ESCUELA POLITÉNICA SUPERIOR DE MONDRAGON UNIBERTSITATEA
MONDRAGON UNIBERTSITATEKO GOI ESKOLA POLITEKNIKOA

Trabajo presentado para la obtención del título de
Titulua eskuratzeko lana

GRADO EN INGENIERÍA EN INFORMÁTICA
INFORMATIKAKO INGENIARITZA GRADUA

Título del Trabajo *Lanaren izenburua*

NUEVOS AVANCES EN LOS SERVICIOS DE CLASIFICACIÓN Y FILTRADO AUTOMÁTICO DE TEXTOS CON DESTINO AL SISTEMA TENTU-REVISTA ELECTRÓNICA PERSONALIZADA.

Autor *Egilea* ANDER BOLUMBURU CASADO

Curso *Ikasturtea* 2018/2019

Título del Trabajo *Lanaren izenburua*

NUEVOS AVANCES EN LOS SERVICIOS DE CLASIFICACIÓN Y FILTRADO AUTOMÁTICO DE TEXTOS CON DESTINO AL SISTEMA TENTU-REVISTA ELECTRÓNICA PERSONALIZADA.

Nombre y apellidos del autor
Egilearen izen-abizenak
BOLUMBURU CASADO, ANDER

Nombre y apellidos del/los director/es del trabajo
Zuzendariaren/zuzendarien izen-abizenak
AITOR OROBENGOA ORTUBAI
LARRINAGA, FELIX

Lugar donde se realiza el trabajo
Lana egin deneko lekua
ISEA S.COOP

Curso académico
Ikasturtea
2018/2019

RESUMEN

El principal objetivo del proyecto *'Nuevos avances en los servicios de clasificación y filtrado automático de textos con destino al sistema Tentu – Revista electrónica personalizada'* es el de participar en el desarrollo de unos servicios web que faciliten la clasificación temática y el filtrado de textos, además de la creación de una aplicación que gestione la contratación y promoción de estos. Dichos servicios, serán de utilidad para el producto existente Tentu, una revista electrónica personalizada que recoge información de una gran variedad de fuentes, cuya información debe de ser clasificada y/o filtrada. Durante esta etapa se realizará una importante toma de decisiones con el fin de proporcionar al cliente final un producto que cumpla con diferentes estándares tanto de calidad, como de seguridad y que aportarán valor al mismo.

Castellano

'Urrats berriak Tentu-Aldizkari Elektroniko Pertsonalizatuak erabiliko dituen testu sailkapen eta iragazketa automatiko alorrean.' proiektuaren xede nagusia sailkapen tematikoan aurrerapenak egitea da, horretarako hori errazten duen web zerbitzu batzuen garapenean parte hartu da. Horretaz gain, zerbitzu hauen kudeaketa errazteko web aplikazio bat egingo da. Tentu aldizkari elektronikoa bere bezero nagusia izango da, horrela, bertan ikus daitekeen eduki guztia sailkatuta edo eta iragaztuta egongo da. Garapenean zehar, erabaki garrantzitsuak hartzeaz gain, funtzionalitate, test eta dokumentazio berrien garapena ere burutuko da, horrela aplikazioaren kalitatea handituz.

Euskara

The main objective of the project 'Further advances in the field of Automatic Classification and Filtering of texts to be used in the context of Tentu-Personal Digital Magazine.' is to participate in the development of web services which facilitate the classification and filtering of texts, in addition to the creation of an application that manages the recruitment and promotion of the same. These services will be useful for the existing product Tentu, a personalized electronic magazine which collects information from a variety of sources, from which the information must be classified and / or filtered. During this stage important decisions will be made in order to provide the end customer a product that meets different standards of quality and safety, which will add value to it.

English

TABLA DE CONTENIDO

1 Introducción 1

1.1 Contexto 1

1.2 Antecedentes 2

1.2.1 ISEA S. COOP. 2

1.3 Problemática 2

1.3.1 Problema a resolver por el producto 2

1.3.2 Producto 3

1.3.3 Estado inicial 8

1.3.4 Retos 8

1.4 Objetivos 9

1.4.1 Análisis y selección de software 9

1.4.2 Desarrollo de la aplicación web 9

1.4.3 Desarrollo de los servicios 9

1.4.4 Garantizar la estabilidad de la aplicación 9

1.4.5 Documentación 10

1.5 Fases del proyecto 10

1.5.1 Fase inicial 11

1.5.2 Fase de preparación del desarrollo 12

1.5.3 Fase de desarrollo del producto 12

1.5.4 Fase de resolución de incidencias 15

1.6 Pliego de condiciones 15

1.6.1 Pliego de condiciones técnicas de desarrollo 15

1.6.2 Pliego de prescripciones técnicas de ejecución 16

2 Desarrollo 17

2.1 Toma de decisiones 17

2.1.1 Posibilidades de desarrollo del *front-end* 17

2.1.2 Alojamiento de la aplicación 27

2.1.3 Gestión del código 31

2.2 Descripción de las tareas realizadas 32

2.2.1 Aprendizaje 32

2.2.2 Diseño de la aplicación 32

2.2.3 Desarrollo de la aplicación 33

2.2.4 Desarrollo de servicios web 46

2.2.5 Presentación Ministerio 57

2.3 Desarrollos adicionales 62

2.3.1 Lanzamiento de Tentu al público y necesidad de rediseño 62

2.3.2 Recopilación de artículos para red neuronal 66

3 Resultados 68

4 Memoria económica del proyecto 70

4.1 Coste instrumental y equipamiento 70

4.2 Coste de personal....70
4.3 Coste total....70
5 Conclusiones y líneas futuras....71
5.1 Conclusiones técnicas....71
5.2 Conclusiones metodológicas....72
5.3 Conclusiones personales....72
5.4 Nuevo producto: Conjunto de servicios + Aplicación web para gestionarlos....73
5.5 Líneas futuras....73
5.6 Posible desarrollo de nueva empresa....74
6 Valoración personal....75
7 Anexos....76
8 Bibliografía....77

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Actores principales de GAIA 4
Ilustración 2: Gantt del proyecto. 11
Ilustración 3: Prototipo de la interfaz principal de GAIA 12
Ilustración 5: Árbol DOM de un documento HTML 20
Ilustración 6: Comparación de la actualización del DOM corriente frente al virtual de React 21
Ilustración 7: Comparación entre el enlace de datos bidireccional y unidireccional 21
Ilustración 8: Prueba de velocidad (ms) 22
Ilustración 9: Interfaz mediante la cual Google proporciona la lista de cambios 24
Ilustración 10: Encuesta de desarrolladores de Stack Overflow 25
Ilustración 11: Tendencia de React frente a Angular en territorio nacional en el último año según Google Trends 26
Ilustración 12: Tendencia de React frente a Angular en territorio internacional en el último año según Google Trends 26
Ilustración 13: Diagrama organizativo de Tentu 29
Ilustración 14: Diagrama organizativo de aplicación 29
Ilustración 15: Prototipo de interfaz principal de GAIA 33
Ilustración 18: Unidades de Atomic Design 34
Ilustración 19: Ejemplos de átomos 34
Ilustración 20: Ejemplo de molécula 35
Ilustración 21: Ejemplo de organismo 35
Ilustración 22: Ejemplo de plantilla 35
Ilustración 23: Ejemplo de página 36
Ilustración 24: Página inicial de GAIA dividida en diferentes componentes 36
Ilustración 25: Gestión de datos Firestore 38
Ilustración 26: Simulador de reglas de Firestore 39
Ilustración 27: Interfaz de Algolia con el índice de palabras malsonantes 40
Ilustración 16: Principales herramientas que proporciona Firebase 41
Ilustración 17: Tutoriales disponibles en la documentación de Firebase 42
Ilustración 28: Diagrama explicativo de los servicios ofrecidos por Heroku 43
Ilustración 29: Interfaz para la modificación de precios de GAIA 44
Ilustración 30: Interfaz para la consulta de precios de GAIA 45
Ilustración 31: Interfaz para la gestión de categorías 45
Ilustración 32: Interfaz para subir un archivo .csv con los temas 46
Ilustración 33: Ejemplo de comunicación entre UZEI e ISEA 48
Ilustración 34: Ejemplo gráfico con el fin de explicar la distancia Levenshtein 49
Ilustración 35: Ejemplos de uso de la distancia Levenshtein en diferentes en diferentes aplicaciones 49

Ilustración 36: Interfaz mediante la cual el usuario puede modificar su perfil 50
Ilustración 37: Interfaz mediante la cual el usuario puede limitar el consumo de sus servicios 51
Ilustración 38: Arquitectura final de GAIA (Clasificación) 55
Ilustración 39: Arquitectura final de GAIA (Filtrado) 56
Ilustración 40: Esquema presentado en la presentación ante el Ministerio de Industria 58
Ilustración 41: Interfaz principal de la aplicación web 59
Ilustración 42: Interfaz de elección de contratación 60
Ilustración 43: Interfaz para la gestión de datos del usuario 60
Ilustración 44: Interfaz para la gestión del contrato de clasificación 60
Ilustración 45: Interfaz para limitar el número de consultas 61
Ilustración 46: Interfaz para la consulta de precios 61
Ilustración 47: Interfaz para modificar los precios de los servicios 61
Ilustración 48: Interfaz con tabla interactiva para la consulta de categorías 61
Ilustración 49: Interfaz para la gestión de categorías 62
Ilustración 50: Comparativa entre about nuevo frente al antiguo 64
Ilustración 51: Comparativa entre login/registro nuevo frente al antiguo 65
Ilustración 52: Comparativa de la interfaz del perfil nuevo frente al antiguo 65
Ilustración 53: Comparativa de la interfaz principal nueva frente a la antigua 65
Ilustración 54: Arquitectura de GAIA (Completa) 68

1 INTRODUCCIÓN

Este documento tiene como objetivo describir el proyecto realizado durante el Trabajo de Fin de Grado "*Nuevos avances en los servicios de clasificación y filtrado automático de textos con destino al sistema Tentu – Revista electrónica personalizada"* en ISEA S. COOP. En el que se explicarán los siguientes apartados: Contexto del proyecto, sus objetivos, fases, desarrollo y finalmente los resultados obtenidos. Además, se incluye un pliego de condiciones en el que se describirán los elementos necesarios para la puesta en marcha del proyecto.

1.1 CONTEXTO

El desarrollo de este proyecto se ha llevado a cabo en ISEA S.COOP, Centro de Innovación y Emprendimiento de carácter privado y sin ánimo de lucro. Especializado en el sector de los servicios empresariales, concretamente, en su departamento de desarrollo, junto con el equipo de desarrolladores.

La empresa ya mencionada ISEA. S. COOP, mediante el denominado proyecto "*Nuevos avances en los servicios de clasificación y filtrado automático de textos con destino al sistema Tentu – Revista electrónica personalizada",* ha tratado de enfocar su rol en el desarrollo de unos servicios web que faciliten esa labor, además de una aplicación que gestione la promoción y contratación de dichos servicios. Cabe destacar que estos, serán de utilidad para un producto propio de la empresa, Tentu (ISEA S. Coop., 2018), una revista electrónica personalizada mediante la cual el cliente puede obtener información de su interés.

Para llevar a cabo dicho proyecto, ISEA cuenta con un equipo de ingenieros especializados en el desarrollo de tareas que conciernen al mismo, que se realizará en el departamento de desarrollo.

En cuanto al código, este es gestionado mediante un sistema de control de versiones. El objetivo es implementar diferentes testeos, que se realicen automáticamente en cada actualización de código fuente, de modo que se compruebe la robustez del avance que aporta dicha actualización de código.

Otro de los objetivos es mantener informado al equipo de desarrollo en caso de error, ya que, si algún tipo de testeo fallase al ser ejecutado, lo ideal sería, que mandase una notificación de dicho error a las cuentas de correo electrónico de todo el equipo, de modo que se cumpliría el objetivo de mantener a todos informados a cerca de la calidad del código.

Con el fin de llevar una adecuada gestión de tareas, se están utilizando dos herramientas. Por un lado, se ha realizado al comienzo del proyecto un diagrama de Gantt general con las tareas globales, para realizarlo se ha empleado la aplicación web *Team Gantt*[1] (Gantt, 2019). Por otro lado, la gestión que conciernen a cada tarea global se ha realizado mediante un sistema de tablas que contiene la herramienta de control de versiones.

[1] Termino que se refiere a la aplicación web diseñada para mejorar la comunicación y colaboración dentro de un equipo.

Después de realizar esta breve contextualización de la realización del proyecto, las siguientes líneas recogerán la descripción de la empresa, incluyendo tanto sus productos y como sus servicios.

1.2 ANTECEDENTES

Teniendo en cuenta la breve contextualización realizada anteriormente, a lo largo de este apartado se introducirán los elementos básicos los cuales son clave para el inicio del proyecto.

1.2.1 ISEA S. COOP.

Como se ha mencionado en el apartado anterior, Innovación en Servicios Empresariales Avanzados – ISEA S.COOP. es un centro de innovación y emprendimiento de carácter privado y sin ánimo de lucro especializado en el sector de los servicios empresariales, promovido por la división de ingeniería y servicios empresariales de la Corporación Mondragón.

ISEA S. COOP. es un agente científico tecnológico integrado en la red vasca de ciencia, tecnología e innovación. En la actualidad forma parte de la Agencia Vasca de Innovación. Adicionalmente, ISEA es un agente homologado del Servicio Vasco de Emprendimiento de SPRI (Sociedad para la Transformación Competitiva), dependiente del Gobierno Vasco.

Se trata de una entidad declarada de utilidad pública por la Consejería de Justicia, Empleo y Seguridad Social del Gobierno Vasco.

En cuanto a la misión de ISEA S. COOP. se refiere, esta radica en mejorar la competitividad del sector de los servicios empresariales mediante la potenciación del desarrollo tecnológico, la innovación y el emprendimiento de nuevas actividades empresariales.

Una vez introducida tanto la empresa, como su misión, en el siguiente apartado se procederá a explicar tanto la necesidad del desarrollo del producto, como el modo en el que se abastecerá esa necesidad.

1.3 PROBLEMÁTICA

En el siguiente apartado de procederá a explicar el problema general que se va a tratar de solucionar mediante el desarrollo del producto.

1.3.1 PROBLEMA A RESOLVER POR EL PRODUCTO

Más del 90% de la información digital disponible es información no estructurada en forma de textos y documentos. En la economía del conocimiento, la reutilización de la información del sector público y privado presenta un considerable potencial económico pues las empresas infomediarias (535 empresas en España) analizan y tratan información del sector público y/o privado para crear productos de valor añadido (1.550 M de € de facturación) destinados a terceras empresas o a la ciudadanía.

El proceso de reaprovechamiento de tal información por las empresas del Sector Infomediario (Generadores de Información, Agregadores de Información, Capacitadores,

Enriquecedores) precisa del desarrollo de una serie de hitos: analizar el formato, preparar y ordenar la información a través de la catalogación y la categorización.

Esta problemática es singularmente relevante en el aprovechamiento de la información abierta, pues es preciso resaltar que existe un importante volumen de información Open Data[2] que no consigue emerger hacia la sociedad pues su contenido es desconocido para los ciudadanos y las empresas.

Por otro lado, el régimen de actuación legal de la prensa on-line, las redes sociales o cualquier otro recurso digital que ofrece puerta abierta a la participación de las personas está determinado por el corpus europeo y español en materia de telecomunicaciones, comercio electrónico, Sociedad de la Información y eAdministración.

Para que la participación en dichos medios sea acorde con el conjunto de dicho corpus legal, se debe garantizar el filtrado de contenidos para que los mismos no incorporen elementos atentatorios a la imagen y privacidad de las personas o incorporen actividades ilegales (promoción de la prostitución, venta de drogas, apologías diversas, etc.). La cumplimentación manual de las labores de filtrado será poco factible y, en todo caso, costosa, por lo que se estima necesaria la incorporación de tecnologías de filtrado de contenidos basados en procesamiento de lenguajes naturales y recursos lexicográficos especializados que posibiliten la ejecución de labores de filtrado automático o supervisado.

Además, cabe destacar que en España no existe ningún servicio on-line de clasificación automática de textos y filtrado de contenidos.

En el mundo, particularmente en Estados Unidos sí que existen servicios on-line de clasificación automática de textos y filtrado de contenidos. Algunos de ellos, especialmente aquellos de origen americano, tales como (*AlchemyAPI, Ambiverse Natural Language Understanding AP, BitextAPI, CogitoAPI, Diffbot, Magnet by Klangoo, MeaningCloud, TextRazor, etc.*) ofrecen la posibilidad de clasificar textos en castellano, pero el fondo lexicográfico que los soporta y, por tanto, la calidad ofrecida, no es comparable al que cuenta el participante en el proyecto UZEI que tras su trayectoria de 38 años, le han conducido a la creación de 60 diccionarios especializados.

Existe un enlace con más información acerca de UZEI en el anexo.

Una vez presentado el problema, en el siguiente apartado se explicará el producto con el que se intentará abastecer esa necesidad.

1.3.2 PRODUCTO

El producto en el que se ha pensado con el fin de poder abastecer esa necesidad es GAIA, el cual consiste en unos servicios web soportados en tecnologías de clasificación y filtrado automática de textos proporcionados por UZEI (Centro Vasco de Terminología y Lexicografía) los

[2] Término que se refiere a la filosofía y practica que persigue que determinados tipos de datos estén disponibles de forma libre para todo el mundo.

cuales clasificaran y/o filtraran los textos recibidos por la persona usuaria. Dichos servicios serán ofertados tanto en euskera como en castellano.

Además, ofrece la clasificación del texto en diferentes estándares como Eurovoc, CDU y NewsCodes, la decisión de la elección de estos estándares será explicada más adelante en el apartado 1.3.2.2.

En la siguiente ilustración, se pueden visualizar los actores participantes en el módulo de interoperabilidad.

Ilustración 1: Actores principales de GAIA

A modo de resumen, se podría decir que el cliente manda al servicio GAIA mediante los servicios desarrollados, el texto que desea clasificar o filtrar, este se comunica con UZEI, del cual espera una respuesta con el texto clasificado y/o filtrado, y finalmente hace eco de esa respuesta al cliente.

Cabe destacar que GAIA con el fin de ofrecer un valor añadido, posee una aplicación web mediante la cual se puede realizar o modificar la contratación de los servicios, además de obtener diferentes instrucciones a cerca de su uso. Del mismo modo, esta aplicación web también sirve como herramienta de gestión para el administrador, con el que puede modificar diferentes parámetros como por ejemplo el precio anual del servicio o incluso la base de datos mediante la cual poder actualizar las referencias de los estándares Eurovoc o NewsCodes, entre otros.

Tras presentar el producto en cuestión, en las siguientes líneas se analizará el objetivo geográfico y temático que se pretende alcanzar mediante el producto.

1.3.2.1 ALCANCE GEOGRÁFICO Y TEMÁTICO

En cuanto a su alcance, debido a que GAIA soporta hoy en día el euskera y castellano se podría decir que su alcance geográfico se limita a territorio nacional. En caso de que en un futuro se aplicase el soporte en inglés, se podría expandir a otros países.

En cuanto al alcance temático, GAIA cuenta con tecnologías de clasificación automática de textos, también denominada categorización de textos o *topic spotting*, que opera sobre la base de un motor lematizador[3] inteligente y taxonomías[4] temáticas desarrolladas por UZEI-Centro Vasco de Lexicografía. De esta manera, es capaz de asignar automáticamente un destino

[3] Término que se refiere la lematización, un proceso lingüístico que consiste en, dada una forma flexionada, hallar el lema correspondiente, es decir, el núcleo de esa palabra.
[4] Término que se refiere en su sentido más general, a la ciencia de la clasificación.

temático especifico a los contenidos recibidos por parte del cliente según 31 temáticas, las cuales se irán ampliando con el paso del tiempo llegando a un máximo de 151.

Una vez explicado el alcance tanto geográfico como temático, se analizará la estructuración que se ha decidido llevar a cabo para las temáticas tratadas por GAIA.

1.3.2.2 ESTRUCTURACIÓN TEMÁTICA DE GAIA

Con el fin de tener una estructuración temática correcta, se ha procedido a analizar las expectativas de los potenciales clientes, expertos, comerciales y técnicos del Consorcio que han planteado la necesidad de llevar a cabo el desarrollo del servicio GAIA. Estas expectativas han concluido con el desarrollo de una primera lista de temáticas en función de las necesidades de estos.

Tras realizar dicho listado, se han analizado un conjunto de Iniciativas Europeas en materia de normalización y lexicografía; entre los que destacan:

- La Clasificación Decimal Universal o CDU.
- SKOS (siglas de *Simple Knowledge Organization System*) es una iniciativa del W3C en forma de aplicación de RDF.
- Eurovoc, tesauro multilingüe y multidisciplinario que abarca la terminología de los ámbitos de actividad de la UE.
- Los resultados de las soluciones de las iniciativas de la Unión Europea en materia de interoperabilidad para las administraciones públicas europeas (ISA).
- El sistema de clasificación de la UNESCO.
- El sistema de clasificación de NEWSCODES de IPTC: IPTC es un consorcio internacional constituido por las agencias de prensa y las empresas de comunicación más importantes del mundo. Entre ellas, se pueden referir:
    - *Alliance Européenne des Agences de Presse.*
    - ANPA (ahora NAA).
    - FIEJ (ahora WAN).
    - *North American News Agencies* (*Associated Press, Canadian Press and United Press International*).

Una vez analizadas las diferentes opciones se ha llegado a las siguientes conclusiones:

- Selección de los estándares más oportunos para el proyecto:
    - El sistema de clasificación de NEWSCODES de IPTC.
    - El sistema de clasificación Eurovoc, tesauro multilingüe y multidisciplinario que abarca la terminología de los ámbitos de actividad de la UE.
    - La Clasificación Decimal Universal o CDU.
- Se ha realizado una lista de 215 temáticas seleccionadas y se ofrecen sus correspondencias en los diversos estándares seleccionados, la cual está adjunta en el anexo.
- Finalmente, se ha realizado un listado de temáticas prioritarias para el desarrollo del proyecto GAIA, el cual está en el anexo.

Tras dejar fijado la estructuración temática a seguir, es importante analizar los orígenes potenciales de los contenidos a clasificar y/o filtrar.

1.3.2.3 ORÍGENES POTENCIALES DE LOS CONTENIDOS A CLASIFICAR/FILTRAR

Como ha sido mencionado, se prevé que gran parte de los contenidos a clasificar/filtrar por los servicios provengan de empresas infomediarias, los cuales utilizarán el contenido categorizado para dar un valor añadido a los servicios.

Estas organizaciones pueden utilizar datos de los siguientes orígenes:

- Redes RSS[5].
- Contribuciones de Instituciones y Empresas.
- Contribuciones de Asociaciones, ONG-s y demás entes vinculados al voluntariado.
- Contribuciones ciudadanas.

De la misma manera, es preciso enunciar que existe un importante volumen de información que no consigue emerger hacia la sociedad o que encuentra dificultades para ello.

Un buen ejemplo de cliente potencial es Tentu – La revista electrónica personalizada, la cual capta a través de más 700 fuentes RSS diferentes contenidos para tematizarlos y posteriormente destinarlos a colectivos interesados. Por tanto, podría considerarse un mecanismo de emparejamiento de la información aportada por un "colectivo interesado en la creación de la información" y entregada a un "colectivo interesado en su recepción".

Existe más información acerca de Tentu, en el anexo.

1.3.2.4 FACILIDAD DE USO Y ACCESIBILIDAD DE GAIA

El servicio GAIA, ha requerido para el comienzo de su uso la especificación, desarrollo e implantación de un conjunto de componentes de software, entre los cuales cabe destacar la aplicación web para contratación, publicidad y administración de los servicios.

En lo que a la aplicación web se refiere, ofrece un servicio adaptativo "Web Responsive Design", soportado en HTML5, CSS3 y JavaScript, que a su vez permiten adaptar los contenidos con destino a diversas dimensiones y tipologías de interfaces de usuario de lectura/manipulación de los contenidos en un entorno multidispositivo tales como ordenadores personales, tabletas electrónicas, teléfonos móviles… Aunque lo más habitual será que se acceda desde un ordenador personal, no está de más dar la posibilidad de acceder desde cualquier dispositivo.

Otro aspecto por destacar, son los servicios web los cuales están documentados mediante *Swagger*[6] (Swagger.io, 2019). Esta aplicación permite al cliente, en este caso un desarrollador, conocer diferentes aspectos de la API como podrían ser:

- Parámetros que debe enviar para recibir la respuesta correspondiente.
- Parámetros que recibirá el cliente en la respuesta.

[5] Término que se refiere a un formato específico para la distribución de contenidos de la web.
[6] Término que se refiere a una herramienta con la que poder documentar los servicios web.

- Proporciona al desarrollador un pequeño *sandbox*[7] en el que podrá realizar pruebas.
- Códigos de error para conocer en todo momento los que se puedan generar.

Dicha documentación está disponible a través siguiente enlace.

Con el fin de comprender la relevancia de una buena documentación se recomienda la lectura de este artículo (Chackray, 2017).

Como se ha pensado en el desarrollador, también se ha publicado una librería NPM[8] (NPM, 2019) de modo que, para hacer uso de los servicios, el usuario solo tendrá que contratarlos e introducir la API Key[9] a la hora de importar la librería mediante la herramienta de gestión de dependencias.

Una vez hecho esto, será posible realizar las consultas y recibir las respuestas en apenas una línea de código. Esto facilita mucho las cosas, ya que el desarrollador no tiene por qué preocuparse de implementar otra librería externa para realizar las peticiones al servidor. Además de reducir drásticamente la cantidad de líneas de código de la aplicación, algo a agradecer tanto por parte del desarrollador actual, como el que pueda incorporarse en un futuro ya que esto facilitará de forma considerable su comprensión.

El enlace hacia la dependencia publicada en NPM está disponible tanto aquí, como en el anexo.

1.3.2.5 INNOVACIÓN

En cuanto a la innovación, que en cualquier caso es importante, resulta conveniente señalizar los principales aspectos a través de los cuales el presente proyecto refleja innovación:

En primer lugar, GAIA ofrece una vía novedosa de aprovechamiento masivo de información pública y privada. Lo cual como se ha comentado anteriormente beneficia a las compañías infomediarias, las cuales, mediante la herramienta ofrecerán datos más precisos a sus clientes.

En segundo lugar, cabe destacar que, el proyecto, se alinea con las indicaciones de la unión europea en su comunicación 2013/37/EU referente a las *"oportunidades de creación de valor en el open data están en el desarrollo de aplicaciones que combinen datos públicos con datos privados y sean empaquetados para satisfacer necesidades específicas de segmentos de clientes".*

Finalmente, otro de los elementos que aportan un carácter innovador al proyecto, es la implementación de las tecnologías de clasificación automática de textos, también denominada *topic spotting*. Esta tecnología, permite la clasificación temática automática de textos tanto en castellano como en euskera. Como se ha indicado anteriormente y bien dice el Ministerio de

[7] Termino que se refiere a un mecanismo para ejecutar programas con seguridad y de manera separada.
[8] Término para referirse a una herramienta de gestión de dependencias.

[9] Una API Key es un identificador que sirve como medio de autenticación de un usuario para el uso de los servicios proporcionados por GAIA.

Industria en su estudio *"Informe sobre el estado de las tecnologías del lenguaje dentro de la Agenda Digital para España"*, más del 90% de la información digital disponible es información no estructurada en forma de textos y documentos en múltiples lenguas.

Tras este análisis a cerca de la innovación que aporta la realización del proyecto, se procederá a explicar el estado actual en el que se encuentra el producto.

1.3.3 ESTADO INICIAL

A la hora de comenzar el desarrollo de este proyecto, no existe ningún tipo de base sobre él. Lo único existente acerca de él es el servicio de clasificación temática realizado por UZEI. Por lo que una parte importante de las labores a llevar a cabo es meditar sobre qué se va a asentar el proyecto y por tanto el producto.

Una de las decisiones más importantes consiste en la selección de la plataforma donde se desarrollarán tanto las aplicaciones como los servicios. Estos deben ser escalables ya que no se sabe la cantidad de usuarios que va a llegar a albergar. Además, la selección de una plataforma adecuada puede facilitar el desarrollo del proyecto, ya que existen algunas que proporcionan facilidades a los desarrolladores.

Otro punto para tener en cuenta es que la plataforma seleccionada sea económicamente viable para la empresa y posibilite la explotación beneficiosa de la actividad empresarial.

Si además de esto, el hecho de seleccionar una herramienta y no otra puede aportar algún tipo de valor al cliente o al desarrollador, será un gran punto a favor de dicha herramienta.

Una vez analizado el estado actual del producto, es hora de analizar los retos que supone el hecho de realizarlo.

1.3.4 RETOS

Tras la analizar los retos que supone realizar el proyecto, se procederá a definir los retos que propone el desarrollo.

Atendiendo a lo establecido tanto en la introducción como en la contextualización de este informe, el principal reto que presenta el proyecto consiste en la realización de la aplicación web la cual gestione la contratación y diferentes parámetros de los servicios, además del desarrollo de los propios servicios en sí.

Dado que el proyecto partía de cero, un primer reto ha consistido en realizar un exhaustivo análisis de las tecnologías y software disponibles y, a su vez, alineados con las necesidades y objetivos empresariales. Además, en base a ese análisis se tomarán unas decisiones que afectarán al transcurso del proyecto. Dicho análisis y toma de decisiones se puede observar en el apartado 2.1 de este mismo documento.

Al mismo tiempo, el hecho de aprender a desarrollar el producto mediante la herramienta seleccionada también supone un reto en sí mismo. Es por ello por lo que una parte razonable del proyecto de dedicará exclusivamente a la formación en torno a la herramienta, ya que se considera vital para el desarrollo el dominio sobre la misma.

Cabe destacar que otro de los retos a cumplir es la correcta integración de la aplicación y los servicios con el fin de asegurar el adecuado funcionamiento. En otras palabras, se debe lograr la correcta comunicación entre ambos elementos de manera a posibilitar una explotación racional del conjunto.

Otro de los mayores retos a tener en cuenta es la adaptación del *front-end*[10]. Conviene destacar que todas las interfaces de la aplicación deben mantener una coherencia estética y funcional, con el fin de lograr una experiencia intuitiva y agradable.

Por último, otro reto sustancial corresponde al gran número de funcionalidades a desarrollar, tanto de servicios internos como externos. Destacando entre ellos los servicios de clasificación de texto de UZEI; una de las piedras angulares del proyecto.

1.4 OBJETIVOS

Tras plantear los retos que presenta GAIA, en el siguiente apartado se definirán los objetivos que se pretenden lograr a través del proyecto.

1.4.1 ANÁLISIS Y SELECCIÓN DE SOFTWARE

En primer lugar, y como objetivo principal, resulta imprescindible el análisis y selección del software de la aplicación y los servicios web. Ante todo, es destacable que actualmente no existe ningún avance de este tipo, lo cual implica la necesidad de analizar las mejores opciones de desarrollo. Para ello se llevará a cabo un detallado proceso de selección, para así garantizar un funcionamiento correcto de la aplicación.

1.4.2 DESARROLLO DE LA APLICACIÓN WEB

En segundo lugar, y remitiéndonos a los retos anteriormente mencionados, una parte del proyecto se basa en el desarrollo de una aplicación web. No obstante, resulta necesario considerar que dicha aplicación debe responder a un estilo común, por lo que resulta obvia la necesidad de desarrollar al mismo tiempo un análisis para determinar la mejor interfaz de usuario, en cuanto a funcionalidad y diseño se refiere. Dicho análisis se puede encontrar en el apartado 2.2.2 de este mismo documento.

1.4.3 DESARROLLO DE LOS SERVICIOS

En tercer lugar, otro de los grandes objetivos del presente proyecto, se refiere al desarrollo de los servicios. Al igual que con la selección del software de desarrollo, es importante llevar a cabo un análisis de mercado con el fin de lograr el mejor producto posible. Además, también se analizará que clase de documentación ofrecen los servicios de hoy en día con el fin de que resulten cómodos e intuitivos de cara al usuario. La descripción sobre el desarrollo se puede encontrar en el apartado 2.2.4 de este mismo documento.

1.4.4 GARANTIZAR LA ESTABILIDAD DE LA APLICACIÓN

[10] Término que se refiere a la interfaz gráfica o capa de presentación de una aplicación.

Desde un punto de vista objetivo, es conveniente admitir que la existencia de controles de calidad cada vez que se actualiza el código fuente, garantiza la estabilidad y robustez de la aplicación. Es por ello por lo que el cuarto objetivo consiste en realizar los testeos de cada funcionalidad desarrollada, de modo que se garantice dicha robustez.

1.4.5 DOCUMENTACIÓN

Es muy importante realizar un buen comentario en el código, ya que el futuro desarrollador que se incorpore al proyecto debe de ser capaz de entender el código para llevar a cabo sus labores. Es por ello por lo que otro de los objetivos consiste en realizar una buena documentación del código.

Además de la documentación del código, resulta conveniente documentar cada punto de entrada ofrecido mediante los servicios, de modo que el usuario sepa en todo momento los parámetros de entrada y salida de cada target[11].

1.5 FASES DEL PROYECTO

Una vez presentados los principales objetivos que tratan de lograrse a través del proyecto, posteriormente se procederá a concretar las principales fases a través de los cuales debe desenvolverse el mismo.

El siguiente diagrama está incluido en el anexo, por si se quisiera apreciar con mayor detalle:

[11] Término utilizado para referirse a cada una de las rutas disponibles en los servicios web.

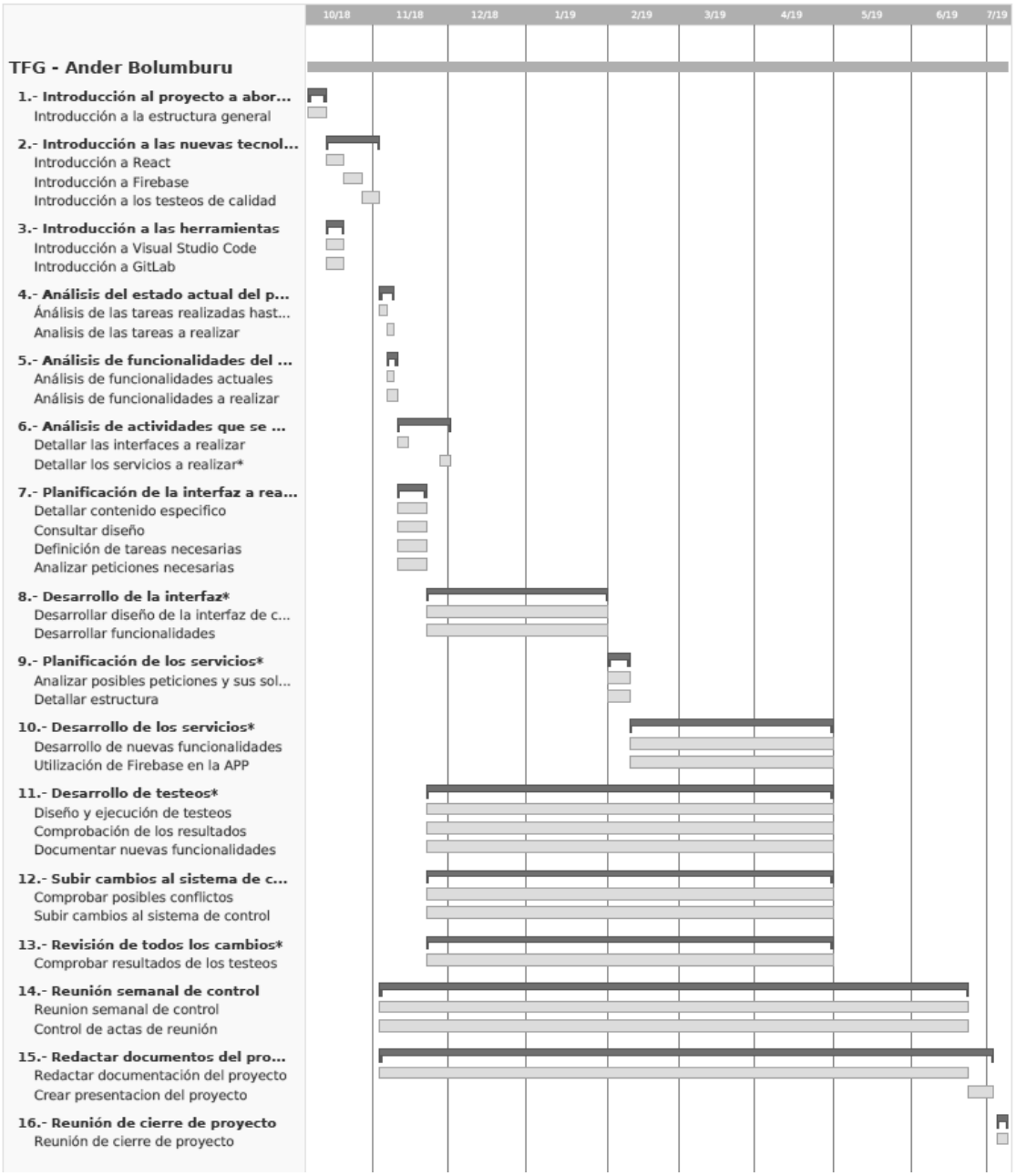

Ilustración 2: Gantt del proyecto.

1.5.1 FASE INICIAL

La fase inicial del proyecto consiste en seleccionar las principales herramientas de desarrollo un tanto desconocidas hasta el momento y aprender su utilización. Por tanto, resulta necesaria la obtención de información referida a los procesos de desarrollo y calidad del código, que sigue la propia empresa. Para ello se realizará una investigación con el fin de comprobar cuáles son las herramientas que más se adecuan a las necesidades del proyecto. Dicha investigación y toma de decisiones puede verse en el apartado 2.1 de este mismo documento.

1.5.1.1 APRENDIZAJE

Tras un análisis de las herramientas disponibles actualmente en el mercado, es conveniente centrarse en la fase de aprendizaje del software seleccionado.

Durante este tramo, se llevará a cabo un proceso de aprendizaje con el fin de que el producto tenga la mayor calidad posible.

1.5.2 FASE DE PREPARACIÓN DEL DESARROLLO

El objetivo de esta fase es la de concretar el planteamiento del primer prototipo de la aplicación web. En dicha fase, tanto el equipo de desarrollo como el equipo de diseño tratan de llegar a un acuerdo en cuanto a las funcionalidades que las diferentes pantallas deben contener.

Dicho prototipo se ha realizado en Adobe XD (Adobe, 2019), dado que posibilita hacer una simulación de la interacción que tendrá el producto final, siendo muy útil tanto para el desarrollador como para el responsable principal.

Ilustración 3: Prototipo de la interfaz principal de GAIA

El prototipo está disponible en la carpeta anexos adjunto a este mismo documento en forma tanto de imágenes, como video o el propio fichero XD en sí. Por otra parte, en caso de que no se disponga de los medios para abrir este tipo de ficheros, se ha publicado un enlace con la herramienta de la propia Adobe. Mediante él, la interacción pueda ser consultada haciendo uso del navegador web.

1.5.3 FASE DE DESARROLLO DEL PRODUCTO

Tras la fase de diseño, el proyecto se adentra en la fase de desarrollo, la cual, tal y como su nombre indica, se basa en la realización tanto de la aplicación como de los servicios web. Una vez seleccionada y estudiada la tecnología a aplicar, se procederá a la producción de ambas.

Cabe destacar que todas las tareas contarán con un seguimiento por parte del responsable, con el fin de garantizar el resultado esperado. Para ello, se realizarán reuniones semanales con el grupo de desarrolladores y el responsable principal.

Además, se debe tener en cuenta que durante el desarrollo se mantendrá contacto con entidades externas como podría ser el caso de UZEI, quien proporciona los servicios de clasificación de textos.

Durante esta fase, tienen importancia las siguientes herramientas:

- Software seleccionado mediante el análisis para el desarrollo de la aplicación web.
- Software seleccionado mediante el análisis para el desarrollo de los servicios web.
- El uso de la herramienta de control de versiones para subir las actualizaciones de código.
- El uso del sistema de integración continua.

1.5.3.1 CREACIÓN DEL *BACK-END*

El siguiente aspecto que se debe tener en cuenta es el desarrollo de las funcionalidades *back-end*[12], entre los cuales cobran especial importancia las encargadas del acceso a datos mediante bases de datos o servicios como la pasarela de pago TPV virtual de Caja Laboral, entre otras muchas.

Se ha decidido dividir el *back-end* en dos servidores. Por un lado, el incluido en *Firebase*, *Functions* (Google, Firebase Functions Documentation, 2018) y por el otro *Heroku* (Heroku, 2018). Los motivos por los cuales se ha tomado la decisión están redactados en el apartado 2.2.3.4 de este documento.

Además de los argumentos de esa decisión, en ese mismo apartado también se explica la finalidad que tiene cada servidor y las acciones que estos ocupan dentro del sistema.

1.5.3.2 CREACIÓN DEL *FRONT-END*

Uno de los aspectos con más importancia en el desarrollo de una aplicación es el de concretar el contenido a mostrar en cada interfaz que contiene el programa. Además de ello, se debe considerar que dicha interfaz debe ser intuitiva para el usuario. Es decir, el objetivo es que el diseño sea lo suficientemente atractivo como para que el usuario cumpla su objetivo con apenas unos clics. Con el fin de lograr esto último, se mantendrá una estrecha colaboración con el equipo de diseño.

Conviene mencionar que cada una de estas interfaces serán mostradas en una reunión llevada a cabo por el responsable y el grupo de desarrolladores. Así, una vez definidas estas interfaces, se procederá a realizar el mencionado desarrollo.

1.5.3.3 CREACIÓN DE TRADUCCIONES

Una vez concretado lo anteriormente establecido, se procederá a añadir los textos que resulten necesarios, en dos idiomas (Castellano y Euskera), mediante el uso de la herramienta de traducciones, en este caso i18next (i18Next, 2019).

Cabe destacar que finalmente no se ha integrado la funcionalidad, o mejor dicho está integrada de forma oculta, debido a que el Ministerio de Industria no solicitaba que la aplicación

[12] Término que se refiere al motor funcional de una aplicación.

estuviera en ambos idiomas. Pero en el supuesto de que interesase añadir el soporte multilenguaje en un futuro, sería sencillo habilitarlo.

1.5.3.4 CONTROL DE CALIDAD

Una vez realizadas las traducciones, se procederá a realizar los testeos de las actualizaciones de estado y servicios. Estos testeos simulan la interacción entre una persona y la aplicación o los servicios.

Las actualizaciones de estado garantizan que los componentes cambian su estado interno, en el cual se almacenan diferentes contenidos como los textos que deben mostrar, el estado de la contratación, el color mediante el cual deben visualizarse a la persona usuaria…

Es importante llevar a cabo la comprobación de que los estados cambian correctamente mediante la interacción del usuario, es por ello el motivo por el que se han realizado diferentes testeos mediante Mocha (Mocha, 2019).

Además de comprobar el correcto funcionamiento de las actualizaciones de estados, también se ha testeado que los servicios funcionen correctamente. Para ello, se han simulado varias peticiones al servidor, cada una con diferentes parámetros, y se ha verificado que la respuesta es acorde a los parámetros enviados en la petición.

Como se ha mencionado anteriormente, estos testeos ocurren cada vez que se actualiza el código, de modo que la herramienta de integración continua se encarga de realizarlos para verificar la robustez de la nueva funcionalidad desarrollada. En caso de que estos testeos no sean superados, el *commit*[13] no se publicará y avisará al equipo de desarrollo mediante correo electrónico, con el fin de darle una solución lo antes posible.

1.5.3.5 CREACIÓN DE LA DOCUMENTACIÓN

El siguiente paso por realizar se basa en escribir la documentación de cada pantalla añadida a la aplicación. El objetivo de esto es que toda funcionalidad quede bien explicada, de modo que, si en el futuro se incorporara algún miembro nuevo al equipo de desarrollo, obtenga información acerca del código mediante dicha documentación escrita.

Por otro lado, con el fin de añadir un extra de comodidad al cliente del producto, se ha decidido publicar documentación mediante *Swagger*, la cual proporciona la posibilidad de referenciar cada *target* del servicio de modo que es posible conocer los parámetros de entrada o salida que contiene cada *path*[14] de los servicios.

1.5.3.6 REVISIÓN POR PARTE DEL RESPONSABLE

Como se ha mencionado anteriormente, cada funcionalidad desarrollada está supervisada por el responsable en la reunión llevada a cabo semanalmente en las instalaciones de la

[13] Término utilizado para referirse a la idea de confirmar un conjunto de cambios provisionales en el código de forma permanente.

[14] Término utilizado para referirse a la ruta disponible en el servidor.

empresa. En dicha reunión se comprueba el cumplimiento de los estándares de desarrollo y calidad de la organización.

1.5.4 FASE DE RESOLUCIÓN DE INCIDENCIAS

Como se ha mencionado anteriormente cada vez que una actualización de código es realizada, el sistema de control de versiones se encargará de comprobar que todas las condiciones acordadas antes de comenzar el desarrollo (métricas, dentado…) se cumplen además de testear el código para verificar la robustez de la funcionalidad desarrollada. Por si todo esto fuera poco, se procederá a realizar un control más exhaustivo del código desarrollado mediante la utilización de herramientas que garanticen el software de calidad.

Tras describir cada una de las fases por las que pasará el desarrollo del proyecto se procederá a explicar los equipos mediante los que se llevarán a término las tareas.

1.6 PLIEGO DE CONDICIONES

En el siguiente apartado, se presentarán los pliegos de desarrollo y ejecución del proyecto.

1.6.1 PLIEGO DE CONDICIONES TÉCNICAS DE DESARROLLO

En este apartado, se definirán tanto los equipos, como los softwares necesarios para el correcto desarrollo del proyecto.

1.6.1.1 ESPECIFICACIONES DE EQUIPOS

Será necesario un equipo con las siguientes características mínimas:

- Procesador Intel Core i3 de 2 núcleos y 2 hilos.
- Memoria RAM de 4GB DDR3.
- Disco duro de 125 GB.
- Sistema Operativo Windows 10.

Se ha desarrollado el proyecto mediante dos equipos diferentes cuyas características son:

Equipo 1:

- Procesador Intel Core i7 4710HQ @ 2.50GHz de 4 núcleos y 8 hilos.
- Memoria RAM de 8GB DDR3L a 1600 MHz.
- Disco duro SSD de 250 GB.
- Sistema Operativo Windows 10 de 64 bits.

Equipo 2:

- Procesador Inter Core i5 8250U @ 2,3 de 4 núcleos y 8 hilos.
- Memoria RAM 8 GB 2133 MHz LPDDR3.
- Disco duro SSD M2 de 256GB.
- Sistema Operativo Mac OS 10.14.5 Mojave.

1.6.1.2 ESPECIFICACIONES DE SOFTWARE

En lo que se refiere al software utilizado durante la ejecución del proyecto, ISEA no proporciona ningún tipo de software preseleccionado, por lo que la selección de este resulta ser parte del propio proyecto. Las principales herramientas utilizadas son las siguientes:

El software necesario para el desarrollo del proyecto es el siguiente:

- **Visual Studio Code** para el desarrollo del código del proyecto.
- **React** para crear aplicaciones web usando solo JavaScript. Este permite componer una interfaz multidispositivo a partir de componentes declarativos.
- **Adobe XD** para el desarrollo del prototipo de las interfaces.
- **GitLab**, un producto integrado líder para el desarrollo de software moderno. Un mismo producto para la administración de problemas, control de versiones, revisión de códigos, CI, CD y monitoreo de este.
- **Yarn o NPM** para la gestión de dependencias y despliegue del proyecto.

1.6.1.3 ESPECIFICACIONES DE CALIDAD DE SOFTWARE

Se debe de tener en cuenta que, a la hora de desarrollar el producto, este debe cumplir unos mínimos estándares de calidad, así como unas normas de codificación concretas. Con el fin de conseguirlo, se utilizarán las herramientas previamente mencionadas.

1.6.2 PLIEGO DE PRESCRIPCIONES TÉCNICAS DE EJECUCIÓN

En este apartado, se definirán todos los materiales y equipos que constituyen el proyecto y son necesarios para su puesta a punto.

1.6.2.1 ESPECIFICACIONES DE MATERIALES Y EQUIPOS

Para el uso de la aplicación desarrollada durante el proyecto, es necesario que la persona usuaria utilice un equipo con las siguientes características:

- Procesador Intel Core i3 de 2 núcleos y 2 hilos.
- Memoria RAM de 4GB DDR3.
- Disco duro de 125 GB.
- Sistema Operativo Windows 10 o macOS.

Asimismo, es necesario un servidor con el proyecto alojado con el fin de que la persona usuaria pueda acceder remotamente y hacer uso de la aplicación.

1.6.2.2 ESPECIFICACIONES DE SOFTWARE

El cliente debe ejecutar el programa en la siguiente lista de programas:

- La parte de contratación se podrá ejecutar en navegadores web tales como Google Chrome, Mozilla Firefox, Microsoft Edge…
- La parte del servicio REST se podrá consumir por el protocolo HTTP.

2 DESARROLLO

Es durante la fase de desarrollo donde tiene lugar la creación tanto de la aplicación como de los servicios web. A lo largo de este apartado se procederá a describir la producción desde el inicio del proyecto, incluyendo la selección del software utilizado, el pliego de condiciones una vez seleccionadas las diferentes especificaciones y finalmente el desarrollo como tal.

2.1 TOMA DE DECISIONES

El principal aspecto que se aborda durante la fase inicial es el análisis y selección de la mejor opción de desarrollo tanto para la aplicación como de los servicios web.

Hoy en día el desarrollo de aplicaciones web se divide en varios apartados. Por un lado, existe el *front-end*, el cual hace referencia a la visualización del usuario navegante y por otro lado el *back-end*, el cual se encarga de recolectar y suministrar los datos que posteriormente serán mostrados mediante el *front-end* al cliente.

Dichos apartados, a la hora de escribir estas líneas, tienen varias alternativas de desarrollo, cada una con sus ventajas y desventajas. Estas serán analizadas en el siguiente apartado.

2.1.1 POSIBILIDADES DE DESARROLLO DEL *FRONT-END*

En el siguiente apartado se valorarán las herramientas React (Facebook, React, 2019) y Angular (Google, Angular, 2017). Estas sirven para el desarrollo del *front-end* de la aplicación. Se valorarán tanto aspectos positivos como negativos de ambas, llegando finalmente a una conclusión.

2.1.1.1 DESCRIPCIÓN DE *FRAMEWORKS*

Angular es un *framework*[15] que facilita la realización *front-end* de una aplicación desarrollado por Google. Es compatible con la mayoría de los editores de código y forma parte del grupo MEAN, la cual es un conjunto de herramientas gratuitas centradas en JavaScript de código abierto para crear aplicaciones web. Este grupo está formado por *MongoDB*, que es una base de datos NoSQL, Express.js, el cual es un *framework* de aplicaciones web, Node.js que es una plataforma de servidor y finalmente la propia Angular. Este *framework* es utilizado por Forbes, WhatsApp, Instagram, HBO y Nike, entre otros para realizar la parte *front-end* de sus aplicaciones web.

Por otro lado, *React* es una librería para JavaScript de código abierto que facilita la realización *front-end* de una aplicación desarrollado por Facebook. Está basado en JavaScript y *JSX*, una extensión PHP desarrollada por Facebook, que permite crear elementos HTML reutilizables para el desarrollo de *front-end*. *React* es utilizado por Netflix, PayPal, Uber, Twitter o Airbnb, entre otros para realizar la parte *front-end* de sus aplicaciones web.

2.1.1.2 CONJUNTO DE HERRAMIENTAS

[15] Término utilizado para referirse a un esquema o patrón para el desarrollo y/o la implementación de una aplicación.

Conviene dejar claro que *React* no es un *framework*, sino una librería que, junto a otras herramientas y librerías adicionales, forman un conjunto con el cual poder llevar a cabo el propósito: Desarrollar el *front-end* de una aplicación. Las herramientas que normalmente los desarrolladores incorporan al desarrollo son las siguientes:

- *Redux* (Redux, 2018) es un contenedor de estado, el cual acelera el trabajo que *React* debe hacer en las aplicaciones con gran tamaño. Este gestiona componentes en aplicaciones con muchos elementos dinámicos además de utilizarse para el renderizado.
- Otra herramienta para destacar es Babel, la cual se encarga de convertir JSX en JavaScript con el fin de que sea entendible de cara a los navegadores web.

Normalmente, en un desarrollo como el que nos ocupa en este proyecto, los componentes de la aplicación están divididos en diferentes ficheros. Por lo que a la hora de compilar la aplicación es necesario unirlos, es por lo que *Webpack* (WebPack.js, s.f.) debe incorporarse al conjunto de herramientas para el desarrollo.

Con el fin de gestionar las diferentes URLs[16] que la aplicación albergará es necesaria la implementación de *React Router* (ReactRouter, s.f.).

Por otro lado, Angular contiene todo lo necesario para empezar a desarrollar una aplicación, como pueden ser las siguientes herramientas:

- *RxJS* (RxJS, s.f.) es una librería para programación asíncrona que reduce el consumo de recursos al establecer múltiples canales de intercambio de datos. La principal ventaja de *RxJS* es que permite el manejo simultaneo de eventos de forma independiente, pero, aunque es compatible con una gran variedad de *frameworks*, es el que normalmente domina a la hora de desarrollar mediante Angular.
- Otra de las herramientas que Angular incluye es Angular CLI, la cual es una potente interfaz de línea de comandos que ayuda a crear las aplicaciones, debuggear[17] , testear[18] y finalmente desplegar.

Asimismo, incluye el renderizador[19] Ivy, que es de nueva generación y provoca un aumento significativo en el rendimiento general de la aplicación.

En la siguiente tabla se puede observar una comparativa a modo de resumen. En la que se pueden observar las principales características de cada herramienta:

[16] Se trata de una secuencia de caracteres que sigue un estándar y que permite denominar recursos dentro del entorno de Internet para que puedan ser localizados.
[17] Se le llama debuggear a la acción que permite la ejecución controlada de un programa o código para seguir cada instrucción ejecutada y localizar así el error.
[18] Los testeos evalúan las aplicaciones para detectar cualquier diferencia entre una entrada dada y su salida esperada.
[19] Término utilizado para referirse al encargado de realizar el proceso de generar una imagen visible.

React vs Angular		
	React	Angular
Compañía	Facebook	Google
Año de lanzamiento	2013	2010 (Primera versión), 2016 (Versión actual)
Lenguaje	JavaScript, HTML y JSX	JavaScript, TypeScript
DOM	Virtual	Real
Vinculación de datos	Unidireccional	Bidireccional
Tamaño de la App	Pequeño	Pequeño
Rendimiento	Alto	Alto
Valoración GitHub (estrellas)	113.719	41.871
Soporte por la comunidad	Alto	Alto
Curva de aprendizaje	Media	Alta

Tabla 1: Tabla comparativa entre React y Angular

2.1.1.3 ORGANIZACIÓN POR COMPONENTES

Ambas herramientas tienen una arquitectura basada en componentes. Lo cual viene a significar que las aplicaciones desarrolladas mediante las citadas herramientas contienen componentes modulares, cohesivos y reutilizables que se combinan entre ellos para construir las interfaces. Este tipo de arquitectura se considera la más conveniente para desarrollos como el que ocupa en este proyecto. En efecto, se estima que ello acelera el desarrollo mediante la creación de componentes individuales que permiten a los desarrolladores ajustar y escalar las aplicaciones empleando poco tiempo a ello.

Por otro lado, teniendo en cuenta cómo se va a organizar el código durante el desarrollo, este aspecto beneficia a las dos herramientas, pero como ambas la incluyen no desequilibra la balanza a favor de ninguna de ellas.

2.1.1.4 CÓDIGO

Angular hace uso de *TypeScript* (TypeScript, s.f.), el cual es un conjunto de JavaScript adecuado para proyectos más grandes. El principal objetivo de *TypeScript* es detectar fácilmente errores de escritura. Otras de las ventajas de *TypeScript* son una mejor navegación, la autocomprobación y una refactorización más rápida del código. Al ser más compacto, escalable y limpio, *TypeScript* es perfecto para grandes proyectos a escala empresarial.

React, en cambio, utiliza JavaScript ES6+ y *JSX* script. *JSX* es una extensión de sintaxis para JavaScript que se utiliza para simplificar la codificación de la interfaz de usuario, haciendo que el código JavaScript parezca HTML. El uso de *JSX* simplifica visualmente el código, lo que permite detectar errores y protegerlo de las inyecciones. *JSX* también se utiliza para la compilación de navegadores a través de Babel, un compilador que traduce el código al formato que es leíble por un navegador web. La sintaxis *JSX* realiza casi las mismas funciones que *TypeScript*, pero puede resultar demasiado complicado de aprender.

En cuanto a este aspecto, el hecho de que *JSX* proteja el código de posibles inyecciones, hace equilibrar la balanza a favor de *React*, puesto que la seguridad es un punto muy importante en el desarrollo del proyecto.

2.1.1.5 DOM

Document Object Model (DOM) es una interfaz de programación para documentos HTML, *XHTML* o *XML*, organizada en forma de árbol que permite a los scripts interactuar dinámicamente con el contenido y la estructura de un documento web y actualizarlos.

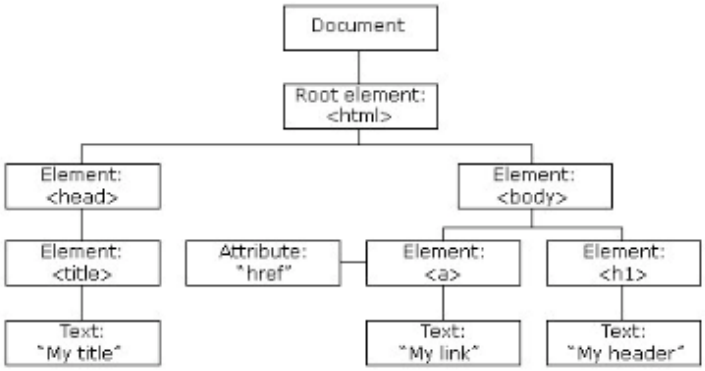

Ilustración 4: Árbol DOM de un documento HTML

Existen dos tipos de *DOMs*: Virtuales y reales. El *DOM* tradicional o real actualiza toda la estructura del árbol incluso si los cambios tienen lugar en un elemento. Por su parte, el *DOM* virtual es una representación mapeada a un *DOM* real que rastrea los cambios y actualiza solo los elementos específicos sin afectar a las otras partes del árbol entero.

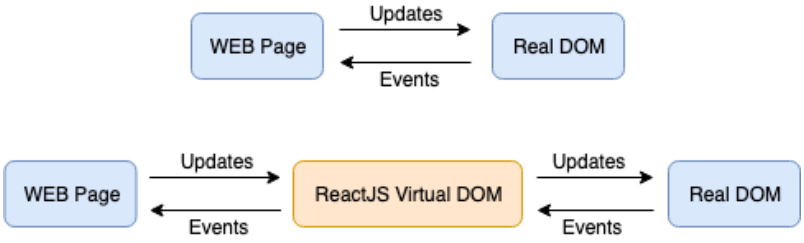

Ilustración 5: Comparación de la actualización del DOM corriente frente al virtual de React

React utiliza *DOM* virtual, mientras que Angular utiliza *DOM* real y, además, el estudio de cambios para detectar qué componente necesita actualizaciones.

Mientras que, sobre el papel, la implementación del *DOM* virtual es más rápida frente a las manipulaciones de *DOM* real, a la hora de la verdad, las implementaciones de detección de cambios de Angular hacen que ambas herramientas sean similares en términos de rendimiento. Por lo que este es otro punto que no desequilibra la balanza a favor ni de una ni de otra herramienta.

2.1.1.6 VINCULACIÓN DE DATOS

La vinculación de datos es el proceso de sincronización entre el modelo y la vista. Existen dos implementaciones básicas de la vinculación de datos: Unidireccional y bidireccional. La diferencia entre ellas radica en el proceso de actualización de la vista.

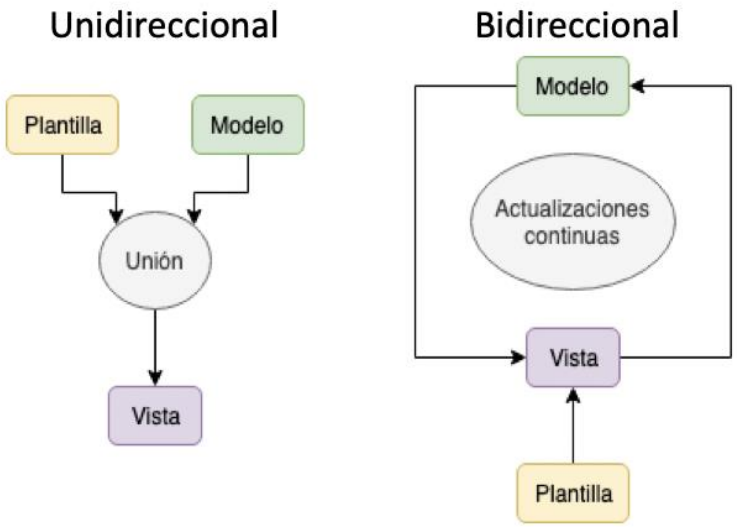

Ilustración 6: Comparación entre el enlace de datos bidireccional y unidireccional

El enlace bidireccional de datos de Angular es similar a la arquitectura Modelo-Vista-Controlador, donde el modelo y la vista están sincronizados, por lo que el cambio de datos afecta a la vista y el cambio de esta última desencadena modificaciones en los datos.

React utiliza el enlace de datos unidireccional. El flujo de datos unidireccional no permite que los elementos hijo afecten a los elementos padre cuando estos se actualizan. Este tipo de enlace de datos hace que el código sea más estable, pero requiere trabajo adicional para

sincronizar el modelo y la vista. Por otro lado, lleva más tiempo configurar las actualizaciones de los componentes principales activados por los cambios en los componentes secundarios.

El enlace unidireccional de datos en *React* es generalmente más predecible. Lo que hace que el código sea más estable y la depuración más fácil. Sin embargo, es más sencillo desarrollar mediante la vinculación de datos bidireccional de Angular.

En cuanto a este apartado, ambas herramientas tienen sus aspectos positivos y negativos por lo que probablemente sea otro aspecto el que haga orientar el desarrollo en una u otra herramienta.

2.1.1.7 TAMAÑO DE LA APLICACIÓN Y RENDIMIENTO GENERAL

AngularJS era conocido por su bajo rendimiento en el caso de aplicaciones complejas y dinámicas. Debido al *DOM* virtual, las aplicaciones desarrolladas mediante *React* funcionan más rápido que las aplicaciones Angular del mismo tamaño.

Sin embargo, las últimas versiones de Angular son ligeramente más rápidas en comparación con la combinación de *React* y *Redux*, según la investigación de Jacek Schae en *freeCodeCamp*.org (Schae, 31). Complementariamente, en esa misma investigación se puede observar que las aplicaciones desarrolladas mediante Angular tienen un tamaño de aplicación más pequeño en comparación con las desarrolladas con *React*: así, su tamaño es de 129KB, mientras que la de su versión desarrollada mediante *React* es de 193 KB.

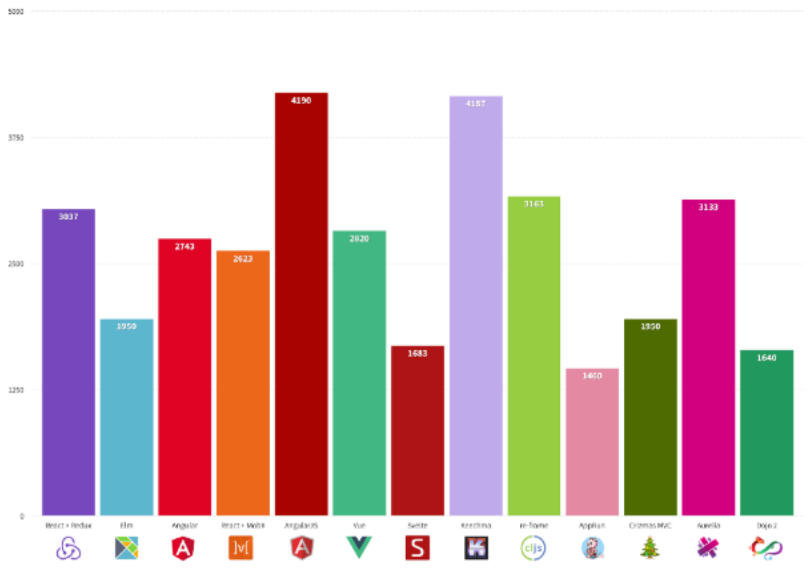

Ilustración 7: Prueba de velocidad (ms)

2.1.1.8 ELEMENTOS DE DISEÑO PRECONSTRUIDOS

La línea de diseño marcada por Google, *Material Design* (Google, Material Design, s.f.) es cada vez más popular en las aplicaciones web. Por lo que es un punto a favor de Angular, pues incluye componentes preconstruidos basados en esta línea de diseño como pueden ser: Formularios, métodos de navegación, botones, indicadores, ventanas emergentes o tablas de

datos. El hecho de que estos componentes estén incluidos hace que el desarrollo de las interfaces de usuario sea más agradable y rápido.

Por otro lado, *React* no incluye nativamente este tipo de componentes, pero cuenta con respaldo por parte de la comunidad, por lo que gran parte de los componentes provienen de ella. Curiosamente, existe una parte de la comunidad que se dedica a desarrollar componentes basados la línea de diseño de Google, *Material Design*. Es por ello por lo que, a efectos visuales, no existe ningún tipo de diferencia en cuanto a una aplicación desarrollada mediante Angular o *React*. Como aspecto negativo, cabe destacar que en caso de querer implementar los componentes basados en *Material Design* en un desarrollo de *React* se deben instalar dependencias, lo cual supone un paso adicional en la implementación respecto a Angular en el supuesto de que se quiera utilizar *Material Design* en la aplicación.

2.1.1.9 PORTABILIDAD

Aunque en el momento de realizar este análisis no se contempla el desarrollo de una aplicación de GAIA para dispositivos móviles, no está de más comprobar este apartado a efectos de posibles escenarios futuros.

En el caso de la portabilidad móvil, ambos *frameworks* contienen herramientas adicionales que permiten a los desarrolladores portear las aplicaciones web existentes a aplicaciones móviles.

En cuanto a Angular, contiene *NativeScript* (NativeScript, s.f.), el cual es un *framework* que utiliza *TypeScript* como lenguaje. Además, la interfaz de usuario se construye mediante XML y CSS. Estos son conocidos por todo el equipo de desarrollo, por lo que la formación no precisaría consumir tiempo.

Un gran punto a favor de esta herramienta es que permite compartir alrededor del 90% del código entre iOS y Android. A resultas, el portado de la aplicación web no supondría mucho esfuerzo para el equipo. La filosofía de *NativeScript* es desarrollar una única interfaz de usuario para móviles y ajustarla para cada plataforma si fuera necesario.

A diferencia de las soluciones hibridas multiplataforma que utilizan el renderizado *WebView*, el *framework* ejecuta la aplicación en una máquina virtual y se conecta directamente a APIs móviles nativas, lo que garantiza un alto rendimiento comparable al de las aplicaciones nativas.

Por otro lado, *React* ofrece su variante móvil llamada *React Native* (Facebook, ReactNative, s.f.), que también ofrece de forma sencilla la portabilidad de una aplicación web a una móvil. *React Native* tiene un enfoque ligeramente diferente en comparación a *NativeScript*: Se puede utilizar la misma interfaz tanto para web como para móvil ya que es *React Native* la encargada de modificar la interfaz.

Además, como punto a favor de *React Native*, cuenta con un enorme respaldo por parte de la comunidad por lo que existen una gran cantidad de componentes y plantillas totalmente gratuitas a disposición de los desarrolladores.

Otro elemento para destacar es que *React Native* también cuenta con renderizado de API nativo como *NativeScript*, pero requiere la construcción de APIs que formen un puente entre la ejecución en JavaScript y los controladores nativos.

Generalmente, ambos *frameworks* resultan ser una gran opción en caso de necesitar tanto una aplicación web como móvil. Mientras que *NativeScript* se centra más en compartir código y reducir el tiempo de desarrollo. Contrariamente, *React Native* precisa tiempos de desarrollo más largos, pero con la ventaja de que, dedicándole más atención, la aplicación se acerca más al aspecto nativo.

2.1.1.10 DOCUMENTACIÓN Y SOPORTE POR PARTE DE LOS DESARROLLADORES

Angular fue creado por Google y la empresa sigue desarrollando el *framework* hoy en día. Desde enero de 2018, Google proporciona el *framework* con soporte a largo plazo, lo cual significa que se centra en la corrección de errores y mejoras. Pero esto tiene un aspecto negativo y es que, debido a su rápido desarrollo, la documentación correspondiente se queda completamente obsoleta en cuestión de semanas. Este aspecto es contrarrestado por el gran apoyo que tiene por parte de la comunidad y por la disposición de esta última a ayudar con cualquier tipo de error o duda que pueda surgir.

Complementariamente, Google ha puesto a disposición de los desarrolladores una herramienta que facilita la lista de cambios a realizar en caso de querer actualizar la versión del *framework*.

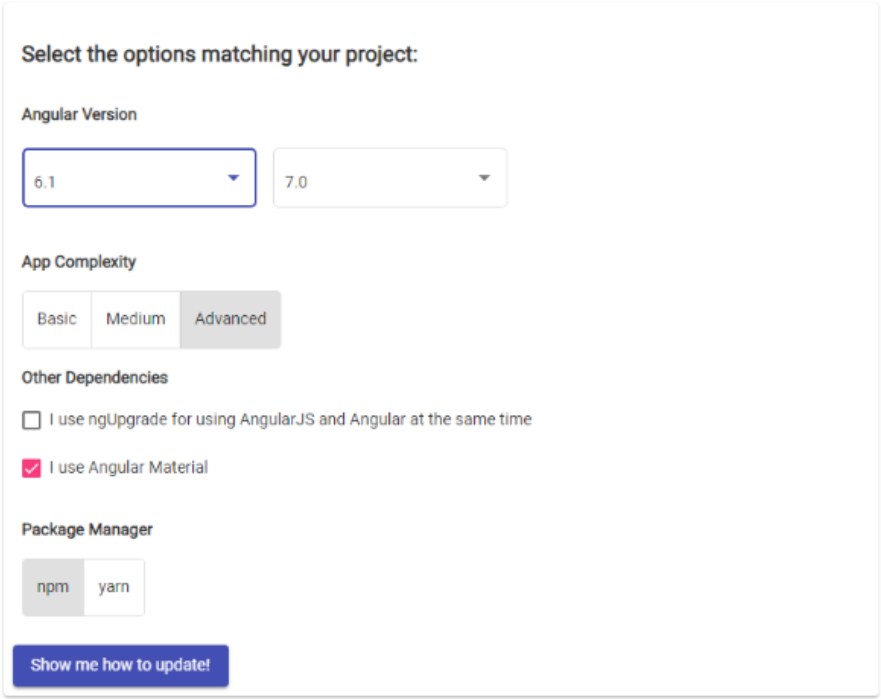

Ilustración 8: Interfaz mediante la cual Google proporciona la lista de cambios

Por otro lado, los desarrolladores de React experimentan un problema similar con la documentación, pues Facebook está constantemente mejorando y solucionando errores de su

herramienta. Sin embargo, este problema es neutralizado de alguna manera por el apoyo de la comunidad, ya que existen una gran cantidad de foros dedicados a la solución de problemas.

2.1.1.11 CURVA DE DIFICULTAD

La curva de aprendizaje de Angular se considera mucho más pronunciada que la de *React*. Angular es un *framework* complejo y con diversas formas de resolver un solo problema.

Como se ha mencionado en el apartado anterior, el *framework* Angular está en constante desarrollo, por lo que los desarrolladores tienen que adaptarse a los cambios. Otro aspecto negativo de *Angular* es el uso de *TypeScript* y *RxJS* ya que, aunque *TypeScript* es similar a JavaScript, su aprendizaje precisa.

Si bien *React* también requiere una atención constante debido a las frecuentes actualizaciones, generalmente es más fácil para los recién llegados y no requiere mucho tiempo de aprendizaje si previamente se conoce el lenguaje de programación JavaScript. Hoy en día, el principal problema de la curva de aprendizaje de *React* reside en la librería *Redux*. Alrededor del 60% de las aplicaciones desarrolladas con *React* lo utilizan. Finalmente, el dominio de *Redux* es una necesidad para el desarrollador de *React*. Además, este último contiene útiles y prácticos tutoriales para principiantes.

2.1.1.12 COMUNIDAD

React es más popular que Angular en GitHub. Tiene 113.719 estrellas y 6.467 visitas, mientras que Angular tiene solo 41.978 y 3.267 estrellas y visitas. Pero, según la encuesta de desarrolladores de StackOverflow (StackOverflow, 2018), el número de personas que trabajan con Angular es ligeramente mayor; 37,6% de los usuarios frente al 28,3% de *React*.

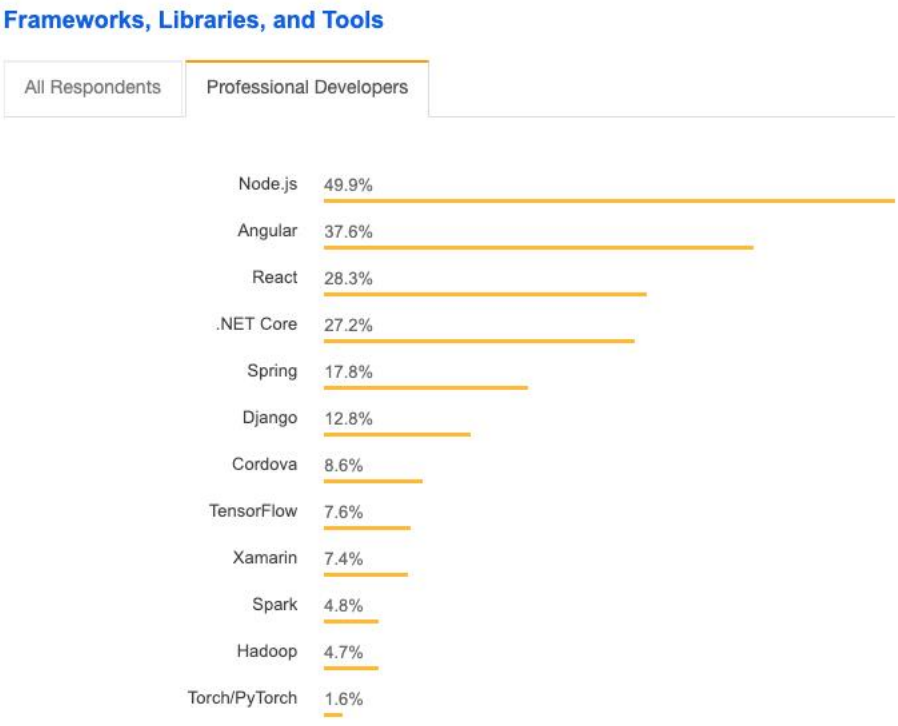

Ilustración 9: Encuesta de desarrolladores de Stack Overflow

Aunque Angular cuente con el apoyo de Google, el 45,6% de los desarrolladores lo consideran uno de los *frameworks* más temidos. Esto último puede deberse a que las versiones antiguas de Angular eran más complicadas de dominar.

Estas cifras son más optimistas para *React* ya que únicamente el 30,6% de los desarrolladores profesionales prefieren trabajar con otra herramienta.

Por último, siempre es conveniente realizar un estudio de mercado del uso actual de los softwares analizados y así, garantizar la mejor elección. Es por ello por lo que se ha realizado un análisis mediante Google Trends para ver a ver cuál es el *framework* que más tendencia marca en la sociedad actual, resultando las siguientes ilustraciones:

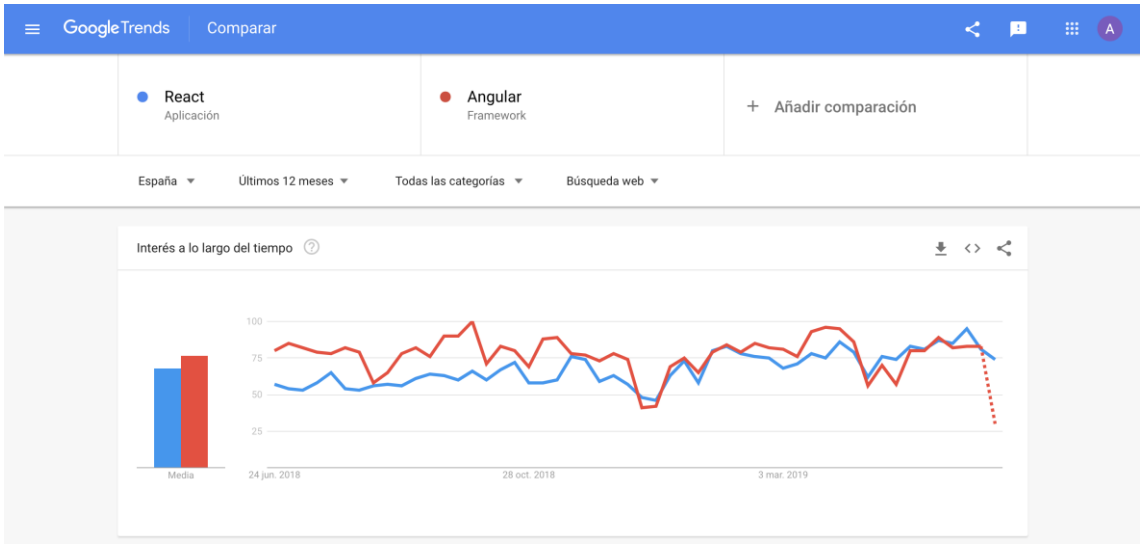

Ilustración 10: Tendencia de React frente a Angular en territorio nacional en el último año según Google Trends

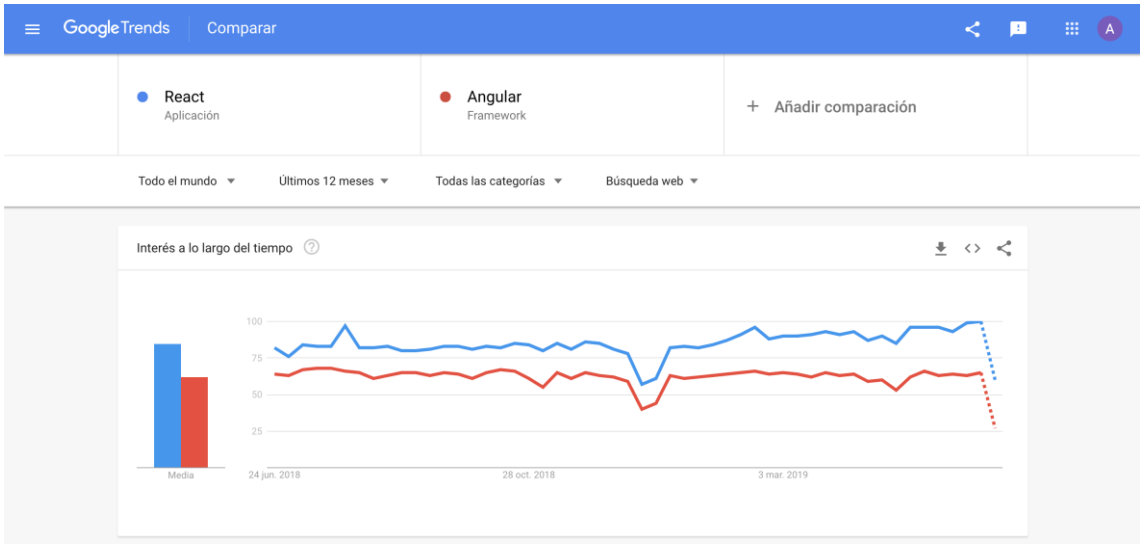

Ilustración 11: Tendencia de React frente a Angular en territorio internacional en el último año según Google Trends

Como se puede observar, en los análisis realizados mediante Google Trends, React resulta muchísimo más interesante a nivel internacional, aunque no tanto a nivel nacional. Algo para tener en cuenta, ya que derivará en un aumento de la comunidad dispuesta a solucionar dudas o posibles problemas.

2.1.1.13 CONCLUSIÓN Y DECISIÓN

Finalmente, tras analizar todos los ámbitos que pueden hacer decantarse por un *framework* u otro, llega la hora de tomar la decisión que marcará el porvenir del desarrollo del proyecto.

Resumiendo, la idea básica de Angular es proporcionar un soporte potente y conjunto de herramientas para una experiencia óptima en el desarrollo de aplicaciones web. Las continuas actualizaciones y el soporte de Google sugieren que el *framework* seguirá activo durante muchos años. Por otra parte, TypeScript aumenta la capacidad de mantenimiento del código, lo cual es importante en aplicaciones como la que se llevará a cabo en este proyecto. Pero este aspecto conlleva el precio de una pronunciada curva de dificultad, lo cual junto con el mal inicio que tuvo el *framework*, ha hecho que un gran número de desarrolladores se decanten por React.

Por otro lado, React sugiere un enfoque mucho más ligero para que los desarrolladores se pongan a trabajar rápidamente y sin necesidad de mucho aprendizaje. Aunque, individualmente no es tan completo como Angular, no es algo que no se pueda solucionar mediante la instalación de diferentes plugins que complementen su funcionamiento, como Redux. Además, debido a su DOM virtual, el rendimiento es prácticamente igual al de Angular. Estos aspectos hacen que el atractivo para el desarrollador sea mayor.

Otro de los puntos a favor de React es que, desde el punto de vista de la seguridad, el hecho de que utilice JSX conduce a un mayor nivel de protección vis a vis de las posibles inyecciones que se puedan recibir en un futuro.

Resumiendo, se podría decir que la batalla entre Angular y React se resume en un “Complejo y flexible, pero complicado de dominar” por parte de Angular frente a un “No tan completo, pero más sencillo de dominar” por parte de React.

Es por ello que teniendo en cuenta el tiempo disponible y el poco margen de maniobra del que se dispondría en caso de confusiones o errores, hace decantar la balanza por React, básicamente por la inmediatez en cuanto a puesta a punto de todo el entorno de desarrollo que reivindica la comunidad.

Complementariamente, si bien no es un elemento técnico comparativo, se ha de tomar en consideración que el equipo de desarrollo de la empresa tiene experiencia previa en React debido a los desarrollos anteriores (Tentu). Ello conduce a que esta decisión cobre más fuerza aún. Ya que en caso de que surjan dudas o problemas, se podrá recibir asistencia inmediata, sin llegar a tener que publicar ningún post en foros temáticos.

Por lo que, finalmente, el *framework* mediante el cual se desarrollará el *front-end* de la aplicación será React. Una vez decidido cómo se desarrollarán las interfaces de la aplicación, se tomará la decisión de donde alojarla.

2.1.2 ALOJAMIENTO DE LA APLICACIÓN

Hoy en día existen varias posibilidades para el alojamiento de la aplicación, con el fin de tomar esta decisión resulta interesante analizar diferentes opciones, ya que cada una tiene sus ventajas y desventajas.

De momento, es una incógnita la cantidad de usuarios simultáneos con la que va a contar la aplicación cuando esta sea lanzada al mercado. Por lo que se debe elegir la opción que más flexibilidad proporcione.

Por este motivo, prácticamente se puede descartar el hosting compartido, ya que este consiste en compartir una misma máquina entre varios clientes. Algo que no es adecuado ya que en caso de tener muchos usuarios simultáneos el rendimiento general de la aplicación se vería afectado.

Una de las opciones que más interesantes parecen en el mercado actual es *serverless*, ya que proporciona mucha flexibilidad y comodidad al desarrollador que lo utiliza. Por otra parte, este tipo de tecnología está en auge hoy en día, por lo que aprender a desarrollar mediante él hará ganar valor al equipo de desarrollo de cara al futuro.

En el siguiente apartado se analizará esta opción en profundidad.

2.1.2.1 ¿QUÉ ES SERVERLESS?

Hoy en día están en constante evolución las plataformas *serverless*, o como su nombre indica, sin servidor. Esta es una arquitectura donde los servidores (físicos o en la nube) dejan de existir para el desarrollador y en cambio el código corre en un entorno de ejecución que administran proveedores como por ejemplo Google, Amazon, IBM o Microsoft.

En el caso de las plataformas *serverless*, estas se ocupan de definir que lenguajes y versiones soporta y el desarrollador se encarga de realizar las aplicaciones mediante dichos lenguajes.

Uno de los beneficios de este tipo de plataformas es que en caso de que la aplicación pase a tener muchos usuarios, el proveedor se encarga de escalar el servidor con el fin de ser capaz de gestionar todos ellos. Lo cual proporciona mucha flexibilidad y comodidad ya que, entre otras cosas, el desarrollador no tiene que estar pendiente de actualizaciones del servidor, entre otras cosas.

Los proveedores cobran por el tiempo de ejecución del código, es decir, los recursos consumidos. Cuanto menos tiempo tarde en ejecutar nuestro código, menor será el coste. Por lo tanto, el objetivo del desarrollador es que la función sea corta y con un único propósito. Por este motivo *serverless* es relacionada frecuentemente con microservicios.

2.1.2.2 ¿EN QUÉ CASOS RESULTA ÚTIL SERVERLESS?

Un claro ejemplo en el cual el hecho de utilizar el alojamiento *serverless* beneficia tanto a la aplicación como al desarrollador que realiza el programa partiendo desde este tipo de *hosting* es Tentu, una revista electrónica personalizada. Esta se encarga de importar desde más de 700 fuentes RSS diferentes contenidos, para posteriormente clasificarlo y entregarlo al lector. Su arquitectura es administrada mediante *Firebase* y es organizada de la siguiente forma:

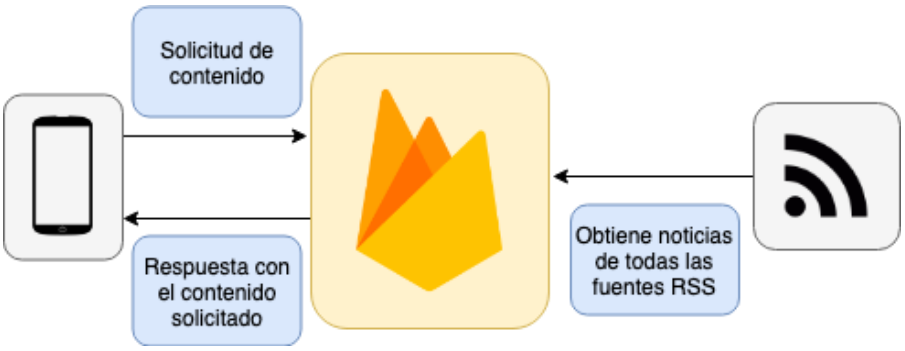

Ilustración 12: Diagrama organizativo de Tentu

El hecho de utilizar este tipo de alojamiento le beneficia claramente, ya que el equipo de desarrollo no tiene que estar pendiente de que el servidor no sea capaz de gestionar la simultaneidad de los usuarios. Además, el hecho de dividirlo todo en pequeñas funciones hace que la organización sea mucho mejor. Por lo que se podría decir que el hecho de utilizar este tipo de alojamiento le beneficia.

Otro ejemplo podría ser el de la realización de una tarea programada, en la que esta se encarga de leer un fichero csv y guardar el resultado en una base de datos cada cierto tiempo.

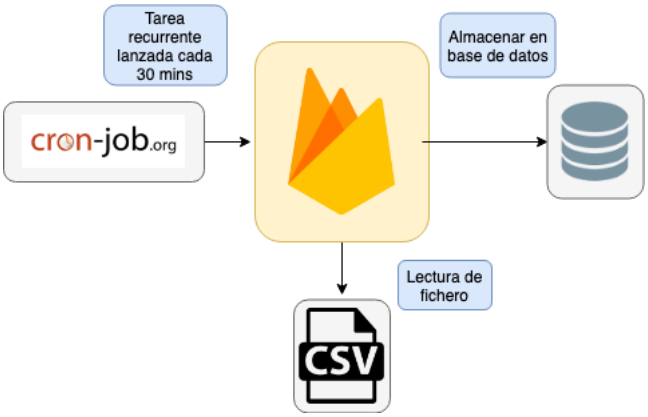

Ilustración 13: Diagrama organizativo de aplicación

En ambos, este tipo de alojamiento beneficia mucho a la organización, pero al igual que otros tipos de alojamiento, también cuenta con ventajas y desventajas que serán analizadas en el siguiente apartado.

2.1.2.3 VENTAJAS Y DESVENTAJAS

Nada es perfecto, *serverless* tiene sus aspectos positivos y negativos, que se analizarán en el siguiente apartado.

2.1.2.3.1 VENTAJAS

Gran parte de ellas tiene que ver con la comodidad que aporta al desarrollador:

- El desarrollador ya no se tiene que preocupar por mantener los servidores donde se aloja y ejecuta su código. Tampoco tendrá que instalar nuevo software, abrir o cerrar puertos, actualizar paquetes…

- Como se ha mencionado antes, esta tecnología es escalable, por lo tanto, el desarrollador no debe preocuparse por aumentar la capacidad del servidor, ni de que el mismo sea incapaz de gestionar todas las peticiones recibidas.
- La facturación del proveedor se basa en tiempo de ejecución, y en algunos casos, el tiempo de ejecución puede estar limitado. Ello conduce a penalizaciones de consumos prolongados en el tiempo.
- Las funciones se pueden integrar fácilmente con otros servicios del mismo proveedor. Si se utiliza *Firebase*, por ejemplo, es muy sencillo implementar la autenticación mediante la cuenta de Google.

2.1.2.3.2 DESVENTAJAS

Una parte de estas tienen que ver con que la tecnología se encuentra en un estado prematuro:

- Los entornos de programación (lenguajes, librerías, etc.) están limitados por el proveedor.
- Es un servicio sin estado, cualquier operación que quiera recordarse entre ejecuciones ha de apoyarse en otros servicios.
- La facturación del proveedor se basa en tiempo de ejecución, y en algunos casos, el tiempo de ejecución puede estar limitado. Ello conduce a penalizaciones de consumos prolongados en el tiempo.
- Las herramientas alrededor de la automatización del despliegue de funciones *serverless* son aún muy inmaduras.

2.1.2.4 PROVEEDORES

El más popular, sin duda, es Amazon con su servicio AWS Lambdas. Sin embargo, otros proveedores ya están ofreciendo soluciones muy similares. Los más conocidos son:

- Amazon, *Amazon Web Services* (Amazon, 2014).
- Google, *Cloud Functions*, *Firebase* (Google, Firebase Functions Documentation, 2018).
- Microsoft, *Azure Functions* (Microsoft, 2010).
- IBM, *OpenWhisk* (IBM, 2014).

2.1.2.5 CONCLUSIÓN Y SELECCIÓN DE PROVEEDOR

A modo de resumen, se puede decir que *serverless* es una nueva herramienta que permite desplegar aplicaciones web sin invertir esfuerzo en infraestructura y pagar tan solo por su uso.

Es diferente a IaaS *(Infrastructure as a* Service), donde se paga por servidores vacíos y el desarrollador se encarga de configurarlos, administrarlos y mantenerlos. También es diferente a PaaS *(Platform as a Service)* donde el desarrollador no administra los servidores, pero asume el coste de tener su código alojado, aunque no esté siendo utilizado continuamente.

Serverless es aún una tecnología nueva, e incluso experimental, ya que existían apartados en *Firebase* los cuales hasta hace unos meses estaban en *beta*[20]. Sin embargo, parece que tienen mucho potencial. En especial en el caso de desarrolladores que no tienen conocimiento en infraestructuras, pero desean implementar aplicaciones.

En el caso de la aplicación objeto de este proyecto, se ha decidido utilizar *Firebase* de Google, ya que tras leer varias comparativas como esta (Liberal, 2017), todas coinciden en que es la más sencilla para realizar la configuración inicial. También coinciden en que cuenta con menos flexibilidad; por ejemplo, con respecto a Amazon Web Services. Contrariamente, a ello contrapone una su sencilla configuración. En resumen, se podría decir que se trataría de un 'sencillo y simple' por parte de *Firebase,* frente a un 'complejo y flexible' por parte de *Amazon*.

Con relación a este último elemento, cabe destacar que esta falta de flexibilidad no es de importancia debido a que la aplicación web a realizar no requerirá mucha complejidad y los servicios que ofrece *Firebase* son más que suficientes.

Asimismo, con el fin de darle aún más peso a la decisión, se debe destacar que, en desarrollos anteriores de la organización, han utilizado *Firebase* como plataforma. Por lo que la configuración resultará más sencilla. De igual modo en caso de tener algún tipo de duda durante el desarrollo se podrá recibir soporte prácticamente inmediato.

En el apartado 2.3.3 de este mismo documento se analiza Firebase en más profundidad, especificando que características se utilizarán durante el desarrollo.

Tras decidir cómo se desarrollará la aplicación y mediante que proveedor se alojará, es el momento de seleccionar la herramienta con la cual se almacenará y gestionará el código fuente de la aplicación.

2.1.3 GESTIÓN DEL CÓDIGO

Una vez definido el software con el que se desarrollará el proyecto, es preciso igualmente elegir una plataforma donde almacenar y tener control sobre el código.

El código desarrollado hasta el momento ha sido almacenado mediante GitLab ya que se complementa muy bien, tanto para almacenar como para gestionar los testeos.

En GitLab, mediante la simple configuración de un archivo yml, un formato de serialización de datos legible, es posible lanzar testeos en la misma plataforma sin la necesidad de una configuración más compleja.

Por otro lado, se ha decidido aplicar varios métodos con el fin de tener una buena gestión del código. Son los siguientes:

[20] Aplicaciones las cuales tienen una o varias funcionalidades que se encuentran en estado de desarrollo y que aún no son al 100% estables.

- **Git Flow** para administrar '*branches*', características y lanzamientos. Se trata de un conjunto de comandos que encapsula algunos comandos de Git[21] en uno, de esta manera, facilita el uso de las mejores prácticas en proyectos que usan Git.
- **Husky** utilizado para lanzar un análisis de la calidad del código, en el nuevo código desarrollado antes de *commitar*.
- **Estilo predefinido** antes de *commitar*. Este estilo proporciona a la gestión del código información adicional en el mensaje, donde se pueden encontrar el tipo de contenido donde destacan:
    - **Feat**: Se refiere a una nueva funcionalidad del código.
    - **Fix**: Se refiere a la reparación de un problema en el código.
    - **Docs**: Se refiere a un cambio en la documentación.
    - **Style**: Se refiere a un cambio en el estilo de la aplicación.

2.2 DESCRIPCIÓN DE LAS TAREAS REALIZADAS

Remontándonos a los inicios del proyecto, las primeras tareas se centraron tanto en el diseño de las aplicaciones, como en la búsqueda de la mejor manera de adaptar ese diseño a una aplicación real mediante el uso de diferentes librerías/*frameworks*.

2.2.1 APRENDIZAJE

A lo largo de este proceso, se ha procedido a analizar la forma más efectiva de aprender un lenguaje nuevo. A tales efectos, existen cursos y documentación online que proporcionan conocimientos básicos o avanzados, estos pueden ser de utilidad para comenzar. Una vez iniciado el desarrollo, la variedad de situaciones a las que se puede enfrentar el desarrollador hace que obtenga experiencia. Ya que al final, enfrentándose a esa variedad de situaciones es como realmente se aprende.

Por lo que se ha decidido comenzar a recibir conocimientos básicos de la herramienta React a través de un curso en Udacity. De esta forma, se garantiza el nivel necesario para realizar una aplicación segura que pueda lanzarse al mercado.

2.2.2 DISEÑO DE LA APLICACIÓN

Cabe destacar que es muy importante dibujar las ideas y crear bocetos antes de empezar los proyectos. Es por ello por lo que los *mockups*[22] son de utilidad para que cuando se desarrolle el diseño final, no se pierda de vista el objetivo, ya que dichas maquetas sirven como referencia durante el desarrollo.

Por este motivo, se ha trabajado junto con el equipo de diseño de la empresa con el fin de lograr que la aplicación sea lo más cómoda posible de cara al usuario además de con una funcionalidad excelente.

[21] Término que se refiere al software de control de versiones.
[22] Término que se refiere a los bocetos en los que se representa cómo se va a mostrar la información.

La primera tarea de la aplicación se centra en el diseño de tanto la pantalla principal como la del área de clientes donde se encargará de gestionar sus datos, recoger sus credenciales, consultar su facturación…

Para ello, el equipo de diseñadores ha realizado un *mockup* de la visualización óptima de las mismas pantallas. Estos diseños se pueden observar en la siguiente ilustración.

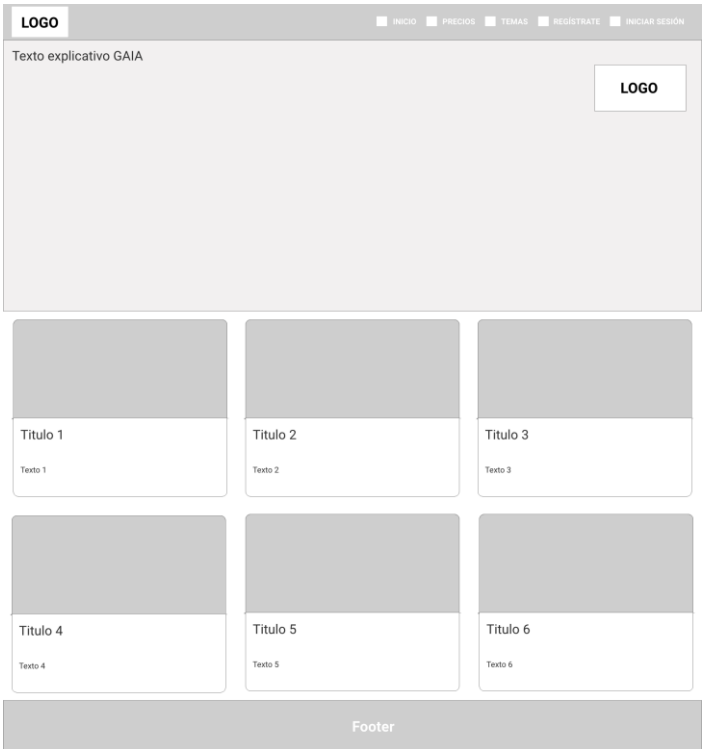

Ilustración 14: Prototipo de interfaz principal de GAIA

Posteriormente, con el fin de facilitar el desarrollo, se han llevado a cabo los *mockups* del resto de las pantallas. Con el propósito de facilitar el conocimiento del resto de pantallas, se ofrece en el anexo el fichero, incluyendo la posibilidad de su apertura con Adobe XD, de modo que se pueda comprobar la interacción. Por otro lado, en caso de que no se disponga de la aplicación, se ha publicado un enlace mediante el programa, con el fin de que se pueda consultar tanto el diseño como la interacción. También se dispone de una carpeta con diferentes capturas de pantalla con el fin de comprobar el diseño, en caso de ser preferible al enlace.

2.2.3 DESARROLLO DE LA APLICACIÓN

Una vez elegida la plataforma sobre la que se desarrollará la aplicación y los principales actores participantes, el siguiente paso consiste en el desarrollo de la primera versión de la aplicación web. A tales efectos se aplica el conocimiento adquirido previamente. Para ello, antes se deben conocer aspectos, como la organización del código.

2.2.3.1 ORGANIZACIÓN DEL CÓDIGO

Antes de comenzar a desarrollar, conviene buscar métodos de organización del código en el proyecto, ya que mantener todo el código estructurado en carpetas proporciona un orden que alguien que se incorpore al desarrollo de forma tardía agradecerá.

Como primer paso a tales efectos, se ha realizado un análisis de los métodos de organización más utilizados. Como el desarrollo se va a basar en el uso de diferentes componentes, colores e iconos, etc. Se ha decidido implementar *Atomic Design*, una idea organizativa realizada por Brad Frost (Frost, 2019).

Atomic Design se basa en la distinción de cinco principales componentes en la organización del código:

- Átomos
- Moléculas
- Organismos
- Plantillas
- Páginas

Cada nivel es más complejo que el anterior, hasta llegar a la más compleja: la página. Esta es un conjunto de todo lo anterior.

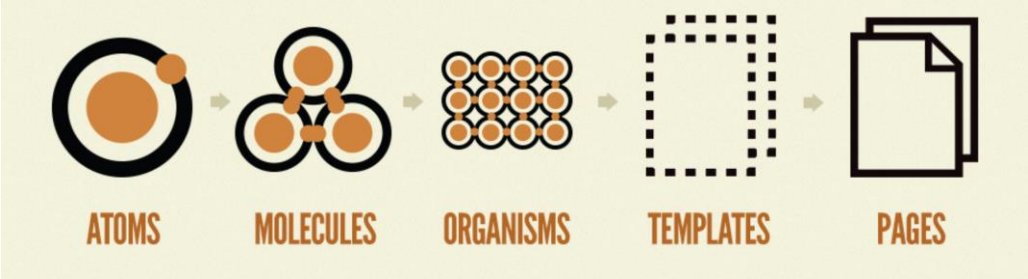

Ilustración 15: Unidades de Atomic Design

En las siguientes líneas se explicará en que consiste cada unidad que compone Atomic Design.

Átomos: Es el componente más simple de todos. Normalmente se les llama átomos a los botones, entradas de texto, o *labels*. Aunque es más raro, los átomos también pueden llegar a incluir fuentes de texto, paletas de colores o incluso animaciones.

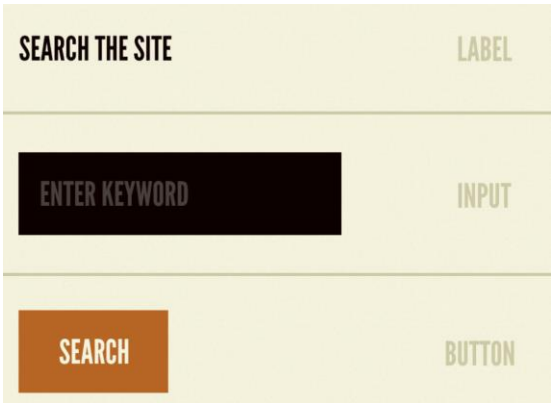

Ilustración 16: Ejemplos de átomos

Moléculas: Las cosas empiezan a ser más interesantes y tangibles cuando se combinan átomos. Las moléculas son grupos de átomos enlazados entre sí. Estas moléculas adquieren sus

propias propiedades y sirven como columna vertebral de nuestros sistemas de diseño. Por ejemplo, una entrada de texto o un botón pueden no ser demasiado útiles por sí solos, pero si se combinan entre ellos pueden llegar a serlo, formando entre ellos una molécula.

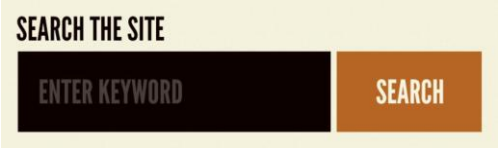

Ilustración 17: Ejemplo de molécula

Organismos: Los organismos son grupos de moléculas que se unen para formar una sección relativamente compleja de una interfaz. Los organismos pueden tener varias moléculas en su interior. Un ejemplo sería la barra superior de una aplicación web o *navbar*, compuesta por varias moléculas o átomos como podrían ser un logotipo, un título o un cuadro de búsqueda.

Al pasar de moléculas a organismos, se fomenta la creación de componentes autónomos, portátiles y reutilizables.

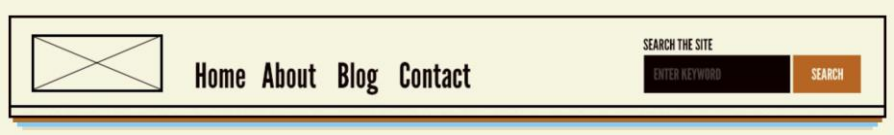

Ilustración 18: Ejemplo de organismo

Plantillas: Las plantillas consisten principalmente en grupos de organismos unidos para formar diferentes páginas. Las plantillas son muy concretas y proporcionan contexto a todas esas moléculas y organismos relativamente abstractos. Las plantillas son también el lugar donde los clientes ven el diseño final de la página.

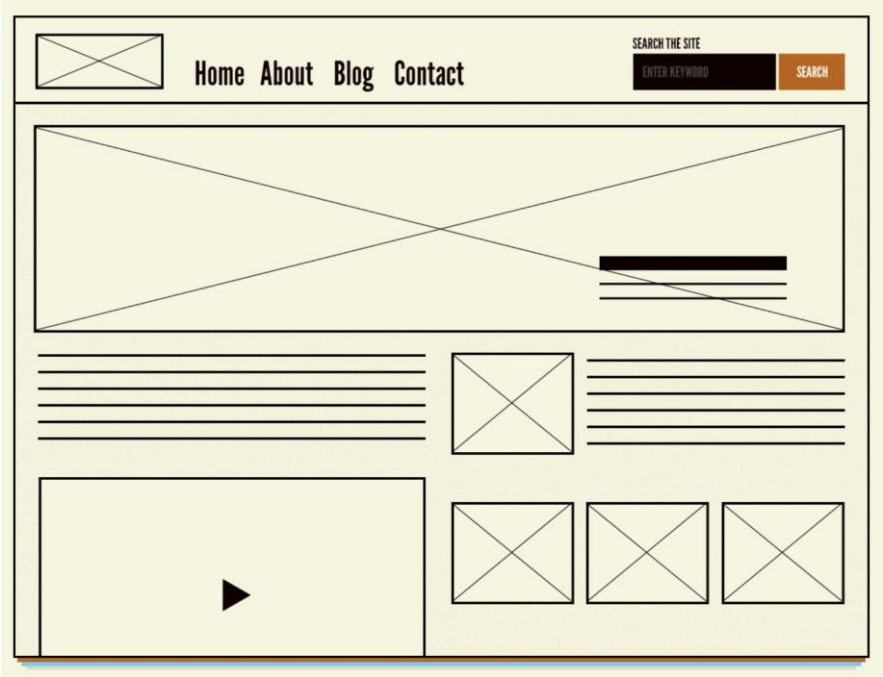

Ilustración 19: Ejemplo de plantilla

Páginas: Las paginas son instancias concretas de las plantillas. Aquí el contenido vacío se reemplaza con contenido representativo real para dar una descripción precisa de lo que un usuario verá en última instancia.

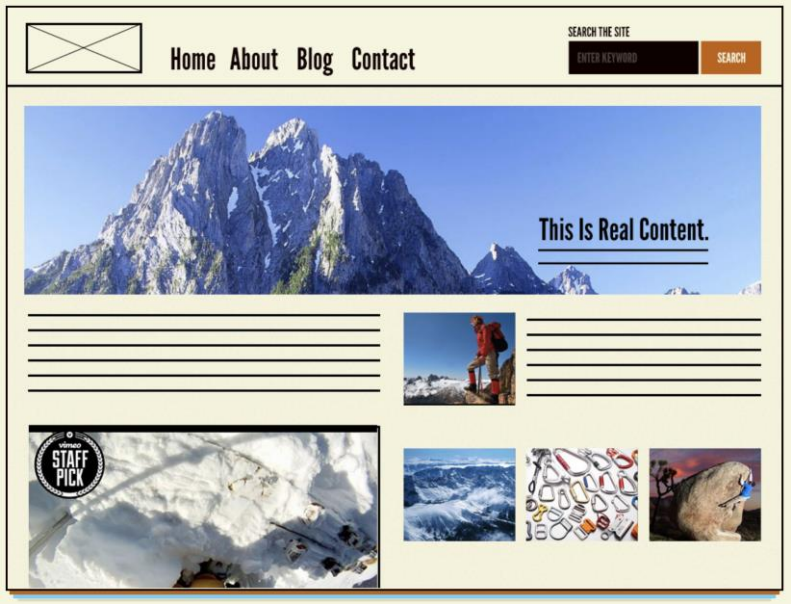

Ilustración 20: Ejemplo de página

En las siguientes líneas se aprecia un ejemplo práctico con la página inicial, en el cual se diferenciará cada uno de sus componentes:

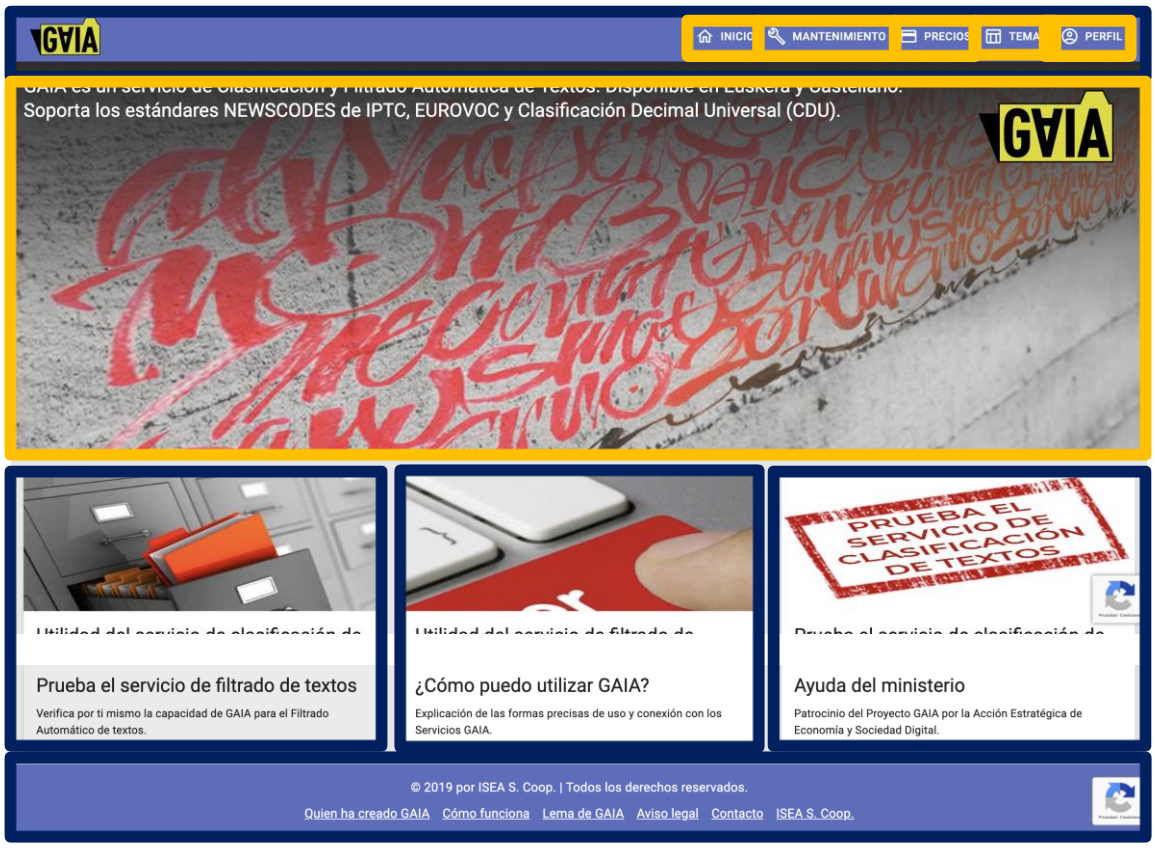

Ilustración 21: Página inicial de GAIA dividida en diferentes componentes

Los elementos que forman la página son los siguientes:

Por un lado, resaltado en naranja se pueden diferenciar los botones los cuales junto a los iconos que los acompañan forman una molécula. Todos estos botones, junto al logotipo del producto, forman el organismo llamado *NavBar*. La finalidad de este organismo es ayudar al usuario en la navegación de la aplicación web.

Por otro lado, se puede observar la molécula del carrusel de imágenes. Esta se encarga de modificar la cabecera cada cierto tiempo. Asimismo, además de las imágenes, contiene un texto informativo y el logotipo del producto.

En cuanto al siguiente área, esta se compone de varias *cards.* Se trata de organismos que contienen su propia interacción independiente. Cada una de ellas tiene una función diferente. Así, unas redireccionan a otra página, como puede ser la que se encarga de hacer la demostración; otras en cambio pueden ampliar la información mediante un diálogo.

Por último, el pie de página o *footer*, es un organismo que contiene un texto informativo y varios enlaces a otras páginas con el fin de ampliar la información.

2.2.3.1.1 ¿POR QUÉ ATOMIC DESIGN?

Atomic Design proporciona una metodología clara para crear sistemas de diseño. Los clientes y los miembros del equipo de desarrollo son capaces de apreciar mejor el concepto de los sistemas de diseño viendo realmente como están compuestos.

Asimismo, da la capacidad de pasar de lo abstracto a lo concreto. Por ello, se pueden crear sistemas que promuevan la consistencia y la escalabilidad, a la vez que muestran las cosas en su contexto final.

Para más información, existe un libro escrito por el mismo Brad Frost en el que se explica en profundidad razonamiento de principio de Atomic Design.

2.2.3.2 REACT

Una vez definidas las necesidades de cada pantalla, la plataforma donde poder desplegar la aplicación y el método para la organización del código, se ha llevado a cabo el desarrollo de la aplicación.

Como ha sido mencionado en el apartado 2.1.1 de este mismo documento, *React* es una herramienta increíble que simplifica el desarrollo de aplicaciones utilizando el conocimiento existente de JavaScript. Su capacidad para un desarrollo de aplicaciones más rápida, así como el intercambio de códigos para plataformas cruzadas sin comprometer la experiencia del usuario, le otorga un certificado de consideración para el desarrollo de aplicaciones.

Si bien es cierto que, *React* es una herramienta increíble, también es cierto que son muchos los inconvenientes que un desarrollador se puede encontrar a la hora de realizar un proyecto de mayor escala, como, por ejemplo:

- La necesidad de estructurar el código de manera eficiente para garantizar el futuro de la aplicación.

- La necesidad de seleccionar los mejores métodos de gestión de contenidos dentro de la propia aplicación.
- La necesidad de seleccionar los mejores paquetes externos para mejorar el funcionamiento de la aplicación.
- La necesidad de gestionar estos paquetes.

Estos requerimientos han conducido a las siguientes resoluciones:

- La necesidad de estructurar el código ha sido resuelta mediante la implementación de *Atomic Design.*
- La gestión de contenidos dentro de la aplicación ha sido resuelta mediante la utilización de Redux en combinación con Sagas.
- La selección de los paquetes externos se ha resuelto mediante la utilización únicamente de paquetes con garantía profesional.
- La gestión de los paquetes externos ha sido resuelta mediante la utilización de NPM.

2.2.3.3 BASE DE DATOS

Otro de los apartados a destacar de GAIA es la base de datos. En este caso se utilizará la incluida en *Firebase*, *Firestore* la cual es una base de datos no SQL y está estructurada de la siguiente forma:

- **Documento**: La unidad de almacenamiento principal es el documento. Un documento es un registro liviano que contiene campos con valores asignados. Cada documento se identifica con un distintivo único por colección.
- **Colección**: Los documentos se almacenan en colecciones, que simplemente son agrupaciones de documentos.

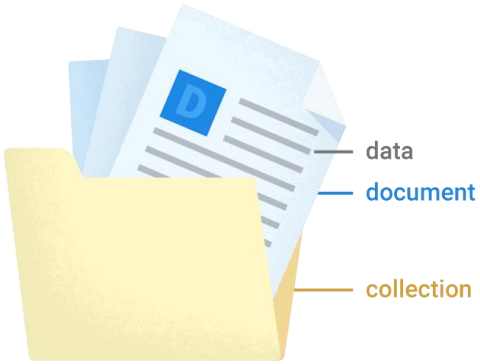

Ilustración 22: Gestión de datos Firestore

Una vez definida la estructura de datos, se ha identificado la información a guardar dentro de la base de datos:

- **Usuarios**: La gestión de los usuarios se realiza mediante *Firebase Authentication*, pero los datos utilizados por cada usuario dentro de la aplicación (contrataciones, códigos correspondientes) son gestionados mediante *Firestore*.

- **Historial de uso por parte de los clientes:** Cada vez que el usuario realice una consulta a los servicios web, esta se verá reflejada en la base de datos con el fin de que el usuario pueda visualizar su historial de consultas.
- **Lenguajes**: Como se ha mencionado anteriormente, la herramienta de traducciones esta implementada de forma oculta. Si en un futuro hiciese falta, no sería costoso implementarlo. Esta implementación oculta conlleva guardar en la base de datos los diferentes lenguajes utilizados en la aplicación.
- **Pagos**: Por cada pago que realiza el usuario, un documento es almacenado con diferentes datos como el coste, tipo de contrato realizado, además de su fecha y la duración de este.

Cabe destacar que toda la información almacenada en esta base de datos esta securizada mediante un sistema de reglas que proporciona *Firebase Firestore*. Por medio de estas reglas, se ha conseguido que ningún dato sea revelado si el usuario que intenta acceder a él no está autenticado, o si tampoco es el administrador.

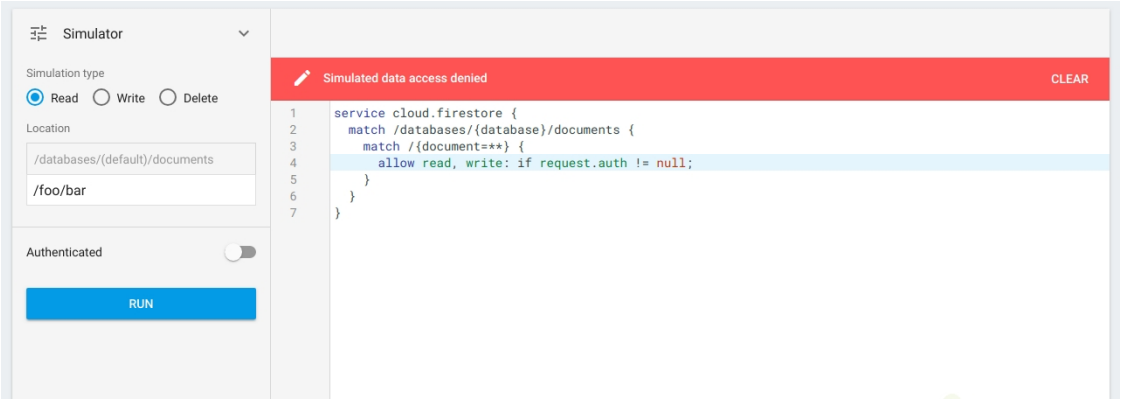

Ilustración 23: Simulador de reglas de Firestore

Asimismo, el sistema de reglas de Firebase Firestore contiene un simulador con el que poder comprobar que todas las reglas están redactadas correctamente. La finalidad de este simulador trata de que no haya ninguna fuga de datos y asegurarse de que nadie que no esté autenticado pueda acceder a los datos.

Por otro lado, con el fin de implementar el servicio de filtrado de contenidos, el equipo de desarrollo ha encontrado la necesidad de incorporar una segunda base de datos al sistema. Se trata de Algolia (Algolia, 2019) un proveedor *search as a service*[23]. Se trata de un servicio que ayuda a los desarrolladores a mejorar el filtrado de contenidos sobre la base de un resultado aproximado a la consulta. Así, por ejemplo, si la palabra introducida es “ostai”, lo calificaría como malsonante ya que esta es parecida a “ostia”.

[23] Término que se refiere al modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía.

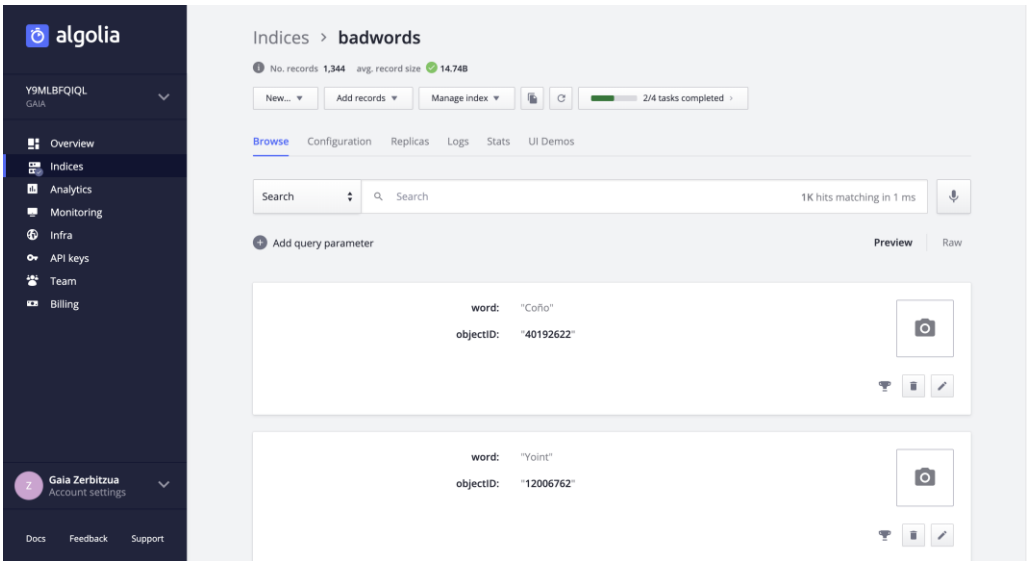

Ilustración 24: Interfaz de Algolia con el índice de palabras malsonantes

Esta segunda base de datos contiene el mismo listado de palabras malsonantes de *Firestore* y ha sido implementada ya que esta última no es capaz de dar ese valor aproximado a la consulta.

Tales bases de datos son manejadas en gran parte por el servicio *back-end*, el cual será analizado en el siguiente apartado.

2.2.3.4 BACK-END

Una vez analizada la gestión de la base de datos mediante *Firebase Cloud Firestore*, en el siguiente apartado se establecerá el funcionamiento del *back-end* de GAIA, el cual se soporta en la combinación de *Firebase Cloud Functions* y *Heroku*.

2.2.3.4.1 FUNCIONES IMPLEMENTADAS MEDIANTE GOOGLE FIREBASE

Antes de conocer las funcionalidades desplegadas mediante *Firebase Functions* es conveniente conocer más acerca de la herramienta *serverless* de Google.

2.2.3.4.1.1 GOOGLE FIREBASE

Firebase es la nueva plataforma de desarrollo móvil en la nube de Google disponible para diferentes sistemas (Android, iOS, Web). Sus prestaciones fundamentales se agrupan en los siguientes grupos funcionales:

- **Desarrollo**: Permite construir aplicaciones, permitiendo delegar determinadas operaciones en *Firebase*, para ahorrar tiempo, evitar bugs y obtener un aceptable nivel de calidad. Entre sus características destacan el almacenamiento, testeo, configuración remota, mensajería en la nube o autenticación.
- **Funciones Analíticas**: Provee una solución gratuita para tener todo tipo de medidas (hasta 500 tipos de eventos), para gestionarlo todo desde un único panel.
- **Escalabilidad**: Permite gestionar a los usuarios de las aplicaciones, pudiendo además captar nuevos. Para ello se disponen funcionalidades como las de invitaciones, indexación o notificaciones.

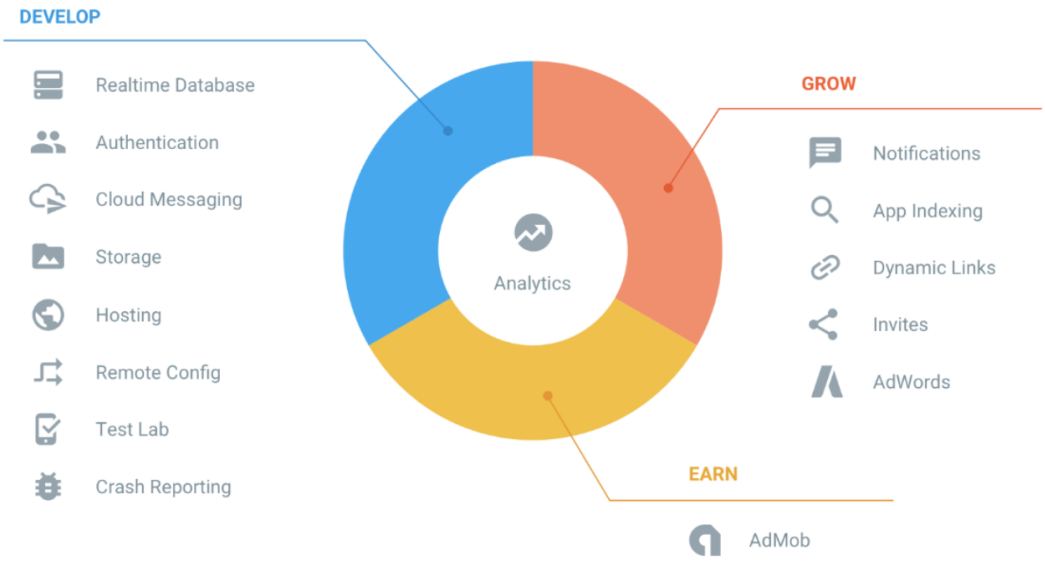

Ilustración 25: Principales herramientas que proporciona Firebase

Del conjunto de los productos que ofrece *Firebase*, GAIA utiliza los siguientes, aunque es posible que en un futuro incorpore alguna funcionalidad más.:

- ***Cloud Firestore Data Base***: Base de datos no relacional que GAIA utiliza para almacenar y sincronizar datos.
- ***Authentication***: Modulo de autenticación de usuarios. Proporciona una solución de identidad de extremo a extremo, compatible con cuentas de correo electrónico/contraseña, autenticación telefónica, inicio de sesión mediante cuentas de Google, Twitter, Facebook…
- ***Cloud Functions***: Permite ejecutar código *back-end* sin administrar servidores. Crea funciones que se activan con productos de *Firebase*, como cambios en los datos en *Realime Database*, o accesos de usuarios nuevos a través de *Authentication*.
- ***Hosting*** de la aplicación web.

En un futuro se espera poder utilizar los siguientes módulos:

- ***Google Analytics***: Permite conocer datos sobre el comportamiento de los usuarios en la aplicación, lo que permitirá tomar mejores decisiones sobre la optimización del marketing y del producto. Entre otros conceptos, se podrá ver el rendimiento de los enlaces directos, etc.

Para obtener información técnica detallada sobre estos elementos, Google ofrece un enlace (Google, Firebase Documentation, 2018), a través del cual es posible acceder a una gran cantidad de documentación, así como de ejemplos o incluso tutoriales.

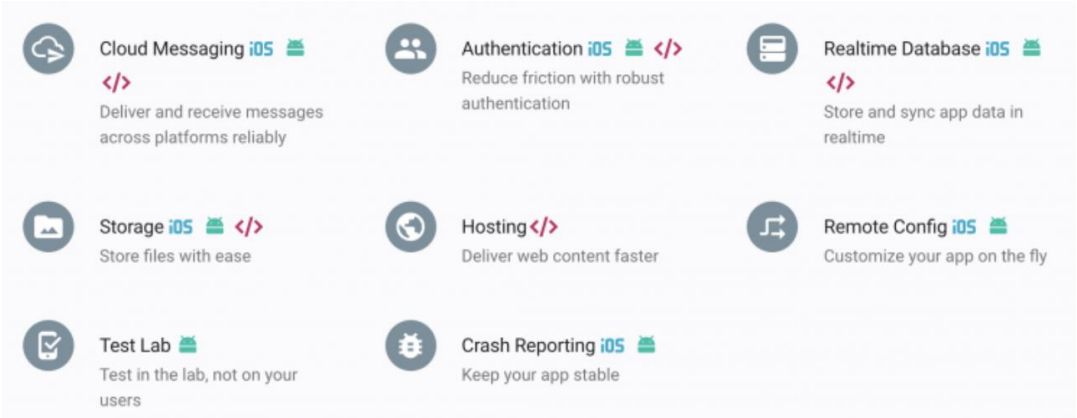

Ilustración 26: Tutoriales disponibles en la documentación de Firebase

Todo ello, tanto para iOS como para Android o web. En la guía se dispone de toda la información necesaria, clasificada por las categorías disponibles en *Firebase*, de modo que será muy intuitivo de leer y resultará sencillo su uso gracias a los ejemplos con código real.

2.2.3.4.1.2 FIREBASE FUNCTIONS

Cloud Functions permite ejecutar de forma automática el código de *back-end* en respuesta a eventos activados por las funciones de *Firebase* y las solicitudes HTTPS. De esta manera, es posible que GAIA gestione diferentes soluciones en la nube.

Las soluciones implementadas mediante *Cloud Functions* son las siguientes:

- **Comprobación del Recaptcha:** Con el fin de evitar que los bots o automatismos interactúen con las páginas web y asegurarse de que detrás de determinadas acciones hay siempre seres humanos se ha decidido respaldar esta comprobación en Cloud Functions.
- **Creación/eliminación de documentos de usuario:** La gestión de usuarios dentro del servicio se realiza mediante *Firebase Authentication.* Este servicio es muy completo, pero GAIA se ve con la necesidad de guardar más datos relacionados con el usuario, por lo que es necesario relacionar un usuario con un documento dentro de *Firestore*. En caso de eliminar la cuenta de dicho usuario este documento no es eliminado, por lo que se ha implementado un *trigger*[24] que se encargue del borrado de dicho documento.

El resto de las soluciones serán implementadas mediante *Heroku* debido a la baja velocidad de *Cloud Functions*. La implementación de Heroku será explicada en el siguiente apartado.

2.2.3.4.2 IMPLEMENTACIÓN DE UN *BACK-END* ALTERNATIVO

Debido a la baja velocidad en la que estas acciones se ejecutan, se ha decidido incorporar un servidor alternativo mediante la plataforma Heroku.

[24] Un trigger o disparados es un objeto que se asocia con tablas y se almacena en base de datos. Su nombre deriva por el comportamiento que presentan en su funcionamiento, ya que se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado.

Heroku es una plataforma como servicio de computación en la nube que soporta diferentes lenguajes de programación. Complementariamente, soporta Node.js, siendo este el entorno de ejecución en el que posteriormente serán desarrollados los servicios.

El principal cometido de este servidor es el de alojar los servicios que mas rapidez requieren de GAIA. Asimismo, también se encarga de gestionar diferentes funcionalidades necesarias para el correcto funcionamiento de la aplicación web, como podrían ser: la pasarela de pago, el servicio de recibir la numeración del siguiente tique o la comprobación de que todos los pagos estén al día.

La pasarela de pago se ocupa de que los usuarios contraten de forma segura los servicios. El proceso que sigue esta pasarela es la siguiente: el navegador del cliente cifra la información bancaria y la hace llegar a la plataforma de pago, que se ocupa de cobrarle al cliente y enviar el dinero a la organización. Durante este proceso, la pasarela asegura la confidencialidad y protección de los datos, lo cual es muy importante. En este caso se utilizará la pasarela de pago de Laboral Kutxa.

Una vez comprueba que el pago se ha realizado correctamente, la pasarela llama a un *path* de respuesta disponible en la plataforma la cual se encarga de modificar diferentes valores en la base de datos.

La numeración del tique no es otra cosa que consultar en la base de datos cual es el número de tique del último pago e incrementarlo, con el fin de que no existan dos tiques repetidos.

Además de ello, en la plataforma existe un *target* con el que poder comprobar si los pagos se han caducado o no. Para ello se ha hecho uso de una aplicación capaz de programar tareas llamada Cron-Job. Con la cual diariamente esa tarea es activada y las contrataciones de los usuarios comprobada.

A modo de resumen, se podría decir que se ha decidido implementar *Heroku*, debido a su mayor potencia y rapidez en comparación a *Firebase Functions*. Este se ocupa de las tareas que requieren mayor potencial y velocidad, como las citadas en este mismo apartado.

Para finalizar, la siguiente ilustración muestra las acciones de las que el servidor de *Heroku* se ocupa:

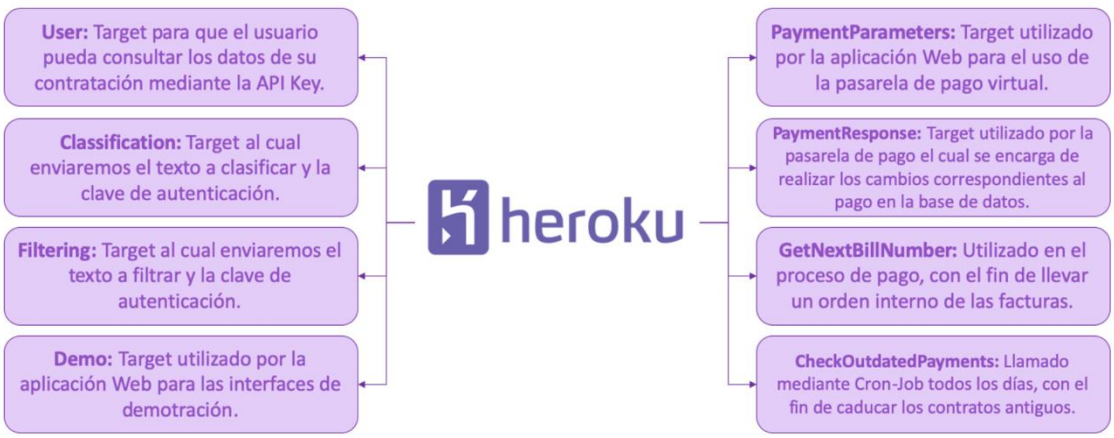

Ilustración 27: Diagrama explicativo de los servicios ofrecidos por Heroku

2.2.3.5 REUTILIZACIÓN DE COMPONENTES Y AUTOMATIZACIÓN DE PROCESOS

Una de las principales ventajas del desarrollo realizado es la reutilización de los componentes. La aplicación ha sido creada a base de componentes externos con el fin de reutilizar gran parte del código y evitar la duplicidad.

Además, se ha logrado uno de los objetivos, que consiste en controlar gran parte de los procesos como:

- En cuanto uno de los desarrolladores realice una actualización del código en el repositorio, el propio proyecto lanza un evento con el fin de realizar testeos y cumplir con el estándar acortado en un momento previo al desarrollo.
- Por otra parte, la herramienta de integración continua de GitLab es capaz de generar análisis de calidad del código, además de realizar los testeos necesarios y revisar cualquier incumplimiento en el estándar impuesto.

2.2.3.6 FUNCIONALIDADES ADMINISTRATIVAS

Hoy en día, el éxito de un producto depende de varios factores, uno de ellos puede ser el precio y otro el valor que aporta este producto respecto a su competencia.

Por este motivo, se ha implementado una interfaz que permite modificar el precio base y los rangos en los que está orientado el producto. Ello permite variar las condiciones de contratación de los servicios en función al volumen de uso de estos. A tales efectos se proporciona la siguiente interfaz:

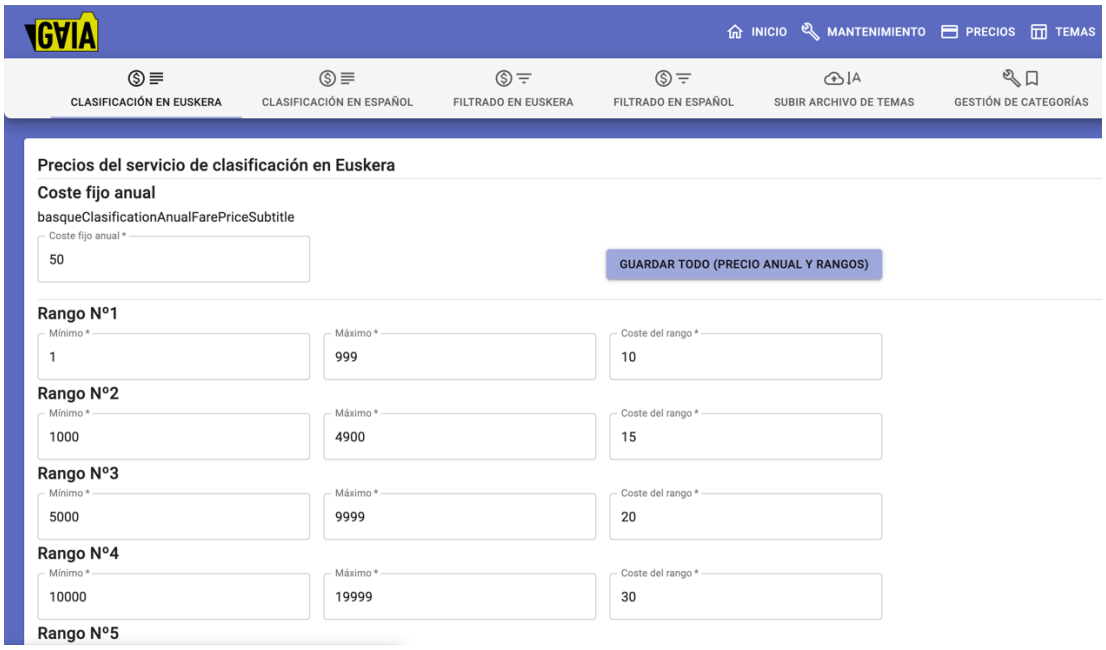

Ilustración 28: Interfaz para la modificación de precios de GAIA

Una vez realizados los cambios, el precio se podrá consultar en la siguiente interfaz:

Ilustración 29: Interfaz para la consulta de precios de GAIA

Otro de los datos a modificar es el listado de temáticas y sus equivalencias, en función a diversos estándares de clasificación internacional. El sistema GAIA ofrece un listado de temáticas actualizadas de la última versión disponible:

- La clasificación Decimal Universal (CDU)
- IPTC NewsCodes.
- EUROVOC.

Es por ello que, a tales efectos, se ha implementado la siguiente interfaz para la gestión de categorías:

Ilustración 30: Interfaz para la gestión de categorías

También se ha tenido en cuenta que con el paso del tiempo pueden surgir un importante volumen de cambios en los estándares de clasificación, aumentando, disminuyendo o cambiando las temáticas. Es por ello que se ha decido añadir otra modalidad en la que el administrador sube un fichero de texto delimitado por comas con un formato concreto. De este modo, el sistema se encarga de reemplazarlo por el listado temático actual. Dicha interfaz es la siguiente:

Ilustración 31: Interfaz para subir un archivo .csv con los temas

Tras llevar a cabo la explicación del desarrollo de la aplicación web, en el siguiente apartado se procederá a describir el desarrollo de los servicios.

2.2.4 DESARROLLO DE SERVICIOS WEB

Una vez finalizado el desarrollo de la aplicación web, se inicia el de los servicios, este, es el verdadero valor del producto ya que es lo que la organización va a poner a la venta realmente.

Por hacer un pequeño resumen sobre en qué consisten los servicios web, se dedican a automatizar la clasificación temática y el filtrado de palabras malsonantes de los textos que estos reciban.

A lo largo de este apartado se observará el desarrollo de estos, empezando por una explicación de la elección realizada en cuanto a la infraestructura.

2.2.4.1 ELECCIÓN DE INFRAESTRUCTURA

Una de las condiciones impuestas por una elección anterior, en este caso el hecho de seleccionar *Heroku* como plataforma de *back-end* y *Node.js* como forma de ejecutar JavaScript en el servidor da la opción, conduce a desarrollar los servicios mediante *Express.js.* Se trata de una infraestructura web, rápida, minimalista y flexible para Node.js.

Su sencilla instalación y aprendizaje de uso aportan un valor añadido, ya que permite a los desarrolladores comenzar a desarrollar sin invertir tiempo en formación.

Por lo que los servicios se desarrollaran mediante Express.js.

2.2.4.2 CATEGORIZACIÓN MEDIANTE SERVICIOS EXTERNOS

Una vez elegida la plataforma donde desplegar los servicios y la infraestructura con la cual desarrollarlos, es momento de centrarse en lo realmente importante, la categorización de los textos.

Para ello se utiliza un servicio externo desarrollado por UZEI, que se encarga de clasificar los textos en base a 31 temáticas (En el momento de escribir estas líneas, hay varias más en desarrollo y existe la previsión de llegar a los 151).

Todo este proceso es posible gracias a la colaboración de UZEI, Centro Vasco de Terminología, que, tras 39 años de trabajo en el ámbito de la lexicografía eusquérica y castellana, ofrece un conjunto de servicios web que son explotados para el proyecto.

El servicio por desarrollar actúa como puente entre el servicio que UZEI ofrece y el cliente, dando posibilidad a ISEA de poder cobrar por dicho servicio.

El proceso de categorización responde a la siguiente lógica:

- En primer lugar, una vez recibido el contenido por parte del cliente, este es enviado en un formato especifico al equipo de UZEI. Este formato debe incluir el idioma y el texto principal a clasificar.
- En segundo lugar, el equipo de UZEI analiza el contenido enviado mediante la utilización de un software que es capaz de basar su respuesta en un análisis extenso con la utilización de más de 200 diccionarios temáticos.
- Una vez analizado el contenido, UZEI devuelve las dos principales categorías o subcategorías a las que pertenece el contenido.
- En caso de error, este contenido no es categorizado y se envía un mensaje de error.

La comunicación entre ambos equipos se basa en un sistema de códigos acordada previamente, el cual indica la categoría a la que pertenece el texto.

El desarrollo de la categorización de contenidos mantiene en continuo contacto a los desarrolladores de UZEI e ISEA. Según un análisis realizado por un antiguo desarrollador de ISEA, el servicio de clasificación de UZEI ofrece los siguientes resultados:

- Temáticas tratadas: 31
- Porcentaje de acierto pleno: 75%
- Porcentaje de error: 24%
- Acierto no pleno: 2%

Cabe destacar que ambas entidades mantienen una estrecha comunicación con el fin de poder mejorar este porcentaje y acercarlo al menos al 90%.

Asimismo, ISEA ha comenzado internamente el desarrollo de su propia red neuronal, en principio es con fin experimental, únicamente con el objetivo de comprobar si es capaz de mejorar la clasificación de contenidos proporcionada por UZEI.

Otro de los objetivos de esta red neuronal podría ser clasificar los textos en base a temáticas generalistas como podrían ser los deportes o la tecnología y después enviar al clasificador de UZEI con el fin de tener una mayor exactitud temática. Es decir, si mediante la red neuronal se concreta que un contenido tiene que ver con el deporte, después se podría enviar al clasificador con el fin de saber si se trata de fútbol o baloncesto.

Con el fin de tener una base de textos con el que poder alimentar el modelo para el desarrollo de la red neuronal, se ha realizado una recopilación de artículos periodísticos, tal y como se muestra en el apartado de desarrollos adicionales.

Dejando el desarrollo de la red neuronal de lado, en la siguiente imagen se muestra un ejemplo de la comunicación entre UZEI e ISEA a través de la aplicación Postman (Postman, s.f.):

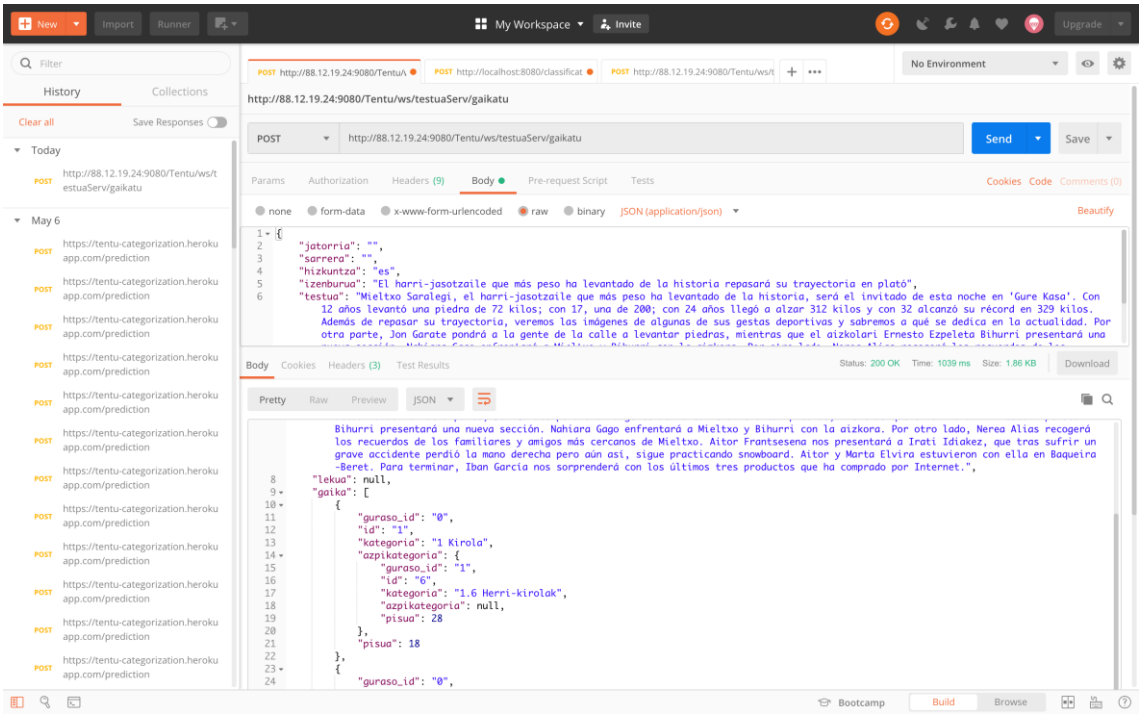

Ilustración 32: Ejemplo de comunicación entre UZEI e ISEA

2.2.4.3 FILTRADO DE CONTENIDOS

El filtrado de contenidos publicados en la mayoría de las plataformas online es uno de los principales problemas hoy en día. Por esta razón, el equipo de desarrolladores trabaja con el equipo de UZEI para lograr analizar textos y determinar si existe algún tipo de contenido inapropiado en los mismos, como podrían ser las palabras malsonantes.

El funcionamiento de este filtrado de contenidos es el siguiente:

- En primer lugar, se interpretan el número de mayúsculas que el texto tiene. Esto es debido a que, la probabilidad de que un contenido escrito completamente en mayúsculas sea inapropiado es grande. Un ejemplo podrían ser los insultos.
- En segundo lugar, el texto es analizado por UZEI. La comunicación se realiza del mismo modo que con la clasificación de contenidos. En este caso UZEI se encarga de analizar y lematizar el texto. Lematizar consiste en conseguir el "núcleo" de cada palabra que compone el texto.
- Posteriormente, el sistema GAIA mediante *Cloud Functions* analiza el texto en busca de coincidencias dentro de *Firebase Firestore*. En caso de encontrar coincidencias con la base de datos indica que palabras son las que han coincidido.

- En caso de no encontrar ninguna coincidencia, este análisis pasa por un filtrado final, consistente en la utilización de una segunda base de datos (Algolia) donde es posible encontrar palabras con cierto margen de error.
- Finalmente, se realiza un análisis mediante la utilización de la técnica de la distancia *Levenshtein.* Se trata de analizar el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Este margen de error es impuesto por el equipo de GAIA y el texto analizado es el inicial no el lematizado.

A continuación, se ilustra mediante un ejemplo en que consiste la distancia *Levenshtein*.

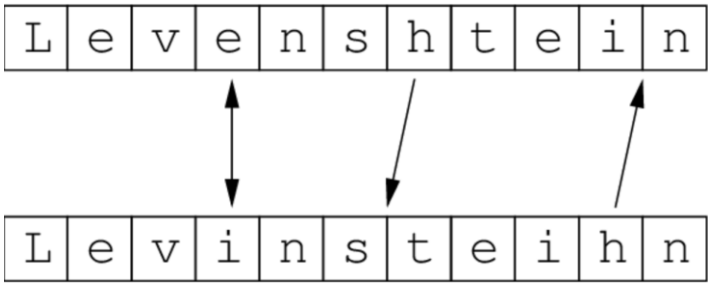

Ilustración 33: Ejemplo gráfico con el fin de explicar la distancia Levenshtein

Refiramos, a modo de ejemplo, que la técnica de la distancia *Levenshtein* en diferentes aplicaciones corrientes, como pueden ser el motor de búsqueda Google o el auto corrector del teléfono móvil.

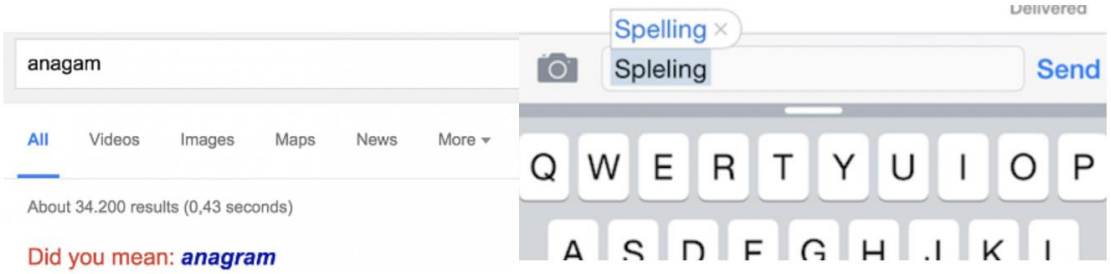

Ilustración 34: Ejemplos de uso de la distancia *Levenshtein* en diferentes en diferentes aplicaciones

A modo de resumen se podría decir que la técnica de la distancia *Levenshtein* se usa, en el caso de GAIA para encontrar palabras similares a una palabra malsonante. Contrariamente, en el caso de los ejemplos citados previamente, en cambio, la aplicación de la distancia *Levenshtein* se utiliza para prevenir posibles confusiones del usuario a la hora de introducir texto.

Tras este proceso el equipo es capaz de identificar contenido inapropiado y palabras malsonantes.

2.2.4.4 AUTENTICACIÓN, SEGURIDAD Y LIMITACIÓN DE LOS SERVICIOS

Con el fin de verificar la identidad del usuario que está utilizando los servicios web, se ha decidido implementar un sistema con *API Key*. Este valor es único y solamente el cliente puede generarlo desde la aplicación web.

La generación de esta *API Key* se basa en un algoritmo que tiene en cuenta la fecha y hora de la solitud de generación, de modo que ningún usuario va a tener nunca una *API Key* igual al de otro usuario.

Esta *API Key* debe ser enviada en el contenido de la petición POST de modo que el servidor pueda verificar que el usuario correspondiente a esa *API Key* tiene la contratación correctamente realizada, entre otras cosas como los códigos equivalentes que quiere recibir en la respuesta del servicio de clasificación.

El mayor problema de seguridad al que enfrentarse en este tipo de servicios es, entre otros, la suplantación de identidad. Es decir que un usuario que no paga los servicios haga uso de ellos mediante la *API Key* de otro que si lo hace.

Para dar solución a este problema se ha implementado un apartado en la página web donde el usuario, en caso de que le hayan robado la clave de identificación, puede regenerarla haciendo clic a un solo botón, inhabilitando al mismo tiempo la antigua clave.

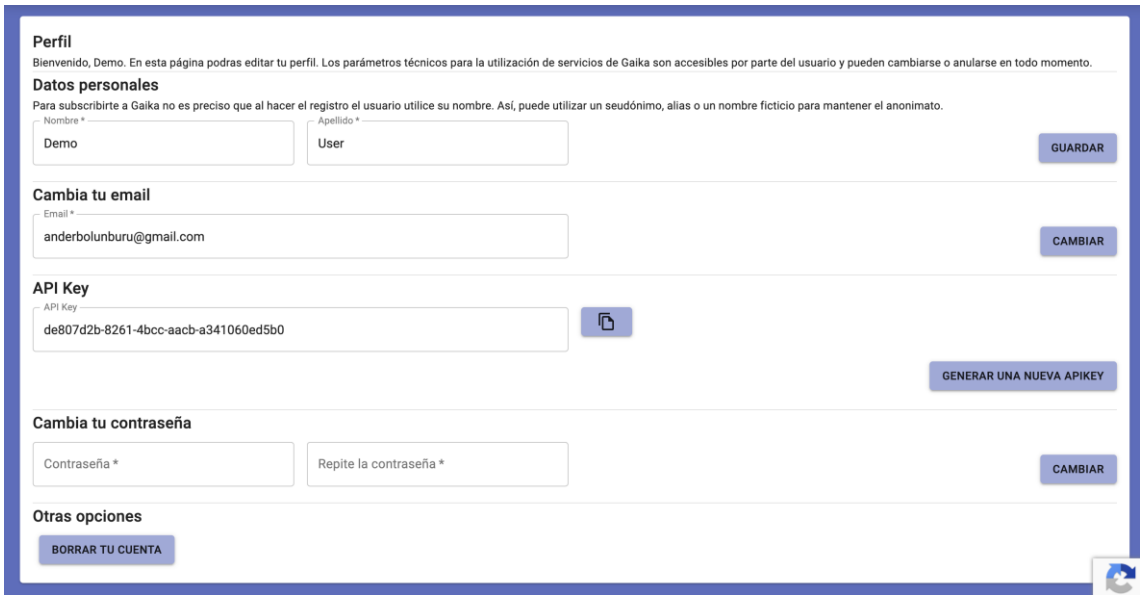

Ilustración 35: Interfaz mediante la cual el usuario puede modificar su perfil

Mediante esta implementación se evitaría que un usuario utilice los servicios gratuitamente aprovechándose de una *API Key* robada a otro usuario.

Tras implantar este sistema, se pensó en otros métodos de securización con el fin de hacer los servicios aún más sólidos. Inicialmente se valoró el hecho de limitar el uso de los servicios a un origen o dominio definido por el usuario mediante la aplicación web, pero tras contrastarlo con el experto en MONDRAGON Unibertsitatea, se decidió no implementarlo debido a que alguien con conocimientos técnicos podría *spoofearlo*[25] sin mucho problema.

A pesar de no añadir ningún otro método de securización, se ha pensado que puede ser oportuno posibilitar que el cliente pueda controlar el consumo de servicios GAIA. Para ello se ha

[25] En términos de seguridad informática, se refiere al uso de técnicas para la suplantación de identidad.

habilitado una pestaña en la aplicación web mediante la que el usuario puede condicionar su uso a través de la siguiente interfaz:

Ilustración 36: Interfaz mediante la cual el usuario puede limitar el consumo de sus servicios

Como se puede observar en la imagen, en este caso el usuario ha decidido limitar su consumo de clasificación a 3 solicitudes por minuto. Con el servicio de filtrado en cambio, ha preferido limitarlo a 30 solicitudes por minuto. De este modo puede llevar un control de su consumo.

Cabe destacar que en caso de que el usuario exceda su propio límite, recibirá un mensaje de error. Conviene recordar que el listado con todas las posibles respuestas del servicio está disponible en la documentación online de *Swagger*.

Además de estos riesgos, también se han contemplado otros en un análisis de riesgos realizado en el siguiente apartado.

2.2.4.5 ANÁLISIS DE RIESGOS

El propósito de este apartado es resumir el proceso de evaluación de posibles riesgos del servicio que se ofrecen con sus respectivas contramedidas.

2.2.4.5.1 POSIBLES RIESGOS

Los riesgos con más probabilidad de afectar a la aplicación son los siguientes:

2.2.4.5.1.1 AMENAZAS DE APLICACIÓN

1. Fuga de información
    a. Datos de usuarios
    b. Datos de contratación
    c. Credenciales asociadas a los usuarios
    d. Historial de uso de los usuarios
2. Inyección
    a. Javascript

   b. XSS
3. Intrusión/Ataques D/DOS
4. Crashes de ejecución
5. Incumplimiento de la LOPD

2.2.4.5.1.2 AMENAZAS DE ENTORNO

En el caso de GAIA las amenazas de entorno son inexistentes ya que la plataforma de alojamiento son *serverless*, por lo que ISEA no se ocupa realmente del mantenimiento de ningún tipo de servidor.

Además, el código fuente está alojado en una plataforma externa por lo que, en caso de avería de los equipos de la organización con motivo del entorno, este no correría peligro.

Por lo que son los proveedores, en este caso Google y GitLab, los que realmente se deben de ocupar de que sus servidores cuenten con la protección necesaria contra las amenazas de entorno más comunes como los cortes de tensión, el desbordamiento de ríos cercanos o el fuego.

2.2.4.5.2 EVALUACIÓN DE RIESGOS

Se consideran los niveles de coste y probabilidad en tres niveles (bajo, medio y alto).

Riesgo	Probabilidad	Coste directo	Coste indirecto	Total
Fuga de información				
Datos de usuario	BAJO	BAJO	BAJO	BAJO
Dataset del servicio	BAJO	MEDIO	ALTO	ALTO
Credenciales asociadas a los usuarios	BAJO	MEDIO	ALTO	ALTO
Inyección				
Javascript	BAJO	BAJO	BAJO	BAJO
XSS	BAJO	BAJO	BAJO	BAJO
Intrusión/Ataques D/DOS	MEDIO	ALTO	ALTO	ALTO
Crashes de ejecución	BAJO	ALTO	ALTO	ALTO
Incumplimiento de la LOPD	BAJO	ALTO	ALTO	ALTO

Tabla 2: Evaluación de riesgos

2.2.4.5.3 EXPLICACIÓN DETALLADA DE LAS AMENAZAS

En el siguiente apartado se dará una explicación detallada de las amenazas que ponen en peligro el funcionamiento del servicio.

2.2.4.5.3.1 FUGA DE INFORMACIÓN

Esta situación se produciría en el supuesto de que alguien un ajeno a la aplicación fuese capaz de recopilar información de los servicios que se están dentro del grupo de usuarios que se encuentran inscritos al servicio. En función del tipo de información que se filtre, la gravedad de este puede variar notablemente.

Es por ello por lo que, como se ha explicado en el documento principal, la base de datos donde se almacena toda la información de los usuarios está securizada mediante las reglas que proporciona *Firebase Firestore* con el fin de que los usuarios no autenticados no puedan acceder a dicha información.

2.2.4.5.3.2 DATOS DE USUARIO

Los datos que se filtren de los propios usuarios no solo supondrán una vulneración de la privacidad de la que la empresa será enteramente responsable, sino que además supone una ilegalidad puesto que es una falta grave registrada en la LOPD.

2.2.4.5.3.3 CREDENCIALES ASOCIADAS A LOS USUARIOS

Es destacable tener en cuenta el factor humano en la creación de las claves de usuario. Por ello, se advertirá en la medida de lo posible a los usuarios de nuevos registros de la importancia de crear una clave lo suficientemente fuerte. Dicha clave no será almacenada ya que es el proveedor *serverless* a quien corresponde el almacenamiento de las credenciales.

De lo que si se ocupa nuestra aplicación es de la entrega de *API Keys* mediante las cuales poder realizar las consultas de clasificación o filtrado, es por ello por lo que en el supuesto de que el usuario tenga la mínima sospecha de que su *API Key* ha sido sustraída, se le posibilitará la renovación de la clave mediante la aplicación web.

2.2.4.5.3.4 INYECCIÓN JAVASCRIPT/XSS

La inyección de código ejecutable en la aplicación supone una grave vulnerabilidad que puede llevar desde alterar la confidencialidad/integridad de los datos hasta afectar a la disponibilidad del servicio.

Cabe destacar que uno de los motivos por los que se ha elegido React, es que el hecho de que utilice JSX, hace que la prevención en cuanto a este tipo de inyecciones sea más sencilla.

2.2.4.5.3.5 INTRUSIÓN/ATAQUES D/DOS

La prevención de ataques de este tipo es costosa. Dada la naturaleza del servicio y su temprano estado de madurez, se ha pospuesto su prevención hasta una vez alcanzada la madurez de este y se hayan completado los planes de previsión asociados al mismo.

2.2.4.5.3.6 CRASHES DE EJECUCIÓN

Se evitará en mayor medida mediante test automatizados y análisis estático del código en cada build. Se aplicará una metodología de integración continua del proyecto para asegurar su perfecta integración de las partes y se esperan corregir la mayor parte de errores posibles que puedan ir surgiendo en etapa de prueba antes de salir a producción.

Tras realizar el análisis de riesgos, se procederá a explicar la base de datos secundaria del sistema GAIA.

2.2.4.6 ALGOLIA

A continuación, se detallan las principales funcionalidades de GAIA que se soportan en Algolia. Esta base de datos es utilizada para resolver la situación previamente explicada en el apartado del filtrado de contenidos.

Es decir, Algolia es utilizado como un segundo filtro de contenidos, el cual puede funcionar con un margen de error. En caso de que ni UZEI ni la base de datos principal de *Firebase* encuentren ninguna coincidencia, esta segunda base de datos es capaz de analizar dicho contenido con un margen de error con el fin de evitar posibles engaños, las búsquedas reconocen los errores ortográficos.

2.2.4.7 FINALIZACIÓN DE LOS SERVICIOS – ARQUITECTURA FINAL

Una vez finalizado el desarrollo de los servicios, la arquitectura de todo el sistema de clasificación ha quedado de la siguiente forma:

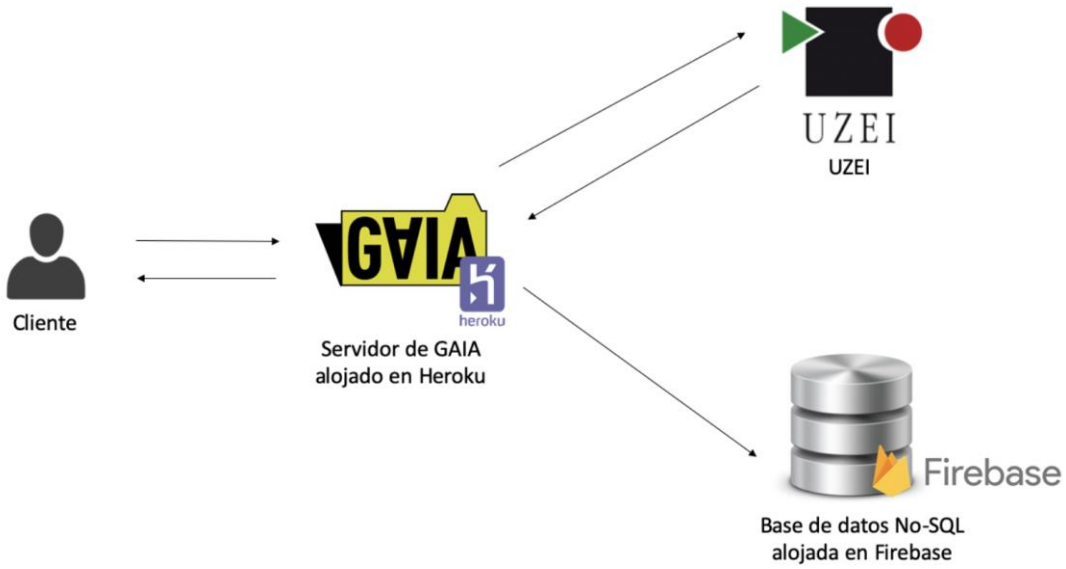

Ilustración 37: Arquitectura final de GAIA (Clasificación)

En la arquitectura mostrada se pueden diferenciar 4 actores principales:

- Cliente
- Servidor de GAIA alojado en *Heroku*
- UZEI
- Base de datos No-SQL alojada en *Firebase*.

Dichos actores toman importancia cada vez que un cliente realiza una consulta al servidor. El procedimiento para seguir por cada consulta de clasificación es el siguiente:

- El cliente le hace una petición POST al *path* de clasificación con su correspondiente *body*, el cual consiste en el texto a clasificar y la *API Key* del usuario, con el fin de poder identificarlo.
- El servidor comprueba…

  - Que el formato en el que se han mandado los parámetros cumple unos requisitos, esta comprobación se realiza mediante diferentes *regular expressions*[26].
  - Que la *API Key* exista en la base de datos.
  - Que el usuario tenga la contratación realizada.
  - Que el límite de uso no se exceda, en caso de que el usuario lo tenga establecido.
- Se hace la petición a UZEI tal y como se ha explicado en el apartado 2.2.4.2.
- Una vez se recibe la respuesta de UZEI, se responde inmediatamente al usuario.
- Posteriormente, se guarda en la base de datos una transacción de la consulta con el fin de tener un control sobre las peticiones que realiza cada usuario. Además, de esta forma el usuario podrá consultar su historial mediante la aplicación web.

El hecho de que se responda al usuario antes de guardar la transacción en la base de datos tiene su lógica ya que, en caso contrario, no se conseguiría más que retrasar la respuesta. Ese retardo es algo a evitar ya que uno de los objetivos es que los servicios sean lo más rápidos posibles.

En cuanto al servicio de filtrado la arquitectura es similar, la única diferencia es que *Algolia* entra en el procedimiento con el fin de darle mayor eficacia al servicio. El esquema de la arquitectura quedaría de la siguiente forma:

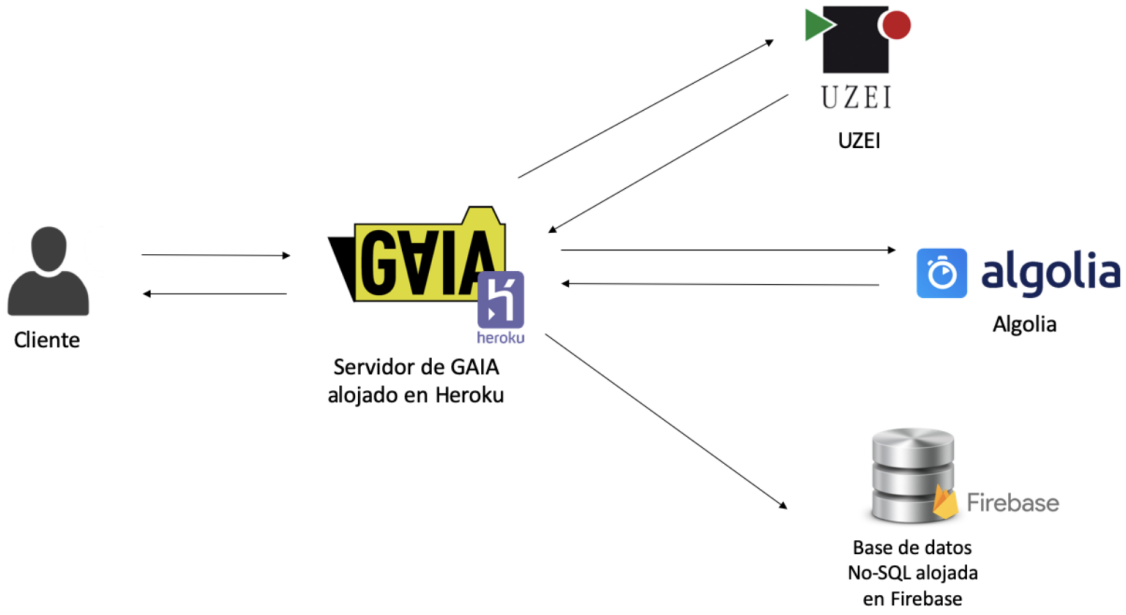

Ilustración 38: Arquitectura final de GAIA (Filtrado)

El procedimiento del servicio es el siguiente:

- Los dos primeros pasos son identicos a los del servicio de clasificación.
- Una vez comprobada la validez del *body* de la petición en el servidor, se envía a UZEI el texto con el fin de que lo lematice.
- Se recibe el contenido lematizado.

[26] Término utilizado para referirse a una secuencia de caracteres que forma un patrón de búsqueda, principalmente utilizada para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones.

- Se comprueba mediante la base de datos alojada en *Firebase* si alguna de las palabras lematizadas por UZEI coincide con las de la base de datos.
- Se reciben las palabras malsonantes encontradas mediante *Firebase*.
- Se comprueba mediante *Algolia* si alguna de las palabras lematizadas coincide. La diferencia con *Firebase* es que esta proporciona más flexibilidad, es decir, dependiendo de la longitud de la palabra lematizada, es posible cambiar una letra o varias de posición con el fin de hacer más estricto (o no) este filtrado.
- Se reciben las palabras malsonantes encontradas mediante *Algolia*.
- Se unen las palabras malsonantes encontradas mediante ambos métodos (excluyendo las repetidas) y se responde al cliente con el listado.
- Por lo motivos enunciados anteriormente, tras la respuesta dada al usuario se guarda la transacción de la consulta en la base de datos.

Tras analizar la arquitectura final tanto del servicio de clasificación como la del filtrado, se procederá a explicar la presentación que se llevó a cabo ante el Ministerio Economía y Empresa.

2.2.5 PRESENTACIÓN MINISTERIO

En cualquier caso, conviene establecer que la fecha límite para asegurar el perfecto funcionamiento de esta primera versión se sitúa en el mes de junio del presente año, por lo que, durante este periodo anterior al plazo de vencimiento, el principal requerimiento y preocupación se centra en la correcta articulación de este.

En la siguiente ilustración se refleja un esquema de los servicios presentados, el cual también está adjunto en el anexo en formato editable por si se desea revisar:

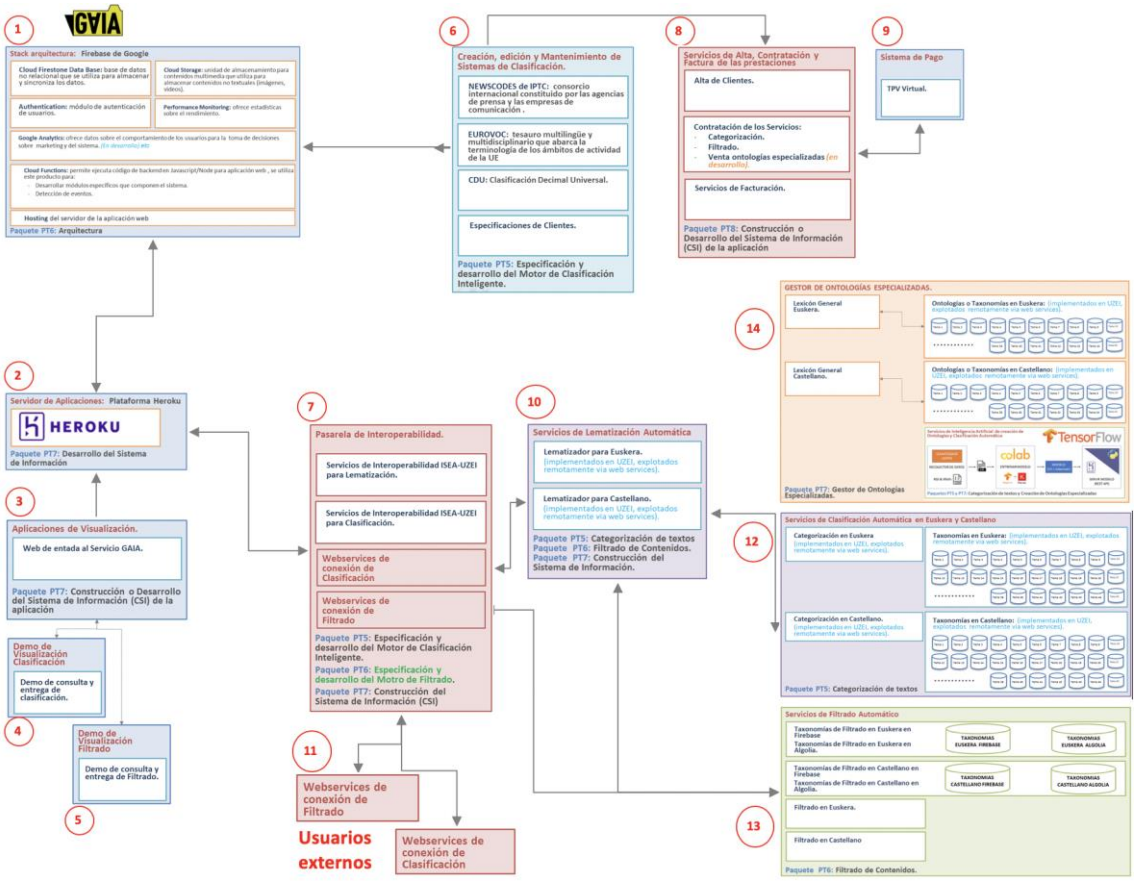

Ilustración 39: Esquema presentado en la presentación ante el Ministerio de Industria

De este modo, los requisitos para la presentación son los siguientes:

- La aplicación debe de cumplir los siguientes requerimientos:
  - Un inicio de sesión mediante el cual el acceso quedase restringido a usuarios.
  - Una interfaz mediante la cual el usuario pueda consultar sus credenciales, contratación y pagos.
  - Una interfaz mediante la cual el usuario pueda realizar una demostración de lo que los servicios le pueden llegar a aportar.
  - Una interfaz mediante la cual un usuario con rol de administrador puede modificar tanto el precio de los servicios, como el contenido temático ofrecido a los clientes.
  - Una interfaz que contenga una tabla interactiva con la cual los usuarios puedan consultar los temas soportados por el servicio.
  - Una interfaz mediante la cual el usuario pueda limitar los servicios, de modo que pueda establecer un límite de uso.
  - Un diseño acorde a lo anteriormente establecido.
  - Una aplicación robusta y rápida.
  - Unos servicios que tarden lo menos posible en recibir la petición y posteriormente enviar la respuesta.

Siguiendo esta línea, se debe establecer que el desarrollo se ha realizado en el siguiente orden:

- En primer lugar, se ha procedido a realizar un inicio de sesión simple mediante el cual el usuario puede realizar la contratación y posteriormente consultarlo.
- En segundo lugar, se han realizado las interfaces con las cuales el administrador puede gestionar tanto los precios como las categorías ofrecidas al cliente.
- Una vez terminadas las principales interfaces de la aplicación, se han desarrollado los servicios que se ofrecerán a los clientes con el fin de clasificar o filtrar sus contenidos.
- Finalmente, tras finalizar el desarrollo de los servicios, se han implementado dos interfaces mediante las cuales el usuario puede comprobar que tipo de respuesta recibiría por parte de los servicios.

El resultado de la primera versión se puede observar en las siguientes ilustraciones:

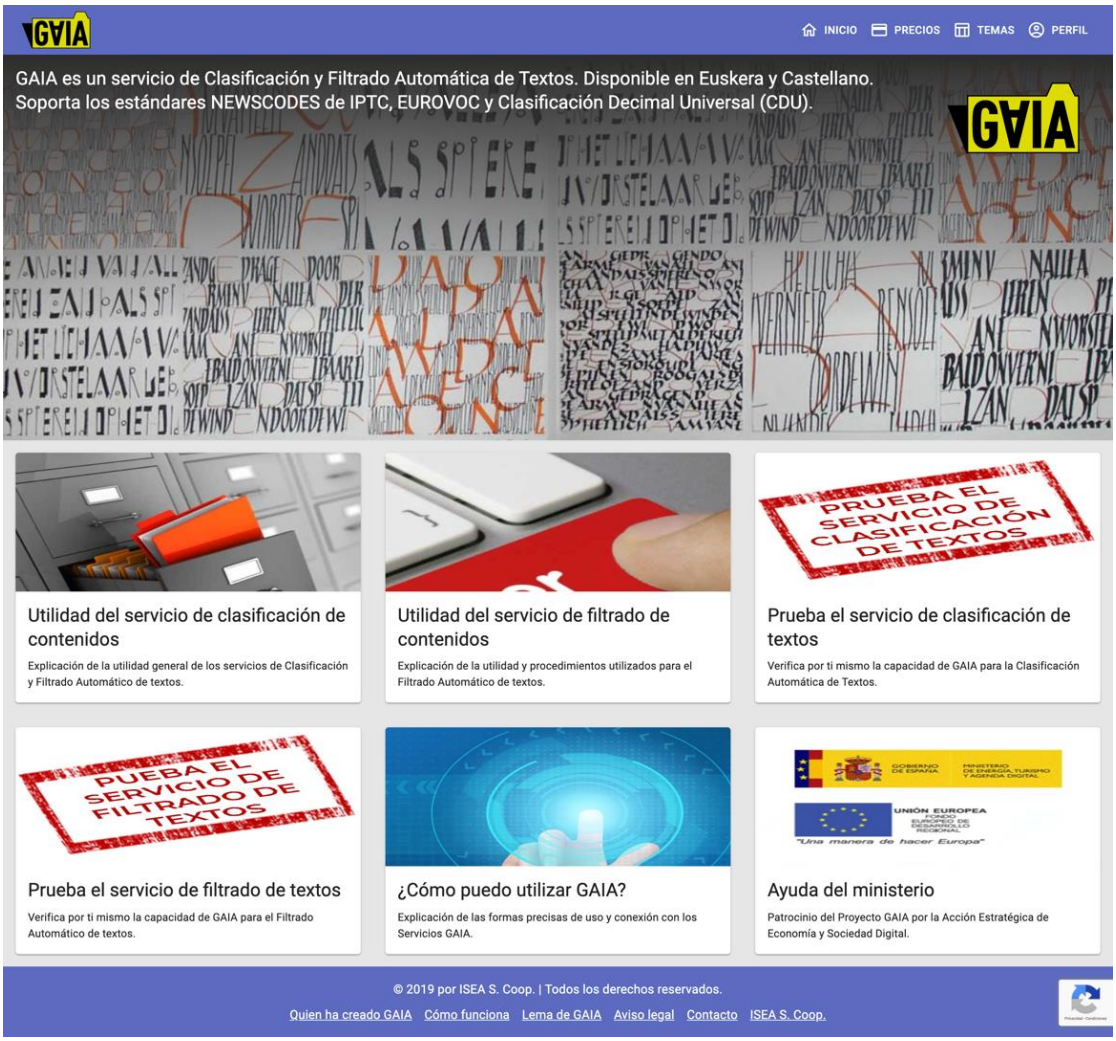

Ilustración 40: Interfaz principal de la aplicación web

Ilustración 41: Interfaz de elección de contratación

Ilustración 42: Interfaz para la gestión de datos del usuario

Ilustración 43: Interfaz para la gestión del contrato de clasificación

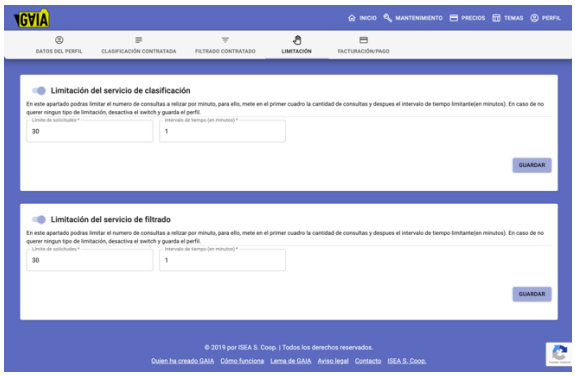

Ilustración 44: Interfaz para limitar el número de consultas

Ilustración 45: Interfaz para la consulta de precios

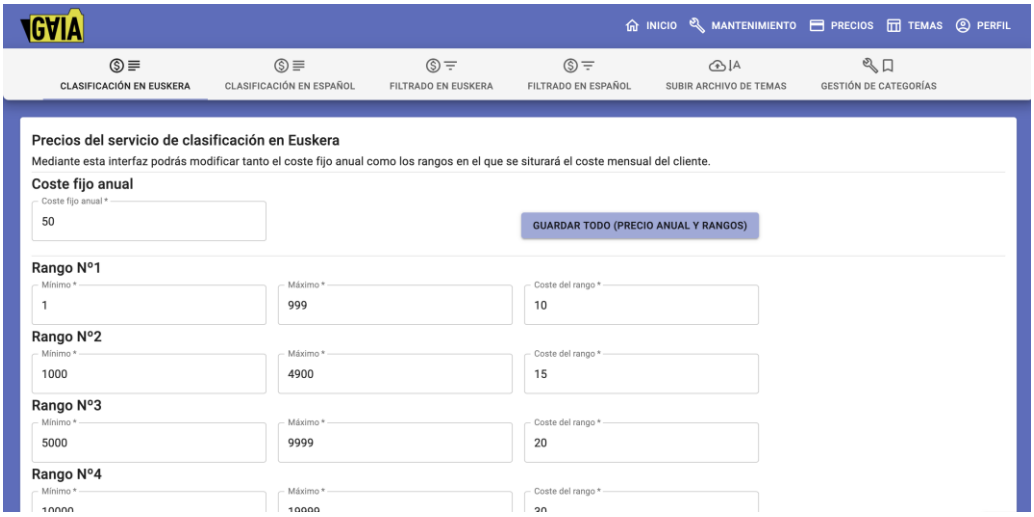

Ilustración 46: Interfaz para modificar los precios de los servicios

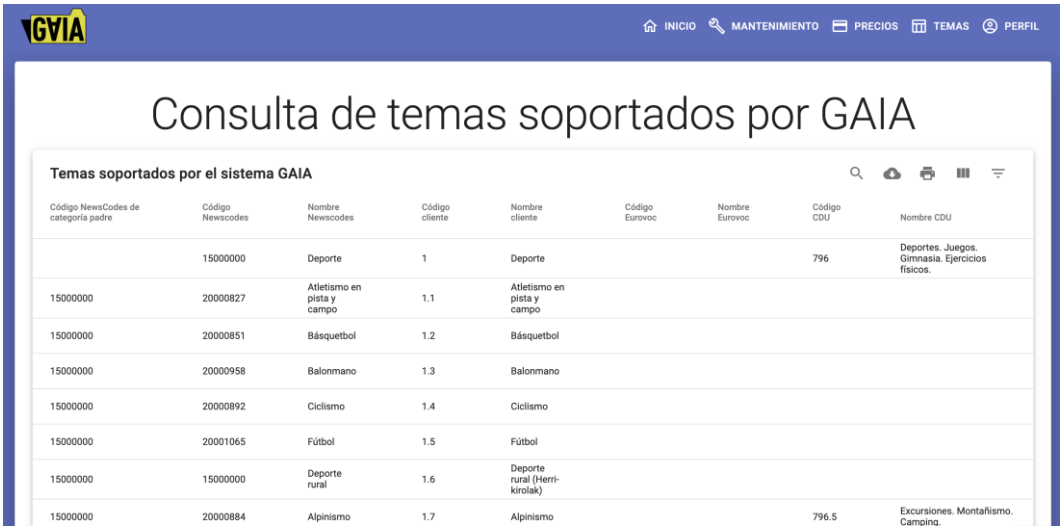

Ilustración 47: Interfaz con tabla interactiva para la consulta de categorías

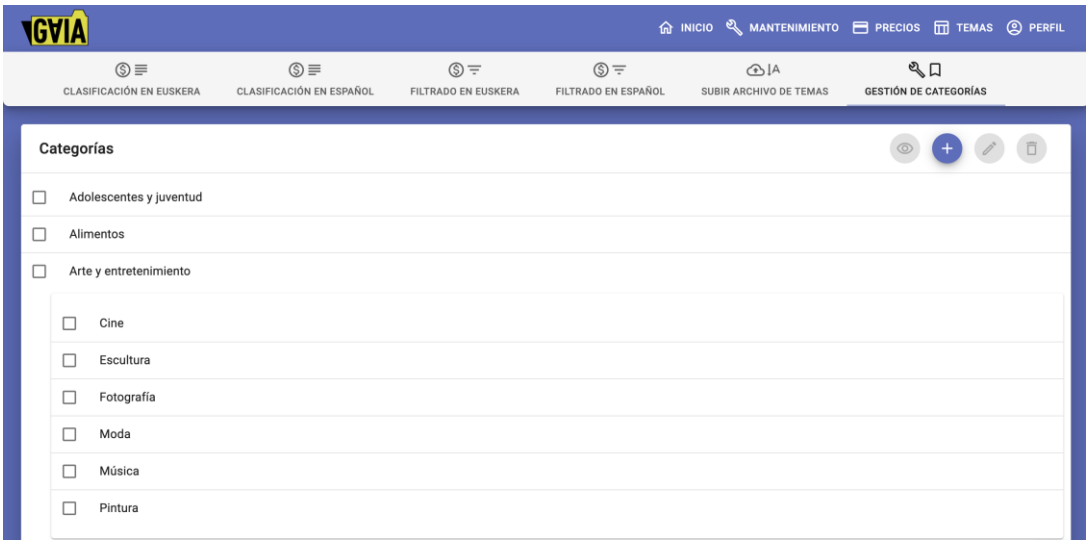

Ilustración 48: Interfaz para la gestión de categorías

Tal y como puede observarse, las principales interfaces han sido realizadas y el funcionamiento principal de la aplicación es correcto. De igual manera, hemos de destacar que se ha efectuado la presentación ante la Secretaría de Estado para la Agenda Digital (Ministerio de Economía y Empresa). Dicha inspección ha sido resuelta a satisfacción del Auditor remitido por el Ministerio.

2.3 DESARROLLOS ADICIONALES

Además de dedicarse al desarrollo del proyecto, se ha compaginado el mismo con otros tipos de tareas como pueden ser el rediseño de una aplicación lanzada por los responsables del Sistema Tentu o la recopilación de artículos para alimentar una red neuronal a desarrollar por la propia organización.

Dichas tareas serán explicadas a lo largo de este apartado.

2.3.1 LANZAMIENTO DE TENTU AL PÚBLICO Y NECESIDAD DE REDISEÑO

Tentu, la revista electrónica personalizada fue lanzada al público del País Vasco en diciembre de 2018. Debido a ello se registraron alrededor de 800 usuarios por lo que el equipo de desarrollo recibió muchísimo *feedback*[27] con posibles mejoras para la aplicación. Una de las cuestiones en la que los usuarios más coincidían era lo confuso que podía llegar a ser la suscripción, además de la navegación a través de la aplicación.

Asimismo, en el *feedback* se hacía mención las muchas interfaces por las que debían navegar con el fin de llegar al principal protagonista de Tentu, el contenido clasificado.

Una vez recibido todo el *feedback* se llegó a la conclusión de que la aplicación debía realizar los siguientes ajustes para ser mucho más amena e intuitiva de cara el usuario:

[27] Término utilizado para referirse a la opinión de los usuarios respecto a una aplicación.

- El número de clics que debe hacer el usuario con el fin de mostrar el contenido debe disminuir. Ya que, si este es alto, la probabilidad de que se equivoque y termine en una interfaz que no desea es alta.
- El número de interfaces por los que el usuario debe navegar para realizar la acción deseada debe ser reducida, este punto tiene relación con el anterior.
- Los contenidos deben de tener más protagonismo, ya que en la versión actual una vez se accede a Tentu, lo primero que el usuario ve es el listado de temáticas disponible y no el contenido, que es la finalidad por la que el usuario realmente inicia sesión en la aplicación.
- Se debería implementar un apartado de "Últimas noticias" con el fin de hacer ver al usuario la cantidad de datos y contenidos que Tentu gestiona.
- El registro/inicio de sesión se debería actualizar, de modo que se pueda acceder con servicios externos como la cuenta de Google, Twitter, Facebook o Outlook. De esta forma, darse de alta en el servicio será cuestión de un par de clics y no de rellenar varios campos, seleccionar temáticas, ciudades, un proceso que puede resultar tedioso para el usuario. Esto puede provocar que finalmente rechace y termine saliendo sin haber completado el registro.
- Además de lo indicado en puntos anteriores, se cree conveniente adaptar Tentu a líneas de diseño actuales con el fin de que sea más atractivo de cara al usuario.
- Asimismo, se debe intentar reducir el número de texto y aumentar la cantidad de imágenes representativas o iconos. El hecho de que haya demasiado texto puede llegar a abrumar al usuario, con su consecuente rechazo hacia la aplicación.
- Otro de los ajustes a destacar es la comprobación de calidad de los *feeds*[28] que alimentan Tentu, ya que existen algunos que apenas aportan contenidos y cuando lo hacen su calidad deja que desear. Esta mejora es a largo plazo ya que la aplicación cuenta con muchas fuentes las cuales van en aumento día a día. Además de revisar los *feeds*, también se añadirán más, ya que, mediante esta comprobación, algunos de los existentes serán eliminados, por lo que se debe encontrar el equilibrio para quedarse con la misma cantidad que al comienzo.

Además de estos ajustes propuestos por los usuarios, el equipo de desarrollo se ha percatado de que la aplicación web y su versión móvil no comparten la misma línea de diseño. Esto es algo relevante, pues compartir una línea de diseño o coherencia artística ofrece al usuario la sensación de que utiliza la misma aplicación. El hecho de compartir línea artística va a provocar que el usuario relacione este tipo de agregadores de contenido con Tentu por lo que podría sentar el precedente en la sociedad.

El hecho de sentar el precedente puede ser algo muy beneficioso para la compañía ya que cuando la sociedad se refiera a este tipo de herramientas se podría referir como Tentu, al igual

[28] Término utilizado para referirse a los documentos con formato RSS, los cuales albergan en su interior noticias o titulares, generalmente con un pequeño resumen del contenido.

que ocurre con el papel de aluminio, el cual es referido como "*Albal*" o los pañuelos de papel, "*Cleenex*", cuando son unas marcas comerciales.

Dejando esto de lado, una vez dispuestos a realizar cambios, el equipo quiere aprovechar la ocasión para implementar la siguiente mejora: Programar ambas aplicaciones con el fin de que estas sean simples consumidores de un *back-end* que proporcione toda la información. Por otro lado, en caso de realizarlo este *back-end* puede ser expuesto y comercializado de alguna forma.

Una vez marcados los puntos a mejorar, el equipo de desarrollo trabajó junto al de diseño dando como resultado, las siguientes mejoras e interfaces.

2.3.1.1 REDISEÑO

Uno de los principales cambios ha sido que con el nuevo diseño el usuario puede utilizar Tentu sin realizar la acción de registrarse. Inscribirse proporciona la posibilidad de personalizar las temáticas o publicar contenidos propios del usuario. De este modo, el usuario entiende para que sirve realmente el producto, ya que interactúa directamente con él.

Se ha querido dar otro enfoque a la primera impresión que se puede llevar el usuario que entre sin saber de qué trata la aplicación, ya que antes, al entrar en Tentu sin tener una inscripción, podía llevar a confusión y a no saber en qué consiste realmente. Como ahora el acceso a Tentu está disponible incluso para usuarios no registrados, en caso de que el cliente quiera informarse sobre cómo funciona, tiene disponible una página completamente nueva a la cual se puede acceder mediante un pequeño botón en el *navbar*.

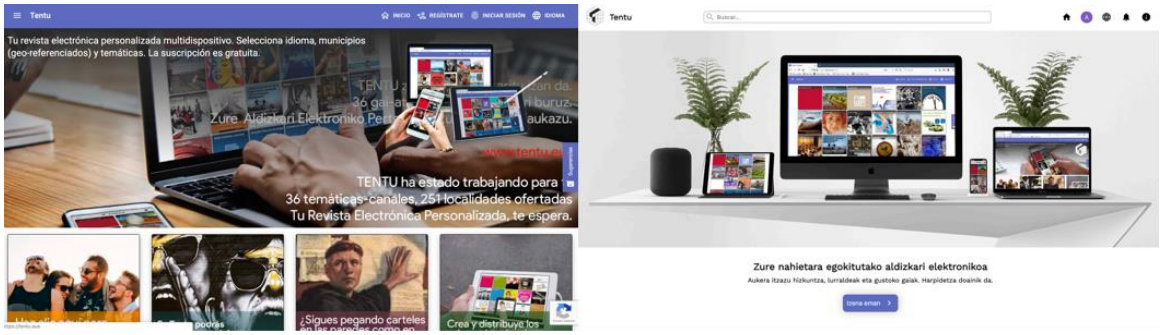

Ilustración 49: Comparativa entre about nuevo frente al antiguo

La acción de registrarse es muchísimo más sencilla, dado que se ha implementado la posibilidad de inscripción mediante servicios externos de autenticación como pueden ser la cuenta de Google, Facebook, Twitter o Outlook. Esto hace que el hecho de registrarse sea más corto ya que se recoge desde el proveedor la información necesaria para el registro.

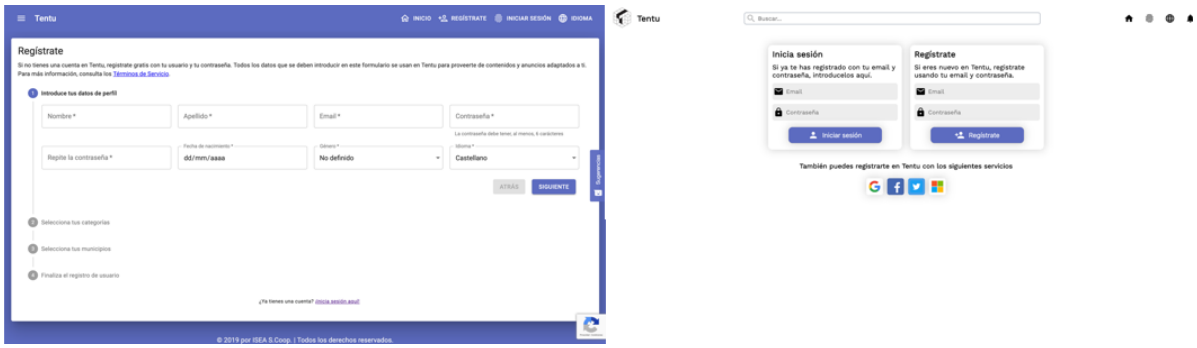

Ilustración 50: Comparativa entre login/registro nuevo frente al antiguo

Una vez realizado el registro es cuando el usuario puede personalizar Tentu desde una nueva interfaz más simplificada, pero sin perder ningún tipo de funcionalidad, es más, ahora se ha implementado el guardado en tiempo real, es decir el usuario no tiene que pulsar el botón para accionar el guardado.

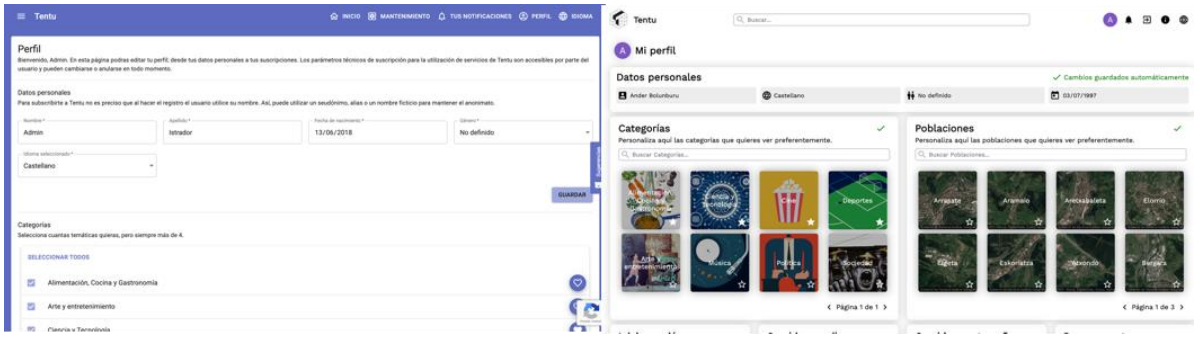

Ilustración 51: Comparativa de la interfaz del perfil nuevo frente al antiguo

Además, ahora es más sencillo navegar a través de la aplicación sin llevar a ningún tipo de confusión. El motivo de esto es que ahora para seleccionar la temática únicamente hay que realizar un clic cuando con el anterior diseño la cantidad de clics a realizar era mayor.

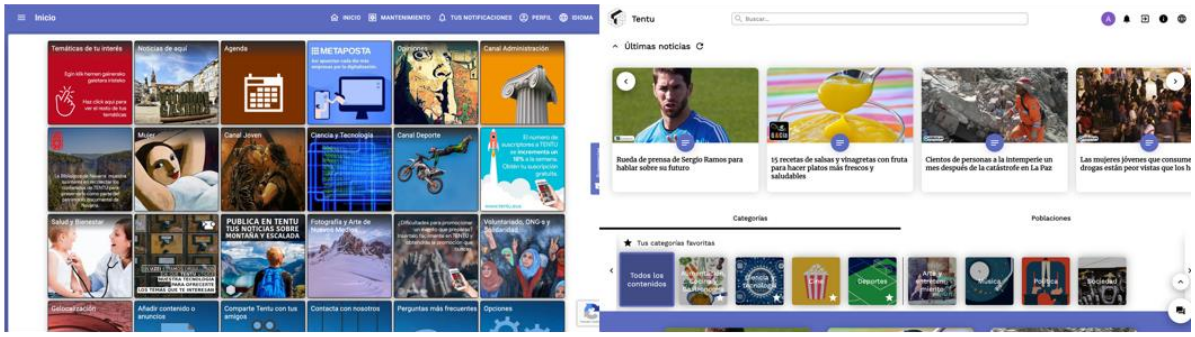

Ilustración 52: Comparativa de la interfaz principal nueva frente a la antigua

Es destacable que ahora el hecho de poder intercambiar los contenidos clasificados por temáticas o por poblaciones es más intuitivo, ya que en este momento es totalmente evidente y antes esa funcionalidad estaba un tanto oculta. Por otro lado, en caso de no tener ninguna población seleccionada como favorita, la aplicación se encarga de organizarlas en base a la ubicación del usuario en función de la localización IP.

Cabe referir el lavado de cara general que se la ha dado a la aplicación utilizando un diseño minimalista. Dando a la herramienta un enfoque atractivo de cara al usuario medio de hoy en día, ya que el abuso de texto puede llegar a causar rechazo en el usuario. Se ha tratado de utilizar el mínimo texto posible y unos iconos muy representativos con el fin de que el usuario intuya la funcionalidad del botón/enlace que lo acompaña.

Además, una de las funciones clave de la aplicación como es Tentu Talks[29], es ahora muchísimo más evidente e intuitivo, ya que en cuanto el usuario haga un mínimo de scroll[30], el botón aparece de forma que es más visible para el usuario.

Tras hablar del rediseño de la aplicación, en el siguiente apartado se procederá a hablar sobre la recopilación de artículos para una red neuronal.

2.3.2 RECOPILACIÓN DE ARTÍCULOS PARA RED NEURONAL

En el momento del lanzamiento de Tentu al mercado, el equipo de desarrollo se dio cuenta de que la clasificación de contenidos realizada por parte de UZEI no era la mejor, debido a que clasificaba artículos en temáticas que poco o nada tenían que ver con el contenido.

Es por ello por lo que el equipo de desarrollo planteó al responsable el desarrollo interno de una pequeña red neuronal experimental, la cual fuera capaz de tratar menos temáticas que UZEI, pero clasificase de forma más efectiva. Dependiendo del resultado, el desarrollo de esta red neuronal podría avanzar o no.

Además, como se ha indicado anteriormente, es posible la introducción de esta red neuronal a modo de filtro, aumentando así, la eficacia de la clasificación de contenidos. Ello se debe a que esta red neuronal solamente se encargaría de clasificarlo en temáticas generalistas. Una vez sabida la temática se consultaría con el clasificador de UZEI, dando como resultado una temática más concreta.

Con el propósito de no complicar en exceso los desarrollos de la inteligencia artificial se seleccionaron inicialmente temáticas cuyo carácter no fuera muy específico; como podrían ser el futbol, baloncesto o balonmano. Así, se optó por las siguientes temáticas:

- Deportes
- Tecnología
- Ciencia
- Moda
- Alimentación
- Cine
- Música
- Coches
- Decoración

[29] Tentu Talks es un sistema de chat incorporado en Tentu mediante el cual el usuario puede realizar comentarios relacionados con el contenido o noticia.
[30] El *scroll* es un término inglés que se utiliza para hablar del desplazamiento de los contenidos 2D que se muestran en la ventana de un navegador. Es ese sencillo texto que el usuario realiza cuando desliza el dedo por la pantalla o utiliza la rueda del ratón para subir o bajar mientras observa los contenidos de una página.

Uno de los primeros pasos para desarrollar la red neuronal, es la recopilación de textos o artículos periodísticos relacionados con cada temática. A tales efectos, se han importado artículos periodísticos correspondientes a cada temática en una base de datos. Dichos artículos se han importado desde las redes RSS que albergan los principales medios de nuestro país como pueden ser el periódico deportivo MARCA (MARCA, s.f.) o el medio tecnológico Xataka (Webedia, s.f.) Para ello se ha hecho uso de un *crawler*, o rastreador, cuya principal función es la de analizar los documentos de un sitio web.

Una vez analizados los documentos, se extrae el cuerpo del articulo y se excluyen posibles palabras clave que pueden confundir a la red neuronal, como pueden ser los pies de cada artículo, los cuales posibilitan la compartición a conocidos mediante redes sociales. Este tipo de gadgets son cada vez más frecuentes en este tipo de páginas, por lo que es conveniente eliminarlos. Para ello se ha hecho uso de *xPath*, mediante el cual se ha obtenido únicamente contenido significante para la red neuronal. De igual manera, se ha intentado que dicho contenido albergue palabras clave como pueden ser *Messi* en cuanto a temas deportivos o *Android* para los tecnológicos.

Con el fin de no tener contenidos duplicados, cada vez que se lanza el importador de contenidos, se comprueba que el contenido correspondiente se encuentre en la base de datos. En caso de que no sea así, se procede a coger el texto significativo del articulo e importarlo. Es por ello por lo que con el paso del tiempo habrá cada vez más contenidos para alimentar la red neuronal.

En el caso de que la red neuronal haya sido conducida correctamente inducirá que la red neuronal tenga una gran variedad de palabras clave que le haga decantarse por la clasificación según una temática u otra. Lo cual hará que la probabilidad de acierto aumente.

Después de explicar los desarrollos adicionales, se procederá a analizar los resultados obtenidos.

3 RESULTADOS

Tras describir detalladamente el desarrollo del proyecto, en este apartado se hará un análisis de los resultados obtenidos.

Con el fin de contextualizar dichos resultados se presenta la estructura actual de GAIA:

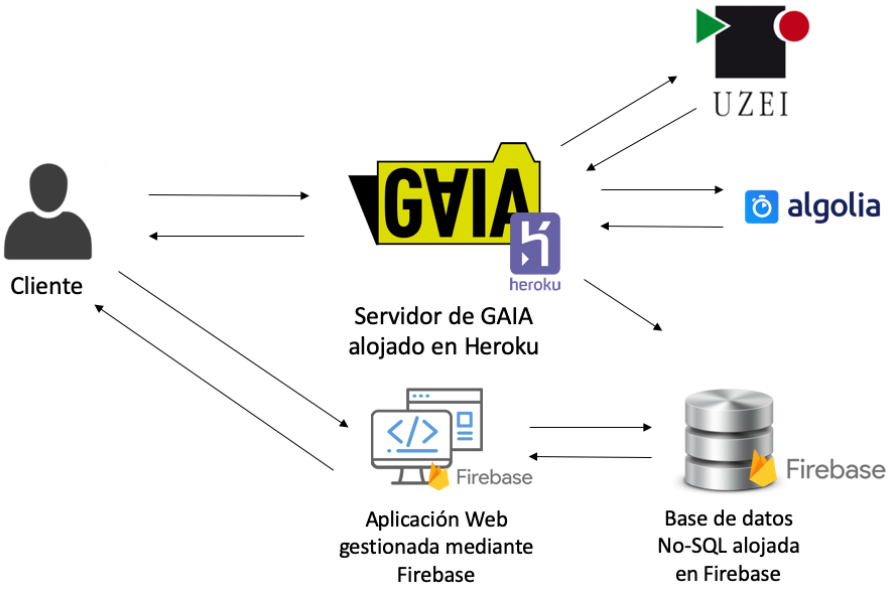

Ilustración 53: Arquitectura de GAIA (Completa)

Como se puede observar, existe una investigación, formación y desarrollo tras el diseño para dar solución al problema inicial. Lo cual ha conllevado obtener conocimientos sobre diferentes ámbitos como pueden ser:

- Desarrollo de código en el lenguaje *Javascript*.
- Gestión de dependencias y compilaciones mediante *NPM*.
- Gestión y control de la plataforma *Firebase*.
- Uso y desarrollo de aplicaciones *React*.
- Uso y desarrollo de funcionalidades *cloud* mediante *Cloud Functions*.
- Gestión de testeos de calidad con *Jest* y *Enzyme*.
- Gestión de *builds* automáticas con *Gitlab CI*.
- Despliegue de un *back-end* alternativo mediante *Heroku*.
- Extracción de datos de una página web mediante *crawlers* y selección de contenido mediante *xPath*.

La principal meta del presente proyecto es la creación, tanto de unos servicios web, como de una aplicación que gestione dichos servicios para el cliente, así como para el administrador. A tales efectos, se ha analizado el mercado con el fin de encontrar la solución óptima para la empresa.

De esta manera, se ha desarrollado una aplicación web mediante *React*, la cual funciona de forma fluida y estable. Dicha aplicación está pensada con el fin de que, en caso de que en un

futuro se desee modificar algún aspecto como pueden ser los precios o la base de datos temática, sea sencillo.

Esta aplicación está alojada en una plataforma *serverless* (Firebase), que deja todo el mantenimiento en manos del proveedor. Este último aspecto es bastante innovador, ya que en caso de que la base de usuarios aumente, el proveedor, en este caso Google, se encarga de escalar el servidor para que sea capaz de gestionar todos ellos.

Además, el hecho de que este alojado en este tipo de plataformas proporciona al desarrollador mucha comodidad ya que gestiona muchos aspectos como el inicio de sesión o el ingreso de datos en *Firebase Firestore*. Esto reduce la cantidad de código de forma considerable, aspecto a agradecer tanto por el desarrollador actual como al que pueda venir en un futuro, ya que el código será mucho más sencillo de comprender y, por lo tanto, se podrá integrar en el desarrollo rápidamente.

Por otro lado, se ha tenido en cuenta desde el comienzo del desarrollo el objetivo de realizar software de calidad, lo cual se ha conseguido ya que se han logrado implementar diferentes testeos. Estos, son ejecutados cada vez que una nueva actualización del código es subida a la herramienta, lo cual verifica la robustez del avance aportado.

Con el fin de dar comodidad al cliente, se ha publicado un post en *Swagger* para que pueda conocer la variedad de respuestas posibles por parte de servicios.

Asimismo, la realidad es que un gran número de los clientes potenciales de este tipo de producto son desarrolladores que, potencialmente, quieran hacer uso de estos servicios en sus aplicaciones. A tales efectos, se ha publicado un paquete en la herramienta de gestión de dependencias *NPM* con el fin de facilitar el uso de los servicios. Mediante esta publicación el desarrollador solamente debe introducir la *API Key* a la hora de importar la dependencia y mediante una única línea de código será capaz de obtener resultado de los servicios.

Además, se ha colaborado en el desarrollo del sistema Tentu, proporcionando ideas y mejoras al rediseño de la aplicación. Ello ha concluido a una aplicación tanto más intuitiva de cara al usuario como adecuada a tiempos actuales en cuanto a diseño se refiere.

También se ha llevado a cabo una recopilación de artículos correspondientes a diferentes temáticas acordadas en la empresa, con el fin de alimentar una red neuronal en desarrollo, la cual tiene como objetivo mejorar la clasificación realizada por una empresa externa.

4 MEMORIA ECONÓMICA DEL PROYECTO

En el siguiente apartado se detallará el coste de las herramientas utilizadas para la realización del proyecto.

4.1 COSTE INSTRUMENTAL Y EQUIPAMIENTO

En este apartado se detalla el coste de los equipos y elementos utilizados para el desarrollo del proyecto.

Elemento	Cantidad	Coste
MacBook Pro (13 inch with Touch Bar, 2018)	1	1600€
Otros accesorios para el ordenador (HUB, adaptadores USB)	1	50€
Otros accesorios como cuadernos, bolígrafos...	1	20€

4.2 COSTE DE PERSONAL

En este apartado se detallan los costes del personal implicado en el proyecto.

Personal	Coste por hora	Horas dedicadas	Coste
Ander Bolumburu	4,36€	1050 horas	4578€

4.3 COSTE TOTAL

Teniendo en cuenta los anteriores apartados, el coste total de este proyecto ha sido de 6258€.

5 CONCLUSIONES Y LÍNEAS FUTURAS

En este apartado, se comentarán las conclusiones que se pueden deducir del proyecto.

5.1 CONCLUSIONES TÉCNICAS

Teniendo en cuenta el apartado sobre los objetivos del proyecto y sobre el estado y planificación de este, se puede concluir que el proyecto se ha realizado correctamente y el desarrollo de tareas ha sido el correcto, habiéndose cumplido los objetivos establecidos en la planificación inicial, la cual figura en el apartado 1.4 de este mismo documento.

En cuanto a las competencias a desarrollar por el alumno, se estima que se han cumplido correctamente:

- **CTFG01**: Redacta correctamente el informe del trabajo final de grado y presenta y defiende con claridad el resultado de dicho trabajo.
    - La redacción del presente informe, junto con la presentación del proyecto como defensa de este.

- **CTFG02**: El alumno/la alumna se ha integrado óptimamente en la empresa, colaborando con el resto de las personas de su entorno, y ha demostrado alto nivel de resolución y autonomía en el desarrollo del Trabajo Fin de Grado.
    - La integración en la empresa se ha llevado a cabo satisfactoriamente, creando tanto relaciones profesionales como personales. El grupo de trabajo ha sido capaz de desarrollar el producto con el fin de presentar una solución atractiva para el cliente.

- **G3I302**: Conocimiento y aplicación de las características, funcionalidades y estructura de los Sistemas Distribuidos basados en web.

    - Se ha diseñado desde cero una estructura compuesta por tanto unos servicios como una aplicación web, todos los cuales están alojados en diferentes plataformas, si bien ello no impide el correcto funcionamiento conjunto.

- **G3I306**: Capacidad de integrar Soluciones TIC y procesos empresariales participando activamente en la especificación, diseño, implementación y mantenimiento de los sistemas de información y comunicación teniendo en cuenta la normativa la regulación de la informática en los ámbitos nacional, europeo e internacional.

    - Se ha participado activamente en la especificación, diseño, implementación tanto de los servicios como de la aplicación web, con el fin de que sea lo más cómodo y atractivo posible de cara al usuario final.

- **T1IT08**: Conocimiento de las materias básicas y tecnologías, que capaciten el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como las que les doten de una gran versatilidad para adaptarse a nuevas situaciones.

    - Partiendo de un desconocimiento inicial total, se ha aprendido a desarrollar React + Redux + Sagas. React es una de las librerías más utilizadas hoy en día y ello aporta un gran valor al desarrollador, dado que proporciona una gran versatilidad a la hora

de crear diferentes aplicaciones web. Además de alojar la aplicación mediante *Firebase*, la cual es una plataforma *serverless* que cada vez es más utilizada por los desarrolladores. También se ha aprendido a desarrollar servicios web mediante *Express.js* y alojarlos en un proveedor *back-end* como es *Heroku*.

5.2 CONCLUSIONES METODOLÓGICAS

Dentro de los métodos empleados, se entiende que es el correcto, dado el buen estado que presenta el proyecto y que el tiempo estimado ha sido el suficiente para llevar a cabo todas las tareas planteadas.

Además, el hecho de que el equipo de desarrollo tenga experiencia previa en el desarrollo de *React* ha agilizado el desarrollo del proyecto, permitiendo avanzar en otras áreas como pueden ser el desarrollo de un modelo capaz de clasificar diferentes temáticas o el rediseño de una de las aplicaciones de la organización como es Tentu.

5.3 CONCLUSIONES PERSONALES

El comienzo del proyecto resultó ser un poco complicado ya que una de las principales tareas a desempeñar era analizar el mercado con el fin de lograr la mejor solución para la empresa. Este análisis al principio fue difícil ya que mis conocimientos sobre el desarrollo web estaban limitados a los recibidos en la Universidad. Habiendo profundizado y actualizado mis conocimientos sobre la oferta de herramientas, he llegado a la conclusión de que están un tanto obsoletas respecto a las actuales. Debido a esto fue difícil decantarse por una herramienta u otra. Finalmente, se decidió llevar a cabo el desarrollo mediante *React*, dicha decisión está ampliamente argumentada en el apartado 2.1.1 de este documento.

No hay que restar importancia a que el desarrollo realizado en *React*, implica comprender conceptos, que al principio pueden resultar complejos como Redux y Sagas, por lo que ha sido necesario hacer frente a conceptos totalmente desconocidos anteriormente. Todo esto ha hecho que la curva de aprendizaje sea un poco más larga de lo habitual. Pero ha sido soportable gracias al apoyo recibido por el equipo de desarrollo de la organización.

Desde un principio se ha atendido a los consejos del equipo en cuanto a la organización de la estructura de la aplicación se refería. Si en un futuro fuera preciso añadir una nueva funcionalidad, ello no resultará complicado. Además, una buena organización de la aplicación conduce a una mejor comprensión por parte de otros futuros desarrolladores.

A pesar de las dificultades de la comprensión de las nuevas tecnologías aplicadas en el proyecto, cabe destacar que, aunque las practicas tuvieron un inicio lento, posteriormente el progreso fue notorio, acelerándose los ciclos de desarrollo.

En cuanto a la experiencia trabajando, he de reconocer que los conocimientos que he adquirido durante este tiempo han sido, globalmente, coincidentes con los que he desarrollado durante el grado. Si bien, obviamente, en la Universidad se trabajar de forma tutelada, mientras que en un entorno real resulta prioritaria la efectividad y la conclusión de los objetivos.

Complementariamente, el proyecto me ha permitido investigar acerca de diferentes herramientas con las que poder crear servicios web, dando como resultado el descubrimiento personal de Express.js; una herramienta que facilita el desarrollo de los servicios respecto al conocimiento previo adquirido en la Universidad.

Por otra parte, destacaría el conocimiento adquirido en el desarrollo de aplicaciones alojadas en *Firebase* de Google. Esta clase de plataformas, en las que el desarrollador se olvida de mantener un servidor, además de resultar muy útiles, están en auge y creo que el hecho de conocerlas aportará mucho valor al desarrollador del futuro.

Finalmente, las practicas han supuesto una gran fuente de conocimiento, ya que antes de comenzarlas apenas tenía conocimiento sobre las herramientas más utilizadas en el desarrollo de aplicaciones web. Ahora, una vez sumergido en el desarrollo, no solo he descubierto la gran variedad de herramientas existentes en el mercado, si no que he aprendido una en profundidad. No he llegado a un dominio absoluto, pero si dispongo de un conocimiento suficiente para desenvolverme con el fin de poder crear aplicaciones web como la de GAIA.

5.4 NUEVO PRODUCTO: CONJUNTO DE SERVICIOS + APLICACIÓN WEB PARA GESTIONARLOS

Mediante las prácticas y los conocimientos obtenidos por ellas se ha desarrollado un nuevo producto para la empresa, dicho producto consiste en un conjunto de servicios y una aplicación web capaz de gestionarlos tanto por parte del usuario como del administrador. Este desarrollo ha hecho necesario el aprendizaje de nuevas tecnologías que no se han desarrollado durante el grado en la Universidad, como la tecnología *React* o el desarrollo mediante *Firebase*.

Bajo mi punto de vista, estos servicios, además de ser muy cómodos de utilizar ya que apenas hay que configurar nada, pueden aportar un valor añadido a revistas o blogs que quieran clasificar sus contenidos en base a una temática.

5.5 LÍNEAS FUTURAS

Como líneas futuras de este proyecto, inicialmente se plantea atraer al mayor número de clientes posibles al uso de los servicios.

En cuanto a los servicios en sí, se debe mejorar la categorización de los contenidos por parte de UZEI, que en colaboración con el equipo de desarrollo logrará mejorar estos porcentajes.

Asimismo, teniendo en cuenta que la aplicación web se dedica a la promoción y gestión de servicios, se podría estudiar la posibilidad de utilizar la aplicación como *marketplace*[31] de otros servicios totalmente diferentes al desarrollado durante el proyecto.

Por otro lado, en cuanto a la aplicación Tentu, sería interesante valorar la división de las funcionalidades que ofrece en diferentes aplicaciones. Esto es debido a que gran parte de los

[31] Los marketplace son aquellos que hacen que la interacción entre clientes y vendedores sea mucho más comoda y sencilla, ya que sirven como “escaparate” en el que se muestran productos y servicios.

usuarios únicamente van a utilizar Tentu para su principal funcionalidad, la cual es leer contenidos.

Es por ello que, con el fin de evitar posibles confusiones, resultaría beneficioso para la aplicación dividir la funcionalidad de publicar contenidos o la contratación del servicio Pandora, en otra aplicación adjunta. El mismo tratamiento se le daría a la acción de administrar la revista, el principal cometido de este rol consiste en la gestión de fuentes, usuarios, contenidos, etc.

Además, se podrían externalizar todas las funciones del *back-end* de la aplicación, con el fin de que tanto la aplicación web como móvil sean simples consumidores de este. Asimismo, se podría estudiar la posibilidad de revelar dicho *back-end* con el fin de cobrar por su uso. El hecho de expornerlo a los clientes puede resultar atrayente a otro sector totalmente diferente al target principal del producto. Lo cual resulta muy interesante ya que ampliaría la variedad de clientes del producto, como pueden ser bibliotecas que deseen almacenar el contenido clasificado como patrimonio.

5.6 POSIBLE DESARROLLO DE NUEVA EMPRESA

Resultaría factible, dado que ISEA se ha adentrado en el desarrollo de software como puede ser la revista electrónica personalizada Tentu o en los propios servicios GAIA desarrollados a través de este proyecto. Complementariamente, además de la red neuronal en desarrollo, se podría considerar la creación de una nueva empresa de desarrollo de software, abstraída del ámbito innovador en el que se mueve ISEA.

Esto posibilitaría la atracción de nuevos clientes y el completo enfoque de la nueva empresa a estos productos.

6 VALORACIÓN PERSONAL

La realización del proyecto en ISEA S. Coop. ha sido muy satisfactoria ya que he aprendido a trabajar en una empresa con un equipo reducido para el desarrollo. Algo no muy habitual ya que normalmente las empresas dedicadas al desarrollo de software cuentan con grupos grandes y abarcan el desarrollo de varios productos simultáneamente.

En mi caso, he estado trabajando en esta empresa desde septiembre de 2018 y durante este tiempo he aprendido a desarrollar software desde otro punto de vista.

Gestionar las *builds* automáticas, implementar testeos de calidad u otros elementos de la ingeniería del software, eran procesos desconocidos para mí. Este proyecto me ha permitido ampliar mis conocimientos sobre todo ello e implementarlo en el desarrollo, aportando así código de mayor calidad.

Llegar a dominar una herramienta completamente nueva de desarrollo como *React* es un gran reto. Han existido momentos de frustración debido al desconocimiento acerca de la herramienta, pero al final por medio de la experiencia ha sido posible llevar a cabo el desarrollo.

Además, el hecho de haber aprendido a desarrollar en una plataforma *serverless* supone un plus importante a mis conocimientos. Ya que además de ser algo totalmente innovador, debido a que el desarrollador se olvida completamente del mantenimiento del servidor, es una tendencia en auge. Diferentes proveedores como Amazon o Google están invirtiendo auténticas millonadas en este tipo de plataformas, con el fin de mejorarlas y aportar un valor diferencial a los clientes. Esta competitividad entre ellas lo único que hace es beneficiar al cliente, en este caso nosotros, los desarrolladores.

Por parte de la Universidad, mi tutor Félix Larrinaga a quien agradezco su ayuda, siempre atento y dispuesto a solventar las posibles dudas que pudiera tener.

Finalmente, en mi caso particular, esta ha sido mi primera experiencia laboral como desarrollador. Algo de lo que de alguna forma me arrepiento, ya que siento que debería haber aprovechado la oportunidad que brinda la universidad, especialmente MONDRAGON Unibertsitatea, y haber trabajado desde el primer momento que la universidad ofrece la posibilidad. Me he dado cuenta de que el hecho de trabajar aporta un gran conocimiento además del adquirido en las clases del grado y no haber trabajado previamente me entristece, pues estimo que, de haber trabajado, mi conocimiento sería mayor. Además del beneficio económico que estas prácticas aportan, algo muy para tener en cuenta, ya que en la adolescencia ese pequeño salario ayuda a tener un tiempo de ocio mejor.

En definitiva, estimo que la realización del proyecto en una empresa ha sido una experiencia interesante y me ha aportado un gran conocimiento. En efecto, gracias a este proyecto he desarrollado nuevas capacidades que, dentro de proyectos llevados a cabo en la universidad, como los POPBL, no hubiese sido capaz de desarrollar.

7 ANEXOS

En este capítulo se incluirán debidamente identificados aquellos que aporten un valor añadido a la memoria para su correcta comprensión.

- Enlace al paquete publicado en NPM.
- Enlace a la documentación publicada mediante Swagger.
- Enlace con información acerca de UZEI.
- *Mockups* realizados mediante Adobe XD.
- Enlace para visualizar los *mockups* online.
- Diagrama Gantt en detalle.
- Documento con información sobre Tentu.
- Diagrama presentado al ministerio en formato editable.
- Documento con temáticas seleccionadas.
- Excel con las temáticas a desarrollar y correspondencias en otros estándares.
- Ejemplo de fichero delimitado con comas, apto para subir a la aplicación web.

8 BIBLIOGRAFÍA

Adobe. (2019). *Adobe XD Product*. Obtenido de Adobe XD: https://www.adobe.com/es/products/xd.html

Algolia. (2019). *Algolia*. Obtenido de Algolia: https://www.algolia.com/

Amazon. (2014). *Amazon Web Services*. Obtenido de https://aws.amazon.com/es/

Chackray. (27 de Enero de 2017). Obtenido de Chakray.com: https://www.chakray.com/es/swagger-y-swagger-ui-por-que-es-imprescindible-para-tus-apis/

Facebook. (2019). *React*. Obtenido de React.js Webpage: https://reactjs.org/

Facebook. (s.f.). *ReactNative*. Obtenido de https://facebook.github.io/react-native/

Frost, B. (2019). *Atomic Design*. Obtenido de http://bradfrost.com/blog/post/atomic-web-design/

Gantt, T. (2019). *Team Gantt*. Obtenido de Team Gantt: https://www.teamgantt.com

Google. (s.f.). Obtenido de Material Design: https://material.io/design/

Google. (2017). *Angular*. Obtenido de Angular.io Webpage: https://angular.io/

Google. (2018). *Firebase Documentation*. Obtenido de https://firebase.google.com/docs/?hl=es-419

Google. (2018). *Firebase Functions Documentation*. Obtenido de Firebase Web Application: https://firebase.google.com/docs/functions

Heroku. (2018). *Heroku*. Obtenido de Heroku Web: https://www.heroku.com/

i18Next. (2019). Retrieved from i18next Webpage: https://www.i18next.com/

IBM. (2014). *OpenWhisk*. Obtenido de https://openwhisk.apache.org/

ISEA S. Coop. (2018). *Tentu, tu revista electrónica personalizada*. Obtenido de Tentu.eus: https://tentu.eus/

Liberal, I. (17 de 8 de 2017). *Medium*. Obtenido de Medium.com: https://medium.com/biko2/serverless-en-aws-y-firebase-e82f3d6fca21

MARCA. (s.f.). *MARCA, diario deportivo*. Obtenido de https://www.marca.com/

Microsoft. (2010). *Azure*. Obtenido de https://azure.microsoft.com/es-es/

Mocha. (2019). *MochaJs*. Obtenido de MochaJs Webpage: https://mochajs.org/

NativeScript. (s.f.). Obtenido de https://www.nativescript.org/

NPM. (2019). *NPM*. Obtenido de NPM: https://www.npmjs.com/

Postman, I. (s.f.). Obtenido de Postman Webpage: https://www.getpostman.com/

ReactRouter. (s.f.). *React Router Guide*. Obtenido de https://reacttraining.com/react-router/web/guides/quick-start

Redux. (2018). Obtenido de https://es.redux.js.org/

RxJS. (s.f.). Obtenido de https://rxjs-dev.firebaseapp.com/

Schae, J. (2018 de 3 de 31). Obtenido de freeCodeCamp: https://www.freecodecamp.org/news/a-real-world-comparison-of-front-end-frameworks-with-benchmarks-2018-update-e5760fb4a962/

StackOverflow. (2018). *StackOverflow*. Retrieved from https://insights.stackoverflow.com/survey/2018/#technology

Swagger.io. (2019). *Swagger*. Obtenido de Swagger.io Webpage: https://swagger.io/

TypeScript. (s.f.). Obtenido de https://www.typescriptlang.org/

Webedia. (s.f.). Obtenido de Xataka: https://www.xataka.com/

WebPack.js. (s.f.). Obtenido de https://webpack.js.org/