DIMENSIÓN SOCIAL	46
8. REQUISITOS	47
8.1 REQUISITOS FUNCIONALES	48
8.1.1 PROCESO DE RECOGIDA DE DATOS	48
8.1.2 DESCRIPCIÓN DE LOS REQUISITOS	49
8.2 REQUISITOS NO FUNCIONALES	53
8.2.1 APLICACIÓN DE LA PSICOLOGÍA APLICADA AL DISEÑO DE UI	53
8.2.2 DEFINICIÓN DE REQUISITOS NO FUNCIONALES	54
9. ESPECIFICACIÓN DE LOS REQUISITOS	58
9.1 DIAGRAMA DE CASOS DE USO	59
9.2 DESCRIPCIÓN DE LOS CASOS DE USO	61
9.3 MODELO CONCEPTUAL	69
9.3.1 ESQUEMA CONCEPTUAL DE LOS DATOS	69
9.3.2 RESTRICCIONES DE INTEGRIDAD	70
9.3.3 DESCRIPCIÓN DE LAS CLASES	70
9.4 MODELO DE COMPORTAMIENTO	72
10.DISEÑO DE LA PLATAFORMA WEB Y DE SU CONTENIDO	82
10.1 ARQUITECTURA LÓGICA	82
10.2 ARQUITECTURA FÍSICA	83
10.3 DESCRIPCIÓN INTERNA DE LAS CAPAS	85
10.3.1 DISEÑO DE LA CAPA DE PRESENTACIÓN	85
10.3.2 DISEÑO DE LA CAPA DE DOMINIO	88
10.3.3 DISEÑO DE LA CAPA DE DATOS Y DE LA CONEXIÓN AL SERVIDOR	89

11. IMPLEMENTACIÓN DEL SISTEMA	94
11.1 PROCESO DE APRENDIZAJE	94
11.2 IMPLEMENTACIÓN	95
11.2.1 UNA MEZCLA DE LENGUAJES, FRAMEWORKS Y LIBRERÍAS	95
11.2.2 ESTRUCTURA INTERNA	97
11.2.3 PERFECCIONAMIENTO POR ITERACIONES DE REFACTORIZACIÓN	106
11.3 DESPLIEGUE DEL SISTEMA	109
11.4 SOLUCIÓN DE PROBLEMAS	110
11.4.1 CREACIÓN DE UN COMPONENTE SCROLLToTOP	110
11.4.2 LA TRADUCCIÓN DE DATOS IMPORTADOS	111
11.4.3 LA INCOMPATIBILIDAD ENTRE ESTILOS PROPIOS Y AJENOS	111
11.4.4 LENTITUD DE HEROKU AL DESPERTAR LA APLICACIÓN	113
12. VALIDACIÓN DEL SISTEMA	114
12.1 CUMPLIMIENTO DE LOS REQUISITOS	114
12.1.1 VALIDACIÓN DE REQUISITOS FUNCIONALES	114
12.1.2 VALIDACIÓN DE REQUISITOS NO FUNCIONALES	121
12.2 VALORACIÓN SOBRE SU UTILIDAD	126
12.3 ANÁLISIS DE ALTERNATIVAS	127
12.4 IDENTIFICACIÓN DE LEYES Y REGULACIONES	128
13. SEGUIMIENTO DEL PROYECTO	130
14. CONCLUSIONES	131
14.1 CUMPLIMIENTO DE LOS OBJETIVOS DEL PROYECTO	131
14.2 LIMITACIONES Y DIFICULTADES	133
14.3 CONOCIMIENTOS APLICADOS Y ADQUIRIDOS	133
14.4 JUSTIFICACIÓN DE LAS COMPETENCIAS	134
14.5 FUTURO DEL PROYECTO	136
14.6 REFLEXIÓN FINAL	137
15. REFERENCIAS BIBLIOGRÁFICAS	138
ANEXOS	142
ANEXO 1	142
ANEXO 2	145
ANEXO 3	149

1. Introducción

1.1 Descripción del problema

Hasta hace relativamente poco, cosa de un año, considerábamos la transformación de servicios analógicos a digitales como una mejora. Como un capricho, podríamos incluso decir. Bastó una pandemia que restringiese nuestra movilidad y que estableciera el distanciamiento social como principal mecanismo de autodefensa para que dicha opción se convirtiera en una necesidad. Y es lógico, en realidad. Es más seguro hacer cualquier trámite bancario desde el móvil que desde una oficina. Es más seguro comprar por Internet que no en un abarrotado centro comercial. Y, por descontado, es más seguro restringir cualquier trámite a través de una pantalla que no cara a cara con un funcionario.

El problema surge cuando esta medida, como muchas otras, no vienen acompañadas de su correspondiente séquito de contramedidas y planes de contingencia. ¿La consecuencia? El incremento de la brecha digital entre los denominados nativos digitales, aquellos que hemos nacido, crecido y vivido en un mundo rodeado de constantes avances tecnológicos, y de los no tan afortunados inmigrantes digitales, aquellos que han tenido que adaptarse a este cambio y, en definitiva, a un nuevo mundo en el que vivir.

Asociamos la digitalización de nuestros servicios a la aparición de jugosos beneficios, pero no prestamos atención a las limitaciones que también comportan. Y si, muchas veces se mantienen los mecanismos convencionales para aquellos que no saben adaptarse al cambio, pero esta opción desaparece cuando la digitalización se convierte en una simple necesidad como es el caso.

En este proyecto se va a dar a conocer el verdadero problema que supone el digitalizar un mundo más por necesidad que por capacidad sin poner la debida atención a la inclusión digital de los más mayores y, no solo eso, se va a tratar de ponerle una solución para que la exclusión social no sea su única opción. Se va a desarrollar una herramienta digital que permita a la gente mayor aprender a usar otras herramientas digitales y que les brinde la oportunidad de empezar una transición hacia la autosuficiente tecnológica.

Pero, para poder solucionar el problema, hemos de conocerlo realmente. ¿Lo hacemos? De acuerdo, sí. El problema se reduce a que las personas mayores no saben usar las nuevas tecnologías, pero, ¿sabemos el motivo? Lo primero que pensará la inmensa mayoría es que una mente envejecida no posee la capacidad de aprender cosas nuevas. Y lo primero que hará este trabajo es desmentir esta idea. Es cierto que las facultades envejecen, pero con bastante motivación y una persistencia mínima el cambio es posible. Sin embargo, esto es solo la punta del iceberg. El verdadero problema está en que no es simplemente una cuestión de fuerza de voluntad, porque la brecha digital es alimentada por numerosos factores:

- **La ausencia de una buena instrucción** desanima a cualquiera de intentar algo nuevo. Es normal que alguien que sepa cómo manejar las nuevas tecnologías no empatice lo suficiente con aquel que las desconoce por completo y, al no ponerse en la piel del otro, solo haga que generar inseguridad y desconfianza hacia lo desconocido.
- **La desventaja** que ofrece el hecho de que la mayoría de herramientas software no estén correctamente adaptadas a las dificultades cognitivas y/o auditivas, así como el mermado de las capacidades físicas y/o mentales, que puede padecer una persona muy mayor.
- **La falta de recursos para aprender**, tanto tecnológicos, como humanos. Los cursos de manejo de ordenadores para la gente mayor y el confinamiento de personas de riesgo son dos elementos completamente incompatibles.
- **Las diferencias en la capacidad de asimilar conceptos.** Alguien mayor requiere de un proceso de aprendizaje continuo. Alguien joven se puede permitir interrupciones porque retiene datos con mayor facilidad. Esto mismo puede aplicarse a la rapidez mental.
- **El no conocer las expectativas** del que quiere aprender tiene más relevancia de lo que parece. Es gratificante satisfacer la razón por la que uno empieza a hacer algo nuevo, pero nos disuade de seguir intentándolo si nuestro objetivo no se cumple. Hemos de averiguar qué espera la gente de avanzada edad de las nuevas tecnologías.
- **La falta de predisposición** que se ha comentado antes. Esta puede ser debida por diversas razones como lo son el sentimiento de incapacidad, el resentimiento o tecnofobia¹, la comodidad que ofrece que alguien cercano use la tecnología en su lugar cuando lo precisa o, simplemente, la falta de motivación.

Como podemos observar, el problema es un puzle de muchas piezas y todas deben de tener el mismo peso e importancia si queremos hacer algo por remediarlo y esto es muy importante tenerlo en cuenta. Porque si lo hacemos, entendemos el problema y si entendemos el problema, podemos solucionarlo. Es por ello que el sistema a desarrollar enseñará conceptos sencillos e indispensables hoy en día para defenderse en la era tecnológica. Se estudiará a los potenciales usuarios para saber sus expectativas, se diseñará una plataforma web a partir de sus opiniones, se presentará el contenido de dos formas distintas, aprendizaje y desafío, de manera que el usuario podrá aprender algo por primera vez o retarse día a día para que el estudio sea continuado, y se intentará dar el empuje necesario para empezar a moverse en dirección a una inclusión digital. Empieza el proceso de creación de la aplicación web “yayOS”.

¹ **Tecnofobia:** aversión o miedo hacia las nuevas tecnologías o dispositivos complejos. Principalmente ordenadores

1.2 Motivaciones

Como hijo y como nieto he vivido en primera persona el problema que supone para un familiar mío la dependencia en otros en lo que al acceso y uso de tecnología se refiere. Una dependencia que ha aumentado considerablemente en tiempos de pandemia. La transición digital ya era rápida antes, pero permitía un cierto margen de adaptación. Pero, ¿ahora? El cambio ha sido de la noche a la mañana prácticamente y los medios para adaptarse se han visto reducidos con la misma rapidez. Y si este proyecto puede aportar un poco de ayuda y que esas personas no dependan tanto de terceros para usar un ordenador, habré cumplido mi objetivo.

Como estudiante de ingeniería informática y de la especialidad de ingeniería del software, quería aprovechar la ocasión que presenta un proyecto como el trabajo final de carrera para hacer algo de lo que sentirme orgulloso. De hacer algo que ha sido ideado por mí y no impuesto por obligación de una materia. Algo que considerara útil y que resolviera un problema real y no uno inventado en un enunciado.

Y, por último, como persona intento siempre superarme. Aprender algo nuevo u obtener una nueva habilidad y este proyecto me brinda la oportunidad de hacerlo porque, aunque algunos de los conocimientos técnicos para desarrollar la plataforma web no me son plenamente desconocidos, sí que me son, por lo menos, bastante distantes. La única asignatura que ha tocado elementos similares en toda la carrera ha sido Aplicaciones y Servicios Web (ASW) y ni siquiera ahí tuve la oportunidad de profundizar demasiado ya que eran proyectos en equipos donde cada uno está destinado a sus tareas concretas y donde siempre hay alguien con más experiencia que se encarga de hacer las cosas más complejas. Por lo que aprender dichos aspectos técnicos me abrirá más puertas en el momento de entrar en el mundo laboral.

1.3 Contenido de la memoria

En este documento se recogen todas las fases del proceso de desarrollo de un proyecto de ingeniería del software de principio a fin. Se empezará analizando su contexto y su alcance, luego la metodología elegida para llevarlo a cabo y, en consecuencia, la planificación temporal. Se estudiarán y especificarán los requisitos que tiene que cumplir el proyecto y, una vez se tenga toda esta información, se empezará el diseño que se irá volviendo realidad progresivamente en la fase de implementación. Una vez el proyecto esté acabado, se analizará el resultado obtenido y se concluirá si se han cumplido los objetivos previstos, los requisitos definidos y las competencias técnicas asociadas acorde con la normativa vigente del grado de Ingeniería Informática en la Facultad de Informática de Barcelona. Se analizará también la sostenibilidad del proyecto y se hará una estimación de los costes que supondría llevarlo a cabo en el mundo laboral de hoy en día. Por último, se mencionarán todas aquellas referencias tanto bibliográficas como webgráficas. Durante todo el documento se podrán ver notas a pie de página con aclaraciones de los términos que así lo requieran.

2. Contextualización

2.1 Marco del proyecto

Este proyecto de desarrollo de software se trata, en realidad, de un trabajo de final de grado para la Facultad de Informática de Barcelona y la Universidad Politécnica de Catalunya. Concretamente para la carrera de Ingeniería Informática y la especialidad de Ingeniería del Software. Es por este motivo que la creación de yayOS tiene como objetivo, aparte de los propios, demostrar las habilidades del estudiante en las competencias esperadas de su categoría y la del proyecto, además de demostrar sus conocimientos aprendidos durante la carrera, así como la capacidad de adquirir aquellos nuevos que le sean necesarios.

2.2 Partes interesadas

Una de las partes fundamentales en el desarrollo de cualquier proyecto es el correcto conocimiento, clasificación y análisis de los actores implicados en él, de sus stakeholders. Conocer las necesidades y expectativas de los involucrados es vital para una exitosa planificación y una correcta definición de los requisitos pues son los principales influyentes en el mismo.

A continuación, procederemos a describirlos detalladamente:

Usuarios

Podría decirse que es el grupo que más en contacto estará con la plataforma web. Es un grupo con un interés particular en las funcionalidades que ofrece y que debe tener acceso a un ordenador.

Objetivos:

- Aprender las lecciones de manejo de ordenador que se le presenta.
- Ponerse a prueba a sí mismo tratando de memorizar las tareas que se le enseñan.
- Que sus sugerencias, quejas o comentarios sean escuchados y que tengan algún valor.

Rol:

- Entrar en lecciones sobre el manejo básico de un ordenador en modo “aprendizaje” o en modo “desafío”.
- Publicar comentarios sobre las lecciones y puntuarlas según su utilidad.
- Enviar feedback sobre la plataforma para mejorarla.

Testers

Grupo reducido de voluntarios que probaran la aplicación bajo supervisión y de los que se estudiará su interacción con ella y las consecuencias de dicha interacción.

Objetivo:

- Comprobar si les es cómodo el uso de la plataforma web.
- Que sus aportaciones sirvan para mejorar la plataforma web.
- Mismos objetivos que los usuarios.

Rol:

- Interactuar con la aplicación como un usuario más.
- Aportar opiniones sobre la plataforma web.

Directora del proyecto

Docente a cargo de la supervisión de la correcta realización del TFG. Aparte de los intereses personales, tiene una serie de obligaciones según la normativa vigente en el grado de Ingeniería Informática Facultad de Informática de Barcelona.

Objetivo:

- Que el TFG se adecue a las especificaciones pertinentes
- Que la realización y resultados del TFG sean satisfactorias.
- Ayudar en la medida de lo posible al alumno que lleva a cabo el TFG.

Rol:

- Asesorar al alumno en todo aquello relativo a el desarrollo de las competencias técnicas y transversales del proyecto.
- Realizar el seguimiento y monitorización del proyecto.
- Supervisar el trabajo realizado.

Desarrollador

Persona encargada del desarrollo de este proyecto. Al formar parte de un TFG individual, el desarrollador engloba un conjunto de roles que en otro tipo de proyectos irán separados como lo son el de ingeniero de requisitos, diseñador, programador...

Objetivo:

- Aplicar los conceptos teóricos aprendidos durante el transcurso de la carrera.
- Aprender todos aquellos conocimientos nuevos necesarios para la realización del proyecto.
- Hacer uso de todos sus esfuerzos para hacerlo lo mejor posible.
- Demostrar su habilidad en las competencias esperadas de un Ingeniero Informático de la Facultad de Informática de Barcelona especializado en el desarrollo de software.

Rol:

- Recabar toda la información necesaria para el desarrollo del proyecto.
- Elaborar la plataforma web especificada a partir de las opiniones recibidas y la memoria explicativa que forman este proyecto.
- Mantener una comunicación continua con su directora y asimilar sus aportaciones.

2.3 Estado del arte

Durante el desarrollo de cualquier proyecto se suele acostumbrar, en la fase más inicial, a realizar un exhaustivo estudio de lo que se conoce como el estado del arte del ámbito o entorno de nuestro producto. En otras palabras, se suele estudiar la competencia directa en el mercado de aquello que queremos implementar. Conocer qué funcionalidades ofrecen los productos ya existentes en nuestro campo nos permite saber en qué podemos diferenciarnos y dar un valor añadido que nos distinga del resto, además de inspirarnos y ver otros puntos de vista posibles para nuestra idea. Así pues, se ha seguido este mismo proceso para este caso y se ha investigado el mercado para saber, en la actualidad, que se está haciendo para ayudar a la población de edad avanzada a adentrarse en el mundo tecnológico.

La lucha por la inclusión digital de nuestros mayores tiene un nombre. La gerontecnología [1] es un campo de investigación tanto académico como profesional que engloba diversas disciplinas como la biología, la psicología, la sociología y la geriatría. Este concepto es la suma del estudio de los aspectos socioculturales y psicológicos del envejecimiento, lo que se conoce como “gerontología”, y del conjunto de métodos, procesos y habilidades usados en la producción de bienes y servicios que conocemos como “tecnología”. La gerontecnología es, por tanto, la aplicación de la tecnología a la gente de avanzada edad.

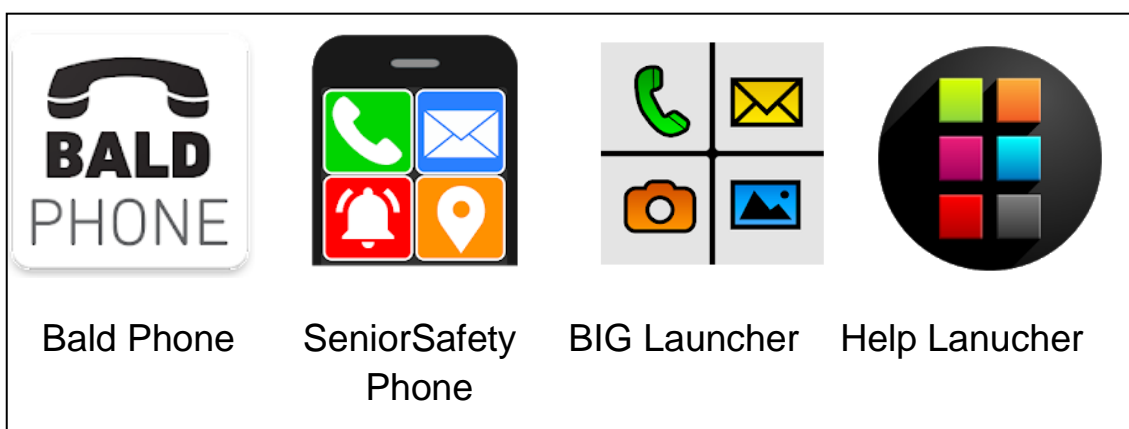
El hecho de que exista este campo de investigación ha permitido en los últimos años demostrar el verdadero peligro de la brecha digital entre nativos e inmigrantes digitales y ha impulsado centenares de proyectos de asistencia tecnológica para reducirla.

Muchos de ellos, los más recientes, son exclusivos para dispositivos móviles. De hecho, hay mucha más oferta para ese tipo de plataforma que no para ordenadores. Después de todo es comprensible. Hoy en día un smartphone es como un ordenador de bolsillo y sus funcionalidades son prácticamente las mismas salvo que con restricciones tal vez a nivel de hardware. Aunque eso no es un impedimento ni mucho menos. Entonces, ¿por qué no desarrollar una aplicación para móviles? Se ha elegido hacer la plataforma para ordenadores por varias razones. La primera es que el sector de la telefonía está mucho más avanzado en este aspecto por lo que resulta interesante explorar otras vías menos adaptadas. Segundo, porque el autor cree que es más cómodo aprender desde la comodidad que ofrecen la pantalla y el teclado de un ordenador pues son más grandes. Por no hablar de que les será más manejable un ratón que no una pantalla táctil. Y, por último, porque a largo plazo darán pie a más oportunidades. Aunque aún no hemos entrado en detalle en los objetivos de este proyecto, uno de ellos es que el

usuario, aparte de aprender, pierda el miedo al ordenador. Brindarle la curiosidad suficiente como para que explore por su cuenta. En ese sentido, un ordenador ofrece una gama de opciones muchísimo mayor que la de un teléfono móvil.

Brevemente se mencionarán algunas aplicaciones de móvil que sirven, entre otras cosas, para adaptar las interfaces de los teléfonos con el fin de hacerlas más accesibles para las personas mayores. Hay una cantidad gigantesca de ejemplos: BIG Launcher, Senior Safety Phone, Help Launcher, Bald Phone, Grand Launcher...

Básicamente muestran una UI con accesos directos a las funcionalidades esenciales de un teléfono móvil de manera que sea fácil de ver y de utilizar.



*Figura 1: Ejemplos aplicaciones para móvil adaptadas a personas mayores.
Fuente: Elaboración propia a partir de los resultados encontrados en Google Play.*

Otras aplicaciones interesantes que se han encontrado son algunas dedicadas a aumentar el volumen del audio de nuestro teléfono, el tamaño de las letras de los mensajes o incluso el teclado. Y, fuera de la adaptación, encontramos apps dedicadas a todo tipo de ámbitos. Desde algunas que te animan a hacer ejercicios mentales sencillos, otras que te recuerdan cuando tomarte la medicación y otras que te ayudan a entrenar tu cerebro.

Como se ha dicho, el mercado de las aplicaciones para dispositivos móviles es muy extenso y, aunque ha estado bien ver las propuestas de aquellas que facilitan la adaptación de sus interfaces, no persiguen los mismos propósitos que este proyecto.

Es por ello que, ahora sí, vamos a ver qué opciones hay disponibles para ordenadores. Se verá un ejemplo de cada tipo de herramienta y, aunque no todas tienen una implicación directa con la enseñanza del uso de un ordenador, se han incluido por alguna razón en particular que será comentada en la columna de conclusiones.

Eldy [2]



Figura 2: Pantalla de Eldy. Fuente: <http://www.eldy.es/>

“Eldy” es un software gratuito multiplataforma implementado por Eldy Asociación, una organización italiana sin ánimo de lucro que promueve la inclusión social y el envejecimiento activo.

Esta herramienta es similar a las que se han visto para móviles. De nuevo, adapta el sistema operativo y configura la interfaz acorde a las necesidades de una persona mayor de manera que la vuelve más intuitiva y cómoda de usar.

Vedoque [3]



Figura 3: Pantalla de Vedoque. Fuente: <https://vedoque.com/>

Es una propuesta que puede sorprender pues se trata de una página web de juegos infantiles. Pero creo que merece la pena mencionarla porque son muchas las páginas web que enseñan a usar un ordenador a personas mayores que recomiendan plataformas de este tipo para que los usuarios ganen agilidad con el uso del ratón y del teclado. De hecho, es una práctica extendida el recomendar estos sitios web de juegos sencillos.

skillfullsenior [4]

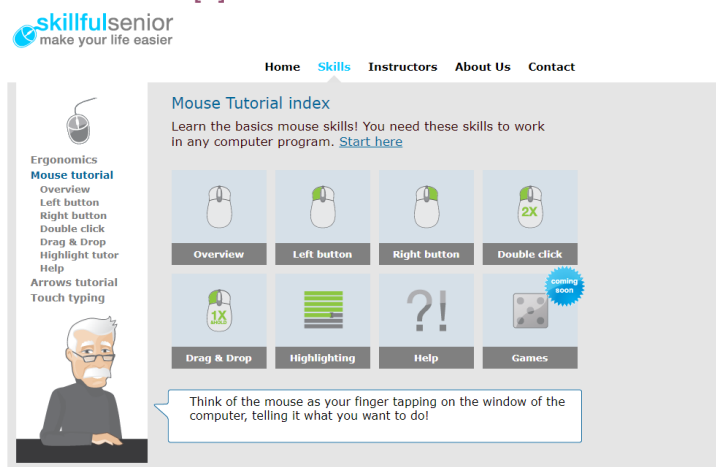


Figura 4: Pantalla de skillfullsenior. Fuente: <https://www.skillfullsenior.com/>

“skillfullsenior” es, por el momento, la propuesta que más se acerca a este proyecto pues es una plataforma web repleta de tutoriales y ejercicios que enseñan cómo usar el ratón y el teclado. Además, tiene un foro donde los usuarios pueden enviarse mensajes y ayudarse los unos a los otros. El único inconveniente es que los juegos interactivos usan Flash Player y, debido a que este ya no está disponible en la red, ya no son accesibles.

GCF Global [5]

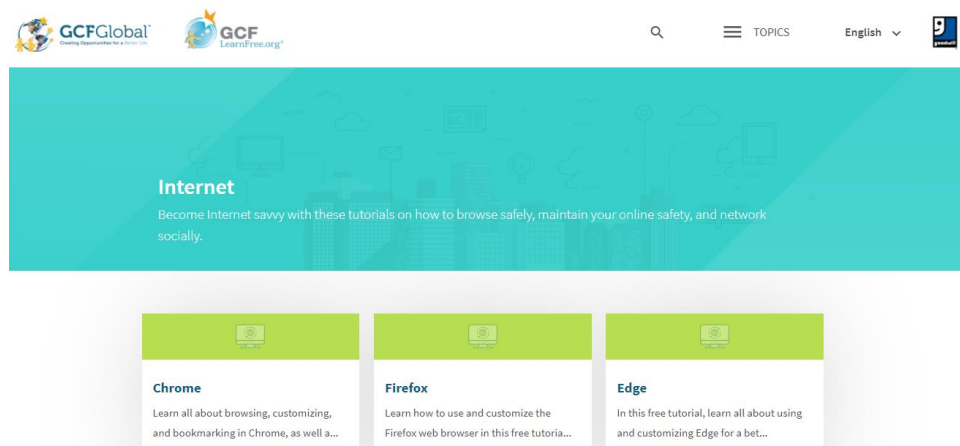


Figura 5: Pantalla de GCF Global. Fuente: <https://edu.gcfglobal.org/es/topics/informatica-e-internet/>

Otro ejemplo de plataforma web es “GCF Global”. Igual que “skillfullsenior” enseña los conceptos más esenciales de un ordenador. En esta ocasión las lecciones van más allá y se centran en el uso de los distintos motores de búsqueda y en consejos para encontrar aquello que queremos en la red de forma rápida y segura. Sin embargo, no es un ejemplo de plataforma adaptada para que la usen personas mayores puesto que el acceso a las explicaciones requiere de conocimientos básicos como manejar menús desplegables, visualizar videos en ventanas de reproducción o saber el significado de ciertos términos a los que solo los nativos digitales están acostumbrados.

Meganga [6]

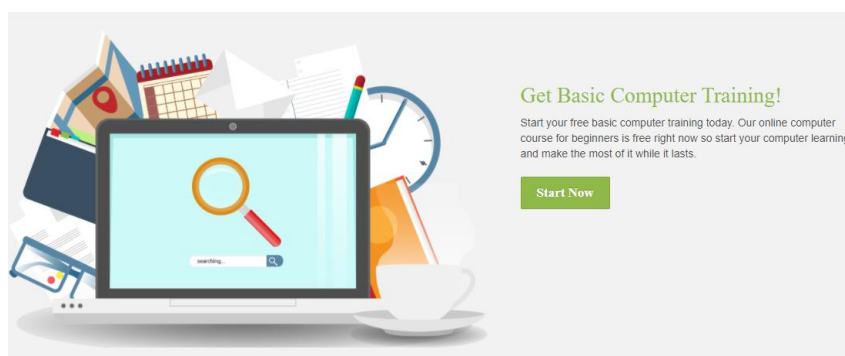


Figura 6: Pantalla de Meganga. Fuente: <https://www.meganga.com/lessons/>

“Meganga” es el tercer ejemplo de plataforma web. En su caso ofrece un abanico mucho más amplio y variado de opciones que las otras dos y también explica conceptos más avanzados sobre el hardware de la computadora. La distribución del contenido es similar a la de GCF Global, excepto que en su caso está indexado por lo que probablemente sea más complejo para alguien mayor acceder mediante el mismo.

Tutoriales de YouTube, Manuales, Cursos...

Por último, no podemos olvidarnos de la infinidad de material que podemos encontrar en la red. Existen canales de YouTube repletos de vídeos enfocados exclusivamente a la enseñanza a personas de edad avanzada, libros y manuales, cursos presenciales u online... Cada uno tiene sus ventajas e inconvenientes por supuesto, pero, en general, implican algo añadido como lo es saber manejar una plataforma de la red o pagar y asistir a un curso presencial. Sin embargo, el vocabulario utilizado para enseñar ciertos aspectos puede ser aplicada a yayOS.

En definitiva, no se ha logrado encontrar realmente un ejemplo de lo que este proyecto quiere hacer. Las tres opciones de plataforma web analizadas cumplen en cuanto a propósito, pero no en cuanto a especificaciones por lo que, a día de hoy no existe ninguna propuesta que impulse el aprendizaje del uso de un ordenador a gente mayor mediante el mismo. Un concepto un tanto paradójico en realidad, pero genuino. La plataforma web a implementar deberá ser un cúmulo de los aspectos positivos que hemos ido mencionando de estas alternativas.

	Apps para el móvil	Eldy	Vedoque	Skill Full Senior	GCF Global	Meganga	Contenido en la red	Este proyecto
Adaptado para personas mayores	✓	✓		✓	✓			✓
Accesible mediante ordenador		✓	✓	✓	✓	✓	✓	✓
Explica cómo usar el teclado y el ratón				✓			✓	✓
Muestra lo básico que se puede hacer en la red					✓	✓	✓	✓
Explica cómo acceder y usar lo que muestra						✓	✓	✓
Actualizado y funcional a día de hoy	✓				✓	✓		✓

Figura 7: Tabla que indica las cualidades que poseen los productos más parecidos a yayOS.

Fuente: Elaboración propia.

2.4 Conceptos previos

2.4.1 La sensación de ser capaz

No hace mucho, cuando hablábamos de las causas que forman la brecha digital, se ha hecho bastante hincapié en la complejidad que presenta el problema a resolver. Y es que la tiene realmente, porque nos planteamos diseñar una plataforma web cuya interfaz y funcionalidades han de presentar unas características muy concretas para que sea intuitiva, cómoda y accesible a un sector de la población que no está, para nada, acostumbrado a su uso. Podríamos decir que los detalles son clave en el momento de definir los requisitos no funcionales. Y para ello, lo mejor es conocer a los enemigos a los que nos tendremos que enfrentar.

Cuando el autor tuvo la idea de realizar este proyecto, lo primero que se le dijo fue: “¿Y cómo pretendes que una persona mayor aprenda esto? Has de tener en cuenta que, por la edad que tienen, no pueden aprender las cosas tan rápido como alguien joven como tú”.

Y pensamos eso porque, en parte, es verdad. Todos estaremos de acuerdo en que el ser humano posee mayor capacidad de aprendizaje en la etapa más temprana de su vida. En los tres primeros años concretamente. A partir de ahí se podría decir que empieza un cierto declive en dicha capacidad que no es perceptible hasta edades más adultas. Dicho declive se ve afectado por numerosos factores, por descontado. Pero no entraremos en detalles.

La neurogénesis² sigue un patrón bastante similar al de la capacidad de aprendizaje y recientemente se ha demostrado la existencia de una neurogénesis adulta, o también llamada neurogénesis postnatal, que permite que a nuestro cuerpo seguir formando nuevas neuronas incluso en edades adultas. Aunque es incierto que esto tenga un impacto en la capacidad de aprendizaje y hoy en día se está investigando, son muchos los estudios que afirman que un incremento en la neurogénesis, incluso en edades avanzadas, produce una mejora en los procesos cognitivos, así como en el aprendizaje y en la formación de memoria.

Es más, William James³ ya introdujo en 1980 en su obra “The principles of psychology” [7] el concepto de plasticidad cerebral, refiriéndose con ese término a la capacidad de nuestro cerebro de moldearse a sí mismo y reestructurarse cuando nos formamos, adquirimos una nueva habilidad o sufrimos influencias interpersonales. De nuevo, no profundizaremos en el tema porque se aleja del propósito de este proyecto, pero en lo que a William James se refiere concluimos que toda materia orgánica, como nuestro

² **Neurogénesis:** proceso de nacimiento de nuevas células a partir de células madre y células progenitoras

³ **William James:** prestigioso filósofo y psicólogo de la Universidad de Harvard conocido como el fundador de la psicología funcional.

cerebro, está dotada de una plasticidad que le permite moldearse al cambio y sobrevivir a él, por lo que puede preservar su integridad a lo largo de su existencia. Incluso en la vejez. Esto es gracias a que lo que somos se almacena dentro de nuestra cabeza no en las mencionadas neuronas, si no en sus conexiones con otras. La creación de nuevas neuronas implica nuevas asociaciones, lo que se traduce en nuevos lugares donde almacenar esa información. Dichas conexiones son denominadas como sinapsis. Y esta reestructuración de la que habla Williams es la habilidad de cambiar la sinapsis entre neuronas. De, como él dice, modificar nuestro mapa neuronal, lo cual da lugar a la introducción de nuevos comportamientos o a su modificación. Por lo que en ningún momento debemos descartar el hecho de que alguien mayor no pueda aprender nada nuevo.

Pero, si esto es cierto y el cerebro puede sobrevivir al cambio en cualquier momento ya sea gracias o no a la neurogénesis adulta, ¿por qué cuesta tanto? Esa es una pregunta muy interesante y su respuesta lo es aún más. No todas las neuronas tienen la misma capacidad de cambiar. O, dicho de otra manera. Hay estructuras neuronales muy estables y que se denominan de función plena, como aquellas que se encargan de nuestras funciones vitales, por ejemplo. Estas estructuras son muy complejas y prácticamente imposibles de modificar y es, a partir de ellas, que se van ramificando el resto. Se dice que las distintas estructuras neuronales maduran asincrónicamente, por lo que el cambio de sinapsis en algunas es más sencillo que en otras. El hábito que queramos introducir o cambiar nos costará más o menos según la estructura neuronal que se vea afectada. Y esto implica que el cerebro dispondrá de más o de menos herramientas para oponerse a dicho cambio. Porque, por lo general, a nadie le gusta modificar su rutina. Aquello que denominamos como zona de confort. Nuestro cerebro, que parte con ventaja porque nos conoce perfectamente, buscará nuestros puntos débiles para que excusemos ese cambio y no lleguemos a producirlo. Y, cuanto más mayor se es, más maduras son nuestras estructuras neuronales y menos conexiones nuevas se producen por lo que más costosa es esa lucha que conocemos como fuerza de voluntad.

Obviamente cada persona es un mundo y esto es una mera generalización de la conducta humana. Y sí, hay muchos factores externos que intervienen. Pero eso no significa que el cambio no sea posible. Como mucho, difícil. ¿Cuán difícil? Bueno, Henry W. W. Potts⁴ y Jane Wardle⁵ realizaron en 2009 un estudio titulado *“How are habits formed: Modelling habit formation in the real world”* [8] y que, entre otras cosas, pretendía dar una respuesta a esta pregunta.

Su objetivo era demostrar que la repetición de un comportamiento en un contexto consistente y de manera continua generaba el hábito. Por lo que cualquier hábito, incluso el de aprender a hacer cosas básicas con un ordenador, puede conseguirse mediante la repetición de dicho acto de manera que acabe desarrollándose de una

⁴ **Henry W.W. Potts:** investigador de la salud digital de la University College London.

⁵ **Jane Wardle:** profesora de psicología clínica de la University College London.

forma eficiente, inintencionada y sin que requiera demasiado esfuerzo en comparación con el que requería al principio.

Para ello, seleccionaron a un grupo de 96 estudiantes universitarios entre 21 y 45 años (30 hombres y 66 mujeres) y les hicieron elegir un hábito saludable que quisieran introducir en sus vidas y un momento del día que les sirviera de “señal”, o de recordatorio, para hacerlo, de manera que la repetición fuera cada día. Basándose en el trabajo de Clark L. Hull⁶ y en los diversos estudios que publicó entre 1943 y 1951, afirmaban que la transformación de un comportamiento a hábito no era un proceso lineal como se había pensado anteriormente, si no que más bien tomaba la forma de una curva asintótica por lo que las primeras iteraciones son las más significativas hasta llegar a un punto donde la repetición del comportamiento no aporta nada a su conversión a hábito.

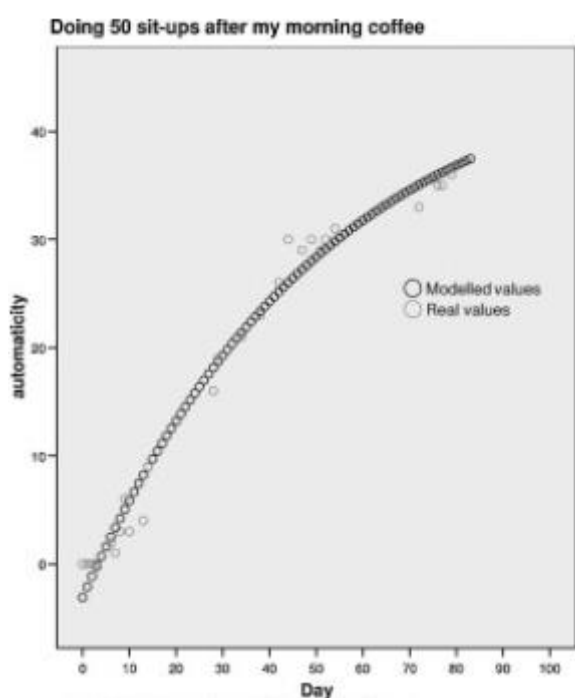


Figura 8: gráfica que representa la diferencia de la progresión de la automaticidad de un hábito entre el modelo esperado y el modelo real. Fuente:

<https://centrespringmd.com/docs/How%20Habit%20are%20Formed.pdf>

A diario les pedían que indicaran, en base a diversos factores, del 0 al 42 cuán automático les había resultado hacer esa actividad en concreto. En la gráfica adjunta en la Figura 8 podemos observar como la diferencia entre el modelo esperado y el modelo real obtenido a partir de los datos de los participantes no varía demasiado. Esta es una media de aquellos participantes cuya intención era acostumbrarse a hacer 50 abdominales después del café de la mañana.

El estudio concluyó que la media de días necesarios para convertir un comportamiento en un hábito era de 66, habiendo tardado 18 días el que menos y 124 el que más. Esta media se hizo teniendo en consideración no solo todos los participantes sino también todos los hábitos aprendidos, por lo que podríamos aspirar a un rango de tiempo similar para nuestro propósito, ciertamente.

En el estudio también observaron las consecuencias que tenía la interrupción del hábito en algún participante por la causa que fuera. Concluyen que no afecta demasiado, con una disminución de 0.29 puntos a 0.55 en la progresión de ese día en concreto. Por lo

⁶ **Clark L. Hull:** prestigioso psicólogo estadounidense centrado en el estudio del aprendizaje y las leyes científicas del comportamiento.

que, aunque es importante que el uso de la plataforma sea continuo al principio para que a la persona mayor le resulte más fácil aprender, se puede permitir algún que otro día de descanso.

De nuevo, insisto, esto depende al fin y al cabo de cada uno y, como podemos observar, hay una diferencia de 106 días prácticamente. En realidad, lo importante es darnos tiempo a nosotros mismos para cambiar. Independientemente de cuánto tardemos. Y, aunque decirlo es fácil, la mentalidad de nuestra sociedad preferirá mil veces antes una recompensa inmediata que la promesa de una recompensa mejor en el futuro. Nos apresuramos a desistir, y eso es lo primero que tenemos que cambiar para tener realmente éxito en nuestros propósitos. ¿Serán 24 días?, ¿serán 60?, ¿serán 200? Dependerá de nuestra salud física, nuestra salud mental, nuestro interés particular, nuestra motivación, nuestro entorno y de un sin fin de factores más. Sin embargo, sería interesante probar la plataforma web con gente real y ver en un plazo de 66 días si se ha logrado convertir ese conocimiento nuevo en hábito o no.

Por tanto, concluimos que el cambio no solo es posible si no que ahora sabemos que todo es, simplemente, cuestión de tiempo.

2.4.2 Psicología aplicada al diseño de la UI⁷

Hasta ahora hemos estado hablando solamente de una parte: la del usuario. Hemos comprendido que el cambio no es imposible pero que puede resultar costoso y hemos visto el papel que juegan la continuidad en el aprendizaje y la insistencia de uno mismo. Pero aún quedan algunas consideraciones que no hemos atacado y, recordemos, dijimos que todas tienen el mismo peso e importancia.

Supongamos que conseguimos romper las barreras del miedo, del “yo no puedo” y de la falta de fuerza de la voluntad y logramos que ese usuario de edad avanzada tenga la intención de utilizar la plataforma. Seguimos teniendo el problema de que queremos crear una interfaz para gente no acostumbrada a su uso. Es por ello que tendremos que centrarnos en el concepto de psicología aplicada al diseño de UI, pero teniendo en cuenta una serie de especificaciones de cara a un sector de la población en concreto que es el de las personas de edad avanzada.

Los principales efectos psicológicos que detallaré a continuación me fueron explicados por primera vez en la carrera cuando cursaba la asignatura de Interacción y Diseño de Interfaces (IDI). Por otro lado, se seguirán las indicaciones de compañías y grupos de diseño como UX Studio [9], F5 Studio [10] o Dexing [11] para adaptar la interfaz y volverla más accesible para el arquetipo de usuario que nos interesa.

⁷ **UI:** Siglas en inglés de User Interface. Entendemos por User Interface, o en español Interfaz de Usuario, el medio por el cual un usuario se comunica con una máquina, equipo, computadora o dispositivo.

En primer lugar, necesitamos una base con la que trabajar y, lo bueno que tienen las bases, es que sirven para cualquier tipo de persona. Y nuestra base es que queremos proporcionar al usuario la mejor experiencia de interacción. Para ello, necesitamos saber cómo funcionan sus procesos cognitivos y conductuales. Es por ello que el papel de la mente humana vuelve a ser clave. A continuación, hablaremos de aquellos efectos que experimentamos y que nos ayudarán a realizar un diseño estratégico que nos permita establecer los objetivos que nos interesan a los usuarios. De ejemplos hay muchos, pero aquí solo encontrarán aquellos que aprovechemos para nuestros propósitos.

Efecto Von Restorff

El efecto Von Restorff es un claro ejemplo de herramienta que nos permite desviar la atención del usuario allá donde nosotros queramos. Y esto sucede porque frente a varios objetos similares, nuestra atención será irremediablemente llamada por aquel que difiera del resto.

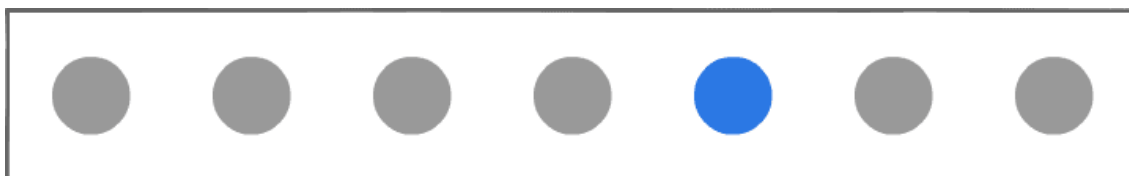


Figura 9: Ejemplo del efecto Von Restorff.

Fuente: <https://www.aunitz.net/ley-09-efecto-von-restorff/>

Esto nos puede ser útil en nuestro caso. Si por ejemplo queremos destacar algún punto en concreto de una lección o de alguna parte en particular de la plataforma, basta con hacerlo destacar del resto para que el usuario fije su atención en él, igual que cualquier persona que ha visto la imagen de arriba se ha fijado rápidamente en el punto de color azul.

Ley de Hick

La Ley de Hick pretende advertirnos de que dar muchas opciones a un usuario es una mala decisión como diseñadores. Está bien ofrecer muchas oportunidades, pero no si las ofrecemos todas a la vez porque producimos un proceso de parálisis decisiva en el usuario, lo que implica un aumento del tiempo que tarda en cumplir su objetivo y lo que repercute en la eficiencia de nuestro diseño. Un ejemplo de esto es lo que hacía la propuesta de Meganga.

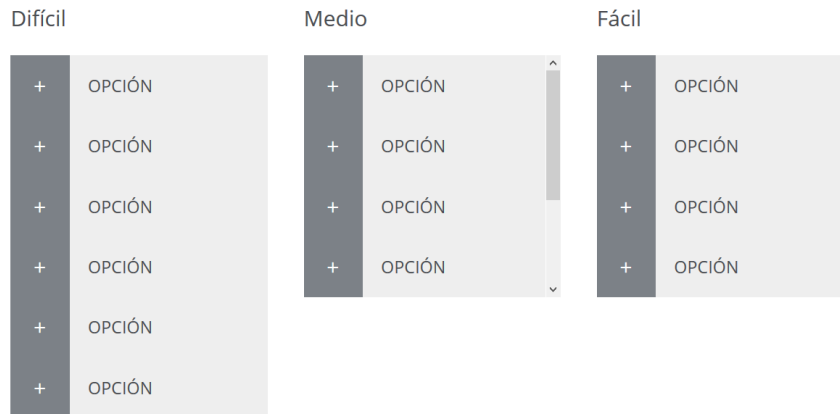


Figura 10: Ejemplo de la Ley de Hick.

Fuente: <https://cosasdigitales.com/usabilidad-ux/principios-psicologicos-interfaces-digitales/>

Además, este problema empeora cuando tratamos con gente mayor ya que sus capacidades de percepción pueden ser peores que las de alguien más joven. Es por ello que lo mejor es reducir las opciones a lo esencial o, en el caso de no ser posible, tratar de limitar el número de opciones que podemos ver a la vez. Aunque es muy poco probable que esta última medida mejore en algo la situación para las personas mayores.

Ley de Fitts

La Ley de Fitts tiene en consideración la relación logarítmica entre la superficie que ocupan los elementos de nuestra interfaz y la distancia a la que se encuentran del usuario. Esto podríamos decir que permite modificar la capacidad del usuario de percibir los elementos y resaltar de nuevo aquellos que nos interesa o que le pueden ser más útiles en un momento determinado.

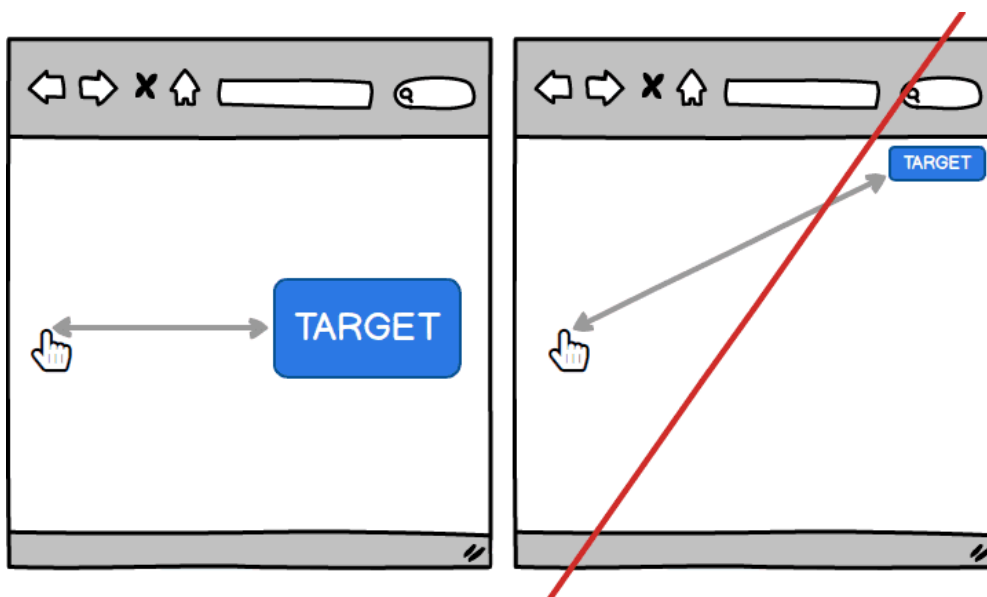


Figura 11: Ejemplo de la Ley de Fitts.

Fuente: <https://www.aunitz.net/ley-01-ley-de-fitts/>

Hay recalcar que, aunque la Ley de Fitts es importante y que, por supuesto, se ha de tener en consideración durante el desarrollo de este proyecto, su efecto no será tan provechoso en este caso por dos razones. Primero, recordemos que hablamos de usuarios poco o nada acostumbrados al uso de interfaces por lo que es un mundo nuevo para ellos. Más adelante hablaremos de cuáles son sus prioridades realmente cuando observan la pantalla de un ordenador o teléfono móvil, pero la distancia entre los elementos no es la principal. Y, segundo, requerimos que los elementos de nuestra interfaz sean, por lo general, bastante grandes. Volvemos a tener el problema del deterioro de las capacidades cognitivas. También entraremos en detalle más adelante.

Leyes de Gestalt

El último punto que trataremos serán las Leyes de Gestalt. Básicamente son un conjunto de directrices o reglas particulares sobre cómo estimular la percepción de los elementos. No vamos a mencionarlas todas porque son demasiadas y, ciertamente, la mayoría no tienen un interés particular para nuestros objetivos, aunque forman parte de cómo nuestra mente trabaja para percibir aquello que vemos. Se detallan a continuación aquellas a las que hay que prestar una especial atención cuando hablamos de usuarios de tercera edad:

- **Principio de semejanza:** Tendemos a agrupar aquellos elementos con formas similares, aunque estén aislados los unos de los otros.
- **Principio de simetría:** Tendemos a agrupar aquellos elementos con la misma simetría, aunque estén aislados los unos de los otros.
- **Principio de continuidad:** Tendemos a agrupar aquellos elementos que siguen un mismo patrón o dirección y a considerarlos como una misma figura.
- **Principio de simplicidad:** Tendemos a organizar nuestros campos perceptuales con rasgos simples, regulares y con formas sencillas.
- **Ley del contraste:** Tendemos a percibir mejor aquellas formas que contrastan más con el fondo.
- **Ley de la memoria:** Una forma es mejor percibida cuantas más veces haya sido vista.

Todos los conceptos introducidos hasta el momento suelen ser usados normalmente en cualquier tipo de diseño de interfaz. Sin embargo, como ya se ha adelantado antes, hay detalles que varían relativamente cuando nos centramos exclusivamente en la gente mayor ya que no siguen el mismo procedimiento metódico que sigue un nativo digital. A fin de cuentas, los usuarios de edad avanzada son más susceptibles a una experiencia negativa que les disuada de seguir usando esa interfaz en concreto. Quizás sea por desconfianza, por miedo o por falta de práctica. Don Norman⁸, todo un experto en la materia, recalca en su ensayo “Emotional Design: Why We Love (Or Hate) Everyday

⁸ **Don Norman:** profesor emérito de ciencia cognitiva en la Universidad de California y profesor de Ciencias de la computación en la Universidad de Northwestern.

Things” [12] la importancia que tiene el efecto emocional en todo aquello que diseñamos. Incluido una plataforma web.

Lo que hacemos tiene un efecto no solo cognitivo sino afectivo en nosotros mismos. Si esta consecuencia afectiva es positiva nuestro instinto primario se relaja, lo que nos permite ser más curiosos, más creativos y genera un entorno cerebral más dispuesto al aprendizaje. Pero, por el contrario, si el efecto es negativo nuestro cuerpo recurre a su instinto más básico: huir. Y para huir, solo hay que hacer clic en una cruz.

Norman asegura que en un usuario mayor las emociones, el Look & Feel⁹ de nuestro producto, juega un papel mucho más importante que el que juega para un usuario joven. Nuestras prioridades son distintas. Mientras que nosotros queremos cumplir el objetivo de una forma rápida, un usuario mayor prefiere fijarse en los detalles que se muestran y en toda la información percibida. Es de suma importancia procurar crear una conexión emocional con el usuario mayor para que se relaje y use la plataforma web. Y más si, en nuestro caso, queremos que esté dispuesto a aprender del contenido que ahí se encuentra.

Para crear esa conexión emocional, hemos de elegir estratégicamente qué elementos vamos a mostrar y cómo hacerlo y es aquí donde algunos principios adquieren un nuevo significado el cual veremos en la especificación de los requisitos no funcionales.

3. Alcance

En este apartado se procederá a comentar en detalle los distintos objetivos que este proyecto espera cumplir, así como los requisitos que se deben satisfacer para que la solución que se ofrece sea válida y acorde a las necesidades de los usuarios. Por último, se estudiarán los posibles obstáculos y los riesgos que suponen frente al desarrollo de este proyecto.

3.1 Objetivos

El principal objetivo de este proyecto no es otro que el de solventar el problema que supone la brecha digital en nuestra sociedad. Que la exclusión social producida por la transición digital no sea una opción, que una persona de edad avanzada que no sepa cómo usar un ordenador, o de cualquiera edad realmente, sea capaz de aprender a hacerlo.

Para conseguir este objetivo, este proyecto ha de ser capaz de:

⁹ **Look & Feel:** Expresión inglesa que, en el ámbito de la informática y del software, representa el conjunto de cualidades y características que le dan a un producto una identidad visual única y que pueden ser percibidos de manera diferente de acuerdo con cada usuario.

- Presentar una solución en un formato adaptado a las posibles capacidades deterioradas de una persona mayor
- Presentar una solución de uso intuitivo, sencillo y que permita a alguien su manejo sin que le sea necesario ningún tipo de conocimiento previo.
- Presentar una solución cuyo contenido en forma de sencillas lecciones permita aprender a hacer cosas básicas con un ordenador y que las lecciones que vayan a aparecer hayan sido elegidas por una encuesta a un grupo de usuarios representativo.
- Presentar una solución que sea fácilmente ampliable y adaptable a los cambios que experimenten dichas necesidades con el tiempo y a las actualizaciones de los servicios que ofrece la red.

Además, como objetivo secundario, este proyecto pretende romper las barreras mentales que este tipo de usuario suele tener frente a la tecnología y llamar su atención o potenciar curiosidad para que se vean capaces a sí mismos de explorar la red por su cuenta y de profundizar en estos nuevos conocimientos.

3.2 Obstáculos y riesgos

Se ha de ser consciente que durante la fase de desarrollo de este proyecto se pueden encontrar algún que otro contratiempo que altere la planificación o, incluso, el alcance inicial esperado. Es interesante tratar de preverlos en la medida de lo posible y tener preparados una serie de planes de contingencia para afrontar los riesgos que suponen.

Podemos detectar los siguientes obstáculos:

- **Inexperiencia en las tecnologías utilizadas:** como ya se ha comentado brevemente en el apartado de motivaciones personales, esta es la primera vez que el desarrollador de este proyecto utilizará este tipo de tecnologías y que llevará a cabo desde cero una plataforma web de esta índole. Por una parte, esto implica un proceso previo de aprendizaje que, en caso de ser más largo que lo previsto, puede retrasar todas las otras fases del proyecto. Por otro lado, la propia inexperiencia puede provocar más interrupciones ya sea por dudas en cómo proceder o por la aparición de errores imprevistos que no se sepa cómo solucionar.
- **Falta de voluntarios:** este proyecto requiere de un grupo de personas con unas características muy concretas para dos fines. El primero es formular los servicios que la plataforma ofrecerá a partir de sus necesidades y, el segundo, será el de probar la aplicación una vez esté terminada para corregir errores e ir iterando sobre distintas implementaciones hasta lograr aquella que mejor se adecue a los criterios de calidad establecidos. El contexto de emergencia sanitaria provocado por el COVID-19 puede afectar a esta captación de voluntarios lo que de nuevo

se traduce en retrasos perjudiciales que afectan a la planificación temporal y a la calidad final del producto.

- **El papel de otras asignaturas y una fecha de entrega fija:** merece la pena recordar que este proyecto no deja de ser un trabajo de final de grado académico por lo que está rodeado de otras asignaturas que también influyen en el tiempo total que el desarrollador dispone hasta la entrega final cuya fecha es inamovible. Los retrasos que se puedan producir a causa de la carga de trabajo del resto de asignaturas afectan directamente a la planificación temporal y a la calidad del producto.

Para todos estos obstáculos existe una única solución y es la de rebajar la carga de trabajo del proyecto. Se empezaría rebajando el número de iteraciones de refactorización en el desarrollo. Cuántas más se hagan, más calidad se tendrá, pero, en un escenario donde el tiempo escasea, prevalece más que la solución sea útil.

Además, para el de la falta de encuestados siempre se podría recurrir a tácticas comunes de recolección de datos como lo son encuestas a transeúntes y centros de mayores. Por otro lado, si lo que faltan son testers, no habrá más remedio que hacerlo con menos de los deseados lo que, como mucho, afectará a la cantidad de feedback recibido y a la precisión de los detalles de los cambios por hacer.

En definitiva, habiendo explorado este apartado a conciencia, se podría resumir en la siguiente tabla DAFO [13]:

DEBILIDADES	AMENAZAS
<ul style="list-style-type: none">• Falta de conocimiento en los aspectos técnicos del desarrollo.• Circunstancias pandémicas que rodean al tipo de usuario al que queremos dirigirnos.	<ul style="list-style-type: none">• Falta de tiempo.• Falta de voluntarios.• Otras asignaturas con las que lidiar.
FORTALEZAS	OPORTUNIDADES
<ul style="list-style-type: none">• Proyecto especialmente diseñado a partir de las necesidades de los usuarios.• Sin competencia directa.	<ul style="list-style-type: none">• Fácilmente ampliable en el futuro.• Impacto positivo en la sociedad.

Figura 12: Tabla DAFO de este proyecto. Fuente: Elaboración propia.

4. Metodología y rigor

4.1 Metodología de trabajo

Elegir la metodología de trabajo adecuada para nuestro proyecto no es una decisión que deba tomarse a la ligera. No existe una forma de proceder aplicable a todo tipo de trabajo. Son muchos los factores a tener en cuenta.

En el momento de seleccionar una se ha tenido en consideración únicamente metodologías a las que ya se ha recurrido en el pasado en cualquier momento del transcurso de la carrera. Todas ellas son lo suficientemente variadas como para que una se asemeje bastante a lo que se requiere. Así no tendremos que recurrir a otras alternativas más desconocidas que, si bien puede que encajen mejor, seguramente retrasarían más el desarrollo del proyecto debido a que no se estaría acostumbrado a su uso.

Se descartará el uso de Test Driven Development [14] ya que se considera que someter a la implementación a numerosas iteraciones de refactorización es contraproducente dadas las circunstancias actuales. Además, este es un trabajo individual por lo que cualquier técnica de Extreme Programming [15] queda directamente descartada.

Dicho esto, la segunda opción a considerar es la metodología Agile [16] que parece ser más apropiada por diversas razones. Una de ellas es porque nos permite adaptarnos fácilmente a los cambios que puedan producirse en los requisitos y las soluciones con el paso del tiempo. Algo que puede suceder ya que este trabajo no va destinado a un cliente en concreto sino a todo un colectivo. Otra razón es que nos permite tener siempre un producto útil, que sea operativo, lo cual nos da más flexibilidad a la hora de aplicar planes de contingencia si hay retrasos en la planificación temporal a causa de alguno de los obstáculos mencionados anteriormente. Y, por último, el autor de este proyecto tiene experiencia previa en el uso de Agile ya que lo ha usado en diversas asignaturas a lo largo de la carrera.

Eligiendo esta metodología, ya solo falta concretar qué técnica Agile. Todo termina reduciéndose a la siguiente decisión: ¿Scrum o Kanban? [17]. Los dos son métodos muy populares dentro de Agile. Tanto, que costó realmente decantarse por uno siendo Kanban finalmente el elegido. Ambos están idealmente pensados para el trabajo en equipo, aunque Scrum le da mucha más importancia al papel que juegan los roles y, como que este es un proyecto académico individual, Kanban parecía ser el más indicado. Además, los dos métodos permiten hacer cambios en las necesidades del trabajo durante su desarrollo, algo indispensable como ya se ha comentado hace un momento. Pero Scrum limita estos cambios al final de cada sprint, momento que tiene lugar cada 2 o 3 semanas dependiendo de su duración. Kanban permite hacer esta modificación al momento pues el flujo de trabajo es siempre modificable. Por último, quizás la envergadura del proyecto no sea tanta como para requerir de un sistema organizado por sprints, una velocidad basada en los puntos de historia y reuniones de

revisión. Nos bastará con la organización y la facilidad de cambiar los casos de usuario de un estado a otro que ofrece Kanban. Cuando analicemos la herramienta de seguimiento a utilizar comprenderemos mejor la aplicación de esta metodología para un solo desarrollador.

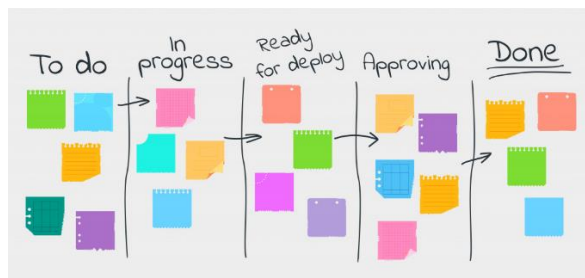


Figura 13: Ejemplo de tablero Kanban.

Fuente: <https://www.bitrix24.es/about/blogs/desarrollo-de-negocios/lo-que-necesitas-saber-sobre-un-tablero-kanban.php>

4.2 Herramientas y recursos para el desarrollo

4.2.1 Herramientas hardware

En cuanto al hardware se utilizará el portátil Aspire E15 del que actualmente se dispone ya que tiene la capacidad suficiente para soportar todo lo necesario para diseñar e implementar el sistema.

4.2.2 Herramientas software

Se utilizarán varios programas en las distintas fases de desarrollo. Para el diseño se usará Adobe XD [18], un editor de gráficos vectoriales que nos permitirá diseñar y crear un prototipo de la maqueta que tendrá la plataforma web y, por ende, simular la experiencia del usuario. Su uso es gratuito y forma parte de los servicios que presta Adobe Creative Cloud.

Para la implementación se utilizará la versión premium del entorno de desarrollo integrado IntelliJ IDEA 2020 [19]. Aunque su uso implica una suscripción que supone un coste anual de 499€ el primer año, el acceso es gratuito para estudiantes.

Para la redacción de esta memoria se utilizará la herramienta de procesamiento de textos Microsoft Word [20], versión del 2016.

Se usará Robo3t [21] para hacer pruebas locales con la base de datos y Postman [22] para testear la API.

4.2.3 Herramientas de seguimiento y control de versiones

Trello

Trello [23] es un software de administración de proyectos cuya interfaz es perfecta para el desarrollo mediante la metodología Agile Kanban porque permite la creación de las distintas columnas que simulan las fases de desarrollo, la adición de tarjetas, el movimiento de ellas de una columna a otra, su asignación a miembros del equipo de desarrollo (que en este caso nos es indiferente) y la importación de archivos multimedia entre otros.

Como que el uso de Trello es tan común cuando se utiliza este tipo de metodología, la propia herramienta ya ofrece incluso plantillas de tableros Kanban. Se utilizará uno de estos diseños predefinidos, aunque se modificarán alguna de las columnas para ajustarse completamente a la división de fases que se precisa. De este modo, contaremos con una columna de cosas pendientes To-Do, una columna de trabajo previo llamada Previous Work (en ella se preparará el contenido que, a posteriori, se incluirá en la implementación), una columna para el trabajo en progreso con el nombre de Doing, una columna Testing para indicar que se está probando lo realizado y, finalmente, una columna para aquellas tareas finalizadas Done.

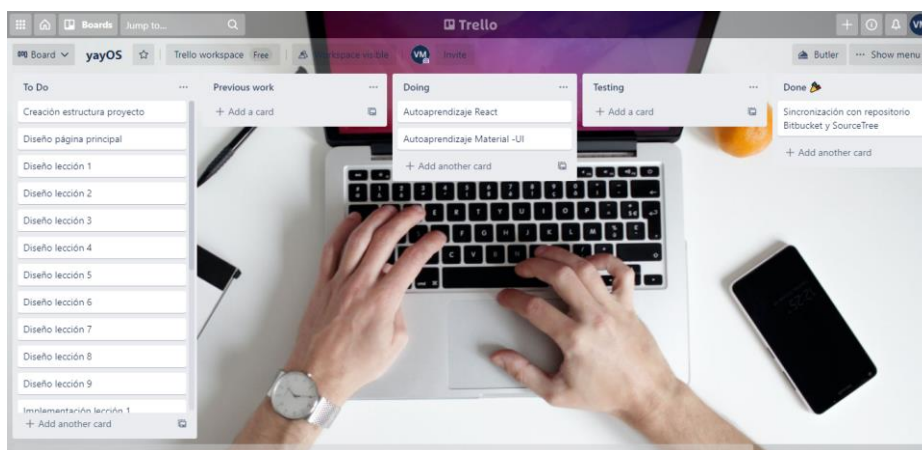


Figura 14: Ejemplo de trabajo con Trello. Fuente: <https://trello.com/b/wkkrJhQX/yayos>

Aunque el movimiento de las tarjetas irá de izquierda a derecha en el tablero, se dará el caso que una tarjeta regrese desde Testing a Doing cuando los testers indiquen que cierto aspecto de la plataforma web debería cambiarse.

Bitbucket y SourceTree

Para el control de versiones usaremos Bitbucket [24] que es un servicio de alojamiento web para los proyectos que utilizan un control de versiones como Git [25]. También se usará SourceTree [26] como herramienta de apoyo para disponer de una representación más visual de los repositorios y de las ramas.

Aunque este proyecto académico sea individual y, en principio, no tendrían que producirse conflictos puesto que no podrá darse el caso de que más de un programador modifique un mismo punto durante la implementación, se va a recurrir al uso de un instrumento de este calibre para mantener una copia de seguridad actualizada del código en todo momento y para poder dividir la implementación en distintas ramas de manera que se puedan probar alternativas sin comprometer el correcto funcionamiento de lo ya implementado.

4.3 Rigor de validación

La validación de todo lo elaborado durante este proyecto se llevará a cabo tanto en la memoria como en el desarrollo de la plataforma web. Por lo que respecta a la redacción de este documento, la validación se llevará a cabo mediante reuniones y correspondencia electrónica continua con la directora del TFG y, en aquellos apartados donde se precise, con los tutores de la asignatura de GEP e, incluso, de APC puesto que en esta última asignatura se practica la preparación de análisis de sostenibilidad enfocados a la realización de este trabajo.

Y, por lo que se refiere a la implementación, como ya se ha dicho, el desarrollo de la plataforma web se irá iterando sobre los comentarios recibidos de los testers que son, eventualmente, futuros usuarios de la plataforma. Por lo que siempre prevalecerá la validación de que todo lo que se produzca sea correcto acorde a los requisitos del proyecto. Aparte del testeo local, por supuesto.

5. Planificación temporal

Este proyecto fue iniciado oficialmente el 1 de febrero de 2021 y se espera presentarlo frente al Tribunal de Especialidad entre finales de junio y principios de julio. Por tanto, se ha supuesto como fecha teórica de finalización el 1 de junio de este mismo año.

Dado que este trabajo entra dentro de la modalidad A, que corresponde a aquellos trabajos realizados dentro de la UPC y sin contar con la participación de ninguna empresa, el tiempo diario de desarrollo impuesto a sí mismo por el autor es de una media de 5 horas diarias, lo que equivale a una duración total aproximada de 595 horas en las que se llevarán a cabo todas y cada una de las tareas que van a ser especificadas a continuación.

Recordemos que usamos una metodología Agile - Kanban, por lo que no tenemos ninguna forma de organizar las tareas como lo serían, por ejemplo, los sprints que sí encontramos en Scrum. Nos basta con saber que tareas hemos de llevar a cabo y las subtareas que implica cada una de ellas. Por tanto, dividiremos el conjunto de trabajo según el ámbito al que pertenece de manera que la estructura queda de la siguiente forma:

5.1 Descripción de tareas

5.1.1 Gestión del proyecto (60 horas)

Aquí se encuentran todas aquellas tareas relacionadas con la administración inicial del proyecto. Dichas tareas fueron realizadas durante la asignatura de Gestión de Proyectos (GEP).

- **T1.1 - Contextualización y alcance (25 horas):** Consiste en la redacción de un documento que incluya la definición del contexto, el marco y el alcance del proyecto. Implica recolectar toda la información relevante sobre el problema a resolver, tanto sus causas como las soluciones actuales, los objetivos y los medios disponibles para alcanzarlos, los posibles obstáculos, y su propia relevancia y justificación.
- **T1.2 - Planificación temporal (15 horas):** Consiste en la redacción de un documento en el que se planifique y programe el desarrollo del proyecto. Se debe prever las tareas que se han de llevar a cabo y estimar el tiempo que se va a tardar en finalizarlas, así como la especificación de dependencias entre ellas. Incluye la realización de un Diagrama de Gantt [27] y un análisis de riesgos a partir de los obstáculos definidos en la tarea anterior, por lo que depende de T1.1.
- **T1.3 - Planificación presupuestaria y análisis de sostenibilidad (15 horas):** Consiste en la redacción de un documento en el que se realice un presupuesto económico del proyecto y un análisis de su sostenibilidad. En este caso, esta tarea depende de T1.2.
- **T1.4 – Integración final del documento (5 horas):** Se realizará un documento que contenga todos los apartados realizados en las tareas anteriores y que sirva de base para la memoria de este TFG. Como que la tarea T1.2 depende de la tarea T1.1 y la tarea T1.3 depende de la tarea T1.2, esta tarea depende de T1.3 dependiendo así, indirectamente, de las 3.

5.1.2 Diseño y arquitectura (150 horas)

A este grupo pertenecen todas aquellas tareas relacionadas con el diseño de la plataforma web, así como la preparación de todos aquellos aspectos adicionales requeridos.

- **T2.1 – Realización de un estudio de usuarios (20 horas):** Esta tarea consiste en la preparación con Google Forms, distribución, realización presencial y análisis de un estudio realizado a potenciales usuarios de la plataforma web con el fin de conocer sus necesidades y preferencias y extraer así información útil para la especificación de requisitos. Depende de T1.1.
- **T2.2 – Especificación de los requisitos (40 horas):** Proceso de análisis y especificación de los requisitos, tanto funcionales como no funcionales, de nuestro sistema mediante la metodología de clasificación Volere. Las dificultades

cognitivas y/o auditivas, así como el mermado de las capacidades físicas y/o mentales, implica un nivel de detalle de los requisitos mucho más concreto que en otros proyectos. Por lo que este proceso es más largo de lo habitual. Depende directamente de T2.1.

- **T2.3 – Preparación del entorno de trabajo (5 horas):** La preparación del entorno implica la descarga e instalación de todo el software necesario para el desarrollo de este proyecto. Esto incluye las herramientas de control y de seguimiento que, como ya se explicó en el capítulo de Metodología y Rigor, son Trello, Bitbucket y SourceTree. Incluye también la instalación de la IDE IntelliJ IDEA 2020.1 x64 y de las librerías necesarias durante la programación, la herramienta Adobe XD para el diseño, Postman y Robo3T. Requiere que, previamente, se haya finalizado T1.1.
- **T2.4 – Aprendizaje de React (25 horas) [28]** Como ya se ha mencionado anteriormente, una de las dificultades de este TFG es la inexperiencia en el uso de las tecnologías requeridas para el desarrollo de la plataforma. Esto implica un proceso de autoaprendizaje en el que se recurrirá a distintos tutoriales y documentación online sobre el uso de React y de librerías como Material UI [29] para poder llevar a cabo la implementación. Depende de T2.3 para poder hacer pruebas de los conceptos asimilados.
- **T2.5 – Diseño del contenido de las lecciones (45 horas):** Implica la redacción, estructuración y creación de las 9 lecciones que formarán el contenido desplegado en la plataforma web. Aunque se sepa de antemano los conceptos a explicar, se deberán buscar formas simples de introducirlos pues se pretende usar un nivel de lenguaje básico para el fácil entendimiento. Se dividirán las distintas lecciones en apartados o secciones, se buscará toda la información relevante que merezca ser enseñada y se adaptará de la forma más óptima. Se calcula una media de 5 horas por lección. Esta tarea depende de la especificación de requisitos T2.2.
- **T2.6 – Diseño de las interfaces de la plataforma web (15 horas):** Consistirá en diseñar cada una de las pantallas que se espera que la plataforma web tenga. Se utilizará la herramienta Adobe XD para ello y se tratará que el diseño final se asemeje lo máximo posible a las capacidades tanto del desarrollador como de las que ofrece el propio React y la librería Material UI. También forma parte de esta tarea la definición de la arquitectura lógica y física del sistema. Depende del diseño de contenido T2.5.

5.1.3 Implementación (175 horas)

A continuación, se detallan todas aquellas tareas que forman parte de la programación de la plataforma web.

- **T3.1 – Implementación de la arquitectura interna (25 horas):** Consiste en programar la base de la plataforma con React. Implica generar un nuevo proyecto React, estructurarlo, enrutar los componentes base que formaran las distintas páginas y generar todas las dependencias de importaciones requeridas. Depende de T2.6.

- **T3.2 – Implementación de la pantalla principal (20 horas):** Consiste en programar la pantalla principal de la plataforma web, aquella que servirá de menú y que contendrá todas las lecciones disponibles que el usuario puede aprender. Incluye la programación de la funcionalidad de filtrado. Depende de T3.1.
- **T3.3 – Implementación de la lección “Teclado y ratón” (5 horas):** Consiste en programar la pantalla de la lección “Teclado y ratón” con la interfaz diseñada y el contenido preparado. Depende de T3.2.
- **T3.4 – Implementación de la lección “Crearse una cuenta de Google” (5 horas):** Consiste en programar la pantalla de la lección “Crearse una cuenta de Google” con la interfaz diseñada y el contenido preparado. Depende de T3.3.
- **T3.5 – Implementación de la lección “Uso de un correo electrónico” (10 horas):** Consiste en programar la pantalla de la lección “Uso de un correo electrónico” con la interfaz diseñada y el contenido preparado. Depende de T3.4.
- **T3.6 – Implementación de la lección “Hacer videollamadas” (15 horas):** Consiste en programar la pantalla de la lección “Hacer videollamadas” con la interfaz diseñada y el contenido preparado. Incluye programar una función de detección de sistemas operativos. Depende de T3.5.
- **T3.7 – Implementación de la lección “Uso de Google Maps” (15 horas):** Consiste en programar la pantalla de la lección “Uso de Google Maps” con la interfaz diseñada y el contenido preparado. Depende de T3.6.
- **T3.8 – Implementación de la lección “Buscar por Internet” (10 horas):** Consiste en programar la pantalla de la lección “Buscar por Internet” con la interfaz diseñada y el contenido preparado. Depende de T3.7.
- **T3.9 – Implementación de la lección “Gestión de ficheros del ordenador” (10 horas):** Consiste en programar la pantalla de la lección “Gestión de ficheros del ordenador” con la interfaz diseñada y el contenido preparado. Depende de T3.8.
- **T3.10 – Implementación de la lección “Comprar por Internet” (15 horas):** Consiste en programar la pantalla de la lección “Comprar por Internet” con la interfaz diseñada y el contenido preparado. Depende de T3.9.
- **T3.11 – Implementación de la lección “Tener una red social” (20 horas):** Consiste en programar la pantalla de la lección “Tener una red social” con la interfaz diseñada y el contenido preparado. Depende de T3.10.
- **T3.12 – Proceso de perfeccionamiento (20 horas):** Consiste en programar todas aquellas mejoras que hagan la página web más atractiva al usuario como su traducción a varios idiomas, la adición de un botón para enviar comentarios, un apartado con una breve introducción sobre la plataforma... Depende de T3.11.
- **T3.13 – Despliegue del sistema (5 horas):** Consiste en adaptar el sistema para que este pueda estar disponible para su uso. Aunque puede adaptarse en cualquier momento, la tarea se dará por finalizada cuando todas las modificaciones estén disponibles en la versión pública, por lo que depende de T3.12.

5.1.4 Revisión y refactorización (52 horas)

En este apartado se incluyen las tareas de revisión del producto implementado y las relacionadas con las modificaciones consecuentes.

- **T4.1 – Proceso de testeo (30 horas):** Durante esta tarea se pondrá a prueba el sistema con un grupo de usuarios que se prestaron voluntarios para ello durante la realización de las encuestas, por lo que son una representación exacta de lo que sería un usuario real. Se tomará constancia de su experiencia, así como de sus opiniones tanto positivas como negativas. Depende de la finalización de T3.13.
- **T4.2 – Refactorización (20 horas):** Se procederá a hacer las modificaciones pertinentes en base a las opiniones recibidas de los testers, por lo que esta tarea depende directamente de la T4.1.
- **T4.3 – Elaboración de conclusiones (2 horas):** Se analizará el resultado del uso del sistema por parte de los usuarios voluntarios y se valorará su nivel de aprendizaje y, por ende, la utilidad de la plataforma. Depende de T4.1.

5.1.5 Documentación y comunicación

Por último, quedan aquellas tareas relacionadas con la elaboración de la documentación de este TFG.

- **T5.1 – Redacción de la documentación (75 horas):** Todas las fases del proyecto, así como todas las tareas vistas y por ver en este apartado, deberán de ser documentadas en una memoria que se ha de incluir junto a la plataforma web. Se aprovechará la documentación previa realizada en GEP, y se irá avanzando el resto durante el desarrollo del proyecto. Por lo que depende de todas las tareas, aunque no es necesario ni aconsejable esperar a su finalización conjunta si no que, a medida que las tareas vayan terminándose, se ira actualizando la documentación siendo, de esta forma, más precisa.
- **T5.2 – Comunicación (15 horas):** Se realizarán reuniones periódicas de seguimiento con la tutora del proyecto, así como un intercambio continuo de correspondencia electrónica. Al no poder predecir exactamente los motivos de las reuniones, no se puede especificar una dependencia en concreto. Aunque es obvio que la reunión dependerá de una tarea o conjunto de tareas en particular.
- **T5.3 – Preparación de la presentación final (35 horas):** Se preparará una presentación final de este proyecto para el turno de lectura en la fecha establecida. Será mostrado ante un tribunal y mantendrá un formato acorde a la normativa vigente. Incluye la preparación de un material multimedia de soporte y de un discurso oral. Depende de todas las tareas y, en este caso, sí que depende de su finalización conjunta, por lo que se hará después de T4.3.

En definitiva, observamos que de las 595 horas inicialmente dedicadas al proyecto se consumirían alrededor de 562, lo que deja un margen de 33 horas para lidiar con posibles inconvenientes e imprevistos. En los siguientes dos apartados veremos, respectivamente, una tabla de estimaciones con toda esta información cómodamente

resumida y un diagrama de Gantt donde se expondrá el tiempo de dedicación previsto a las tareas.

5.2 Estimación

ID	TAREA	DURACIÓN	DEPENDENCIAS	RECURSOS (además de un PC)
T1.1	Contextualización y alcance	25 horas	–	Internet, Word
T1.2	Planificación temporal	15 horas	T1.1	Internet, Word
T1.3	Planificación presupuestaria y análisis de sostenibilidad	15 horas	T1.2	Internet, Word
T1.4	Integración final del documento	5 horas	T1.3	Word
T2.1	Realización de un estudio de usuarios	20 horas	T1.1	Google Forms, WhatsApp
T2.2	Especificación de los requisitos	40 horas	T2.1	Internet, Word, Encuesta
T2.3	Preparación del entorno de trabajo	5 horas	T1.1	Internet
T2.4	Aprendizaje de React	25 horas	T2.3	Internet, IntelliJ IDEA
T2.5	Diseño de las lecciones	45 horas	T2.2	Internet, Word
T2.6	Diseño de las interfaces de la plataforma web	15 horas	T2.5	Adobe XD, lecciones
T3.1	Implementación de la arquitectura interna	25 horas	T2.6	IntelliJ IDEA, Bitbucket, Sourcetree
T3.2	Implementación de la pantalla principal	20 horas	T3.1	IntelliJ IDEA, Bitbucket, Sourcetree
T3.3	Implementación de la lección “Teclado y ratón”	5 horas	T3.2	IntelliJ IDEA, Bitbucket, Sourcetree
T3.4	Implementación de la lección “Crearse una cuenta de Google”	5 horas	T3.3	IntelliJ IDEA, Bitbucket, Sourcetree
T3.5	Implementación de la lección “Uso de un correo electrónico”	10 horas	T3.4	IntelliJ IDEA, Bitbucket, Sourcetree
T3.6	Implementación de la lección “Hacer videollamadas”	15 horas	T3.5	IntelliJ IDEA, Bitbucket, Sourcetree
T3.7	Implementación de la lección “Uso de Google Maps”	15 horas	T3.6	IntelliJ IDEA, Bitbucket, Sourcetree
T3.8	Implementación de la lección “Buscar por Internet”	10 horas	T3.7	IntelliJ IDEA, Bitbucket, Sourcetree
T3.9	Implementación de la lección “Gestión de ficheros del ordenador”	10 horas	T3.8	IntelliJ IDEA, Bitbucket, Sourcetree
T3.10	Implementación de la lección “Comprar por Internet”	15 horas	T3.9	IntelliJ IDEA, Bitbucket, Sourcetree
T3.11	Implementación de la lección “Tener una red social”	20 horas	T3.10	IntelliJ IDEA, Bitbucket, Sourcetree
T3.12	Proceso de perfeccionamiento	20 horas	T3.11	IntelliJ IDEA, Bitbucket, Sourcetree
T3.13	Despliegue del sistema	5 horas	T3.12	IntelliJ IDEA, Google Hosting
T4.1	Proceso de testeo	30 horas	T3.13	yayOS, testers, Word
T4.2	Refactorización	20 horas	T4.1	IntelliJ IDEA, Bitbucket, Sourcetree
T4.3	Elaboración de conclusiones	2 horas	T4.1	Word
T5.1	Redacción de la documentación	75 horas	Todas (gradual)	Internet, Word
T5.2	Comunicación	15 horas	–	Gmail, Google Meet
T5.3	Preparación de la presentación final	35 horas	T4.3	Internet, Power Point

Figura 15: Tabla con la estimación de horas, dependencia y recursos de cada tarea. Fuente: Elaboración propia

5.3 Diagrama de Gantt

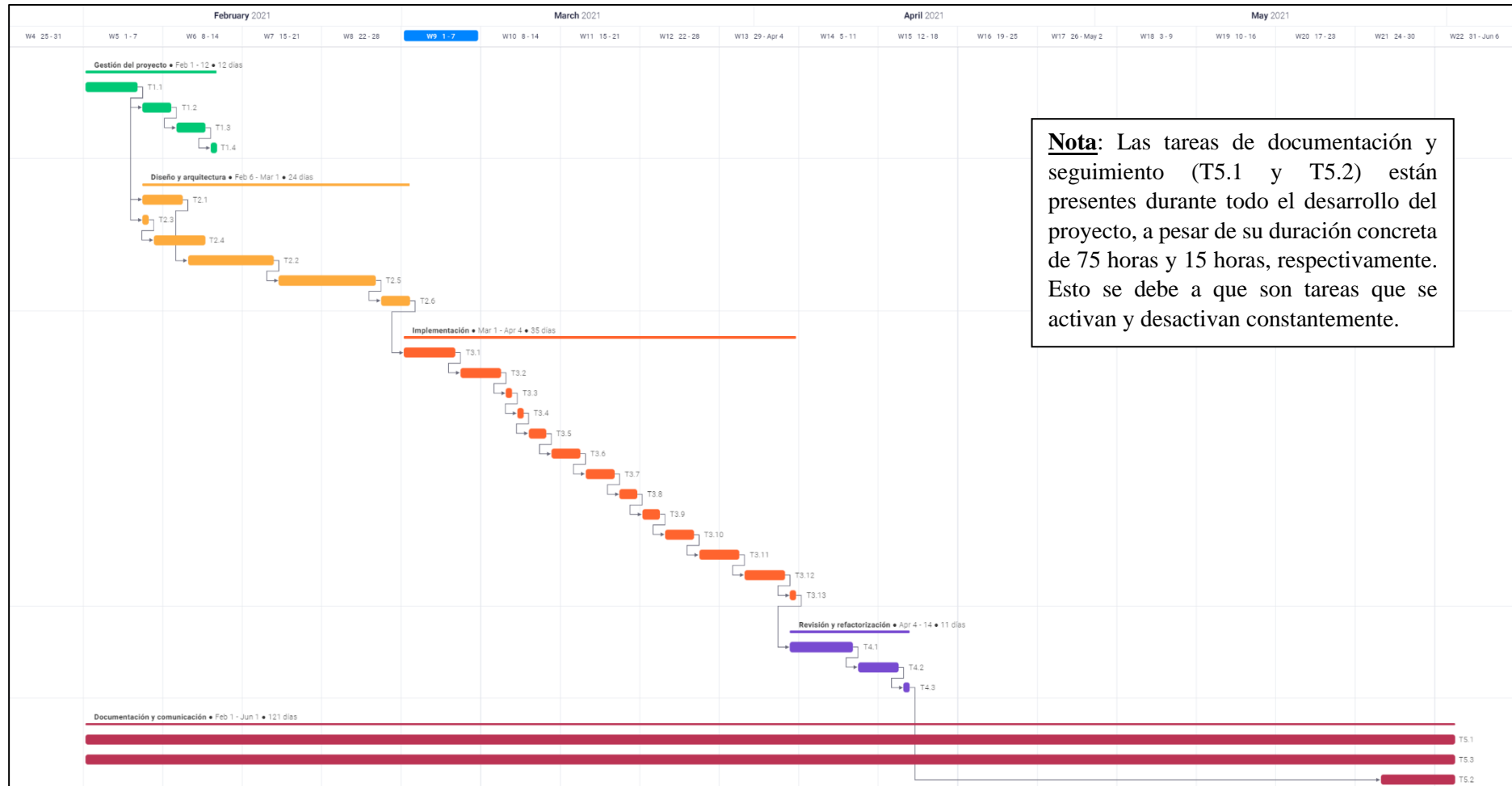


Figura 16: Diagrama de Gantt. Fuente: Elaboración propia.

5.4 Gestión de riesgos

Anteriormente se ha dado un rápido vistazo a los posibles obstáculos que podrían interferir en el desarrollo de este proyecto afectando, directamente, en la planificación temporal y en la calidad del mismo. Se ha adoptado una medida general que se resumía en acortar la carga de trabajo y el nivel de perfeccionamiento de la misma, pero, ahora que se han visto las distintas tareas en las que se divide la elaboración del sistema, será interesante ver exactamente cuáles de estas se ven afectadas y cómo repercutirán en ellas las acciones tomadas para reducir el impacto que puedan tener dichos obstáculos.

Recordémoslos brevemente:

Riesgo	Probabilidad	Horas adicionales
Inexperiencia en las tecnologías utilizadas	Alta	+20-30h
Falta de voluntarios	Baja	+5h-10h
El papel de otras asignaturas y una data de entrega fija	Baja	+10h

Figura 17: Tabla con el nombre de los obstáculos, su probabilidad de aparición y las horas a sumar que supondría en la planificación. Fuente: Elaboración propia.

Hay que tener en cuenta que en la planificación temporal ya contábamos con un margen de 33 horas, que era la diferencia entre el tiempo predestinado al desarrollo del proyecto (a 5 horas diarias durante 4 meses daba 595 horas), y el tiempo real una vez sumábamos el tiempo que conllevaba cada tarea (562 horas). Esta medida reduce considerablemente la preocupación por las horas adicionales y, en especial, la del riesgo que supone la inexperiencia en las tecnologías utilizadas puesto que su probabilidad es alta. Con todo y eso, no hay que fiarse ya que cualquier retraso en alguna tarea por cualquier imprevisto acortará drásticamente dicho margen. Es por ello, que se requiere de otras alternativas.

5.4.1 Inexperiencia en las tecnologías utilizadas

El no tener prácticamente ningún conocimiento en el uso de React hace más propensa la aparición de bugs y más costosa su resolución. Por no hablar de que el proceso de programar se vuelve más lento. Para mitigar estos retrasos, ya se ha tenido consideración de este riesgo durante la planificación temporal, por lo que las horas calculadas en el apartado de estimaciones ya tienen en cuenta que la implementación de las partes de la plataforma web conllevará más horas de las que le llevaría a un desarrollador con más experiencia.

Sin embargo, un plan alternativo sería aprovechar aquellas horas sobrantes de tareas que se han llevado a cabo antes de la fecha prevista y reinvertirlo en profundizar en los conocimientos de esta biblioteca de JavaScript. Para ello, se podría buscar contenido en la red o preguntar a contactos con más experiencia en su uso que pudieran compartir algún que otro consejo.

En caso de que esto no fuera suficiente, y siendo pues una situación extrema, se optaría por eliminar la lección de “Tener una red social” puesto que de todas las propuestas fue la menos votada. Si aun así no fuera suficiente, la siguiente en caer sería la de “Comprar por Internet”.

5.4.2 Falta de voluntarios

La falta de voluntarios es un problema que puede presentarse de dos formas completamente distintas. Puede aparecer en el momento de realizar la encuesta que dictaminará qué lecciones se mostrarán en la plataforma web, y también puede aparecer en el momento de buscar posibles futuros usuarios que prueben el sistema para detectar fallos o mejoras.

En el primer caso, lo que se puede hacer es recurrir a estudios previamente realizados en vez de finalizar el nuestro. Estaría bien que el proyecto se basara exclusivamente en los datos recogidos por el mismo ya que serán más concretos y actuales. Pero eso implica una mayor cantidad de tiempo invertido que no buscar en la red algún informe estadístico sobre los problemas más frecuentes con los que se encuentra la gente de avanzada edad en el uso de nuevas tecnologías.

Por otro lado, la búsqueda de voluntarios es algo que inicialmente estaba pensado para llevarse a cabo exclusivamente en la tarea T2.1, ya que la misma encuesta ofrecía la oportunidad de testar en un futuro el sistema. Pero, en realidad, la participación de dichos voluntarios no será necesaria hasta la finalización de la tarea T3.13 como bien se puede observar en la Figura 4, por lo que la búsqueda de testers puede alargarse durante la fase de diseño y la de implementación reutilizando de nuevo aquellas horas sobrantes de otras tareas ya finalizadas.

5.4.3 El papel de otras asignaturas y una data de entrega fija

La presencia de ambos elementos es algo intrínseco en el desarrollo de este proyecto y se ha intentado adecuar la planificación lo máximo posible. Por lo general, como medida preventiva, se ha evitado elegir abril como mes de finalización del proyecto y se ha apostado por la fecha más lejana, junio. También se han elegido asignaturas que no comportaran una carga excesiva de trabajo. Por último, se ha elegido una cantidad de horas al día de dedicación al proyecto realista en base al resto de factores.

En este caso no existe una alternativa reactiva aparte de hacer horas extra. Ir más allá de las 5 diarias establecida. A pesar de esto, durante la elección de las optativas elegidas para este último cuatrimestre de carrera se ha tenido en mente el esfuerzo que supone el TFG y es por ello que se ha elegido Arquitectura de PC (APC) que, aunque por el nombre pueda parecer que no posee relación con el marco de este proyecto, resulta que una importante parte de la materia es la correcta realización de análisis de sostenibilidad desde un punto de vista tanto económico, como social y medioambiental.

6. Gestión económica

A pesar de que este proyecto es un trabajo académico de final de grado y que, por tanto, no está ni remunerado ni se espera obtener de él algún tipo de beneficio económico, es interesante simular un análisis presupuestario de lo que costaría realmente llevarlo a cabo. De esta forma se puede tener una idea más clara de si el proyecto sería viable o no en un ámbito más profesional.

6.1 Identificación de los costes

Lo primordial para saber si es económicamente viable es saber cuánto va a costar desarrollarlo. En otras palabras, se han de identificar y estimar cada uno de los gastos que pueden surgir. Para ello, se ha de tener en cuenta una serie de factores entre los cuales se incluyen no solo a los empleados y los costes del software y del hardware usado. Se ha de considerar una serie de costes más indirectos, pero igualmente presentes, como lo son los gastos generales del emplazamiento en el que se trabaja (luz, agua, Internet...), los impuestos que hay que pagar, los costes imprevistos que pueden surgir (horas extras, tareas no previstas...) y el dinero reservado por si se ha de recurrir a uno de los planes de contingencia preparados.

Las siguientes estimaciones se han hecho redondeando al alza.

6.1.1 Recursos Humanos

Aunque la única persona que trabaja en este proyecto es el autor del mismo, en el ámbito laboral esto no sería así y los distintos roles que son asumidos aquí por una única persona serían, en realidad, asignados a distintos empleados. Es por esta razón que lo primero que se ha de hacer si se quiere estimar el coste de los recursos humanos necesarios para el proyecto es calcular el coste que supondrá pagar su salario. Teniendo en cuenta el alcance del proyecto y su nivel de complejidad, se ha decidido que se requeriría un grupo de 5 empleados formados por un director, un ingeniero del software junior y otro senior, un diseñador de UI y, por último, un probador de software o tester que, en vez de ser un voluntario, sería alguien con experiencia en el ámbito.

En la Figura 18 se pueden observar el coste que supondría por hora los salarios de los empleados. Dichos promedios se han hecho consultando ofertas actuales de empleo en las plataformas más utilizadas para ello en España, como lo son Glassdoor [30] e Indeed [31]. Para dividir los salarios anuales en horas se ha presupuesto una media de 243 días de trabajo a 8 horas diarias, lo que supone unas 1944 horas anuales de trabajo.

Rol	Sueldo bruto	Sueldo neto (bruto x1.3)
Product Manager	23,15 €/h	30,03 €/h
Ingeniero del software junior	10,80 €/h	14,04 €/h
Ingeniero del software senior	21,09 €/h	28,05 €/h
UI designer	14,41 €/h	18,73 €/h
Tester	13,38 €/h	17,39 €/h

Figura 18: Sueldos promedio por hora de los distintos roles. Fuente: Elaboración propia.

Teniendo en cuenta el coste que supone por hora la implicación de los distintos roles en el desarrollo del proyecto, podemos estimar el coste total que supondrá el salario de los participantes si multiplicamos el sueldo neto por la cantidad de horas que dedicará cada uno según la planificación realizada en el apartado anterior. Una operación que puede verse reflejada en la Figura 19, situada a continuación.

Rol	Sueldo neto	Horas dedicadas según la planificación	Estimación coste total
Product Manager	30,03 €/h	185 h	5.555,55 €
Ingeniero del software junior	14,04 €/h	90 h	1.263,60 €
Ingeniero del software senior	28,05 €/h	85 h	2.384,25 €
UI designer	18,73 €/h	150 h	2.809,50 €
Tester	17,39 €/h	52 h	904,28 €
TOTAL:			1.2917,18 €

Figura 19: Coste de cada rol en base al tiempo dedicado al proyecto. Fuente: Elaboración propia.

Redondeando al alza, estaríamos ante un coste en salarios de prácticamente 13.000 €.

6.1.2 Recursos Hardware

En el marco de este proyecto se está usando, por la parte de hardware, un portátil Acer Aspire E15 que, en su momento (febrero de 2018), costó alrededor de 700 € pese a que ahora en el mercado se puede encontrar por poco menos de 500 € [32]. Suponiendo que este proyecto se estuviera desarrollando por el equipo mencionado en el apartado anterior, necesitaríamos como mínimo 5 ordenadores de las mismas características por lo que, adquiridos ahora, implicaría un gasto de 2.500€.

Pero esta estimación no es del todo precisa, pues lo corriente sería amortizar la compra. Para calcular dicha amortización, se hará el siguiente factor de conversión:

$$\frac{\text{Coste (euros)}}{\text{Vida útil (años)} \times \text{Días laborables al año} \times \text{Dedicación diaria (horas)}} \times \text{Duración del proyecto (horas)}$$

Considerando el coste de un portátil de 500 €, una vida útil de 4 años, los 243 días laborables al año mencionados ya antes, una dedicación diaria de 5 horas y una duración total de 562 horas como se indicaba en la planificación temporal, podemos concluir en la Figura 20 que destinaríamos, al alza, 300€:

Recurso Hardware	Precio medio	Amortización
1 portátil Acer Aspire E15	500 €	57,81 €
5 portátiles Acer Aspire E15	2500€	289,05 €

Figura 20: Precio y amortización del hardware. Fuente: Elaboración propia

6.1.3 Recursos Software

Aunque la mayoría de herramientas software requeridas para el desarrollo de este proyecto son gratuitas, existen ciertos aspectos que merece la pena comentar.

Para la implementación del sistema se está usando la versión “Ultimate” de la IDE IntelliJ IDEA, que vendría a ser la versión premium de la plataforma. Aunque su versión gratuita sería suficiente para el desarrollo del proyecto, si extrapolamos las circunstancias actuales al hipotético escenario del mundo laboral, el uso de esta versión mejorada supone un coste anual de 499 € [33] el primer año y con rebajas en los posteriores. Su uso es gratuito exclusivamente para alumnos universitarios, por lo que tendríamos que tener en cuenta el coste adicional que supone.

Por otro lado, en el marco real de este TFG el autor ha requerido de un proceso de aprendizaje previo de React [34] que ha supuesto un pago de 12,99€. Tratando de imitar al máximo posible las circunstancias que afectan a este proyecto, el ingeniero del software junior también habrá requerido de este curso por lo que es otro coste más a incluir. Además, se debe recalcar que dicho precio incluye un descuento del 88%, por lo que en realidad el precio del curso es de 109,99 €.

Teniendo en cuenta las dos licencias de la versión Ultimate de la IDE IntelliJ IDEA (una por programador) y el curso de React, esto suma, al alza, otros 1.118 €.

6.1.4 Gastos generales

A continuación, se va a calcular los costes que supone el espacio de trabajo. Para facilitar esta estimación, se va a optar por el alquiler mensual de una oficina compartida o *coworking*. Se ha buscado diversas ofertas en las principales plataformas que ofrecen este tipo de servicio en Barcelona. La opción por la que se ha optado puede verse en la Figura 21.

Coworking	Precio mensual	Tiempo requerido	Precio total
Meet BCN [35]	292 €	4 meses	1.168 €

Figura 21: Gasto del alquiler de un espacio de trabajo. Fuente: Elaboración propia.

Al dedicar 5 horas diarias al proyecto, el alquiler no ha podido ser a tiempo parcial lo que ha encarecido 72€ su precio. Sin embargo, en el caso de que se requieran horas extras ya se cuenta con un lugar en el que hacerlas sin tener que pagar más.

6.1.5 Gastos imprevistos y de contingencia

Aunque por el momento hemos tratado de redondear al alza para mitigar los costes imprevistos al máximo, no es suficiente. Se tendrá en consideración los posibles imprevistos que afecten a los gastos en recursos humanos. En cuanto al pago de software o gastos generales se cree que es improbable que un repentino suceso implique un aumento en el gasto y, en el caso de que se produjera una avería en alguno de los equipamientos hardware, como todos serían recién comprados estarían cubiertos por la garantía.

Recuperando la tabla vista en el apartado de gestión de riesgos disponemos de las horas extras que implicaban los obstáculos en la planificación temporal del proyecto. En la Figura 22 veremos el coste añadido que dichos riesgos implican.

Riesgo	Máximo de horas extra	Rol al cargo	Sueldo neto	Coste adicional
Inexperiencia en las tecnologías utilizadas	30 h	Ingeniero del software junior	14,04 €/h	421,20 €
Falta de voluntarios	10 h	UI designer	18,73 €/h	187,30 €
TOTAL:				608,5 €

Figura 22: Coste adicional de los riesgos identificados. Fuente: Elaboración propia.

Al alza, supondremos 610€ de imprevistos.

Como se podrá observar, se ha eliminado el obstáculo que supone para este proyecto la presencia de dos asignaturas durante la realización del mismo puesto que esta es una dificultad que posee el autor de este trabajo y que, difícilmente, pueden tener los empleados.

La inexperiencia en las tecnologías utilizadas es un problema que se ha asignado únicamente al programador junior, puesto que es más propenso a cometer errores que no un senior, por lo que el precio de las horas extras será estimado teniendo en cuenta su sueldo neto. Y, además, la falta de voluntarios ha sido asignada al UI designer puesto que él es el encargado de la realización de la encuesta que permitirá deducir los requisitos no funcionales del sistema, que incluyen el diseño del mismo.

Por último, el coste estimado a contingencias es un gasto añadido que supone un porcentaje en concreto del valor total que supone el proyecto y que, en este tipo de campo, suele rondar entre el 10 y el 20%. Es por esa razón que consideraremos un 15% de margen para la contingencia. En la Figura 23 podemos ver este cálculo y el coste estimado final del proyecto.

Tipo	Coste al alza
Recursos humanos	13.000 €
Recursos hardware	300 €
Recursos software	1.118 €
Gastos generales	1.168 €
Total previo	15.586 €
Contingencia (+15%)	2.337.9 €
Imprevistos	610 €
TOTAL:	18.534 €

Figura 23: Coste total estimado del proyecto. Fuente: Elaboración propia.

6.2 Control de gestión

Tan importante es una correcta estimación del presupuesto final del proyecto como la implantación de medidas para controlar el seguimiento de dicho presupuesto. De nada sirve tener una ruta de gastos económicos si no se consulta si se está siguiendo. Es por ello que se requiere de mecanismos para asegurar que los gastos del proyecto se encuentran en todo momento bajo los niveles contemplados.

A continuación, se detallan una serie de indicadores que serán consultados y actualizados durante el desarrollo del trabajo y los métodos mediante los cuales serán calculados:

- **Desviación del coste de horas por tarea**
 - Se calculará multiplicando la diferencia entre las horas estimadas a cada tarea y las horas reales dedicadas por el coste real. El uso de Trello como herramienta de seguimiento facilitará este recuento.
- **Desviación total costes personales por actividad**
 - Una vez esté terminado todo el proyecto, si agrupamos los costes reales de todas las tareas y hacemos la diferencia con el coste estimado en el apartado de recursos humanos, obtendremos la diferencia entre lo teórico y lo práctico.
- **Desviación total costes imprevistos**
 - Consiste en hacer, de nuevo, la diferencia entre los costes de imprevistos estimados y los reales.
- **Desviación total de horas**
 - Una vez más, se debe hacer la diferencia entre las horas totales estimadas hasta el momento y las reales requeridas.
- **Desviación total de costes**
 - Bastará con sumar todos los costes reales y ver la diferencia con el presupuesto final estimado. Al no haberse presupuesto costes adicionales ni en el hardware, ni en el software ni en el alquiler de la zona de trabajo por los motivos ya mencionados, la diferencia debería ser exclusivamente a causa de los recursos humanos.

7. Análisis de sostenibilidad

Se espera que este proyecto pueda ayudar a las personas mayores a utilizar las nuevas tecnologías y, más concretamente, los ordenadores. Esto supone un impacto social considerable puesto que puede llegar a significar un cambio positivo en las vidas de estas personas. Es por eso que, a continuación, se detallará un análisis de sostenibilidad enfocado desde un punto de vista no solo social, sino que también se contemplarán las perspectivas medioambientales y económicas. De esta manera podremos analizar en profundidad el verdadero impacto de yayOS tanto en su desarrollo, como en su vida útil y los riesgos que pueda afrontar la plataforma.

7.1 Dimensión medioambiental

Ciertamente puede parecer que carezca de sentido el valorar el impacto medioambiental de un proyecto de esta índole. Al tratarse de una solución software que no requiere de la producción de un objeto físico, no parece que vaya a tener un efecto real similar al que ofrece la producción de otro tipo de producto. Si que es cierto que los recursos hardware requeridos no han surgido de la nada y, seguramente, han sido producidos mediante energías no renovables. Pero como que el portátil utilizado no ha sido comprado recientemente, es algo que se escapa de las manos del autor de este TFG. Por eso, desde un punto de vista ecológico, nadie puede verse notablemente perjudicado por su desarrollo, ni se ha contemplado decisiones éticas al respecto de ningún tipo puesto que no hay alternativas reales más allá del uso de recursos hardware fabricados a partir de energías renovables.

Desde el punto de vista de la vida útil se puede llegar a la misma conclusión. Un sistema software no es algo que consuma nada a excepción de la electricidad que necesita la máquina que lo reproduce. Pero dicha alimentación es incondicional, puesto que el consumo eléctrico entre un ordenador simplemente encendido y un ordenador que utiliza la plataforma yayOS es prácticamente el mismo.

Con una huella ecológica prácticamente inexistente, no se ven alternativas para haber hecho las cosas de otra manera ni riesgos que la hicieran aumentar.

7.2 Dimensión económica

En el caso económico sí que se ha estimado el coste total del proyecto como bien puede observarse en el apartado de presupuestos. Sin embargo, se ha hecho simulando unas circunstancias laborales ficticias puesto que este es un proyecto académico no remunerado y sin aspiraciones económicas, por lo que, mientras que en el análisis presupuestario puede parecer que el proyecto es muy poco viable debido a la gran diferencia entre costes generados y beneficios obtenidos, en el marco real esto no es

así puesto que los gastos son prácticamente nulos al igual que los beneficios que se espera obtener. Los recursos usados que han implicado un gasto real han sido, principalmente, el pago de un pequeño curso de React. En cuanto al hardware o los gastos del lugar de trabajo son independientes de la realización de este TFG por lo que no deben de tenerse en cuenta. Del mismo modo, el software usado es gratuito por la condición de estudiante del autor, por lo que no paga ninguna de las licencias que si debería pagar una empresa para su uso. Y, por último, este proyecto no requiere la contratación de empleados ni el pago de salarios. Por tanto, la única vía existente de ahorro hubiese sido recurrir a contenido gratuito de la red para el proceso de autoaprendizaje.

Durante su vida útil el sistema ni consume ni genera ya que su despliegue se ha hecho en una plataforma gratuita y se accede mediante el navegador del propio usuario. En el caso de que el uso de yayOS incrementara en el futuro sí que se tendría que valorar el uso de servidores y dominios de pago, así como alguna forma de sufragar dichos gastos (publicidad, donaciones, lecciones premium de pago...).

Al no generar beneficios y comportar unos gastos mínimos, no existen riesgos aparentes que perjudicaran la viabilidad del proyecto. El único beneficio que espera obtener el autor es el académico.

7.3 Dimensión social

yayOS es un proyecto que aporta mucho tanto a nivel personal como a la sociedad. Ya se han comentado los motivos que empujaban al autor a llevar a cabo este TFG y aquellos beneficios que esperaba obtener: dar solución a un problema real, ampliar sus conocimientos de cara al mundo laboral, ganar práctica en el desarrollo y gestión de proyectos... Y, a nivel social, se espera que este proyecto pueda reducir la brecha digital y ayudar a las personas más mayores a aprender a usar, o al menos a descubrir, la infinidad de cosas que puede uno hacer cuando es capaz de utilizar un ordenador.

A lo largo de la vida útil, se cree y se espera que este proyecto sirva a muchas personas que hoy en día se ven incapaces de usar un ordenador a vencer sus miedos y ser autosuficientes en el ámbito tecnológico. La solución ofrecida es algo totalmente nuevo o, por lo menos, se presenta de una forma completamente distinta. La medida no soluciona la brecha digital, pero sí que da un primer empujón para hacerlo. Al fin y al cabo, todo depende también de la fuerza de voluntad.

Y ese es el único riesgo real de este proyecto. Que la falta de voluntad de un colectivo escéptico y desesperanzado perjudique el futuro de esta plataforma.

8. Requisitos

Todo producto está correctamente desarrollado si los requisitos están correctamente definidos, pues sólo cumpliendo dichos requisitos se consigue alcanzar el objetivo deseado. Entendemos por requisito aquella capacidad que ha de tener nuestro sistema o una parte en concreto de él para resolver un problema o satisfacer una necesidad. Es por este motivo que conocer el problema es vital y por eso se le ha dado tanta importancia hasta ahora en este trabajo en particular. No basta con saber que existe una brecha digital entre grupos sociales de distintas edades y sus consecuencias. Tenemos que conocer las necesidades del público al que queremos dirigirnos para lograr formular una descripción de los requisitos del producto que vamos a ofrecerles. Como afirman Suzanne Robertson y James Robertson¹⁰ en su obra “Mastering the requirements process [36]: *“If the right product is to be built, then the right requirements have to be discovered”*. Y es que, en realidad, da igual si estamos hablando de un producto de software, como es el caso, o de uno de hardware o de un servicio o de lo que sea. Los requisitos existen tanto si los analizamos como si no. Es el hecho de especificarlos lo que permite entender realmente el problema y que nos ayuda a señalar la solución apropiada frente a una baraja de posibles soluciones alternativas.

¿Y qué se entiende por requisitos? Suzanne y James Robertson distinguen dos tipos principales de requisitos y los definen de la siguiente forma en su libro: Por un lado, están aquellos que se denominan como requisitos funcionales, que son, a fin de cuentas, las funcionalidades que ha de tener nuestro sistema o, en otras palabras, los servicios que debe ofrecer para ser útil para su operador. Y los llamados requisitos no funcionales, que engloban toda característica o propiedad que ha de tener el sistema para ser aceptable para el dueño y el operador. Veremos la importancia que tienen estos últimos en especial para nuestro caso pues nuestro proyecto no servirá de mucho, por muy bien definidas que estén las funcionalidades, si no está correctamente adaptado. Es por eso que a este tipo de requisitos también se les llama requisitos de calidad.

Para la realización de este apartado se tendrán en cuenta dos fuentes de instrucción. Por un lado, se aplicarán los conceptos aprendidos en las asignaturas cursadas en la carrera como Ingeniería de Requisitos (ER) y Gestión de Proyectos de Software (GPS). Y, por el otro, profundizaremos dichos conocimientos aprovechando que tenemos acceso a una de las guías más conocidas sobre la especificación de requisitos de la que ya se ha hecho mención en el párrafo anterior.

Para definir ambos tipos requisitos seguiremos el modelo de clasificación Volere, todo un referente hoy en día en la definición de requisitos. Nació en 1995 con la necesidad de establecer un lenguaje común para que los distintos stakeholders de un proyecto pudieran tener una concepción general de los requisitos de un problema a pesar de que

¹⁰ **Suzanne Robertson y James Robertson:** aclamados analistas de requisitos a nivel internacional y directores de la compañía Atlantic Systems Guild.

sus enfoques y sus áreas de experiencia fueran completamente diferentes. Un lenguaje no técnico que permitiera a los distintos miembros de toda la cadena de desarrollo comprender inequívocamente lo que es necesario. Explicado brevemente, Volere define una forma simple de escribir cada requerimiento, denominada como “Atomic Requirement Shell”, que consiste en rellenar una serie de atributos que lo definan. Podemos ver un ejemplo de su contenido en el siguiente ejemplo extraído de la página 397 de la obra a la cual se ha ido haciendo referencia desde el principio de este apartado del proyecto.

The diagram shows the 'Atomic Requirement Shell' template with various fields and annotations:

- Requirement #:** Unique id
- Requirement Type:** The type from the template
- Event/BUC/PUC #:** Id of events / use cases that need this requirement
- Description:** A one sentence statement of the intention of the requirement
- Rationale:** A justification of the requirement
- Originator:** The stakeholder who raised this requirement
- Fit Criterion:** A measurement of the requirement such that it is possible to test if the solution matches the original requirement
- Customer Satisfaction:** Degree of stakeholder happiness if this requirement is successfully implemented. Scale from 1 = uninterested to 5 = extremely pleased.
- Customer Dissatisfaction:** Measure of stakeholder unhappiness if this requirement is not part of the final product. Scale from 1 = hardly matters to 5 = extremely displeased.
- Priority:** The relative importance of the requirement
- Conflicts:** Other requirements that cannot be implemented if this one is
- Supporting Materials:** Pointer to documents that illustrate and explain this requirement
- History:** Creation, changes,

The logo 'Volere' and 'Copyright © Atlantic Systems Guild' are also visible.

Figura 24: Atomic Requirement Shell de Volere.

Fuente: Robertson, S., & Dr, Robertson. J. (2012).

Mastering the Requirements Process: Getting Requirements Right. Third Edition. Addison-Wesley ISBN: 9780321815743

8.1 Requisitos funcionales

8.1.1 Proceso de recogida de datos

Para conocer las necesidades funcionales que este proyecto de plataforma web tenía que cubrir se realizó un estudio estadístico a un grupo de potenciales usuarios mediante Google Forms, un software gratuito de administración de encuestas. Se buscaron personas mayores de 65 años que tuvieran un conocimiento sobre el uso de un ordenador que fuese prácticamente nulo.

Debido al contexto de situación de emergencia sanitaria provocada por el COVID-19, el número de usuarios encuestados no es muy elevado. Las circunstancias que rodean a este tipo de usuario implican, en la mayoría de los casos, la presencialidad del autor de

este proyecto o la de alguien de confianza durante el desarrollo de la encuesta ya que la mayoría desconoce cómo contestar a la encuesta desde un teléfono móvil. Esto complicó mucho la obtención de datos en el margen de tiempo establecido, principalmente por las restricciones de confinamiento y movilidad reducida vigentes, y limitó el número de participantes a 18 (8 hombres y 10 mujeres de entre 65 y 100 años). De estas personas, 5 se ofrecieron voluntarias para testear la plataforma una vez esté terminada por lo que serán ellos los que comprueben que se cumplan o no los criterios de aceptación de los requisitos. A pesar de estas circunstancias no se optó por activar el plan de contingencia por falta de voluntarios en ningún momento.

Se les preguntó previamente si sabían cómo usar un ordenador y si, en caso de ser no su respuesta, si les gustaría aprender. El 28% dijo que no tenían intención de hacerlo, así que se les preguntó cuál era el motivo y las respuestas más comunes eran “Porque no lo necesito”, “Porque me parece muy complejo y no me veo capaz” y “Porque si necesito hacer algo ya lo hace alguien por mí”. Al 72% restante se les pidió que marcaran de una lista de opciones previamente preparada qué les gustaría aprender a hacer y de ahí surgieron las lecciones que la plataforma contendrá y que, a continuación, detallaremos como requisitos funcionales de nuestro sistema. En los anexos 1 y 2 se puede encontrar, respectivamente, una copia de la encuesta y de sus resultados.

8.1.2 Descripción de los requisitos

La descripción de los requisitos funcionales se hará siguiendo el método de descripción que presenta Volere y que se ha visto anteriormente, pero de una forma más simplificada. Esto se debe a que es un proyecto académico de un solo desarrollador y que, a causa de esto, su magnitud no es lo suficientemente compleja como para citar todos los puntos o atributos que el modelo original incluye. Como todos los requisitos de este apartado son funcionales y forman parte del mismo tipo, este no será especificado igual que tampoco lo serán otras características como la prioridad, el índice de satisfacción o insatisfacción del cliente o la persona al cargo del proyecto. Sólo se mostrarán los conceptos importantes y que varían realmente entre las distintas funcionalidades como lo son la descripción, su justificación y su criterio de aceptación. A continuación, se procede a detallarlos:

#1.

- **Descripción:** El sistema ha de permitir a un usuario aprender cómo se utiliza el ratón y el teclado.
- **Justificación:** Para enseñar al usuario a usar un ordenador es primordial que aprenda a interactuar con él, por lo que le deben de ser introducidos los dispositivos de entrada más básicos.
- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá distintas enseñanzas sobre el uso del ratón y el teclado y una zona de práctica para este último. Dicho apartado estará resaltado para que el usuario sepa que es lo primero que debe hacer.

#2.

- **Descripción:** El sistema ha de permitir a un usuario aprender a hacerse una cuenta de Google.
- **Justificación:** Muchas de las cosas que se van a aprender requieren el uso de softwares gratuitos de Google como lo son Gmail o Google Maps. Estos servicios requieren la posesión de una cuenta así que se ha decidido que lo mejor es ayudar al usuario a que se haga una cuenta primero en caso de no tenerla.
- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá una guía paso a paso de cómo crearse una cuenta en Google.

#3.

- **Descripción:** El sistema ha de permitir a un usuario aprender a usar un correo electrónico.
- **Justificación:** Hoy en día el uso de un correo electrónico es básico. Es necesario en prácticamente todos los trámites administrativos y suele ser solicitado en cualquier registro de cualquier índole. De las lecciones sugeridas en la encuesta fue la tercera más votada.
- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá una guía paso a paso de cómo crearse una dirección de correo electrónico mediante Gmail, como enviar correos, como leer los recibidos, como navegar por las distintas opciones que ofrece la interfaz...

#4.

- **Descripción:** El sistema ha de permitir a un usuario aprender cómo hacer una videollamada con Skype.
- **Justificación:** De todas las lecciones que se ofrecieron, la de aprender cómo hacer videollamadas fue la más demandada con un 72% de los votos, es decir, el 100% de las personas que si querían aprender. Se ha elegido la plataforma Skype por su popularidad principalmente, puesto que por mecánica es más o menos similar al resto de opciones que ofrece el mercado.
- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá una guía paso a paso de cómo instalar Skype, crearse una cuenta, agregar contactos, hacer y contestar videollamadas y cómo solucionar problemas de audio o de imagen.

#5.

- **Descripción:** El sistema ha de permitir a un usuario aprender a usar Google Maps.
- **Justificación:** Esta es una opción que no fue sugerida en ningún momento como una opción que los encuestados pudieran elegir, pero al autor le pareció buena idea añadirla puesto que es una herramienta muy útil para saber cómo desplazarse a los sitios o, como apuntó una encuestada, para ver mundo.
- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá una serie de lecciones sobre cómo usar la herramienta de Google Maps. Cómo buscar lugares o direcciones, explorar las distintas opciones para viajar hasta ellas, cómo usar el modo Street View...

#6.

- **Descripción:** El sistema ha de permitir a un usuario aprender a buscar por Internet.
- **Justificación:** Una vez el usuario sepa usar el teclado y el ratón y pierda un poco el miedo al ordenador, querrá empezar a buscar información en la red por su cuenta. Es por ello que es necesario enseñarle qué es un motor de búsqueda, cómo se utiliza, y dar un par de consejos sobre cómo encontrar resultados de forma rápida y segura. Aunque esta opción no estaba en la encuesta como tal, sí que lo estaba una que era “Me gustaría poder ver las noticias por Internet”, que fue la cuarta más votada.
- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá información de los principales motores de búsqueda actuales y de cómo hacer un correcto uso de ellos.

#7.

- **Descripción:** El sistema ha de permitir a un usuario aprender a usar el explorador de archivos.
- **Justificación:** Si el usuario aprende a navegar por la red, tendrá que aprender cómo puede almacenar aquella información que encuentre y dónde puede encontrarla más tarde. Esta es una lección considerada como básica por lo que no estaba en la encuesta.
- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá una explicación detallada de lo que es el explorador de archivos, de cómo acceder a él o al escritorio y de cómo crear carpetas, guardar elementos descargados en ellas, copiarlos, moverlos, borrarlos y cómo vaciar la papelera de reciclaje.

#8.

- **Descripción:** El sistema ha de permitir a un usuario aprender cómo tener su propia cuenta de Facebook.
- **Justificación:** Empatada con la lección de buscar noticias por Internet, está también fue la cuarta más votada en la encuesta. Personalmente encuentro que es dar un paso más puesto que su complejidad es mayor que lo ofrecido hasta el momento, pero tener una red social abre muchas puertas como recuperar el contacto con familiares y amigos lejanos o no tan lejanos. Se ha elegido Facebook, de nuevo, por su popularidad.
- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá una guía paso a paso de cómo crearse una cuenta en Facebook y de cómo funcionan sus servicios principales: buscar gente, ver perfiles, reaccionar a contenido...

#9.

- **Descripción:** El sistema ha de permitir a un usuario aprender cómo llevar a cabo compras online mediante Amazon.
- **Justificación:** Es la segunda opción más votada de la encuesta con un 61% de los votos. Tal vez sea la lección más compleja, y por eso la he dejado la última. Comprar por Internet siempre es peligroso, pero es una necesidad que ha crecido exponencialmente en tiempos de pandemia y Amazon no solo es un servicio que ha demostrado ser fiable y eficiente, también ofrece muchas opciones de entrega personalizadas.

- **Criterio de satisfacción:** El sistema tendrá un apartado al que el usuario podrá acceder haciendo clic y que contendrá una serie de consejos sobre cómo comprar por Internet mediante el uso de la plataforma mencionada.
- #10.
- **Descripción:** El sistema ha de permitir a un usuario filtrar la lista de lecciones disponibles.
 - **Justificación:** Aunque por ahora el número de lecciones es reducido, se espera que este se amplíe en el futuro en base a la demanda de los usuarios. Se requiere, por tanto, de un sistema de filtrado que permita al usuario encontrar fácilmente una lección en particular en la que esté interesado.
 - **Criterio de satisfacción:** El sistema tendrá la opción de filtrar los resultados de lecciones mostrados. Por el momento solo se podrá filtrar por grado de dificultad, puesto que el usuario no conoce, tal vez, los conceptos y la terminología suficiente como para saber qué buscar por palabras.
- #11.
- **Descripción:** El sistema ha de permitir a un usuario desplazarse por las distintas secciones que conforman una lección.
 - **Justificación:** Las lecciones están divididas en distintas secciones según el apartado al que hagan referencia dentro de las mismas.
 - **Criterio de satisfacción:** El sistema tendrá una lista en forma de menú lateral con los apartados de la lección y se mostrará aquel que haya sido seleccionado. También habrá flechas que permitirán ir al siguiente apartado o al anterior.
- #12.
- **Descripción:** El sistema ha de permitir a un usuario poder comentar y puntuar una lección en particular.
 - **Justificación:** Para saber el grado de satisfacción del usuario con la lección, su contenido y la forma en la que está explicada se ha de permitir que el usuario pueda dar su opinión en específico. De esta forma los administradores podrán comprobar la efectividad del enfoque propuesto y hacer cambios si lo consideraran oportuno.
 - **Criterio de satisfacción:** El sistema tendrá un apartado de valoraciones por cada lección en el que los usuarios podrán añadir su opinión sobre la lección y una valoración en forma de estrellas.
- #13.
- **Descripción:** El sistema ha de permitir a cualquiera visualizar las valoraciones de una lección.
 - **Justificación:** Una sección de comentarios no tiene sentido si nadie puede leerlos. De esta forma la gente podrá ver qué opina el resto sobre una lección en particular y sobre la forma en la que está explicada.
 - **Criterio de satisfacción:** El sistema mostrará los últimos comentarios registrados y dará la oportunidad también de verlos todos. A su vez, mostrará la media actual de la lección calculada en tiempo real a partir de todas las valoraciones registradas de aquella lección.

#14.

- **Descripción:** El sistema ha de permitir a los administradores gestionar los comentarios registrados.
- **Justificación:** Se requiere de un cierto control y gestión sobre los comentarios.
- **Criterio de satisfacción:** El sistema dotará de herramientas a los administradores para poder buscar comentarios por palabra coincidente o identificador. También permitirá su eliminación bajo su criterio.

#15.

- **Descripción:** El sistema ha de permitir a un usuario enviar un mensaje a los desarrolladores con sugerencias o quejas.
- **Justificación:** Similar al requisito #12, el usuario podrá opinar de forma general sobre la plataforma yayOS, así como reportar algún error o aportar alguna sugerencia para mejorar.
- **Criterio de satisfacción:** El sistema tendrá un botón de contacto que dará a un breve formulario para rellenar y en el que el usuario podrá aportar su opinión general sobre la plataforma e incluir un correo electrónico de contacto si su circunstancia lo requiriese.

#16.

- **Descripción:** El sistema ha de permitir a los administradores gestionar las sugerencias enviadas por los usuarios.
- **Justificación:** Al igual que con los comentarios, se requiere de un cierto control y gestión de los formularios de sugerencias enviados por los usuarios.
- **Criterio de satisfacción:** El sistema dotará de herramientas a los administradores para poder buscar y visualizar sugerencias por palabra coincidente o identificador. También permitirá su eliminación bajo su criterio.

#17.

- **Descripción:** El sistema contendrá desafíos.
- **Justificación:** En aquellas lecciones donde sea razonable, se introducirán pequeños ejercicios de pruebas relacionados con el tema de la lección para facilitar el aprendizaje de algún concepto mediante su práctica.
- **Criterio de satisfacción:** El sistema tendrá en aquellas lecciones donde corresponda accesos a pequeños juegos cuyas características dependerán del ámbito de la lección.

8.2 Requisitos no funcionales

8.2.1 Aplicación de la psicología aplicada al diseño de UI

Anteriormente, en el apartado de conceptos previos, se ha hecho una primera aproximación sobre la importancia de entender las diferencias entre la psicología aplicada al diseño de una UI por una persona acostumbrada al uso de interfaces y por una que no lo está tanto.

Se ha concluido que un usuario de edad avanzada es más propenso a una experiencia negativa que le haga “huir” de nuestra plataforma y que nuestra única baza es centrar nuestros esfuerzos en el Look & Feel para que esto no suceda. La identidad visual de la interfaz debe estar correctamente adaptada a las dificultades cognitivas del usuario y tiene que tener un diseño visualmente atractivo, cómodo y, lo más importante, intuitivo.

Se ha logrado definir una serie de especificaciones visuales que serán, a continuación, transformadas en requisitos de calidad.

8.2.2 Definición de requisitos no funcionales

De nuevo, se usará una versión simplificada de la plantilla original de Volere. Sin embargo, en esta ocasión los dividiremos en distintos grupos según el tipo de requisito no funcional al cual pertenecen porque se distinguen según el ámbito al que afectan. Esta división, será llevada a cabo siguiendo los estándares y la terminología de la clasificación Volere tal y cómo consta en el apéndice A de la obra referenciada desde la página 435 hasta la 456.

Requisitos de apariencia (10a):

Engloba todos aquellos requisitos relacionados con la estética del sistema.

#18.

- **Descripción:** La interfaz del sistema ha de evitar el uso de colores pastel.
- **Justificación:** El uso de este tipo de colores, a pesar de ser muy común hoy en día, no aporta suficiente contraste por lo que dificulta su percepción por parte de un usuario con dificultades cognitivas.
- **Criterio de satisfacción:** La interfaz evitará el uso de este tipo de color.

#19.

- **Descripción:** La interfaz del sistema ha de evitar el uso de colores azulados en aquellas zonas más relevantes.
- **Justificación:** Las tonalidades azules son las primeras en verse peor cuando una persona empieza a experimentar dificultades cognitivas.
- **Criterio de satisfacción:** La interfaz evitará el uso de este tipo de color en zonas de considerable importancia. De hecho, se puede usar tonos azules en zonas de menor importancia para resaltar así las principales con otros colores.

#20.

- **Descripción:** La interfaz del sistema ha de evitar el uso de colores rojizos y verdes en general.
- **Justificación:** La pigmentación roja o verde son las que peor ve la gente con dificultades cognitivas.
- **Criterio de satisfacción:** La interfaz evitará el uso de colores rojos o verdes.

#21.

- **Descripción:** La interfaz del sistema ha de usar colores que contrasten.
- **Justificación:** El contraste entre elementos permite la diferenciación del fondo y los componentes que se muestran. Facilita el reconocimiento cognitivo.
- **Criterio de satisfacción:** La interfaz usará colores claros para el fondo y oscuros para los componentes o viceversa.

Requisitos de estilo (10b):

Engloba todos aquellos requisitos relacionados con los rasgos particulares del sistema.

#22.

- **Descripción:** La interfaz del sistema usará una sola fuente de texto.
- **Justificación:** El uso de distintas fuentes dificulta la percepción cognitiva de lo que se está leyendo.
- **Criterio de satisfacción:** La interfaz usará una sola fuente de texto salvo en aquellas situaciones en las que convenga hacer una distinción.

Requisitos de facilidad de uso (11a):

Engloba todos aquellos requisitos relacionados con adaptar el acceso al sistema.

#23.

- **Descripción:** La interfaz del sistema usará una fuente de texto grande y simple.
- **Justificación:** El uso de fuentes de texto pequeñas o adornadas en exceso dificulta la percepción cognitiva del usuario.
- **Criterio de satisfacción:** La interfaz usará una fuente de texto mayor o igual a 15 puntos siempre que sea posible y se usará una de las recomendadas a continuación: Roboto, Helvetica, Arial, Futura, Avant Garde o Verdana.

#24.

- **Descripción:** La interfaz del sistema optará por un diseño minimalista.
- **Justificación:** Las personas de edad avanzada no son multitarea por lo que no puede haber más de un foco de atención en una misma pantalla.
- **Criterio de satisfacción:** Se intentará, en la medida de lo posible, restringir la cantidad de elementos relevantes mostrados al mismo tiempo. También se evitará el uso de ventanas emergentes, las animaciones o transiciones entre ventanas y las pantallas de introducción. El acceso al contenido será directo.

#25.

- **Descripción:** La interfaz de usuario tratará de mostrar todo su contenido sin necesidad de hacer scrolling (desplazamiento vertical).
- **Justificación:** El desplazamiento vertical de las páginas web para mostrar más contenido es un concepto poco interiorizado en usuarios no acostumbrados al

uso de software. Esto crea el peligro de que el usuario no sepa acceder a toda la información disponible.

- **Criterio de satisfacción:** La interfaz evitará siempre que sea posible el uso del scrolling y lo sustituirá por más pantallas accesibles mediante un clic. Por otro lado, se resaltarán la barra de movimiento vertical si fuera imprescindible su uso.

#26.

- **Descripción:** La interfaz del sistema mantendrá la mayor cantidad de elementos posibles en las mismas condiciones durante el cambio de página.
- **Justificación:** Minimizar la carga de memoria del usuario facilita que memorice los pasos a seguir en futuros usos de la plataforma web.
- **Criterio de satisfacción:** La interfaz mantendrá todo objeto o acción visible de la plataforma durante el cambio de pantalla con las mismas características siempre y cuando sea posible. Es decir, siempre el cambio de pantalla no se produce por la modificación explícita de uno o varios componentes en particular.

#27.

- **Descripción:** La interfaz del sistema indicará las instrucciones básicas para operar la plataforma.
- **Justificación:** Durante los primeros contactos es posible que el usuario no sepa qué ha de hacer o qué puede ofrecerle la plataforma.
- **Criterio de satisfacción:** La interfaz incluirá un apartado explicativo donde enseñará cómo debe moverse por yayOS, cómo ha de utilizarlo y los fines que tiene.

Requisitos de personalización e internacionalización (11b):

Engloba todos aquellos requisitos relacionados con la traducción del contenido a distintas lenguas y las decisiones personales.

#28.

- **Descripción:** El sistema estará adaptado a distintas lenguas.
- **Justificación:** El sistema ha de ofrecer la oportunidad de aprender a cuantos más usuarios mejor, por lo que tiene que estar traducido a las lenguas más comunes.
- **Criterio de satisfacción:** Por ahora, el sistema será traducido al castellano, catalán e inglés. No se descarta ampliar a más lenguas en el futuro en caso de que sobrara tiempo en la fase de implementación.

Requisitos de aprendizaje (11c):

Engloba todos aquellos requisitos que especifican cuán fácil ha de ser el uso del producto.

#29.

- **Descripción:** El sistema ha de poder ser usado por cualquier usuario sin que este reciba previamente algún tipo de entrenamiento antes de su uso.
- **Justificación:** La interfaz ha de estar diseñada de manera que el usuario pueda aprender por su cuenta sin que nadie le enseñe.
- **Criterio de satisfacción:** El 60% de los usuarios que van a testear la plataforma, 3 de 5, han de ser capaces de, como mínimo, completar una de las lecciones de la plataforma sin ningún tipo de soporte o ayuda en el periodo de tiempo que les sea necesario.

Requisitos de comprensión y cortesía (11d):

Engloba todos aquellos requisitos relacionados con hacer comprensibles los conceptos, terminología y metáforas del producto al resto de usuarios.

#30.

- **Descripción:** Los textos han de ser claros y directos. Dirigidos a un nivel básico de lectura.
- **Justificación:** Se trata de aprender. No puede usarse terminología a la que un nativo digital está acostumbrado a no ser que se explique su significado.
- **Criterio de satisfacción:** Evitar el uso de expresiones comunes en el ámbito de los ordenadores a excepción de aquellas que vaya a ser explicado su significado.

#31.

- **Descripción:** Los iconos mostrados en la interfaz han de incluir un texto aclaratorio.
- **Justificación:** El uso de iconos en el ámbito de Internet y los ordenadores está comúnmente extendido, pero puede resultar completamente desconocidos para usuarios con experiencia nula en su uso.
- **Criterio de satisfacción:** Añadir una palabra o texto a los iconos que indique su función y, en caso de que no sea posible, añadir títulos alternativos en los iconos e imágenes que puedan verse al posar el cursor sobre ellos.

Requisitos de escalabilidad o extensión (12g):

Engloba todos aquellos requisitos relacionados con los aumentos de tamaño del producto.

#32.

- **Descripción:** El sistema debe de estar preparado para expandirse cómodamente.
- **Justificación:** Las necesidades del usuario pueden cambiar en cualquier momento. Se requiere de un sistema fácilmente adaptable a estos cambios.
- **Criterio de satisfacción:** El sistema ha de estar diseñado de manera que la inclusión, modificación o eliminación de las lecciones y sus elementos

relacionados se haga de forma rápida y eficiente. Por rápida se entiende que no conlleve mucho tiempo y por eficiente que no implique demasiados cambios.

Requisitos del ambiente físico esperado (13a):

Engloba todos aquellos requisitos relacionados en el ambiente que rodea al usuario durante el uso del producto.

#33.

- **Descripción:** Alguien con conocimientos básicos en el uso de ordenadores debe mostrarle cómo acceder a la plataforma web y facilitar su acceso en el futuro.
- **Justificación:** Una persona que no sabe de ordenadores no puede encontrar esta herramienta de ayuda por su cuenta. Un familiar, amigo o conocido debe mostrársela al inicio para que sepa de su existencia y poder así utilizarla.
- **Criterio de satisfacción:** Conocer a alguien que sepa buscar la plataforma web por Internet y pueda mostrársela al futuro usuario.

Requisitos de producción (13d):

Engloba todos aquellos requisitos relacionados con la producción del producto.

#34.

- **Descripción:** El sistema debe ser mostrado en una plataforma web por lo que requerimos de conexión a Internet.
- **Justificación:** Su acceso es mucho más rápido y cómodo para un usuario no acostumbrado a los ordenadores que la descarga, instalación y uso de un programa.
- **Criterio de satisfacción:** El sistema ha de estar constantemente actualizado y disponible.

9. Especificación de los requisitos

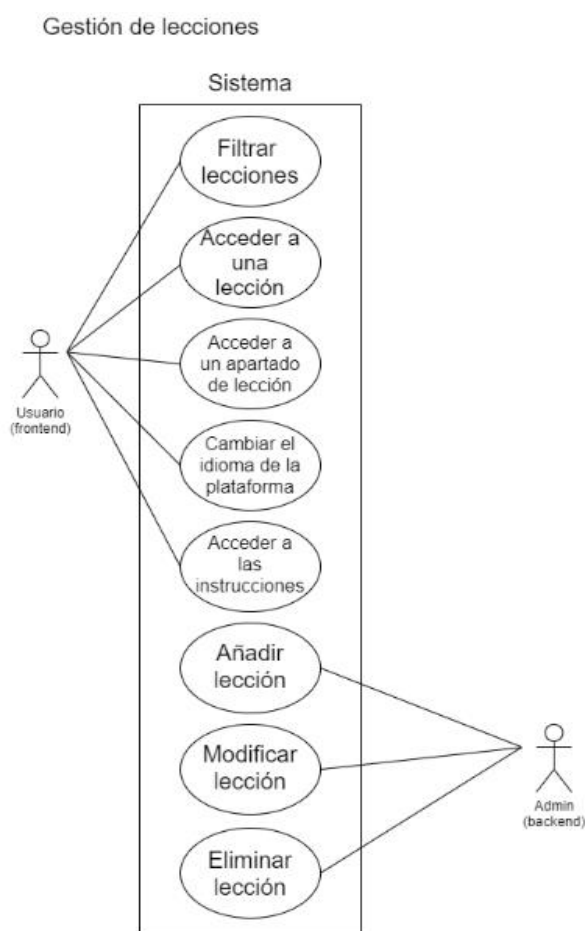
En el apartado anterior hemos podido ver una serie de requisitos funcionales que el sistema debe cumplir junto con un conjunto de requisitos no funcionales que aseguren la calidad de la solución propuesta. A continuación, se van a analizar más en detalle las exigencias mencionadas para poder describir mejor el comportamiento del sistema de cara al usuario. Esto se hará con el fin de encontrar una definición más técnica de lo que suponen los requisitos anteriores, de tratar de agrupar aquellos que tengan alguna relación y, por último, de poner los límites que el proyecto ha de abarcar.

Para realizar esta descripción completa del comportamiento del sistema se realizará un diagrama de casos de uso, un modelo conceptual y un modelo de comportamiento. Veamos en qué consiste cada uno.

9.1 Diagrama de casos de uso

El término “caso de uso” fue creado por Ivar Jacobson¹¹ en 1987 para definir la interacción entre un usuario y el sistema mediante la partición de este último en fragmentos. Estas particiones debían hacerse basándose en la visión que tiene el usuario del sistema. Por tanto, y según su definición actual [37] el caso de uso debe representar el conjunto de acciones de interacción que un agente externo, ya sea el mismo usuario u otro servicio de terceros, puede hacer en el sistema.

El diagrama de casos de uso es una forma de representación gráfica de estos casos de uso y de los actores que intervienen en ellos. Recordemos que los casos de uso surgen directamente de los requisitos detallados en el apartado anterior.



En la Figura 25 podemos ver los casos de uso relacionados con la interacción entre el usuario o administrador y las lecciones que contendrá la plataforma. Vemos todos los requisitos funcionales relacionados con las distintas enseñanzas (#1 - #9) agrupados en el caso de uso de acceder a una lección y vemos el manejo de la misma en el resto. También vemos representado el requisito no funcional #27 mediante la acción de acceder a las instrucciones y, por la parte del administrador, vemos los casos de uso referentes al requisito funcional de escalabilidad y extensión (#32).

*Figura 25: Diagrama UML de casos de uso de la interacción con el menú principal de yayOS.
Fuente: Elaboración propia.*

¹¹ **Ivar Jacobson:** ingeniero en ciencias de la computación de origen sueco al que se le atribuye la creación de los Diagramas de Secuencia, un diagrama usado hoy en día para modelar la interacción entre objetos en un sistema según UML.

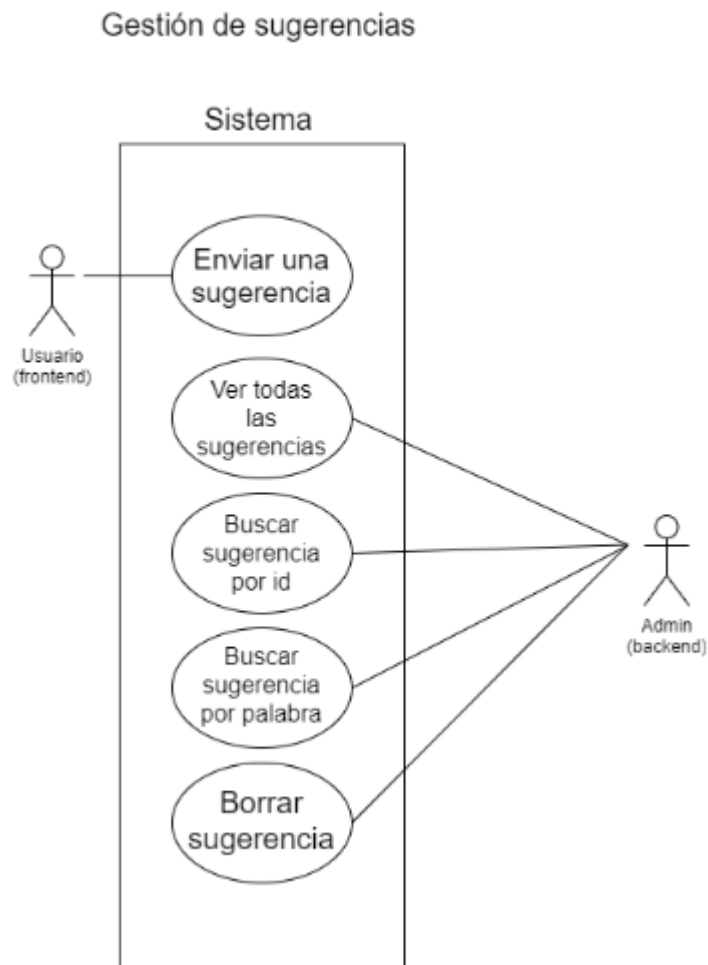
En la Figura 26 pueden verse los casos de uso relacionados con el sistema de comentarios de la plataforma. De nuevo observamos representados algunos de los requisitos funcionales relacionados con esta capacidad. También queda reflejado en el diagrama la diferencia de privilegios que tiene cada tipo de actor. El usuario solo puede ver y añadir, mientras que el administrador puede buscarlos y eliminarlos con fines analíticos y/o de mantenimiento. En el próximo apartado veremos más en detalle que implicación tiene esto.



*Figura 26: Diagrama UML de casos de uso de la interacción con el sistema de comentarios de yayOS.
Fuente: Elaboración propia.*

Por último, en la Figura 27 podemos observar los casos de uso relacionados con el sistema de sugerencias de la plataforma. Recordemos que estas son privadas y generales (no dependen de una lección en concreto), por lo que podemos ver cómo afecta esta diferencia a los casos de uso requeridos y los privilegios que tiene cada tipo de actor. En el siguiente apartado veremos también estas disimilitudes más en profundidad.

Figura 27: Diagrama UML de casos de uso de la interacción con el sistema de sugerencias de yayOS. Fuente: Elaboración propia.



9.2 Descripción de los casos de uso

Ahora que ya han sido vistos todos los posibles casos de uso del sistema, se va a proceder a describirlos detalladamente como ya se ha adelantado anteriormente. Para ello se estructurará la información en una tabla en la que se recogerá el nombre del caso de uso, el actor que lo interpreta, las precondiciones que se tienen que dar para que el caso tenga lugar (si es que las hay), el suceso que activa el caso de uso, el escenario de éxito, sus extensiones y, si se diera el caso, escenarios alternativos.

Hay que distinguir un escenario alternativo de una extensión. En esta especificación se tratará como extensión aquella posible continuación del escenario principal en el caso de que se produzca algún obstáculo (por ejemplo, la falta de campos por rellenar, no disponibilidad de los datos solicitados...). En cambio, se considerará escenario alternativo a aquellos escenarios en los que el caso de uso puede cumplirse de otra forma a la que se considera la principal. De la misma forma, un escenario alternativo puede tener extensiones también.

Se ha elegido esta estructura porque es la que se ha enseñado en la asignatura de Ingeniería de Requisitos (ER) y es con la que el autor de este proyecto tiene más experiencia. Además, recoge de forma clara y estructurada toda la información que se precisa mostrar.

Caso de uso	Filtrar lecciones.	Actor	Usuario.
Precondición	El usuario ha de estar en la vista del menú principal.		
Disparador	El usuario quiere filtrar la lista de lecciones disponibles.		
Escenario principal de éxito			
<div>1. El usuario indica al sistema el parámetro mediante el cual quiere filtrar la lista de lecciones disponibles y su valor.</div> <div>2. El sistema selecciona las lecciones que cumplan con el valor del parámetro especificado.</div> <div>3. El sistema muestra la lista de lecciones filtrada.</div>			
Extensión: No hay lecciones que cumplan con el filtrado			
<div>4. El sistema muestra un mensaje indicando que la búsqueda ha obtenido 0 resultados.</div>			

Caso de uso	Acceder a una lección.	Actor	Usuario.
Precondición	El usuario ha de estar en la vista del menú principal.		
Disparador	El usuario elige una lección y quiere entrar en ella para empezar a aprender.		
Escenario principal de éxito			
<div>1. El usuario indica una lección en concreto.</div> <div>2. El sistema carga y renderiza los componentes correspondientes a la lección y hace desaparecer los del menú principal.</div>			
Extensión: La lección está dividida en grupos de apartados			
<div>3. El sistema carga el menú secundario de acceso a cada subgrupo en la que se divide la lección.</div> <div>4. El usuario indica a qué parte de la lección quiere acceder.</div> <div>5. El sistema carga el contenido correspondiente a la lección y hace desaparecer el menú secundario.</div>			

Caso de uso	Cambiar el idioma de la plataforma.	Actor	Usuario.
Precondición	-		
Disparador	El usuario quiere cambiar el idioma de la plataforma.		
Escenario principal de éxito			
<div>1. El usuario indica el idioma en el que quiere visualizar la plataforma.</div> <div>2. El sistema recarga la vista traducida al idioma seleccionado.</div> <div>3. El sistema cambia internamente el idioma en el que tendrá que cargar el resto de componentes en futuras peticiones.</div>			

Caso de uso	Acceder a las instrucciones.	Actor	Usuario.
Precondición	El usuario ha de estar en la pantalla del menú principal.		
Disparador	El usuario quiere saber cómo manejar yayOS.		
Escenario principal de éxito			
<div>1. El usuario indica que quiere acceder a las instrucciones de uso de la plataforma.</div> <div>2. El sistema carga y renderiza los componentes correspondientes a las instrucciones de uso de la plataforma.</div>			

Caso de uso	Añadir lección.	Actor	Administrador.
Precondición	La nueva lección no debe existir previamente.		
Disparador	El administrador del sistema quiere añadir una lección a la plataforma.		
Escenario principal de éxito			
<div>1. El administrador indica el id de la nueva lección.</div> <div>2. El administrador indica el título de la nueva lección para todos los idiomas disponibles.</div> <div>3. El administrador indica la descripción de la nueva lección para todos los idiomas disponibles.</div> <div>4. El administrador indica el grado de dificultad.</div> <div>5. El administrador indica la imagen de la nueva lección.</div> <div>6. El administrador carga el contenido de la lección para todos los idiomas disponibles.</div> <div>7. El sistema crea y enlaza la nueva lección.</div> <div>8. El sistema crea la sección de comentarios de la nueva lección.</div>			

Caso de uso	Modificar lección.	Actor	Administrador.
Precondición	La lección debe existir.		
Disparador	El administrador del sistema quiere modificar el contenido de una lección de la plataforma.		
Escenario principal de éxito			
<div>1. El administrador indica el cambio que desea hacer sobre una lección.</div> <div>2. El sistema actualiza los datos sobre la lección.</div>			
Extensión: El cambio requiere de actualizar las traducciones			
<div>3. El administrador indica los cambios en las traducciones.</div> <div>4. El sistema actualiza las traducciones de los elementos especificados.</div>			

Caso de uso	Eliminar lección.	Actor	Administrador.
Precondición	La lección debe existir.		
Disparador	El administrador del sistema quiere eliminar una lección de la plataforma.		
Escenario principal de éxito			
<div>1. El administrador indica la lección a eliminar.</div> <div>2. El sistema elimina la lección especificada.</div> <div>3. El sistema elimina los elementos vinculados a aquella lección.</div> <div>4. El sistema elimina todos los comentarios de esa lección de la base de datos.</div>			

Caso de uso	Añadir comentario.	Actor	Usuario.
Precondición	La lección debe de existir y tener un apartado de valoraciones. El usuario ha de estar en la vista de comentarios de la lección.		
Disparador	El usuario quiere publicar un comentario sobre una lección en particular.		
Escenario principal de éxito			
<div>1. El usuario indica que quiere añadir un comentario.</div> <div>2. El sistema carga el formulario de envío de comentarios.</div> <div>3. El usuario introduce los datos requeridos en el formulario.</div> <div>4. El sistema valida los datos introducidos.</div> <div>5. El sistema almacena el comentario en la base de datos.</div> <div>6. El sistema indica al usuario que el comentario se ha añadido correctamente.</div> <div>7. El sistema redirige al usuario a la vista de los comentarios.</div>			

Extensión: Faltan datos o su formato no es válido	
5.	El sistema indica al usuario que faltan datos por rellenar o que su formato no es el correcto.
6.	El usuario hace las correcciones pertinentes en los datos del formulario.
7.	El sistema valida los datos introducidos.
8.	El sistema indica al usuario que el comentario se ha añadido correctamente.
9.	El sistema redirige al usuario a la vista de los comentarios.

Caso de uso	Ver últimos comentarios.	Actor	Usuario.
Precondición	El usuario ha de estar en la pantalla del menú principal.		
Disparador	El usuario elige una lección e indica que quiere entrar en su sección de valoraciones.		
Escenario principal de éxito			
<div>1. El usuario indica que quiere acceder al apartado de valoraciones de una lección concreta.</div> <div>2. El sistema muestra los últimos 15 comentarios registrados en la base de datos en orden cronológico descendente.</div> <div>3. El sistema calcula la calificación media de todos los comentarios registrados en la base de datos.</div> <div>4. El sistema muestra la calificación media de la lección.</div>			
Extensión: No hay comentarios / La búsqueda produce un error			
<div>2. El sistema indica que no hay comentarios de esa lección.</div> <div>3. El sistema indica una calificación media de la lección de 0 estrellas.</div>			

Caso de uso	Ver todos los comentarios.	Actor	Usuario.
Precondición	El usuario ha de estar en la vista de la sección de comentarios de una lección. La vista ha de contener al menos un comentario.		
Disparador	El usuario quiere ver todos los comentarios.		
Escenario principal de éxito			
<div>1. El usuario indica que quiere ver todos los comentarios.</div> <div>2. El sistema muestra todos los comentarios registrados en la base de datos en orden cronológico descendente.</div>			

Caso de uso	Buscar comentario por id.	Actor	Administrador.
Precondición	La base de datos ha de contener comentarios.		
Disparador	El administrador quiere buscar un comentario en concreto en la base de datos.		
Escenario principal de éxito			
<div>1. El administrador indica el identificador del comentario.</div> <div>2. El sistema busca en la base de datos el comentario con identificador igual al introducido.</div> <div>3. El sistema muestra todos los datos del comentario buscado.</div>			
Extensión: No hay comentarios con el id especificado / La búsqueda produce un error			
<div>4. El sistema indica que no hay comentarios con ese id y devuelve un error.</div>			

Caso de uso	Buscar comentario por palabra.	Actor	Administrador.
Precondición	La base de datos ha de contener comentarios.		
Disparador	El administrador quiere buscar los comentarios que contengan una palabra en concreto.		
Escenario principal de éxito			
<ol style="list-style-type: none">1. El administrador indica la palabra o frase que debe contener el comentario.2. El sistema busca en la base de datos todos los comentarios que tengan un nombre de usuario idéntico a la palabra o frase indicada.3. El sistema busca en la base de datos todos los comentarios cuyo contenido contenga la palabra o frase indicada.4. El sistema muestra todos los datos de todos los resultados de ambas búsquedas.			
Extensión: No hay comentarios con la palabra o frase especificada / La búsqueda produce un error			
<ol style="list-style-type: none">5. El sistema indica que no hay comentarios que contengan la palabra o frase especificada y devuelve un error.			

Caso de uso	Eliminar comentario.	Actor	Administrador.
Precondición	El comentario ha de existir.		
Disparador	El administrador quiere eliminar un comentario.		
Escenario principal de éxito			
<div>1. El administrador indica que quiere eliminar un comentario.</div> <div>2. El administrador introduce el id del comentario.</div> <div>3. El sistema elimina el comentario seleccionado.</div>			
Extensión: El comentario no existe / La petición produce un error			
<div>4. El sistema indica que el comentario especificado no existe y devuelve un error.</div>			

Caso de uso	Enviar una sugerencia.	Actor	Usuario.
Precondición	El usuario ha de estar en el menú principal.		
Disparador	El usuario quiere enviar una sugerencia a los desarrolladores.		
Escenario principal de éxito			
<ol style="list-style-type: none">1. El usuario indica que quiere enviar una sugerencia.2. El sistema carga el formulario de envío de sugerencias.3. El usuario introduce los datos requeridos en el formulario.4. El sistema valida los datos introducidos.5. El sistema almacena la sugerencia en la base de datos.6. El sistema indica al usuario que la sugerencia ha sido enviada correctamente.7. El sistema redirige al usuario al menú principal.			
Extensión: Faltan datos o su formato no es válido			
<ol style="list-style-type: none">6. El sistema indica al usuario que faltan datos o que su formato no es el correcto.7. El usuario hace las correcciones pertinentes en los datos del formulario.8. El sistema valida los datos introducidos.9. El sistema indica al usuario que el comentario se ha añadido correctamente.10. El sistema redirige al usuario al menú principal.			

Caso de uso	Ver todas las sugerencias.	Actor	Administrador.
Precondición	-		
Disparador	El administrador desea consultar todas las sugerencias enviadas.		
Escenario principal de éxito			

<ol style="list-style-type: none"> 1. El usuario indica que quiere ver todas las sugerencias. 2. El sistema muestra todas las sugerencias registradas en la base de datos.
Extensión: No hay sugerencias / La búsqueda produce un error
<ol style="list-style-type: none"> 2. El sistema indica que no hay sugerencias registradas y envía un error.

Caso de uso	Buscar sugerencia por id.	Actor	Administrador.
Precondición	La base de datos ha de contener sugerencias.		
Disparador	El administrador quiere buscar una sugerencia en concreto en la base de datos.		
Escenario principal de éxito			
<div>1. El administrador indica el identificador de la sugerencia.</div> <div>2. El sistema busca en la base de datos la sugerencia con identificador igual al introducido.</div> <div>3. El sistema muestra todos los datos de la sugerencia buscada.</div>			
Extensión: No hay sugerencias con el id especificado / La búsqueda produce un error			
<div>2. El sistema indica que no hay sugerencias con ese id y devuelve un error.</div>			

Caso de uso	Buscar sugerencia por palabra.	Actor	Administrador.
Precondición	La base de datos ha de contener sugerencias.		
Disparador	El administrador quiere buscar las sugerencias que contengan una palabra en concreto.		
Escenario principal de éxito			
<div>1. El administrador indica la palabra o frase que debe contener la sugerencia.</div> <div>2. El sistema busca en la base de datos todas las sugerencias que tengan un nombre de usuario idéntico a la palabra o frase indicada.</div> <div>3. El sistema busca en la base de datos todas las sugerencias cuyo contenido contenga la palabra o frase indicada.</div> <div>4. El sistema muestra todos los datos de todos los resultados de ambas búsquedas.</div>			
Extensión: No hay sugerencias con la palabra o frase especificada / La búsqueda produce un error			
<div>3. El sistema indica que no hay sugerencias que contengan la palabra o frase especificada y devuelve un error.</div>			

Caso de uso	Eliminar sugerencia.	Actor	Administrador.
Precondición	La sugerencia ha de existir.		
Disparador	El administrador quiere eliminar una sugerencia.		
Escenario principal de éxito			
<div>1. El administrador indica que quiere eliminar una sugerencia.</div> <div>2. El administrador introduce el id de la sugerencia.</div> <div>3. El sistema elimina la sugerencia seleccionada.</div>			
Extensión: La sugerencia no existe / La búsqueda produce un error			
<div>4. El sistema indica que la sugerencia especificada no existe y devuelve un error.</div>			

9.3 Modelo conceptual

El siguiente paso en esta especificación es definir un modelo conceptual de datos que, a efectos prácticos, no deja de ser otro tipo de representación del sistema salvo que esta vez desde el punto de vista de los componentes y los datos que lo forman.

9.3.1 Esquema conceptual de los datos

En la Figura 28 pueden verse los elementos que forman la plataforma web yayOS y los atributos por los que están representados:

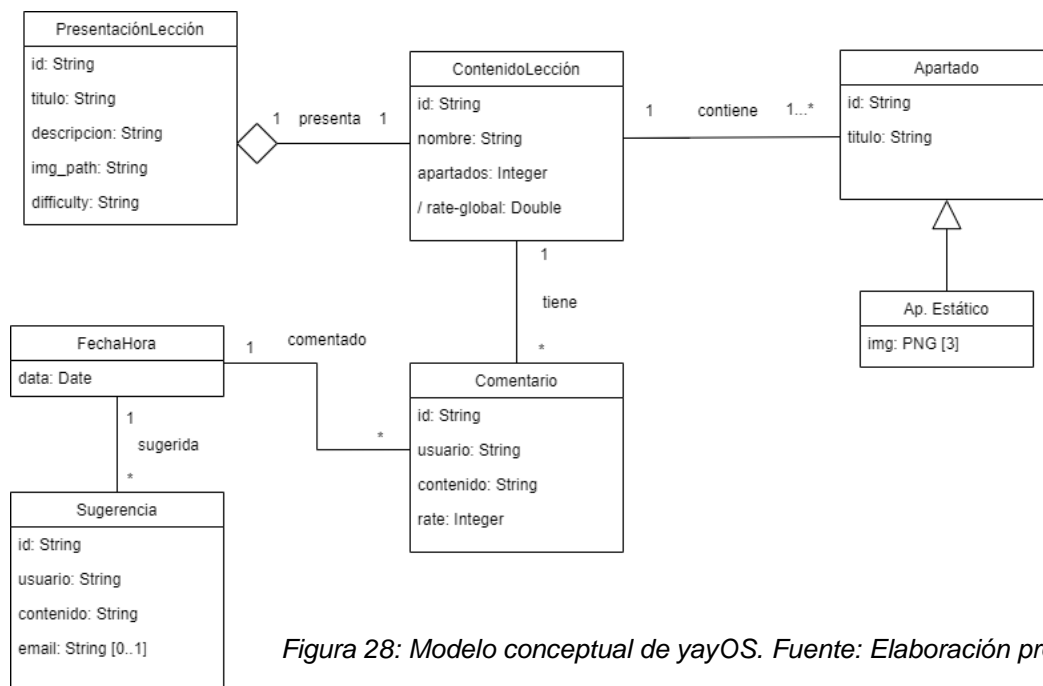


Figura 28: Modelo conceptual de yayOS. Fuente: Elaboración propia.

Como se puede observar la entidad más importante es ContenidoLección pues las distintas lecciones son los pilares del sistema. De esta clase saldrían todas las demás: su presentación en el menú principal, los comentarios asociados, sus apartados... La única entidad que iría por libre serían las sugerencias puesto que, como ya se ha visto hasta ahora, no dependen exclusivamente de una lección si no que se refieren a todo el sistema en conjunto. Distinguimos también una subclase de apartados a la que se ha denominado como “Ap. Estático”. Esto se debe a que se pretende crear contenido que requiera algún tipo de interacción con el usuario (requisito funcional #17) y otro que será únicamente de lectura. Por ahora hacemos esta distinción en base a los atributos, pero más adelante veremos las implicaciones que esto tiene a nivel funcional.

9.3.2 Restricciones de integridad

Este modelo conceptual incluye una serie de restricciones de integridad que aseguran el funcionamiento esperado en el diseño. Dichas restricciones no pueden ser incluidas en el esquema como tales y es por eso que serán detalladas posteriormente:

- **Claves externas:** (PresentaciónLección, id), (ContenidoLección, id), (Apartado, id), (Comentario, id), (FechaHora, data), (Sugerencia, id).
- **RT1:** El atributo “rate” de Comentario ha de ser mayor o igual que 1 y menor o igual que 5.
- **RT2:** Un ContenidoLección ha de estar asociada a tantos elementos de Apartado como indique el atributo “apartados”.
- **RT3:** El atributo derivado “global-rate” de ContenidoLección se calcula a partir de la media aritmética de todos los atributos “rate” de los Comentarios asociados con ContenidoLección.

9.3.3 Descripción de las clases

En este apartado va a verse una explicación más detallada de lo que representa cada clase y sus atributos.

- **ContenidoLección:** representa el grupo de lecciones que pueden aprenderse en el sistema.
 - id: identificador que utilizará el sistema para distinguirlo internamente.
 - nombre: es el nombre de la lección. Se mostrará en la interfaz cuando se acceda a la lección.
 - apartados: atributo que indica cuántos apartados tiene una lección. Será usado por el sistema para saber cuántos apartados debe de cargar.
 - / rate-global: atributo derivado usado por el sistema para saber la calificación global de una lección y mostrarla en la interfaz. Es calculado como indica la restricción textual número 3 en el apartado anterior y será usado por el sistema para representarlo en forma de estrellas.

- **PresentaciónLección:** representa los “anuncios” de las lecciones disponibles que se presentarán al usuario en el menú principal de la plataforma.
 - id: identificador que utilizará el sistema para distinguirlo internamente.
 - título: es el título de la lección. Se mostrará en la interfaz del sistema y, eventualmente, será distinto al título que contendrá la lección en la clase anterior. Esto se hará así para presentar el contenido evitando usar una terminología técnica hasta que no sea introducida a posteriori y respetando los criterios de aceptación del requisito no funcional de comprensión y cortesía #30.
 - descripcion: resumen de los objetivos de la lección. Se verán en la interfaz.
 - img-path: enlace de la imagen que se mostrará en el anuncio de la lección. Este atributo será usado por el sistema para cargar la imagen.
 - difficulty: atributo que representa cuán difícil es la lección. Se verá también en la interfaz.

- **Apartado:** representa los apartados en los que se divide una lección.
 - id: identificador que utilizará el sistema para distinguirlo internamente.
 - título: es el título del apartado. Se mostrará en la interfaz del sistema a modo de título de capítulo que tiene la lección. También funcionará como un índice reactivo.

- **Ap. Estático:** subclase de Apartado que representa los apartados estáticos que son aquellos que no tendrán una funcionalidad o con los que los usuarios no podrán interactuar y serán simplemente una imagen con su contenido. Tiene todos los atributos de la clase padre y, además:
 - img [3]: vector de 3 imágenes con el contenido del apartado estático. Cada imagen representa lo que se enseña en ese apartado de esa lección en concreto. Son 3 debido a las traducciones, de manera que una imagen estará en castellano, otra en catalán y otra en inglés. El tamaño de este vector aumentaría si lo hicieran el número de traducciones, pero de momento solo se traducirá a estos tres idiomas como indica el criterio de aceptación del requisito no funcional de personalización e internacionalización número #28.

- **Comentario:** representa los comentarios de los usuarios sobre las lecciones.
 - id: identificador que utilizará el sistema para distinguirlo internamente.
 - usuario: es el nombre del usuario que ha escrito el comentario. Será visible en el comentario. La plataforma no posee ni sistema de registro ni autenticación por lo que el nombre de usuario no es ninguna credencial. Dos comentarios distintos pueden tener el mismo nombre de usuario.
 - contenido: atributo que representa el comentario en sí. También será visible.
 - rate: representa la calificación que da el usuario a la lección del 1 al 5. Será usado internamente por el sistema para representarlo en forma de estrellas, las cuales serán visibles en el comentario.

- **Sugerencia:** representa las sugerencias de los usuarios sobre el sistema. Estas sugerencias serán usadas por los desarrolladores y administradores para mejorar los servicios. No serán públicas por lo que ninguno de los atributos será visualizado en la interfaz.
 - id: identificador que utilizará el sistema para distinguirlo internamente.
 - usuario: es el nombre del usuario que ha escrito la sugerencia. La plataforma no posee ni sistema de registro ni autenticación por lo que el nombre de usuario no es ninguna credencial. Dos sugerencias distintas pueden tener el mismo nombre de usuario.
 - contenido: atributo que representa la sugerencia en sí.
 - email: es el email del usuario que ha hecho la sugerencia por si, por las circunstancias, fuera necesario ponerse en contacto con la persona. Este atributo es opcional, por lo que su valor puede ser nulo.
- **FechaHora:** representa el momento en el que se ha registrado un elemento en la base de datos. Este elemento puede ser un comentario o una sugerencia. Como se ha podido ver solo los comentarios son públicos y visibles en la interfaz, por lo que estos mostrarán el único atributo que tiene esta clase en la plataforma.
 - data: atributo que contiene la fecha de creación de un elemento. Este atributo será usado internamente por el sistema para transformarlo a un formato relativo en el que se diga cuánto tiempo hace que se ha hecho la publicación en vez de la fecha exacta en la que se ha hecho.

9.4 Modelo de comportamiento

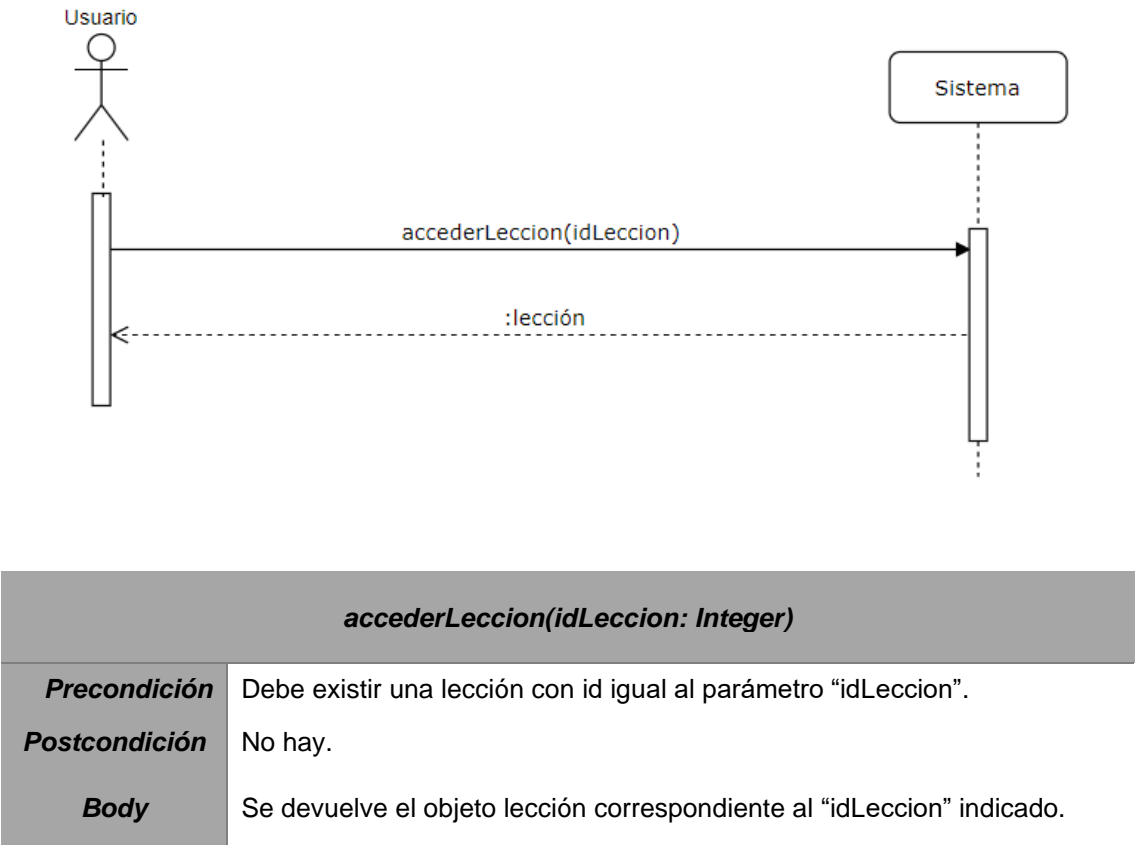
Para terminar, el último apartado de esta especificación de requisitos consiste en desarrollar un modelo de comportamiento para todos los casos de uso de la aplicación web para definir la interacción entre los actores y el sistema y sus consecuencias.

Por tanto, caso por caso, veremos un diagrama de secuencia UML con las acciones que emprenden los actores sobre el sistema y, en un contrato de operación, las condiciones anteriores y posteriores que deben cumplirse, y el resultado generado.

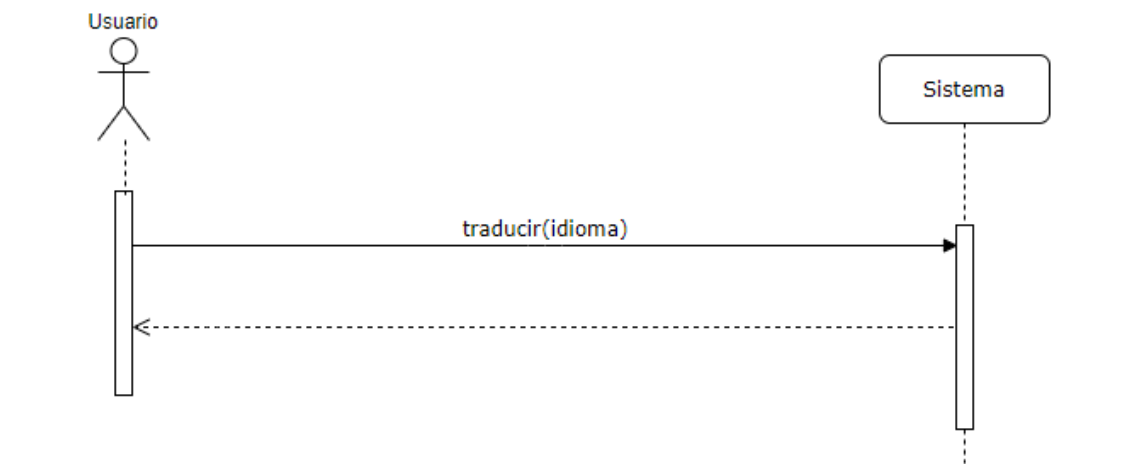
Filtrar lecciones



Acceder a una lección

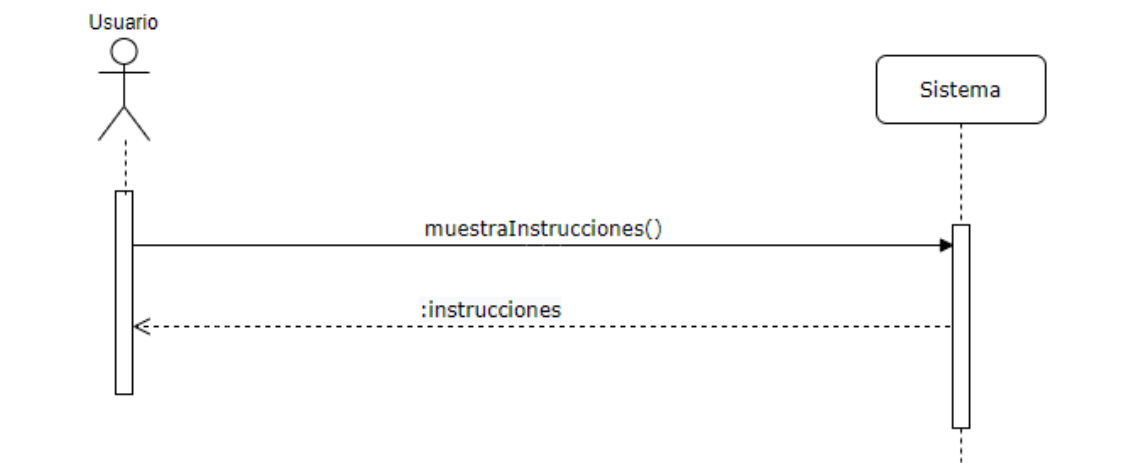


Cambiar el idioma de la plataforma



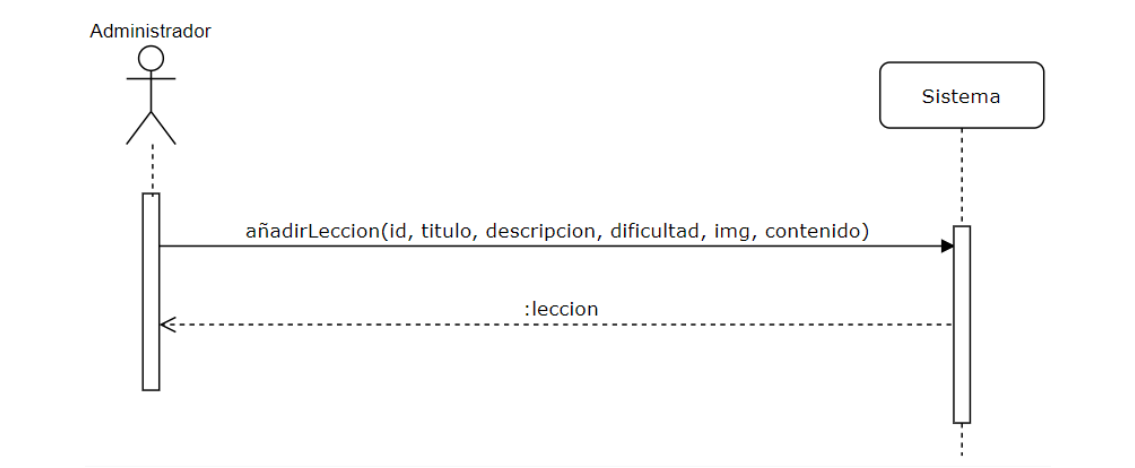
traducir(idioma:String)	
Precondición	El idioma debe de estar registrado en los ficheros de traducción.
Postcondición	La interfaz ha de estar traducida al idioma indicado.
Body	No hay.

Acceder a las instrucciones



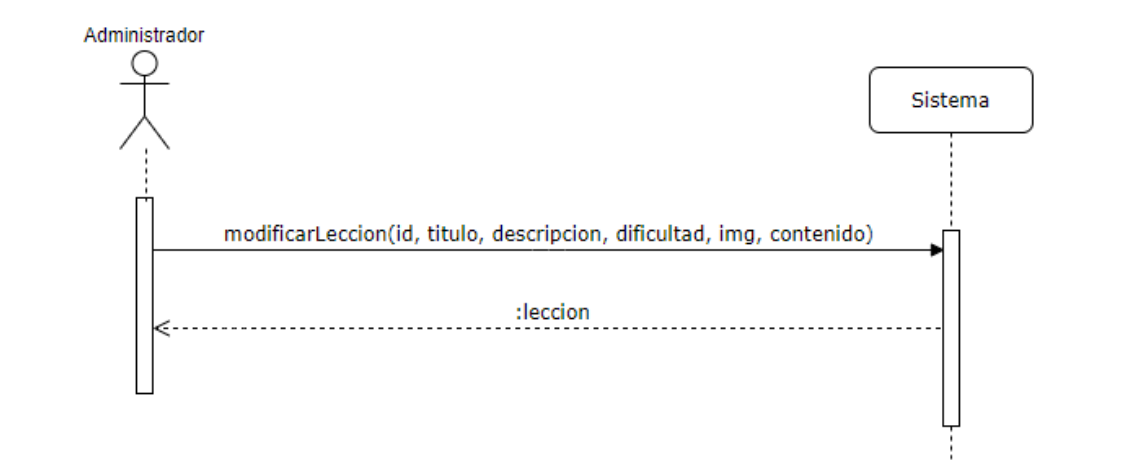
muestraInstrucciones()	
Precondición	No hay.
Postcondición	No hay.
Body	Se devuelve el objeto instrucciones.

Añadir lección



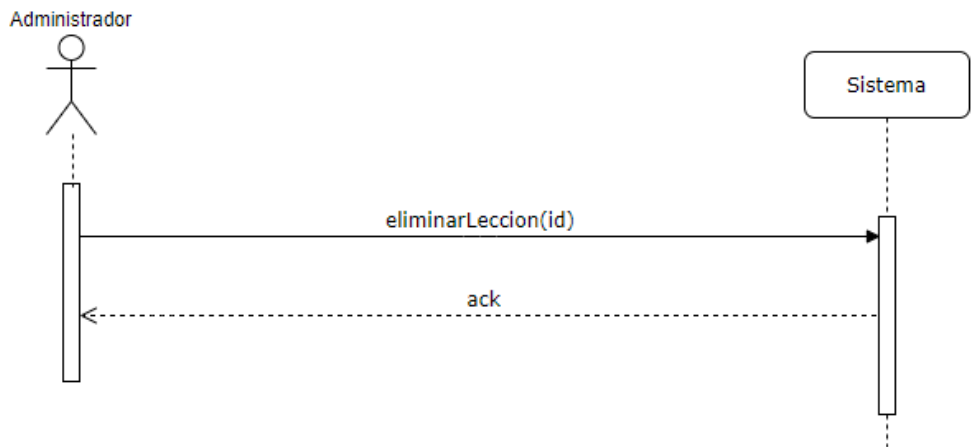
<i>añadirLeccion(id: Integer, titulo: String[3], descripción: String[3], dificultad: Integer, img: String, contenido: Leccion)</i>	
<i>Precondición</i>	No debe existir una lección el id indicado. El parámetro “contenido” debe cumplir con la estructura acordada.
<i>Postcondición</i>	Debe existir la lección creada.
<i>Body</i>	Devuelve la lección nueva creada.

Modificar lección



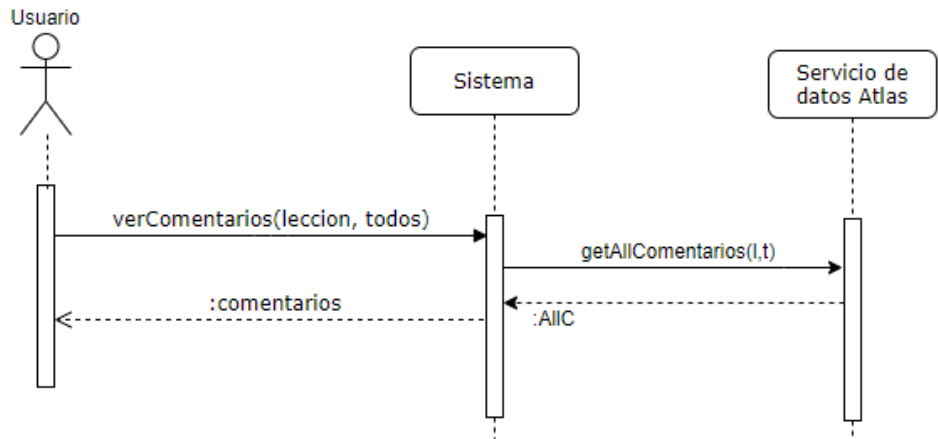
<i>modificarLeccion(id: Integer, titulo: String[3], descripción: String[3], dificultad: Integer, img: String, contenido: Leccion)</i>	
<i>Precondición</i>	Debe existir una lección el id indicado. Los parámetros han de cumplir con los tipos y estructuras determinados.
<i>Postcondición</i>	La lección será modificada con los parámetros que no eran nulos.
<i>Body</i>	Devuelve la lección con los cambios aplicados.

Eliminar lección



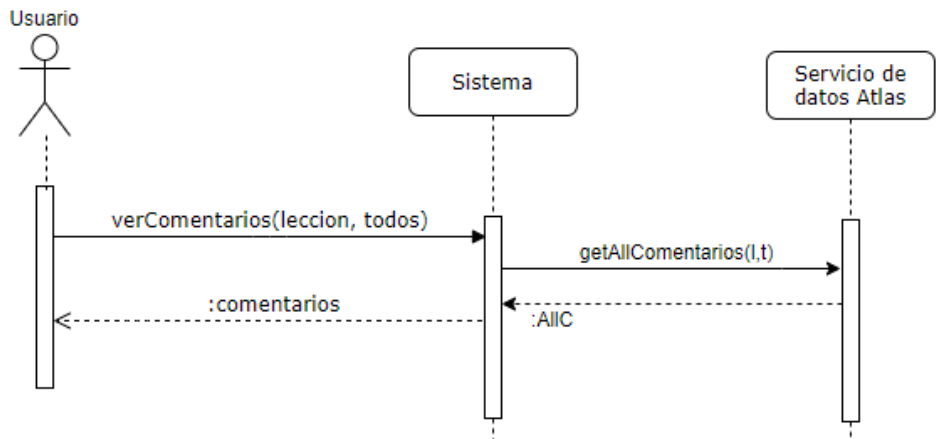
<i>eliminarLeccion(id: Integer)</i>	
Precondición	Debe existir una lección el “id” indicado.
Postcondición	La lección es eliminada del sistema.
Body	Devuelve un mensaje de confirmación.

Ver últimos comentarios



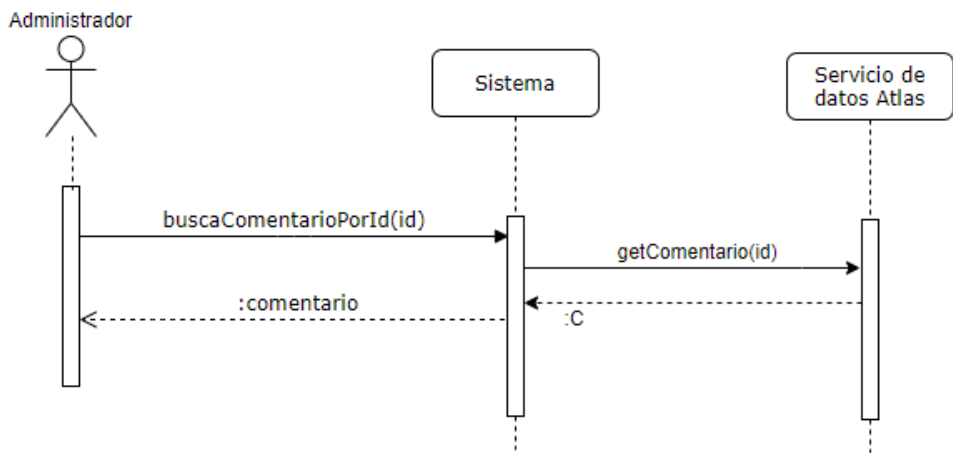
<i>verComentarios(leccion: Integer, todos: Boolean)</i>	
Precondición	Debe existir la lección indicada. El parámetro “todos” debe de ser falso.
Postcondición	No hay.
Body	El sistema de gestión de base de datos devuelve los últimos 15 comentarios de esa lección y su puntuación media global.

Ver todos los comentarios



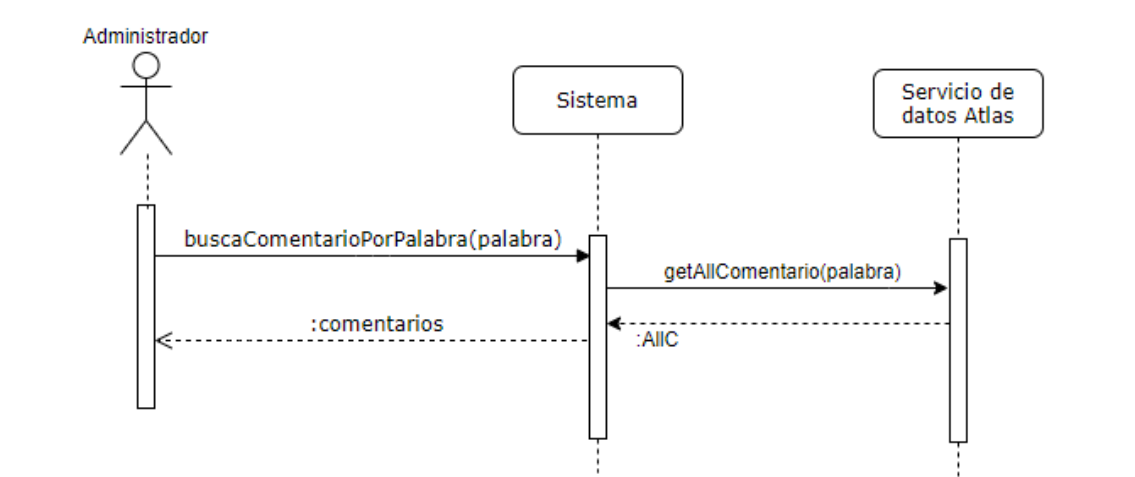
<i>verComentarios(leccion: Integer, todos: Boolean)</i>	
Precondición	Debe existir la lección indicada. El parámetro “todos” debe de ser verdadero.
Postcondición	No hay.
Body	El sistema de gestión de base de datos devuelve y su puntuación media global.

Buscar comentario por id



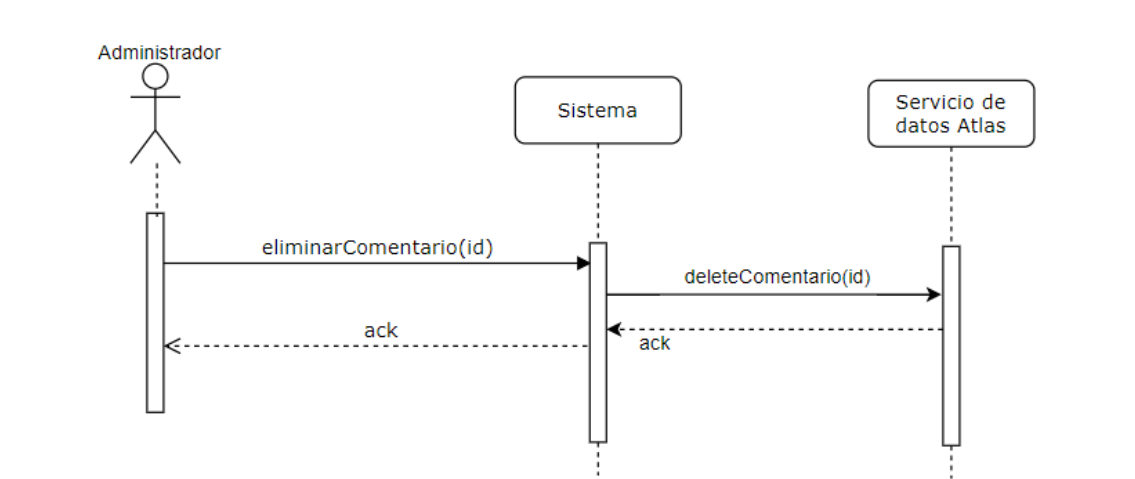
<i>buscarComentarioPorId(id: String)</i>	
Precondición	En la base de datos existe un comentario con ese “id”.
Postcondición	-
Body	Se devuelve el comentario con “id” coincidente.

Buscar comentario por palabra



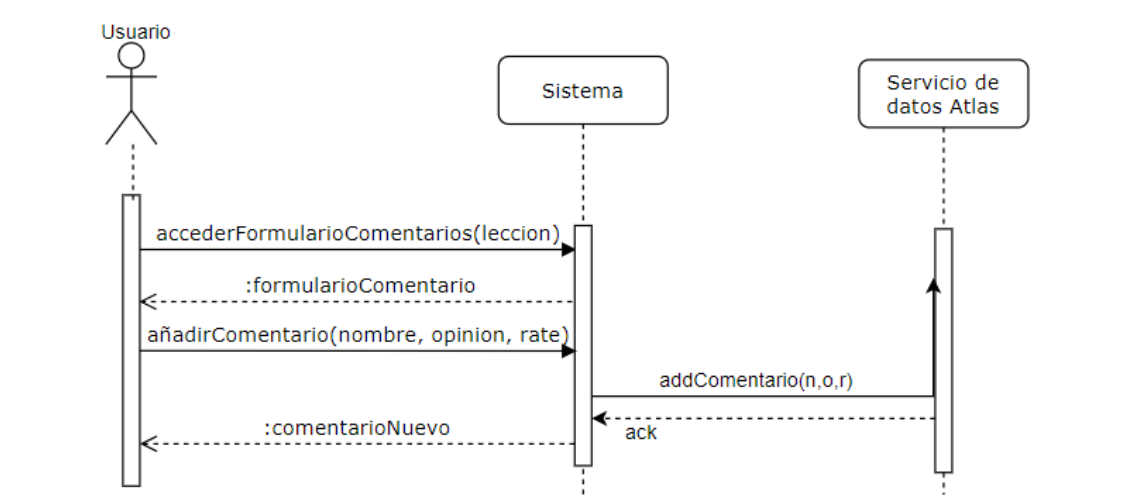
<i>buscarComentarioPorPalabra(palabra: String)</i>	
Precondición	-
Postcondición	-
Body	Se devuelve un conjunto con todos los comentarios cuyo nombre de usuario o texto contengan la “palabra” especificada.

Eliminar comentario



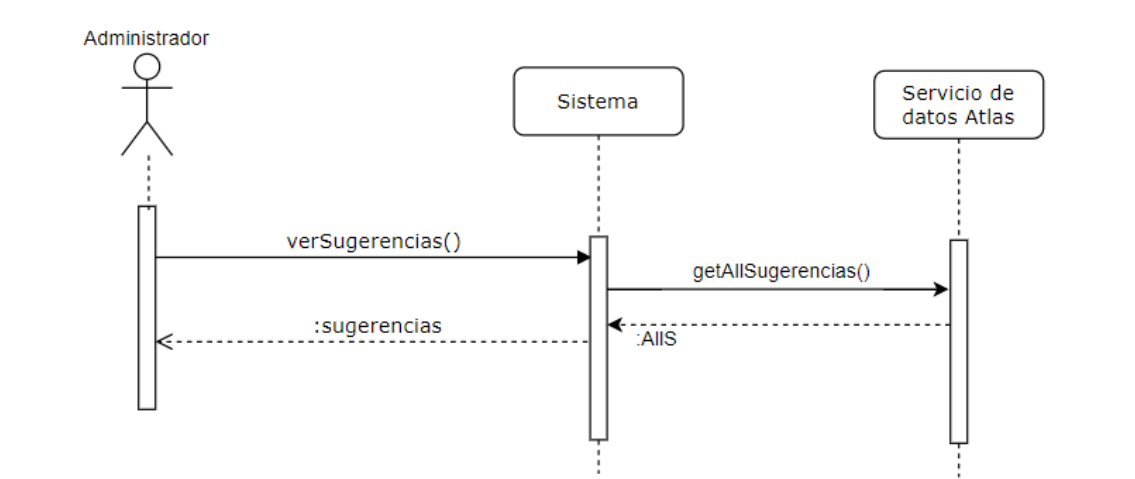
<i>eliminarComentario(id: String)</i>	
Precondición	En la base de datos existe un comentario con ese “id”.
Postcondición	El gestor de la base de datos ha eliminado el comentario de la base de datos.
Body	Se devuelve una confirmación del proceso.

Añadir comentario



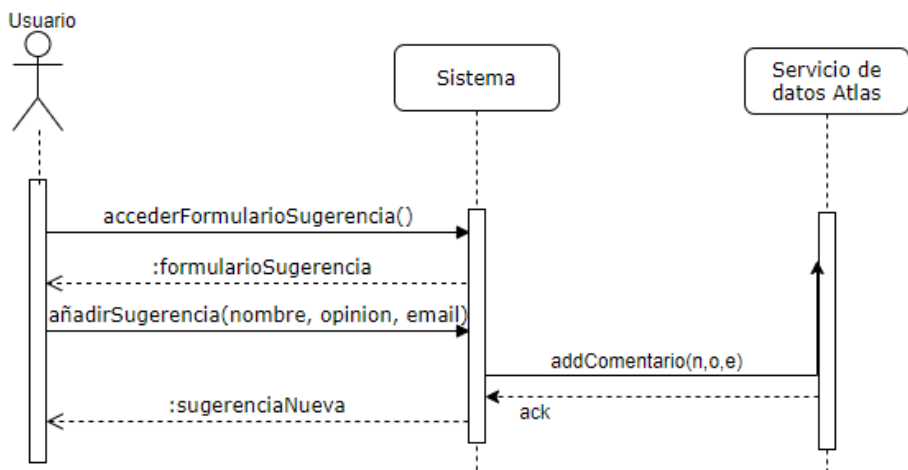
<i>accederFormularioComentarios(lección: Integer)</i> <i>añadirComentario(nombre:String, opinión: String, rate: Integer)</i>	
Precondición	Ningún parámetro ha de ser nulo.
Postcondición	El comentario es añadido a la base de datos.
Body	Para la primera petición se devuelve el formulario de envío de comentarios. Para el segundo se devuelve el comentario nuevo creado.

Ver todas las sugerencias



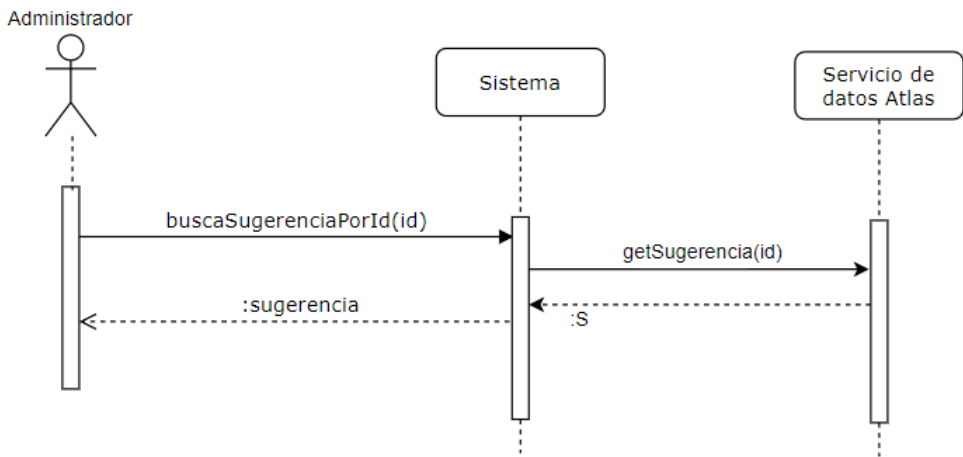
<i>verSugerencias()</i>	
Precondición	No hay.
Postcondición	No hay.
Body	Se devuelven todas las sugerencias de la base de datos.

Añadir sugerencia



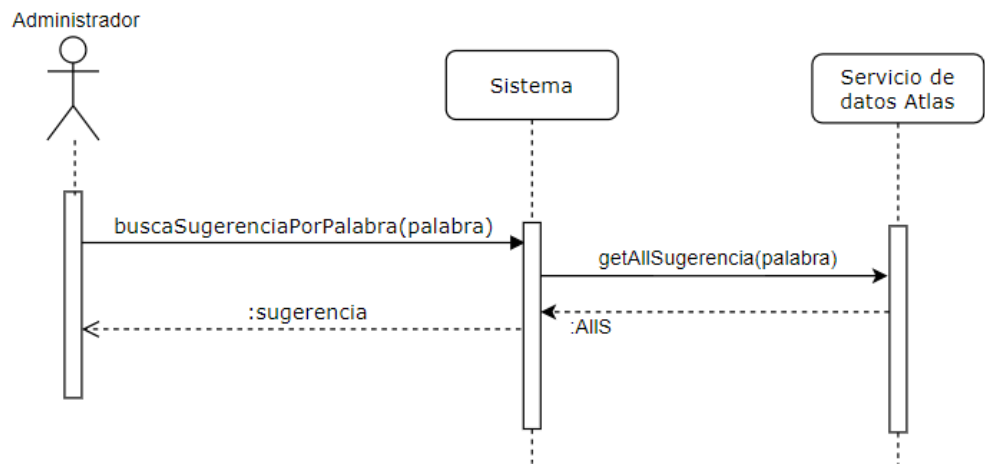
<i>accederFormularioSugerencia()</i> <i>añadirSugerencia(nombre:String, opinión: String, email: String)</i>	
Precondición	El nombre y la opinión no pueden ser nulas.
Postcondición	La sugerencia es añadida a la base de datos.
Body	La primera petición devuelve el formulario de envío de sugerencias. La segunda petición devuelve la sugerencia nueva creada.

Buscar sugerencia por id



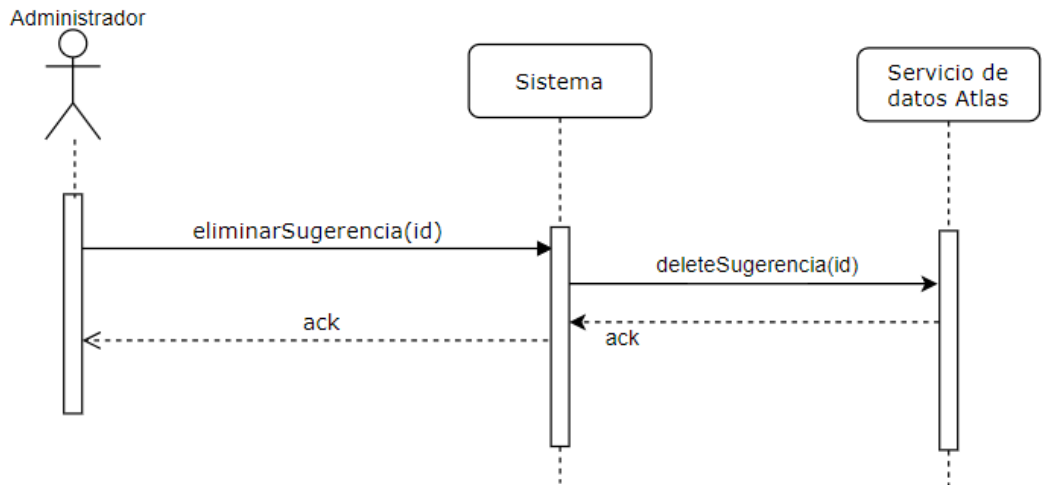
<i>buscarSugerenciaPorId(id: String)</i>	
Precondición	En la base de datos existe una sugerencia con ese "id".
Postcondición	-
Body	Se devuelve la sugerencia con un identificador coincidente.

Buscar sugerencia por palabra



<i>buscarSugerenciaPorPalabra(palabra: String)</i>	
Precondición	-
Postcondición	-
Body	Se devuelve un conjunto con todas las sugerencias cuyo nombre de usuario, texto o correo electrónico contengan la palabra especificada.

Eliminar sugerencia



<i>eliminarSugerencia(id: String)</i>	
Precondición	En la base de datos existe una sugerencia con ese “id”.
Postcondición	El gestor de la base de datos ha eliminado la sugerencia de la base de datos.
Body	Se devuelve una confirmación del proceso.

10. Diseño de la plataforma web y de su contenido

En el apartado anterior se ha completado una especificación sumamente detallada de los requisitos, tanto funcionales como de calidad, que han sido directa o indirectamente exigidos por el tipo de usuario al que nos dirigimos. Ahora es el momento de definir los componentes y entidades que forman el sistema, su estructura y su funcionamiento interno para que sean capaces de funcionar tal y como se ha prometido que lo harían de cara a los casos de uso vistos antes.

10.1 Arquitectura lógica

Lo primero que haremos será definir el tipo de arquitectura lógica que estructurará nuestro sistema. Recordemos que tenemos una serie de componentes que queremos que muestren una información y que dicha información a veces requiere de un procesamiento previo. También hay que tener en cuenta que no todos los datos mostrados proceden del mismo sitio. Mientras que algunos son estáticos (o no suelen cambiar) como lo es lo que se va a enseñar en cada lección, existen otro tipo de datos constantemente modificados como lo son los comentarios de los usuarios o sus sugerencias que es muy probable que procedan de un servidor externo que aloje una base de datos. Además, en estos últimos, hemos de distinguir una forma de acceso entre el usuario corriente y el administrador que eran nuestros dos principales actores. Dadas todas estas características, lo más lógico parece ser adoptar una arquitectura lógica en 3 capas [38] que nos permita dividir independientemente las funcionalidades a la hora de programar: una capa de presentación que se encargue exclusivamente de mostrar el contenido, una capa de dominio que valide y calcule los datos y que mantenga la consistencia de los mismos, y una capa de gestión de datos que sea la fuente de la información extraída, procesada y mostrada.

Se ha elegido esta arquitectura por dos sencillas razones. La primera es porque una arquitectura de esta índole permite centralizar el control del servicio en un único lugar. La plataforma web yayOS no requiere ni de microservicios ni de un sistema distribuido. Nos basta con un elemento central que gestione todas las peticiones que le lleguen. Y, en segundo lugar, este tipo de distribución es más favorable en términos de escalabilidad y, puesto que los requisitos pueden cambiar en cualquier momento, eso es algo imprescindible para este proyecto.

Sin embargo, recordemos que los datos a gestionar pueden ser estáticos, y proceder de un sistema de ficheros, o ser consultados en una base de datos. Esta dualidad, cuyas implicaciones veremos con más profundidad en el siguiente apartado, hacen necesario recalcar dos estructuraciones distintas dentro de la propia capa puesto que los componentes que llevarán a cabo las tareas de computación no serán los mismos. Esta

decisión puede parecer contraproducente. Teniendo ya una base de datos, ¿por qué no utilizarla también para almacenar las lecciones? Si bien es cierto que incluir el material lectivo en la base de datos simplificaría el sistema y lo volvería más cambiante, se ha optado por separarlos debido a que de esta forma el acceso de los usuarios al contenido de la web está siempre asegurado. A fin de cuentas, la base de datos que usaremos será proveída por un agente externo (lo veremos más adelante) y eso implica que un fallo por su parte afecte negativamente a las funcionalidades de yayOS. Si separamos las lecciones de esta base de datos no solo evitamos que esto suceda, si no que abrimos la oportunidad de hacer cualquier migración u operación por el estilo sin comprometer el acceso a la funcionalidad principal que es la de que los usuarios usen la plataforma web para aprender.

Podemos ver una representación gráfica de la arquitectura en la Figura 29.

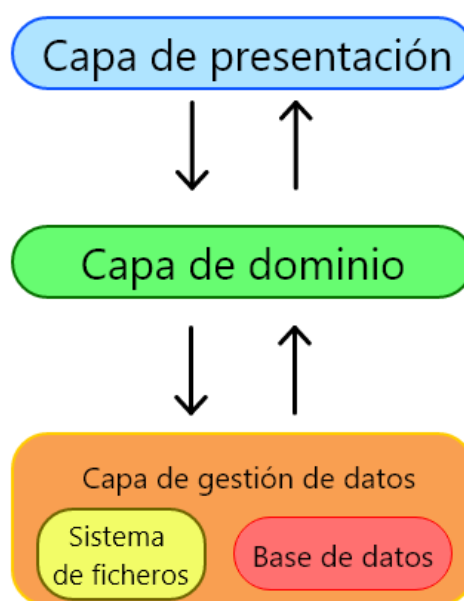


Figura 29: Arquitectura lógica de 3 capas del sistema. Fuente: Elaboración propia.

10.2 Arquitectura física

La arquitectura en 3 capas es una especialización de las arquitecturas cliente-servidor y, puesto que el producto final de este trabajo es el de hacer una aplicación web, la arquitectura física en la que se implementará el diseño del apartado anterior ha de estar enfocada de manera que existan dos figuras que se comuniquen la una con la otra: un proveedor de un servicio y un usuario demandante.

Distribuiremos las funcionalidades mediante distintos elementos físicos a los que llamaremos “nodos”. Por definición, un nodo es una representación de un recurso computacional con memoria y capacidad de procesamiento [39]. Por tanto, en la arquitectura física que vamos a definir, los nodos son sus piezas clave. Sus engranajes. Ellos tendrán la capacidad de ejecutar los distintos componentes y dispondrá de

conexiones con otros nodos con los cuales se pasarán información. Y aquí aparece otro concepto, el de componente. Definamos componente como una parte del sistema que tiene un rol en concreto.

Empecemos por identificar los distintos nodos que conformaran la arquitectura física y, para cada uno, los componentes por los que está formado. Todo empieza con un usuario que interactúa con la capa de presentación mediante inputs y outputs. Esto lo hace gracias al navegador que tendrá en su computadora el cual tendrá la capacidad de gestionar la capa de presentación y de recibir y presentar los datos que le sean entregados. Como se ha dicho, el usuario introducirá inputs en la interfaz y estos serán procesados por la lógica de la capa de presentación que estará alojada en un servidor web que, para distinguirlo, denominaremos como “Front Server”. Si la petición está relacionada con la visualización del contenido de las lecciones, lo único que tendrá que hacer la lógica de la capa de presentación es extraer la información que necesite del sistema de ficheros que se encuentre en el mismo Front Server.

Si, por el contrario, la petición es de algún caso de uso relacionado con los comentarios, sugerencias o similares, entonces la lógica de la capa de presentación tendrá que transformar la petición que reciba en una petición HTTP a una API almacenada en otro servidor web que denominaremos como “Back Server”. Por último, el Back Server analizará la petición y la transformará una vez más y solicitará al servidor de datos que contiene la base de datos que le proporcione la información que requiera o que le permita hacer las modificaciones pertinentes.

En la Figura 30 puede observarse a modo de resumen un diagrama UML de despliegue de la arquitectura física del sistema.

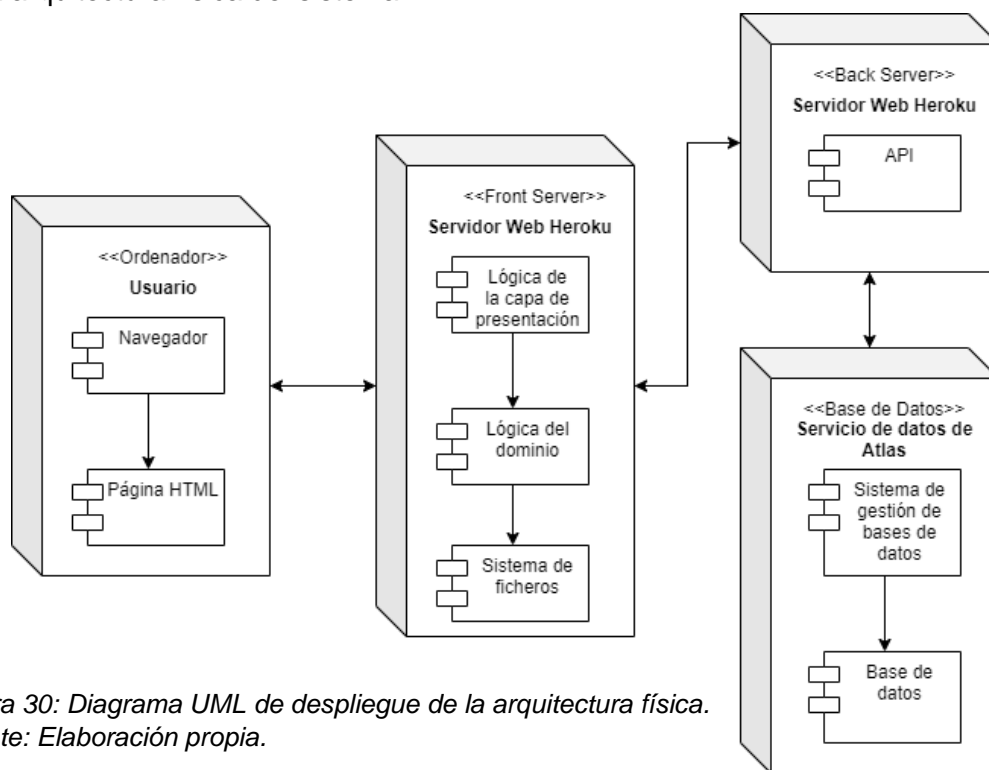


Figura 30: Diagrama UML de despliegue de la arquitectura física.
Fuente: Elaboración propia.

10.3 Descripción interna de las capas

En este apartado trataremos en profundidad cada una de las capas en cada uno de los componentes en los que se encuentran. También se incluirá un ejemplo de aplicación de caso de uso mediante un diagrama de secuencia en el que veremos cómo es la interacción de una petición por todas las partes que componen la arquitectura física.

10.3.1 Diseño de la capa de presentación

El diseño de la capa de presentación va a depender directamente de los requisitos. Las vistas deberán tener una composición y distribución que permita alojar todas las funcionalidades exigidas y deberá respetar en todo momento los requisitos de calidad que, recordemos una vez más, son de vital importancia para el usuario al que nos dirigimos.

Al definir la arquitectura lógica del sistema se ha especificado que el único deber de la capa de presentación es, valga la redundancia, presentar la información. Es por ello que a nivel de diseño lo único que nos preocupa es la apariencia de la interfaz de la plataforma. Así que, a continuación, introduciremos algunos mockups¹² diseñados con la herramienta Adobe XD de la que ya se ha hecho mención en apartados anteriores.

Vista del Menú principal



Figura 32: Diseño preliminar del menú principal de yayOS. Fuente: Elaboración propia

¹² **Mockup:** en diseño, es considerado como una maqueta detallada del diseño de un producto. En este caso, de las pantallas o vistas de una web.

En la Figura 32 se observa el menú principal. Es la vista más importante porque será la primera que cualquier persona verá al acceder a yayOS. Su diseño es clave, ya que debe captar al usuario desde el primer momento.

Se apostará por una estructura tradicional formada por una cabecera (*header*), una marquesina (*slider*), el contenido propio que será una lista con todas las lecciones disponibles (*body*) y, finalmente, un pie de página (*footer*).

Se ha elegido una temática de colores que contraste lo suficiente y que, psicológicamente, tengan el efecto deseado en el usuario. Mientras que los tonos anaranjados ofrecen una sensación de confianza, de éxito y de valentía, los tonos oscuros transmiten seguridad y sofisticación [40]. Como se puede observar se respetan todos los requisitos de apariencia, estilo y de usabilidad acordados. En este caso es imposible no hacer uso del desplazamiento vertical puesto que se tiene que mostrar mucho contenido y se ha preferido hacerlo de forma espaciada. Sin embargo, se incluirá en la cabecera y/o en las instrucciones un cartel de advertencia explicando que debe desplazarse verticalmente (hacer *scroll*) para ver todo el contenido.

Desde la cabecera se podrá cambiar el idioma de la plataforma y, desde esta vista, se podrá acceder a todas las que van a verse a continuación.

Vista de una lección



Figura 33: Diseño preliminar de una lección de yayOS. Fuente: Elaboración propia

Desde el menú principal podemos acceder a cualquier lección. Para las lecciones se apostará por el diseño de la Figura 33 y estarán divididas en apartados acorde al requisito no funcional #25 y, para respetar el número #26, se presentará solamente un apartado siendo, por defecto, el primero. El usuario podrá navegar ya sea por las flechas inferiores hacia el siguiente o anterior apartado o bien a uno en concreto a través del menú de la barra lateral que estará formado por botones con enlaces a cada uno de los apartados. Se mantiene la misma temática tanto de apariencia como de estilo.

Vista de los comentarios y de crear un comentario



Figura 34: Diseño preliminar de la sección de comentarios de una lección de yayOS.
Fuente: Elaboración propia.

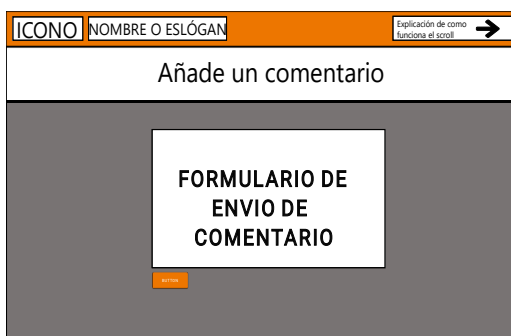


Figura 35: Diseño preliminar de la sección de envío de comentarios de yayOS.
Fuente: Elaboración propia.

Desde el menú principal podemos acceder indistintamente a la vista de la lección como a la de sus comentarios. En la Figura 34 vemos un esbozo de cómo sería. En ella se verían los últimos comentarios registrados y de ahí podríamos activar el caso de uso de ver todos los comentarios. De nuevo el desplazamiento vertical es inevitable. En la parte superior podemos ver la calificación global actual de la lección. De esta vista podemos ir a la de crear un comentario, cuyo diseño queda reflejado en la Figura 35. Simplemente es un formulario que hay que rellenar. Al enviar, regresamos a la vista de comentarios.

Se ha evitado generar el formulario de envío de comentarios en la misma vista de su visualización (como suele hacerse) para respetar el criterio de aceptación del requisito no funcional #24.

Vista de instrucciones de uso y envío de sugerencias



Figura 36: Diseño preliminar de las instrucciones de uso de yayOS.
Fuente: Elaboración propia.

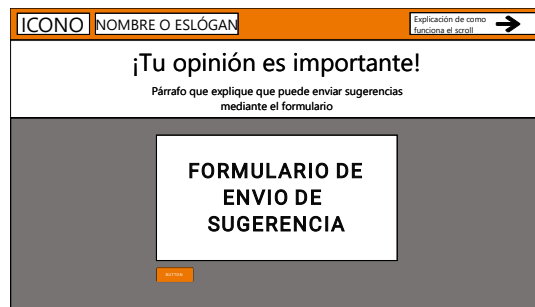


Figura 37: Diseño preliminar del formulario de sugerencias de yayOS.
Fuente: Elaboración propia.

Por último, desde el menú principal se podrá acceder también a las instrucciones de uso (Figura 36) y al formulario de envío de sugerencias (Figura 37). Para el primero se optará por una descripción rápida de cómo acceder y cómo usar las principales funcionalidades

que ofrece la plataforma. En el segundo será un formulario sencillo muy similar al de enviar comentarios. La única diferencia serán los campos a rellenar como ya se ha podido observar en el modelo conceptual de la especificación de requisitos. De nuevo, ambas vistas respetan todos los criterios de aceptación de los requisitos no funcionales pertinentes.

Teniendo en cuenta todas estas pantallas y como se navega de unas a otras por el sistema, obtenemos el siguiente diagrama navegacional (Figura 38). Los nombres de la flecha son una aclaración de la acción o intención del usuario que provoca el cambio de vista. Se espera que dichos cambios se hagan mediante clics en botones o elementos similares.

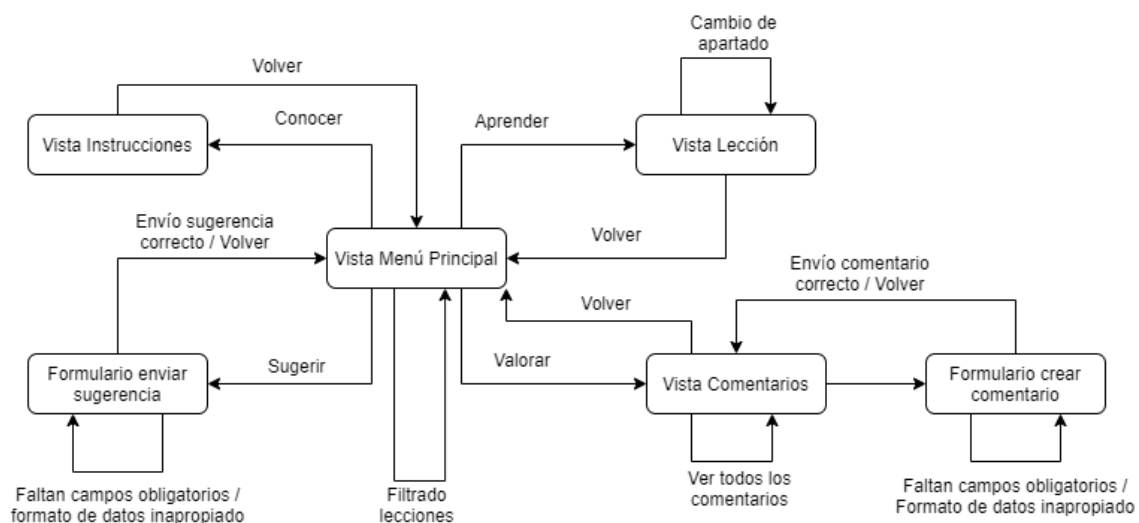


Figura 38: Mapa navegacional de las vistas del sistema. Fuente: Elaboración propia

10.3.2 Diseño de la capa de dominio

En esta capa se encontraría toda la lógica encargada de procesar la información recibida de la capa de gestión de datos para poder adecuarla al formato y propósito requerido por la capa de presentación. Es, por tanto, un puente entre ambos extremos.

Como se ha podido determinar a partir de la arquitectura física, a veces se limitará a trabajar dentro del sistema de ficheros del propio Front Server y, en otros casos, tendrá que comunicarse con el Back Server para acceder a los datos alojados en la base de datos. Sin embargo, este acceso se hará mediante métodos que tienen su origen en la lógica de la capa de presentación y que van deviniendo en más y más profundidad. En la Figura 39 podemos ver las clases y los atributos del sistema que serán punto de partida de todas las operaciones lógicas.

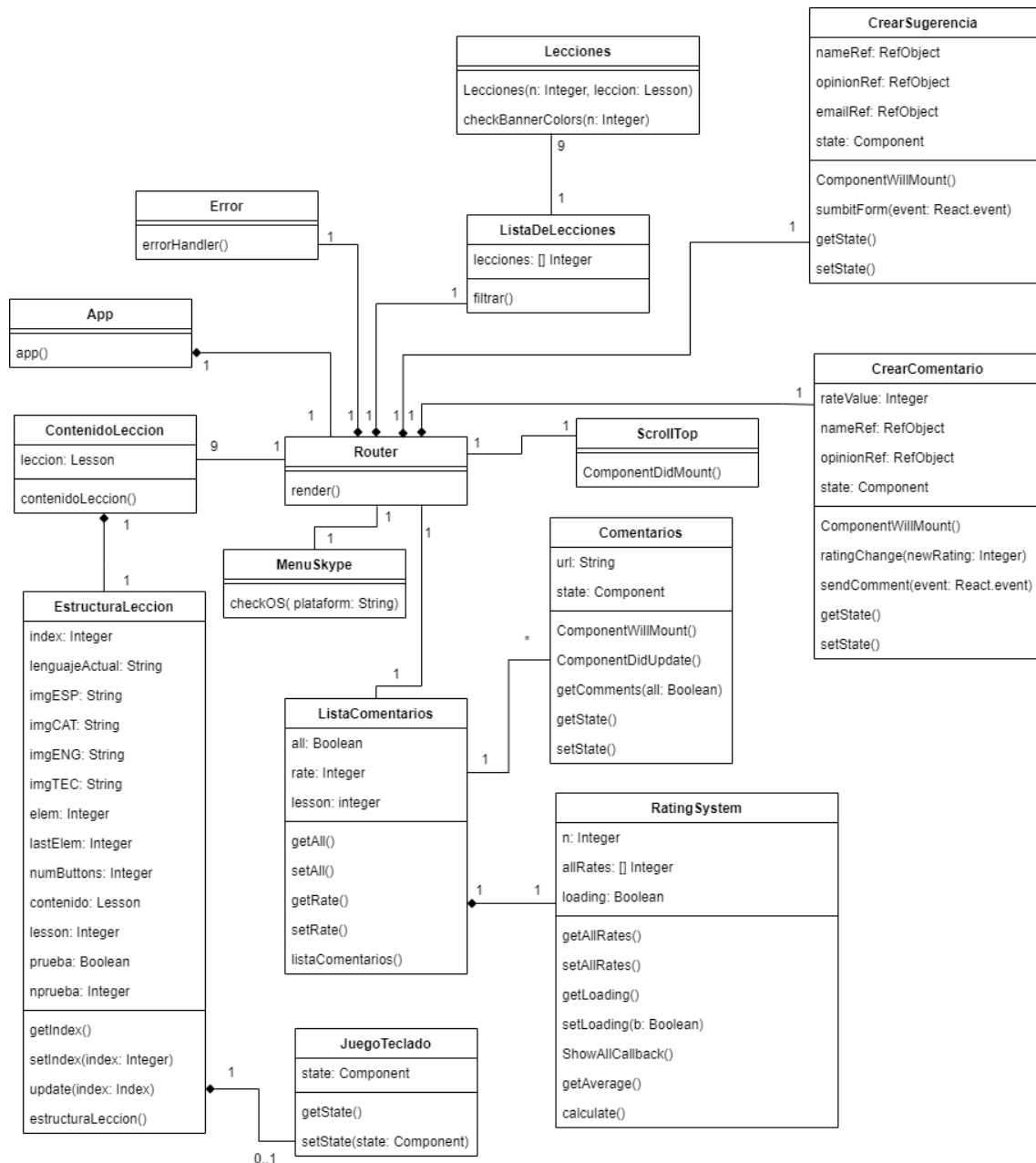


Figura 39: Diagrama de clases UML del dominio del sistema. Fuente: Elaboración propia.

10.3.3 Diseño de la capa de datos y de la conexión al servidor

Por último, veamos el diseño interno de la capa de gestión de los datos. En la descripción de la arquitectura física se ha hecho suficiente hincapié en la diferencia que suponen las dos fuentes de datos que maneja el sistema. El primer grupo engloba la totalidad del contenido de las lecciones, es decir, el material visual que se mostrará al usuario. Volviendo atrás podemos observar cómo, en la planificación temporal, el diseño estético de este contenido requería una cantidad de horas considerable y no es para menos. Es el principal motivo por el cuál esta plataforma existe.

Recordemos como en el modelo conceptual se había visto que una lección contiene una serie de apartados. Es necesario distinguir aquellos elementos que poseen una funcionalidad o, dicho de otra forma, que permiten una interacción con el usuario, y los que no. Técnicamente hablando, distinguir los elementos de la subclase Ap. Estático del diagrama de la Figura 28.

Empecemos por estos puesto que son los más sencillos. Cada apartado no funcional es una diapositiva con una serie de enseñanzas de un tema en concreto. Su elaboración, al igual que los mockups de la capa de presentación, se ha hecho mediante la herramienta Adobe XD. En la Figura 40 vemos un ejemplo:

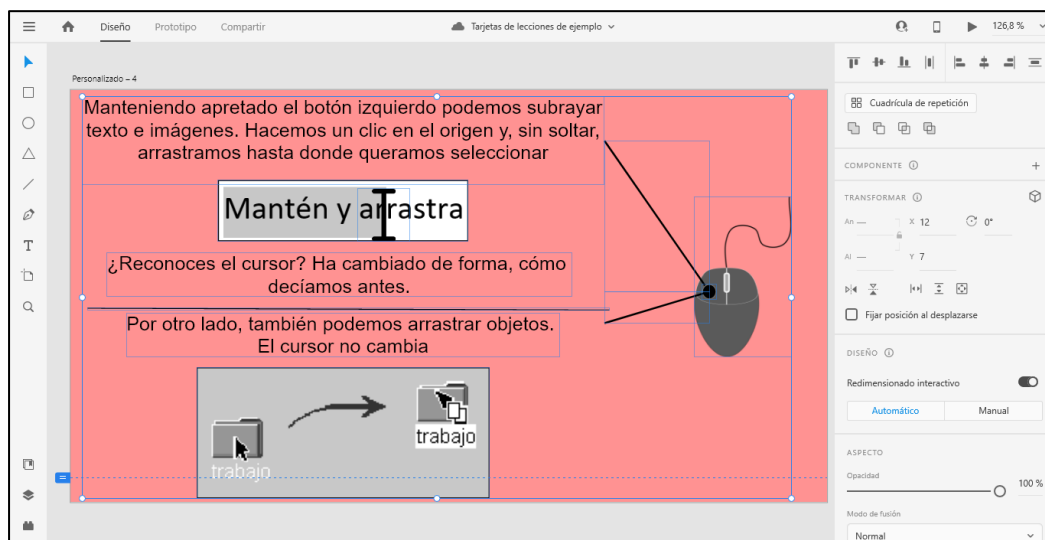


Figura 40: Ejemplo del proceso de creación del contenido de la lección “Teclado y Ratón” de yayOS con Adobe XD. Fuente: Elaboración propia

El proceso de creación de datos estáticos es siempre el mismo. Para cada lección se ha de estructurar primero el contenido que se pretende mostrar, luego llevar a cabo un breve estudio del estado actual de la disciplina o del uso de una herramienta en particular, idear las explicaciones, transformarlas a un lenguaje básico de entendimiento, diseñar e incluirlas en las diapositivas, hacerlas estéticamente atractivas, y asignarla como un apartado de la lección. Por último, se debe traducir el contenido para que esté disponible en todos los idiomas.

Esto en cuanto a su diseño estético. En cuanto a su diseño interno como dato, la estructura de toda lección debe seguir un formato en concreto. Se ha elegido el lenguaje JSON como representación del formato de los datos por lo que, al ser información en un sistema de ficheros, estarán directamente escritos en este lenguaje. De esta forma, además, se define ya un formato concreto por si, en el futuro, se quiere permitir que otro rol que no sea el de administrador pueda modificar las lecciones.

Toda lección debe de estar representada por los siguientes atributos:

- Un identificador que sea un número entero.
- El título de la lección que se verá en la interfaz.
- El número de apartados que tiene la lección.

- Un Booleano que indique si contiene una prueba interactiva (un apartado no estático).
- El número de apartado en el que se encuentra la prueba interactiva. Si el atributo booleano es falso, este atributo será igual a nulo.
- Un vector con tantos objetos JSON como indique el “número de apartados” de manera que cada uno representa una de las partes de la lección. Cada objeto JSON contendrá:
 - El título del apartado de la lección.
 - La diapositiva (imagen en PNG) de ese apartado en castellano.
 - La diapositiva (imagen en PNG) de ese apartado en catalán.
 - La diapositiva (imagen en PNG) de ese apartado en inglés.

Las imágenes serán, a su vez, otros ficheros que formarán parte del sistema de ficheros que se encuentra en el Front Server. Serán representados mediante objetos que contendrán sus importaciones y dichos objetos serán asignados a estos campos de los objetos JSON. El apartado que contenga un juego interactivo no contendrá imágenes. La lógica de la capa de presentación será la encargada de determinar si debe de cargar las imágenes o los componentes interactivos que pertoqueen. Dichos componentes, que no dejan de ser fragmentos de código, también se encontrarán en el sistema de ficheros del Front Server. De nuevo, es más práctico almacenarlos así que no en base de datos en un servidor y acceder a ellos cada vez.

En segundo lugar, tenemos los datos alojados en la base de datos. Se accederá a ellos mediante peticiones a una API alojada en el Back Server. La mayoría cumplen las necesidades básicas de los casos de usos de la especificación de requisitos. Aunque habrá otras que satisfarán necesidades internas de la plataforma.

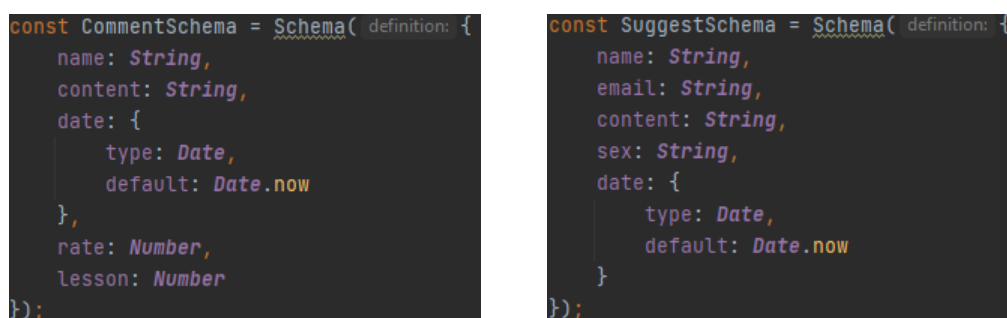
- **Comentarios**
 - Por parte del usuario
 - Petición que devuelve todos los comentarios de la base de datos o los últimos si la ruta contiene la clave opcional “last”.
GET /comment/:lesson/:last?
 - Petición que inserta un comentario nuevo en la base de datos.
POST /save
 - Por parte del administrador
 - Petición que devuelve el comentario asociado al id pasado como parámetro.
GET /comment/:id
 - Petición que devuelve todos los comentarios que contienen la palabra pasada como parámetro.
GET /search/:search
 - Petición que elimina un comentario de la base de datos.
DELETE /comment/:id
 - Por parte del sistema
 - Petición que retorna todas las valoraciones de una lección.
GET /rates/:lesson

- **Sugerencias**

- Por parte del usuario
 - Petición que envía una sugerencia nueva a la base de datos.
POST /form/sabe
- Por parte del administrador
 - Petición que devuelve todas las sugerencias de la base de datos.
GET /form/suggestions
 - Petición que devuelve la sugerencia asociada al id pasado como parámetro.
GET /form/suggestions/:id
 - Petición que devuelve todas las sugerencias que contienen la palabra pasada como parámetro.
GET /form/search/:search
 - Petición que elimina una sugerencia de la base de datos
DELETE /form/suggestions/:id

Estas peticiones son gestionadas por la API para extraer la información de la base de datos. Como podía verse en la Figura 39, la base de datos está gestionada por el Servicio de Datos de Atlas [41] que es un servicio que permite almacenar datos en las nubes de distintos proveedores. Se ha elegido un esquema no relacional por lo que en ningún momento se está usando el formato de filas y columnas e índices relacionados entre tablas. En vez de eso, el uso de este tipo de arquitectura permite tener los datos como ficheros JSON con datos de la forma clave-valor, lo que facilitará mucho el diseño a posteriori si pretendemos trabajar con React que utiliza JavaScript. Además, se asemeja mucho al tipo de datos que trata la lógica de la capa de presentación con los datos estáticos obtenidos del sistema de ficheros del Front Server, por lo que así potenciamos la reutilización de código.

En la Figura 41 se pueden observar los esquemas que seguirán los comentarios y las sugerencias almacenados en la base de datos por los usuarios. Dado el alcance actual del proyecto, estos son los únicos datos almacenados por el momento. La relación entre ambas colecciones es nula, otra razón de más para evitar una base de datos relacional.



```
const CommentSchema = Schema( definition: {  
  name: String,  
  content: String,  
  date: {  
    type: Date,  
    default: Date.now  
  },  
  rate: Number,  
  lesson: Number  
});  
  
const SuggestSchema = Schema( definition: {  
  name: String,  
  email: String,  
  content: String,  
  sex: String,  
  date: {  
    type: Date,  
    default: Date.now  
  }  
});
```

Figura 41: Esquemas de Modelo de Comentarios y Sugerencias. Fuente: Elaboración propia.

Para poner punto final a este apartado, se adjunta en la Figura 42 un ejemplo de la interacción de todos los elementos de la arquitectura para el flujo de ejecución el caso de uso “Ver últimos comentarios”.

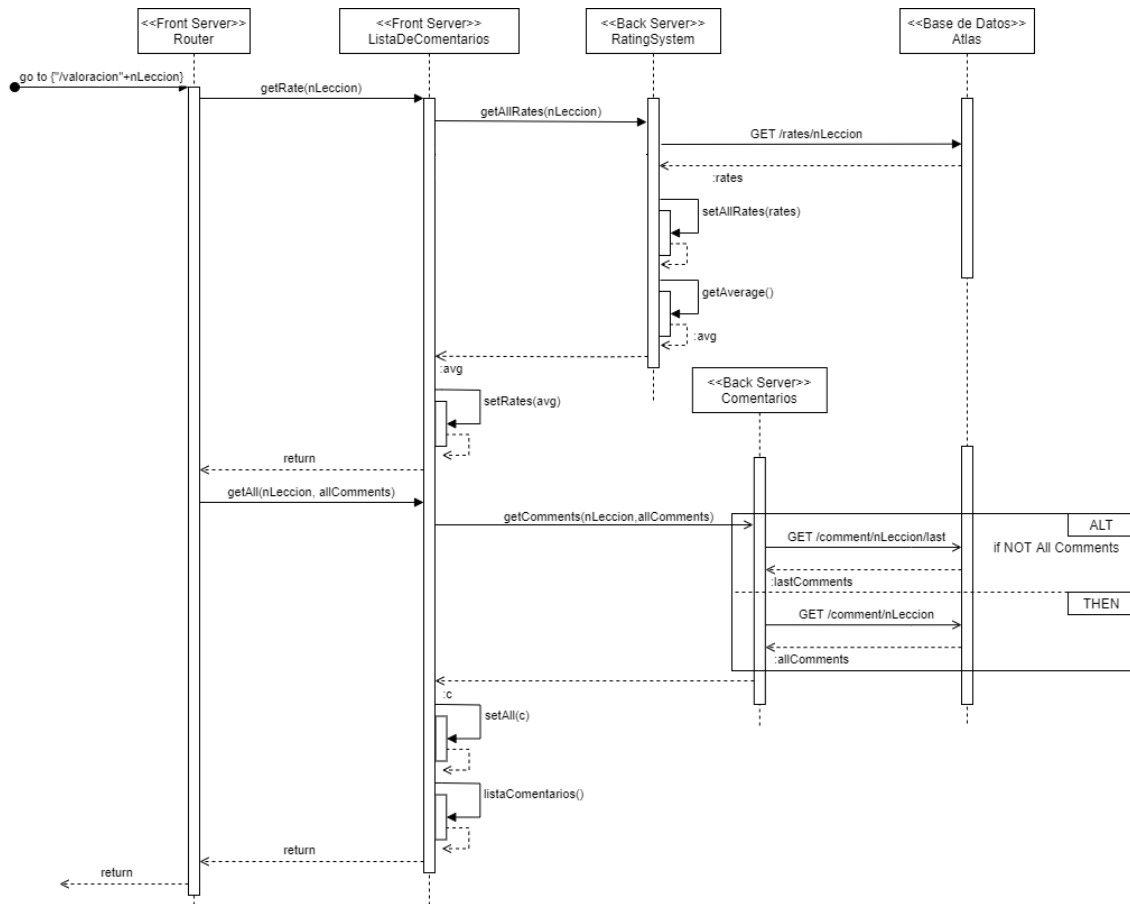


Figura 42: Diagrama de secuencia del caso de uso “Ver últimos comentarios”.
Fuente: Elaboración propia.

El usuario indica al sistema que quiere ver los últimos comentarios. El componente encargado de enlazar las vistas, que en el diagrama de clases ha sido denominado como “Router” (Figura 40), forma parte de la lógica de la capa de presentación y está alojado en el Front Server. Él será el encargado de comunicarse con los componentes pertinentes en el Back Server. Primero, pide la puntuación media de la lección que ha recibido como parámetro en el input. Recordemos que cuando se visualizan los comentarios se quiere ver también su puntuación media. Pide al componente “RatingSystem” del Back Server la puntuación media de la lección y ese, a su vez, hace una petición a la base de datos para que le dé todas las puntuaciones de todos los comentarios. Una vez se tiene este dato, se calcula la media y es devuelto.

A continuación, se piden los comentarios en sí, lo que degenera en otra comunicación con el Back Server, esta vez con el componente “Comentarios”, y en otra llamada a la base de datos. Observamos que el comportamiento varía en base al valor del atributo booleano allComments. Si es verdadero, se hace una llamada para obtener todos los comentarios. Como que en este caso solo queremos los últimos, el atributo será falso y, por tanto, entraremos en la primera condición.

Una vez se tienen todos los datos estos serán renderizados por la capa de presentación.

11. Implementación del sistema

En este capítulo de la memoria se detallará el proceso de implementación del sistema a partir de los requisitos y el diseño definido. Como ya se ha explicado en el capítulo de metodología de trabajo, el uso de Kanban permitirá seguir el flujo de tareas de la planificación temporal de manera lineal y haciendo pequeñas interrupciones cuando se produzca algún imprevisto o contratiempo. En principio, el orden estipulado de las tareas evita la necesidad puntual de desarrollar dos tareas distintas en paralelo porque sean dependientes entre sí.

A continuación, se empezará por explicar un poco cómo ha sido el proceso de aprendizaje de las tecnologías utilizadas y las herramientas a las cuáles se ha recurrido. Seguidamente, se analizará con detalle cómo se ha programado, qué patrones o enfoques se han seguido, qué decisiones se han tomado y qué motivos o razonamientos ha habido durante el proceso. Por último, se hablará sobre la estrategia seguida para desplegar el sistema siguiendo la arquitectura física concebida en el apartado de diseño.

11.1 Proceso de aprendizaje

Ya se ha hecho mención en varias ocasiones a lo largo de esta memoria: uno de los objetivos de este proyecto era que el autor del mismo adquiriera nuevos conocimientos que le sirvieran en el ámbito laboral y fue por ello que consideró que aprender a programar y diseñar aplicaciones web podía serle de utilidad y, más concretamente, a hacerlo con React.

No se partía completamente de cero en cuanto al lenguaje JavaScript pues durante la carrera sí que había llevado a cabo varios proyectos que lo habían requerido. Sin embargo, tanto el uso de JSX como la programación orientada a componentes de React eran algo completamente nuevo. También era bastante principiante en el trabajo en local, el despliegue de productos software, la construcción de una API, el uso de bases de datos no relacionales y en la instalación y utilización de librerías.

Se ha decidido que lo mejor era prescindir del contenido gratuito que alberga la red y acudir directamente a fuentes de confianza y que ofrezcan unas garantías mínimas. Es por ello que se ha accedió a un curso de pago, al cual ya se ha hecho referencia en apartados anteriores, y que es de la plataforma de aprendizaje en línea Udemy [42], la cual se dedica a la formación virtual tanto académica como profesional a nivel internacional. En él se explicaba paso a paso la elaboración completa de una página web primero en HTML, luego la maquetación en CSS y, por último, la transformación de la misma a React y a otros frameworks.

Se fueron siguiendo las indicaciones paso a paso y, al mismo tiempo, se fue adaptando lo aprendido a las necesidades de este proyecto y poco a poco la plataforma fue tomando forma. Con eso y la resolución espontánea de dudas en foros de Internet se

terminó por consolidar unos conocimientos básicos que permitían desarrollar la web sin problemas y profundizar en lo aprendido si era necesario y sin complicaciones.

Se calculan unas 6,10 horas visualizadas de tutoriales de React y otras 3,95 horas entre JSX y CSS. Por otro lado, se ha dedicado 2,78 horas de tutoriales para el desarrollo de la API y la preparación y conexión con la base de datos, y otras 2,48 horas para descubrir cómo sincronizar el frontend con el backend. Esto junto con las diversas prácticas realizadas hace un total superior a 30 horas, lo que sobrepasa el tiempo estimado en la planificación temporal. En el apartado de seguimiento se verá qué se hizo para ajustar esa desviación.

11.2 Implementación

Todos los ingredientes están listos: los requisitos que el sistema debe cumplir, los objetivos que tiene que alcanzar, el diseño al cual se debe asemejar, la planificación que se ha de seguir y los conceptos teóricos necesarios para llevarlo a cabo. Solo queda, por tanto, empezar a programar.

En este apartado se detallarán todas aquellas decisiones tomadas durante la fase de implementación y su motivo, junto con todos los elementos que han formado parte del proceso.

11.2.1 Una mezcla de lenguajes, frameworks y librerías



*Figura 43: Algunos de los lenguajes de programación, frameworks y librerías usados.
Fuente: Elaboración propia.*

La plataforma web yayOS ha sido desarrollada con el lenguaje de programación orientado a objetos JavaScript y, más concretamente, con una de sus librerías para crear interfaces más utilizadas: React. Se ha elegido esta opción porque es de código abierto, porque hoy en día está muy bien documentada y, por último, porque permite construir aplicaciones reactivas con datos dinámicos, es decir, que pueden cambiar en cualquier momento. Y esto es perfecto para yayOS ya que de esta manera se puede adaptar una interfaz que sirva de contenedor para toda la información que se quiere mostrar.

React trabaja con componentes que, a efectos prácticos, no dejan de ser partes lógicas que ocupan un trozo determinado de la pantalla o, dicho de otra forma, un elemento software visual de un tamaño concreto. A dicho componente podemos asignarle unas

propiedades llamadas *props* que son como sus atributos. Por otra parte, también existe el concepto de estado, *state*, que es una representación momentánea del propio componente. Es precisamente el uso del *state* lo que permitirá tener datos dinámicos.

Para este proyecto se ha optado por usar la sintaxis JSX para describir la interfaz de la plataforma. Puede resultar vagamente familiar a HTML y, de hecho, la inmensa mayoría de las etiquetas y variables se llaman igual. Sin embargo, JSX permite juntar la lógica de renderizado con la de la interfaz. Es decir, incluir código funcional dentro de las etiquetas o incluso asignarlas a variables. De esta manera, se ha podido optar por generar el contenido de las lecciones con Adobe XD, importarlo a un fichero de datos desde el propio sistema de ficheros y luego incrustarlo en el lenguaje JSX. Esta opción permite crear un contenido mejor desde el punto de vista estético. Y, hay que recordar, que el Look & Feel de la plataforma es de vital importancia.

Además de esto, también ha sido necesario el uso de un lenguaje de diseño gráfico como lo es CSS para maquetar y dar estilo a la estructura generada mediante JSX.

Por último, ha sido preciso instalar las siguientes librerías adicionales:

- **Material UI:** Framework con numerosos recursos estéticos programados que ha servido para reducir código en las hojas de estilo en cascada CSS.
- **React Router** [43]: Por defecto, React no incluye ninguna herramienta para enturar los componentes por lo que la navegación entre distintas páginas no es posible sin la instalación de una librería.
- **I18next** [44]: Un framework de internacionalización hecho por y para JavaScript que permite detectar y cargar traducciones de cualquier idioma y de hacerlo de forma eficiente porque implementa un sistema de almacenamiento por caché.
- **Axios** [45] : Librería encargada de hacer promesas HTTP que nos permitirá establecer la comunicación con el Back Server.
- **Dotenv** [46]: Se usará esta librería para poder generar variables de entorno.
- **React Moment** [47]: Librería que permite transformar una fecha (para este caso la de publicación de un comentario) a un formato relativo. Es decir, indicar cuanto tiempo hace que se publicó en vez de decir cuando se hizo.
- **Simple React Validator** [48]: Librería que facilitará la validación del formato de los datos introducidos en un formulario.
- **Sweet Alert** [49]: Librería que permite generar ventanas emergentes estéticamente atractivas disparadas mediante eventos.
- **React Start Rating Component** [50] y **React Start Ratings** [51]: ambas librerías han sido usadas para incluir las valoraciones en formato de cinco estrellas que pueden visualizarse en la interfaz y que pueden usarse como campo de un formulario.
- **React Magnifier** [52]: Librería utilizada para implementar la funcionalidad de lupa para el cursor.

La API Restful alojada en el Back Server ha sido elaborada también mediante React y mediante Node.js [53], una plataforma de desarrollo que permite crear eventos de ejecución asíncronos para aplicaciones web. A su vez, también ha requerido la instalación de algunas dependencias:

- **Express** [54]: Framework que funciona sobre Node.js que nos permitirá crear las rutas que trabajarán sobre las peticiones HTTP. Es indispensable para hacer la API con Node.js.
- **Body-Parser** [55]: Dependencia que convertirá los datos de un formulario recibidos a JSON para que puedan ser tratados.
- **Mongoose** [56]: Librería que permite la interacción con la base de datos.
- **Validator** [57]: Librería que permite la validación del formato de los datos recibidos en el backend.

11.2.2 Estructura Interna

Siguiendo la división especificada en la arquitectura física, se analizará el código del sistema de los componentes que formen el Front Server separado de los que formen el Back Server aunque para ambos se haya usado el mismo framework.

11.2.2.1 Frontend [58]

Una de las características más interesantes de React es que permite incluir operaciones lógicas en el renderizado de componentes, en el JSX. Esto implica la unión de la lógica de la capa de presentación y del dominio en un mismo fichero. Esto simplifica mucho a nivel de código, pero puede hacer que los conceptos sean menos claros y la organización más compleja. Es por ello que es primordial el uso de técnicas que nos faciliten el reúso de componentes, así como la asimilación de los mismos. Y es en este punto cuando se opta por adoptar un patrón de diseño contenedor-componente.

También conocido como *Smart & Dumb*, este es un patrón de diseño basado en componentes originalmente exclusivo para React que, en la actualidad, ha ido abarcando numerosos frameworks. Fue popularizado en 2015 por Dan Abramov [59] y defiende la división de los elementos que forman el sistema en dos categorías: contenedores y componentes. La sintaxis puede confundir un poco al principio porque se usa el término componente tanto para hablar de los elementos en general de un sistema basado en React como de la subcategoría “componente” que tiene el patrón. Para evitar confusiones, se hará referencia a los mismos con las traducciones literales de sus nombres en inglés: “contenedores” y “presentativos”.

Los componentes presentativos son aquellos que solo se preocupan por cómo se muestran los datos. Les es indiferente su procedencia, no importan otros componentes y rara vez tienen estado propio por lo que suelen ser representados como funciones en vez de clases (aunque más adelante veremos la razón por la cual este código incluye varios componentes presentativos con estado). Su única función es mostrar la información de la que disponen.

En cambio, los componentes contendores son los encargados reales de gestionar el estado de la información, de transmitirla a los otros componentes para que hagan lo que deban hacer con ella.

Esta separación de conceptos, aparte de hacer más entendible las funciones y partes de cada elemento del código, permite reusar mejor los componentes y alterar la forma de presentar la información sin afectar a la lógica que la procesa.

Pueden encontrarse numerosos ejemplos reales de estos beneficios en el código de yayOS. El más representativo es el que puede verse en la Figura 44. Cuando el usuario indica que quiere acceder al contenido de una lección se pasa a ese componente contenedor la lección y este, a partir de la información recibida del sistema de ficheros, envía a los componentes presentativos una serie de propiedades para que ellos representen los datos. Se puede observar que, en este caso concreto, este componente no requiere de un estado para hacer esto pues, aunque es un componente contenedor porque maneja el envío de información, puede ser representado como una función.

```
import React from "react";
import {LessonSlider} from "../LessonSlider";
import {lessonsData} from "../assets/LessonsData";
import {LessonStructure} from "../LessonStructure";

export function LessonContent({lesson}) {

  return (
    <React.Fragment>
      <LessonSlider title={lessonsData.find(x => x.id === lesson).title}/>
      <LessonStructure numOfButtons={lessonsData.find(x => x.id === lesson).numeroBotones}
        content={lessonsData.find(x => x.id === lesson).contenido}
        lesson={lesson}
        isInteractive= {lessonsData.find(x => x.id === lesson).prueba}
        nInteractive= {lessonsData.find(x => x.id === lesson).nprueba} />
    </React.Fragment>
  );
}
```

Figura 44: Ejemplo componente contenedor. Fuente: Código de yayOS.

```
import React from "react";
import {useTranslation} from "react-i18next";

export function LessonSlider({title}) {
  const [t] = useTranslation("global")

  return (
    <div className="lessonSlider">
      <h1>
        {t(title)}
      </h1>
    </div>
  );
}
```

Y, cómo se puede ver en la Figura 45, un componente presentativo como lo sería la marquesina de la lección “LessonSlider” solo representa la información que recibe de su padre. En este caso, solo sería el título que ha de mostrar.

Figura 45: Ejemplo componente presentativo. Fuente: Código de yayOS.

Sin embargo, este patrón no impide que un componente presentativo no use la lógica para representar los datos. Puede tener estados si los necesita para mostrar la información correctamente. En el Anexo 3 de esta memoria puede verse el código del otro componente hijo: “LessonStructure”. Ahí puede observarse como este componente presentativo contiene atributos globales, estados e incluso funciones. Sin embargo, la función de estos elementos es únicamente presentar datos de la mejor manera posible por lo que es un componente presentativo.

De este modo, el Front Server está formado por una serie de componentes JavaScript contenedores y presentativos que usan el framework de React, pero no son los únicos elementos. Si se echa un vistazo a la Figura 47 se puede observar la distribución del sistema y en la estructura en la que se organiza.











 <code>assets</code>	Existe un componente padre de todo el sistema, App.js, del cual se irán formando los componentes, contenedores o presentativos, que hay en la carpeta “components”. Todos y cada uno usarán los estilos CSS de maquetación registrados en el fichero App.css y las variables globales y de entorno declaradas en Global.js y var.env. El enlace entre componentes queda definido en el fichero Router.js. Por último, el Front Server usa datos extraídos de un sistema de ficheros que se encuentra en la carpeta de “assets”, y unas traducciones que se encuentran en la carpeta de “translations”. El resto de ficheros son necesarios para el arranque del sistema.
 <code>components</code>	
 <code>translations</code>	
 <code>App.css</code>	
 <code>App.js</code>	
 <code>App.test.js</code>	En la carpeta de assets se encuentran la colección de diapositivas de todas las lecciones en formato png y otro tipo de recursos visuales como imágenes e iconos, y dos ficheros JSON con el resto de datos que requieren las lecciones: número de apartados, títulos, qué diapositiva corresponde a cada lección, la información que se muestra en el menú principal... Ya se vio un ejemplo de su estructura cuando se describió la capa de gestión de datos en el apartado de diseño. Por otro lado, en la carpeta de traducciones encontramos también tres ficheros JSON, cada uno con las traducciones en formato clave valor, de manera que cada fichero contiene la misma lista de claves, pero valores distintos según el idioma. Luego, en los componentes, en vez de introducir el texto allá donde se requiera se llama a una función de la librería i18next y se le envía como parámetro la clave correspondiente al texto que se pretende mostrar. El sistema ya se encargará de qué fichero debe coger el texto en base al lenguaje actual detectado.
 <code>Global.js</code>	
 <code>Router.js</code>	
 <code>index.css</code>	
 <code>index.js</code>	
 <code>reportWebVitals.js</code>	
 <code>setupTests.js</code>	
 <code>theme.js</code>	
 <code>var.env</code>	

Figura 47:
Estructura interna
del Front Server.
Fuente: código de yayOS

Otra característica que merece la pena destacar es que se ha optado por un sistema de enrutado de componentes en vez de vistas. Técnicamente hablando la vista que el usuario ve es siempre la misma o está generada a partir de la misma base. Lo que sucede cuando el usuario interactúa con la interfaz es un cambio en el conjunto de componentes que se renderizan, lo que produce la sensación de un cambio de vista.

Esta decisión fue tomada, en parte, para preservar el criterio de aceptación del requisito no funcional #22 “La interfaz del sistema mantendrá la mayor cantidad de elementos posibles en las mismas condiciones durante el cambio de página”. De esta forma se tiene la certeza absoluta de que solo los componentes que si o si han de ser cambiados se modifiquen. Además, hace que la transformación más dinámica y sutil, y evita que se recargue la página. De hecho, la forma que tienen los textos de renderizarse mediante el sistema de traducciones ayuda a este propósito puesto que el cambio de idioma no provoca un recargo de la página. La traducción se hace automáticamente. Vemos un ejemplo del enrutado en la Figura 48:

```
return (  
  <BrowserRouter>  
    <ScrollToTop>  
      <Header/>  
  
      <main>  
        <Switch>  
          <Route exact path="/" component={ListOfLessons}/>  
  
          <Route exact path="/leccion1" render={(props) => (  
            <LessonContent {...props} lesson={1}/>  
          )}/>  
          </>  
          <Route exact path="/valoracion1" render={(props) => (  
            <ListOfComments {...props} lesson={1}/> )}/>  
          </>  
  
          <Route exact path="/leccion2" render={(props) => (  
            <LessonContent {...props} lesson={2}/>  
          )}/>  
          </>  
          <Route exact path="/valoracion2" render={(props) => (  
            <ListOfComments {...props} lesson={2}/> )}/>  
          </>  
  
          <Route exact path="/nuevoComentario" component={CreateCommennt} />  
          <Route exact path="/form" component={Formulario}/>  
          <Route exact path="/instr" component={Instrucciones} />  
        </Switch>  
      </main>  
    </BrowserRouter>  
  )
```

Figura 48: Fragmentos del enrutado de los componentes. Fuente: Código de yayOS.

Como puede observarse, por cada ruta del sistema no se carga una página si no un componente que, en la mayoría de los casos, contendrá más componentes hijos. Es el caso de las rutas “/leccion1” y “/valoracion1” que corresponderían a la primera lección y a su sección de comentarios. Para estas rutas se renderiza un componente padre al que se le pasan las propiedades necesarias (son componentes contenedores). También vemos al final de la imagen otras rutas que renderizan otros componentes que no requieren de propiedades. Además, se puede contemplar como hay componentes que siempre están presentes sea cual sea la vista como lo es la cabecera o, aunque en este fragmento del código no se vea, el pie de página.

Para finalizar este análisis del frontend se va a mostrar un ejemplo de cómo hace el Front Server para comunicarse con el Back Server. Para ello, en la Figura 49 se puede advertir la comunicación que mantiene el componente contenedor encargado de dar al usuario la capacidad de enviar comentarios con la API alojada en el backend.

```
sendComment = (e) => {
  e.preventDefault();
  this.changeState();
  if(this.validator.allValid()){axios.post(Global.url + '/save', this.state.comment)
    .then(res => {
      if(res.status === 200) {this.setState({
        comment: res.data.comment,
        status: 'success'
      });
      sweetalert({
        title: this.props.t('create-comment-form.swal.1'),
        text: this.props.t('create-comment-form.swal.2'),
        icon: "success",
      });
    }
    else {
      this.setState({
        comment: res.data.comment,
        status: 'failed'
      });
    }
  });
}
else{
  this.setState({
    status: 'failed'
  });
  this.validator.showMessages();
  this.forceUpdate();
}
}
```

Figura 49: Ejemplo comunicación frontend y back. Fuente: Código de yayOS

Cuando el usuario haga clic en el botón de envío de la interfaz, un disparador percibirá dicha acción y se llamará a esta función. Tras comprobar que los datos introducidos en el formulario sean válidos, hará una petición post del comentario que ha sido almacenando en el estado propio del componente. Como que en este caso está configurado por defecto solo comprueba que el campo no esté vacío. Si la petición tiene éxito, la API regresa un código de aceptación y la interfaz renderiza una ventana emergente estéticamente atractiva indicando que el comentario se ha publicado con éxito (de la mano de la librería sweetalert). En caso contrario, el comentario no se enviaría. Si en lugar de eso la validación no es correcta, la interfaz indicaría los pasos a seguir para complementar el formulario. Por defecto la librería utilizada para validar los formularios se encarga de mostrar los mensajes. Se han modificado tanto su contenido como apariencia para adaptarlo a los requisitos de calidad de este proyecto.

Nombre:

Escribe tu nombre o uno inventado...

Este campo es obligatorio.

Opinión:

Escribe tu opinión sobre la lección: ¿Está bien redactada? ¿Hay algo que no quede claro? ¿Crees que falta información?

Este campo es obligatorio.

Haz clic izquierdo en las estrellas y califica la lección

★ ★ ★ ★ ★

Figura 50: Ejemplo de formulario incompleto. Fuente: <https://yayosplataformaweb.herokuapp.com/nuevoComentario>

11.2.2.2 Backend [60]

Falta por analizar la otra parte del sistema. El comportamiento interno de la API alojada en el Back Server. Para su implementación también se ha seguido un patrón de diseño para facilitar la reutilización del código, su adaptabilidad y para hacerlo más inteligible. Se ha apostado por el uso del patrón de diseño Modelo-Vista-Controlador [61] que divide la API en 3 capas de abstracción distintas, aunque no siguiendo el estándar al que estamos acostumbrados como se refleja en la Figura 51.

Tenemos los modelos con los que se dará formato a los datos enviado y recibidos de la base de datos, sí, y los controladores que definen esta interacción. Sin embargo, en vez de vistas lo

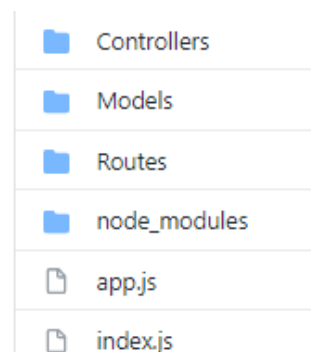


Figura 51: Estructura interna del Back Server. Fuente: código de yayOS

que tendremos serán unas rutas a las que asignaremos estos comportamientos. Estas rutas serán las que usarán el Front Server para mandar peticiones a la API como ya se ha podido ver en el ejemplo de añadir un comentario de la Figura 49. De hecho, en la Figuras 51,52 y 53 podemos ver la continuación de este ejemplo. Antes se ha visto el código que solicitaba a la API la inclusión del nuevo comentario y ahora su proceso de adición.

Lo primero en intervenir es el enrutado. Cuando se recibe una petición para añadir un comentario se ha de ceder el control a la operación “save” del controlador de comentarios.

```
'use strict'

const express = require('express');
const CommentController = require('../Controllers/Comments');

const router = express.Router();

//rutas de prueba
router.get('/test-de-controlador', CommentController.test);

//rutas útiles
router.post('/save', CommentController.save);
router.get('/comment/:lesson/:last?', CommentController.getComments);
router.get('/comment/:id', CommentController.getOneComment);
router.delete('/comment/:id', CommentController.delete);
router.get('/search/:search', CommentController.search);
router.get('/rates/:lesson', CommentController.getRates);
module.exports = router;
```

Figura 51: Router de los comentarios. Fuente: código de yayOS.

El controlador, una vez recibe la petición, recoge sus parámetros y los valida. Recordemos que solo comprueba si los campos están vacíos. En caso de que faltase algún parámetro se avisaría al frontend. Si no, se procede a crear un objeto comentario siguiendo el modelo establecido de la Figura 53. Se asignan los valores pertinentes y se hace la operación de añadir el comentario a la base de datos. Se notificará cualquier error que se produzca durante esta operación. Podemos ver cómo las opciones de respuesta de la API son las mismas que luego transmite el frontend al usuario.

Para no saturar a un usuario de avanzada edad con mensajes de error que lo alteren se ha optado por que el intento fallido de subir un comentario de indistintamente un mensaje 200 aunque no se haya cumplido el objetivo. De esta forma lo único que sucede en el frontend es que el comentario no se manda, pero los campos se mantienen igual y la vista no varía. Simplemente todo queda como si el botón de envío nunca hubiese sido pulsado.

```

save: (req, res) => {
  //Recoger parametros por post
  const params = req.body;
  //Validar datos
  try {
    var validate_name = !validator.isEmpty(params.name);
    var validate_content = !validator.isEmpty(params.content);
  } catch (err) {
    return res.status(500).send({
      status: 'error',
      message: 'Faltan datos'
    });
  }
  if (validate_name && validate_content) {
    //Crear el objeto a guardar
    var comment = new Comment();

    //Asignar valores
    comment.name = params.name;
    comment.content = params.content;
    comment.rate = params.rate;
    comment.lesson = params.lesson;

    //Guardar el articulo
    comment.save((err, commentStored) => {
      if (err || !commentStored) {
        return res.status(404).send({
          status: 'error',
          message: 'Comentario no guardado'
        });
      }
      //Devolver una respuesta
      return res.status(200).send({
        status: 'sucess',
        message: 'Comentario guardado'
      });
    })

  } else {
    return res.status(200).send({
      status: 'error',
      message: 'Datos no válidos'
    });
  }
},

```

Figura 52: Función save del controlador de comentarios. Fuente: Código de yayOS.

El modelo de los comentarios contiene en su mayoría campos que se rellenan con los parámetros de la petición, pero por ejemplo la fecha se establece a partir de la actual.

```
'use strict'

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const CommentSchema = Schema({
  name: String,
  content: String,
  date: {
    type: Date,
    default: Date.now
  },
  rate: Number,
  lesson: Number
});

module.exports = mongoose.model('Comment', CommentSchema);
```

Figura 53: Modelo de los comentarios. Fuente: código de yayOS.

En el fichero App.js se define este vínculo entre los tres niveles de abstracción y, en el fichero index.js, la conexión con la base de datos de mongoose para que las peticiones vayan directamente al gestor de la base de datos alojada en el servicio de Atlas del cual ya se ha hablado en el capítulo de diseño. En la Figura 54 podemos ver cómo se realiza esta conexión, aunque se ha censurado aquellas partes del código en las que se muestra información sensible.

```
'use strict'

require('dotenv').config({path: '../src/var.env'});

const mongoose = require('mongoose');
const app = require('./app')

mongoose.set('useFindAndModify', false);
mongoose.Promise = global.Promise;
try {
  mongoose.connect('mongodb+srv://<user>:<password>@<cluster>.mongodb.net?retryWrites=true&w=majority',
    {useNewUrlParser: true}, () => console.log(" Mongoose is connected"))
} catch (e) {
  console.log("could not connect");
}

const dbConnection = mongoose.connection;
dbConnection.on("error", (err) => console.log(`Connection error ${err}`));
dbConnection.once("open", () => console.log("Connected to DB!"));
```

Figura 54: Conexión a la base de datos. Fuente: Código de yayOS.

La base de datos solo trabaja por el momento con comentarios y sugerencias por lo que su funcionamiento dentro de las tres capas de la API es independiente, es decir, está distribuido en ficheros distintos.

11.2.3 Perfeccionamiento por iteraciones de refactorización

Se ha decidido que la mejor forma para asegurar el cumplimiento de los criterios de aceptación de los requisitos funcionales y no funcionales es haciendo un proceso de refactorización dividido en fases iterativas. Repetir, cuantas más veces mejor, el proceso de afinamiento. Para ello, habrá una primera fase de desarrollo general de todas las funcionalidades y características que la plataforma web aspira a tener y, a posteriori, se irán haciendo refactorizaciones y mejoras a partir de las opiniones y sugerencias de los probadores de software. La primera iteración será, por tanto, la que llevará más tiempo puesto que engloba la inmensa mayoría de las tareas relacionadas con la parte de implementación. El resto de iteraciones formarán parte de la tarea de perfeccionamiento (T3.12).

Iteración 1

Durante el desarrollo de la primera iteración se decidió alterar un poco el alcance de la misma y se hizo un proceso de refactorización muy concreto y selectivo cuando se tenía implementado el menú principal, las dos primeras lecciones y el sistema de traducción. Se tomó esta decisión para valorar la eficacia de los cimientos y del enfoque de la aplicación tanto a nivel funcional como de calidad.

Se pidió a los probadores que valoraran el producto para comprobar si su uso era intuitivo o si se tenía que hacer algún cambio en el enfoque de la aplicación. Esta prueba se hizo en local y con la presencialidad del autor de este proyecto debido a que aún no se había hecho el despliegue de la aplicación web. En general, y sin entrar en detalles, se detectaron los primeros bugs y puntos débiles. El acceso a las funcionalidades parecía ser bastante cómodo, pero se modificaron y profundizaron algunas explicaciones de las dos lecciones. Una vez se realizaron estas correcciones, se terminó el resto de tareas correspondientes y se dio a cada tester el enlace de la aplicación desplegada en Heroku para que, esta vez sí, pudieran poner a prueba el sistema desde un entorno ajeno al del desarrollador.

Iteración 2

La segunda iteración arranca con la resolución de los problemas y con la aceptación de las sugerencias proporcionadas por los testers al final de la iteración anterior. Aparte de las correcciones propias que hizo el autor de este proyecto, en la Figura 55 pueden verse las soluciones planteadas a cada una de ellas. Cabe recalcar que la mayoría de opiniones se repetían en algunos testers, pero, para simplificar, solo se mencionará en uno de ellos. Puede observarse también que, además de las cinco personas de avanzada edad que se prestaron voluntarias a probar la aplicación, se contó con la ayuda de otros tres testers con conocimientos básicos en el desarrollo de software que quisieron probar la plataforma y dar opiniones más técnicas sobre su uso.

Tester	Aportación	Corrección
Tester 1	No se le ocurrió que las flechas situadas debajo de las diapositivas de las lecciones eran para avanzar o retroceder en la lección.	Se cambió el diseño estético de las flechas y en el apartado de instrucciones se especificó su funcionalidad.
	Consideró que la explicación de cómo crearse una cuenta en Google era poco clara.	Se reescribió el apartado y se le dedicaron más diapositivas e imágenes.
Tester 2	No entendió que la lección estaba dividida en apartados y que los botones laterales eran un menú interactivo para cambiar de diapositiva.	Se resaltó en la lista de apartados lateral el apartado correspondiente a la diapositiva visualizada en ese momento avanzándolo respecto al resto 25 píxeles y poniéndole un borde luminoso que resaltara con el fondo.
Tester 3	Creyó que las imágenes de las lecciones eran interactivas	Se recalcó en las diapositivas que toda imagen de un formulario era solamente una imagen.
	No se daban por entendidos algunos términos y explicaciones en la lección “Gestionando los ficheros de mi ordenador”.	Se añadieron más diapositivas y se ampliaron las explicaciones.
	Experimentó dificultad para encontrar las teclas en el teclado en la lección “Teclado y ratón”	Se añadieron indicaciones que facilitarían el uso de las teclas y se incluyó la prueba interactiva para practicar con el teclado.
Tester 4	Se advirtió que al tester le costaba cada vez recordar que tenía que usar otra pestaña para hacer lo que le indicaba la plataforma	Se hizo más hincapié en ese aspecto en todas las lecciones en general
Tester 5	No supo por dónde empezar.	Se enfatizó el uso de las instrucciones y se ha recalcado la posibilidad de desplazarse verticalmente con la barra lateral.
	No supo que eran las “cookies” puesto que en la lección de “Cómo buscar por Internet” nunca se explicaron.	Se añadió una explicación sencilla de qué son las “cookies”, por qué nos las piden y qué sucede al aceptarlas.
Tester especial 1	Las vistas no se ven bien en pantallas de alta resolución.	Se cambió el tamaño definido en el css por uno que fuera en función de las dimensiones de la pantalla. Se comprobó con un simulador online [62]
	El color del mensaje de advertencia en los formularios apenas se ve con el fondo negro de la interfaz.	Se ha agrandado el mensaje, añadido un color rojizo que llame más la atención y contraste más con el fondo y se ha subrayado para enfatizarlo más.
Tester especial 2	Se sugirió otra tonalidad de fondo oscura pero más clara con la que el aspecto era más alegre y se mantenía el contraste.	Se cambio el color de fondo #292727 por el #353333.
	Cree oportuno incluir una funcionalidad que consista en que se puede regresar al menú principal en cualquier momento haciendo clic en el logo de yayOS.	Se ha añadido el enrutado al menú principal en un método onClick en la imagen del logo de yayOS.
Tester especial 3	Cuando Heroku tarda en cargar el Back Server y la conexión con la base de datos, el usuario no sabe si la conexión ha fallado o es que debe de esperar.	Se añadieron mensajes que diferenciaban ambos casos. La primera conexión con Heroku siempre es lenta, pero eso ya depende de la capacidad de la plataforma de despliegue.

Figura 55: Tabla con la resolución de los comentarios de los testers. Fuente: Elaboración propia.

Iteración 3

En esta iteración se dio a probar una nueva versión refactorizada de la aplicación. La sensación general entre los testers fue de mejoría respecto de la otra. Funcionalidades más bien definidas, capacidades más claras y elementos más visibles. Sin embargo, surgieron algunos problemas a raíz de estas modificaciones.

El más significativo fue un bug que surgió tras cambiar el tamaño fijo de los elementos de las lecciones a tamaños que dependieran de la resolución de las pantallas. Ya se ha comentado en el apartado de diseño que las vistas contienen un pie de página que no posee ninguna funcionalidad realmente más allá que la de imitar el tipo de página web que el usuario encuentra por Internet y adecuar así el entorno lo máximo posible a lo que sería una experiencia real de navegación. La posición del pie de página venía dada por la altura de todos los elementos anteriores. Al no ser esta altura fija, tampoco lo era su posición por lo que si la vista contenía pocos elementos, como sería el caso de una vista sin comentarios, el pie de página era mostrado en medio de la pantalla. Se solucionó colocando todos los elementos en un contenedor que tuviera una altura mínima igual a la de la pantalla.

Se modificó también el modo que tenía el sistema de restringir el cambio entre las diapositivas de una lección. De la manera que estaba implementado en la iteración anterior, si el usuario se encontraba en la primera diapositiva solo le aparecía la posibilidad de ir a la siguiente ya que no tiene sentido ir hacia atrás porque no existe una diapositiva anterior. Y lo mismo sucedía con la última. En otras palabras, las flechas en cada sentido solo eran visibles si existían más diapositivas en ese sentido. Para evitar la continua aparición y desaparición de elementos (algo que, en realidad, hubiese contradicho el criterio de aceptación del requisito no funcional #26), se hizo que ambas flechas estuvieran siempre presentes pero que cuando uno de los movimientos no fuera posible su respectiva flecha tuviera un color grisáceo. Obviamente no se produce ningún efecto si se interactúa con ella.

Se modificaron algunas explicaciones, tanto de las lecciones como de la sección de instrucciones, y se añadió una prueba interactiva en la lección de “Gestionando los ficheros de mi ordenador” que permitía a un usuario hacer pruebas con la ventana emergente y subir a la web una foto para visualizarla. Así podía practicar todo lo relacionado con selección de ficheros, extensiones, etc.

Por último, se añadió una funcionalidad más junto con estas flechas y fue la posibilidad de convertir el ratón en una lupa que aumentara el contenido de las lecciones. Esto se hizo a raíz de algunas observaciones sobre que el texto de ciertas imágenes de ejemplo dentro de las diapositivas era demasiado pequeño. Aunque este texto no era relevante en ninguno de los casos, se decidió crear la lupa de todas formas.

Iteración 4

La adición del apartado interactivo del gestor de ficheros implicó una refactorización del código y de la manera que tenía la lógica de la capa de presentación de saber cuándo renderizar una diapositiva y cuándo no y esto entró en conflicto con la lupa puesto que esta funcionalidad solo sirve para hacer zoom en una imagen, no en un componente reactivo. Hubo que arreglar esos fallos.

En esta iteración se terminaron de traducir todas las diapositivas y se terminó de ajustar pequeños detalles en las mismas y de la aplicación web en general relacionados, principalmente, con su adaptabilidad.

11.3 Despliegue del sistema

A finales de la primera iteración llegó, por fin, el tan ansiado momento: desplegar el sistema y que estuviera disponible para el público. Sin embargo, el cómo hacerlo fue una cuestión un tanto más delicada. Era importante saber elegir la mejor manera para que se adecuara a nuestras necesidades.

Actualmente existe una gran diversidad de plataformas, tanto de pago como gratuitas, que permiten este proceso. Al no disponer de servidor propio, se ha optado por buscar soluciones económicas y que, a ser posible, no implicaran demasiado cambio en los ficheros propios de la implementación puesto que trabajábamos con una cantidad de datos importados considerable. Una de las primeras opciones en barajarse fue usar Google Cloud Service [63] o el servicio Amplify de Amazon Web Services [64]. Ambas opciones son gratuitas, aunque una implica algunos cambios en el código fuente para adaptar el despliegue y en la otra, en cambio, puede hacerse simplemente sincronizándose con un repositorio. Incluso acepta Bitbucket, la herramienta de control de versiones usada en este proyecto. Sin embargo, ambos incluían periodos de prueba gratuitos muy limitados y no parecía muy claro dónde estaban los límites a partir de los cuales se empezaba a cobrar. Fue por eso que, aún y a sabiendas de sus limitaciones, se recurrió finalmente al servicio de computación en nube Heroku [65], con el que ya se tenía experiencia previa pues se había trabajado en la asignatura de Aplicaciones de Servicios Web (ASW).

Heroku permite la vinculación directa con un repositorio de GitHub. Solo se tuvo que clonar el repositorio de una herramienta a otra, crear una aplicación en Heroku y vincularla. Incluso se podía activar el despliegue automático y que se creara una nueva versión de la plataforma cada vez que se actualizara la rama de GitHub, pero nunca se llegó a activar esa opción porque no había control de errores. Se fue redesplegando manualmente tras cada actualización.

En total fueron lanzadas dos aplicaciones. Claramente, el Front Server y Back Server. La comunicación entre ambas viene dada por las llamadas de una a las rutas de la otra que hemos podido ver anteriormente. En un fichero Global se creó una variable url que contuviera la ruta base de todas las peticiones. La ruta base es, eventualmente, el enlace del despliegue del Back Server en Heroku. También puede verse comentada la ruta de testeo local usada durante las pruebas en el desarrollo.

```
var Global = {  
  //url: "http://localhost:3900"  
  url: "https://yayos-backend.herokuapp.com"  
}  
export default Global;
```

Figura 56: Conexión entre Front Server y Back Server. Fuente: código de yayOS.

Y luego esta variable es usada en las peticiones HTTP como ya hemos visto en la Figura 49.

11.4 Solución de problemas

Aparte de los numerosos bugs o momentos de incertidumbre que se tuvieron durante la implementación de la aplicación web, surgieron algunos problemas que obligaron a cambiar el enfoque o el formato de ciertos ficheros. En este apartado veremos algunos de esos sucesos imprevistos, cómo se lidió con ellos y las razones tras las cuales se solucionó de la manera que se hizo. No hay que confundir el contenido de esta sección con aquellos cambios que se produjeron en las distintas iteraciones durante la fase de desarrollo. Aquí se hablará de obstáculos o inconvenientes que se encontraron durante la primera iteración y que definieron la forma de implementar la plataforma.

11.4.1 Creación de un componente ScrollToTop

Como se ha explicado anteriormente, se recurrió a un tipo de enrutamiento que permitía elegir qué componentes concretos había que renderizar según la ruta a la cual el usuario quería acceder.

Pero esto produjo un problema inesperado y es que cuando un usuario descendía en el menú principal y accedía a una lección los nuevos componentes se renderizaban, pero, como que la vista es realmente igual, el usuario permanecía a la misma altura que había descendido. La sensación era la de ir a otra vista y verla directamente a media altura o desde el pie de página en vez de empezar por la parte superior. ¿Solución? Obligar al sistema a que hiciera, tras cada enrutamiento, un desplazamiento vertical hasta el pixel (0,0) de la página. Y de esto es de lo que se encarga el componente “ScrollToTop” mediante el método *componentDidUpdate*. React ofrece este método para hacer comprobaciones cuando los componentes de una vista se actualizan. Si la localización ha variado respecto a la última actualización, se trasladará la vista a su parte superior.

```
class ScrollToTop extends Component {
  componentDidUpdate(prevProps) {
    if (this.props.location !== prevProps.location) {
      window.scrollTo(0, 0)
    }
  }

  render() {
    return this.props.children
  }
}
```

Figura 57: Componente ScrollToTop. Fuente: código de yayOS.

11.4.2 La traducción de datos importados

Cuando se terminó la tarea de implementación de segunda lección, se decidió que lo mejor era tratar de integrar el framework de traducción. Se pensó que esta integración podía conllevar cambios en la forma que tenían los componentes de recibir los datos y se valoró que era mejor hacerlo cuanto antes para que las alteraciones fueran mínimas. Fue una decisión muy acertada.

Como ya se ha explicado en la parte de implementación, se ha generado una carpeta de traducciones en las que se ha incluido un fichero JSON para cada idioma. A base de etiquetas se pueden asignar las cadenas de caracteres a traducir y, según en el idioma que se esté, se consulta esa cadena en un fichero o en el otro.

Este sistema funciona sin problema tanto para componentes presentativos, que son eventualmente funciones flecha, como para componentes contenedor. Pero, para este último caso, solo lo hacía si los datos estaban incrustados dentro del JSX. En otras palabras. Si existe un componente que debe mostrar texto por pantalla y se le indica explícitamente a ese componente el texto a mostrar, no hay problema. Solo se debe sustituir ese texto por una llamada a la función encargada de la traducción y pasarle como parámetro la clave correspondiente.

Pero este mismo método no funciona cuando los datos proceden de una fuente de datos importada del sistema de ficheros del propio Front Server porque el texto real no está en ese fichero si no que estamos pasando un “enlace” a otro lugar donde se encuentra la información. Y esto es algo que se hace tanto para mostrar la información del menú principal como para la del contenido individual de cada lección, puesto que se reutiliza siempre el mismo componente presentativo. Esto era un grave problema pues impedía totalmente la traducción. ¿Cómo se solucionó? Se optó por convertir todos estos componentes de clase que usaban datos importados por funciones exportadas llamadas por sus componentes padres. De esta manera, en vez de pasarle la propiedad se le pasaba la clave de la traducción. Luego, hubo que sustituir la información que se encontraba en los ficheros acerca de las lecciones por las claves de las traducciones y, en los ficheros de las traducciones, escribir el contenido para cada idioma disponible en la plataforma. En definitiva, una cantidad de horas considerables invertidas en cambiar el formato y la localización de los datos y, a posteriori, traducirlos.

Y esta es la razón por la que, aunque no sea habitual, en este proyecto hay tantos componentes presentativos con estado propio: por las traducciones.

11.4.3 La incompatibilidad entre estilos propios y ajenos

La capacidad del sistema de convertir el ratón en una lupa y que el usuario pueda ver ampliada la porción de la diapositiva que desee fue una funcionalidad muy acertada, pero trajo consigo algún que otro inconveniente que hubo que resolver.

Las diapositivas que se muestran en las lecciones están sometidas a una serie de restricciones de estilo en el CSS que las hacen más atractivas estéticamente y cómodas de visualizar. Sin embargo, este estilo entra en conflicto con el que otorga por defecto la librería de la lupa magnificadora.

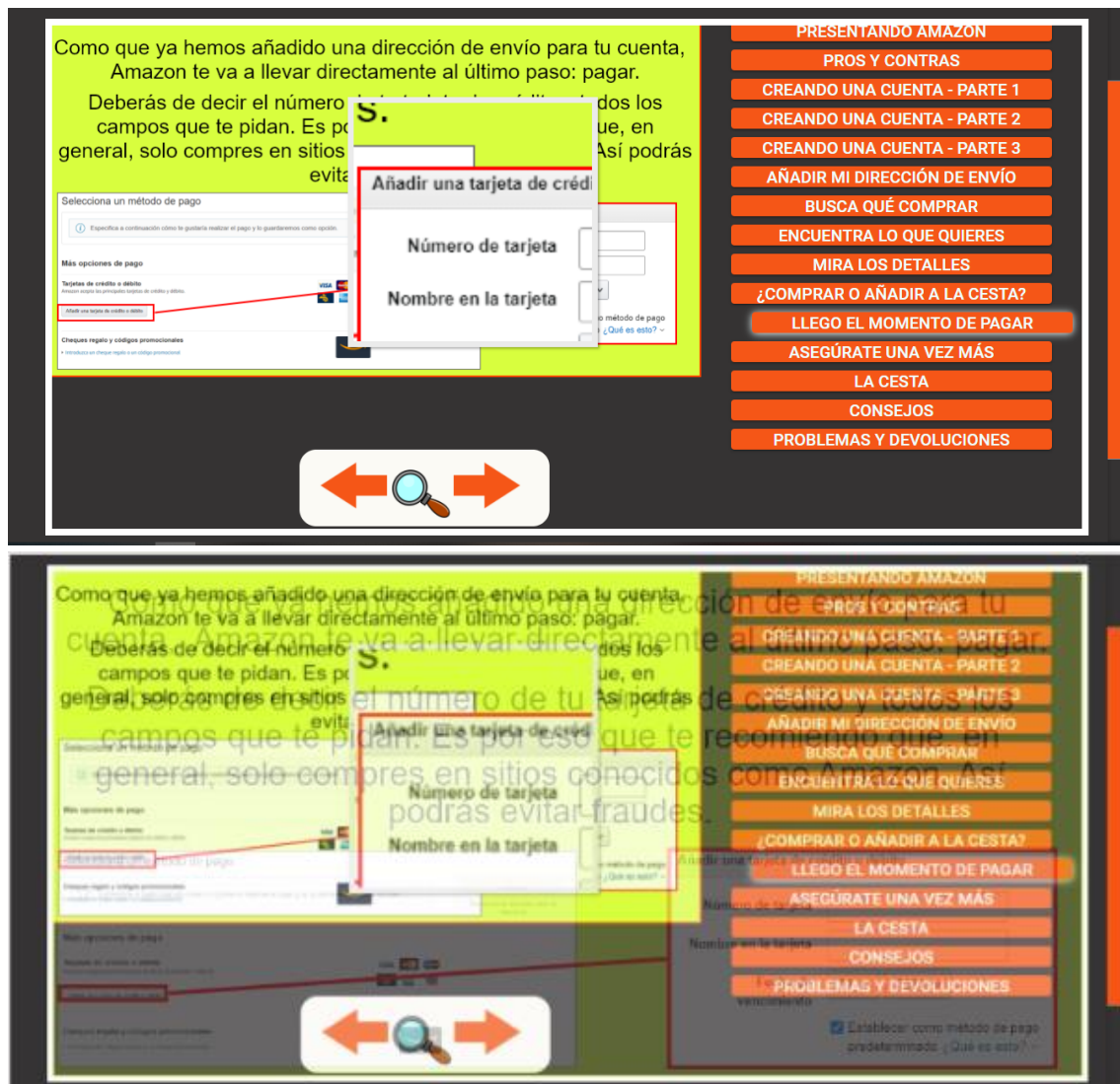


Figura 58: Recreación del comportamiento real de una imagen ampliada.

Fuente: Elaboración propia.

En la Figura 58 puede observarse el efecto real que tiene sobre la imagen cuando la lupa aplica el aumento. Lo que sucede realmente es que la imagen incrementa su tamaño, aunque esta no es visible para el usuario a excepción de la zona que ocupe la lupa. Aunque la imagen no se vea es detectada por el resto de elementos de la interfaz, de manera que los obligaba a desplazarse cuando la lupa se activaba para no colisionar los unos con los otros, por lo que no debería haber ningún otro elemento en la zona que ahora ocupa la imagen de baja opacidad de la figura. Y esto no es correcto.

Para solucionarlo se asignaron dos tipos de estilos distintos a la imagen: el normal, y otro que contrarrestara la expansión. En base a si el zoom está o no activado se carga una imagen u otra. Por defecto, cuando el sistema carga el contenido de una lección ya

cargaba por cada apartado tres diapositivas, una por cada idioma. Con este cambio se duplican el número de diapositivas cargadas. Tres para cada estilo, seis en total.

11.4.4 Lentitud de Heroku al despertar la aplicación

Heroku nos ofrece una alternativa gratuita y eficaz para desplegar el sistema, pero tiene sus limitaciones como ya se ha mencionado en el apartado de despliegue. La más significativa es el límite de horas mensuales que la aplicación puede estar funcionando.



Figura 59: Ratio de horas usadas. Fuente:
<https://dashboard.heroku.com/apps/yayosplataformaweb/activity>

Esta plataforma de distribución utiliza un contenedor Linux ligero y aislado para correr las aplicaciones que, para planes de uso gratuito, ofrece un total de 550 horas mensuales de uso. Para prevenir el exceso de horas, la aplicación pasa a modo inactivo al cabo de 30 minutos de no ser usada. Es por esa razón que puede advertirse una cierta lentitud cuando se accede al sistema puesto que tiene que activarse primero. Y, por supuesto, esta lentitud no se produce solamente cuando accedemos a la plataforma yayOS si no también la primera vez que se haga una petición a la base de datos porque, recordemos, el backend está desplegado en otra aplicación distinta también mediante Heroku.

Al ser esta plataforma un mero prototipo y al no disponer de liquidez para costear un plan de pago, no se ha planteado cambiar por el momento de distribuidora. Ciertamente es una limitación ajena a las capacidades del desarrollador. La aplicación funciona y, una vez se carga la primera vista o la primera petición a la API, todas las funcionalidades avanzan a la rapidez que deberían porque el sistema ya se ha despertado.

Se ha investigado si existen formas de avisar periódicamente a las aplicaciones para que se mantengan despiertas y, aunque existen, esto no sería una solución porque la web estaría constantemente activa y se superaría el límite de 550 horas. Como que esto es en el presente un proyecto académico se ha decidido obviar esta limitación y adaptarse a las circunstancias. Por ello, la interfaz avisará al usuario cuando se detecten estos periodos de lentitud y le indicará que espere. Del mismo modo, se bloquearán los formularios no mostrando el botón de envío hasta que no se detecte que la API está operativa para evitar fallos durante la subida de comentarios o sugerencias.

Para sobrepasar el límite, el sistema tendría que estar funcionando como mínimo 18 horas diarias, algo altamente improbable dadas las circunstancias actuales. Este sería,

por tanto, un problema a solucionar en el futuro cambiando el servicio proveedor por otro de más calidad y prestaciones o recurriendo a un plan de pago de este sistema de distribución.

12. Validación del sistema

El siguiente paso tras finalizar el proceso de implementación es comprobar que el sistema creado cumple los propósitos y especificaciones detallados en los requisitos definidos en el octavo capítulo de esta memoria. Recordemos que los requisitos aseguran la utilidad del sistema y, por tanto, el cumplimiento de los objetivos que pretende alcanzar. Así que es de vital importancia asegurarse de que la solución es práctica y eficaz para resolver el problema que lo empezó todo.



Para hacer esta validación se repasará uno por uno los requisitos funcionales y de calidad y se observará si se satisface su criterio de aceptación y, en caso de no ser así, si el motivo es suficientemente justificado (si las circunstancias son especiales, si acatar a rajatabla el requisito en una situación concreta contradice otro requisito de mayor importancia, si se ha visto a posteriori que realmente el requisito estaba mal definido o ha cambiado durante el transcurso del desarrollo del proyecto...).



También se hablará con los testers que han estado probando la aplicación web para que den constancia de su utilidad y del efecto que ha tenido en sus vidas.


12.1 Cumplimiento de los requisitos


12.1.1 Validación de requisitos funcionales



Se ha llevado a cabo una comprobación manual de todas las funcionalidades del sistema tanto por parte del desarrollador como del grupo de testers a nivel de aplicación web, es decir, en el sistema ya desplegado. Como que los requisitos funcionales han sido especificados mediante una serie de casos de uso en el capítulo 9 de esta memoria se considera que la mejor forma de validarlos es mediante el correcto cumplimiento de todos los casos de uso. Lógicamente, la validación por parte de los testers voluntarios no se tendrá en cuenta en aquellos casos de uso cuyo actor sea el administrador y, por tanto, no se especificará en estos casos cuantos testers han logrado llevar a cabo ese caso de uso en particular si no que se observará, exclusivamente, el éxito o fracaso del administrador que, a efectos prácticos, es el desarrollador del proyecto.



Filtrar lecciones			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El usuario introduce la dificultad y el sistema muestra únicamente las lecciones con esa dificultad.		8/8	Filtra la lista de lecciones correctamente.
El usuario introduce la dificultad y el sistema no muestra ninguna lección porque no existen con esa dificultad.		8/8	Para la prueba se han eliminado las lecciones difíciles del sistema. Si se buscan no muestra resultados.



Acceder a una lección			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El usuario accede a una lección y el sistema la muestra.		8/8	Muestra todas las lecciones correctamente.
El usuario accede al menú secundario de una lección y de ahí accede a una parte concreta.		8/8	Solo la lección sobre el manejo de Skype incluye submenús. Funciona correctamente.



Cambiar el idioma de la plataforma			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El usuario puede cambiar el idioma de la interfaz.		8/8	La plataforma puede verse en castellano, catalán e inglés. La traducción se mantiene en futuras navegaciones.



Acceder a las instrucciones			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El usuario accede a la vista de instrucciones y el sistema la muestra.		8/8	Las instrucciones se muestran correctamente.


Añadir una lección			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador introduce todos los elementos requeridos en el sistema de ficheros y crea una nueva lección.			La nueva lección aparece, es accesible, y se visualiza correctamente.





Modificar lección			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador cambia los datos pertinentes de la lección (incluso traducciones) y modifica la lección.			Tras realizar los cambios, al acceder a la lección se ve la nueva versión.





Eliminar lección			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador elimina todo elemento relacionado con la lección.			La lección no vuelve a verse en el sistema. Se ha comprobado al validar la extensión del escenario de éxito del caso de uso del filtrado de lecciones.



Añadir comentario			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El usuario rellena el formulario y publica un comentario de una lección. Luego es redirigido a la vista de los comentarios de la lección.		8/8	El comentario aparece, queda registrado en la base de datos y su puntuación afecta a la media global de la lección.
El usuario deja el formulario incompleto y el sistema le advierte de los campos que falta por rellenar.		8/8	Aparecen mensajes debajo de los campos vacíos indicando que su completado es obligatorio.


Ver últimos comentarios			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El usuario accede al apartado de comentarios de una lección y estos son mostrados junto con su calificación media.		8/8	Puede verse los últimos 15 comentarios y la calificación global.
El usuario accede al apartado de comentarios de una lección, pero la interfaz le indica que no hay comentarios para mostrar.		8/8	Aparecen un mensaje que advierte de que aún no hay comentarios registrados.



Ver todos los comentarios			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El usuario indica que quiere ver todos los comentarios y el sistema los muestra.		8/8	Pueden verse todos los comentarios. La calificación media no varía. Si no hay se dice.



Buscar comentario por id			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador indica un identificador y el sistema le devuelve el comentario con ese identificador.			Se visualizan todos los datos del comentario con identificador igual al indicado.
El administrador indica un identificador no válido y el sistema no encuentra coincidencias.			Se devuelve un error interno a nivel de API.



Buscar comentario por palabra			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador indica una cadena de caracteres y el sistema le devuelve los comentarios que la contengan.			Se visualizan todos los datos de todos los comentarios que contengan la cadena en el nombre o el cuerpo del comentario.
El sistema no encuentra comentarios que contengan la cadena de caracteres especificada.			Se devuelve un error interno a nivel de API.



Eliminar comentario			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador indica un identificador y el sistema elimina el comentario de la base de datos.			El comentario es eliminado de la base de datos. Si se busca posteriormente, la API devolverá error.





El administrador indica un identificador no válido y el sistema no encuentra coincidencias.			Se devuelve un error interno a nivel de API porque no existe el comentario que se quiere borrar.
---	---	---	--





Enviar una sugerencia			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El usuario rellena el formulario y envía una sugerencia a la base de datos. Luego es redirigido al menú principal.		8/8	La sugerencia aparece en la base de datos y el usuario es redirigido.
El usuario deja el formulario incompleto y el sistema le advierte de los campos que falta por rellenar.		8/8	Aparecen mensajes debajo de los campos vacíos indicando que su completado es obligatorio.

Ver todas las sugerencias			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador indica que quiere visualizar todas las sugerencias.			Pueden verse todas las sugerencias y sus campos a nivel de API.

Buscar sugerencia por id			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador indica un identificador y el sistema le devuelve la sugerencia con ese identificador.			Se visualizan todos los datos de la sugerencia con identificador igual al indicado.


El administrador indica un identificador no válido y el sistema no encuentra coincidencias.			Se devuelve un error interno a nivel de API.
---	---	---	--


Buscar sugerencia por palabra			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador indica una cadena de caracteres y el sistema le devuelve las sugerencias que la contengan.			Se visualizan todos los datos de todas las sugerencias que contengan la cadena en el nombre, el cuerpo o el email de la sugerencia.
El sistema no encuentra sugerencias que contengan la cadena de caracteres especificada.			Se devuelve un error interno a nivel de API.


Eliminar sugerencia			
¿Cuál es el escenario a validar?	¿El desarrollador ha finalizado con éxito el escenario del caso de uso?	¿Cuántos testers han finalizado con éxito el escenario del caso de uso?	¿Observaciones a declarar?
El administrador indica un identificador y el sistema elimina la sugerencia de la base de datos.			La sugerencia es eliminada de la base de datos. Si se busca posteriormente, la API devolverá error.
El administrador indica un identificador no válido y el sistema no encuentra coincidencias.			Se devuelve un error interno a nivel de API porque no existe la sugerencia que se quiere borrar.


12.1.2 Validación de requisitos no funcionales


Para la comprobación del cumplimiento de los requisitos no funcionales se analizará que el diseño del sistema cumpla en todo momento cada uno de los criterios de satisfacción impuestos durante la definición de cada requisito de calidad. Se puede observar que:


Requisito no funcional: La interfaz ha de evitar el uso de colores pastel.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
La interfaz no debe usar colores pastel.	<p>Por lo general, se han evitado las tonalidades suaves o colores con grandes dosis de blanco en todos los elementos de la interfaz.</p> <p>Pueden encontrarse tonalidades de colores más pálidas en el fondo de las diapositivas, pero su uso queda plenamente justificado puesto que se pretende resaltar el contenido que hay en ellas (principalmente letras oscuras).</p>	


Requisito no funcional: La interfaz ha de evitar el uso de colores azules en zonas importantes.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
La interfaz no debe usar tonalidades azules en zonas importantes o de acceso generalizado.	<p>No se encuentran elementos de color azul en la interfaz salvo en zonas irrelevantes como en las imágenes usadas para presentar las lecciones o en el fondo de algunas diapositivas.</p> <p>Todo conjunto de elementos que desempeña una función principal ha tenido otro color.</p>	


Requisito no funcional: La interfaz ha de evitar el uso del color rojo y verde.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
La interfaz no debe usar tonalidades rojizas o verdosas en general.	<p>La interfaz no ha recurrido a estos colores a excepción de aquellos casos en los que se ha requerido transmitir alguno de los mensajes que se les atribuye normalmente o para el fondo de ciertas diapositivas.</p> <p>ROJO – Peligro, difícil, advertencia...</p> <p>VERDE – Disponible, libre, seguro, fácil...</p>	


Requisito no funcional: La interfaz del sistema usará una sola fuente de texto.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
La interfaz usará una sola fuente de texto salvo en aquellas situaciones en las que convenga hacer una distinción	La interfaz conserva el estilo Roboto en todos sus componentes con texto. Solo se han hecho distinciones (negrita, cursiva, tamaño, color...) para distinguir títulos de subtítulos y del cuerpo de texto normal.	


Requisito no funcional: La interfaz del sistema usará una fuente de texto grande y simple.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
La interfaz usará una fuente de texto mayor o igual a 15 píxeles. Usará: Roboto, Helvetica, Arial, Futura, Avant Garde o Verdana.	Los textos Roboto que se visualizan en la interfaz comprenden los siguientes tamaños: <ul style="list-style-type: none"> • Títulos a 64px, subtítulos 24px, botones 16px. • Párrafos del menú principal a 16px. • Texto de las lecciones oscila entre 15px y 30px dependiendo de la densidad de texto. • Textos de advertencia en formularios 30px. • Texto en botones 20px y en selectores 16px. • Texto en las pruebas interactivas de las lecciones oscila entre 20 y 30px. 	


Requisito no funcional: La interfaz ha de usar colores que contrasten.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
La interfaz usará colores claros para el fondo y oscuros para los componentes o viceversa.	La temática de la plataforma juega con un color anaranjado claro y un grisáceo más oscuro. El texto suele ser negro y el fondo blanco o un color suave. A veces el texto puede ser del mismo tono naranja o un rojo muy intenso para los mensajes de advertencia en los campos de los formularios. En estos casos el fondo es gris suave.	


Requisito no funcional: La interfaz del sistema optará por un diseño minimalista.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
Restringir la cantidad de elementos mostrados a la vez. Ni pop-ups ni transiciones animadas. Acceso directo.	Cada vista muestra lo esencial de la misma. Se ha priorizado la poca densidad de elementos siempre que ha sido posible. No hay transiciones entre pantallas y solo se han incluido ventanas emergentes en situaciones imprescindibles como es el caso de los mensajes de confirmación en una interacción con la base de datos.	


Requisito no funcional: La interfaz tratará de mostrar su contenido sin hacer scrolling.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
<p>La interfaz evitará que el usuario deba desplazarse verticalmente.</p> <p>Se explicitará y resaltará la posibilidad si su uso fuese inevitable.</p>	<p>Se ha resaltado el diseño de la barra vertical de manera que sigue la temática estética de la plataforma.</p> <p>Aunque depende de la resolución de la pantalla, en principio se ha evitado el uso del desplazamiento vertical en todos los casos salvo en el menú principal, las instrucciones, y en la vista de visualización de comentarios. La partición de este contenido en ventanas sería contraproducente.</p> <p>Se ha asegurado que una de las primeras cosas en verse en la plataforma sea la recomendación al usuario de que acceda al apartado de instrucciones. Ahí una de las primeras cosas en explicarse es el funcionamiento del desplazamiento vertical.</p>	


Requisito no funcional: La interfaz del sistema mantendrá la mayor cantidad de elementos posibles en las mismas condiciones durante el cambio de página.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
<p>La interfaz mantendrá todo componente visible de la interfaz durante el cambio de pantalla con las mismas características siempre y cuando sea posible.</p>	<p>La plataforma muestra su contenido siguiendo una misma estructura en aquellos elementos relacionados:</p> <ul style="list-style-type: none"> • Independientemente del avance por una lección, siempre se ve la diapositiva actual, el menú lateral y el panel de navegación. • Los comentarios siguen un mismo formato. • Los dos formularios son estéticamente iguales salvo por los campos que contienen. • La temática se mantiene en todo momento. • La traducción o cambios de vista no implica un recargo de la página. 	


Requisito no funcional: La interfaz del sistema indicará las instrucciones básicas para operar la plataforma.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
<p>La interfaz incluirá un apartado explicativo donde enseñará como debe moverse por yayOS, como ha de utilizarlo y los fines que tiene.</p>	<p>La aplicación incluye un apartado de instrucciones en el que se explican los propósitos de la herramienta, cómo debe usarse y las oportunidades que ofrece.</p> <p>El acceso a las instrucciones y su recomendación son uno de los primeros elementos que ve el usuario al acceder al sistema.</p>	


Requisito no funcional: El sistema estará adaptado a distintas lenguas.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
El sistema será traducido al castellano, catalán e inglés. No se descarta ampliar a más lenguas en el futuro.	Todo el contenido del sistema y la interfaz es traducible al castellano, catalán e inglés.	


Requisito no funcional: El sistema ha de poder ser usado por cualquier usuario sin que este reciba previamente algún tipo de entrenamiento antes de su uso.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
El 60% de los usuarios que van a testear la plataforma han de ser capaces de, como mínimo, completar una de las lecciones de la plataforma sin ningún tipo de soporte o ayuda en el periodo de tiempo que les sea necesario.	<p>Para la comprobación de este requisito no se ha tenido en cuenta a los 3 testers especializados que si poseen conocimientos básicos en informática.</p> <p>4 de los 5 testers en condiciones similares a las de cualquier usuario lograron completar la primera lección sin necesidad de recibir ninguna indicación.</p> <p>El único que falló lo hizo en una versión anterior del sistema a la que probaron los otros cuatro. Su fallo motivó diversos cambios en las indicaciones del menú principal y de la vista de instrucciones.</p> <p>Con un 80% de éxito, se supera el margen establecido.</p>	

Requisito no funcional: Los textos han de ser claros y directos. Dirigidos a un nivel básico de lectura.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
Evitar el uso de expresiones comunes en el ámbito de los ordenadores a excepción de aquellas que vaya a ser explicado su significado.	<p>Todo término técnico o perteneciente a la “jerga de los ordenadores” ha sido explicado si no se ha podido evitar su uso.</p> <p>El contenido de las lecciones ha sido sometido a diversas correcciones a causa de esto durante la fase de implementación y con las sugerencias u opiniones de los testers.</p>	

Requisito no funcional: Los iconos mostrados en la interfaz han de incluir un texto aclaratorio.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
Todo botón o icono debe incluir una etiqueta que indique su función.	<p>Todo elemento que dispare una funcionalidad del sistema, normalmente un botón, imagen o icono, incluye un texto que indica lo que hace.</p> <p>Además, si el usuario deja el cursor sobre estos elementos, aparece una etiqueta explicando su función.</p>	

Requisito no funcional: El sistema debe de estar preparado para expandirse cómodamente.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
El sistema ha de estar diseñado de manera que la inclusión, modificación o eliminación de las lecciones y sus elementos relacionados se haga de forma rápida y eficiente.	<p>A pesar de la dificultad y la cantidad de horas que requiere crear el contenido de una lección, su adicción, modificación o eliminación es relativamente rápida y sencilla.</p> <p>Dado el alcance y objetivos actuales de este proyecto se ha descartado que alguien que no sea el desarrollador pueda expandir el sistema, por lo que este tipo de cambios solo son posibles mediante modificaciones en el sistema de ficheros en el Front Server lo cual respeta el criterio de aceptación del requisito.</p>	

Requisito no funcional: Alguien con conocimientos básicos en el uso de ordenadores debe mostrarle cómo acceder a la plataforma web y facilitar su acceso en el futuro.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
Conocer a alguien que sepa buscar la plataforma web por Internet y pueda mostrársela al futuro usuario.	El sistema ha sido presentado a los testers por el desarrollador de este proyecto o alguien de confianza de su entorno. Ha sido esta persona quien ha facilitado las herramientas y el acceso al sistema.	

Requisito no funcional: El sistema debe ser mostrado en una plataforma web por lo que requerimos de conexión a Internet.		
Criterio de aceptación	Observaciones en el sistema	¿Se cumple con el requisito?
El sistema ha de estar constantemente actualizado y disponible.	<p>El sistema está desplegado en un servidor público de una plataforma gratuita. La gente accede a él mediante sus navegadores. A pesar de las prestaciones, el servicio está siempre disponible.</p> <p>En caso de fallo en la base de datos, alojada en un servidor ajeno al de la plataforma, no afectaría a la visualización de las lecciones puesto que están incluidas en el sistema de ficheros del Front Server.</p>	

12.2 Valoración sobre su utilidad

Una vez comprobado el cumplimiento de todos los requisitos, llega el momento de analizar las valoraciones de los cinco usuarios voluntarios, sus impresiones, el cambio que ha supuesto para ellos el uso de la herramienta y sus intenciones futuras. Para ello se les ha preguntado a cada uno por separado y se ha juntado todas sus reflexiones en la siguiente tabla de la Figura 60. Se ha incluido un breve apartado llamado “perfil” para contextualizar un poco a cada participante.

TABLA DE CONCLUSIONES FINALES DE LOS VOLUNTARIOS			
	PERFIL DEL VOLUNTARIO	OPINIÓN PERSONAL	PUNTACIÓN MEDIA
VOLUNTARIO 1	91 años, jubilada, ex peluquera	“Lo he entendido todo y está muy bien explicado, pero yo para estas cosas soy muy negada. Si me pongo, sí, luego lo saco. Pero hay que ponerse cada día un rato.”	9 lecciones completadas. 3.9/5
VOLUNTARIO 2	72 años, jubilado, ex conserje	Me ha gustado. Se entiende fácil, aunque son muchas cosas para asimilar. No creo que compre nunca por internet, pero gracias por la ayuda.”	7 lecciones completadas. 4.3/5
VOLUNTARIO 3	75 años, ama de casa	“Me ha gustado mucho y me ha parecido todo muy interesante. A veces encuentro que hay demasiada información de golpe y me pierdo un poco.”	9 lecciones completadas. 4.8/5
VOLUNTARIO 4	82 años, jubilada, ex cocinera de comedor escolar.	“Me gusta que se pueda cambiar el idioma en todo momento. Me cansa un poco tener que ir cambiando de ventana todo el rato, pero, claro, no hay otra.”	8 lecciones completadas 4.4/5
VOLUNTARIO 5	87 años, jubilada, ex enfermera	“Está bien, pero yo no necesito hacer estas cosas. Ni correo, ni... Si quiero llamar llamo al fijo. Me ha gustado ver fotos del pueblo dónde viví en Internet, eso sí.”	6 lecciones completadas 3.8/5

En general todos fueron capaces de seguir las instrucciones y de completar las lecciones propuestas (o la mayoría de ellas). Cuando se les preguntó si seguirían usando la aplicación en el futuro solo los voluntarios 1, 3 y 4 respondieron afirmativamente. Esto y el hecho de que la mayoría ha completado, por lo menos, el 66% de las lecciones hace concluir que el uso de la plataforma ha despertado el interés de los voluntarios en este ámbito tan desconocido para ellos y ha logrado enseñarles de una manera muy básica los conceptos introducidos. Esto alienta a pensar que es muy

probable que, con una dedicación constante y con un poco de esfuerzo, estas personas puedan empezar los primeros pasos hacia la autosuficiencia tecnológica.

12.3 Análisis de alternativas

En general el autor del proyecto considera la idea propuesta como una solución sólida, adecuada y que realmente resuelve el problema de la brecha digital. Se ha podido ver en el estado del arte que no existe realmente otra alternativa para ordenadores, y eso ha animado al autor a seguir con esta vía.

Sin embargo, han sido varias las personas de su círculo más cercano que le han expresado en alguna ocasión la posibilidad de integrar algo similar para dispositivos móviles puesto que son hoy en día más comunes o es más fácil tener acceso a ellos. Pero por las razones mencionadas al principio se ha mantenido fiel a la idea de la aplicación web. Además, aunque el sistema propuesto no enseña a usar un móvil inteligente, sí que muestra cómo utilizar varias herramientas que son perfectamente accesibles desde nuestros teléfonos, por lo que sí que puede tener una ligera repercusión en esta otra rama de la tecnología.

En cuanto a la distribución en general dentro de la plataforma no se ha considerado otra estrategia para estructurar y mostrar el contenido. El autor ha realizado un exhaustivo estudio de cómo presentar las lecciones y de cómo adaptarlas de la mejor forma posible para los usuarios de edad avanzada y las conclusiones extraídas han sido la base de este diseño, por lo que no ve otra manera mejor de llevar a cabo el proyecto. No al menos hasta no tener la opinión de un grupo de usuarios de un tamaño considerable que desmienta esta creencia.

Si que se ha planteado posibles mejoras descartadas por falta de tiempo o de habilidad a la hora de programar y que serían alternativas interesantes que considerar y llevar a cabo en el futuro.

Una de ellas, y de la cual ya se ha hablado brevemente en el apartado de solución de problemas del capítulo de implementación, es usar otra alternativa para el despliegue porque sí que es cierto que la limitación de 550 horas y la transición al modo inactivo no son características ideales. Pero de nuevo insistir en que era una de las opciones más viables a nivel calidad-precio y que, dadas las circunstancias, Heroku era una alternativa que cumplía con creces con las expectativas del prototipo. Si la aplicación tuviera éxito y el número de usuarios que interaccionasen con ella fuese elevado sería no solo interesante si no necesario cambiar esta plataforma distribuidora por alguna de pago que ofreciera mejores prestaciones. Y lo mismo pasaría con el sistema que gestiona la base de datos. Esto implicaría encontrar algún método de sufragar los costes de mantener el servicio. Podría incluirse publicidad, aceptar donaciones de usuarios o de sitios web de micromecenazgo¹³ o, tal vez, implementar un sistema de registro y un contenido premium accesible mediante el pago de una cuota mensual.

¹³ **Micromecenazgo:** En términos generales, es una práctica que consiste en la búsqueda de financiación mancomunada por parte de donantes independientes que, simplemente, simpatizan con la causa.

De hecho, el registro de usuarios fue una opción que estuvo sobre la mesa en la fase de diseño, pero quedó descartada debido a que realmente no aportaba ningún valor a la plataforma más que el de dificultar el acceso a usuarios con poco o ningún conocimiento en el manejo de ordenadores. Si, implementar esta funcionalidad hubiera servido para demostrar aún más las habilidades del desarrollador de cara a un punto de vista más académico, pero como ingeniero del software uno tiene que saber distinguir aquellos requisitos que el sistema necesita de cara al usuario y cuales no y, en este caso, este no era uno de ellos.

Otra alternativa descartada fue la de que cualquier usuario pudiera subir a la plataforma lecciones sobre cualquier tema relacionado con el uso de ordenadores y crear una especie de comunidad de enseñanza para gente mayor. Aunque la idea es buena, esto implicaría renunciar a la certeza de que todo el contenido que se enseña en la web es útil. A fin de cuentas, las lecciones tienen que ser comprensibles a un nivel básico de entendimiento. No pueden usar terminología o hacer referencia a conceptos que una persona mayor no entiende. No sin explicarlos, al menos. Si se quisiera mantener una especie de red social de profesores que suben contenido y de gente que puede aprender de ellos, aunque fuera reducido, debería de haber un grupo de personas dedicadas a supervisar las lecciones subidas antes de ser publicadas, algo actualmente inviable.

Las enseñanzas de las lecciones han sido elaboradas siguiendo los estrictos criterios bajos los cuales se ha diseñado la aplicación y adaptándolas a las necesidades y capacidades de la gente mayor. Detrás de cada explicación hay una razón por la cual es de esta forma y no de otra y esto, por ahora, es lo que permite que el contenido mostrado sea de utilidad. Si se hubiera añadido esta funcionalidad, esta característica fundamental no estaría presente.

Por último, mencionar que hubo más lecciones candidatas a aparecer en la plataforma, pero al final se decidió que, para el prototipo, nueve eran más que suficientes. En el siguiente capítulo veremos las desviaciones respecto a la planificación temporal y la razón por la cual quedarse únicamente con 9 lecciones fue una decisión acertada. De nuevo, si la aplicación tuviera muchos usuarios y un éxito considerable se ampliaría su contenido. Algunas de estas lecciones que nunca llegaron a ver la luz del día fueron: “Aprender a navegar por YouTube”, “Funcionalidades básicas del ordenador” (encendido, reinicio, apagado, comprobación de periféricos, uso de programas simples preinstalados en el sistema operativo...) o “Uso de editores de texto como Word” entre otros.... Se crearían, también, lecciones a partir de las sugerencias enviadas por los usuarios y se rescatarían aquellas propuestas menos votadas en la encuesta que se hizo para investigar las preferencias del arquetipo de los futuros usuarios. Seguramente habría también continuaciones de algunas de las lecciones existentes para resolución de dudas o problemas frecuentes.

12.4 Identificación de leyes y regulaciones

Para terminar el capítulo de validación, habría que dotar al proyecto por un momento de un enfoque normativo y legislativo. Al fin y al cabo, todo producto debe cumplir una serie

de ordenanzas y regulaciones. Aunque, al tratarse de un proyecto académico sin ánimo de lucro, lo cierto es que apenas se ve afectado por las regulaciones legislativas actuales del país.

Sin embargo, analizando en profundidad las capacidades y el funcionamiento interno del sistema, así como los servicios de terceros que utiliza, se ha concretado las distintas normativas que le afectan, directa o indirectamente, y si se cumple lo que dictaminan:

- yayOS acepta el envío de comentarios y de sugerencias para mejorar la calidad de su servicio. Al no poseer un sistema de autenticación de usuarios, los formularios pueden rellenarse con datos cuya veracidad no es comprobada. Deben de ser válidos en cuanto a formato, pero por ejemplo un mismo usuario puede enviar dos comentarios con nombres distintos e, incluso, que ninguno sea su verdadero nombre. A pesar de ello, sí que para el caso de las sugerencias es opcional añadir en el formulario el correo electrónico para que los desarrolladores pudieran contactar con el autor de la sugerencia si fuese necesario. Considerando como sensible esta información, ha de estar sujeta a la Directiva 95/46/CE (Reglamento general de protección de datos) y, más concretamente, por la ley orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales aprobada por las Cortes Generales de España en base a la directiva europea.

Al no usar una base de datos propia, la protección de los datos que en ella se almacena corre a cargo del servicio externo que la gestiona, en este caso el servicio de alojamiento en nube Atlas de MongoDB. Revisando las condiciones de uso de este servicio, MongoDB asegura el cumplimiento de la normativa europea, pero indica que él mismo usa servicios de otras compañías para alojar las bases de datos y, en lo que a yayOS se refiere, esta otra compañía es Amazon Web Services que, según afirma en su Modelo de Responsabilidad Compartida, él es responsable de la seguridad de su almacenamiento.

- En cuanto al uso de cookies, yayOS no recoge ninguna por lo que no tendría ninguna obligación legislativa. Heroku, la plataforma de despliegue en la que se aloja el sistema, sí que recoge cookies, pero recoge las mismas que el navegador por lo que, a lo que el sistema respecta, no está sujeto a ninguna ley.
- El resto de normativas, como el Real Decreto 1112/2018 del 7 de septiembre sobre accesibilidad de los sitios web y aplicaciones para dispositivos móviles del sector público o similares, no serían aplicables a este proyecto puesto que no tienen una finalidad económica ni están relacionadas con los servicios públicos del país.
- Finalmente, este trabajo debe cumplimentar todo aquello especificado en la normativa del TFG del Grado en Ingeniería Informática de la Facultad de Informática de Barcelona.

13. Seguimiento del proyecto

En la planificación temporal de esta memoria se calculó que se requerían un total de 562 horas para completar el proyecto, aunque, teniendo en cuenta las fechas de inicio, finalización y la cantidad de horas que se pretendía dedicar, se podía permitir ampliar este margen de tiempo hasta 595 horas.

Tanto el proceso de desarrollo de la aplicación web como el redactado de esta memoria fueron finalizados antes de la fecha límite por lo que, técnicamente, los plazos fueron cubiertos y la planificación fue acertada. Sin embargo, hubo diversas desviaciones a lo largo de todo el proceso de las cuales se dejará constancia en este apartado.

Echando un vistazo de nuevo al diagrama de Gantt de la Figura 16, podemos observar que la última tarea terminaba la semana del 12 al 18 de abril. Esto sin contar las tareas de documentación y seguimiento que se alargaban hasta la fecha de entrega el 1 de junio y que no eran continuas. La implementación de la aplicación web terminó realmente una semana después, el 28 de abril, suponiendo unas 50 horas extras más de las esperadas. ¿A qué se debió esta desviación?

En parte, hubo una subestimación de las horas que se creían necesarias para diseñar el contenido de las lecciones ya que no se tuvo en cuenta el trabajo que supondría traducir las diapositivas a mano. Esto hizo que se dedicaran 120 horas reales sobre las 45 previstas. A pesar de ello, hubo una segunda desviación que disminuyó el impacto de esta y es que se pensó que la implementación de los componentes de la interfaz para cada lección sería individual cuando, finalmente, se reprodujo todo de una misma manera por lo que la cantidad de horas dedicadas a eso disminuyó de 105 a 45. Por tanto, estaríamos hablando de un incremento de 75 horas frente al ahorro de 60, por lo que solo se suman realmente 15 horas.

Por otro lado, ya se ha comentado que se dedicaron 30 horas en vez de 25 al autoaprendizaje, lo que sumaría otras 5 horas extras al recuento. Y, por último, deberían sumarse otras 30 horas extras dedicadas a la implementación de la API y a problemas de vinculación entre el Front Server, el Back Server y la base de datos. En total sale un incremento de 50 horas.

Ahora bien, ¿ha supuesto un problema? Realmente no pues, como se ha dicho, el proyecto ha sido entregado dentro del plazo y se contaba con la suficiente holgura en el margen de tiempo establecido como para permitirse este tipo de contratiempos. Por un lado, tenemos las 33 horas que sobraban de la planificación temporal y, por otro, podemos considerar que se ha activado el plan de contingencia de “Inexperiencia en las tecnologías utilizadas”, cuya probabilidad era alta, y que se han requerido 17 horas de las adicionales disponibles. Estas 17 horas extras se han conseguido invirtiendo el tiempo sobrante de tareas finalizadas ante de tiempo tal y como se sugirió en la formulación del plan de contingencia. Principalmente de tareas de contextualización. Ambas suman las 50 por lo que la planificación ha podido soportar esta desviación.

Otros cambios que no han tenido un impacto real en la misma ha sido la alteración del orden de las lecciones que se iban a enseñar en la plataforma web (y por tanto el orden de su diseño e implementación). Mientras que, en un inicio, las tareas estaban

ordenadas de tal manera que primero iba el diseño de la lección acerca de las videollamadas, luego la de cómo buscar por Internet y, finalmente, sobre el sistema de ficheros del ordenador, se creyó más conveniente mostrar primero cómo manejar el buscador de Internet, luego los ficheros y, para acabar, Skype. Este cambio no ha supuesto ningún coste temporal extra, ni económico, ni de ningún otro tipo. Se ha hecho por conveniencia a la hora de introducir según que conceptos a los usuarios. Es más sencillo manejar el proceso de instalación de Skype si antes se ha adquirido experiencia en el uso de las carpetas y el manejo de los ficheros y, para ello, es más sencillo si primero se aprende a moverse por Internet, cosa que no ofrecía el orden inicial.

14. Conclusiones

Llegados al final del proceso solo queda la reflexión. ¿Se han cumplido los objetivos?, ¿se podría haber hecho mejor?, ¿qué habría que cambiar y qué habría que mantener?, ¿cuáles son los siguientes pasos a dar? Este apartado tratará, a continuación, de contestar todas estas preguntas.

14.1 Cumplimiento de los objetivos del proyecto

La única manera para saber si el producto desarrollado es útil es ver si se cumplen los objetivos que se propuso desempeñar. Unos objetivos que, recordemos, surgían de las propias necesidades de las partes interesadas en el proyecto. Comprobémoslo uno por uno. Acorde a los objetivos:

¿El sistema presenta una solución en un formato adaptado a las posibles capacidades deterioradas de una persona mayor?

Previamente se ha realizado un estudio sobre las características visuales del sistema y, a partir de ahí, se han elaborado una serie de requisitos no funcionales que, a posteriori, se ha visto cómo se cumplían a rajatabla en la validación de requisitos. Por lo tanto, si, la solución posee un formato correctamente adaptado.

¿El sistema presenta una solución de uso intuitivo, sencillo y que permita a alguien su manejo sin que le sea necesario ningún tipo de conocimiento previo?

De nuevo, este mismo estudio ha permitido estructurar el contenido y mostrarlo de una forma intuitiva, sencilla y de fácil acceso para gente no acostumbrada al uso de interfaces. Se ha podido ver, tanto en el cumplimiento de los requisitos funcionales como en las valoraciones de los usuarios voluntarios, que nunca se ha producido ningún

problema o inconveniente que haya dificultado el acceso a dicho contenido. Cabe recalcar que el 80% de los usuarios lograron terminar la primera lección sin ayuda, por lo que fueron perfectamente capaces de manejar la plataforma.

¿El sistema presenta unas lecciones sencillas que permitan aprender a hacer cosas básicas con un ordenador y elegidas por una encuesta a un grupo de usuarios representativo?

Todas las lecciones introducen conceptos simples haciendo uso de un lenguaje básico de entendimiento y evitando en todo momento el uso de terminología específica. A no ser que sea necesario en cuyo caso se explica con antelación su significado. Los requisitos no funcionales relacionados con este aspecto han satisfecho sus criterios de aceptación.

Los temas tratados fueron escogidos mediante una encuesta a potenciales usuarios como ya se comentó en capítulos anteriores. 5 lecciones salen directamente de las opciones más votadas: Aprender a hacer videollamadas en primer lugar, comprar online en segundo, usar un correo electrónico la tercera y aprender a buscar por Internet y el uso de Facebook empatadas en cuarta posición. La lección de Google Maps no fue propuesta, pero si sugerida por varios encuestado de forma indirecta, así que también se incluyó. El resto (uso de teclado y ratón, creación de una cuenta de Google y gestión de ficheros) fueron añadidas por conveniencia o necesidad.

¿El sistema presentar una solución que sea fácilmente ampliable y adaptable a los cambios que experimenten dichas necesidades con el tiempo y a las actualizaciones de los servicios que ofrece la red?

Aunque el proceso de creación de nuevo contenido es laborioso y costoso porque tiene que cumplir una serie de estrictas características, su adicción, modificación y/o eliminación es relativamente sencilla como se ha podido observar a lo largo del diseño e implementación. Únicamente se debe interaccionar con el sistema de ficheros del Front Server y eso, por ahora, es un privilegio que solo el desarrollador posee.

Además, como objetivo secundario, ¿el sistema potencia la curiosidad del usuario para que se vea capaz de explorar la red por su cuenta y de profundizar en estos nuevos conocimientos?

La sensación de éxito y satisfacción está estrechamente relacionada con el incremento en la motivación. Se espera que la solución propuesta supere las expectativas de los usuarios, pues se ha diseñado a partir de sus objetivos personales, y que su capacidad de superación les anime a proseguir avanzando en el uso de las nuevas tecnologías. Por el momento, el 60% de los voluntarios aceptaría seguir usando la aplicación por su cuenta. Por supuesto, un grupo de prueba de tan solo 5 voluntarios no es significativo.

Pero es un buen comienzo y, en lo que al marco de este proyecto se refiere, sí que cumple con el objetivo asignado.

14.2 Limitaciones y dificultades

El proyecto ha ido avanzando a un ritmo constante y sin retrasos gracias a un constante esfuerzo y dedicación. Y, por descontado, no siempre ha sido fácil.

La pandemia ha afectado a nuestras vidas de muchas maneras. En cuanto a capacidad y en cuanto a motivación. Además, el distanciamiento social, las restricciones de movimiento y que el grupo de usuarios al que va dirigido este producto sea categorizado como de riesgo ha hecho más difícil encontrar voluntarios a los que encuestar o captar como testers, y esto habrá vuelto los requisitos tal vez menos precisos. Un grupo de potenciales usuarios más numeroso habría resultado en una mejor definición de sus necesidades y, habría proporcionado más opiniones y, por ende, una mayor detección de problemas o, al menos, de sugerencias en la parte de refactorizado.

Por otro lado, no se puede ignorar que la falta de conocimientos técnicos a la hora de implementar era también una limitación. Aunque se haya enfocado en todo momento como una motivación, puesto que lo ha sido realmente ya que este proyecto ha permitido a su autor expandir sus conocimientos, lo cierto es que es innegable que la falta de práctica hace más costoso el avance y la falta de habilidad limita muchas veces el abanico de opciones a elegir.

Hablando de carencias, también habría que tener en cuenta que los recursos hardware empleados han sido suficientes, sí, pero podrían haber sido mucho mejores. Y esa es otra limitación pues ralentiza el proceso a la hora de utilizar según que recursos software exigentes en lo que a recursos de CPU, gráfica o memoria se refiere. La compilación, las pruebas, el despliegue, las instalaciones de dependencias, las repentinas actualizaciones de los programas utilizados... A esto habría que añadir, además, un entorno de trabajo que no siempre idílico.

14.3 Conocimientos aplicados y adquiridos

Para el desarrollo de este proyecto se han aplicado un conjunto de disciplinas que le han sido enseñadas a su autor a lo largo de la carrera y, más concretamente, durante sus estudios en la especialización de ingeniería del software.

Yendo por orden, encontramos los primeros conocimientos en la contextualización del proyecto. La definición del problema, la búsqueda de una solución, su alcance, los requisitos que debe de tener esta solución... Todos estos apartados del desarrollo de software le han sido enseñados en asignaturas como Ingeniería de Requisitos (ER) y Gestión de Proyectos de Software (GPS). Esta última también ha ayudado durante la elaboración de la planificación temporal, la detección de riesgo, la creación de planes de contingencia y la gestión económica pues se tuvo que pasar por este mismo proceso

en trabajos de la asignatura. Fue ahí donde por primera vez trabajó con una metodología Kanban, la que ha sido usada en este proyecto como se comentaba anteriormente.

Desde que el autor emprendió el grado en informática se le ha concienciado en el uso responsable y sostenible de la informática y la memoria contiene un apartado con un análisis de sostenibilidad desde un punto de vista ecológico, económico y social. Se ha elaborado este capítulo con la ayuda del temario de la optativa de Arquitectura de PC (APC), donde se ha enseñado a redactar informes de esta índole. Esta asignatura ha proporcionado también consejos en el análisis de fuentes de información y de su veracidad.

En cuanto al diseño han sido indispensables los conocimientos adquiridos en diversas asignaturas, comunes y de especialidad, del Departamento de Ingeniería de Servicios y Sistema de Información (ESSI) como lo son la Introducción a la Ingeniería del Software (IES), Arquitectura del Software (AS), Aplicaciones y Servicios Web (ASW), Proyecto de Ingeniería del Software (PES) y otras de otros departamentos como Estructura de Datos y Algoritmos (EDA), Proyectos de Programación (PROP) o Proyecto Aplicado de Ingeniería (PAE). De aquí ha surgido la especificación de requisitos, la definición del modelo conceptual y de comportamiento, el diseño de la arquitectura lógica y física y, en definitiva, la experiencia en el desarrollo de software en general.

De estas asignaturas, algunas fueron también útiles en la parte de implementación. Principalmente ASW, PROP y PES. Y, aunque a nivel de programación se ha recurrido bastante al autoaprendizaje porque no existen asignaturas que enseñen en concreto las tecnologías requeridas para este proyecto, obviamente todo conocimiento nuevo es posible gracias a las bases de programación que obtuvo en las asignaturas de Programación 1 y 2 (PRO1 y PRO2). Gracias al desarrollo de este proyecto el autor ha adquirido conocimientos en el uso de React, CSS, JSX, y ha ganado práctica en la creación de una API, en el despliegue de proyectos y en la vinculación con bases de datos. También le ha servido para profundizar sus habilidades con JavaScript y en el proceso de desarrollo en general.

Como podemos ver son muchos los conocimientos involucrados en este trabajo de final de grado. Una forma de agrupar todo lo aprendido a lo largo de 4 años de carrera.

14.4 Justificación de las competencias

Este TFG está sujeto a una serie de competencias que su autor tiene que satisfacer. Por competencia se entiende un conjunto de conocimientos, habilidades, actitudes y valores que le capacitan a resolver un problema en un contexto académico con una serie de garantías. Estas competencias fueron elegidas por él mismo y validadas por la tutora a cargo de la supervisión de este proyecto. Además, a cada una de ellas se le ha sido asignada un grado de implicación pudiendo variar entre “En profundidad”, “bastante” o “un poco”. Define la relevancia que tiene cada una de ellas y su importancia de cara a su evaluación.

CES1.1: Desarrollar, mantener y evaluar sistemas y servicios software complejos y/o críticos. [En profundidad]

Este proyecto se ha basado en la transformación de un problema y unos requisitos a una solución, y se ha hecho mediante el desarrollo, mantenimiento y evaluación de un sistema software complejo. Se han identificado en profundidad los requisitos funcionales y no funcionales y, a partir de ellos, se ha diseñado el sistema para, posteriormente, implementarlo y, finalmente, validarlo, completando así todas y cada una de las fases del proceso. Este ha sido el propósito central del proyecto, y por tanto tiene asignada la competencia con más peso de todas las relacionadas. El sistema no es únicamente una aplicación si no un conjunto de elementos distintos que funcionan de forma independiente y que se comunican los unos con los otros. Por un lado, tenemos una interfaz visible mediante un navegador que interacciona con el usuario, y, por el otro, una API rest que procesa las peticiones y que recoge datos alojados en una base de datos externa. Dos elementos que funcionan como un solo y que permiten satisfacer todas las necesidades y cumplir todos los objetivos que este proyecto pretendía.

CES1.3: Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción de software que se pudieran presentar. [Un poco]

Durante las primeras fases del proceso se detectaron todos aquellos obstáculos potenciales y aquellos riesgos que podían suponer una amenaza real al avance del proyecto. A pesar de que esta competencia tiene un grado de implicación menor al resto, se ha actuado acorde a las expectativas y se generaron estrategias para evitar esas situaciones y, en caso de hacerlo, se formularon planes de contingencia para mitigar al máximo sus efectos. Una prueba de que estos planes fueron elaborados correctamente es que algunos de ellos tuvieron que llegar a aplicarse a causa de diversas desviaciones en la planificación temporal y, sin embargo, el proyecto terminó en la fecha prevista cumpliendo con todos los plazos de entrega. Esta desviación ha sido comentada en el capítulo13 de esta memoria.

Además, hubo una constante monitorización del estado actual del proyecto, así como del seguimiento de la planificación y de los recursos requeridos para el desarrollo lo que facilitó aún más la detección precoz de riesgos y, por consecuente, su evasión.

CES2.1: Definir y gestionar los requisitos de un sistema software. [Bastante]

Se ha venido diciendo en todos los capítulos. La definición y especificación de los requisitos era una tarea crucial para este proyecto pues la verdadera dificultad residía en adaptar la interfaz y el contenido a un tipo de usuario que no tiene ninguna experiencia previa en el uso de interfaces y que no entiende el contenido que se le presenta. Aunque el trabajo sea el desarrollo de un producto software y por ende la competencia relacionada a ese aspecto sea catalogada como “en profundidad”, esta es

una parte de tremenda importancia y por eso tiene una implicación categorizada como “bastante”. Y se ha cumplido la competencia pues, como ha podido observarse en la contextualización del proyecto, se ha realizado un extenso estudio de las dificultades del problema y, por consecuente, las cualidades que ha de tener su solución. A partir de aquí se definieron una serie de requisitos mediante la notación Volere que, a posteriori, fueron distribuidos entre distintos casos de uso y mediante una serie de directrices e indicaciones en el diseño. Para terminar, se evaluó y concluyó que todos los requisitos cumplían con los criterios de satisfacción mínimo por lo que observamos que se han definido y gestionado acorde a lo esperado.

14.5 Futuro del proyecto

Aunque este sea el último apartado de esta memoria y, por tanto, de este proyecto, en realidad es el principio puesto que yayOS no termina aquí. Es imprescindible reafirmar la necesidad de plataformas como la propuesta para facilitar la inclusión de las personas mayores en las nuevas tecnologías y en la reducción de la brecha digital. Es por ello que se planea seguir expandiendo y evolucionando el sistema más allá de los propósitos académicos que originalmente le dieron forma.

A corto plazo, se pretende eliminar todas aquellas carencias que tiene el prototipo. Principalmente cambiar o mejorar las condiciones de la plataforma de despliegue. Así el acceso a yayOS sería más dinámico y sencillo. Como se ha comentado antes, debería analizarse si existen otras alternativas que ofrecieran mejores prestaciones de forma gratuita y, si ninguna fuera lo suficientemente apropiada, se recurriría a una de pago y se trataría de sufragar los costes de mantenimiento empezando por micromecenazgo (lo que implicaría una campaña de difusión por redes sociales contundente) y, en caso de no ser suficiente, la adición de publicidad.

Una vez hecho esto, tendría que asegurarse que el sistema se visualiza correctamente en cualquier resolución de pantalla porque se han detectado pequeños fallos estéticos en resoluciones muy pequeñas. Habría que ajustar eso para que se visualizara correctamente en cualquier dispositivo. Incluso móviles.

El siguiente paso sería contar con un grupo de usuarios voluntario mucho mayor que el que se ha usado ahora y detectar más precisamente aquello que sirve y aquello que debe mejorarse: cambios en las explicaciones, en el diseño de la interfaz, en la estructuración del contenido... También habría que realizar un segundo estudio para conocer qué más cosas les gustaría aprender a hacer con un ordenador para saber así en qué aspectos enfocar las futuras lecciones.

Si la aplicación tuviera muchos usuarios sería interesante replantearse la implantación de un sistema de autenticación. Es cierto que, en este momento, el hecho de que cada usuario tenga su propia sesión no aporta realmente nada como ya se ha comentado. Pero este cambio, en una perspectiva de un sistema mayor, sí que puede suponer mejoras interesantes. Si aumenta el número de usuarios debería de haber alguna forma

de distinguir quién publica cada comentario y no como está actualmente. Si aumenta el número de lecciones el sistema podría detectar aquellas que más utiliza alguien en concreto y recomendarle que pruebe otras que tengan relación. Si aumenta el interés y la interacción podrían organizarse charlas o contenido esporádico al que acceder mediante la plataforma. La implementación de una funcionalidad abre las puertas a la creación de muchas otras. Por supuesto este inicio de sesión sería opcional y solo serviría para permitir el acceso a funcionalidades concretas. El aprendizaje no debe tener obstáculos, así que el uso de las lecciones no requeriría de un inicio de sesión. Pero si subir un comentario, por ejemplo.

Más a largo plazo, se podría contactar con organizaciones, asociaciones o entidades que promovieran la reducción de la brecha digital y que estuvieran dispuestas a donar ordenadores de segunda o, incluso, de primera mano. De esta manera, yayOS podría implementar una funcionalidad que permitiera adquirir o alquilar estos ordenadores con el fin de acercar las nuevas tecnologías a aquellas personas que no tienen acceso. Recordemos que, aunque este no era un problema que este proyecto pueda solucionar, sí que es uno de los factores que agranda la brecha y estaría bien que yayOS pudiera ayudar también de alguna forma.

Por último, se ampliaría la cantidad de lenguajes en los que el sistema está disponible. Que la capacidad de aprender tuviera los menos límites posibles.

14.6 Reflexión final

Los cambios pueden ir para mejor o para peor, pero rara vez son inevitables. Como estudiante de informática creo que la transición digital es una mejora. Puede que esa mejora no siempre sea visible pues estamos acostumbrados a hacerlo de una forma y de repente nos obligan a hacerla de otra. Pero todo cambio está justificado por alguna razón de tal manera que realmente, de un modo u otro, sí que supone un progreso. Ahora bien, esto no es así cuando la transición implica la exclusión. Todo el mundo tiene el mismo derecho a avanzar y, aunque haya gente reticente a ese cambio, al menos debe existir la oportunidad para las personas que sí lo elijan. Y eso es algo que, por desgracia, no ha sucedido en la pandemia.

Ya se decía en las motivaciones personales. El autor de este proyecto ha vivido en primera persona el problema que supone una adaptación tecnológica mal gestionada. Y espera de todo corazón que el sistema propuesto pueda dar ese empujoncito para que las personas que quieran formar parte del cambio puedan hacerlo y, los que no, mostrarles las oportunidades nuevas que el cambio ofrece y animarles a hacerlo.

Estos cuatro meses han servido para ponerse en la piel de los más mayores y ver el mundo con sus ojos y, diciéndolo claramente, con sus dificultades. Y ojalá sean muchas las personas a las que esta aplicación, este proyecto, pueda ayudar.

15. Referencias bibliográficas

- [1] "Gerontechnology - Wikipedia." <https://en.wikipedia.org/wiki/Gerontechnology> (accessed Apr. 03, 2021).
- [2] "Eldy Seniors Computer Software for Elderly." <http://www.eldy.es/> (accessed Feb. 28, 2021).
- [3] "Juegos educativos Vedoque de Matemáticas, lectura, inglés para aprender jugando en Primaria e Infantil." <https://vedoque.com/> (accessed Apr. 03, 2021).
- [4] "Senior Computer Skills Index | Learn basic computer skills." <https://www.skillfulsenior.com/skills/index.php> (accessed Feb. 28, 2021).
- [5] "Cursos gratis de Informática e Internet en GCFGlobal." <https://edu.gcfglobal.org/es/topics/informatica-e-internet/> (accessed Feb. 28, 2021).
- [6] "Lessons." <https://www.meganga.com/lessons/> (accessed Feb. 28, 2021).
- [7] W. James, "Brain Plasticity and Habit in William James: an Antecedent for Social Neuroscience," 2012, doi: 10.5209/rev_PSLA.2012.v3.n1.38737.
- [8] P. Lally, C. H. M. van Jaarsveld, H. W. W. Potts, and J. Wardle, "How are habits formed: Modelling habit formation in the real world," doi: 10.1002/ejsp.674.
- [9] "Designing Apps For Seniors: 5 Traits Worth Considering - UX Studio." <https://uxstudioteam.com/ux-blog/apps-for-seniors/> (accessed Feb. 28, 2021).
- [10] "Age-friendly: UI UX design thinking for senior citizens | F5 Studio." <https://f5-studio.com/articles/age-friendly-ui-ux-design-thinking-for-senior-citizens/> (accessed Feb. 28, 2021).
- [11] "Enfocar el diseño UX para las personas mayores - Design." <https://design.es/como-acercar-el-diseno-ux-a-las-personas-mayores/> (accessed Feb. 28, 2021).
- [12] "Emotional Design | Book by Don Norman." <https://www.nngroup.com/books/emotional-design/> (accessed Apr. 03, 2021).
- [13] "Herramienta DAFO." <https://dafo.ipyme.org/Home> (accessed Feb. 28, 2021).
- [14] "Introduction to Test Driven Development (TDD)." <http://agiledata.org/essays/tdd.html> (accessed Feb. 28, 2021).
- [15] "Extreme Programming: A Gentle Introduction." <http://www.extremeprogramming.org/> (accessed Feb. 28, 2021).
- [16] "Manifesto for Agile Software Development." <http://agilemanifesto.org/> (accessed Feb. 28, 2021).
- [17] "Kanban frente a scrum | Atlassian." <https://www.atlassian.com/es/agile/kanban/kanban-vs-scrum> (accessed Feb. 28, 2021).

- [18] “Adobe XD | Herramienta rápida y potente de diseño y colaboración de experiencias e interfaces de usuario.” <https://www.adobe.com/es/products/xd.html> (accessed Mar. 04, 2021).
- [19] “IntelliJ IDEA: el IDE de Java eficaz y ergonómico de JetBrains.” <https://www.jetbrains.com/es-es/idea/> (accessed Mar. 04, 2021).
- [20] “Microsoft Word: software de procesamiento de texto | Microsoft 365.” <https://www.microsoft.com/es-es/microsoft-365/word> (accessed Apr. 03, 2021).
- [21] “Robo 3T | Free, open-source MongoDB GUI (formerly Robomongo).” <https://robomongo.org/> (accessed Jun. 04, 2021).
- [22] “Postman | The Collaboration Platform for API Development.” <https://www.postman.com/> (accessed Jun. 04, 2021).
- [23] “Trello.” <https://trello.com/es> (accessed Feb. 28, 2021).
- [24] “Bitbucket | The Git solution for professional teams.” <https://bitbucket.org/product/> (accessed Feb. 28, 2021).
- [25] “GitHub: Where the world builds software · GitHub.” <https://github.com/> (accessed Feb. 28, 2021).
- [26] “Sourcetree | Free Git GUI for Mac and Windows.” <https://www.sourcetreeapp.com/> (accessed Feb. 28, 2021).
- [27] “Diagrama de Gantt - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Diagrama_de_Gantt (accessed Mar. 04, 2021).
- [28] “React – Una biblioteca de JavaScript para construir interfaces de usuario.” <https://es.reactjs.org/> (accessed Mar. 04, 2021).
- [29] “Material-UI: A popular React UI framework.” <https://material-ui.com/> (accessed Mar. 04, 2021).
- [30] “Búsqueda de empleo en Glassdoor | Encuentra el empleo perfecto para ti.” <https://www.glassdoor.es/index.htm> (accessed Mar. 09, 2021).
- [31] “Ofertas de trabajo, bolsa de trabajo | Buscar empleo en Indeed España.” <https://es.indeed.com/> (accessed Mar. 09, 2021).
- [32] “Ordenador Port. Acer E5-573g-50ky 15,6"/Ci5/4gb/1t - Portátil - Ordenadores Portátiles - Informática - Tecnología | Kómpratelo.” <https://www.kompratelo.com/tecnologia/informatica/ordenadores-portatiles/portatil/ordenador-port-acer-e5-573g-50ky-15-6-ci5-4gb-1t.html> (accessed Mar. 21, 2021).
- [33] “Monthly and yearly plans with JetBrains Toolbox.” <https://www.jetbrains.com/store/#commercial?billing=yearly> (accessed Mar. 09, 2021).
- [34] “Master en Frameworks JavaScript: Aprende Angular, React, Vue | Udemy.” <https://www.udemy.com/course/master-en-frameworks-javascript-aprende-angular-react-vue-js/?couponCode=FEBRERO21> (accessed Mar. 08, 2021).

- [35] “Espacio de coworking en Barcelona | Meet BCN - Coworking Barcelona.” <https://meetbcn.com/coworking-barcelona/espacio-coworking-barcelona/> (accessed Mar. 08, 2021).
- [36] S. Robertson and James Robertson, *Mastering the requirements process*, Third Edition. Addison-Wesley, 2013.
- [37] “(No Title).” https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use-case_2_0_jan11.pdf (accessed Apr. 05, 2021).
- [38] “1. Layered Architecture - Software Architecture Patterns [Book].” <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html> (accessed Apr. 18, 2021).
- [39] “Tutorial de Diagrama de Despliegue | ¿Qué es un Diagrama de Despliegue.” <https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/> (accessed Apr. 30, 2021).
- [40] “How Color Impacts Conversion Rates and UX | UserTesting Blog.” <https://www.usertesting.com/blog/color-ux-conversion-rates> (accessed Apr. 18, 2021).
- [41] “Managed MongoDB Hosting | Database-as-a-Service | MongoDB.” <https://www.mongodb.com/cloud/atlas2> (accessed Apr. 29, 2021).
- [42] “Cursos en línea: aprende de todo y a tu propio ritmo | Udemy.” https://www.udemy.com/?utm_source=adwords-brand&utm_medium=udemyads&utm_campaign=NEW-AW-PROS-Branded-Search-SP-SPA_.ci_.sl_SPA_.vi_.sd_All_.la_SP_.&tabei=7&utm_term=_.ag_53604040718_.ad_254061738916_.de_c_.dm_.pl_.ti_kwd-310556426868_.li_1005424_.pd_.&gclid=CjwKCAjwnPOEBhA0EiwA609ReQvrX9D4O7mxLFxsUFzdh5shCzUCsXB8H__husSn51fhUisOQzhpmxoCvR8QAvD_BwE (accessed May 13, 2021).
- [43] “React Router: Declarative Routing for React.js.” <https://reactrouter.com/> (accessed Apr. 03, 2021).
- [44] “i18next documentation.” <https://www.i18next.com/> (accessed Apr. 03, 2021).
- [45] “axios - npm.” <https://www.npmjs.com/package/axios> (accessed May 02, 2021).
- [46] “dotenv - npm.” <https://www.npmjs.com/package/dotenv> (accessed May 02, 2021).
- [47] “react-moment - npm.” <https://www.npmjs.com/package/react-moment> (accessed May 02, 2021).
- [48] “simple-react-validator - npm.” <https://www.npmjs.com/package/simple-react-validator> (accessed May 02, 2021).
- [49] “sweetalert2 - npm.” <https://www.npmjs.com/package/sweetalert2> (accessed May 02, 2021).
- [50] “react-star-rating-component - npm.” <https://www.npmjs.com/package/react-star-rating-component> (accessed May 02, 2021).

- [51] "react-star-ratings - npm." <https://www.npmjs.com/package/react-star-ratings> (accessed May 02, 2021).
- [52] "react-magnifier 3.0.4 on npm - Libraries.io." <https://libraries.io/npm/react-magnifier> (accessed May 02, 2021).
- [53] "Node.js." <https://nodejs.org/es/> (accessed May 02, 2021).
- [54] "express - npm." <https://www.npmjs.com/package/express> (accessed May 02, 2021).
- [55] "body-parser - npm." <https://www.npmjs.com/package/body-parser> (accessed May 02, 2021).
- [56] "mongoose - npm." <https://www.npmjs.com/package/mongoose> (accessed May 02, 2021).
- [57] "validator - npm." <https://www.npmjs.com/package/validator> (accessed May 02, 2021).
- [58] "Vmoraís22/yayOS." <https://github.com/Vmoraís22/yayOS> (accessed May 08, 2021).
- [59] "Presentational and Container Components | by Dan Abramov | Medium." https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0 (accessed May 07, 2021).
- [60] "Vmoraís22/yayos-back." <https://github.com/Vmoraís22/yayos-back> (accessed May 08, 2021).
- [61] "Model–view–controller - Wikipedia." <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller> (accessed Apr. 18, 2021).
- [62] "Website resolution test." http://www.infobyip.com/testwebsiteresolution.php?url=http%3A%2F%2Flocalhost%3A3000%2Fleccion1&width=1366&height=768&in_browser=true (accessed May 04, 2021).
- [63] "Servicios de cloud computing | Google Cloud." <https://cloud.google.com/> (accessed Mar. 23, 2021).
- [64] "Creación rápida de aplicaciones móviles y web | AWS Amplify | Amazon Web Services." <https://aws.amazon.com/es/amplify/> (accessed Mar. 23, 2021).
- [65] "Platform as a Service | Heroku." <https://www.heroku.com/platform> (accessed Mar. 23, 2021).

Anexos

Anexo 1

En este anexo se adjunta una copia de la encuesta distribuida a usuarios potenciales de la plataforma para conocer sus necesidades y expectativas.

¿Qué te gustaría aprender a hacer con un ordenador?

Encuesta destinada a gente con escasa o nula experiencia en el uso de ordenadores. Principalmente gente de avanzada edad. Se agradece la máxima sinceridad posible.

1. Eres... *

☐ Hombre.

☐ Mujer.

2. ¿Cuántos años tienes? *

☐ Menos de 65 años.

☐ Entre 65 y 79 años.

☐ Entre 80 y 89 años.

☐ 90 años o más.

3. ¿Tienes un ordenador en casa o en la de un familiar o amigo? *

☐ Si.

☐ No.

4. ¿Tienes idea de cómo usar un ordenador? *

- ☐ No.
- ☐ Muy poco.
- ☐ Me defiendo.
- ☐ Si.

5. ¿Te gustaría aprender lo básico? *

- ☐ Si
- ☐ Podría intentarlo.
- ☐ No.

6. En el caso de haber respondido que "No" a la pregunta anterior, ¿podrías marcar el motivo? Puedes seleccionar más de una respuesta.

- ☐ Porque no lo necesito.
- ☐ Porque me parece muy complicado y no me veo capaz de aprender a usarlo.
- ☐ Porque me da miedo hacer algo mal.
- ☐ Porque si necesito hacer algo ya lo hace alguien por mí.
- ☐ Porque no me apetece.
- ☐ Otra razón.

7. Imagínate por un momento que pudieras, de repente, aprender a usar fácilmente un ordenador. ¿Qué te gustaría ser capaz de hacer? De nuevo, puedes seleccionar más de una respuesta. *

- ☐ Me gustaría tener una red social con la que hablar con mi familia y amigos. Incluso con los que he perdido el contacto.
- ☐ Me gustaría tener un correo electrónico con el que recibir y enviar mensajes no solo a mi familia o amigos, si no a médicos, oficinas de administración, etc.
- ☐ Me gustaría poder comprar por Internet de forma segura.
- ☐ Me gustaría poder hacer videollamadas con mis seres queridos.
- ☐ Me gustaría aprender a hacer trámites a través del ordenador.
- ☐ Me gustaría aprender a usar el ordenador: escribir por el teclado, buscar cosas por Internet, organizar mis cosas...
- ☐ Me gustaría poder ver las noticias desde Internet.
- ☐ Me gustaría poder ver películas y series bajo demanda.

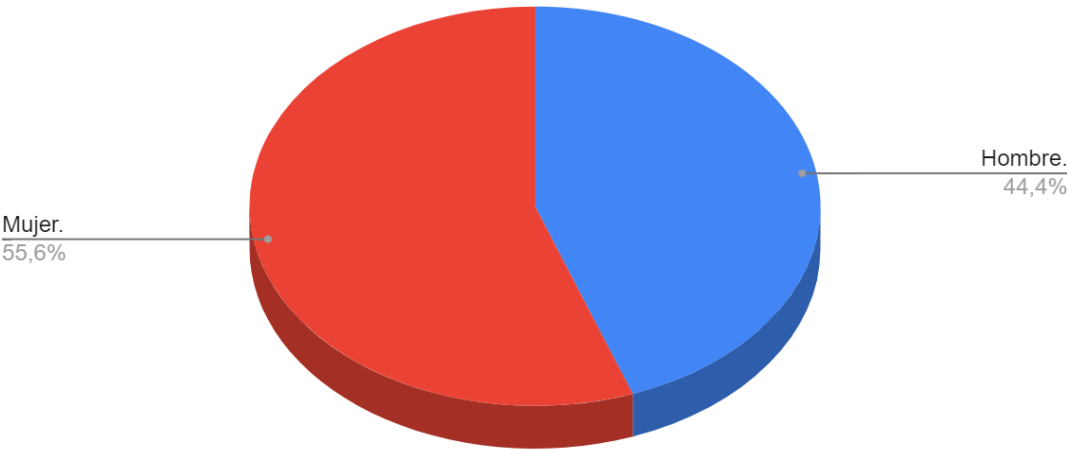
8. Si hay alguna otra razón por la cual te gustaría aprender a usar un ordenador, por favor coméntela brevemente a continuación:

9. Estoy desarrollando una plataforma web que permita a gente de avanzada edad aprender a usar un ordenador. Cosas sencillas. Si estás interesado en probarla cuando esté lista y, a lo mejor, empezar a aprender realmente a usar un ordenador escribe aquí tu nombre y la persona que te ha pasado esta encuesta.

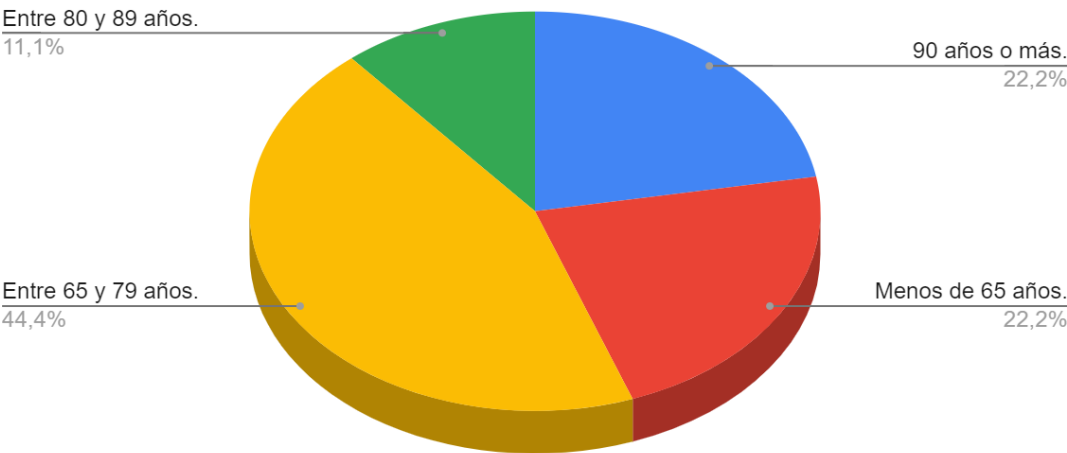
Anexo 2

En este anexo se detallan los resultados de la encuesta adjunta en el anexo 1. Se mostrarán los datos en un formato adaptado para su mejor comprensión y, posteriormente, se incluirá un enlace a la tabla de datos original. Dadas sus dimensiones es inviable incluirla en el formato de esta memoria.

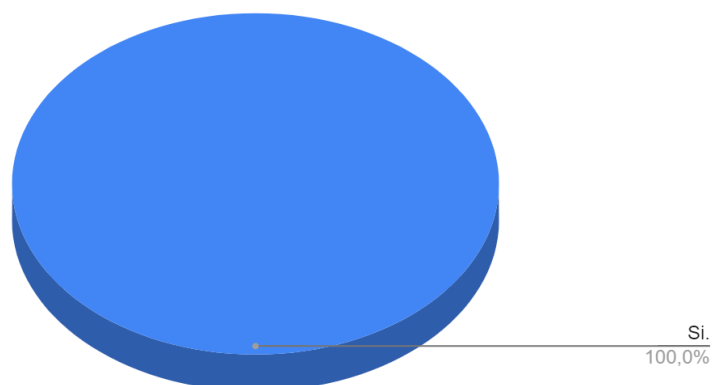
Eres...



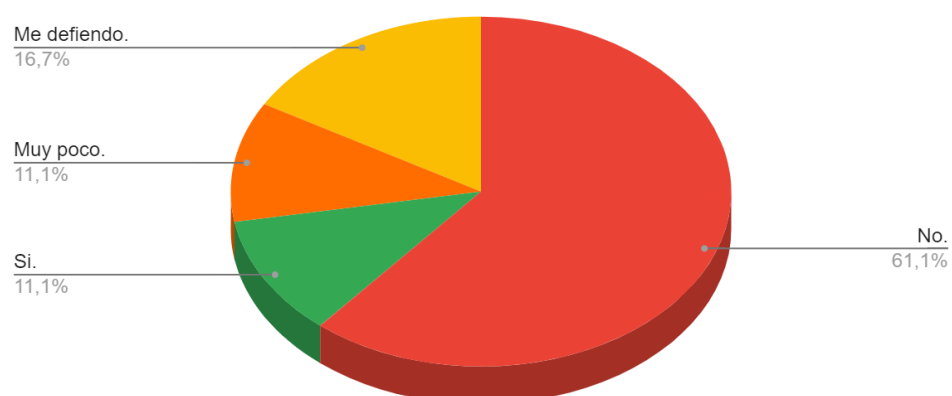
¿Cuántos años tienes?



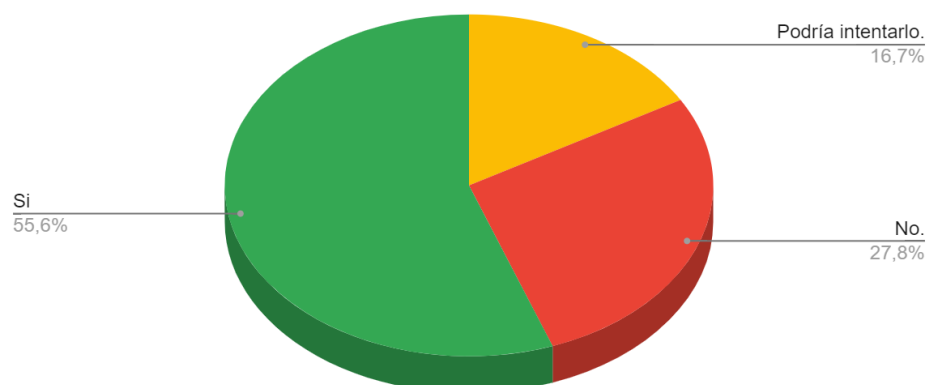
¿Tienes un ordenador en casa o en la de un familiar o amigo?



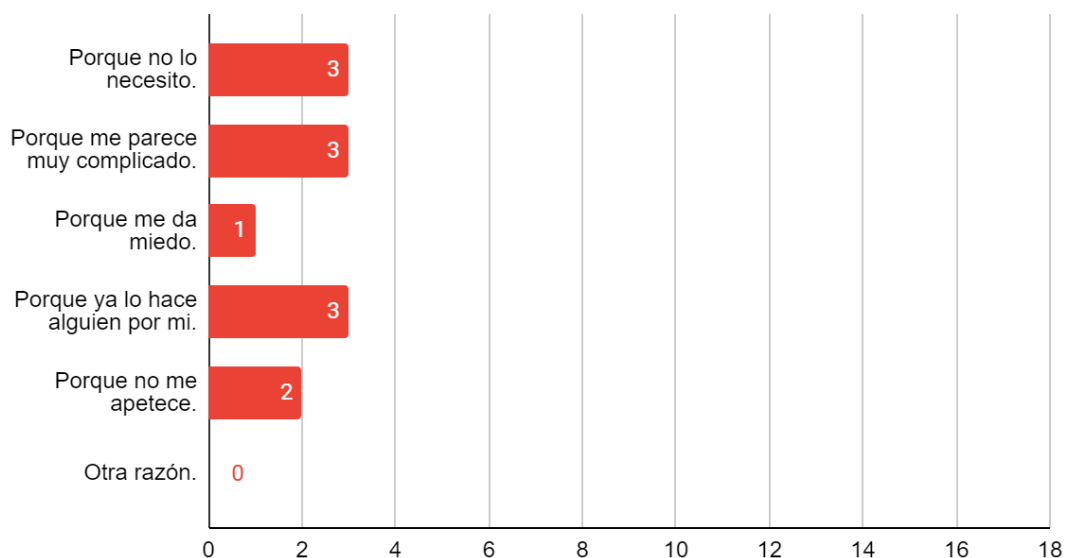
¿Tienes idea de cómo usar un ordenador?



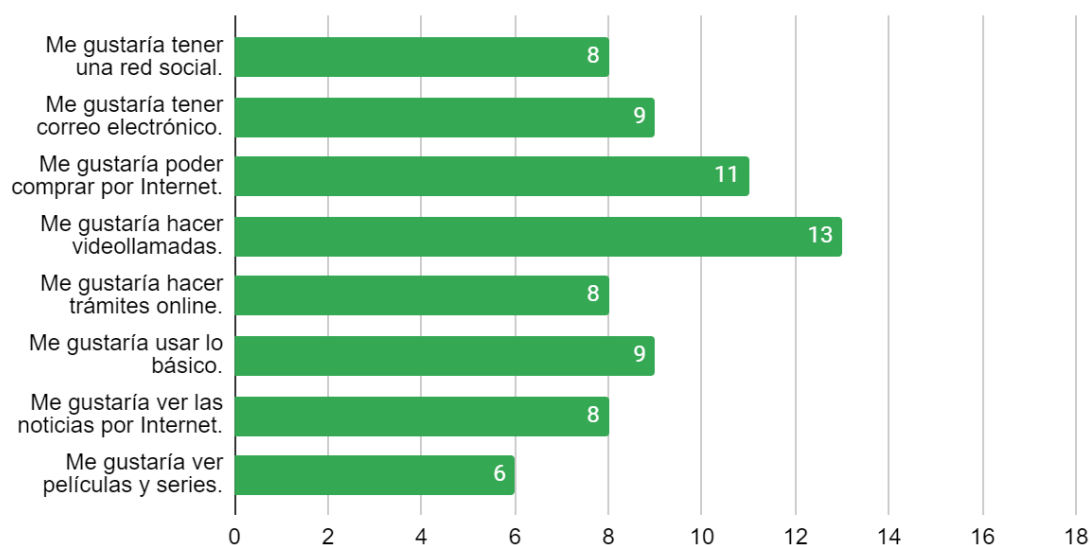
¿Te gustaría aprender lo básico?



Motivos por los que los usuarios no quieren aprender a usar un ordenador. De todos los encuestados, cuántos no quieren aprender...



¿Qué les gustaría aprender a los usuarios? De todos los encuestados, cuántos quieren...



Enlace con los resultados sin tratar:

<https://docs.google.com/spreadsheets/d/1CElhOBHHISKSBq8LnXbgpS7V1gVDT9l4dm1gf4G6CXo/edit?usp=sharing>

Si hay alguna otra razón por la cual te gustaría aprender a usar un ordenador, por favor coméntela brevemente a continuación:

5 respuestas

Ver mundo

Para estar al día en nuevas tecnologías

Usar lo de los mapas de internet

Arreglar problemas basicos del ordenador, configuraciones etc...

Por saber más cosas

Anexo 3

```
import React, {useCallback, useState} from "react";
import Global from "../Global";
import left from "../assets/images/left.png"
import right from "../assets/images/right.png"
import noleft from "../assets/images/noleft.png"
import noright from "../assets/images/noright.png"
import {useTranslation} from "react-i18next";
import Magnifier from "react-magnifier";
import zoomIMG from "../assets/images/zoom.png"
import nozoomIMG from "../assets/images/nozoom.png"

let lastIndex = 0

export function LessonStructure({numOfButtons, content, lesson, isInteractive, nInteractive}) {

  /*GLOBAL VAR*/
  const [t, i18n] = useTranslation("global")
  const [index, setIndex] = useState(0)
  const changeIndex = useCallback((v) => setIndex(v), [])
  const [zoom, setZoom] = useState(false)
  const actualLanguage = i18n.language
  let elem, lastElem;

  /*JSX IMAGES FRAGMENT*/
  const imgTEC = <div id="juego-teclado"> {Global[lesson]}</div>
  const imgESP = <img className="normal-diapo" src={content[index].imges} alt="img"/>
  const imgCAT = <img className="normal-diapo" src={content[index].imgcat} alt="img"/>
  const imgENG = <img className="normal-diapo" src={content[index].imgen} alt="img"/>
  const imgESPM = <Magnifier className="mag-diapo" src={content[index].imges} width="65%" zoomFactor={1.25}
    zoomImgSrc={content[index].imges} mgWidth={250} mgHeight={250} mgBorderWidth={7}
    mgShape="square"/>
  const imgCATM = <Magnifier className="mag-diapo" src={content[index].imgcat} width="65%" zoomFactor={1.25}
    zoomImgSrc={content[index].imges} mgWidth={250} mgHeight={250} mgShape="square"/>
  const imgENGM = <Magnifier className="mag-diapo" src={content[index].imgen} width="65%" zoomFactor={1.25}
    zoomImgSrc={content[index].imges} mgWidth={250} mgHeight={250} mgShape="square"/>

  /*FUNCTIONS*/
  function update(i) {
    if (i < numOfButtons && i >= 0) {
      lastElem = document.getElementById("button" + lastIndex);
      lastElem.classList.remove("ButtonFocus");
      lastIndex = i
      elem = document.getElementById("button" + i);
      changeIndex(i)
      elem.classList.add("ButtonFocus");
    }
  }
  function checkDiapoLanguage() {
    if(!zoom){
      if(actualLanguage === "es") return imgESP
      else if(actualLanguage === "en") return imgENG
      else return imgCAT
    }
    else {
      if(actualLanguage === "es") return imgESPM
      else if(actualLanguage === "en") return imgENGM
      else return imgCATM
    }
  }
}
```

```

return (
  <div className="all">
    <div className="lessonStructure">
      {(isInteractive && index === nInteractive - 1) ?
        (imgTEC) : checkDiapoLanguage()
      }
    <aside id="sidebar">
      {content.map((c, i) => (
        <div className="sidebar-item">
          {(isInteractive && i === nInteractive - 1) ?
            <button id={"button" + i} className="tryButton"
              onClick={(event) => update(i)}>{t(c.textoBoton)}</button> :
            (i !== index) ?
              <button id={"button" + i} className="lessonButton"
                onClick={(event) => update(i)}>{t(c.textoBoton)}</button> :
              <button id={"button" + i} className="lessonButton ButtonFocus"
                onClick={(event) => update(i)}>{t(c.textoBoton)}</button>
            }
          </div>
        )}}
    </aside>
    <div className="clearfix"/>
    <br/><br/>
    <div id="panel-navegacion">
      {(index > 0) ?
        <img src={left} alt="left" title="Anterior"
          onClick={(event) => update(index - 1)}> :
        <img src={noleft} alt="left" title="Anterior"
          onClick={(event) => update(index - 1)}>
      }
      {(zoom) ?
        <img src={zoomIMG} alt="auto-zoom" title="auto-zoom"
          onClick={(event) => setZoom(!zoom)}> :
        <img src={nozoomIMG} alt="auto-zoom" title="auto-zoom"
          onClick={(event) => setZoom(!zoom)}>
      }
      {(index < (numOfButtons - 1)) ?
        <img src={right} alt="right" title="Siguiente"
          onClick={(event) => update(index + 1)}> :
        <img src={nорight} alt="right" title="Siguiente"
          onClick={(event) => update(index + 1)}>
      }
    </div>
    <br/><br/>
  </div>
</div>
)
}

```