



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**JIZT**

**Generación de resúmenes abstractivos en  
la nube mediante Inteligencia Artificial**



Presentado por Diego Miguel Lozano  
en Universidad de Burgos — 11 de febrero de 2021  
Tutores: Dr. Carlos López Nozal y  
Dr. José Francisco Díez Pastor



---

# Índice general

---

<b>Índice general</b>	<b>I</b>
<b>Índice de figuras</b>	<b>III</b>
<b>Índice de tablas</b>	<b>IV</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	7
<b>Apéndice B Especificación de Requisitos</b>	<b>13</b>
B.1. Introducción . . . . .	13
B.2. Descripción global . . . . .	15
B.3. Catalogo de requisitos . . . . .	16
B.4. Especificación de requisitos . . . . .	19
<b>Apéndice C Especificación de diseño</b>	<b>25</b>
C.1. Introducción . . . . .	25
C.2. Elección del nombre del proyecto . . . . .	25
C.3. Diseño de datos . . . . .	27
C.4. Diseño procedimental . . . . .	34
C.5. Diseño arquitectónico . . . . .	34
C.6. Diseño de interfaces . . . . .	38
<b>Apéndice D Documentación técnica de programación</b>	<b>43</b>
D.1. Introducción . . . . .	43

D.2. Estructura de directorios . . . . .	43
D.3. Manual del programador . . . . .	43
D.4. Compilación, instalación y ejecución del proyecto . . . . .	43
D.5. Pruebas del sistema . . . . .	43
<b>Apéndice E Documentación de usuario</b>	<b>45</b>
E.1. Introducción . . . . .	45
E.2. Requisitos de usuarios . . . . .	45
E.3. Instalación . . . . .	45
E.4. Manual del usuario . . . . .	45
<b>Bibliografía</b>	<b>47</b>

---

# Índice de figuras

---

A.1.	Horas estimadas frente horas empleadas. . . . .	2
A.2.	Diagrama Gantt del proyecto. . . . .	4
A.3.	Visualización del <i>lead</i> y <i>cycle time</i> . . . . .	4
A.4.	Gráfico de <i>lead</i> y <i>cycle time</i> medios. . . . .	5
A.5.	Diagrama de flujo acumulado desde el comienzo del proyecto. . . . .	5
A.6.	Distribución de las tareas según su tipo. . . . .	6
A.7.	Análisis DAFO de JIZT. . . . .	10
A.8.	Resumen de la licencia GNU GPLv3 . . . . .	11
B.1.	Diagrama de casos de uso. . . . .	19
C.1.	Diagrama de clases del <i>Dispatcher</i> . . . . .	29
C.2.	Diagrama E/R de la base de datos (tipos de datos de PostgreSQL). . . . .	30
C.3.	Diagrama de clases del Pre-procesador de textos. . . . .	31
C.4.	Diagrama de clases del Codificador de textos. . . . .	32
C.5.	Diagrama de clases del Generador de resúmenes. . . . .	33
C.6.	Diagrama de clases del Post-procesador de textos. . . . .	34
C.7.	Primera aproximación para el diseño arquitectónico del <i>backend</i> . . . . .	36
C.8.	Diseño final de la arquitectura del <i>backend</i> . . . . .	37
C.9.	Diseño final de la arquitectura de la aplicación. . . . .	38
C.10.	Ideas, ideas, y más ideas. Pero solo unas pocas buenas. . . . .	39
C.11.	JIZT: generación de resúmenes mediante IA. . . . .	39
C.12.	Variaciones sobre el logo de JIZT. . . . .	40
C.13.	Iteraciones sobre la pantalla principal. . . . .	41
C.14.	Imágenes de la aplicación para su publicación en Play Store. . . . .	41

---

# Índice de tablas

---

A.1. Desglose de costes fijos del proyecto . . . . .	7
A.2. Desglose de costes directos del proyecto . . . . .	8
A.3. Desglose de costes indirectos del proyecto . . . . .	8
A.4. Listado de dependencias . . . . .	12
B.1. CU-01 Solicitar resumen . . . . .	19
B.1. CU-01 Solicitar resumen . . . . .	20
B.2. CU-02 Seleccionar modelo . . . . .	20
B.3. CU-02 Seleccionar modelo . . . . .	21
B.4. CU-06 Consultar historial de resúmenes . . . . .	21
B.4. CU-06 Consultar historial de resúmenes . . . . .	22
B.5. CU-07 Consultar resumen . . . . .	22
B.6. CU-08 Ver texto original . . . . .	22
B.6. CU-08 Ver texto original . . . . .	23
B.7. CU-09 Compartir el resumen . . . . .	23
B.8. CU-10 Borrar el resumen . . . . .	24
C.1. Resultados de la encuesta. . . . .	26

## *Apéndice A*

---

# **Plan de Proyecto Software**

---

## **A.1. Introducción**

La planificación de todo proyecto es un proceso de gestión organizado e integrado, centrado en las actividades necesarias para asegurar una exitosa consecución del proyecto. Esta planificación contribuye a una mejor utilización de los recursos, y a un uso óptimo del tiempo asignado a un proyecto, algo crucial en todo proyecto, y especialmente relevante en el caso de un Trabajo de Fin de Grado (TFG).

En este apartado se recogen los aspectos más relevantes en cuanto a la planificación temporal de nuestro proyecto, así como su viabilidad, tanto económica como legal.

## **A.2. Planificación temporal**

Una de las primeras decisiones que se llevaron a cabo en el marco del proyecto JIZT, fue la elección de la metodología de desarrollo *software* que adoptaríamos.

Lo primero que llevamos a cabo fue un análisis de las necesidades y limitaciones que presentaba nuestro proyecto, las cuales eran: tiempo limitado, eficiencia y velocidad, motivación y progreso del proyecto, y satisfacción de los usuarios.

Derivado de estas necesidades, se decidió que emplearíamos una metodología ágil. Las principales metodologías ágiles consideradas fueron Scrum, Kaban y Programación Extrema (XP).

Finalmente, nos decantamos por Kanban. Esta decisión estuvo en gran medida motivada por el hecho de que una planificación temporal rígida y prefijada, como ocurre por ejemplo en Scrum, era muy difícil de llevar a cabo en este proyecto dada nuestra inexperiencia con muchas de las herramientas y *frameworks* utilizados.

Kanban nos permitía un flujo continuo de trabajo, permitiendo añadir historias de usuario no contempladas inicialmente si así se consideraba apropiado.

En total se estimaron 465 horas, de las cuales se emplearon realmente 464:

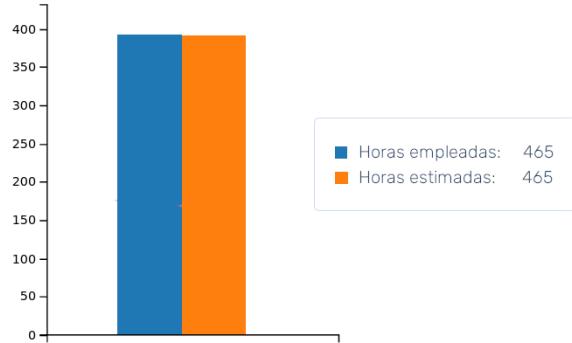


Figura A.1: Horas estimadas frente horas empleadas.

En esta planificación, se emplearon dos conceptos ágiles fundamentales:

- **Historia de usuario:** se trata de una descripción de una funcionalidad del *software* a implementar.
- ***Epics*:** agrupan historias de usuario que conformen una misma *feature*, o funcionalidad a desarrollar.

En total, se especificaron 99 historias de usuario repartidas entre 8 *epics*<sup>1</sup>. Estos *epics* fueron, ordenados de manera cronológica:

---

<sup>1</sup>Para la representación de las historias de usuarios y los *epics* a GitHub se emplearon *Issues* y *Milestones*, respectivamente.

- **Puesta en marcha:** tareas preliminares de organización y puesta en marcha del proyecto (elección de metodologías, herramientas, etc.).
- **Motor de Resumen v0.1:** implementar una primera versión del Motor de Resumen a través de los modelos preentrenados proporcionados por el módulo `transformers` de Hugging Face [1].
- **Arquitectura Microservicios v0.1:** implementar una primera versión reducida de la Arquitectura de Microservicios, configurando el componente Ingress de Kubernetes [2], y dos microservicios: el Dispatcher y el Pre-procesador de textos.
- **Arquitectura Microservicios v0.2:** continuar con la implementación de la arquitectura de microservicios, añadiendo la capacidad de realizar peticiones asíncronas y desarrollando la arquitectura dirigida por eventos. De momento, se sigue trabajando con una versión de la misma, esto es, con el Dispatcher y el Pre-procesador de textos.
- **Arquitectura Microservicios v0.3:** una vez disponemos de una versión reducida de nuestra arquitectura que funciona correctamente en local, el siguiente paso es desplegarla en Google Kubernetes Engine (GKE) [3]. Además, se deben implementar los microservicios restantes (Codificador, Motor de Resumen y Post-procesador) y la base de datos para gestionar los resúmenes.
- **Cliente v0.1:** desarrollar el cliente (aplicación) que consumirá la API y permitirá al usuario final obtener resúmenes de sus textos. Dicho cliente se implementará con ayuda de Flutter [4], por lo que en principio estará disponible en plataformas móvil, *web* y escritorio.
- **Arquitectura Microservicios v0.4:** ampliar la especificación de la API para que en las peticiones se puedan detallar todos los parámetros del resumen. Continuar con la mejora del sistema.
- **Documentación v0.1:** escribir la Memoria y los Anexos. Generar una primera versión de la documentación de la API REST y del código perteneciente a JIZT.

En la Figura A.2 se recoge un diagrama Gantt con el objetivo de facilitar la comprensión de la dimensión temporal del proyecto.

Dos conceptos importantes dentro de la metodología son *lead time* y *cycle time* [5]. Veamos qué significa cada uno de ellos.

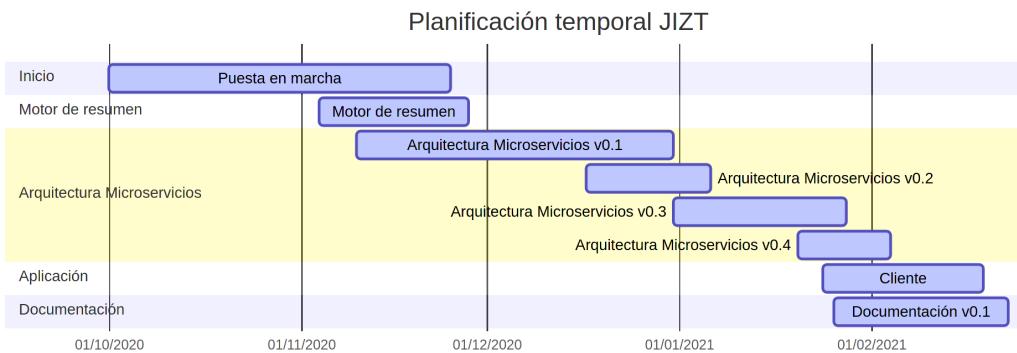


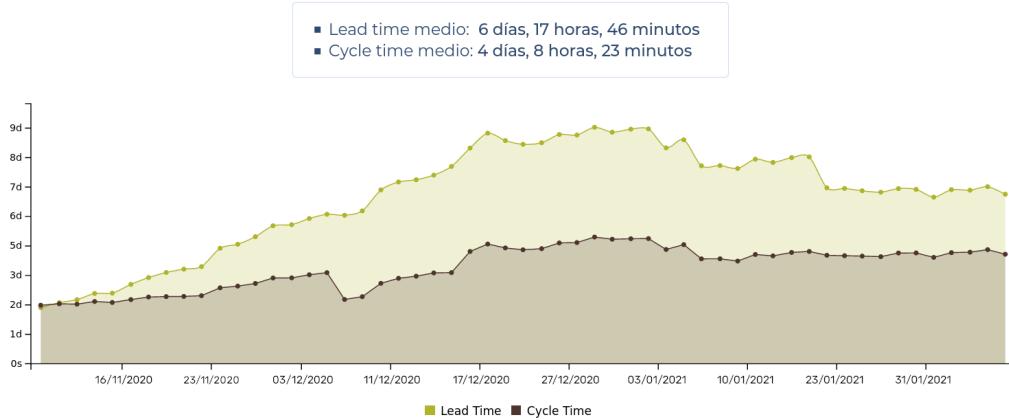
Figura A.2: El proyecto comenzó el 1 de octubre de 2020, y finalizó el 16 de febrero de 2021.

- *Lead time*: es el período que transcurre entre la aparición de una nueva tarea en el flujo de trabajo y su salida final del sistema. Dicho de otro modo, es el tiempo total que el cliente está esperando la entrega de una parte del producto.
- *Cycle time*: es la cantidad de tiempo que el equipo realmente empleó en una tarea, es decir, no se cuenta el tiempo que una tarea estuvo «en espera». Por lo tanto, el tiempo del ciclo debe comenzar a medirse cuando la tarea pasa a la columna «trabajando», y no antes.



Figura A.3: Explicación gráfica del *lead* y *cycle time* sobre un tablero Kanboard.

Esta métrica nos aporta información que nos permite conocer cuánto tiempo tardaremos en entregar una determinada parte del producto. Es importante mantener el *lead* y *cycle time* tan cortos como sea posible, a fin de mantener pocas tareas «en ejecución» (WIP), permitiendo mantener un flujo constante de trabajo y aportar valor al cliente de manera frecuente.

Figura A.4: Gráfico de *lead* y *cycle time* medios.

Como vemos en la [anterior figura](#), el *lead time* medio fue de algo menos de 7 días, y el *cycle time* de 4 días y 8 horas. Como es lógico, las primeras tareas se completaron más rápido, pero según la complejidad de las mismas fue incrementándose, también se reflejo en los tiempos. En el punto central del proyecto, se alcanzó una media de *lead time* de 9 días, aunque el *cycle time* se mantuvo por debajo de los 5, lo que indica que existía un mayor número de tareas esperando a ser atendidas.

Otro de los gráficos propios de Kanban que nos puede ofrecer información valiosa es el llamado diagrama de flujo acumulado (CFD, por sus siglas en inglés). Este gráfico muestra el número de tareas que hay en cada columna a lo largo del tiempo.

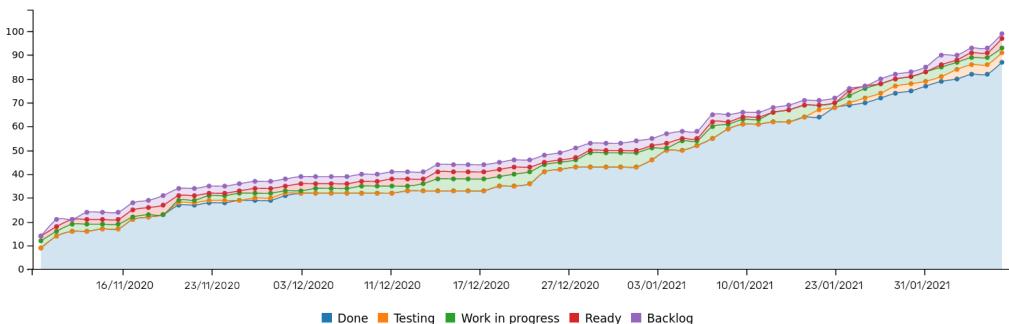


Figura A.5: Diagrama de flujo acumulado desde el comienzo del proyecto.

Como podemos ver en el [anterior diagrama](#), el trabajo en las diferentes columnas se distribuyó de forma correcta, no apareciendo grandes diferencias

entre ellas. En este gráfico, también podemos apreciar que en la parte central del proyecto, las tareas en «*Work in progress*» fueron algo mayores que en el resto de columnas, lo cual es comprensible.

El diagrama de flujo acumulado obtenido muestra también que el ritmo de trabajo fue constante, incrementándose ligeramente hacia el final del proyecto.

Podemos visualizar también la distribución de las tareas en función de su tipo:

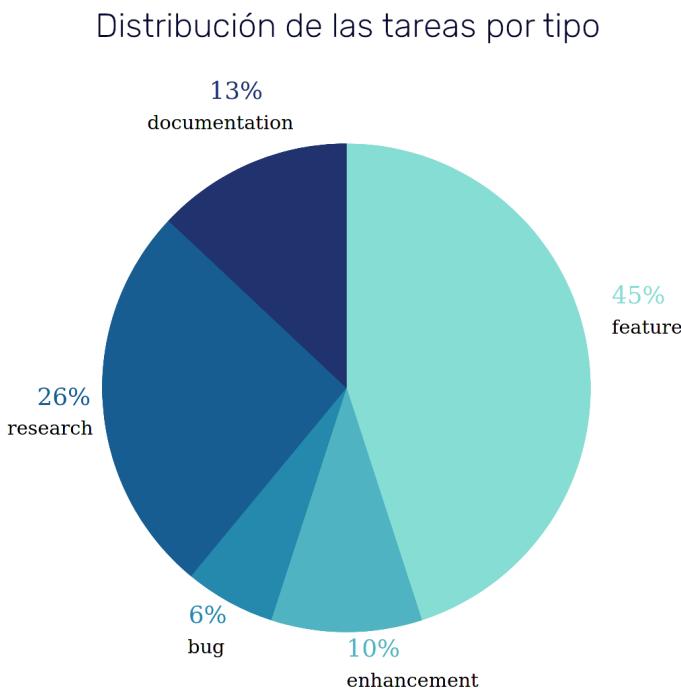


Figura A.6: Distribución de las tareas según su tipo.

Como es lógico, la mayor parte de las tareas se dedicaron a ofrecer nuevas funcionalidades (*feature*), aunque gran número de ellas se dedicaron al aprendizaje y búsqueda de información (*research*), lo cual también parece ajustarse a la realidad, puesto que como ya hemos mencionado, muchas de las herramientas y técnicas que hemos utilizado eran nuevas para nosotros.

Para finalizar esta sección, cabe mencionar que en el [repositorio del proyecto](#), y en su [tablero Kanban](#), se puede encontrar información más detallada de cada historia de usuario y *epic*.

## A.3. Estudio de viabilidad

### Viabilidad económica

Uno de los puntos cruciales a la hora de estudiar la viabilidad de un proyecto, y que en muchos casos determina el éxito o el fracaso del mismo, es la viabilidad económica.

En esta sección analizamos los costes y beneficios de JIZT.

#### Costes del proyecto

En nuestro caso, dividiremos los costes del proyecto en costes fijos, directos e indirectos.

#### *Costes fijos*

Los costes fijos son aquellos costes invariables que debemos abonar, independientemente del desarrollo del proyecto [6].

CONCEPTO	IMPORTE
Servicio de Internet	200,00 €
Servicio de Luz <sup>1</sup>	225,00 €
Materiales de oficina	5,00 €
Salarios <sup>2</sup>	9911,88 €
Salario mensual neto	1000,00 €
Retenciones <sup>1</sup> por IRPF (24%) <sup>3</sup>	528,63 €
Cuotas a la Seg. Social (30,6%) <sup>4</sup>	674,01 €
Salario mensual bruto	2202,64 €
<b>TOTAL</b>	<b>10341,88 €</b>

<sup>1</sup> Costes calculados para 4,5 meses, con tarifa de mercado libre y potencia contratada de 3,3 kW (precio mensual medio de 50 €).

<sup>2</sup> Costes calculados para 4,5 meses.

<sup>3</sup> Según la tabla de retenciones por IRPF aplicable al ejercicio 2021 [7]

<sup>4</sup> Porcentaje para autónomos según la Ley 11/2020, de 30 de diciembre, de Presupuestos Generales del Estado para el año 2021 [8].

Tabla A.1: Desglose de costes fijos del proyecto.

### ***Costes directos***

Los costes directos son aquellos costes derivados directamente del desarrollo del proyecto.

CONCEPTO	IMPORTE	IMPORTE AMORTIZ.
Costes de <i>hardware</i> <sup>1</sup>	2509,58 €	79,49 €
Ordenador personal	845,00 €	63,37 €
Smartphone Android	215,00 €	16,12 €
Servicio GKE <sup>2</sup> de Google Cloud	1449,58 €	-
Costes de <i>software</i> <sup>3</sup>	89,95 €	16,86 €
Adobe Illustrator	89,95 €	16,86 €
<b>TOTAL</b>	<b>2599,53 €</b>	<b>96,35 €</b>

<sup>1</sup> Se han calculado con una amortización de 5 años, habiendo sido utilizado 4,5 meses.  
<sup>2</sup> Google Kubernetes Engine [3].  
<sup>1</sup> Se han calculado con una amortización de 2 años, habiendo sido utilizado 4,5 meses.

Tabla A.2: Desglose de costes directos del proyecto.

### ***Costes indirectos***

Los costes indirectos son aquellos que no dependen directamente del desarrollo del proyecto.

CONCEPTO	IMPORTE
Dominio <a href="http://jizt.it">jizt.it</a>	4,81 €
Cuenta de Google Play	20,76 €
Impresión de la Memoria y el cartel del TFG	40,00 €
<b>TOTAL</b>	<b>65,57 €</b>

Tabla A.3: Desglose de costes indirectos del proyecto.

### ***Costes totales del proyecto***

Considerando las tres categorías de costes recogidas anteriormente, la suma de los costes totales del proyecto asciende a **13006,10 €**.

## Beneficios

La API REST de JIZT se ofrece en tres planes de suscripción diferentes.

- **Gratis:** este plan se ajusta a las necesidades de cualquier usuario regular que no vaya a realizar un uso exhaustivo del servicio. Se permiten 5.000 peticiones a la API REST, pudiendo hacerse hasta 5 peticiones por minuto. No incluye soporte técnico.
- **Estándar:** para aquellas empresas o particulares que van a realizar un uso más intensivo del servicio. Se incluyen 15.000 peticiones a la API REST, pudiendo hacerse hasta 15 peticiones por minuto. Incluye soporte técnico y de integración. El precio es de 166 €/mes.
- **Personalizado:** para aquellos usuarios cuyas necesidades no encajen en ninguno de los anteriores precios. El precio se establecerá en función de los requerimientos concretos del usuario.

En cuanto a la aplicación, es totalmente gratuita y no contiene publicidad.

## Análisis DAFO

Tras llevar a cabo un pequeño análisis de mercado, hemos identificado que las principales debilidades, amenazas, fortalezas y oportunidades de nuestro proyecto son las siguientes:



Figura A.7: Análisis DAFO de JIZT.

## Viabilidad legal

### Licencia del código fuente del proyecto

Desde un primer momento nuestra intención era licenciar el proyecto bajo una licencia de *Software Libre*. Dentro de este entorno, se han considerado las tres licencias más extendidas: Apache-2.0, MIT, y GPLv3.

Tras una lectura exhaustiva de las cláusulas de cada una de ellas, así como de opiniones en *blogs*, charlas, foros, etc., y tras una profunda reflexión, considerando especialmente la licencia MIT y la GPLv3, hemos tomado la decisión de que nuestro software estará licenciado bajo **GNU GPLv3** [9], cuyos puntos principales se recogen en la Figura A.8.

 dmlls/jitz is licensed under the <b>GNU General Public License v3.0</b>																		
<p>Los permisos de esta licencia copyleft están condicionados a la puesta a disposición del código fuente completo de las obras con licencia y de las modificaciones. Deben conservarse los avisos de copyright y de licencia. Los colaboradores proporcionan una concesión expresa de los derechos de patente.</p>																		
<table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding-bottom: 5px;"><b>Permisos</b></th> <th style="text-align: left; padding-bottom: 5px;"><b>Limitaciones</b></th> <th style="text-align: left; padding-bottom: 5px;"><b>Condiciones</b></th> </tr> </thead> <tbody> <tr> <td style="padding-top: 5px;">✓ Uso comercial</td> <td style="padding-top: 5px;">✗ Responsabilidad</td> <td style="padding-top: 5px;">① Avisos de licencia y copyright</td> </tr> <tr> <td style="padding-top: 5px;">✓ Modificación</td> <td style="padding-top: 5px;">✗ Garantía</td> <td style="padding-top: 5px;">② Indicar los cambios</td> </tr> <tr> <td style="padding-top: 5px;">✓ Distribución</td> <td></td> <td style="padding-top: 5px;">③ Publicar el código fuente</td> </tr> <tr> <td style="padding-top: 5px;">✓ Uso de la patente</td> <td></td> <td style="padding-top: 5px;">④ Emplear la misma licencia</td> </tr> <tr> <td style="padding-top: 5px;">✓ Uso privado</td> <td></td> <td></td> </tr> </tbody> </table>	<b>Permisos</b>	<b>Limitaciones</b>	<b>Condiciones</b>	✓ Uso comercial	✗ Responsabilidad	① Avisos de licencia y copyright	✓ Modificación	✗ Garantía	② Indicar los cambios	✓ Distribución		③ Publicar el código fuente	✓ Uso de la patente		④ Emplear la misma licencia	✓ Uso privado		
<b>Permisos</b>	<b>Limitaciones</b>	<b>Condiciones</b>																
✓ Uso comercial	✗ Responsabilidad	① Avisos de licencia y copyright																
✓ Modificación	✗ Garantía	② Indicar los cambios																
✓ Distribución		③ Publicar el código fuente																
✓ Uso de la patente		④ Emplear la misma licencia																
✓ Uso privado																		

Figura A.8: Resumen de la licencia GNU GPLv3. Imagen extraída y traducida de <https://github.com/dmlls/jitz/blob/main/LICENSE>.

Las principales razones de nuestra elección son:

- Pese a que la licencia MIT pueda parecer más permisiva en un primer lugar, ya que no obliga a que el código fuente se mantenga abierto en un futuro, creemos que a largo plazo esta «permisividad», paradójicamente, puede resultar en una limitación de sí misma. Esto es, el hecho de que ese supuesto *software* «libre» se pueda volver *software* «cerrado», lo excluye en primer lugar de esa definición de «libre», en nuestra opinión.
- Este proyecto no podría existir sin las contribuciones de *software* libre anteriores. Por ello, queremos asegurar que este proyecto siempre se mantenga abierto para poder ayudar a otros y retroalimentarse con los aportes de la comunidad.
- El simple hecho de elegir una licencia, conlleva un sinnúmero de implicaciones morales, económicas, sociales, etc., pero es algo necesario, ya que el *software* sin licencia explícita se toma por defecto como *copyright*.

### Listado de dependencias

Todas las dependencias del proyecto se encuentran licenciadas bajo licencias compatibles con GNU GPLv3.

A continuación, se recoge una lista detallada de las mismas:

Dependencia	Versión	Descripción	Licencia
<b>API REST</b>			
NLTK	3.5	Utilidades de NLP.	Apache v2.0
transformers	4.1.1	Modelos pre-entrenados.	Apache v2.0
truecase	0.0.12	Truecaser.	Apache v2.0
confluent-kafka	1.5.0	Cliente de Kafka para Python.	Apache v2.0
strimzi-operator	0.21.0	Kafka en Kubernetes.	Apache v2.0
postgres-operator	4.5.1	PostgreSQL en Kubernetes.	Apache v2.0
blingfire	0.1.3	Utilidades de NLP.	MIT
marshmallow	4.5.1	Serialización y deserialización.	Apache v2.0
<b>Aplicación</b>			

Tabla A.4: Listado de dependencias.

1

### Licencia de la documentación del proyecto

La totalidad de la documentación de JIZT se distribuye bajo licencia GNU Free Documentation License (GFDL) [10].

Esta licencia es una adaptación al contexto de la documentación de la GNU General Public License (GPL), la cual está pensada para licenciar código fuente.

La GFDL da permiso a los lectores de copiar, redistribuir y modificar (excepto las «secciones invariables») una obra y exige que todas las copias y derivados estén disponibles bajo la misma licencia. Las copias también pueden venderse comercialmente, pero, si se producen en grandes cantidades (más de 100), el documento original o el código fuente deben ponerse a disposición del destinatario de la obra [10].

## *Apéndice B*

---

# **Especificación de Requisitos**

---

## **B.1. Introducción**

En este apéndice se recoge una descripción del sistema *software* a desarrollar. Más concretamente, se presentan aspectos como el propósito, alcance y objetivos generales del mismo, así como los requisitos específicos que el sistema debe implementar.

La forma en la que hemos organizado la Especificación está basada parcialmente en las recomendaciones del estándar IEEE 830-1998, considerado como la principal referencia para la aplicación de buenas prácticas en la escritura del Documento de Especificación de Requisitos (SRS).

### **Propósito**

El propósito de la presente Especificación de Requisitos tiene como objetivo ofrecer una descripción detallada del sistema *software* desarrollado.

Este documento está orientado principalmente al equipo de desarrollo del proyecto, pero queda también a libre disposición de cualquier persona que pudiera estar interesada.

### **Alcance**

El sistema *software* a desarrollar recibirá el nombre de JIZT. El objetivo y funcionalidad principal de JIZT será la generación de resúmenes abstractivos en la nube. Dicho sistema estará orientado tanto a usuarios con conocimientos de informática básicos, así como usuarios con experiencia en el campo.

Para los primeros, es decir, usuarios con conocimientos básicos, se ofrecerá una aplicación multiplataforma desarrollada con la usabilidad en mente.

Para aquellos usuarios más avanzados, además de la aplicación, se proporcionará una API REST, permitiéndoles realizar peticiones directamente, e incluso desarrollar sus propias aplicaciones que consuman dicha API.

Se pondrá especial atención en el desarrollo de una documentación completa, accesible y de fácil comprensión. Esta documentación recogerá tanto los aspectos técnicos de la aplicación y la API REST, así como manuales de uso para el usuario y para el desarrollador.

## Definiciones, siglas, y abreviaciones

A continuación se recogen las definiciones, siglas y abreviaciones más relevantes para el proyecto:

- **NLP**: Procesamiento de Lenguaje Natural (*Natural Language Processing*).
- **Resumen extractivo**: aquel resumen generado a partir de la *extracción* de las frases del texto consideradas más relevantes. Es decir, este tipo de resúmenes contienen únicamente frases tomadas literalmente del texto original.
- **Resumen abstractivo**: aquel resumen que incluye palabras o expresiones que no aparecen en el texto original.
- **API REST**: servicio *web* que proporciona una serie de *endpoints* para llevar a cabo operaciones HTTP (métodos), que proporcionan acceso para crear, recuperar, actualizar o eliminar los recursos del servicio.
- **Endpoint**: punto de entrada en un canal de comunicación cuando dos sistemas interactúan. En una API REST los *endpoints* son generalmente URLs.
- **HTTP**: *Hypertext Transfer Protocol*, protocolo para sistemas de información distribuidos, colaborativos e hipermedia. Este protocolo es la base de la comunicación de datos para la World Wide Web (WWW).
- **URL**: *Uniform Resource Locator*, referencia a un recurso web que especifica su ubicación en una red informática y un mecanismo para recuperarlo.
- **Frontend**: la infraestructura *frontend* incluye todos aquellos elementos del sistema con los que el usuario final interactúa.

- **Backend:** parte de la infraestructura que contiene el conjunto de servidores, bases de datos, APIs y los sistemas operativos que alimentan el *frontend* de una aplicación.

## B.2. Descripción global

### Objetivos generales

Los principales objetivos del sistema *software* a desarrollar son los siguientes:

- Implementar un servicio de generación de resúmenes abstractivos en la nube, empleando modelos pre-entrenados del estado del arte.
- Implementar el *backend* que haga posible esta generación de resúmenes, así como una API REST para exponer el *backend*.
- Diseñar la arquitectura del *backend* con aspectos como la flexibilidad, la escalabilidad y la alta disponibilidad como pilares fundamentales.
- Implementar un *frontend* (aplicación multiplataforma) que consuma la REST API y dé la posibilidad de generar resúmenes a cualquier usuario.
- Implementar una interfaz (tanto en el *backend* como en el *fontend*) que permita a los usuarios especificar los parámetros con los que el resumen será generado, como por ejemplo, su longitud relativa al texto original.

### Características del usuario

Los usuarios potenciales de JIZT se pueden dividir en dos categorías fundamentales:

- Estudiantes de edades comprendidas entre los 15 y 25 años. El uso principal que harán del producto será el resumen de textos académicos.
- Personas de edades comprendidas entre los 25 y 50 años, cuyo principal uso será el resumen de noticias y artículos periodísticos.

En cuanto al sexo, el uso por parte tanto de hombres como mujeres será aproximadamente equivalente.

Un tercer tipo de usuario, no incluido anteriormente, serían partidos interesados en el uso del producto, como empresas o particulares, que fueran a hacer un uso extensivo y exhaustivo del mismo. Este tipo de usuario requiere unas prestaciones más exigentes, pudiendo llegar a solicitar el despliegue de JIZT en sus propias dependencias.

Es por ello que el producto desarrollado deberá ser lo más independiente del entorno en el que se despliegue como sea posible.

### B.3. Catalogo de requisitos

A continuación se detallan los requisitos funcionales y no funcionales del producto a desarrollar.

#### Requisitos funcionales

- **RF-1 Solicitar resumen:** la API REST debe proporcionar un *end-point* para que el usuario pueda solicitar un resumen de un texto.
  - **RF-1.1 Elegir modelo de generación de resumen:** el usuario deberá ser capaz de especificar el modelo de generar su resumen.
  - **RF-1.2 Especificar longitud relativa del resumen:** el usuario deberá ser capaz de especificar la longitud del resumen generado, de manera relativa al texto original.
  - **RF-1.3 Especificar parámetros del resumen:** el usuario deberá ser capaz de especificar los parámetros concretos con los que se generará su resumen.
- **RF-2 Historial de resúmenes:** el usuario podrá acceder a los resúmenes que ha generado recientemente.
- **RF-3 Compartir resumen:** se le brindará al usuario la opción de compartir el resumen generado a través de otra aplicación a su elección.
- **RF-4 Copiar resumen:** se le brindará al usuario la opción de copiar el resumen generado.
- **RF-5 Borrar resumen:** el usuario deberá ser capaz de borrar permanentemente un resumen previamente generado.

- **RF-6 Pegar desde el portapapeles:** la aplicación ofrecerá una opción para que el usuario pueda pegar el texto a resumir desde el portapapeles de forma sencilla.
- **RF-7 Mostrar metadatos:** el sistema brindará al usuario los metadatos relativos al resumen generado, como la hora a la que fue creado.
- **RF-8 Pre-procesado del texto:** el sistema será capaz de recibir texto con errores de formateo (exceso de espacios, saltos de carro situados en mitad de una frase, etc.), así como caracteres «extraños». Independientemente de lo anterior, el resumen generado aparecerá correctamente formateado y sin los mencionados caracteres.
- **RF-9 Textos arbitrariamente largos:** el sistema será capaz de producir resúmenes de cualquier texto, independientemente de la longitud de los mismos. No se espera, no obstante, que el tiempo de resumen de textos extremadamente largos esté por debajo del orden del minuto.

## Requisitos no funcionales

- **RNF-1 Escalabilidad:** la arquitectura del sistema deberá permitir el escalado del mismo de forma rápida y sencilla.
  - **RNF-1.1 Autoescalado:** el sistema podrá escalarse de manera automática en momentos en los que la carga de trabajo así lo requiera. Del mismo modo, cuando dicha carga remita, deberá disminuir su escala, a fin de consumir los mínimos recursos posibles.
- **RNF-2 Alta disponibilidad:** el sistema deberá garantizar el acceso al mismo por parte de los usuarios en el 99,99 % de los casos.
  - **RNF-2.1 Tolerancia frente a fallos:** el sistema deberá ser capaz de recuperarse de forma automática de posibles errores o problemas de funcionamiento de cualquiera de sus componentes en un tiempo menor a los 2 minutos.
- **RNF-3 Eficiencia:** el sistema deberá ser capaz de generar un elevado número de resúmenes provenientes de diferentes usuarios de forma simultánea, sin que el tiempo medio de resumen se vea afectado.

- **RNF-4 Seguridad lógica y de datos:** se debe garantizar la correcta protección de todos los datos manejados por el sistema.
- **RNF-5 Privacidad:** se debe asegurar la protección de los datos de carácter personal.
  - **RNF-5.1 Anonimidad:** en ningún caso se recopilará información de los usuarios que permita determinar la identidad de los mismos. No obstante, el sistema no es responsable de garantizar que los textos introducidos no contienen información de carácter personal.
- **RNF-6 Usabilidad:** el tiempo medio de aprendizaje de la aplicación por parte de los usuarios deberá ser inferior a los 5 minutos. Además, el sistema contará con documentación en línea detallada del producto.
- **RNF-7 Multiplataforma:** se distribuirán los binarios de la aplicación necesarios para su ejecución en móvil (Android e iOS), *web* (Google Chrome, Mozilla Firefox, Safari y Microsoft Edge), y escritorio (Linux, Apple y Windows).
- **RNF-8 Tamaño reducido:** el peso de la aplicación no debe superar los 30 MB.

## B.4. Especificación de requisitos

### Diagrama de casos de uso

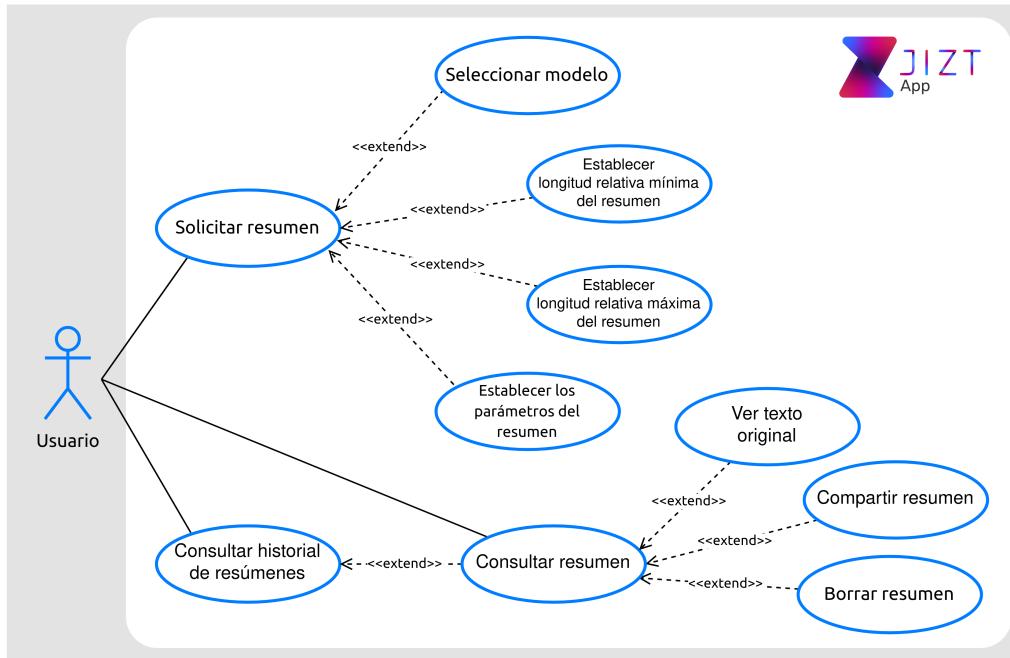


Figura B.1: Diagrama de casos de uso.

### Actores

Existe un único actor: el usuario que hace uso de la aplicación.

### Casos de uso

CU-01	Solicitar resumen
<b>Descripción</b>	Solicitar la generación de un resumen a partir de un texto.
<b>Autor</b>	Diego Miguel Lozano
<b>Requisitos relacionados</b>	RF-1, RF-6, RF-8, RF-9
<b>Precondición</b>	La API REST se encuentra accesible.
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El usuario inicia la aplicación.</li> <li>2. El usuario hace <i>click</i> en</li> </ol>

Tabla B.1: CU-01 Solicitar resumen

CU-01	Solicitar resumen
	el área de texto.
	3. El usuario introduce el texto a resumir o, alternativamente, lo pega desde el portapapeles.
	4. El usuario pulsa en el botón « <i>Summarize</i> ».
	5. Se muestra un indicador de «procesando».
	6. Se muestra un indicador de «resumen completado».
	7. Se muestra el resumen generado.
<b>Postcondición</b>	El usuario ha obtenido el resumen de su texto.
<b>Excepciones</b>	API REST inaccesible.
<b>Incluye</b>	-
<b>Extiende</b>	-
<b>Prioridad</b>	Muy alta.
<b>Frecuencia de uso</b>	Muy alta.
<b>Importancia</b>	Crítica.
<b>Comentarios</b>	-

Tabla B.1: CU-01 Solicitar resumen

CU-02	Seleccionar modelo
<b>Descripción</b>	Seleccionar el modelo con el que se llevará a cabo el resumen.
<b>Autor</b>	Diego Miguel Lozano
<b>Requisitos relacionados</b>	RF-1.1
<b>Precondición</b>	-
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El usuario pulsa en el ícono de opciones en el cuadro de texto.</li> <li>2. El usuario elige el modelo.</li> </ol>
<b>Postcondición</b>	El modelo ha sido seleccionado.
<b>Excepciones</b>	-
<b>Incluye</b>	-
<b>Extiende</b>	-
<b>Prioridad</b>	Baja.
<b>Frecuencia de uso</b>	Media.
<b>Importancia</b>	Media.
<b>Comentarios</b>	Por implementar.

Tabla B.2: CU-02 Seleccionar modelo

CU-02	Seleccionar modelo
<b>Descripción</b>	Seleccionar el modelo con el que se llevará a cabo el resumen.
<b>Autor</b>	Diego Miguel Lozano
<b>Requisitos relacionados</b>	RF-1.1
<b>Precondición</b>	-
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El usuario pulsa en el ícono de opciones en el cuadro de texto.</li> <li>2. El usuario elige el modelo.</li> </ol>
<b>Postcondición</b>	El modelo ha sido seleccionado.
<b>Excepciones</b>	-
<b>Incluye</b>	-
<b>Extiende</b>	CU-01
<b>Prioridad</b>	Baja.
<b>Frecuencia de uso</b>	Media.
<b>Importancia</b>	Media.
<b>Comentarios</b>	Por implementar.

Tabla B.3: CU-02 Seleccionar modelo

CU-06	Consultar historial de resúmenes
<b>Descripción</b>	Visualizar la lista de resúmenes generados previamente.
<b>Autor</b>	Diego Miguel Lozano
<b>Requisitos relacionados</b>	RF-2, RF-3, RF-4, RF-5, RF-7
<b>Precondición</b>	Haber generado al menos un resumen previamente.
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El usuario pulsa en «<i>See all</i>» en la pantalla principal.</li> <li>2. Se muestra la lista de resúmenes previos.</li> </ol>
<b>Postcondición</b>	Se visualizan los resúmenes generados.
<b>Excepciones</b>	-
<b>Incluye</b>	-
<b>Extiende</b>	CU-07
<b>Prioridad</b>	Alta.
<b>Frecuencia de uso</b>	Alta.

Tabla B.4: CU-06 Consultar historial de resúmenes

<b>CU-06</b>	<b>Consultar historial de resúmenes</b>
<b>uso</b>	
<b>Importancia</b>	Alta.
<b>Comentarios</b>	Si aún no se ha generado ningún resumen, la lista se mostrará vacía.

Tabla B.4: CU-06 Consultar historial de resúmenes

<b>CU-07</b>	<b>Consultar resumen</b>
<b>Descripción</b>	Consultar un resumen generado previamente.
<b>Autor</b>	Diego Miguel Lozano
<b>Requisitos relacionados</b>	RF-2
<b>Precondición</b>	Haber generado al menos un resumen previamente.
<b>Flujo normal</b>	<p>Flujo 1:</p> <ol style="list-style-type: none"> <li>1. El usuario pulsa en uno de los resúmenes que aparecen en el inferior de la pantalla.</li> </ol> <p>Flujo 2 (alternativa):</p> <ol style="list-style-type: none"> <li>1. El usuario pulsa en «<i>See all</i>» en la pantalla principal.</li> <li>2. Se muestra la lista de resúmenes previos.</li> <li>3. El usuario pulsa en uno de los resúmenes.</li> </ol>
<b>Postcondición</b>	Se ha mostrado el resumen seleccionado.
<b>Excepciones</b>	-
<b>Incluye</b>	-
<b>Extiende</b>	CU-06
<b>Prioridad</b>	Muy alta.
<b>Frecuencia de uso</b>	Alta.
<b>Importancia</b>	Crítica.
<b>Comentarios</b>	-

Tabla B.5: CU-07 Consultar resumen

<b>CU-08</b>	<b>Ver texto original</b>
<b>Descripción</b>	Visualizar el texto a partir del cual se ha generado el resumen.
<b>Autor</b>	Diego Miguel Lozano

Tabla B.6: CU-08 Ver texto original

CU-08	Ver texto original
<b>Requisitos relacionados</b>	RF-2, RF-7
<b>Precondición</b>	Haber generado un resumen.
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El usuario pulsa en «<i>Original</i>».</li> </ol> <p>cuadro de texto.</p>
<b>Postcondición</b>	Se ha mostrado el texto original.
<b>Excepciones</b>	-
<b>Incluye</b>	-
<b>Extiende</b>	CU-07
<b>Prioridad</b>	Alta.
<b>Frecuencia de uso</b>	Alta.
<b>Importancia</b>	Crítica.
<b>Comentarios</b>	-

Tabla B.6: CU-08 Ver texto original

CU-09	Compartir el resumen
<b>Descripción</b>	Compartir el resumen generado a través de otra aplicación.
<b>Autor</b>	Diego Miguel Lozano
<b>Requisitos relacionados</b>	RF-3
<b>Precondición</b>	Haber generado un resumen.
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El usuario pulsa en el ícono de compartir.</li> <li>2. Se muestra una lista de aplicaciones.</li> <li>2. El usuario pulsa en la aplicación a través de la cual quiere compartir el resumen</li> </ol>
<b>Postcondición</b>	Se ha compartido el resumen.
<b>Excepciones</b>	-
<b>Incluye</b>	-
<b>Extiende</b>	CU-07
<b>Prioridad</b>	Media.
<b>Frecuencia de uso</b>	Media.
<b>Importancia</b>	Media.
<b>Comentarios</b>	-

Tabla B.7: CU-09 Compartir el resumen

<b>CU-10</b>	<b>Borrar el resumen</b>
<b>Descripción</b>	Borrar un resumen generado.
<b>Autor</b>	Diego Miguel Lozano
<b>Requisitos relacionados</b>	RF-5
<b>Precondición</b>	Haber generado un resumen.
<b>Flujo normal</b>	<ol style="list-style-type: none"> <li>1. El usuario pulsa en el ícono de borrar.</li> <li>2. Se eliminar el resumen y la aplicación vuelve a la pantalla principal.</li> </ol>
<b>Excepciones</b>	-
<b>Incluye</b>	-
<b>Extiende</b>	CU-07
<b>Prioridad</b>	Alta.
<b>Frecuencia de uso</b>	Baja.
<b>Importancia</b>	Alta.
<b>Comentarios</b>	-

Tabla B.8: CU-10 Borrar el resumen

## *Apéndice C*

---

# Especificación de diseño

---

## C.1. Introducción

En este apéndice se recogen las características de diseño que implementan los requisitos descritos en el apéndice anterior, a fin de establecer los criterios que se deberán seguir en el desarrollo del sistema.

## C.2. Elección del nombre del proyecto

Para la elección del nombre bajo el cual se englobara todo el servicio de resumen de textos en la nube, se consideraron numerosas opciones, de las cuales, las más relevantes fueron:

- **Summit**: acrónimo procedente de *Summarize it*. Además, «summit» tiene el significado de «cumbre» o «cima» en inglés, sugiriendo el alto rendimiento y potencia del servicio, alcanzando los más *altos* resultados.
- **Jizt**: esta palabra, pronunciada en inglés, suena casi idénticamente a «*gist*», la cual se puede traducir como «esencia» o «quid de la cuestión».
- **Halb**: al revés se lee «*blah*» (de *blah, blah, blah*, en inglés) y en alemán significa «mitad», evocando a la idea de resumen. Además se podría ver como una referencia al supercomputador HAL 9000 de la película y posterior novela *2001: Una Odisea del Espacio*.

De estas tres, se juzgaron más interesantes las dos últimas; la primera resulta menos original debido a que ya existen numerosas empresas, productos y proyectos con este nombre.

En ese momento, se llevó a cabo una pequeña encuesta para determinar el nombre final. Dado que el proyecto aspira a tener un carácter internacional, el objetivo de la encuesta no era tanto obtener un gran numero de votaciones, sino que las que tuviéramos fueran muy variada: en total se preguntó a 22 personas de 15 nacionalidades diferentes.

A continuación, se incluye una tabla resumiendo los resultados de la encuesta:

<b>Elección de Nombre</b>		
	<i>jizt</i>	<i>halb</i>
España	2	3
Alemania	3	0
México	1	1
Taiwán	1	0
Japón	0	1
Escocia	0	1
China	0	1
Francia	0	1
Turquía	0	1
Vietnam	1	0
Hungría	1	0
Bielorrusia	1	0
Corea del Sur	1	0
Malasia	1	0
Serbia	1	0
<b>Total</b>	<b>13</b>	<b>9</b>

Tabla C.1: Resultados de la encuesta.

Aunque los resultados de la encuesta estuvieron bastante ajustados, finalmente la balanza se decantó del lado de JIZT. Algunos de los participantes destacaron su «sonido vibrante y moderno». Por otro lado, el término Halb pareció no convencer a los alemanes, quienes la veían como una palabra demasiado convencional (recordamos que en alemán tiene el significado de «mitad»).

## C.3. Diseño de datos

JIZT comprende dos dominios independientes: el *backend* y el *frontend*. El primero, el *backend*, incluye la API REST, así como los diferentes microservicios que hacen posible la generación de resúmenes en la nube. A su vez, el *frontend* está conformado por la aplicación multiplataforma que se comunica con el *backend* para proporcionar los resúmenes a los usuarios.

Dado que, como decíamos, el diseño de datos es independiente en el caso del *backend* y del *frontend* (siempre que se introduzcan mecanismos para transformar los datos de un dominio al otro, el cual es nuestro caso), atenderemos a cada uno de estos diseños por separado.

### ***Backend***

El *backend* sigue una arquitectura de microservicios. Debido a la propia naturaleza de esta arquitectura, cada uno de estos microservicios tiene una implementación independiente del resto.

Por tanto debemos atender a las clases y sus relaciones en cada microservicio por separado. No obstante, existen dos clases que van a aparecer en todos los microservicios. Se trata de las clases relacionadas con Kafka, esto es, el consumidor y el productor<sup>1</sup>:

- **Productor de Kafka (*Producer*)**: esta clase contiene la lógica que posibilita el envío los mensajes correspondientes a uno o varios Kafka *topics*. Los Kafka *topics* concretos a los que envía el mensaje varían en función del microservicio.
- **Consumidor de Kafka (*Consumer*)**: un consumidor presta atención constante a los *topics* a los que está suscrito, y en el caso de que se haya producido algún mensaje, lo consume. Los Kafka *topics* concretos a los que un consumidor está suscrito varían en función del microservicio.

Una vez introducidas estas dos clases comunes, veamos las clases concretas de cada microservicio.

---

<sup>1</sup> Para más información sobre Kafka, se recomienda acudir al capítulo de «Técnicas y herramientas» de la Memoria.

## Microservicio *Dispatcher*

El *Dispatcher* se encarga de recibir las peticiones de los clientes, validarlas, y reenviarlas hacia el microservicio correspondiente. Como tal, implementa la API REST.

Este microservicio cuenta con las siguientes clases:

- **Servicio *Dispatcher* (*DispatcherService*)**: clase principal del microservicio. Dirige las interacciones entre las instancias del resto de clases.
- **Resumen (*Summary*)**: se trata de la representación de un resumen, por lo que es una de las clases centrales en todo el proyecto. Entre sus campos más importantes se encuentran su *id*, el cual permite identificar inequívocamente un resumen, así como el texto del propio resumen, y el texto fuente a partir del cual se ha generado el resumen.
- **Interfaz Resumen DAO (*SummaryDAOInterface*)**: interfaz DAO (*Data Access Object*) para abstraer el acceso a los datos independientemente de la base de datos empleada.
- **Factoría Resumen DAO (*SummaryDAOFactor*)**: clase que recoge las instancias particulares de los DAOs correspondientes cada base de datos.
- **PostgreSQL DAO Resumen (*SummaryDAOPostgresql*)**: DAO concreto para la base de datos PostgreSQL. esta base de datos almacena los resúmenes generados.
- **Esquema para Petición de Texto Plano (*PlainTextRequestSchema*)**: esta clase contiene la estructura (campos) que debe seguir el cuerpo de las peticiones HTTP de los clientes a la hora de realizar una operación POST sobre la API REST. Se incluye la distinción de «texto plano», dado que en un futuro se podrán realizar peticiones enviando una URL o un documento, y por lo tanto necesitaremos otro esquema diferente para cada uno de estos casos. Este esquema se emplea también para los mensajes producidos por el *Dispatcher* al *topic* del *Pre-procesador* (el siguiente microservicio).
- **Esquema para Respuesta (*ResponseSchema*)**: estructura (campos) del cuerpo de la respuesta HTTP de la API REST ante una petición POST o GET por parte del cliente. Contiene los detalles del resumen producido.

- **Esquema para Mensajes Consumidos**

(*TextPostprocessingConsumedMsgSchema*): estructura (campo) que presenta un mensaje consumido por el *Dispatcher*. Estos mensajes procederán del *topic* del microservicio Post-procesador de textos.

### Diagrama de clases

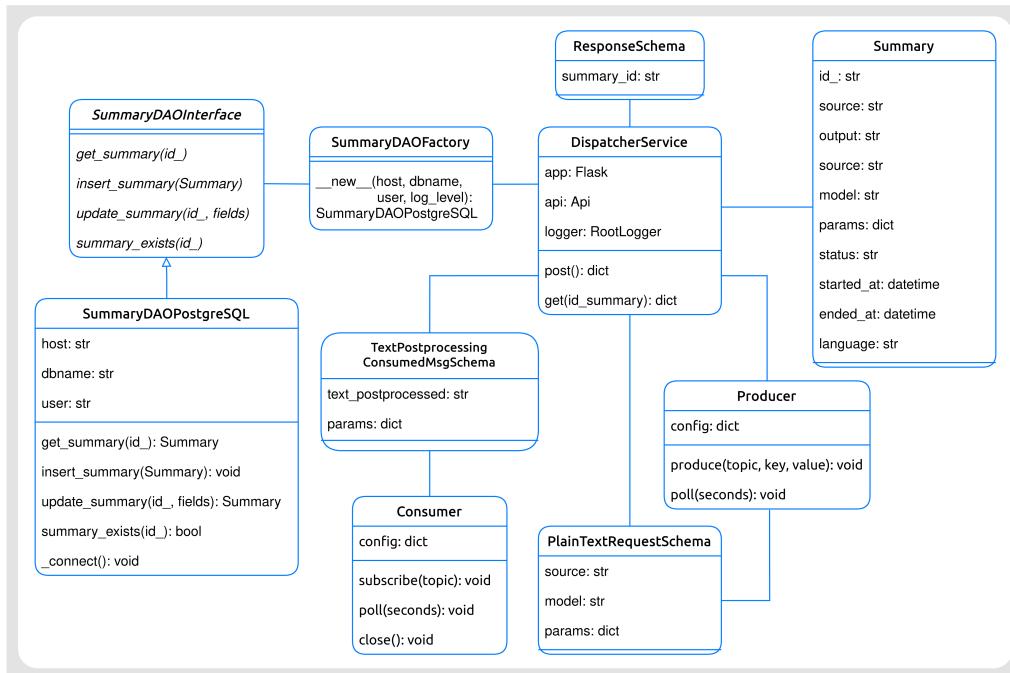


Figura C.1: Diagrama de clases del *Dispatcher*.

### Diagrama E/R de la base de datos

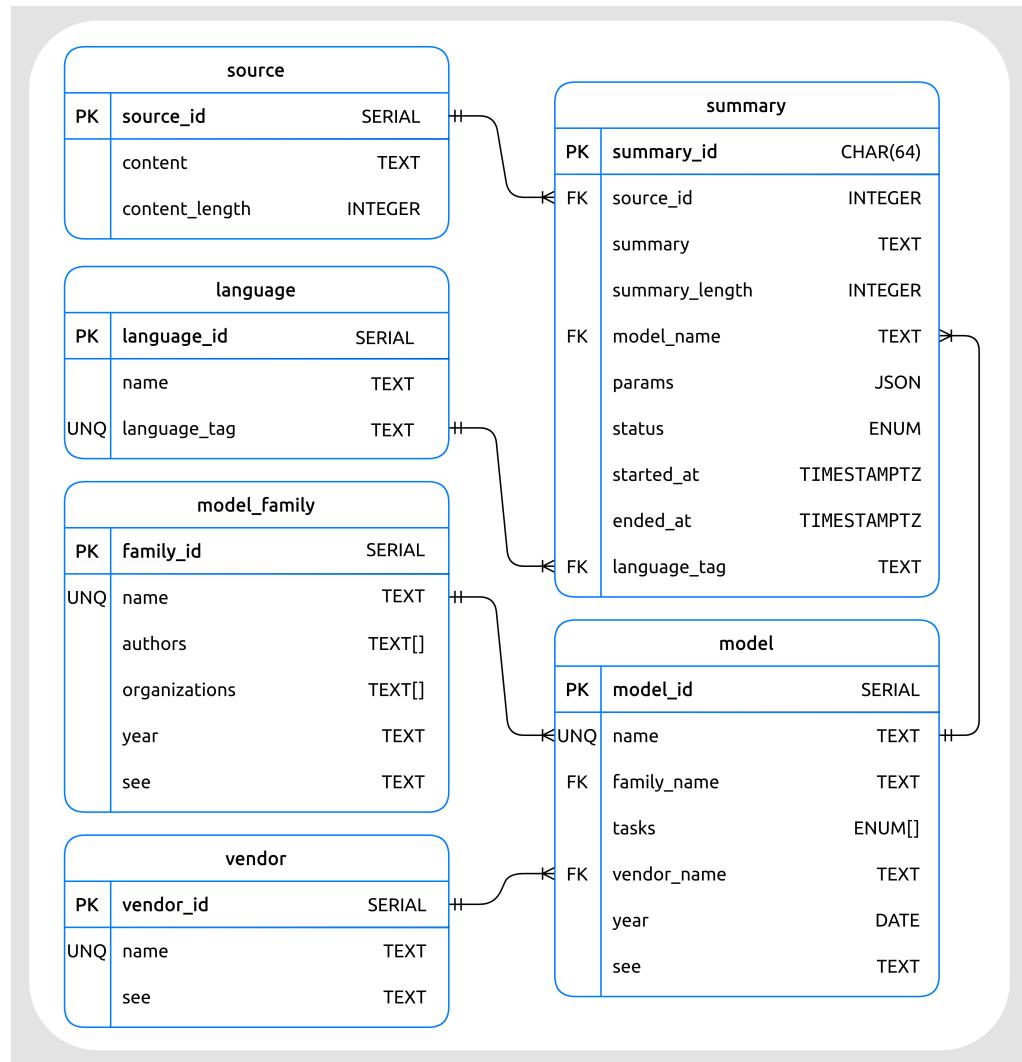


Figura C.2: Diagrama E/R de la base de datos (tipos de datos de PostgreSQL).

### Microservicio Pre-procesador de textos

El Pre-procesador se encarga de realizar un primer procesado de los textos de entrada, a fin de que sean lo más cercanos como sea posible a la entrada que espera el modelo generador de resúmenes.

Este microservicio está compuesto por siguientes clases:

- **Pre-procesador de textos (*TextPreprocessor*)**: esta clase es la encargada de realizar el pre-procesado del texto.
- **Esquema para Mensajes Consumidos (*TextPreprocessingConsumedMsgSchema*)**: estructura (campos) que presenta un mensaje consumido por el Pre-procesador. Estos mensajes procederán del microservicio *Dispatcher*.
- **Esquema para Mensajes Producidos (*TextEncodingProducedMsgSchema*)**: estructura (campos) que presenta un mensaje producido al *topic* del siguiente microservicio (el Codificador de textos).

### Diagrama de clases

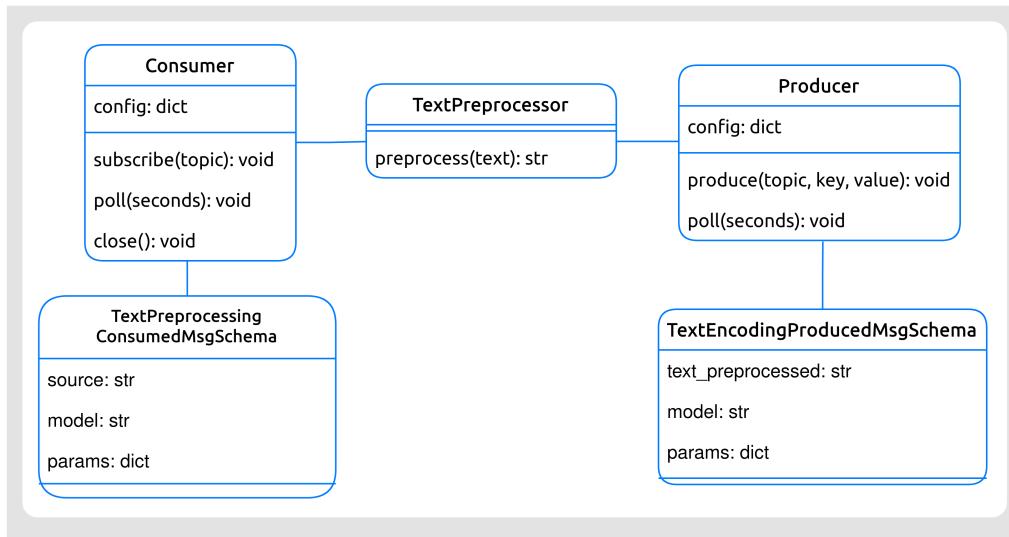


Figura C.3: Diagrama de clases del Pre-procesador de textos.

## Microservicio Codificador de textos

Este microservicio se encarga de codificar el texto y dividirlo en fragmentos menores, a fin de respetar la longitud máxima de entrada del modelo generador de resúmenes.

El Codificador cuenta con las siguientes clases:

- **Codificador y divisor de textos (*SplitterEncoder*)**: esta clase es la encargada de realizar el codificado y división del texto.

- **Esquema para Mensajes Consumidos**  
**(*TextEncodingsConsumedMsgSchema*)**: estructura (campos) que presenta un mensaje consumido por el Codificador. Estos mensajes procederán del microservicio Pre-procesador de textos.
- **Esquema para Mensajes Producidos**  
**(*TextSummarizationProducedMsgSchema*)**: estructura (campos) que presenta un mensaje producido al *topic* del siguiente microservicio (el Generador de resumen).

### Diagrama de clases

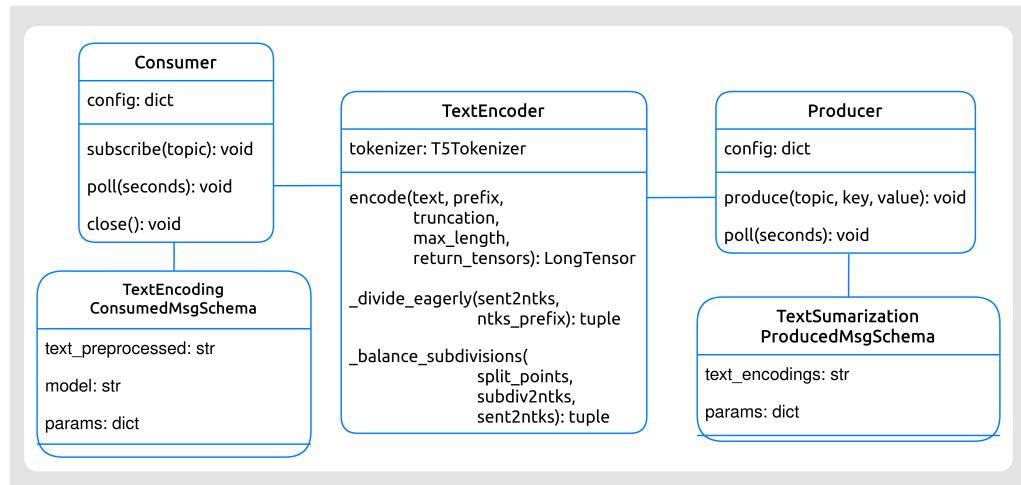


Figura C.4: Diagrama de clases del Codificador de textos.

### Microservicio Generador de resúmenes

Este microservicio se encarga de generar resúmenes a partir del texto codificado que recibe del Codificador.

Las clases del Generador de resúmenes son:

- **Generador de resúmenes (*Summarizer*)**: esta clase es la encargada de generar los resúmenes.
- **Esquema para Mensajes Consumidos**  
**(*TextSummarizationConsumedMsgSchema*)**: estructura (campos) que presenta un mensaje consumido por el Generador de resúmenes. Estos mensajes procederán del microservicio Codificador de textos.

- **Esquema para Mensajes Producidos** (*TextPostprocessingProducedMsgSchema*): estructura (campos) que presenta un mensaje producido al *topic* del siguiente microservicio (el Post-procesador de textos).

### Diagrama de clases

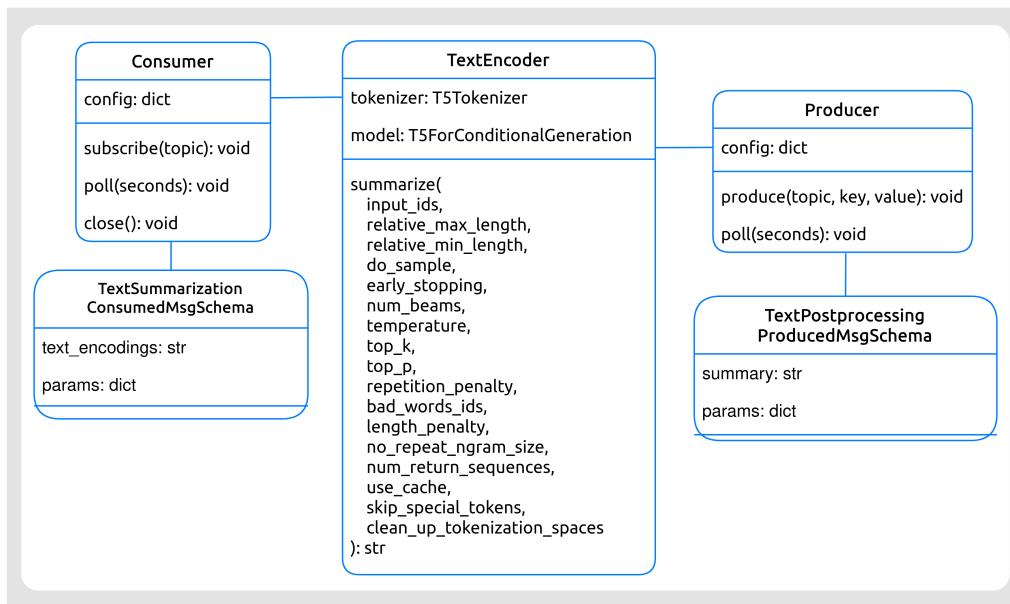


Figura C.5: Diagrama de clases del Generador de resúmenes.

## Microservicio Post-procesador de textos

Este microservicio lleva a cabo el post-procesado del resumen entregado por el Generador de resúmenes.

El Post-procesador de textos tiene las siguientes clases.

- **Post-procesador de textos** (*TextPostprocessor*): esta clase es la encargada de post-procesar el texto.
- **Esquema para Mensajes Consumidos** (*TextPostprocessingConsumedMsgSchema*): estructura (campos) que presenta un mensaje consumido por el Post-procesador de textos. Estos mensajes procederán del microservicio Generador de resúmenes.

- **Esquema para Mensajes Producidos (*ReadyProducedMsgSchema*):** estructura (campos) que presenta un mensaje producido al siguiente *topic*, *Ready*.

### Diagrama de clases

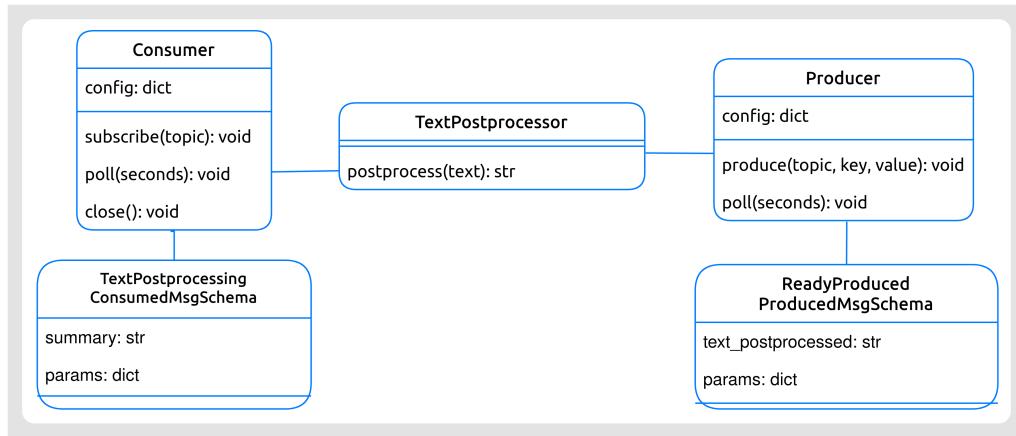


Figura C.6: Diagrama de clases del Post-procesador de textos.

## *Frontend*

TODO

## C.4. Diseño procedimental

TODO

## C.5. Diseño arquitectónico

Dado que el diseño arquitectónico conforma uno de los principales aspectos a destacar de todo el proyecto, se ha incluido de manera detallada en la Memoria.

## *Backend*

Por tanto, en esta sección llevaremos a cabo un resumen de los principales puntos de la arquitectura, e incluiremos información adicional acerca de los

retos que surgieron, y cómo se resolvieron, dando con el diseño arquitectónico final.

Como se indica en la Memoria, para el *backend* se ha desarrollado una **arquitectura de microservicios**, a través de la cual podemos separar cada paso en la generación de resúmenes (pre-procesado, codificación, generación del resumen, post-procesado), en módulos independientes, favoreciendo los siguientes aspectos:

- Gracias a la división en microservicios, conseguimos una gran flexibilidad; los cambios en uno de los pasos en la generación del resumen no influyen al resto.
- Facilita la detección y corrección de cuellos de botella en el proceso, dado que podemos monitorizar de forma precisa el rendimiento de cada microservicio.
- La arquitectura es fácilmente escalable tanto en términos de los microservicios ya existentes, aumentando el número de réplicas de cada microservicio (permitiendo la generación en paralelo de varios resúmenes), como en términos de la adición de nuevos microservicios, por ejemplo, diferentes modelos para distintos idiomas.
- Asegura una alta disponibilidad, ya si uno de los microservicios falla, el sistema crea una nueva instancia y finaliza el microservicio defectuoso. Adicionalmente, con una arquitectura de microservicios eliminamos cualquier posible punto de único fallo (*single point of failure*).

Otro de los principales aspectos de la arquitectura es cómo se lleva a cabo la comunicación y el correcto enrutado de los mensajes entre los diferentes microservicios. Al tratarse de un proceso secuencial, la salida de un microservicio será la entrada del siguiente.

Inicialmente, se pensó en resolver esta situación mediante el patrón de *routing-slips* (hojas de ruta) [11]. Con este patrón se incluye en el propio mensaje la ruta que este debe seguir, por lo que, implementando un *router* en cada microservicio que interpretara la hoja de ruta, podríamos, resolver el problema del enrutado. En cuanto a cómo se llevaría a cabo la comunicación, esta se podría hacer mediante peticiones HTTP, de forma que cada microservicio implementara su propia REST API.

Siguiendo esta estrategia, junto con el patrón de API *gateway*, a través que se ofrece un punto de entrada al *backend*, el diseño de la arquitectura quedaba del siguiente modo:

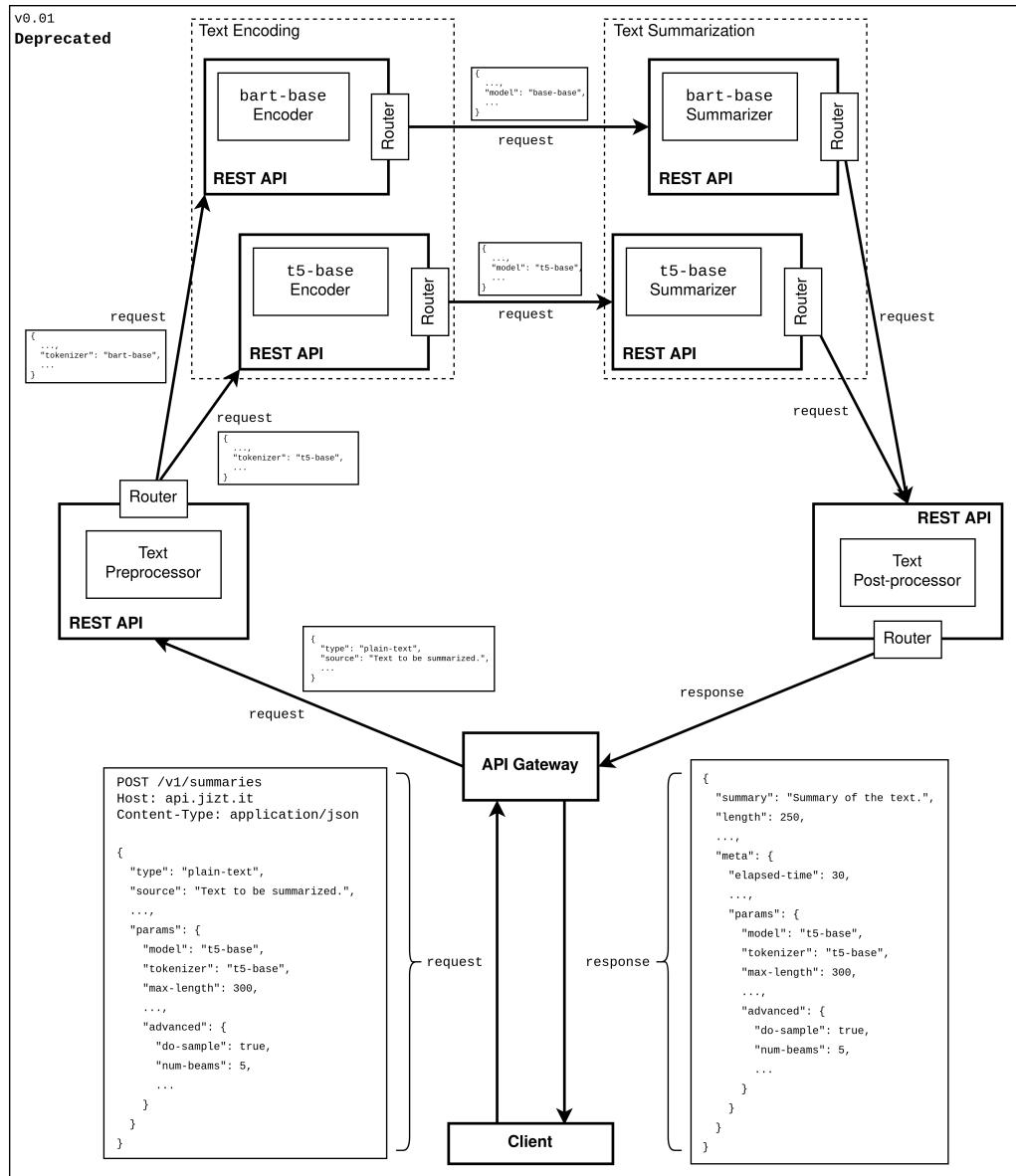


Figura C.7: Primera aproximación para el diseño arquitectónico del *backend*.

Poco después, conocimos acerca de la **arquitectura dirigida por eventos** [12], y de Kafka [13], una de las herramientas más apropiadas y avanzadas a día de hoy para este tipo de arquitectura<sup>2</sup>.

<sup>2</sup> De nuevo, referimos al lector a la Memoria, donde se recogen las principales ventajas, tanto de este patrón arquitectónico, como de Kafka.

Con este nuevo diseño, la arquitectura se simplificaba en gran medida, y problemas como el escalado, o la entrega fiable de mensajes, corrían a cargo de Kafka, quienes gestionaba estos y otros aspectos de manera automática.

La arquitectura final del *backend*, por tanto, tiene el siguiente aspecto:

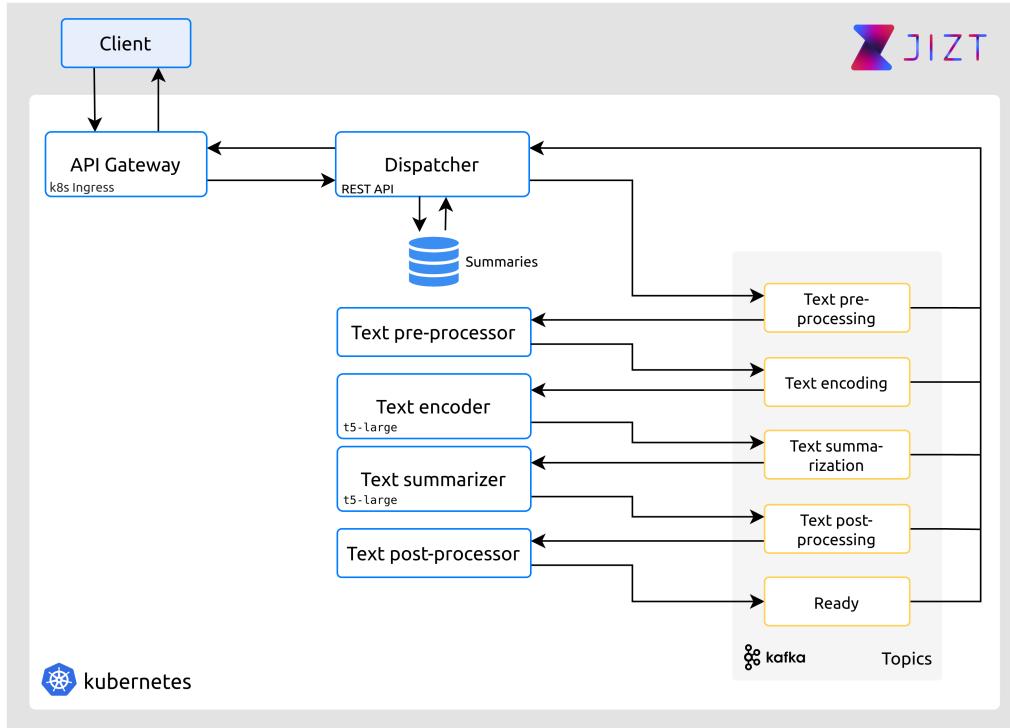


Figura C.8: Diseño final de la arquitectura del *backend*.

## Frontend

Gracias a la vibrante comunidad de Flutter [4], dar con la arquitectura más apropiada para la aplicación resultó un proceso más fluido.

Antes de comenzar este proyecto, habíamos oído del concepto de *Clean Architecture* [14], aunque no habíamos indagado muy en profundidad en sus proposiciones. No obstante, en nuestra formación de Flutter, apareció de nuevo, e incluso supimos que existe un paquete para Flutter que simplifica la implementación de este patrón [15].

A continuación, nos informamos sobre el patrón BLoC [16], muy popular también dentro de la comunidad Flutter. Por suerte, también existe un paquete para la implementación de este patrón [0].

Finalmente, la arquitectura quedó de la siguiente forma:

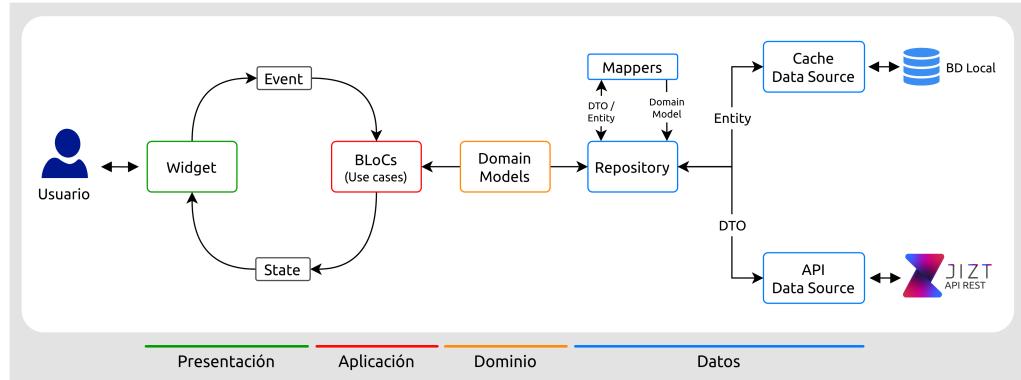


Figura C.9: Diseño final de la arquitectura de la aplicación.

La información completa acerca de la arquitectura de la aplicación se encuentra, asimismo, en la Memoria.

## C.6. Diseño de interfaces

### Diseño del logo

Una de las intenciones detrás de este proyecto siempre ha sido tratar de crear una cierta imagen corporativa y de producto. Por ello, el diseño de nuestra carta de presentación, es decir, nuestro logo, ocupó un papel central en las primeras iteraciones del proyecto.

En la búsqueda creativa de un logo atractivo, moderno y memorable, se experimentó con numerosos posibles diseños en papel.

Finalmente, dimos con un diseño que parecía tener potencial. Corrimos hacia el ordenador, y nos sumergimos en Adobe Illustrator, un editor de gráficos vectoriales muy popular. Por experiencia previa en diseño gráfico (autodidacta), sabemos que, teniendo una buena idea de partida, «el resto está hecho».

Así pues, el logo final, nuestra carta de presentación, acabó luciendo del siguiente modo:

Se trabajó también sobre una serie de variaciones, de forma que el logo se pudiera ajustar a cada situación de color:

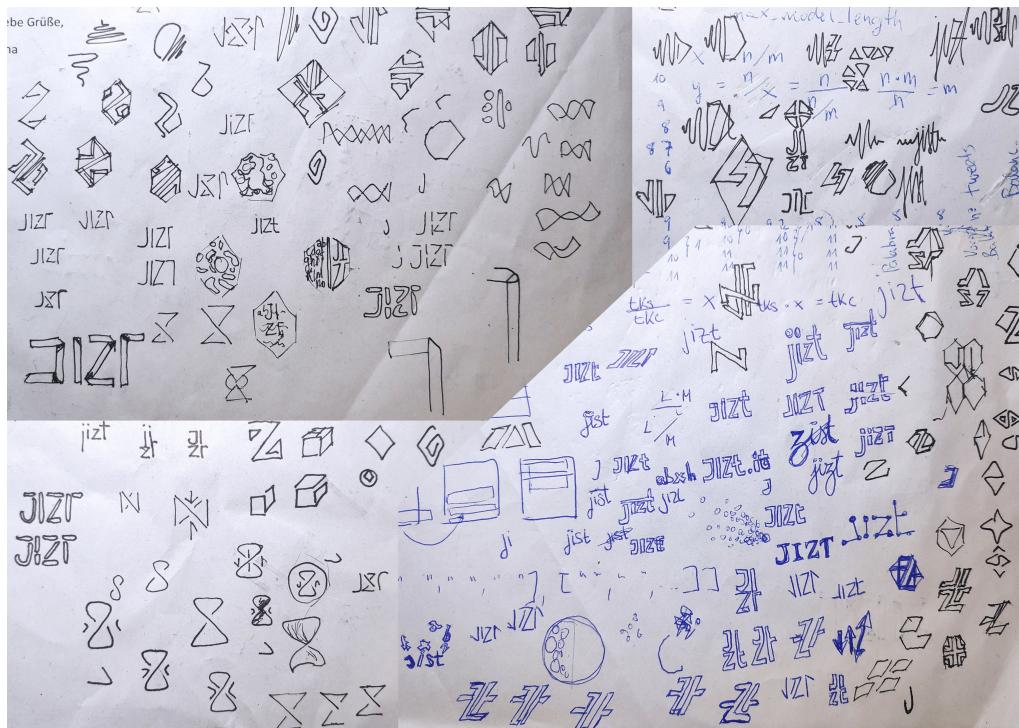


Figura C.10: Ideas, ideas, y más ideas. Pero solo unas pocas buenas.



Figura C.11: JIZT: generación de resúmenes mediante IA.

### Diseño de la interfaz gráfica de la aplicación

Para el diseño de la interfaz gráfica de usuario de la aplicación, trabajamos directamente sobre el ordenador, esta vez con el programa Inkscape, también editor de gráficos vectoriales, pero en este caso *open-source* y gratuito.

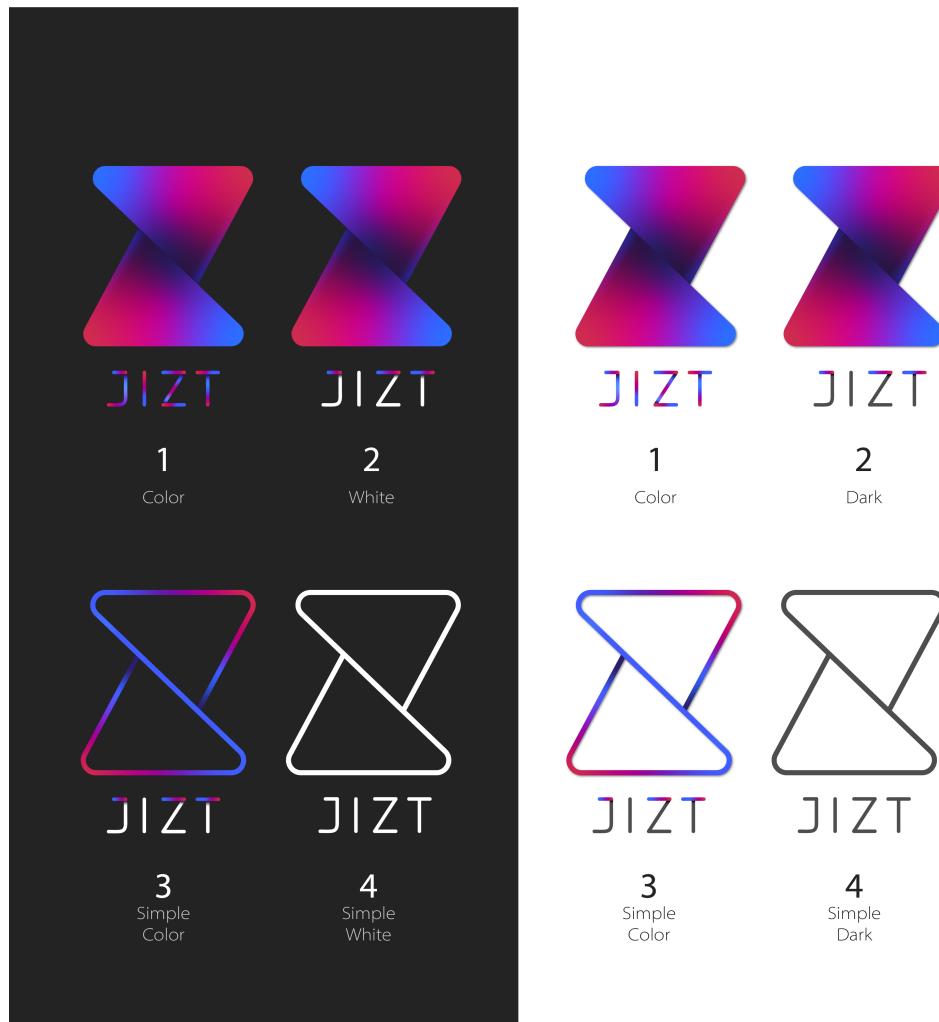


Figura C.12: Variaciones sobre el logo de JIZT.

Se llevaron a cabo diferentes iteraciones hasta dar con un diseño que nos acabó convenciendo.

Siguiendo el estilo de la pantalla principal, se diseñaron el resto de pantallas:

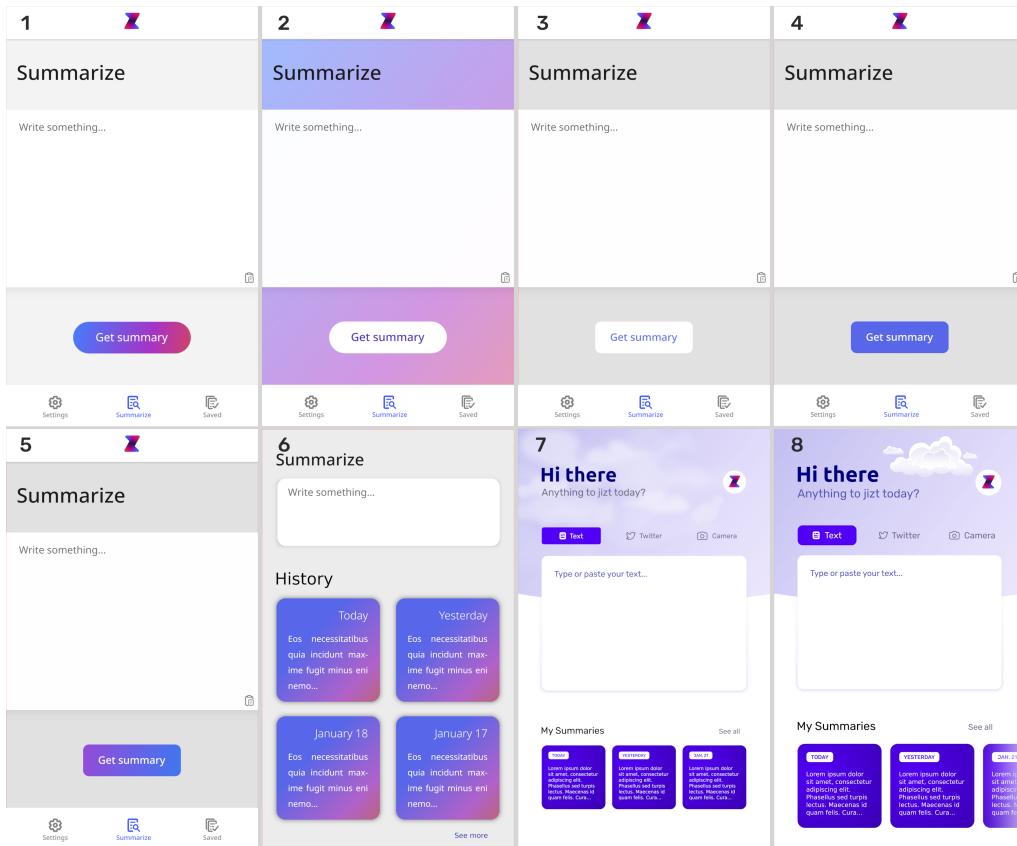


Figura C.13: Iteraciones sobre la pantalla principal.

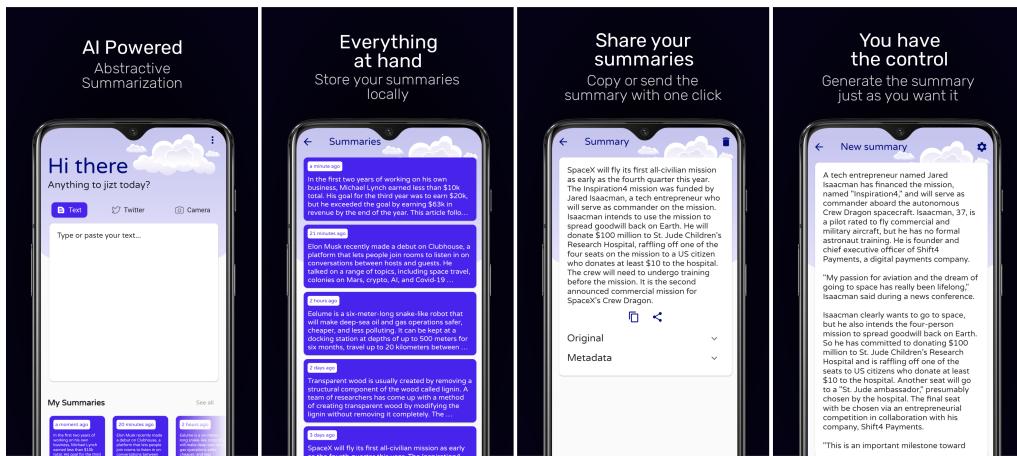


Figura C.14: Capturas de pantalla de la aplicación para su publicación en Play Store.



*Apéndice D*

---

## **Documentación técnica de programación**

---

- D.1. Introducción**
- D.2. Estructura de directorios**
- D.3. Manual del programador**
- D.4. Compilación, instalación y ejecución  
del proyecto**
- D.5. Pruebas del sistema**



*Apéndice E*

---

## **Documentación de usuario**

---

- E.1. Introducción**
- E.2. Requisitos de usuarios**
- E.3. Instalación**
- E.4. Manual del usuario**



---

## Bibliografía

---

- [1] Hugging Face. *Transformers*. Sep. de 2020. URL: <https://huggingface.co/transformers/index.html>. Último acceso: 08/02/2021.
- [2] Kubernetes. *Ingress*. Feb. de 2021. URL: <https://kubernetes.io/docs/concepts/services-networking/ingress>. Último acceso: 08/02/2021.
- [3] Google Cloud. *Google Kubernetes Engine (GKE)*. Oct. de 2020. URL: <https://cloud.google.com/kubernetes-engine>. Último acceso: 08/02/2021.
- [4] Flutter. *Flutter - Hermosas apps nativas en tiempo record*. Sep. de 2020. URL: <https://esflutter.dev>. Último acceso: 08/02/2021.
- [5] David J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, abr. de 2010. ISBN: 0984521402.
- [6] Juan Antonio Pérez López. *Fundamentos de la dirección de empresas*. RIALP, abr. de 2018. ISBN: 9788432149184.
- [7] Agencia Tributaria. *Cuadro informativo tipos de retención aplicables*. 2021. URL: [https://www.agenciatributaria.es/static\\_files/Sede/Programas\\_ayuda/Retenciones/2021/CUADRO\\_TIPOS\\_RETENC\\_IRPF21.doc](https://www.agenciatributaria.es/static_files/Sede/Programas_ayuda/Retenciones/2021/CUADRO_TIPOS_RETENC_IRPF21.doc). Último acceso: 08/02/2021.
- [8] Agencia Estatal Boletín Oficial del Estado. *Boletín Oficial del Estado, Núm. 341*. Dic. de 2020. URL: <https://www.boe.es/eli/es/1/2020/12/30/11/dof/spa/pdf>. Último acceso: 08/02/2021.
- [9] The GNU Operating System y the Free Software Movement. *GNU General Public License v3.0*. Jun. de 2007. URL: <https://www.gnu.org/licenses/gpl-3.0.en.html>. Último acceso: 09/02/2021.

- [10] The GNU Operating System y the Free Software Movement. *GNU Free Documentation License*. Nov. de 2008. URL: <https://www.gnu.org/licenses/fdl-1.3.html>. Último acceso: 09/02/2021.
- [11] Codit. *The Routing Slip Pattern*. Jul. de 2019. URL: <https://www.codit.eu/blog/the-routing-slip-pattern>. Último acceso: 10/02/2021.
- [12] Wikipedia - La enciclopedia libre. *Arquitectura dirigida por eventos*. Ago. de 2020. URL: [https://es.wikipedia.org/wiki/Arquitectura\\_dirigida\\_por\\_eventos](https://es.wikipedia.org/wiki/Arquitectura_dirigida_por_eventos). Último acceso: 10/02/2021.
- [13] Apache Software Foundation. *Apache Kafka*. Nov. de 2020. URL: <https://kafka.apache.org>. Último acceso: 10/02/2021.
- [14] Robert Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson Education, 2015. ISBN: 9780134494166.
- [15] Shady Boukhary y Rafael Monteiro. *Flutter Clean Architecture Package*. Ene. de 2021. URL: [https://pub.dev/packages/flutter\\_clean\\_architecture](https://pub.dev/packages/flutter_clean_architecture). Último acceso: 07/02/2021.
- [16] Didier Boelens. *Reactive Programming - Streams -BLoC*. Ago. de 2018. URL: <https://www.didierboelens.com/2018/08/reactive-programming-streams-bloc>. Último acceso: 10/02/2021.
- [0] Felix Angelov. *Bloc Package*. Dic. de 2020. URL: <https://pub.dev/packages/bloc>. Último acceso: 10/02/2021.