



**UNIVERSIDAD DE JAÉN**  
*Nombre del Centro*

Trabajo Fin de Grado

# **GENERADOR AUTOMÁTICO DE POESÍA EN ESPAÑOL**

**Alumno: Manuel García López**

Tutor: Prof. D. Arturo Montejo Ráez  
Dpto: Informática

**Junio, 2019**

MANUEL GARCÍA LÓPEZ

GENERADOR AUTOMÁTICO DE POESÍA  
EN ESPAÑOL



Universidad de Jaén  
Escuela Politécnica Superior de Jaén  
Departamento de Informática

Don Arturo Montejo Ráez , tutor del Proyecto Fin de Carrera titulado: Generación Automática de Poesía en Español, que presenta Manuel García López, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2019

El alumno:

Manuel García López

Los tutores:

Arturo Montejo Ráez

MANUEL GARCÍA LÓPEZ

GENERADOR AUTOMÁTICO DE POESÍA  
EN ESPAÑOL

## Agradecimientos

*En primer lugar a mi familia, por la educación recibida a lo largo de tantos años, por apoyarme en todo lo necesario y por seguir apostando porque yo pudiera alcanzar todos mis objetivos y sueños, donde otros no hubieran dado ni la mitad de lo que ellos han dado. Si he llegado hasta aquí, ha sido por ellos.*

*A Javier Hortelano y Alejandro Ureña, por las interminables horas de estudio y de prácticas juntos, por su incondicional ayuda y su determinación y esfuerzo para lograr sacar todas las prácticas, retos y tareas que tanto la vida como la carrera nos ha impuesto. Sin ellos, este viaje por el grado no hubiera sido igual.*

*A mis amigos de toda la vida, por estar siempre a mi lado, aconsejándome y apoyándome de forma desinteresada e incondicional, por poder contar en todo momento con ellos. Sé que la vida aún nos guarda muchas más aventuras.*

*A todos mis amigos y compañeros de Jaén y de tantos lugares del mundo que he conocido estos años. Gracias por haber participado en mi viaje y haberme enseñado tantas cosas nuevas. Espero que sigan haciéndolo.*

*Por último a todos los profesores de la Universidad de Jaén y de la Universidad de Umeå, por haber participado de mi formación académica durante todo este tiempo. Haciendo especial mención para Ángel Luis García Fernández por haberme ayudado junto a mi familia a cumplir satisfactoriamente ese sueño llamado Erasmus, además de guiarme con gusto y de forma desinteresada especialmente durante los primeros años de carrera; y a Arturo Montejo Ráez, por haber despertado mi curiosidad y pasión por el mundo web, además de haberme prestado atención y ayuda en el trascurso de este proyecto y de cara a decidir mi futuro venidero.*

*A todos, muchas gracias.*

## Índice

<b><u>Agradecimientos.....</u></b>	<b><u>4</u></b>
<b><u>1. Introducción .....</u></b>	<b><u>8</u></b>
1.1 Contextualización del proyecto .....	8
1.2 Propósito general y objetivos .....	9
1.3 Resultados esperados .....	10
<b><u>2. Análisis .....</u></b>	<b><u>11</u></b>
2.1 Problema a abordar .....	11
2.2 Trabajos previos .....	12
2.3 Redes neuronales profundas .....	17
2.4 Técnicas de procesamiento del lenguaje .....	19
2.5 Análisis de objetivos .....	20
2.6 Análisis de requisitos .....	21
2.6.1 Requisitos funcionales .....	22
2.6.2 Relación entre objetivos y requisitos funcionales .....	24
2.6.3 Requisitos no funcionales .....	25
2.7 Análisis de casos de uso .....	26
2.7.1 Presentación de casos de uso .....	26
2.7.2 Diagrama de casos de uso .....	27
2.7.3 Definición de casos de uso .....	27
2.7.4 Matriz de trazabilidad .....	37
<b><u>3. Planificación .....</u></b>	<b><u>38</u></b>
3.1 Tareas a realizar .....	38
3.2 Cronograma – Diagrama de Gantt .....	41
3.3 Estimación de costes .....	42
3.3.1 Costes humanos .....	42
3.3.2 Costes hardware y software .....	46
3.3.3 Costes adicionales .....	47

<b>4. Diseño .....</b>	<b>49</b>
4.1 Aplicación web .....	49
4.1.1 Diseño de la estructura de clases .....	49
4.1.2 Diseño de la interfaz .....	52
4.1.2.1 Estilo de la interfaz .....	52
4.1.2.2 Diseño de pantallas .....	54
4.1.2.3 Storyboards o Mockups .....	55
4.1.2.4 Menú de herramientas .....	64
4.2 Diseño de la base de datos y elección de tecnologías del lenguaje .....	68
4.2.1 Entidades intervinientes en nuestro modelado de datos .....	71
4.2.2 Relaciones intervinientes en nuestro modelado de datos .....	71
4.2.3 Tablas de datos .....	72
4.2.4 Justificación de la elección de MongoDB .....	72
4.2.5 Estudio de volumetría .....	75
4.2.5.1. Memoria necesaria por registro y por tabla .....	75
4.3 Arquitectura de la red neuronal .....	76
4.3.1 Dibujo orientativo de funcionamiento .....	79
<b>5. Implementación .....</b>	<b>81</b>
5.1 Entorno de programación .....	81
5.2 Controladores y tabla de acciones .....	82
5.3 Front-end .....	87
5.3.1 Vistas .....	87
5.3.2 Layout .....	88
5.3.3 Campos de texto .....	88
5.3.4 Menú de herramientas .....	89
5.4 Back-end .....	90
5.4.1 Controladores .....	90
5.4.2 Modelo y operaciones con la base de datos .....	90
5.5 Conexión Front-end – Back-end .....	90
5.5.1 Conexión a la base de datos .....	91
5.5.2 Intercambio de información entre front-end y back-end .....	91
5.6 Redes neuronales implementadas .....	91

<b>6. Validación y evaluación .....</b>	<b>94</b>
6.1 Requisitos del sistema .....	94
6.1.1 Librerías, paquetes y versiones necesarias para la ejecución de la aplicación .....	94
6.2 Justificación de los parámetros de ejecución de la red neuronal .....	95
6.3 Ejemplos de salida .....	108
6.4 Plan de pruebas de la web .....	109
6.4.1 Condiciones del equipo de pruebas .....	110
6.4.2 Pruebas para cada caso de uso .....	110
6.5 Experiencia de usuario .....	111
6.6 Gestión de errores .....	114
6.7 Seguridad .....	114
<b>7. Manual de usuario .....</b>	<b>117</b>
7.1 Instalación y puesta en marcha .....	117
7.2 Navegar por la web .....	118
7.2.1 Introducción .....	118
7.2.2 Terminología .....	118
7.2.3 Secciones de la aplicación .....	120
<b>8. Conclusiones .....</b>	<b>126</b>
8.1 Reflexiones finales .....	126
8.2 Revisión de lo aprendido .....	127
8.3 Posibles mejoras .....	127
<b>9. Bibliografía .....</b>	<b>130</b>
Anexo I Índice de tablas .....	130
Anexo II Índice de figuras .....	132
Anexo III Enlaces bibliográficos .....	134



## 1. Introducción

Para comenzar la memoria, vamos a dar una visión global del proyecto a abordar, así como una idea general de en qué consiste y la motivación que lleva a hacer el mismo.

### 1.1 Contextualización del proyecto

La tecnología y especialmente la informática, están sufriendo un auge espectacular en la últimas décadas. Tanto es así, que gracias al desarrollo de componentes y máquinas más potentes, así como la evolución en el ámbito de la investigación, la cual está consiguiendo técnicas en el procesado de la información cada vez más eficientes y complejas, se están logrando en términos computacionales y de tiempo real una serie de hazañas impensables hace apenas unos años.

Un gran ejemplo de ello es el procesamiento del lenguaje natural y la aplicación de redes neuronales profundas para el estudio y comprensión de la lengua hablada y escrita.

En este proyecto concreto, abordaremos mediante técnicas de procesamiento del lenguaje natural y redes neuronales profundas cómo generar poesía de forma automática en base a unas poesías pre-establecidas como datos de entrenamiento, de una serie de autores a elegir por el usuario.

#### ***- Premisas a abordar***

El análisis y generación de un poema no es un problema trivial. Podríamos interpretar el análisis y generación de un poema como si de un texto plano se tratase y sin embargo, estaríamos cometiendo un enorme error, pues un poema se compone de muchos más objetos que un simple texto como sería una redacción o una serie de notas.

Al igual que una carta o un artículo de investigación, un poema es un tipo de texto estructurado, el cual se compone de una serie de elementos los cuales, deben reproducirse de la forma más adecuada posible a la hora de generar nuestro poema.

Dichos elementos se dividen en: título y cuerpo, así como posibles anotaciones del autor y la fecha de creación o publicación del poema.

Dentro de dichos elementos, encontramos un elemento clave, el cuerpo del poema. Y es que es en el cuerpo del poema, donde reside la especial dificultad de la generación automática de poesía.

La estructura y forma que debe adquirir el cuerpo vienen dadas tanto por la métrica como por el idioma en el que se escribe dicho poema.

Simplificar el problema ignorando la necesidad de tener un problema de métrica perfecta y sin mezcla de idiomas, facilitará enormemente la generación de poemas y será la línea que seguiremos durante el desarrollo de este proyecto: Generación automática de poesía en español sin adecuación métrica.

No obstante, vamos a desarrollar brevemente, aunque sin entrar al más mínimo detalle, el porqué la formación de una correcta métrica, implica un complejo y profundo procesado del texto, así como implicaría un profundo sesgo en los contenidos que tenemos de cada autor, reduciendo así la eficacia del entrenamiento de la inteligencia artificial que desarrollaremos para llevar a cabo nuestra misión.

Como ya hemos dicho, analizar y generar la métrica correctamente requiere de un conocimiento del lenguaje interviniente, y de la estructura silábica del poema. A su vez, esta estructura silábica, depende de los juegos del lenguaje de cada escritor, y de la fonética del lenguaje. Por lo que, sin conocer el lenguaje a analizar, es casi imposible establecer una métrica de forma correcta. Así mismo, existe el problema de las métricas regionales, las cuales son particulares de cada país o región, a veces pareciéndose bastante entre sí, y haciendo más complicado el análisis correcto de las mismas, para una posterior aplicación en la generación de nuevos poemas que sigan los cánones métricos adecuados y deseables en todo momento.

## **1.2 Propósito general y objetivos**

De esta forma el propósito general del proyecto, consiste en proporcionar una aplicación capaz de generar texto de forma automática que se asemeje a la poesía de un autor concreto. De tal forma que esta permita llevar a cabo un estudio experimental sobre la distinta forma de escribir de cada autor. Así mismo, las distintas interacciones con la aplicación en forma de generación de

poemas quedarán reflejadas en una base de datos no relacional, para poder ver visualmente el resultado global de dichas interacciones.

Siendo más precisos se va a realizar una aplicación web (debido a la gran facilidad de uso y de accesibilidad que proporciona Internet y el mundo web) que maneje internamente la inteligencia artificial capaz de generar estos textos de forma automática con aspecto de poesía y la base de datos que almacene la información que se vaya generando.

### **1.3 Resultados esperados**

Al finalizar el desarrollo e implementación de la aplicación, se tratarán de lograr los siguientes objetivos:

1. Una aplicación funcional en la web, que permita de forma sencilla, rápida y eficaz, crear y gestionar un gran volumen de poemas de forma automática, en un tiempo de aprendizaje mínimo por parte del usuario.

#### **- Metodología del proyecto**

1- Análisis: Especificación detallada sobre las necesidades y requisitos a cubrir por el sistema, de cara a cumplir con los objetivos propuestos por el proyecto.

2- Planificación y diseño: Estimación de tareas, esfuerzo y costes, unido a un diseño inicial o *mockup* que especifique y conforme sus diferentes elementos de arquitectura.

3- Implementación: Código fuente detallado de la aplicación final en funcionamiento, y manual de usuario.

4- Validación: Redacción de la memoria, y evaluación de los distintos estudios realizados durante el testeo de la aplicación.

## **2.Análisis**

En este apartado identificaremos y desarrollaremos los aspectos y elementos esenciales que se tendrán en cuenta en fases posteriores como son la planificación, el diseño y la implementación.

### **2.1 Problema a abordar**

La aplicación de generación de poesía debe ser capaz de generar de forma simple y cómoda una poesía automáticamente en base a un autor elegido, y de mostrarla a través de su interfaz web, así como almacenarla y luego volver a indexarla en los resultados de poesía generales.

Por tanto, las funcionalidades básicas con las que la aplicación ha de contar son las siguientes:

- 1- Debe poseer una interfaz clara y sencilla de tipo web, que permita al usuario manejarse por ella sin necesidad de ayuda externa.
- 2- Debe ofrecer una serie de autores en los que basar la poesía que se generará posteriormente de forma automática.
- 3- Debe poder mostrar dicha poesía una vez generada, de una forma clara y legible.
- 4- Debe ser capaz de guardar los poemas generados
- 5- Debe ser capaz de indexar todos los resultados sin parámetros específicos.
- 6- Debe ser capaz de indexar todos los resultados con parámetros específicos.
- 7- Debe ser capaz de mostrar al usuario el resultado de diversas redes neuronales para permitirle comparar y observar diferencias.

## 2.2 Trabajos previos

Antes de comenzar con el análisis de cómo se ha llevado a cabo la implementación de dichos objetivos en nuestro problema a abordar, veamos algunos de los principales trabajos de investigación sobre la generación automática de poesía que ya existen en la comunidad científica.

Como podemos imaginar, la generación automática de poesía es un tema que lleva intentando ser tratado casi desde el comienzo de las técnicas de tratamiento del lenguaje. Donde si bien, no se ha logrado aún alcanzar la excelencia debido a la dificultad que hemos indicado durante la introducción, sí es verdad que se han dado grandes pasos, especialmente en el principal lenguaje de investigación, el inglés, donde ya se ha logrado resolver con cierta suficiencia el aspecto de la correlación métrica entre los versos de los poemas, así como la rima entre los distintos versos, tal y como se ve en trabajos como “Generating topical poetry” de la universidad de California del sur.

El enfoque de la generación automática de poesía suele estar muy ligado al enfoque específico de la Generación Automática del Lenguaje (GAL), un ámbito a su vez propio del Procesamiento del Lenguaje Natural (PLN).

El auge en el interés del estudio de la generación automática del lenguaje, no solo a nivel poético, sino en muchos otros ámbitos como la generación automática de resúmenes o la generación automática de conversaciones a través del uso de asistentes personales virtuales, ha venido de la mano de técnicas de aprendizaje computacional, como Redes Neuronales o Aprendizaje automático.

Estas técnicas nos permiten entrenar modelos de probabilidad estadística, de forma lo suficientemente potente y eficiente como para generar texto de forma relativamente rápida y con una calidad aceptable.

No obstante, existen numerosas formas de abordar la problemática de la generación automática del lenguaje, así como existen diferentes formas de entrenar los modelos que podemos crear gracias a las Redes Neuronales.

En generación del lenguaje automático, el enfoque más habitual suele ser el de la generación de lenguaje mediante Redes Neuronales Recurrentes o Recurrent Neural Networks (RNN). Existen

también otra serie de enfoques como el dado por Redes Neuronales Convolucionales o Convolutional Neural Networks (CNN), o redes de Generación de texto contradictorio, en inglés, Generative Adversarial Networks (GAN).

### **-Recurrent Neural Networks**

Las redes recurrentes neuronales, en especial aquellas del tipo Redes de Memoria a largo y corto plazo o Long short-term memory Networks (LSTMs), generan potentes modelos, que han sido muy exitosos en el modelado secuencial de datos (Mikolov et al, 2010).

Su funcionamiento básico transforma una secuencia de vectores de entrada en una secuencia de estados ocultos, donde cada uno mantiene un resumen de las entradas hasta ese momento. Los modelos del lenguaje utilizados por las RNN son autoregresivos. Sin embargo estos, no transportan información sobre las salidas del sistema en tiempo  $t$ , sino que utilizan las salidas como nuevas entradas de la red en una función de llamada recurrente. De igual modo, una función *softmax*, es aplicada a cada vector de resultados de salida, para transformarlos en un valor probabilístico (Adversarial Generation of Natural Language, 2017).

### **- Convolutional Neural Networks**

Los modelos de redes convolucionales también ofrecen modelos de secuenciación de datos de forma unidimensional bastante satisfactorios (Dauphin et al. 2016; Zhang et al. 2015).

Estos modelos, basan su funcionamiento en filtros de convolución a través de las variables de tiempo y dimensión de entrada, las cuales se tratan como canales. (He et al. 2016).

El uso de estos filtros se realiza en series de 5 bloques con una doble capa unidimensional por bloque. Estas capas transforman las salidas de los bloques en vectores de datos, posteriormente normalizados usando una función *softmax* y una activación *ReLU*. (Gulrajani et al. 2017; Ioffe and Szegedy, 2015; Nair and hinton, 2010).

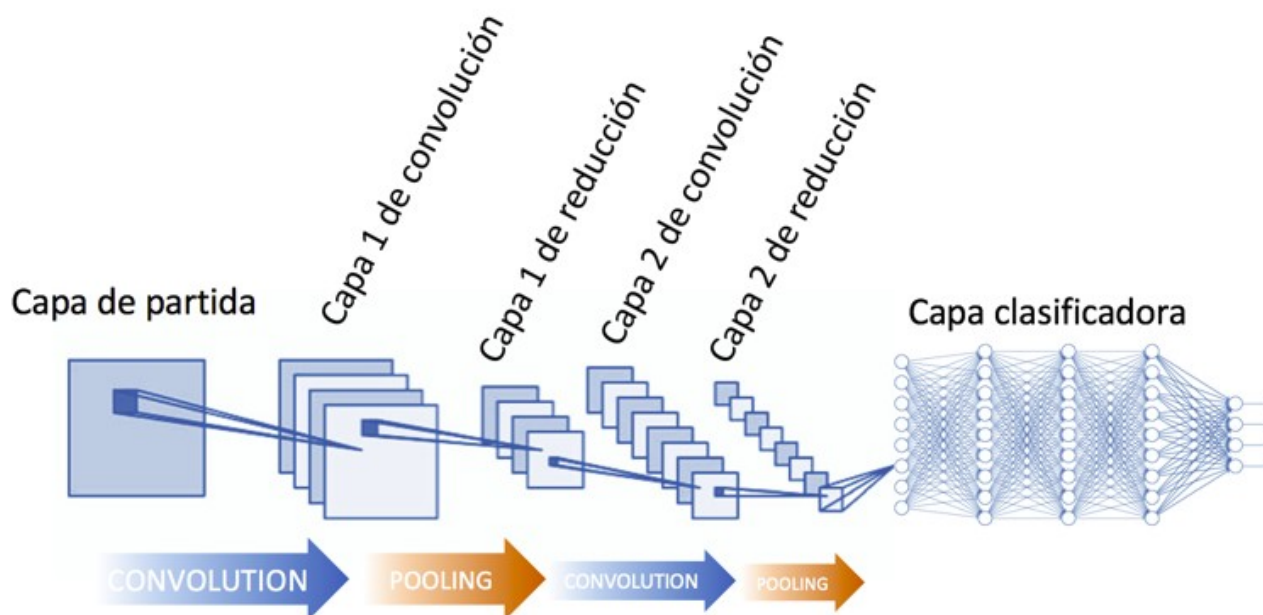


Imagen 2.1. Red neuronal convolucional. Fuente: <http://www.diegocalvo.es/red-neuronal-convolucional/>

### - Generative Adversarial Networks

Este tipo de redes son, en realidad, un marco o una serie de especificaciones técnicas usadas a la hora de entrenar modelos contradictorios formulados por un proceso de aprendizaje como si de un juego *minimax* de dos jugadores se tratase. Un generador  $G$ , trata de generar ejemplos lo más parecidos posibles a los datos reales. Minimizando un discriminador logarítmico, la red es entrenada para diferenciar los datos reales de los datos generados usando *minimax*. El generador es entrenado para maximizar dicho logaritmo, ya que consigue resultados mejores en etapas de entrenamiento mas tempranas. (Goodfellow et al. 2014)

Una vez teniendo en cuenta las distintas formas planteadas para la generación automática del lenguaje, veamos qué trabajos se han llevado a cabo para su aplicación específica en cuanto la generación automática de poesía.

Tal y como ya hemos indicado en la parte introductoria, la especial complejidad de un poema, reside entorno a la métrica y a la rima. (Marjan Ghazvininejad et al, 2016) nos da un esbozo aproximado de qué enfoque es el más estudiado actualmente para abordar esta problemática.

Dividiendo las diferentes tareas a realizar en:

- 1- Vocabulario: Mediante una larga lista de palabras específicas.
- 2- Palabras similares: Seleccionadas por tópico relacionado.
- 3- Palabras con rimas: Estableciendo al menos una rima para cada palabra del vocabulario.
- 4- Autómata finito de aceptación: El cual revisará que dada una estrofa o poema, cumplan la métrica esperada.
- 5- Camino de extracción: Aplicando mediante RNN, una puntuación al poema generado.

La idea será la de aplicar a una red de generación de texto automático, una serie de modificadores, que permitan conservar la semántica y la métrica deseable del poema. Desarrollemos punto por punto la necesidad de realizar cada una de la tareas anteriores, expuestas en el trabajo.

1- El vocabulario compondrá la serie de datos fundamentales de entrada que aportar a nuestra red neuronal, al fin y al cabo serán el listado de palabras válidas con las que generar el poema.

2- Las palabras similares serán todas aquellas intercambiables por las palabras de los textos originales, capaces de mantener el contexto y cohesión semántica.

3- Lista de palabras con rimas: Para ser capaces de mantener las diversas rimas entre versos.

4- Autómata de aceptación: Se trata de un agente inteligente que determinará si el poema generado es bueno al cumplir las restricciones específicas que sean necesarias para ser considerado un poema.

5- Puntuación: Los poemas generados que mejor puntuación tengan son los que marcarán un precedente, a fin de que la red neuronal trate de adaptar los poemas generados a aquellos con mejor nota.

El enfoque de uso de este tipo de listas de palabras, será el de establecer un peso diferente según el peso de la sílaba tónica de cada palabra, de forma, que pudiéramos determinar en base a dicho peso, qué palabra o palabras son las más adecuadas para seguir el verso, así como poder llevar el conteo métrico correspondiente.



Para realizar dicha tarea, se nos propone el siguiente modelo de marcado:

010      1 0    10      101

Attending on his golden pilgrimage

Donde los ceros corresponden con las sílabas que no llevan el peso de la pronunciación y los unos, con las sílabas que sí llevan dicho peso. A partir de estos valores, se utiliza un sistema métrico denominado “slant rhyme”, el cual crea una clase par cada palabra no aguda, asignando un código a cada sonido vocálico, a partir de la última sílaba tónica, y añadiendo las consonantes que siguen al último sonido vocálico. Así pues, para la palabra (attending) su clase asociada sería:

- Attending (010) → “ending” (10)

- e → IY1

- i → IH0

- “i-ng” → NG

- Clase asociada: **IY1 \* IH0 NG**

Además, en cuanto a concordancia de la palabra, en este caso teniendo en cuenta la primera palabra, (*attending*), determinan que el ritmo de dicha palabra coincide con su clase asociada (IY1 \* IH0 NG).

El enfoque sería buscar palabras similares en dicha estructura de fuerza, y además que concordasen en cuanto a su ritmo, de entre toda la lista de palabras con relación semántica, es decir aquellas, que no alterasen de forma profunda el significado de la oración.

Por ejemplo: *Needing*, la cual tiene marcado 10, y ritmo (IY1 \* IH0 NG). Por lo que nuestro sistema de validación daría por buena la relación entre *attending* y *needing*, colocándolas como candidatas entre sí para nuestra lista de palabras con rima, y cohesión semántica.

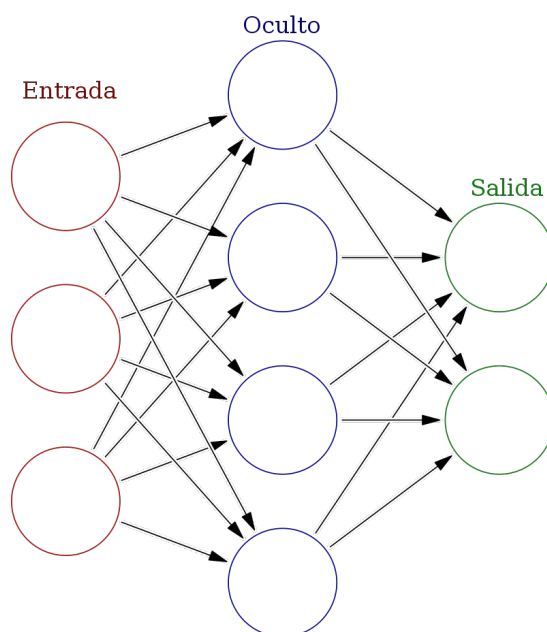
Finalmente, todas las listas de palabras semánticamente compatibles y que rimen con la palabra original, entrarían junto al modelo de la red neuronal concreta, para generar un nuevo poema. (Marjan Ghazvininejad et al, 2016)

### 2.3 Redes neuronales profundas

Las redes neuronales profundas son una particularidad de las redes neuronales tradicionales, que incorporan de forma eficiente nuevos métodos de aprendizaje profundo computacional, o en inglés, “deep learning”. Este tipo de programación, a diferencia de la programación basada en reglas, se basa en el reconocimiento de patrones.

El interés de usar redes neuronales profundas en este proyecto, es debido a la necesidad de modelar numerosos generadores de poemas (al menos, un generador por autor) sin contar con ningún tipo de reglas en especial a la hora de aplicar dicha generación. De esta manera seremos capaces de basándonos en un modelo entrenado a raíz de cada corpus específico, generar un poema lo más parecido posible a los poemas originales del autor.

El funcionamiento de las redes neuronales es simple, pues tendremos una serie de valores de entrada que, al introducirse en la estructura de la red neuronal, serán modificados por una serie de pesos establecidos en la red, para dar finalmente unos nuevos valores de salida.



*Imagen 2.2. Estructura general de una red neuronal. Fuente: [https://es.wikipedia.org/wiki/Red\\_neuronal\\_artificial](https://es.wikipedia.org/wiki/Red_neuronal_artificial).*

En particular, en este proyecto, trabajaremos con una función de propagación hacia atrás, de tal modo que se persiga minimizar la pérdida total del sistema al ir alterando los pesos de las

neuronas en cada iteración del sistema con respecto al peso deseado total del sistema.

No obstante, las redes neuronales no son una estructura y ente únicos, sino que existen en una amplia variedad y diversidad. En concreto nuestro proyecto cuenta con una serie diversa de características en función de las cuales nos decantaremos por la selección de un tipo de red neuronal u otro. Como ya hemos observado en el apartado de trabajos previos, las redes neuronales más habituales dentro de la generación de textos automáticos serían las contradictorias, las recurrentes y las convolucionales.

En este caso, se centrará el estudio en redes neuronales recurrentes y en su habilidad para generar texto de forma automática y con forma de poesía, a fin de comprobar su eficacia de forma más clara, analizando la propagación de secuencias textuales mediante algoritmos de retroalimentación o vuelta atrás programada que proporcionan capacidad a la red para “memorizar” los datos que proporcionan resultados óptimos.

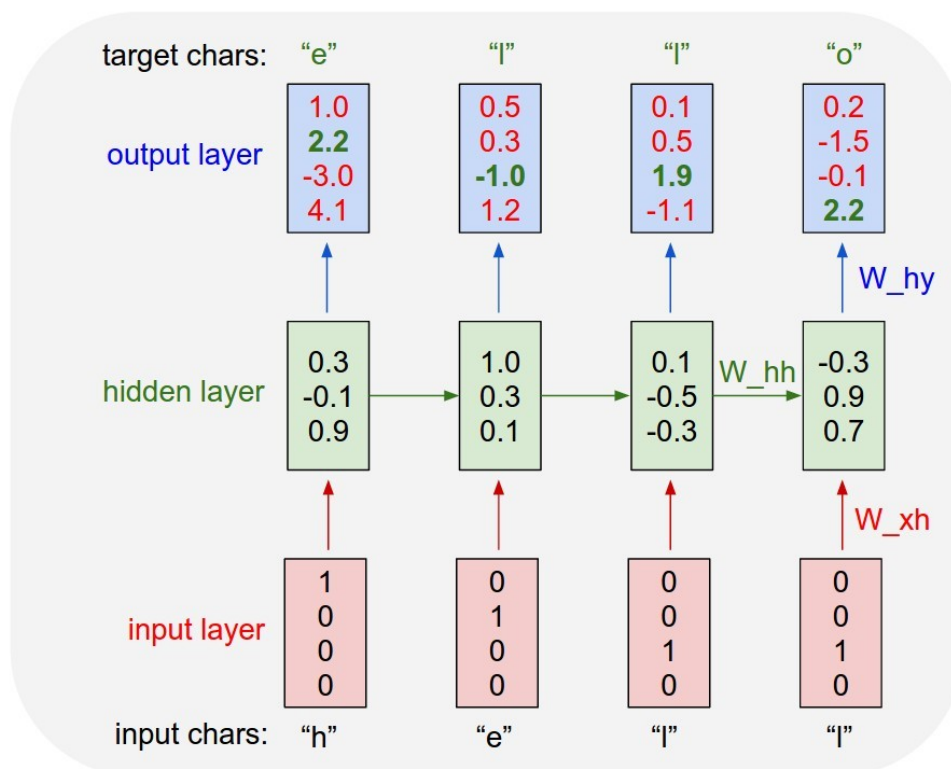


Imagen 2.3. Funcionamiento de una red neuronal recurrente.

Fuente: <https://www.i-ciencias.com/pregunta/33816/recurrentes-vs-redes-neuronales-recurrentes-que-se-aplica-mejor-para-la-pnl>

## 2.4 Técnicas de procesamiento del lenguaje

En el apartado de las técnicas del procesamiento del lenguaje, podemos encontrar tres características fundamentales a describir, y que aplicaremos en nuestro sistema durante el transcurso de la implementación de la red neuronal.

1- Word\_to\_vec

2- Entrenamiento basado en corpus

3- Tokenización

1.Word\_to\_vec

Word to vec es una técnica de generación de modelos con similitudes entre sí, usados para producir incrustación de palabras. Esta técnica, de uso exclusivo en nuestra red neuronal de palabras, toma como entrada el corpus de texto (en nuestro caso ya tokenizado) y produce un espacio vectorial de varios cientos de dimensiones, donde cada palabra se corresponde con un espacio único de dicho vector. Las palabras se ubican especialmente según su semántica, de forma que las palabras con contexto similar se encuentran más cerca entre sí que aquellas con un contexto más diferenciado.

2.Entrenamiento basado en corpus

Un corpus hace referencia a un conjunto cerrado de textos o datos representativos, destinados a un uso particular.

Un entrenamiento supervisado se basa en el uso de un gran volumen de datos, (en nuestro caso entre ocho y diez mil líneas por corpus) para tokenizar las palabras del corpus, y posteriormente asociarlas a los vectores de nuestro word\_to\_vec.

En este proyecto la orientación del uso de diferentes corpus (uno por autor) estará motivada por la necesidad de estudiar todos y cada uno de los autores, así como sus diversas facetas y particularidades al escribir poesía.

Los valores habituales para entrenar un corpus textual con el que posteriormente generar nuevo texto, son de al menos cien mil líneas de texto, lo cual supone un impedimento a la hora de generar poesía basada en autor, ya que es complicado encontrar autores con semejante volumen de versos disponibles.

### 3.Tokenización

Se entiende tokenización como el proceso de normalización de un texto, generalmente en base a las palabras del mismo, mediante segmentación textual o normalización del formato de las palabras que componen dicho texto.

De este modo, tomaremos cada unidad de texto tokenizada o token, como una instancia de un texto y un tipo dados.

De este modo, y al ser cada token, un tipo por sí mismo, podemos agruparlos de diversas formas como, por ejemplo, por listas ordenadas de token, o por cantidad de veces que estos han aparecido.

La forma de tokenizar un texto, dependerá fundamentalmente de la expresión regular que sea aplicada durante el proceso de tokenización; entendiéndose por expresión regular, todo aquel conjunto de caracteres que separarán entre sí cada uno de dichos tokens.

## 2.5 Análisis de objetivos

La aplicación web debe proporcionar una interacción entre la red neuronal generadora de poemas alojada en el back-end, la base de datos y la interfaz del usuario, de forma clara, concisa e intuitiva. En otras palabras, debe permitir tanto la navegación por las distintas vistas de la web como la creación de poemas, así como la lectura de los ya creados, de forma cómoda y sencilla para un usuario inexperto. Por lo tanto, las funcionalidades que han de ser incorporadas en dicha aplicación, serán las siguientes:

- 1- Navegación entre vistas a través de una interfaz web.
- 2- Creación de poemas de cada autor disponible en la web.

- 3- Consulta de los poemas ya creados y almacenados en la web.
- 4- Impresión de los poemas en formato de texto o PDF.
- 5- Realización de operaciones básicas de inserción, y selección sobre la base de datos, por medio de la interfaz web.
- 6- Consulta por parámetros sobre la base de datos de la web.
- 7- Comparación entre distintos tipos de redes neuronales y parámetros de entrenamiento.

## 2.6 Análisis de requisitos

Durante el desarrollo de este apartado, recogeremos y analizaremos los distintos requisitos funcionales y no funcionales que ha de tener nuestra aplicación web. Para ello tendremos en cuenta que por requisito funcional entendemos todo y cada uno de los servicios o funcionalidades (descritos en el apartado anterior) que debe garantizar nuestro proyecto. Y por requisitos no funcionales, entendemos todos aquellos que sin estar descritos de forma funcional, ayudan a cumplimentar todas las propiedades del sistema. Es importante recordar que el enfoque seguido, no solo durante la fase de requisitos sino durante todos los apartados que vendrán posteriormente en esta memoria, es un enfoque iterativo, esto, quiere decir, que en este caso, los requisitos aquí expuestos, son los requisitos demandados en el momento actual, pero al tratarse de un producto software, conforme avance el uso del mismo, estos requisitos podrían ir evolucionando y cambiando el producto a medida que transcurra su tiempo de vida.

En la siguiente tabla se describirán las funcionalidades expuestas anteriormente:

Dichas funcionalidades aparecen expuestas bajo la estructura código y descripción, donde el código tendrá un formato del tipo “FUNC-XX” y la descripción expone de que trata cada funcionalidad.

Código aplicado	Descripción de la funcionalidad
FUNC-01	Navegación entre vistas a través de una interfaz web
FUNC-02	Creación de poemas en todo momento de cada autor disponible en la web
FUNC-03	Consulta de los poemas ya creados y almacenados en la web
FUNC-04	Impresión de los poemas en formato de texto PDF

FUNC-05	Realización de operaciones básicas de inserción, y selección sobre la base de datos, por medio de la interfaz web
FUNC-06	Consulta parametrizada de poemas sobre la base de datos de la web
FUNC-07	Comparación entre distintos tipos de redes neuronales y parámetros de entrenamiento.

Tabla 2.1. Funcionalidades. Elaboración propia.

Una vez recogidas estas funcionalidades, pasaremos a establecer cuales serán los requisitos funcionales y no funcionales del sistema.

### 2.6.1 Requisitos funcionales

Como hemos indicado anteriormente, los requisitos funcionales son aquellos que se relacionan directamente con las funcionalidades u objetivos del sistema. Para clasificarlos y nombrarlos, usaremos la estructura: código, nombre, descripción y funcionalidad asociada a cada uno de estos requisitos.

1- Código: Código de identificación de cada requisito, bajo el formato “RF-XX”.

2- Nombre: Nombre de la tarea correspondiente a cada requisito.

3- Descripción: Descripción de la tarea del requisitos.

4- Funcionalidades asociadas: Funcionalidades que cumple el requisito.

Código aplicado	RF-01
<b>Nombre</b>	Navegación mediante interfaz web
<b>Descripción</b>	Este requisito, debe garantizar el cambio entre las distintas vistas de la interfaz, de forma intuitiva
<b>Funcionalidades asociadas</b>	FUNC-01 - Navegación entre vistas a través de una interfaz web FUNC-03 - Consulta de los poemas ya creados y almacenados en la web

<p>FUNC-05 - Realización de operaciones básicas de inserción, y selección sobre la base de datos, por medio de la interfaz web</p>
--

Tabla 2.2. Requisito funcional 1. Elaboración propia.

Código aplicado	RF-02
<b>Nombre</b>	Consultar todos los elementos disponibles en la base de datos
<b>Descripción</b>	Este requisito, debe garantizar la selección en todo momento, de aquellos poemas ya creados y guardados en la web
<b>Funcionalidades asociadas</b>	<p>FUNC-03 - Consulta de los poemas ya creados y almacenados en la web</p> <p>FUNC-05 - Realización de operaciones básicas de inserción, y selección sobre la base de datos, por medio de la interfaz web</p>

Tabla 2.3. Requisito funcional 2. Elaboración propia.

Código aplicado	RF-03
<b>Nombre</b>	Guardado del poema
<b>Descripción</b>	A la hora de crear un poema, se dará la opción al usuario de guardarlo en algún formato de texto estándar
<b>Funcionalidades asociadas</b>	FUNC-04 - Impresión de los poemas en formato de texto PDF

Tabla 2.4. Requisito funcional 3. Elaboración propia.

Código aplicado	RF-04
<b>Nombre</b>	Generación de poemas de forma automática
<b>Descripción</b>	La aplicación debe poseer una vista mediante la que el usuario debe ser capaz de generar un poema, en base al autor seleccionado, dentro de los disponibles en la web
<b>Funcionalidades asociadas</b>	<p>FUNC-01 - Navegación entre vistas a través de una interfaz web</p> <p>FUNC-02 - Creación de poemas en todo momento de cada autor disponible en la web</p>

Tabla 2.5. Requisito funcional 4. Elaboración propia.



Código aplicado RF-05	
<b>Nombre</b>	Consulta parametrizada
<b>Descripción</b>	La aplicación debe ser capaz de realizar consultas parametrizadas sobre la base de datos de la aplicación web, a través de uno de los campos del poema.
<b>Funcionalidades asociadas</b>	FUNC-03 - Consulta de los poemas ya creados y almacenados en la web FUNC-05 - Realización de operaciones básicas de inserción, y selección sobre la base de datos, por medio de la interfaz web FUNC-06 - Consulta parametrizada de poemas sobre la base de datos de la web

Tabla 2.6. Requisito funcional 5. Elaboración propia.

Código aplicado RF-06	
<b>Nombre</b>	Comparación entre distintos modelos y redes neuronales
<b>Descripción</b>	La aplicación debe ser capaz de mostrarnos distintos resultados provenientes de diferentes modelos. Estos modelos se deben diferenciar entre sí.
<b>Funcionalidades asociadas</b>	FUNC-07 - Comparación entre distintos tipos de redes neuronales y parámetros de entrenamiento.

Tabla 2.7. Requisito funcional 6. Elaboración propia.

### 2.6.2 Relación entre objetivos y requisitos funcionales

Una vez completado el análisis de los requisitos funcionales, veamos qué requisitos validan qué funcionalidad y si se validan de forma correcta todas las funcionalidades.

	RF-01	RF-02	RF-03	RF-04	RF-05	RF-06
FUNC-01	✓			✓		
FUNC-02				✓		
FUNC-03	✓	✓			✓	
FUNC-04			✓			✓
FUNC-05	✓	✓			✓	
FUNC-06					✓	
FUNC-07						✓

Tabla 2.8 Requisito funcionales vs objetivos del sistema. Elaboración propia.

Como se puede observar, nuestros requisitos funcionales satisfacen todas las funcionalidades de nuestra aplicación web.

### 2.6.3 Requisitos no funcionales

En cuanto a los requisitos no funcionales, tal y como se ha mencionado anteriormente, no están orientados a cumplir funcionalidades del sistema o de la aplicación, sino a proporcionar características o capacidades adicionales que mejoren el rendimiento del sistema y ayuden al cumplimiento de los requisitos funcionales. En nuestro caso, definiremos los siguientes:

1- Persistencia de datos. Los datos como los poemas generados, se guardan de forma automática en la base de datos, la cual debe ser capaz de conservarlos, aunque haya cualquier fallo en el sistema.

2- Las consultas a la base de datos han de ser rápidas, puesto que se han de indexar una gran cantidad de poemas en poco tiempo, a fin de aportar una experiencia cómoda al usuario.

3- La interfaz debe ser intuitiva y manejable, con un tiempo de aprendizaje mínimo por parte del usuario.

4- El tiempo de respuesta de la interfaz debe ser el más rápido posible.

5- La carga de la interfaz debe ser fluida, sin retardos parciales y sin bloqueos del sistema.

## 2.7 Análisis de casos de uso

En cuanto a los casos de uso tendremos en cuenta las diversas acciones que cada usuario puede realizar dentro de la aplicación, así como las respuestas que el sistema tiene a estas acciones.

El usuario será el encargado de comprobar tras una interacción mínima con el sistema, que todos estos casos de uso se garantizan durante el funcionamiento de la aplicación de forma correcta y sin errores.

### 2.7.1 Presentación de casos de uso

La estructura de definición de nuestra tabla de casos de uso será: código aplicado + nombre, donde el código tendrá el formato “CDU-XX” y el nombre será la tarea correspondiente a dicho código.

Código aplicado	Nombre
CDU-01	Navegar por las vistas de la interfaz
CDU-02	Seleccionar el país del poeta
CDU-03	Seleccionar poeta
CDU-04	Escribir título, y semilla
CDU-05	Generar poema
CDU-06	Imprimir poema
CDU-07	Buscar por campos clave
CDU-08	Consultar listado de poemas
CDU-09	Comparar redes neuronales

Tabla 2.9. Casos de uso. Elaboración propia.

### 2.7.2 Diagrama de casos de uso

A continuación se muestran en forma de diagrama, los diferentes casos de uso que hemos identificado durante la planificación de la aplicación.

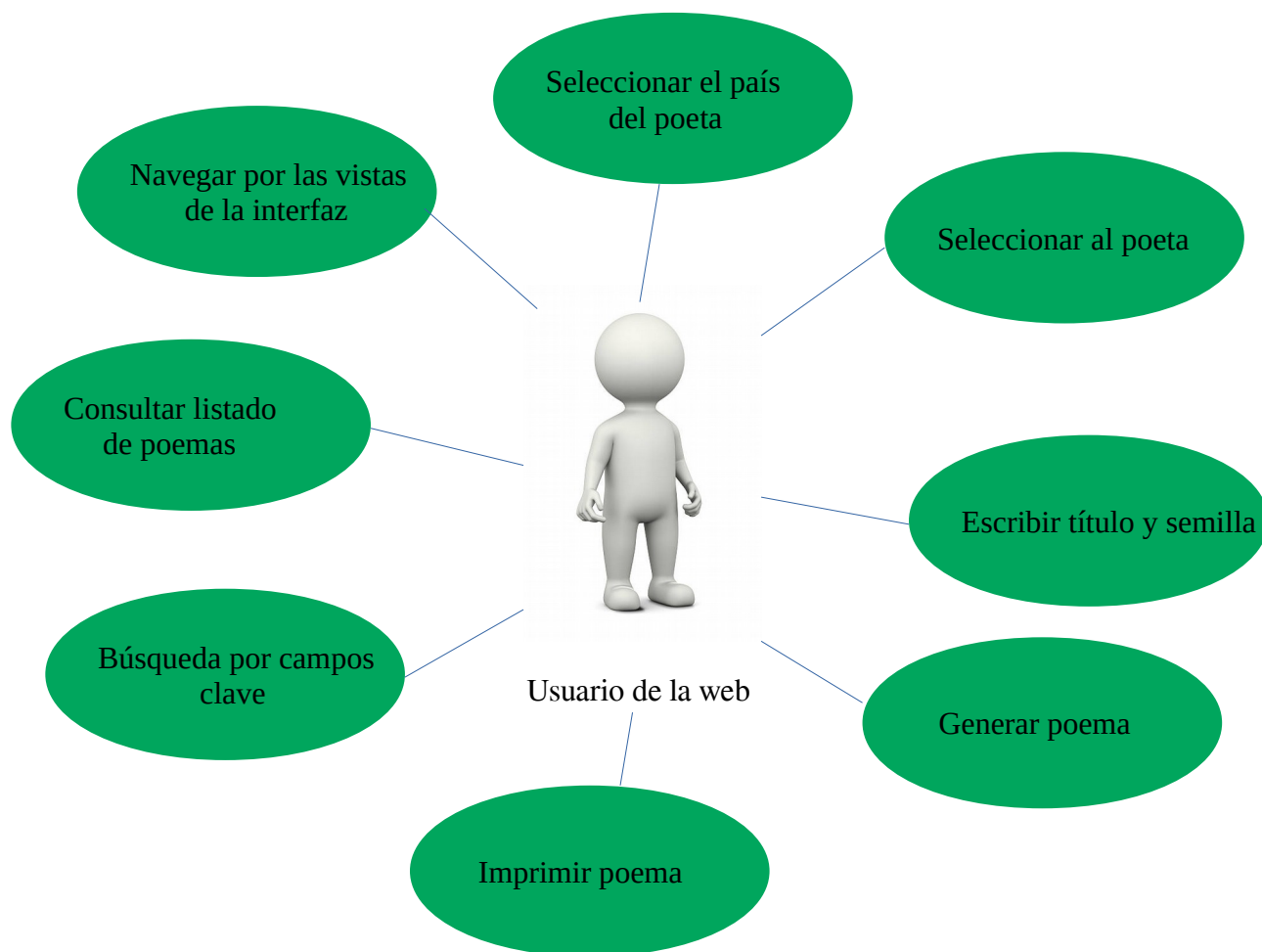


Imagen 2.4. Diagrama de casos de uso. Elaboración propia.

### 2.7.3 Definición de casos de uso

En este apartado vamos a proceder a detallar de manera más profunda los casos de uso de los que disponemos y vamos a relacionarlos con los requisitos y objetivos iniciales del proyecto.

<b>CDU-01</b>	<b>Navegar por las vistas de la interfaz</b>
<b>Requisitos asociados</b>	RF-01 Navegación mediante interfaz web
<b>Funcionalidades asociadas</b>	FUNC-01 Navegación entre vistas a través de una interfaz web
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario, partiendo de la vista principal o inicial de la aplicación y a través de las distintas extensiones de la barra de navegación, es capaz de moverse por la web.
<b>Condición anterior</b>	Acceder a la aplicación y a alguna de sus vistas.
<b>Flujo natural</b>	1- Inicio de la aplicación.  2- Selección de la nueva vista deseada a través de un click en las secciones disponibles en la barra de navegación.  3- Una vez finalizada la actividad en la página web, se puede cerrar la página bajo el procedimiento habitual.
<b>Flujo alternativo</b>	1- Acceso directo a la página deseada, escribiendo la url junto al controlador de la vista deseada.  2- Cierre y navegación bajo el procedimiento habitual.
<b>Condición posterior</b>	(Sin condición posterior)
<b>Observaciones</b>	(Sin observaciones)

Tabla 2.10. CDU-01 Navegar por las vistas de la interfaz. Elaboración propia.

CDU-02	Seleccionar país del poeta
<b>Requisitos asociados</b>	RF-01 Navegación mediante interfaz web  RF-04 Generación de poemas de forma automática
<b>Funcionalidades asociadas</b>	FUNC-01 Navegación entre vistas a través de una interfaz web  FUNC-02 - Creación de poemas en todo momento de cada autor disponible en la web
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario debe moverse desde la vista que esté a la vista de la aplicación principal. Una vez allí, en el desplegable, seleccionará el país del autor que desee.
<b>Condición anterior</b>	Acceder a la vista de generación de poemas.
<b>Flujo natural</b>	1- Acceso a la vista de generación de poemas automáticos.  2- Apertura del desplegable donde se muestra el listado de países con algún poeta disponible en la web.  3- Selección del país deseado.
<b>Flujo alternativo</b>	(No existe un flujo alternativo)
<b>Condición posterior</b>	Para la generación del poema, el usuario deberá proseguir con la selección y rellenado de los campos del formulario de forma correcta.
<b>Observaciones</b>	Si en todo momento, durante el proceso de selección de los campos a rellenar por el usuario, este decide cambiar de vista, la información rellenada, no se mantendrá al volver a la vista.

Tabla 2.11. CDU-02. Seleccionar el país del poeta. Elaboración propia.

CDU-03	Seleccionar al poeta
<b>Requisitos asociados</b>	RF-01 Navegación mediante interfaz web  RF-04 Generación de poemas de forma automática
<b>Funcionalidades asociadas</b>	FUNC-01 Navegación entre vistas a través de una interfaz web  FUNC-02 - Creación de poemas en todo momento de cada autor disponible en la web
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario debe moverse desde la vista que esté a la vista de la aplicación principal. Una vez allí, en el desplegable, una vez seleccionado el país, el formulario se actualizará con los autores de dicho país disponibles.
<b>Condición anterior</b>	Acceder a la vista de generación de poemas y selección de un país.
<b>Flujo natural</b>	1- Apertura del desplegable con el listado de poetas pertenecientes al país previamente seleccionado.  2- Selección del poeta deseado.
<b>Flujo alternativo</b>	(No existe un flujo alternativo)
<b>Condición posterior</b>	Para la generación del poema, el usuario deberá proseguir con el rellenado de los campos del formulario de forma correcta.
<b>Observaciones</b>	Si en todo momento, durante el proceso de selección de los campos a rellenar por el usuario, este decide cambiar de vista, la información rellenada, no se mantendrá al volver a la vista.

Tabla 2.12. CDU-03. Seleccionar al poeta. Elaboración propia.

CDU-04	Escribir título y semilla
<b>Requisitos asociados</b>	RF-01 Navegación mediante interfaz web  RF-04 Generación de poemas de forma automática
<b>Funcionalidades asociadas</b>	FUNC-01 Navegación entre vistas a través de una interfaz web  FUNC-02 - Creación de poemas en todo momento de cada autor disponible en la web
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario debe moverse desde la vista que esté a la vista de la aplicación principal. Una vez allí, seleccionados país y autor poético, el usuario deberá escribir el título correspondiente con el poema que se va a generar, así como dar una semilla de inicio del poema.
<b>Condición anterior</b>	Acceder a la vista de generación de poemas y selección de un país y un autor de dicho país.
<b>Flujo natural</b>	1- Rellenado del campo título  2- Rellenado del campo semilla.
<b>Flujo alternativo</b>	Los campos título y semilla, son totalmente opcionales, por lo tanto, no rellenar alguno de ellos, o ambos, no interrumpirá el flujo de generación del poema.
<b>Condición posterior</b>	(No hay condición posterior)
<b>Observaciones</b>	Si en todo momento, durante el proceso de selección de los campos a rellenar por el usuario, este decide cambiar de vista, la información rellenada, no se mantendrá al volver a la vista.

Tabla 2.13. CDU-04. Escribir título y semilla. Elaboración propia.



CDU-05	Generar poema
<b>Requisitos asociados</b>	RF-01 Navegación mediante interfaz web  RF-04 Generación de poemas de forma automática
<b>Funcionalidades asociadas</b>	FUNC-01 Navegación entre vistas a través de una interfaz web  FUNC-02 - Creación de poemas en todo momento de cada autor disponible en la web
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario debe moverse desde la vista que esté a la vista de la aplicación principal. Una vez allí, cumplimentados todos los datos obligatorios, y los datos opcionales del formulario, el usuario debe clicar en generar poema.
<b>Condición anterior</b>	Acceder a la vista de generación de poemas y selección de un país y un autor de dicho país.
<b>Flujo natural</b>	1- Hacer clic en generar poema.
<b>Flujo alternativo</b>	(No existe un flujo alternativo)
<b>Condición posterior</b>	La aplicación se congelará esperando a recibir la respuesta del servidor, el cual una vez recibidos los datos, tarda varios segundos en devolver el poema generado final.
<b>Observaciones</b>	Si en todo momento, durante el proceso de generación del poema, se decide dar marcha atrás, los datos se pierden, y la generación del poema se detiene.

Tabla 2.14. CDU-05. Generar poema. Elaboración propia.

CDU-06	Imprimir poema
<b>Requisitos asociados</b>	RF-03 Guardado del poema RF-04 Generación de poemas de forma automática
<b>Funcionalidades asociadas</b>	FUNC-01 Navegación entre vistas a través de una interfaz web FUNC-04 Impresión de los poemas en formato de texto pdf
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario puede elegir una vez generado el poema, si desea imprimirlo en formato pdf.
<b>Condición anterior</b>	Haber hecho clic en generar poema.
<b>Flujo natural</b>	1- Hacer clic en imprimir poema. 2- Guardar el poema en el lugar deseado de su dispositivo.
<b>Flujo alternativo</b>	(No existe un flujo alternativo)
<b>Condición posterior</b>	(No existen condiciones posteriores)
<b>Observaciones</b>	Un poema, solo podrá ser impreso en el momento en que se genera, pues al suponerse anónimo, el poema solo corresponderá de forma virtual al autor en el momento de su creación. Posteriormente a su creación el poema se guarda de manera automática en la base de datos.

Tabla 2.15. CDU-06. Imprimir poema. Elaboración propia.

CDU-07	Búsqueda por campos clave
<b>Requisitos asociados</b>	RF-05 Consultar todos los elementos disponibles en la base de datos  RF-03 Guardado del poema
<b>Funcionalidades asociadas</b>	FUNC-03 Consulta de los poemas ya creados y almacenados en la web  FUNC-05 Realización de operaciones básicas de inserción, y selección sobre la base de datos, por medio de la interfaz web  FUNC-06 - Consulta parametrizada de poemas sobre la base de datos de la web
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario podrá a través de la barra de búsqueda realizar búsqueda por campos clave como el título o el autor del que se ha generado el poema.
<b>Condición anterior</b>	Acceder a la página web.
<b>Flujo natural</b>	1- Abrir la barra de búsqueda alojada en la barra de navegación.  2- Escribir las palabras clave a buscar.  3- Activar la búsqueda.
<b>Flujo alternativo</b>	(No existe un flujo alternativo)
<b>Condición posterior</b>	(No existen condiciones posteriores)
<b>Observaciones</b>	La búsqueda no ofrecerá resultados si la web no tiene almacenadas las palabras clave exactas que el usuario ha introducido para su búsqueda.

Tabla 2.16. CDU-07. Búsqueda por campos clave. Elaboración propia.

CDU-08	Consultar listado de poemas
<b>Requisitos asociados</b>	RF-02 Consultar todos los elementos disponibles en la base de datos  RF-03 Guardado del poema
<b>Funcionalidades asociadas</b>	FUNC-03 Consulta de los poemas ya creados y almacenados en la web  FUNC-05 Realización de operaciones básicas de inserción, y selección sobre la base de datos, por medio de la interfaz web
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario podrá visualizar de forma directa todos los poemas registrados en la base de datos de la página web hasta la fecha. Estos, aparecerán ordenados desde el más reciente hasta el más antiguo.
<b>Condición anterior</b>	Acceder a la página web.
<b>Flujo natural</b>	1- Hacer clic en la sección poemas generados dentro de la barra de navegación.
<b>Flujo alternativo</b>	1- Acceder a través de la url, invocando el controlador correspondiente.
<b>Condición posterior</b>	(No existen condiciones posteriores)
<b>Observaciones</b>	La búsqueda no ofrecerá resultados si la web no tiene almacenado ningún poema.

Tabla 2.17. CDU-08. Consultar listado de poemas. Elaboración propia.

CDU-09	Comparar redes neuronales
<b>Requisitos asociados</b>	RF-06 Comparación entre distintos modelos y redes neuronales
<b>Funcionalidades asociadas</b>	FUNC-07 Comparación entre distintos tipos de redes neuronales y parámetros de entrenamiento.
<b>Participantes</b>	Usuario de la web
<b>Descripción</b>	El usuario podrá visualizar de forma directa los resultados obtenidos por diferentes redes neuronales entrenadas con diferentes parámetros y formato de predicción del siguiente elemento, es decir, caracteres o palabras.
<b>Condición anterior</b>	Acceder a la página web.
<b>Flujo natural</b>	1- Hacer clic en la sección testea la red dentro de la barra de navegación.
<b>Flujo alternativo</b>	1- Acceder a través de la url, invocando el controlador correspondiente.
<b>Condición posterior</b>	(No existen condiciones posteriores)
<b>Observaciones</b>	Los poemas mostrados son de ejemplo, al probar cada red neuronal, se generarán poemas diferentes de forma constante.

Tabla 2.18. CDU-09. Comparar redes neuronales. Elaboración propia.

**2.7.4 Matriz de trazabilidad**

Habiendo completado ya el análisis de los diferentes casos de uso que hemos identificado en nuestra aplicación web, podemos observar ahora de forma concisa, cual es la relación entre los casos de uso y los requisitos funcionales identificados previamente, para identificar si se cumplimentan todos estos requisitos y por tanto, todas las funcionalidades originales de la aplicación.

	CDU-01	CDU-02	CDU-03	CDU-04	CDU-05	CDU-06	CDU-07	CDU-08
RF-01	✓	✓	✓	✓	✓			
RF-02							✓	✓
RF-03						✓	✓	✓
RF-04		✓	✓	✓	✓	✓		
RF-05							✓	

Tabla 2.19. Matriz de trazabilidad. Elaboración propia.

Como se puede observar, la matriz de trazabilidad muestra que todos los requisitos se cumplen en al menos un caso de uso, garantizando el cumplimiento de las funcionalidades básicas del sistema, que a su vez ya estaban satisfechas por nuestros requisitos funcionales.

### **3. Planificación**

En esta sección de la memoria del proyecto se abordará la planificación previa a su ejecución. En esta planificación se abordarán temas tan importantes como las estimaciones de costes, recursos humanos, recursos materiales y estimaciones de tiempo necesarias para el desarrollo de la aplicación.

Dichas estimaciones serán realizadas en base a un intervalo de tiempo establecido desde la fecha inicial del proyecto.

El proyecto será dividido en etapas o fases, las cuales podrán englobar a su vez diferentes subtareas o no. Cada fase y su ejecución determinarán la evolución del proyecto. Del mismo modo, las diferentes fases con sus correspondientes tareas, se encontrarán ordenadas en base a una línea temporal, donde cada una de ellas tendrá una duración estimada que servirá de apoyo para medir la duración total del proyecto.

Es importante recordar que los costes totales que se expondrán a continuación, pertenecen a un desarrollo del proyecto de forma comercial y con perspectiva de ser vendido como producto.

Finalmente, se harán las estimaciones anteriormente nombradas sobre costes, recursos y tiempo.

#### **3.1 Tareas a realizar**

Las tareas consideradas como fases críticas del proyecto que se encontrarán a continuación, tendrán asociadas una duración temporal, una relación de precedencia y una relación de orden entre sí, establecidos a razón del criterio de desarrollo del proyecto.

Dichas tareas serán expuestas de forma visual en la tabla que encontraremos a continuación, donde además se las dotará de un identificador a modo de guía.

Identificador	Tarea	Precedentes	Duración	Fecha inicio	Fecha fin
A	Lectura de artículos sobre generación de lenguaje y poesía	-	7 días	16 septiembre 2018	24 septiembre 2018
B	Revisión de trabajos existentes	A	4 días	25 septiembre 2018	28 septiembre 2018
C	Análisis de requisitos	B	6 días	1 Octubre 2018	8 Octubre 2018
D	Estudio de la arquitectura	C	4 días	9 Octubre 2018	12 Octubre 2018
E	Casos de uso	D	6 días	15 Octubre 2018	22 Octubre 2018
F	Tecnologías y lenguaje	D	3 días	15 Octubre 2018	17 Octubre 2018
G	Storyboard	E,F	14 días	23 Octubre 2018	9 Noviembre 2018
H	Diseño de la base de datos	E,F	7 días	23 Octubre 2018	31 Octubre 2018
I	Estudio entre redes neuronales de caracteres y palabras	C	28 días	9 Octubre 2018	15 Noviembre 2018
J	Implementación RNN	I	28 días	16 Noviembre 2018	25 Diciembre 2018
K	Implementación Web	G	18 días	12 Noviembre 2018	5 Diciembre 2018



L	Implementación Base de datos	H	3 días	1 Noviembre 2018	5 Noviembre 2018
M	Pruebas	I,G,H	14 días	26 Diciembre 2018	14 Enero 2019
N	Corrección de errores	M	7 días	15 Enero 2019	23 Enero 2019
O	Manual de usuario	N	5 días	24 Enero 2019	30 Enero 2019
P	Redacción de la memoria	-	154 días	9 Septiembre 2018	30 Enero 2019

Tabla 3.1 Planificación temporal del proyecto. Elaboración propia.

A continuación se va a utilizar un diagrama de Gantt, el cual mediante una representación gráfica permitirá visualizar de forma más cómoda cada tarea o actividad en función de la duración prevista para cada una de ellas.

En caso de que el mismo proyecto fuese realizado por un equipo de trabajadores conjuntos, esta planificación podría servir para acelerar la ejecución del proyecto, permitiendo que varios integrantes del grupo trabajen de forma conjunta entre ellos.

El diagrama será de formato sencillo, donde las filas indicarán las diferentes tareas a realizar y las columnas representarán una línea temporal acorde a la duración estimada del proyecto completo.

Dicho diagrama de Gantt ha sido realizado en base a las estimaciones de duración y relaciones de precedencia entre las tareas de la tabla de planificación temporal del proyecto. Cabe destacar que la memoria es una tarea especial del proyecto, ya que se trata de algo que se debe ir realizando a lo largo de toda la ejecución del mismo, y por lo tanto supondrá el inicio y fin del proyecto.

## 3.2 Cronograma - Diagrama de Gantt

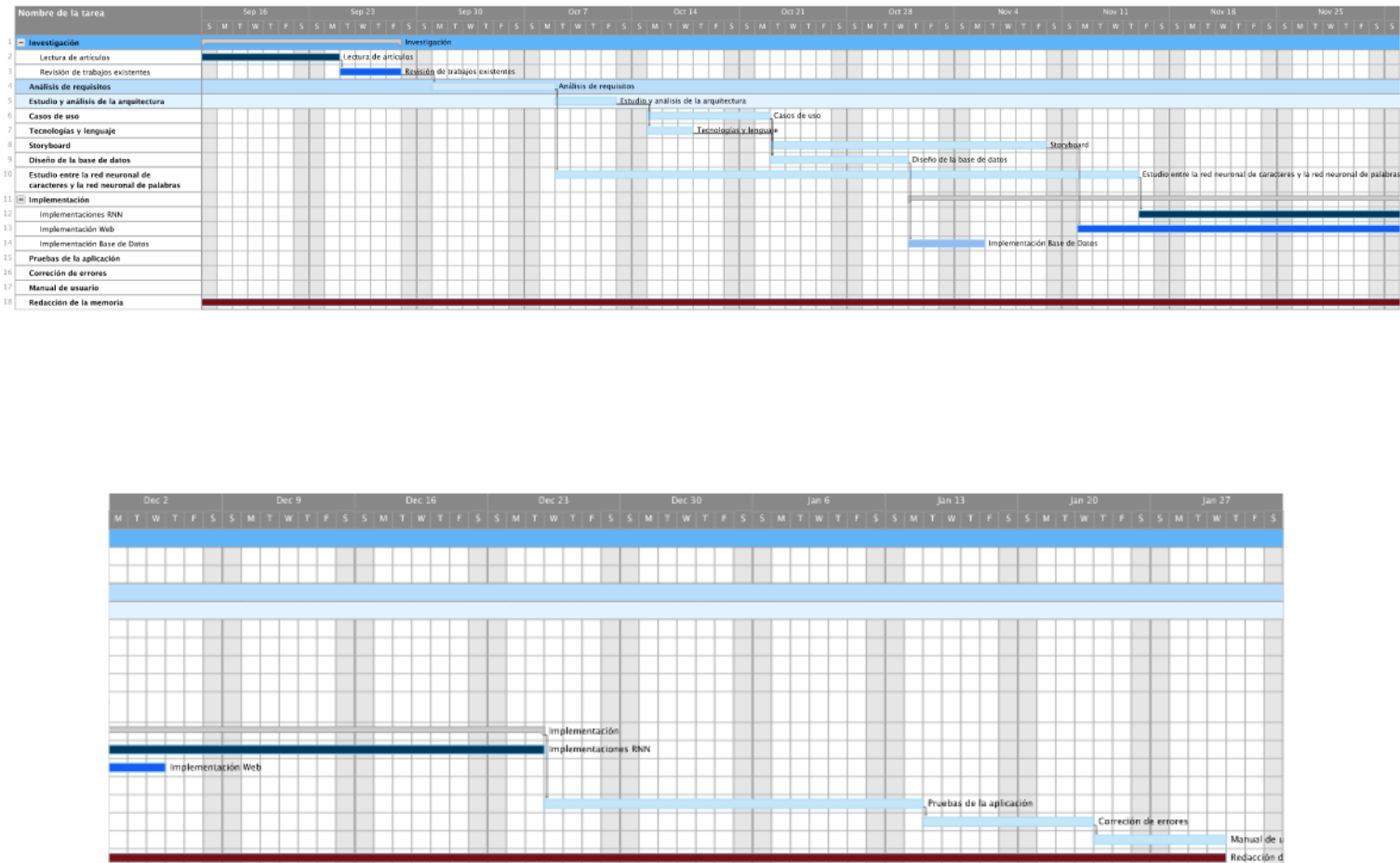


Imagen 3.1 Diagrama de Gantt. Elaboración propia.

### **3.3 Estimación de costes**

En este apartado se realizará un análisis de los costes de desarrollo y ejecución del proyecto. Es decir, se tendrán en cuenta los costes de recursos directos asociados al proyecto, e indirectos, provenientes por la propia realización de cualquier actividad en nuestro sector.

Las estimaciones de costes están realizadas a partir del tiempo total de duración estimado del proyecto, de forma que a más tiempo se requiera tener contratado a un empleado, más coste requiere la realización del proyecto.

Los diferentes costes a calcular se agruparán en este caso dependiendo del tipo de objeto que representen:

- 1- Costes humanos.
- 2- Costes hardware.
- 3- Costes software.
- 4- Servicios externos.

#### **3.3.1 Costes humanos**

Los costes o recursos humanos hacen referencia al número y tipo de empleados que necesitaremos para la realización de un proyecto o una aplicación concreta. Como ya sabemos nuestro proyecto se divide en diferentes tareas, las cuales serán abordadas por distintos tipos de perfiles de trabajadores.

Los costes asociados a cada uno de estos trabajadores dependerá del coste base de emplear a dicho trabajador, así como del tiempo que cada trabajador tenga que dedicar. Este tiempo es directamente proporcional a la duración de las tareas que se le asignen.

Por lo tanto, en base a las diferentes tareas incluidas en nuestro proyecto, se calculará el número y tipo de empleados necesarios y a continuación, el sueldo a pagar a cada uno.

Es importante tener en cuenta que no todo el año es laborable, por lo que se tendrán en cuenta

los siguiente días en los que no se podrá trabajar:

- 1- 30 días de vacaciones por año
- 2- 48 sábados al año (excluyendo los incluidos en las vacaciones)
- 3- 48 domingos al año (excluyendo los incluidos en las vacaciones)
- 4- 14 días festivos por año

Sumando todos estos días se obtiene un total de 140 días no laborables, los cuales restados al total de días anuales dentro de un año natural (365), nos dará 225 días hábiles al año por trabajador.

Además, se supondrá que los trabajadores trabajan a jornada completa de 8 horas al día. Al multiplicar esto por el número días hábiles de un año, se obtiene un total de 1800 horas hábiles.

Se ha estimado que los perfiles necesarios serán los siguientes:

- 1- Jefe de proyectos → 2548,20 €
- 2- Analista programador → 2009,36 €
- 3- Programador web senior → 2678,57 €
- 4- Programador web junior → 1892,86 €

Veamos cual será el coste de cada uno de ellos en relación a las tareas asociadas y la duración de sus contratos. El sueldo por hora se calculará en relación al sueldo anual, el cual se ha calculado a su vez en base al sueldo mensual encontrado en los portales web como indeed y OHR, multiplicado por unas estimadas 14 pagas mensuales.

A estos sueldos se les ha añadido la parte correspondiente que la empresa deberá abonar a la seguridad social y que se detallará a continuación.

**Jefe de proyectos (2548,50 mensual)**

Gastos ssgg

- 1- Contingencias comunes -> 23,6% (de la nómina mensual) -> 601.45
  - 2- Prestaciones por desempleo -> 5,5% (de la nómina mensual) -> 140.17
  - 3- Accidentes laborales o enfermedad profesional 3,5% (de la nómina mensual) -> 89.20
  - 4- Formación profesional 0,6% (de la nómina mensual) -> 15.29
  - 5- FOGASA (Quiebra de la empresa) 0,9% -> 22.94
- Total ssgg -> 869.05
- Total por trabajador mensual -> 3417.55

**Analista Programador (2009.36 mensual)**

Gastos ssgg

- 1- Contingencias comunes -> 23,6% (de la nómina mensual) -> 474.21
  - 2- Prestaciones por desempleo -> 5,5% (de la nómina mensual) -> 110.51
  - 3- Accidentes laborales o enfermedad profesional 3,5% (de la nómina mensual) -> 70.33
  - 4- Formación profesional 0,6% (de la nómina mensual) -> 12.06
  - 5- FOGASA (Quiebra de la empresa) 0,9% -> 18.08
- Total ssgg -> 685.19
- Total por trabajador mensual -> 2694.55

**Programador Web Senior (2678.57 mensual)**

Gastos ssgg

- 1- Contingencias comunes -> 23,6% (de la nómina mensual) -> 632.14
  - 2- Prestaciones por desempleo -> 5,5% (de la nómina mensual) -> 147.32
  - 3- Accidentes laborales o enfermedad profesional 3,5% (de la nómina mensual) -> 93.75
  - 4- Formación profesional 0,6% (de la nómina mensual) -> 16.07
  - 5- FOGASA (Quiebra de la empresa) 0,9% -> 24.11
- Total ssgg -> 913.39
- Total por trabajador mensual -> 3591.96

**Programador Web Junior** (1892.86 mensual)

## Gastos ssgg

1- Contingencias comunes -&gt; 23,6% (de la nómina mensual) -&gt; 446.71

2- Prestaciones por desempleo -&gt; 5,5% (de la nómina mensual) -&gt; 104.10

3- Accidentes laborales o enfermedad profesional 3,5% (de la nómina mensual) -&gt; 66.25

4- Formación profesional 0,6% (de la nómina mensual) -&gt; 11.36

5- FOGASA (Quiebra de la empresa) 0,9% -&gt; 17.04

Total ssgg -&gt; 645.46

Total por trabajador mensual -&gt; 2538.32 (esto por número de trabajadores y por número de meses)

Puesto profesional	Sueldo	Tareas asignadas	Tiempo de trabajo	Total
Jefe de proyectos	47845,70€ anuales 26,58€ por hora	Redacción de la memoria	99 días x 2 horas al día = 198 horas	5262,84€
Analista programador	37723,70€ anuales 20,96€ por hora	Análisis de requisitos Análisis de la arquitectura Análisis de los casos de uso Análisis de las tecnologías y el lenguaje Storyboard Diseño de la Base de Datos	40 días x 8 horas al día = 320	6707,20€
Programador web senior	50287,44€ anuales 27,94€ por hora	Lectura de artículos Generación del lenguaje y poesía	67 días x 8 horas al día = 536	14975,84€

		Estudio entre redes neuronales de caracteres y de palabras		
		Implementación de la aplicación (RNN, BD, web)		
		Corrección de errores		
Programador web junior	35536,48€ anuales 19,74€ por hora	Plan de pruebas  Manual de usuario	17 días x 8 horas al día = 136 horas	2684,64€
<b>Total</b>				<b>29630,52€</b>

Tabla 3.2 Costes humanos. Elaboración propia.

### 3.3.2 Costes de hardware y software

Una vez vistos los costes humanos que tendría el proyecto, veamos los costes hardware (materiales, máquinas, instalación, dispositivos externos, etc) que requerirá el desarrollo del proyecto. Así como los costes software (Sistema operativo o programas a instalar).

El proyecto se ha realizado sobre el siguiente equipo con las siguientes especificaciones técnicas.

Componentes	Especificaciones
<b>Procesador</b>	Intel core i7-4510U 2.00GHZ
<b>Memoria RAM</b>	16 GB
<b>Gráfica</b>	Nvidia GEFORCE 860 M
<b>Disco Duro</b>	1TB HDD

Tabla 3.3 Especificaciones ordenador de desarrollo. Elaboración propia.

Hardware	Unidades	Precio unitario	Tiempo de uso	Coste por mes
<b>Ordenador personal</b>	1	800€	6 meses	133,33€

Tabla 3.4 Costes Hardware. Elaboración propia.

Software	Unidades	Precio Unitario	Tiempo de uso	Coste total
<b>Sistema Operativo Ubuntu 18.04</b>	1	0€	99 días	0€
<b>Visual Studio Code</b>	1	0€	99 días	0€
<b>Python 3.7 y librerías de Python</b>	1	0€	99 días	0€
<b>Libre Office 6.0</b>	1	0€	99 días	0€
<b>Navegador Web Mozilla Firefox versión 58</b>	1	0€	99 días	0€
<b>GitLab Ujaen</b>	1	0€	99 días	0€
<b>Total</b>				0€

Tabla 3.5 Costes Software. Elaboración propia.

### 3. 3. 3 Costes adicionales

En el apartado de costes adicionales se colocarán los costes de los servicios derivados de trabajar cada día. Estos servicios contemplados serán los siguientes:

Servicio	Coste al mes	Tiempo de uso	Coste total de uso
<b>Internet fibra óptica</b>	80€	3,3 meses	264€
<b>Servicio de luz</b>	20€	3,3 meses	66€
<b>Material fungible</b>	0€	3,3 meses	0€
<b>Total</b>			330€

Tabla 3.6 Costes adicionales. Elaboración propia.



Así pues una vez calculados los costes parciales del proyecto, veamos el coste total añadiendo el IVA y un beneficio estimado del 6%.

Categoría	Coste
<b>Costes humanos</b>	29630,52€
<b>Costes hardware</b>	800€
<b>Costes software</b>	0€
<b>Costes adicionales</b>	330€
<b>Total costes</b>	30760,52€
<b>Beneficio (6%)</b>	1845,63€
<b>Total sin IVA</b>	32606,15€
<b>IVA (21%)</b>	6847,29€
<b>Total</b>	<b>39453,44€</b>

Tabla 3.7. Total de costes. Elaboración propia.

Una vez valorado el coste total de realizar dicho proyecto, se podrían valorar diversas opciones para abaratarlo como la reducción del beneficio o la contratación de un mayor número de personal relativo a los roles más económicos. Por ejemplo en este caso podría prescindirse del programador web senior y contratar otro programador web junior. Este tipo de decisiones ya serían valoración de la empresa.

## **4.Diseño**

La fase de diseño tratará de explicar cómo la aplicación se ha ido desarrollando desde el punto de vista visual, desde el punto de vista de la estructuración de la base de datos y por último desde el punto de vista de la arquitectura de la red neuronal o en nuestro caso de las redes neuronales.

### **4.1 Aplicación web**

La aplicación web se divide en dos partes desde el punto de vista del diseño: El diseño de la estructura de clases y el diseño de la interfaz web. Veamos las especificaciones de ambas partes por separado, así como la evolución de los mismos a lo largo de las iteraciones de nuestro producto software en base a las diferentes necesidades puntuales.

#### **4.1.1 Diseño de la estructura de clases**

La estructura de clases tiene dos partes claramente diferenciadas, la primera es la parte front-end en la que se da la visualización de los datos disponibles en la web, así como la interacción con el usuario. Y la segunda es el back-end que será donde se lleven a cabo las tareas de computación compleja. Sin embargo debido a que nuestra aplicación está enfocada a la visualización de los poemas generados de forma automática por el back-end, su diseño no estará compuesto de una multitud de clases significativas. Además, dichas clases no presentarán la estructura habitual de un paradigma de programación embebida, sino que presentarán una estructura bajo un paradigma de programación web. En este caso el paradigma utilizado serán el patrón Model-Vista-Controlador o MVC.

Se debe tener en cuenta que la aplicación maneja, aunque de varias formas diferentes, un único objeto, que será el objeto Poema.

### **- Front-end**

En el front-end se encontrarán las visualizaciones de las únicas dos interacciones complejas que tendrá esta aplicación, la generación de poemas y la consulta de dichos poemas. Estas interacciones se denominarán tareas.

1- Visualización de la generación de poemas: La generación de poemas tendrá consigo la tarea de trasladar los campos rellenos del formulario requerido para generar el poema, devolver un poema, con dichos datos colocados de forma coherente y presentarlos al usuario de forma ordenada.

2- Visualización de la consulta de poemas: La consulta será la tarea de petición en indexación de los datos almacenados en la base de datos por medio de un parámetro o por carácter general. La consulta deberá devolver un listado de poemas con aquellos elementos que satisfagan las restricciones.

### **- Back-end**

En el back-end se hallarán las mismas tareas principales junto a una serie de subtareas adicionales. El back end se divide debido al patrón MVC en modelo y controlador, por lo que esta misma división implica un aumento en el número de subtareas. Veámoslo en detalle:

1- Generación de poemas: Para la generación de poemas, el controlador, deberá ser capaz de gestionar la información recibida por la vista (formulario del usuario) y de generar un poema en base al modelo correspondiente del autor seleccionado. Así mismo deberá encargarse de llamar a la función encargada de insertar dicho poema en la base de datos y devolver otra vista con dicho poema.

2- Consulta de poemas: Para dicha tarea se debe realizar una consulta a la base de datos (por parte del controlador y a través del modelo), la cual según los parámetros de los que se disponga, devuelva unos poemas u otros.

Todo esto debemos enmarcarlo como hemos indicado anteriormente dentro del patrón MVC, pero ¿en qué consiste dicho patrón y por qué se ha seleccionado ?

El patrón MVC hace referencia a la estructura de trabajo de la aplicación web, en la cual cada tarea completa (esto es cada interacción del usuario que precise del uso de la base de datos) estará dividida en Modelo (parte de código donde se procesa la información relativa a la base de datos, se sitúa en el Back-end), Vista (parte visual del código consistente en una página web que se muestra al usuario, se sitúa en el front-end) y Controlador (parte del código cuya función es principalmente manejar la aplicación y satisfacer los objetivos).

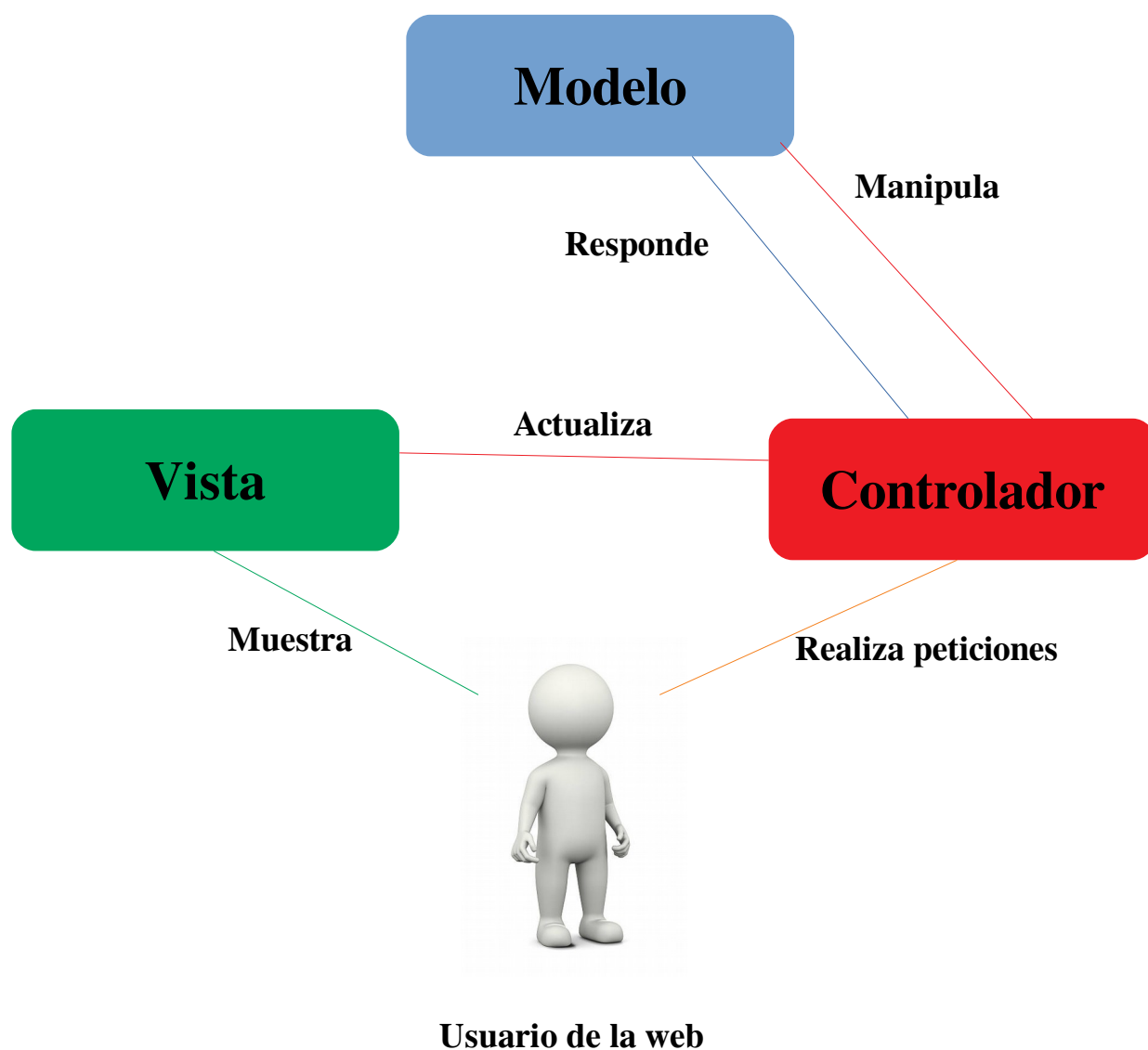


Imagen 4.1. Estructura patrón MVC. Elaboración propia.

#### **4.1.2 Diseño de la interfaz**

El apartado del diseño de la interfaz se centrará no solo en la apariencia general de la web, sino también en el comportamiento de la misma, así como en sus detalles. Se hará hincapié, en el cumplimiento tanto de requisitos funcionales, como de los casos de uso asociado. Por último se analizarán también los requisitos no funcionales. Hay que hacer mención a que el proyecto, ha sufrido una importante evolución en el aspecto del apartado visual. Por lo que en numerosos, subapartados, se detallarán los cambios realizados en sus diferentes fases. Estos cambios harán referencialos a la versión original del storyboard, la versión beta realizada para la noche de los investigadores del año 2018, y la versión actual de la web.

##### **4.1.2.1 Estilo de la interfaz**

Para hablar de forma correcta del estilo vamos a definir y establecer los aspectos de los elementos web de forma que a la hora de visualizar dicha aplicación de forma gráfica sobre la pantalla de nuestro dispositivo, reconozcamos dichos elementos de forma clara y sin complicaciones añadidas.

1- Colores: Originalmente los colores propuestos serán colores pastel y colores claros, con un fuerte predominio del blanco puro que se corresponderá con la barra de navegación y con el fondo de la aplicación. Esto permitirá resaltar de forma acorde nuestros botones de acción y las distintas secciones del DOM o Document Object Model (estructuración de los objetos dentro de un documento web, que permite el acceso dinámico a dichos objetos), mediante el uso de bordes. La decisión de adoptar estos colores está motivada por la transparencia y luminosidad que arroja el color blanco, la cual resulta acorde a la temática web expuesta.

Durante la segunda y tercera versión del diseño de la web los colores blancos del fondo de la aplicación y de la barra de navegación, serán sustituidos por un tono gris blanquecino para el fondo y un tono casi totalmente negro en la barra de navegación. La elección del cambio de color es debida a dos motivos.

1- La barra de navegación no destaca sobre el fondo de la página web, lo cual puede llevar a

una ilusión de inexistencia, se debe a que lo habitual en las tendencias web actuales sea que exista un claro contraste entre barra de navegación y resto de la aplicación.

2- El ligero oscurecimiento del color blanco del fondo ha sido realizado debido a que el blanco puro, en pantallas con un intensivo brillo, puede resultar molesto para el usuario, por lo que la solución mejor valorada fue su reemplazo por un gris muy claro y de tonalidad mate.

3- Iconos: En la primera versión, en la que los iconos son escasos y llamativos, se opta por indicar realmente que función les corresponde, a fin de no dar lugar a interpretaciones de tipo alguno. Además se destacan del fondo de la barra o del fondo de la aplicación de forma clara, con el propósito de romper un poco la tonalidad blanca predominante y así llamar la atención del usuario para que sepa donde están dichos elementos. Solo el icono de búsqueda y el icono de la aplicación permanecen como tales.

En la segunda y la tercera visión al cambiar los colores predominantes en la aplicación y en la barra de navegación, se opta por hacer que los iconos dejen de ser llamativos, adoptando una línea de diseño sencilla y simple, donde al destacar la propia barra de navegación sobre el resto de la página. El usuario ya es capaz de identificar sin problema los distintos iconos en ella, aún así, en la segunda versión se usará un fondo verde pastel para los iconos de la barra y un fondo azul pastel para los iconos de la aplicación. En la tercera versión los iconos de la barra perderán su fondo y se integrarán por completo en el estilo de la propia barra.

3- Herramientas: Las herramientas disponibles en el programa serán todas aquellas que nos permitan un manejo de las diferentes funcionalidades de la aplicación de forma sencilla y evitando al usuario tener que comprender el proceso existente tras la aplicación.

En la primera versión y la segunda versión las herramientas se organizaron en un submenú, ubicado en el lateral izquierdo de la aplicación. Sin embargo debido a la simplificación y reestructuración de contenidos sufrida por la aplicación web en la tercera versión de la misma, todas las herramientas remanentes han sido movidas a la barra superior o barra de navegación. Por lo que de este modo la interacción principal que realizará el usuario se llevará a cabo con esta barra. Es en esta donde fundamentalmente se recogerán operaciones de cambio de vistas y consultas a la base de

datos.

#### **4.1.2.2 Diseño de pantallas**

Como ya se ha indicado anteriormente un aspecto fundamental del uso de esta aplicación recaerá en el tiempo de aprendizaje, el cual se buscará minimizar todo lo posible. Esto es debido a que la funcionalidad o intención primaria del proyecto no será realizar una aplicación web compleja, sino realizar una investigación sobre las redes neuronales y su comportamiento en la generación de poemas de forma automática, enfocados específicamente en el lenguaje castellano. Para ello la idea es hacer una interfaz web en la que se apoyará dicha investigación, para visualizar los datos del estudio de forma intuitiva y manejable.

En relación la simplificación del manejo de la web se verá que la aplicación presenta varias versiones.

-Menús

Originalmente tanto en la versión del storyboard o Mockup (Maqueta) de la aplicación como en la segunda versión existen los siguientes menús:

##### **1- Menú en la barra de navegación**

Este menú contendrá diversas funcionalidades como son el acceso a la base de datos de poemas, mediante la barra de búsqueda, el acceso a la generación de poemas y el acceso a la traducción al inglés de la aplicación.

##### **2- Menú lateral**

Este menú por su parte será el encargado de proporcionarnos información sobre los distintos poetas, temática poética, país de origen de los poetas (a fin de aportar contexto) y una opción de búsqueda para los últimos poemas generados.

Sin embargo en la última versión se ha optado por eliminar el menú lateral. Se trata de una decisión adoptada para conseguir la simplificación de uso y teniendo en cuenta el objetivo a abordar

por parte del proyecto descrito anteriormente, quedando por lo tanto un único menú.

### 3- Menú en la barra de navegación

Este menú aglutinará todas las funcionalidades, así como el cambio de vistas para pasar a generar poemas, las búsquedas en la base de datos y la traducción al inglés.

#### - Búsquedas

Hemos citado este elemento previamente al enmarcarse dentro del menú de la barra de navegación, pero por sí mismo merece tener una mención propia, ya que esta búsqueda se presenta en doble formato. En este caso sin presentar diferentes versiones al respecto.

#### 1- Indexación de los poemas de la web con carácter general

El primer formato consiste en indexar todos los poemas de la web por medio de una acción de cambio de vista. Esto devolverá todos los poemas generados o hasta un determinado límite a fin de no colapsar la conexión en caso de existir una ingente cantidad de poemas en la base de datos.

#### 2-Búsqueda por parámetros

Este formato consiste en dada una palabra, realizar una búsqueda de dicha palabra como palabra clave dentro de la base de datos y devolver una consulta con aquellos poemas de la base de datos que alberguen dicha clave.

### **4.1.2.3 Storyboards o Mockups**

Una vez tenemos definidos los diferentes elementos de la interfaz de nuestra aplicación, veamos como quedarían en escena dichos elementos. Prestando atención también a los cambios pertinentes que se han sucedido entre las versiones de la aplicación.



-Menú inicial de la aplicación

1ª versión

The screenshot shows the initial menu of the first version of the application. It has a sidebar on the left with the following options: **Poetas**, **País de origen**, **Temática poética**, and **Últimos poemas generados**. The main content area contains six colored buttons for selection: **Selecciona un poeta:** (yellow), **Selecciona una temática:** (cyan), **Selecciona una métrica:** (green), **Selecciona una tipografía:** (orange), **Introduce tu nombre:** (purple), and **Introduce un título:** (grey). Each button has three horizontal lines below it. At the bottom of the main area is a large green button labeled **Generar poema**. The top of the interface includes the 'Autopoetry' logo, a language selector set to 'English', and a search button labeled 'Búsqueda avanzada'.

Imagen 4.2. Menú inicial versión uno. Elaboración propia.

2ª versión

The screenshot shows the initial menu of the second version of the application. It has a dark header with the text 'iMachado' and a language selector set to 'English'. The main content area is divided into three columns. The left column features a black and white portrait of Antonio Machado, followed by the text 'Sobre Antonio Machado' and a list of links: **Página principal**, **Información del autor**, **Temática poética**, **Ejemplos del poemario**, and **Listado de obras del autor**. The middle column contains the heading '¿Qué es iMachado?' followed by a description: 'iMachado es una aplicación capaz de generar a través del uso de una inteligencia artificial un poema de características y estilo similares a las del propio autor sevillano'. Below this is a question '¿Te gustaría servir de inspiración para generar poesía?' and a form with three fields: 'Elige tu propio nombre de autor:' (Nombre del autor), 'Elige el título del poema:' (Título del poema), and 'Introduce una frase de semilla para el poema:' (Introduce una frase (máximo 40 caracteres)). The right column contains a large blue button labeled **Generar Poema**.

Imagen 4.3. Menú inicial versión dos. Elaboración propia.

3º versión



Imagen 4.4. Menú inicial versión tres. Elaboración propia.

Como se puede observar la primera y segunda versión difieren de la tercera en que no se muestra de forma directa la generación del poema, sino que en la tercera versión esta acción se desplaza a una vista íntegramente dedicada a ella, mientras que el menú inicial está dedicado a informar de los diferentes aspectos con los que trabajará la aplicación web.

- Vista del poema recién generado

1ª versión



English

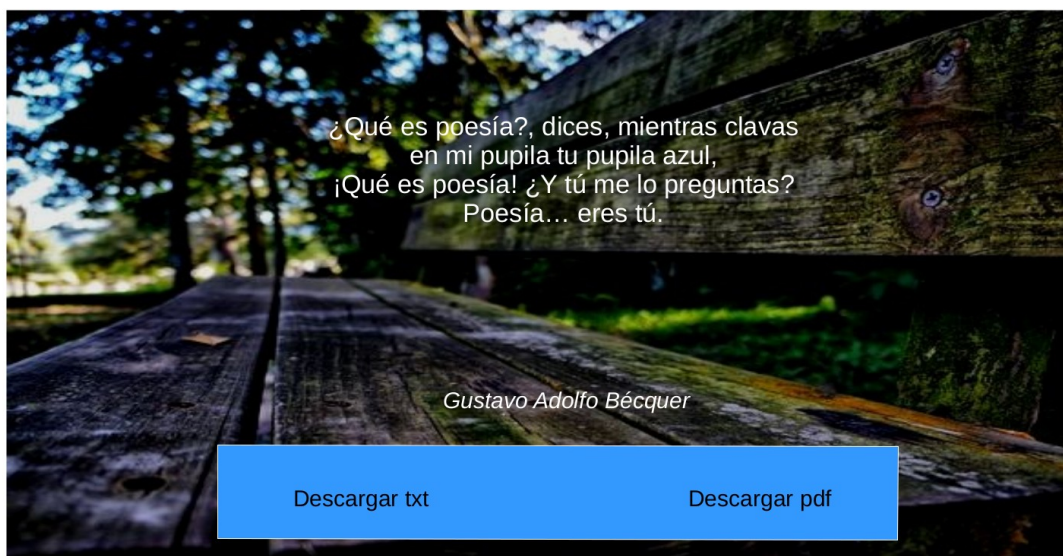


Imagen 4.5. Poema recién generado versión uno. Elaboración propia.

2ª versión

**Sobre Antonio Machado**

- Página principal
- Información del autor
- Temática poética
- Ejemplos del poemario
- Listado de obras del autor

dia  
vimos partir hacia un país lejano .  
hoy tiene ya los sienes plateadas .  
un  
de la la .....  
renacimiento ..... del ..... la ..... de la en ..... en ..... en una el  
penetrar .....  
marcha de años desprecia está ..... era dormía contemplan una uno hijos y da blancos lanza .....  
triste que embrozado ..... capa ..... luz un barba ..... asombra de ha llamaba ..... hijos el reir sobre  
montes ..... dadas conmigo criado ..... mil sobre regalar día marlin la agrios  
tragedias sería ..... la montaña ..... y madre rosa oro que fué o intervalos la recamado ..... con a il los  
ansios antes ..... un vino ..... y motivo que cose realidad ..... vida es los verso eternamente ..... al largo como  
hermano verdadero los ocaso ..... en duelo ..... obra campana ..... miguel tercera vi el os  
Escrito por iMachado  
Semilla dada por  
Poema escrito por iMachado, una inteligencia artificial sencilla basada en redes neuronales que ha  
aprendido de leer, letra a letra, la obra de Antonio Machado  
Noche Europea de los Investigadores - 28 de septiembre de 2018  
Grupo de investigación SINAI (Sistemas Inteligentes de Acceso a la Información) - <http://sinaiujenes>  
Universidad de Jaén

Imagen 4.6. Poema recién generado versión dos. Elaboración propia.

3ª versión

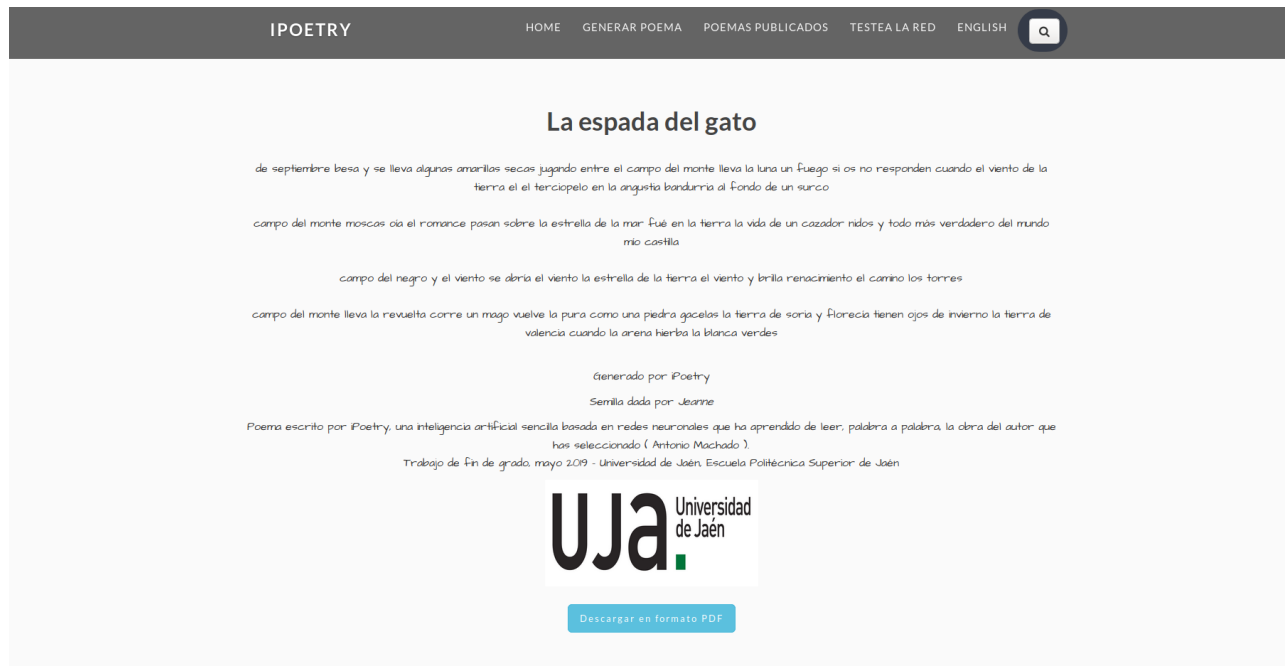



Imagen 4.7. Poema recién generado versión tres. Elaboración propia.

Como se puede observar el poema generado se muestra siempre con un estilo similar de forma que sea lo más amable posible a los ojos del usuario.

- Información sobre la vida del autor y su poemario

1ª versión



English

Búsqueda avanzada

“El alma que puede hablar con los ojos,  
también puede besar con la mirada”

Gustavo Adolfo Bécquer

Información

Información

Poemario

Gustavo Adolfo Bécquer, fue un poeta y narrador español, perteneciente al movimiento del Romanticismo. Por ser un romántico tardío, ha sido asociado igualmente con el movimiento posromántico.

Su obra más célebre son las [Rimas y Leyendas](#), un conjunto de poemas dispersos y relatos, reunidos en uno de los libros más populares de la literatura hispana.

Imagen 4.8. Información poeta versión uno. Elaboración propia.

2ª versión

iMachado

English



Sobre Antonio Machado

- Página principal
- Información del autor
- Temática poética
- Ejemplos del poemario
- Listado de obras del autor

¿Quién fue Antonio Machado?

Antonio Machado Ruiz (Sevilla, 26 de julio de 1875-Colliure, 22 de febrero de 1939) fue un poeta español, el más joven representante de la Generación del 98. Su obra inicial, de corte modernista (como la de su hermano Manuel), evolucionó hacia un intimismo simbolista con rasgos románticos, que maduró en una poesía de compromiso humano, de una parte, y de contemplación casi taoísta de la existencia, por otra; una síntesis que en la voz de Machado se hace eco de la sabiduría popular más ancestral. Dicho en palabras de Gerardo Diego, «hablaba en verso y vivía en poesía». Fue uno de los alumnos distinguidos de la Institución Libre de Enseñanza, con cuyos idearios estuvo siempre comprometido. Murió en el exilio en la agonía de la Segunda República Española.

Biografía

Sevilla

La familia de la madre de Machado tenía una confitería en el barrio de Triana, y el padre, Antonio Machado Álvarez, era abogado, periodista e investigador del folclore, trabajo por el que llegaría a ser reconocido internacionalmente con el seudónimo de «Demófilo». En otra vivienda del mismo palacio son vecinos sus abuelos paternos, el médico y naturalista Antonio Machado Núñez, catedrático y rector de la Universidad de Sevilla y convencido institucionalista, y su esposa, Cipriana Álvarez Durán, de cuya afición a la pintura quedó como ejemplo un retrato de Antonio Machado a la edad de cuatro años.




Imagen 4.9. Información poeta versión dos. Elaboración propia.


En la tercera versión, como ya se ha indicado previamente, no existe un apartado de información ni de carácter poético, ni un listado de poemas de cada autor, ni información en lo referente a la vida del mismo. Esto es así, a causa de que no hemos considerado esta información, como una información de carácter lo suficientemente relevante con relación al objetivo del estudio de la generación de poemas, por lo que será un aspecto a estudiar si se debe incluir en posteriores versiones.

- Consulta parametrizada

1ª versión





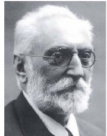
☐ English


Búsqueda avanzada

---

Poetas

---

Miguel de Cervantes

Miguel Hernández

Miguel de Unamuno


---

Títulos de poemas

---



Oda a Miguel

Mi Miguel


Imagen 4.10. Consulta parametrizada versión uno. Elaboración propia.



3ª versión



Imagen 4.11. Consulta parametrizada versión tres. Elaboración propia.

-Consulta a los últimos poemas generados

1ª versión

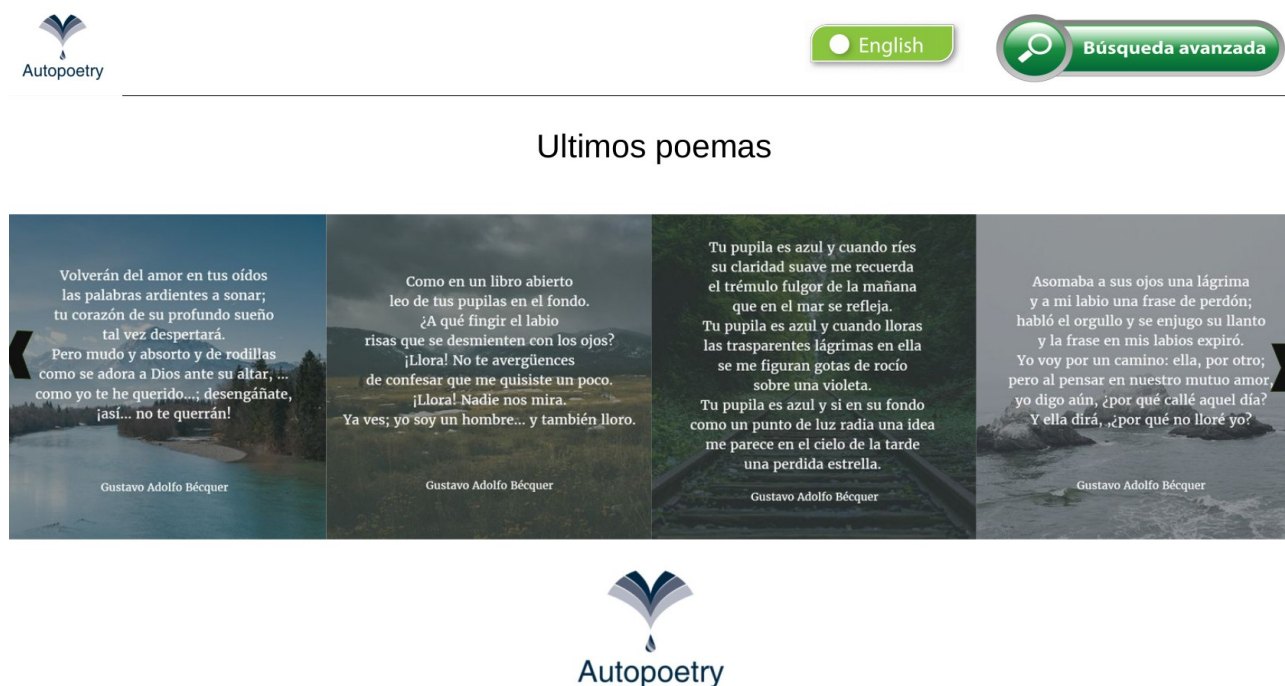


Imagen 4.12. Consulta últimos poemas versión uno. Elaboración propia.

3ª versión



Imagen 4.13. Consulta últimos poemas versión tres. Elaboración propia.

Al tratarse de un modelo experimental, en la versión dos no existe ninguna implementación de las búsquedas tanto parametrizadas como de poemas generales, pues esta no estaba preparada para guardar los diferentes poemas que se iban generando y por lo tanto no disponía de una base de datos acorde para realizar dicha tarea.

Finalmente como novedad en el apartado de diseño cabe destacar una nueva sección de la web incorporada en la tercera versión de la misma. Esta nueva sección es la que se ha orientado al estudio de algunas de las redes neuronales más significativas y que permitirá a los diferentes usuarios de la web la interacción con las mismas, además de la generación de diferentes poemas en base a cada una de ellas y de sus respectivos parámetros.




3ª versión



Imagen 4.14. Comparativa de redes neuronales versión tres. Elaboración propia.

#### 4.1.2.4 Menú de herramientas

Este apartado se centrará en explicar claramente en qué consistirán los distintos elementos de nuestra aplicación, los cuales tendrán una acción particular asociada. Para dicha acción se ha tratado de hacer que las metáforas visuales se reduzcan al mínimo de tal forma que dichos elementos de la web no sean capaces de inducir al usuario a cometer error alguno.

Acción	Vista principal
<b>Descripción</b>	Home, nos rediregirá en todo momento a la vista principal de nuestra aplicación a fin de mostrar información de interés sobre el proyecto.
<b>Visualización</b>	Icono: 
<b>Activación</b>	Se debe hacer clic sobre el icono que se desee usar.
<b>Efecto</b>	Cambio de vista.

Nos redirige a la vista de inicio.

Tabla 4.1. Descripción de la herramienta “Icono home”. Elaboración propia.

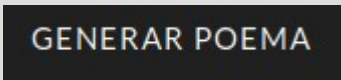
Acción	Vista de generar poemas
<b>Descripción</b>	Este icono nos permitirá trasladarnos a la vista de generar poemas.
<b>Visualización</b>	Icono: 
<b>Activación</b>	Se debe hacer clic sobre el icono que se desee usar.
<b>Efecto</b>	Cambio de vista. Nos redirige a la vista para crear un poema de forma automática.

Tabla 4.2. Descripción de la herramienta “Icono generar poema”. Elaboración propia.


Acción	Vista de testear las redes
<b>Descripción</b>	Este icono nos permitirá trasladarnos a la vista donde podremos probar diversos tipos de redes neuronales.
<b>Visualización</b>	Icono: 
<b>Activación</b>	Se debe hacer clic sobre el icono que se desee usar.
<b>Efecto</b>	Cambio de vista. Nos redirige a la vista para testear redes .

Tabla 4.3. Descripción de la herramienta “Icono testear redes”. Elaboración propia.

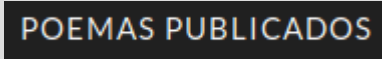
Acción	Vista poemas publicados
<b>Descripción</b>	Este icono nos permitirá visualizar los poemas residentes en la base de datos
<b>Visualización</b>	Icono: 
<b>Activación</b>	Se debe hacer clic sobre el icono que se desee usar.
<b>Efecto</b>	Cambio de vista. Nos lleva a una vista donde se indexarán todos los poemas contenido por la base de datos.

Tabla 4.4. Descripción de la herramienta “Icono poemas publicados”. Elaboración propia.


Acción	Menú principal en inglés
<b>Descripción</b>	Nos llevará al menú de inicio de la aplicación en lengua inglesa.
<b>Visualización</b>	Icono: 
<b>Activación</b>	Se debe hacer clic sobre el icono que se desee usar.
<b>Efecto</b>	Cambio de vista. Nos redirige al menú principal en inglés de la aplicación.

Tabla 4.5. Descripción de la herramienta “Menú principal en inglés”. Elaboración propia.

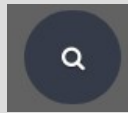
Acción	Barra de búsqueda
<b>Descripción</b>	Embebida en la barra de navegación, la barra de búsqueda nos permitirá hacer búsquedas de poemas por título del mismo.
<b>Visualización</b>	Icono: 
<b>Activación</b>	Se debe hacer clic sobre el icono.
<b>Efecto</b>	Cambio de vista.  Al hacer clic sobre el icono, se realiza una búsqueda sobre la base de datos de poemas indexando únicamente aquellos poemas que contengan la palabra o palabras buscadas en su título.

Tabla 4.6. Descripción de la herramienta “Barra de búsqueda”. Elaboración propia.


Acción	Generar poema
<b>Descripción</b>	Este control es el encargado de generar el nuevo poema.
<b>Visualización</b>	Icono: 
<b>Activación</b>	Se debe hacer clic sobre el icono.
<b>Efecto</b>	Cambio de vista.  Al hacer clic sobre el icono, se mandan los valores del formulario al controlador de back-end, el cual tras unos segundos devolverá el poema generado.

Tabla 4.7. Descripción de la herramienta “Generar poema”. Elaboración propia.


Acción	Imprimir poema
<b>Descripción</b>	Este control es el encargado de imprimir el poema generado.
<b>Visualización</b>	Icono: 
<b>Activación</b>	Se debe hacer clic sobre el icono.
<b>Efecto</b>	Abrir vista secundaria. Al hacer clic sobre el icono, abre una nueva vista que nos da la opción de imprimir el documento generado en formato pdf.

Tabla 4.8. Descripción de la herramienta “Imprimir poema”. Elaboración propia.

#### 4.2 Diseño de la base de datos y elección de tecnologías del lenguaje.

Para diseñar la base de datos se ha partido problema que la atañe, el cual pese a ser de una complejidad elevada no requiere inicialmente de un gran volumen de datos en cuanto al uso intensivo de su base de datos. Además se tratará de que la base de datos sea lo más compatible posible con las tecnologías web de las que haremos uso. Para ello se realizará en primer lugar una revisión de las diferentes tecnologías a utilizar, así como el porqué se han seleccionado dichas tecnologías.

En la aplicación e pueden distinguir tres campos o apartados fundamentales, los cuales requerirán de un tipo de tecnología u otra. Estos campos o apartados serán:

- 1- Front-end.
- 2- Back-end.
- 3- Base de datos.

**- Front – end**

En este apartado será donde se haga uso de un gran número de lenguajes y tecnologías, las cuales combinadas permitirán la construcción de una página web robusta, rápida y agradable de cara al usuario. En general, las tecnologías aplicadas a este apartado, serán las mismas que se apliquen a la mayoría de páginas web que actualmente podemos encontrar online.

El primer aspecto es que se va a prescindir del uso de grandes plataformas como Angular o React. Estas plataformas pese a ser una gran opción como marco de trabajo, no nos interesarán de cara a este proyecto al tratarse de un proyecto centrado en el estudio de las redes neuronales más que en la realización de un gran proyecto web.

Es por tanto que se usarán las siguientes tecnologías y lenguajes de forma combinada:

1- HTML5 → Su elección viene determinada por ser el lenguaje de marcado para hipertexto necesario para construir toda aplicación web.

2- CSS3 → CSS es un lenguaje de hojas de estilos en cascada, el cual resulta especialmente útil a la hora de dar forma y embellecer nuestra aplicación web, porque redimensiona el aspecto original de los elementos básicos de HTML5. Además CSS3 tiene diversos desarrolladores de plantillas y elementos estandarizados como es el caso de Bootstrap 4, el cual ayudará a aportar un estilo agradable a la web de forma mucho más sencilla.

3- JavaScript → JavaScript es un lenguaje de programación asíncrona. Esto va permitir un intercambio de variables entre el front-end y back-end (además del que ya aporta HTML5 por sí mismo a través de los formularios). Además gracias a los scripts en este lenguaje, se podrá dar a la aplicación funcionalidades propias dentro del lado front-end, sin necesidad de tener que forzar una carga excesiva del back-end. Gracias a su interacción con los distintos elementos de HTML y a su posibilidad de modificarlos y operar con ellos y otros valores como cualquier otro lenguaje de programación.

4-JQuery → JQuery al igual que JavaScript es un lenguaje de programación basado en una librería, cuyo objetivo es realizar diversas modificaciones de todos los elementos HTML que

se encuentren en el DOM (Document Object Model).

5- Python → El lenguaje de Python estará embebido directamente en la web, es decir, no será lenguaje Python de forma usual, sino adaptado a las diferentes secciones de código HTML. Este lenguaje al igual que JavaScript y al igual que HTML, permitirá establecer las conexiones y el intercambio de datos e información necesarios entre el lado front-end y el lado back-end.

### **-Back – end**

El apartado back – end será el apartado especializado en la gestión de la base de datos y los controladores. Recordemos que se está siguiendo un patrón de arquitectura MVC, donde la vista será la única parte de dicho patrón que sea colocada en el front – end.

A diferencia del front-end, el back-end suele estar controlado por el mismo lenguaje de programación que se encarga de comunicar los datos con el front-end. Si tenemos como lenguajes de comunicación JavaScript, HTML (ambos puramente lenguajes de navegador) y Python, lo más lógico será utilizar Python puesto que se trata de un lenguaje compatible totalmente con la programación a ambos lados de la web, tanto como en el caso de la programación embebida en el navegador, como en el caso de la programación dedicada a la computación en el lado del servidor.

Además el uso de Python supondrá un elemento clave en la construcción web, al ser un lenguaje altamente compatible con el tipo de base de datos que necesitaremos usar.

También cabe destacar que el uso de Python, proporcionará facilidades extra, como es el uso de uno de sus numerosos frameworks de trabajo. Se trata de FLASK, un marco de trabajo orientado específicamente a páginas web de pequeño tamaño como la nuestra.

Por último se usará la tecnología proporcionada por Keras y Tensorflow. Esta tecnología permitirá la construcción de la red neuronal de una forma mucho más sencilla gracias a los algoritmos de entrenamiento, optimización y activación disponibles en sus librerías.

**- Base de datos**

Antes de colocar el foco de atención sobre la base de datos que se va a escoger, se valorará el por qué tomar dicha decisión. Para ello se va a tratar de atender al volumen de datos y a los diferentes tipos de datos (junto a su distribución en tablas) que se van a necesitar.

Uno de los métodos comunes a la hora de establecer la estructura de la base de datos y los distintos objetos que manejará, es el método del modelado entidad – relación. Para el cual se deberán identificar las distintas entidades intervinientes en la aplicación y las relaciones existentes entre ellas.

**4.2.1 Entidades intervinientes en el modelado de datos**

En este caso las entidades serán todas aquellas que se refieran a objetos propios del uso de la aplicación. Sin embargo debido a la simplificación realizada en el manejo de los datos que se almacenan en la web, para realizar un uso de la misma mas cómodo y sencillo, se puede observar con facilidad que existe solamente un único tipo de objeto relevante, el poema. Con él se realizarán operaciones básicas de tipo CRUD (inserción, modificación, borrado y selección).

En este caso la simpleza en el número de entidades y relaciones entre ellas vendrá impuesta al no existir necesidad alguna de creación de usuarios que deban tener un perfil, ni tener las capacidades de logearse o registrarse en la aplicación. El enfoque que se ha querido dar en este proyecto desde el punto de vista del usuario es de un perfil de usuario anónimo.

**4.2.2 Relaciones intervinientes en el modelado de datos**

Las relaciones pretenden plasmar las interacciones posibles entre las distintas entidades de la base de datos. Pero debido a que solamente se va a hacer uso de una entidad en la base de datos con operaciones básicas, se puede determinar que no existirán relaciones en el modelo de datos.



### 4.2.3 Tablas de datos

Una vez habiendo determinado que la aplicación dispondrá un único objeto, el cual no posee relaciones. Se procederá a establecer qué propiedades ha de poseer dicho objeto para operar con él en base a las necesidades de la aplicación con respecto a la base de datos.

Poema	Descripción	Tipo de variable
<b>Título</b>	Título del poema colocado por el usuario.	String
<b>Autor</b>	Nombre del usuario que ha creado el poema.	String
<b>Poema</b>	Poema generado	String
<b>Fecha</b>	Fecha en formato internacional cuando se creó el poema	ISODate
<b>Identificador</b>	Identificador unívoco del poema	ObjectId

Tabla 4.8. Tabla de datos de poema. Elaboración propia.

### 4.2.4 Justificación de la elección de MongoDB

Una vez teniendo claros los elementos a almacenar por cada poema en la base de datos, se tendrá que realizar una serie de preguntas de competencia para saber qué tipo de base de datos convendrá más a la aplicación.

Antes de realizar preguntas de competencia, se verán brevemente que tipos de bases de datos se pueden encontrar:

#### 1-SQL

Son un tipo de bases de datos muy populares que garantizan siempre las propiedades de atomicidad mediante rollback e integridad de los datos. Sin embargo, presentan problemas de escalabilidad, y ante grandes volúmenes de información pueden sufrir de pérdida de

rendimiento.

## 2-NoSQL

No garantizan la atomicidad e integridad de los datos, sin embargo su fuerza radica en que se trata de bases de datos ampliamente escalables y descentralizadas. Lo que las hace óptimas para grandes volúmenes de datos que no han de tener una relación entre sí.

### **-Preguntas de competencia**

#### 1- ¿He de indexar solamente los poemas por identificador?

Los poemas deberían ser indexados por cualquiera de sus parámetros, bien sea nombre de autor, título, identificador, etc. De hecho el número de identificador el cual será nuestra clave primaria, aporta una relevancia escasa en cuanto a la información del poema que un usuario normal puede obtener de él. En otras palabras, un usuario no va a estar cómodo con una búsqueda por identificador.

Una posible solución sería hacer que la clave primaria o clave de búsqueda (en caso de tener una base de datos de tipo SQL) sea un elemento más representativo, como por ejemplo el título del poema y la fecha de creación.

Otra posible solución sería no utilizar una base de datos SQL, sino una base de datos de tipo NoSQL, la cual nos permite indexar un objeto o entidad por cualquiera de sus campos, sin necesidad de que estos sean únicos. Esto proporcionará la opción a indexar todos los resultados compatibles a nuestra búsqueda, sin tener que conocer tantas características del objeto a buscar.

#### 2- ¿Qué volumen de datos espero que maneje la base de datos de mi aplicación?

El volumen de datos es siempre algo incierto y dependerá en este caso proporcionalmente del uso que se requiera dar a la aplicación en general y del volumen de poemas que se vayan a crear, el cual puede llegar a suponer un volumen considerable.

Una opción es nuevamente elegir SQL. Sin embargo con grandes volúmenes de datos y ante la posible necesidad de escalar el uso de las base de datos, es muy recomendable escoger una base de datos NoSQL la cual presentará menor problema de escalabilidad en caso de albergar un gran volumen de datos.

### 3- ¿Qué es mejor en base a mi modelo de datos relacional?

Debido a que no se poseerá ningún modelo de datos de tipo relacional, la mejor opción sería escoger una base de datos de carácter NoSQL.

### 4- Componente didáctica

Con motivo de la asimilación y aprendizaje en el uso de nuevas tecnologías. A pesar de no tratarse de un elemento real de elección, el uso de bases de datos de carácter NoSQL supondrá un valor añadido en la realización de dicho proyecto para todo el que maneje dicho tipo de base de datos.

#### **-Elección de la base de datos**

En base a todas estas preguntas de competencia y a sus respectivas respuestas podemos determinar que se tratará de una opción más interesante el uso de una base de datos de tipo NoSQL frente a una base de datos de tipo SQL.

Ahora bien existen en el mercado una amplia multitud de bases de datos de tipo NoSQL, en este caso la elección de dicha base de datos va a venir marcada por la compatibilidad y fiabilidad conocidas con respecto al lenguaje back-end que se ha decidido usar, ya que esta base de datos se tendrá que comunicar con el controlador a través del modelo en dicho lenguaje.

Habiendo establecido Python como lenguaje de programación back-end, y conociendo la popularidad de la base de datos MongoDB, se ha decidido utilizar dicho tipo de base de datos junto a la librería o paquete de conexión con Python, llamado PyMongo.

#### 4.2.5 Estudio de volumetría

Un aspecto que adquiere especial relevancia, es el del crecimiento potencial del almacenamiento de datos. Por este motivo, a continuación se realizará un pequeño estudio sobre el posible crecimiento y evolución de la base de datos de la aplicación.

##### 4.2.5.1. Memoria necesaria por registro y por tabla

Los objetos que se van a usar en nuestro base de datos de Mongo serán objetos de tipo BSON (extensión de JSON), es decir, aquellos que no se pueden encontrar en los modelos de objetos básicos de JSON (JavaScript Object Notation) como fechas o Object ID.

1-ObjectId: Se trata de un objeto pequeño, único en cada tabla de uso, rápido de crear y ordenado. Consta de 12 bytes, se usa de identificador.

2- String: Utiliza UTF-8 para almacenar toda clase de caracteres internacionales. Consta de 32 bits + 1 byte de cabecera (5 bytes).

3- IsoDate: Representa las fechas en mongo en formato UTC. Consta de 64 bits (8 bytes).

Cabe recordar, que en base a este tipo de objetos los cuales aparecerán en el objeto poema, el cual estará compuesto por:

1-ObjectId: Solamente uno, que será el identificador del poema.

2- String: Tres en total, correspondientes con los datos de título, autor y contenido del poema.

3- IsoDate: Solamente uno, que almacenará la fecha de creación del poema.

Tabla poema	Memoria por registro	Número de registros inicial	Tamaño total inicial
<b>ObjectId * 1</b>	12 bytes	0	0KB
<b>String * 3</b>	5 bytes * 3 = 15 bytes	0	0KB
<b>IsoDate * 1</b>	8 bytes	0	0KB
<b>Total</b>			<b>0KB</b>

Tabla 4.9. Volumen inicial de datos almacenados. Elaboración propia.

Ahora se estimarán una serie de registros sobre la web anuales en base a la población hispano-parlante a la cual se dirigirá el uso de esta aplicación y al interés global por tener poesía de forma automática generada por ordenador. Dicho interés será establecido en base a un criterio decidido por nosotros mismos como una tasa lineal del 2% de la población hispano-parlante.

En 2017 esta población se cifró según el instituto Cervantes en 572 millones de personas, suponiendo que ese dato se haya mantenido estable. Se puede calcular que un 2% de dicha población supondrá que esta aplicación tendrá un potencial foco de interés en aproximadamente 11,4 millones de personas al año. Suponiendo que cada persona generará al menos un poema basado en uno de nuestro autores, se tendrán 11,4 millones de poemas nuevos cada año. Veamos por lo tanto, que supondría esta tasa de poemas nuevos en volumen de datos para la aplicación.

Tabla poema	Memoria por registro	Número de registros inicial	Tamaño total inicial
<b>ObjectID * 1</b>	12 bytes	11,4 millones	136.8 MB
<b>String * 3</b>	5 bytes * 3 = 15 bytes	11,4 millones	171 MB
<b>IsoDate * 1</b>	8 bytes	11,4 millones	91,2 MB
<b>Total</b>			<b>399 MB</b>

Tabla 4.10. Crecimiento anual estimado de los datos almacenados. Elaboración propia.

Se puede estimar por tanto, que en base a los datos supuestos anteriormente, la base de datos crecería linealmente entorno a los 400 megabytes por año. En principio, este volumen de información no sería muy elevado. De todas formas, al haber escogido MongoDB como base de datos, se puede asegurar que dicha generación de información, no debería llegar a representar un problema a corto y medio plazo, por lo que este hecho, sumado al uso que vamos a realizar de la base de datos (consultas e inserciones), puede asegurar que MongoDB será una opción idónea.

### 4.3 Arquitectura de la red neuronal

En cuanto a la arquitectura de la red neuronal se usará una red neuronal de tipo recurrente. Uno de sus mayores beneficios, lo se encontrará al prestar atención en uno de sus subtipos

especialmente.

### -Tipos de redes neuronales recurrentes

1- Redes neuronales simples (SRN): Este tipo de redes neuronales recurrentes son usadas como base para el desarrollo e implementación de otras redes recurrentes más avanzadas. Su característica fundamental, es el uso de la retroalimentación o backpropagation.. Principalmente se focaliza su uso en reconocimiento de voz y reconocimiento de escritura a mano.

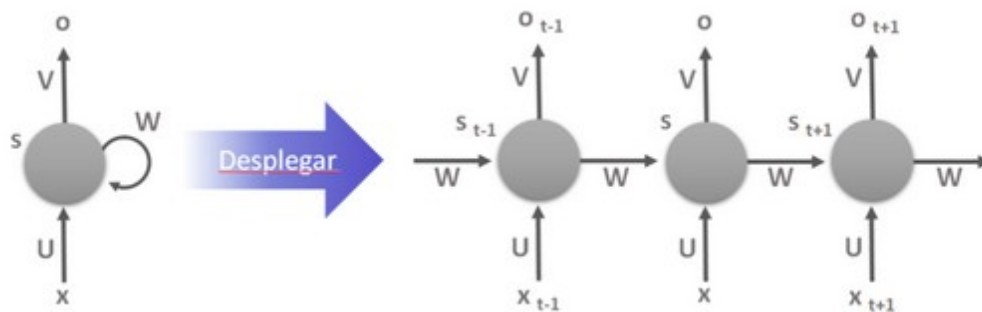


Imagen 4.15. Redes neuronales simples. Fuente [www.diegocalvo.es](http://www.diegocalvo.es).

2- Redes LSTM (Long Short Term Memory) : Las redes neuronales convencionales presentan diversas complicaciones a la hora de la retroalimentación, debido a que los gradientes de retropropagación crecen o decrecen de forma excesiva con el tiempo, lo cual provoca arrastrar errores con el tiempo.

Las redes LSTM gestionan la información para borrar aquellos gradientes que no nos dan buenos resultados y almacenar aquellos que sí.

Sus características fundamentales serán las puertas de entrada, olvido y salida. La puerta de entrada controla el acceso de nueva información a la red. La puerta de olvido controla la eliminación de datos que no generan buenos resultados, a fin de establecer una discriminación entre datos buenos y malos, y así dejar paso a nuevos datos de entrada. Finalmente la puerta de salida controla cuando usar la memoria de los datos almacenados para generar nuevos resultados.

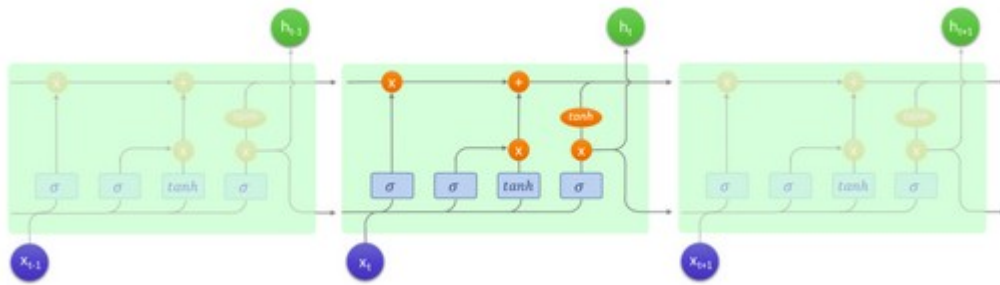


Imagen 4.16. Redes neuronales recurrentes LSTM. Fuente [www.diegocalvo.es](http://www.diegocalvo.es).

3- Redes GRU (Gated recurrent unit): Este tipo de redes son más simples que las LSTM debido a que tienen menor número de parámetro, no presentan puertas de salida y pueden capacitar la red neuronal mas rápidamente.

Sus principal diferencia es que poseen funciones de actualización y reajuste. La actualización indicará qué contenido permanecerá en la red de cara a la siguiente entrada de datos, mientras que el reajuste definirá como incorporar el contenido anterior en la red.

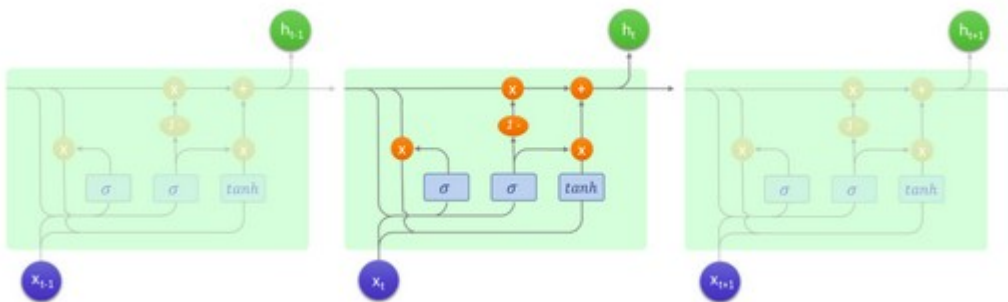


Imagen 4.17. Redes neuronales recurrentes GRU. Fuente [www.diegocalvo.es](http://www.diegocalvo.es).

### -Elección de la red neuronal

Por el gran volumen de datos del que se dispondrá generalmente del orden de entre 8 y 10 mil líneas de texto versado por autor, el subtipo mas útil de red neuronal recurrente es el LSTM (Long Short Term Memory) o redes de memoria a corto plazo. Este tipo de redes registran una pequeña memoria capaz de recordar los datos mas relevantes de los mejores pesos entre las distintas iteraciones de su ejecución. Este hecho permitirá solventar el hecho de que grandes redes neuronales recurrentes con grandes volúmenes de información, suelen perder información relevante a lo largo de las iteraciones. La memoria a corto plazo de la red va por lo tanto renovándose de forma

constante con los mejores datos de pesos registrados. Además este hecho facilita la implementación del recorrido de la red mediante propagación hacia atrás.

### 4.3.1 Dibujo orientativo de funcionamiento

A continuación se observará un poco mas en detalle el funcionamiento de una red LSTM.

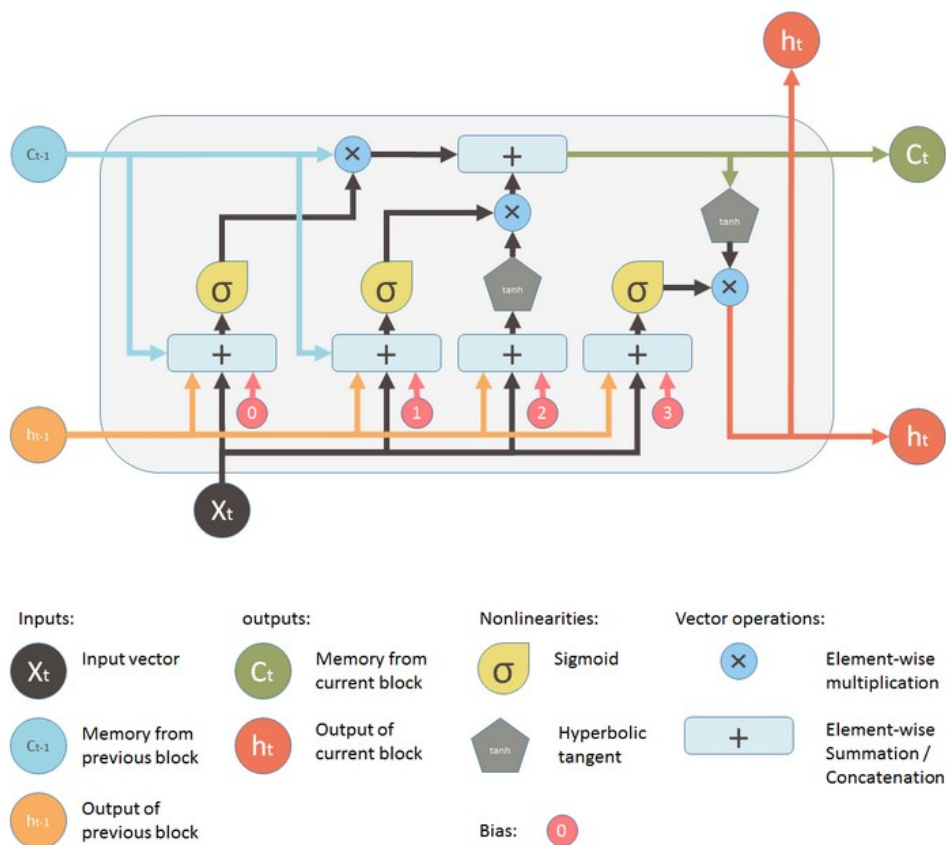


Imagen 4.18. Red neuronal LSTM en detalle. Fuente [www.medium.com](http://www.medium.com).

Como se puede observar en la imagen la red tomará tres valores de entrada (puerta de entrada):

- 1- Un vector de entrada con el paso actual de ejecución.
- 2- Memoria procedente de la ejecución anterior.
- 3- Valor de salida de la ejecución anterior.

Esta red generará en base a estos valores y a los valores del bias y la función sigmoid asociada



introducida en la red, unos nuevos valores de memoria y de salida.

En la parte superior de la imagen, se puede observar una línea de ejecución entre los valores  $C_{t+1}$  y  $C_t$ , esta línea de ejecución será la encargada de controlar y unificar la memoria de los datos generados y la memoria antigua a fin de decidir que memoria permanecerá en la siguiente ejecución (puerta de olvido).

Finalmente, para generar la salida de la ejecución actual, vuelve a seguir un proceso similar al de la primera etapa o puerta de entrada, la diferencia será que esta vez se usará la memoria que hemos generado en esta ejecución, en vez de la generada en la ejecución anterior. De esta forma se generará la salida que se usará en la siguiente ejecución (o la salida que en la última de las ejecuciones programadas se guardará en el modelo de datos).

## **5.Implementación**

Habiendo finalizado la fase de diseño de la aplicación, se puede comenzar con la implementación de todos los elementos definidos. La implementación se detallará por lo tanto al final de este capítulo de la memoria junto a todo el proceso de desarrollo del proyecto.

Se va a proceder a describir los detalles de la implementación así como los diferentes aspectos de la metodología de desarrollo aplicada.

### **-Metodología de desarrollo**

Como ya se ha mencionado anteriormente la metodología a usar será una metodología iterativa o metodología ágil bajo el marco de trabajo web de Scrum. Estas metodología y marco de trabajo se basan en el desarrollo iterativo de los proyectos por medio del troceamiento o division de las diferentes tareas a realizar de cada uno de ellos. De forma que los errores se puedan ir corrigiendo sucesivamente en cada entrega realizada cada pequeño intervalo de tiempo y a fin de tener un producto funcional en etapas anteriores a la propia finalización del mismo. Después de dicha finalización, el producto solamente necesitará ir siendo revisado y actualizado periódica y continuamente en base a las necesidades del usuario, por lo que el proceso de completitud del proyecto no podrá alcanzar nunca el 100% mientras el cliente siga demandando un mantenimiento de dicho producto.

### **5.1 Entorno de programación**

Teniendo en cuenta los lenguajes de programación mencionados en el apartado de selección de tecnologías y lenguajes. El marco de programación elegido será Flask. Dicho marco de trabajo resulta bastante simple de usar, además de proporcionar una gran potencia y versatilidad para pequeñas aplicaciones web desarrolladas en Python. Python es un lenguaje excelente para desarrollar este proyecto, no solo por el hecho de su versatilidad en cuanto a la comunicación con los lenguajes web, sino porque además resultará excelente a la hora de construir nuestra red neuronal mediante el uso de Tensorflow y Keras.

## 5.2 Controladores y tabla de acciones

Los controladores serán los encargados de realizar las tareas computacionales más costosas de la aplicación web. Para entender mejor el funcionamiento de los mismos se observará por medio de diagramas, sus distintas interacciones y comunicaciones con las vistas y los modelos.

Tendremos ocho controladores en la web:

- 1- Controlador principal
- 2- Controlador encargado de mostrar el formulario que crease el poema
- 3- Controlador encargado de crear el poema
- 4- Controlador encargado de indexar todos los poemas
- 5- Controlador encargado de indexar poemas por palabras clave
- 6- Controlador encargado de comparar redes neuronales
- 7- Controladores bilingües
- 8- Controladores de errores

- 1- Controlador principal

Este controlador es el encargado de cargar el menú principal. Además en caso de ser la primera vez que la web sea cargada en un servidor o en caso de querer realizar una exportación de datos por motivos de seguridad o traslado de la base de datos, este controlador será el encargado de realizar las llamadas a las funciones de exportación e importación de la base de datos de poemas. Durante el habitual trascurso de la web estas llamadas a las funciones permanecerán deshabilitadas a fin de evitar sobrecargas durante el funcionamiento de la aplicación.

Este controlador no mantendrá por lo tanto el patrón MVC al uso. Su funcionalidad no será establecer un intercambio de datos con el modelo para luego mostrarlos en la vista, sino que simplemente es el encargado de realizar de forma puntual interacciones con dicho modelo a modo de mantenimiento del conjunto de la web.

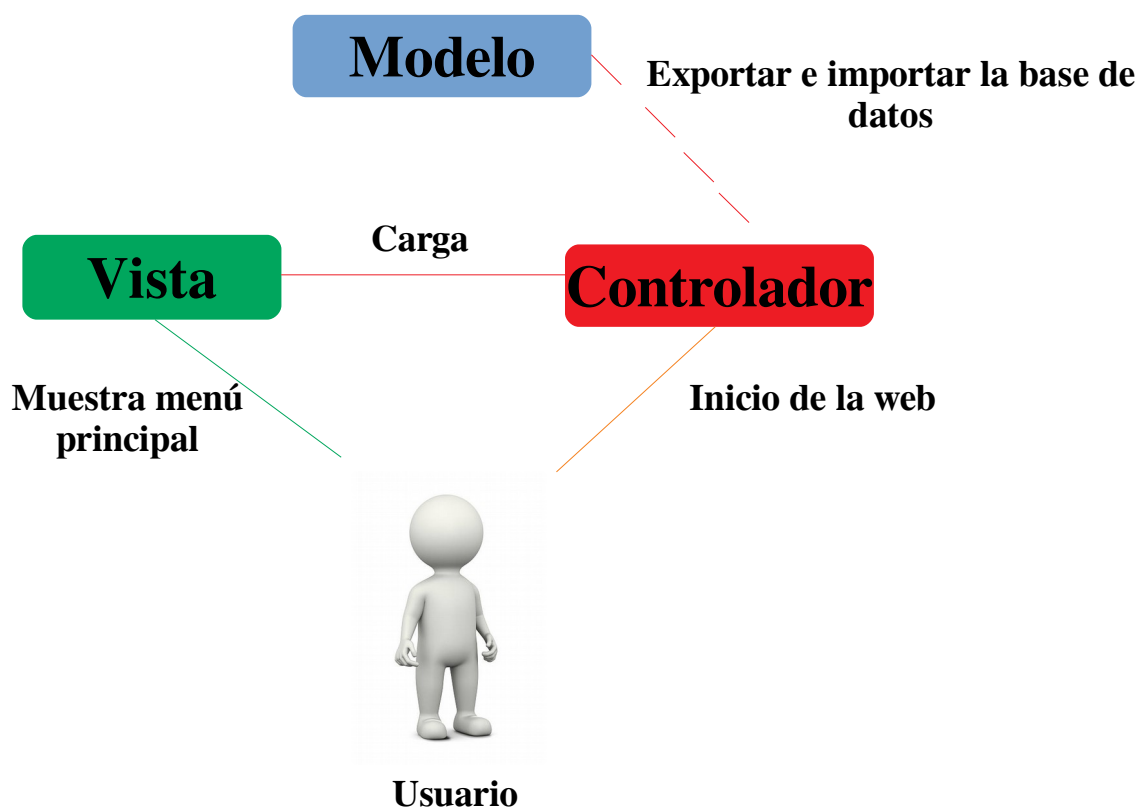


Imagen 5.1. Controlador principal. Elaboración propia.

## 2- Controlador encargado de mostrar el formulario que crease el poema

Este controlador será el encargado de llamar a la vista que mostrará el formulario a rellenar por el usuario. En este caso este controlador no realiza ningún tipo de actividad con el modelo por lo que se puede decir que su estructura es simplemente Vista – Controlador.

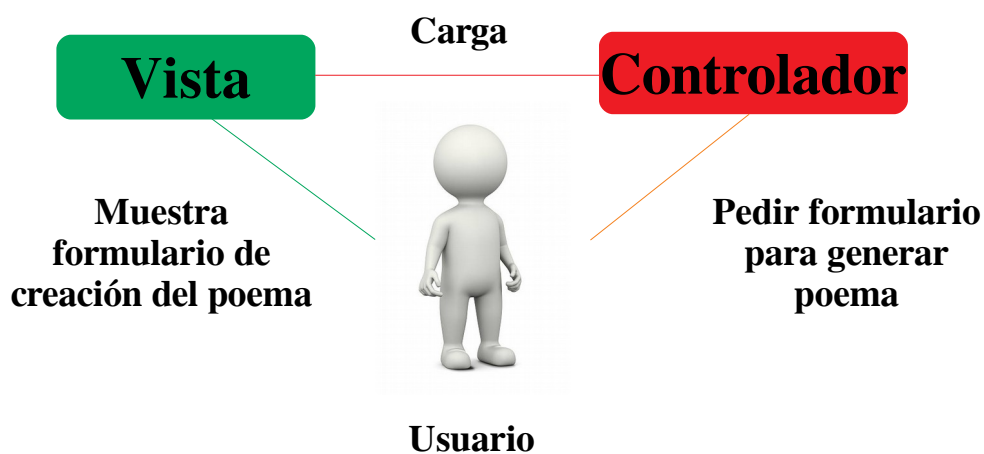


Imagen 5.2. Controlador encargado de mostrar el formulario. Elaboración propia.

### 3- Controlador encargado de crear el poema

Este controlador a diferencia del encargado de mostrar el formulario, traslada los datos de dicho formulario a la red neuronal y se encarga de adaptar el texto generado por la red neuronal. A un formato más legible al que además añadirá los diferentes elementos extraídos del formulario y que no serán útiles para la creación del poema. Por último, además de mostrar el poema al usuario, también lo insertará en la base de datos.

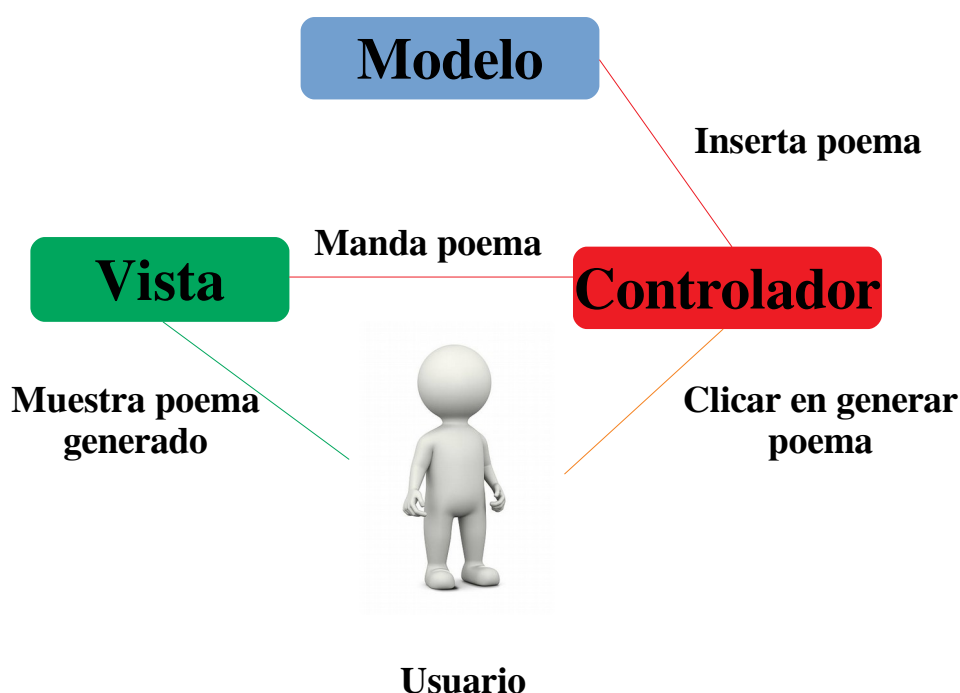


Imagen 5.3. Controlador encargado de crear el poema. Elaboración propia.

### 4- Controlador encargado de indexar todos los poemas

Este controlador será el encargado de indexar toda la información disponible en la base de datos de la web hasta un determinado límite. La consulta a realizar será genérica, de tal forma que los elementos que se mostrarán posteriormente en la vista serán aquellos que se hayan generado más recientemente.

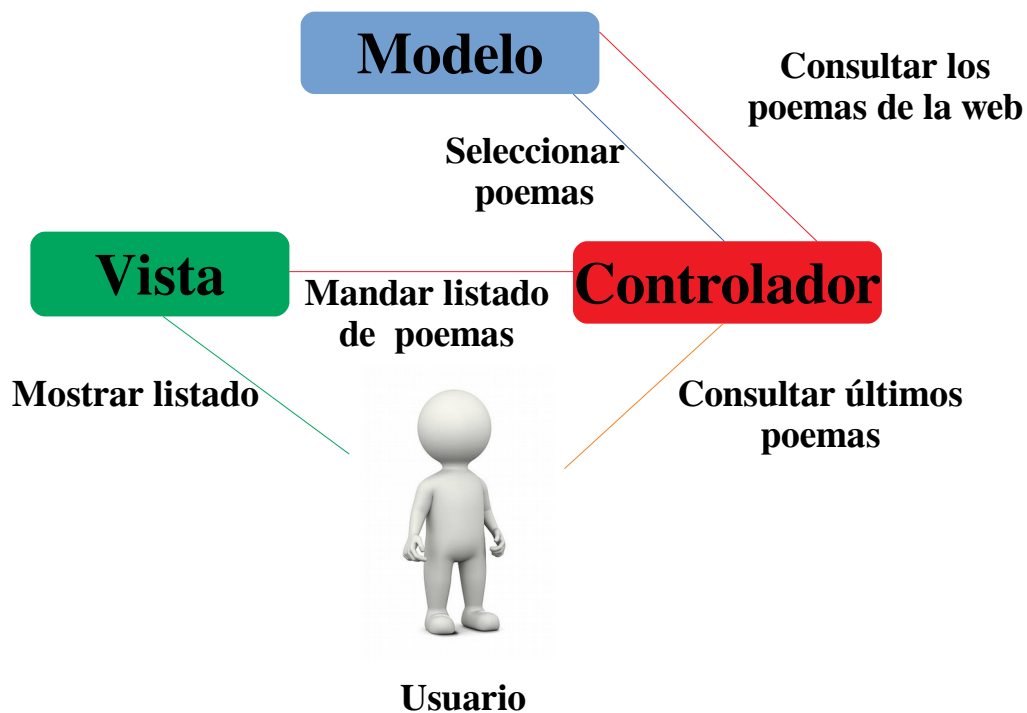


Imagen 5.4. Controlador encargado de indexar poemas. Elaboración propia.

#### 5- Controlador encargado de indexar poemas por palabras clave

Este controlador realizará una búsqueda similar a la que hace el controlador anterior. La única diferencia entre ambos residirá en el uso de una palabra clave como elemento de búsqueda.

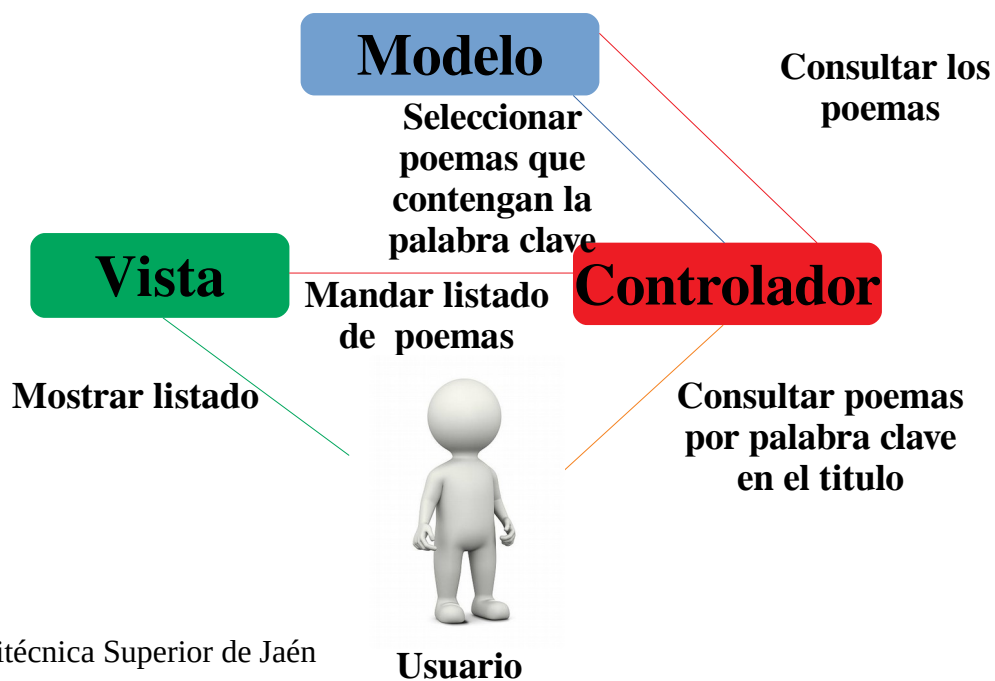


Imagen 5.5. Controlador encargado de indexar poemas por palabras clave. Elaboración propia.

#### 6- Controlador encargado de comparar redes neuronales

Este controlador realizará una función similar al controlador encargado de generar poemas. Su diferencia residirá en que en vez de escoger modelos de una misma red neuronal, escogerá diferentes modelos de diferentes redes neuronales con diferentes parámetros de entrenamiento.

Además este controlador no ofrecerá opción al usuario a elegir diversos autores, ya que el objetivo será establecer una comparación entre diversos poemas generados para un mismo autor con un mismo corpus.

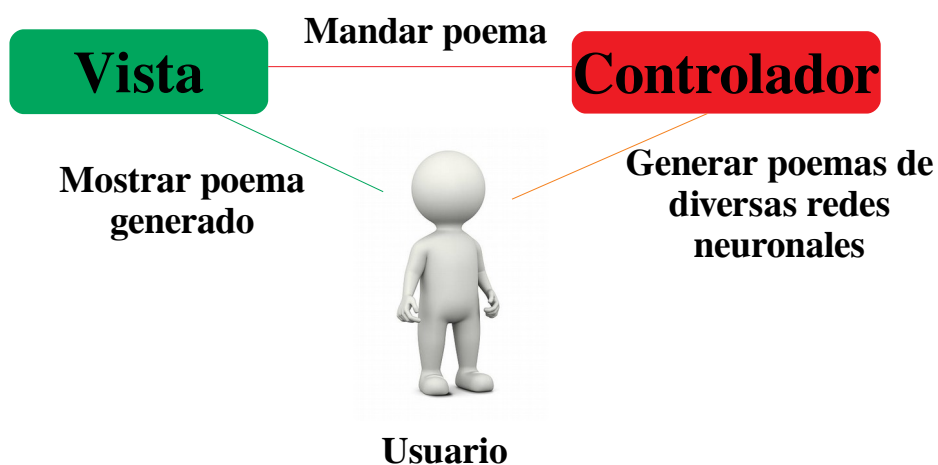


Imagen 5.6. Controlador encargado de comparar poemas de diversas redes. Elaboración propia.

#### 7- Controladores bilingües

Los controladores bilingües estarán encargados de realizar la misma serie de funciones que los controladores en español. La diferencia entre estos, será que se encargen de manipular las vistas en inglés en nuestro caso, pero trabajando sobre el mismo modelo de datos y las mismas redes neuronales. El objetivo de disponer de estos controladores no será otro que dotar de internacionalización a la web.

Al poseer el mismo funcionamiento que sus homólogos en lengua materna, en este caso el castellano, seguirán la misma estructura de funcionamiento.

## 8- Controladores de errores

El objetivo de los controladores de errores no será otro que el de gestionar diversos imprevistos provocados por acciones no previstas dentro del flujo natural de la aplicación.

Cuando dichos controladores se accionen deberán informar de forma escueta al usuario sobre el fallo producido, así como reconducir al menú inicial a los usuarios a través de una acción determinada.

Se manejarán los siguientes errores básicos:

- 1- Error 403 – Forbidden Access: Acceso denegado
- 2- Error 404 – Page Not Found: Página o sección de la web no encontrada
- 3- Error 405 – Method Not Allowed: Acción del usuario indebida
- 4- Error 500 – Internal Server Error: Error interno del servidor

### **5.3 Front-end**

En el análisis ya tratamos brevemente el apartado de front-end definiendo cuales serían las tareas de las que el apartado front-end tendría que encargarse. Además también tratamos las tecnologías y lenguajes que intervendrían en el desarrollo de este apartado fundamental de nuestra aplicación.

El front-end se podría definir como la capa de aplicación encargada de visualizar los datos e información que reside en nuestra base de datos. De igual modo estaría también encargada de proporcionar herramientas visuales para la interacción con los usuarios de la forma más sencilla y cómoda posible para ellos.

#### **5.3.1 Vistas**

Inicialmente vamos a realizar una implementación de las diferentes vistas de la aplicación de



forma estática, es decir, sin conexión real con la base de datos ni con el controlador encargado de generar poemas. El objetivo de comenzar por esta parte será el de adaptar la web para que esta sea completamente funcional antes de interaccionar de forma automática con la base de datos y el generador de poemas (en nuestro caso, nuestra red neuronal), y que no se produzcan errores derivados de la parte web al conectar la base de datos o la red neuronal y que esto nos lleve a error. Del mismo modo cabe recordar que la creación de la base de datos y de la red neuronal es independiente de la creación de la aplicación web, por lo que son procesos que se pueden desarrollar en paralelo.

### **5.3.2 Layout**

Una parte fundamental de la aplicación web será el uso de layouts que estructuren de forma sencilla cada una de las diferentes vistas de la aplicación. Esto se debe hacer de forma que se comporten de manera similar sin perder la forma de la vista a la hora de redimensionar la ventana del navegador correspondiente y evitar así fallos por solapamientos de las diferentes secciones de la web. Este layout deberá cumplir con todos los aspectos detallados en el storyboard de la forma más fidedigna posible.

Para un correcto cumplimiento y mantenimiento de la estructura en procesos de redimensionamiento de la ventana del navegador, se introducirá un modelo de celdas o marcos de cajas, los cuales subdividirán el espacio disponible en el DOM, y lo asignarán a las distintas secciones de código intervinientes en dicho DOM en todo momento. Al estar hecha esta división en base al porcentaje de la ventana, este porcentaje siempre podrá mantenerse proporcional a pesar de aumentar o disminuir el número total de pixels asociados a la pantalla.

El modelo de celdas se llevará a cabo mediante los elementos div, pertenecientes a CSS, estos elementos, a veces extremadamente costosos de manejar y de adaptar de forma correcta, serán clave en una buena adecuación estructural de la información a mostrar en la web.

### **5.3.3 Campos de texto**

Dicho elemento, se encuentra únicamente presente en la vista de generación de poemas, y en la barra de búsqueda dentro del menú de navegación. En este caso, distinguiremos entre dos tipos de

campos de texto, los de selección y los de relleno manual.

1- Los campos de selección, presentan un desplegable con múltiples opciones, encargadas de establecer la elección de una única opción propiamente formateada para que a la hora de generación del poema, este sea elegido de entre los modelos disponibles y no genere un fallo en el sistema.

2- Los campos de relleno manual, serán opcionales, es decir, no es necesario obligar al usuario a escribir nada en ellos para generar el poema. Además se podrá encontrar en todo momento, la barra de búsqueda, la cual también es opcional.

Cabe destacar, que los datos introducidos en los campos de relleno manual, no presentan grandes restricciones, mas allá de haber limitado el campo de la semilla a 40 caracteres. Los datos pueden ser introducidos en cualquier formato dentro de la codificación de caracteres UTF-8 y en cualquier disposición.

#### **5.3.4 Menú de herramientas**

Inicialmente se optó porque la aplicación consistiese en dos menús de herramientas, uno situado en la barra de navegación de la aplicación web y el otro situado en el lateral izquierdo de la web.

El menú de la barra de navegación se iba a encargar de realizar consultas parametrizadas a la base de datos y de la traducción al inglés de la web. Mientras que el menú lateral izquierdo se iba a dedicar a la gestión de la información de cada uno de los autores de la web seleccionados.

Sin embargo durante el proceso de desarrollo de la aplicación se ha ido determinando que era una mejor opción en términos de sencillez, el incluir el menú lateral en la propia barra de navegación.

A su vez se ha de destacar que el menú de herramientas, actuará como un menú de pestañas. Esto lo hará ofreciendo cada una de las funcionalidades o herramientas importantes de la aplicación por separado, al mismo tiempo que se alterna entre las diferentes vistas existentes siguiendo la

filosofía de una pestaña para cada acción importante en la web como por ejemplo: generar poema, imprimir poema, buscar poemas, etc.

Por último cabe resaltar el hecho de que solo se encontrará navegación por cuatro pestañas desde la barra de navegación a las que se sumarán la pestaña de traducción y la barra de búsqueda. Esto dará a la aplicación un aspecto más familiar que va en correlación con la tendencia de las distintas aplicaciones existentes en el mercado las cuales todas presentan similares formatos.

## **5.4 Back-end**

Una vez teniendo claro lo realizado por el front-end de la aplicación, se procederá a definir las tareas serán las encargadas de hacer por nuestro back-end, el cual será el encargado de albergar los controladores de la aplicación y la comunicación con la base de datos.

### **5.4.1 Controladores**

Los controladores son el elemento clave dentro de la comunicación en cualquier diseño que siga el patrón MVC. Los controladores, permitirán manejar el intercambio de datos entre el front-end y la base de datos, además de ser capaces de realizar tareas de computación, el procesado de textos. Por último serán estos los que nos permitan movernos entre las diferentes vistas por medio de sus invocaciones.

### **5.4.2 Modelo y operaciones con la base de datos**

El modelo será el encargado de recibir las acciones del controlador y de realizar de forma directa la comunicación con la base de datos por medio de inserciones o consultas.

## **5.5 Conexión Front-end – Back-end**

En la conexión entre front-end y back-end vamos a realizar mayor incapié en dos partes fundamentales, la conexión a la base de datos por parte de nuestra aplicación y el intercambio de información entre las dos partes del sistema y la propia base de datos.

### 5.5.1 Conexión a la base de datos

Para realizar dicha conexión se va a utilizar la ruta de la base de datos la cual dejaremos establecida como variable de configuración del siguiente modo:

```
app.config['MONGO_URI'] = "mongodb://localhost:27017/poemdb"  
mongo = PyMongo(app)
```

De este modo se enlaza la configuración de la aplicación a la base de datos que va a ser usada con mongo, en este caso la base de datos concreta será llamada poemdb. De ahora en adelante todo el proyecto utilizará la variable mongo como acceso.

### 5.5.2 Intercambio de información entre front-end y back-end

El intercambio de información entre la base de datos se realiza directamente con la parte back-end de la aplicación. Dicha parte se comunica con la base de datos por medio del archivo models.py, el cual contiene las operaciones de consulta y de inserción.

El archivo models.py se comunica a su vez con el controlador pertinente en el archivo routes.py, el cual se encarga de procesar la información extraída de la base de datos, de generar poemas nuevos, e insertarlos en la base de datos y de cargar o cambiar de vistas durante la navegación.

Son por lo tanto los controladores los que se encargarán del intercambio propio de información entre las diferentes partes de la aplicación.

## 5.6 Redes neuronales implementadas

Se dispondrá de dos redes neuronales diferentes en la aplicación web una red neuronal basada en caracteres, y una red neuronal basada en palabras. La motivación del desarrollo de ambas redes neuronales será la comparación en los resultados obtenidos de ambas.

Una de las motivaciones de realizar dicha comparación será la de observar si la red neuronal de caracteres es capaz de con un simple vocabulario de caracteres pertenecientes a la lengua castellana, obtener combinaciones de caracteres lo suficientemente buenos como para formar texto legible por los usuarios además de poseer sentido. Además la red de caracteres será mucho menos pesada en cuanto a la información de la que ha de disponer para generar un nuevo poema.

Las ventajas a priori de la red neuronal de palabras serán la mejor generación de textos. Esta mejor generación estará basada en que una predicción de frases o texto, es más fácil de realizar en base a solo palabras. El problema fundamental de la red de caracteres será que no solo hay que generar predicción de palabras sino también predicción de caracteres, lo cual puede llevar a producir un mayor error.

### **- Diferencias**

Las diferencias fundamentales vendrán a la hora de asignar los pesos de las palabras y de construir la red neuronal.

Sobre código se verá que la red neuronal de caracteres se utiliza tanto como para entrenar modelos como para generar nuevo texto a raíz de esos modelos previamente generados.

La red neuronal de caracteres poseerá la siguiente estructura:

```
model.add(LSTM(128, input_shape=(maxlen, len(chars))))  
model.add(Dense(len(chars)))  
model.add(Activation('softmax'))  
optimizer = RMSprop(lr=0.01)  
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

El modelo a usar será un modelo de Keras el cual estará compuesto de 128 capas ocultas con una densidad igual a la longitud de caracteres introducidos y una activación 'softmax'. Estos valores han sido seleccionados tras realizar una pequeña búsqueda sobre los valores que mejores resultados producen en una red de tipo LSTM para generación de texto.

Sin embargo tras observar que los resultados efectivamente distan mucho del objetivo a alcanzar. Se ha optado por cambiar de red neuronal de caracteres a una red neuronal de palabras. La red neuronal de palabras presenta tres diferencias principales respecto a la red de caracteres.

#### 1- División entre red de entrenamiento y red de generación.

Se ha optado por diferenciar de forma clara, entre las partes de entrenamiento y generación. La motivación de realizar dicha división será no solo la simplificación del código, sino también la simplificación de las llamadas desde el controlador de la aplicación web encargado de comunicarse con la red neuronal.

#### 2- Establecimiento de pesos anteriormente pre-entrenados.

La red neuronal de palabras utilizará unos pesos pre-entrenados para asignación de palabras similares, especialmente adaptados para la lengua castellana y almacenados en una matriz “word2vec”. Estos pesos serán utilizados para ir variando las palabras generadas del texto.

#### 3- Mayor complejidad estructural.

Su estructura será la siguiente:

```
model.add(pretrained_embedding_layer(word_to_vec_map, word_to_index))
model.add(Bidirectional(LSTM(128, return_sequences=False, dropout=0.1,
recurrent_dropout=0.1)))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(vocab_size+1, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy')
```

Los valores de la red han sido tomados inicialmente partiendo de los valores utilizados por la red de caracteres. Sin embargo podemos apreciar un mayor número de elementos, así como una doble activación y dropout. El objetivo de esta serie de elementos será el de capacitar a la red con una mayor variabilidad para así poder dar con resultados más diversos.

## **6. Validación y evaluación**

Una vez finalizado el proceso de implementación de la aplicación procederemos a realizar la fase de validación y pruebas de la aplicación. Dicha fase se realizará mediante el testeo exhaustivo de la propia aplicación. Durante el cual se tratará de identificar los posibles fallos y subsanarlos. Además será muy importante el testeo de la red neuronal, con el fin de identificar los mejores parámetros de configuración que nos aporten las mejores soluciones.

### **6.1 Requisitos del sistema**

Para comenzar el testeo realizaremos un breve repaso de las diferentes tecnologías de las que necesitaremos disponer en nuestro sistema. A fin de poder realizar dicho testeo en cualquier ordenador.

#### **1- Sistema operativo**

Necesitaremos un sistema operativo que nos permita realizar la ejecución de scripts de Python mediante consola. Esto es: Sistemas operativos Linux, Mac Os o Windows 10. Para sistemas operativos del tipo Windows 8.1, 8 o 7, la opción más cómoda, sería la de instalar el programa Anaconda, el cual integra de por sí Python junto a una consola de ejecución propia.

#### **2- Navegador de acceso a internet**

Cualquier navegador del tipo Google Chrome, Mozilla Firefox, Bing, Opera o similar en su versión actual será suficiente para aprovechar todas las tecnologías que la web utiliza en su parte front-end.

#### **6.1.1 Librerías, paquetes y versiones necesarias para la ejecución de la aplicación.**

Todos los requisitos en cuanto a librerías, paquetes y versiones necesarias para ejecutar de forma correcta tanto el back-end, como la base de datos, serán satisfactoriamente cumplidos al ejecutar el script de instalación que viene preparado con este proyecto. Por lo tanto solo será

necesaria la ejecución de dicho script para instalar y poder poner en marcha el sistema.

## 6.2 Justificación de los parámetros de ejecución de la red neuronal

Hemos de tener en cuenta que durante las pruebas de la red neuronal hemos testado dos redes de tipo LSTM diferentes. Una formada por caracteres y otra formada por palabras.

Veamos antes cuales serán los parámetros que intervendrán en nuestra red:

### - Parámetros utilizados en ambas redes neuronales:

#### 1- Epochs

Las epochs hacen referencia al número de veces que se va a ejecutar la red neuronal, antes de guardar un resultado de entrenamiento definitivo. Un mayor número de epochs implica generalmente una mejora en la pérdida de información de los parámetros de aprendizaje.

Sin embargo aunque esta pérdida decae inicialmente de forma acusada, tiende a estabilizarse cuando se han ejecutado ya un número considerable de ejecuciones. Por lo que lo óptimo, sería encontrar el rango de epochs donde la red neuronal ya no progresa más. Esto se debe a que el inconveniente de aumentar el número de epochs, es que aumentaremos proporcionalmente el tiempo de ejecución de la red. (Antonio Gulli y Sujit Pal, 2017, p. 98)

#### 2- Batch size

El batch size o tamaño del lote hace referencia al número de ejemplos que la red neuronal usa durante el entrenamiento. Este número suele fijarse entre 10 y 1000.

Al igual que en el caso de las epochs, un mayor número del tamaño del lote se traducirá directamente en un mayor tiempo de ejecución por epoch. Si los ejemplos escogidos por la red neuronal son adecuados conforme al problema que queremos solventar, la red producirá mejores resultados. Por el contrario si los ejemplos no son adecuados, es decir son heterogéneos, un mayor tamaño de batch size dará en peores resultados. (Antonio Gulli y Sujit Pal, 2017, p. 105)



### 3- Max len

Este parámetro hace referencia al número máximo de palabras seleccionadas por cada secuencia o ejemplo.

En este caso un alto número de max len no necesariamente proporcionará mejores resultados. Esto se debe a que se puede coger una secuencia redundante la cual limite la diversidad de palabras contenidas en ella. Así mismo, un número bajo de max len podría no recoger palabras suficientes, dando como consecuencia una entropía baja en el sistema, y por lo tanto unos malos resultados.

### 4- Número de neuronas

El número de neuronas está directamente relacionado con la capacidad de generar mejores resultados por parte del sistema. Por lo general un alto número de neuronas será beneficioso para el funcionamiento de la red.

Sin embargo los números demasiado elevados de las mismas pueden comprometer desde la eficiencia del sistema, hasta la estabilidad de la ejecución. Al haber un alto número de neuronas se eleva la probabilidad de pérdida de información por parte del sistema, la cual si es elevada podría ser crítica, impidiendo un resultado adecuado.

Por otro lado un mayor número de neuronas requiere de un mayor número de recursos computacionales. Esto puede desencadenar en un fallo de ejecución del sistema si ese número de recursos se acerca al límite disponible en la máquina. (Antonio Gulli y Sujit Pal, 2017, p. 102)

### 5- Return sequences

Cuando en Keras activamos esta propiedad estamos permitiendo que el valor oculto de salida sea devuelto por cada entrada de tiempo, es decir, en cada ejecución del sistema.

## 6- Dropout

Por lo general si no existe valor de Dropout, nuestra red neuronal conectará los valores de entrada de los diversos nodos de la red. El valor Dropout, es un valor de desconexión probabilística medido de 0 a 1 en Keras, el cual nos indicará la probabilidad de desconexión de dichas entradas durante la ejecución.

Una desconexión implicará que a la entrada de cada bloque LSTM será excluida de la activación del nodo y de la actualización de peso.

Es por esto por lo que introducir un valor bajo de Dropout podrá aumentar ligeramente la variabilidad generacional de información, permitiendo no perder todos los valores de todos los bloques de la generación anterior.

Del mismo modo, un valor alto de Dropout, podría suponer la práctica totalidad de la pérdida de la memoria de LSTM y por lo tanto crear un aprendizaje mínimo de la red.

## 7- Recurrent Dropout

El recurrent Dropout funciona de forma similar al Dropout. En este caso dicho elemento afecta al estado recurrente de la red, alterándolo de igual forma que el Dropout con los valores de entrada de los nodos.

En keras este elemento también se encuentra limitado a un rango entre 0 y 1.

## 8- Dense

El parámetro Dense hace referencia a la conexión de cada nodo de la neurona con los nodos de la siguiente capa oculta. Según esta definición una capa completamente conectada sería una capa densa. (Antonio Gulli y Sujit Pal, 2017, p. 61)

Lo habitual es encontrar capas de densidad media. Por ejemplo: Si nuestra red contiene 128 neuronas o nodos, la densidad será de 64.

## 9- Activation

Los parámetros activation o funciones de activación, son sumas ponderadas de todas las entradas de la capa anterior, las cuales generan un resultado no lineal que se pasa a la siguiente capa.

En Keras dispondremos de numerosas funciones de activación, siendo las más destacadas las activaciones Relu, Sigmoid y Tanh.

## 10- Optimizer

El optimizador hace referencia al tipo de implementación de descenso de gradiente que se quiere aplicar sobre la red neuronal.

El descenso de gradiente hace referencia a la minimización de la pérdida mediante a cálculos realizados con respecto a los parámetros del modelo, los cuales estarán condicionados al entrenamiento. Este algoritmo ajustará los parámetros iterativamente con el fin de dar con la mejor combinación de pesos y sesgos que minimizen los valores de pérdida de la red.

Keras proporciona varios optimizadores dando resultados diferentes en función del que apliquemos.

## 11- Loss

Se entiende el atributo Loss como aquel encargado en realizar el valor absoluto de la diferencia entre los valores que está prediciendo un modelo y los valores reales de las etiquetas. Así mismo se ha de aclarar que se entiende por etiquetas a una parte concreta del resultado obtenido.

Es decir, el valor loss o pérdida hace referencia a lo malo que resulta en modelo obtenido en una epoch concreta en base a una determinada medición.

En Keras podemos encontrar numerosas mediciones de la pérdida como la pérdida por error

cuadrático o la pérdida logística.

Una vez definidos los diversos parámetros que se encontrarán entre las dos redes neuronales, veamos cuales se definen en cada una y los valores asociados a ellos.

### **-Bidirectional networks vs SVM networks**

No se puede cerrar el apartado de las características de las redes neuronales utilizadas en este proyecto sin hablar de que dichas redes son de tipo bidireccional. ¿Que significa esto y porqué se han usado?

Tradicionalmente en aprendizaje computacional se usan muchas veces las denominadas máquinas de soporte vectorial. Estos algoritmos proporcionan buenos resultados, sin embargo, requieren del uso de numerosos y pesados elementos de procesamiento lingüístico, como el uso de lematizadores, etiquetadores o identificadores de elementos sintácticos.

En contraposición las redes bidireccionales solo necesitan del uso de vectores embebidos y pre-entrenados como el word\_to\_vec propuesto en este proyecto. Gracias a estos vectores, las redes neuronales bidireccionales son capaces de aportar resultados similares a las máquinas de soporte vectorial en un menor tiempo de cómputo.

### **-Red neuronal de caracteres**

Durante la implementación se realizó una serie de testeo inicial con una red neuronal de caracteres. El objetivo de probar con una red neuronal de este tipo, era el de comprobar si la predicción de caracteres era lo suficientemente buena como para generar palabras con el formato y variedad deseadas para este proyecto.

El testeo pronto reveló que aunque no producía resultados malos en cuanto a generación de palabras. La relación semántica entre ellas era inexistente debido a que cada carácter se predecía en relación al anterior. Esto generaba una falta de concordancia lógica entre palabras la cual se determinó que era imposible de subsanar por medio de la reestructuración de la red neuronal o por la variación de sus parámetros de configuración.

Veamos un ejemplo de salida obtenido con la red neuronal de caracteres en el que se mostrarán los detalles expuestos.

Los valores utilizados en la red neuronal de caracteres serán los siguientes:

- 1- Número de capas ocultas: 128
- 2- Batch size: 128
- 3- Max len: 40
- 4- Epochs: 200
- 5- Dense: Longitud de los caracteres (Dinámica)
- 6- Activación: Softmax

Ejemplo:

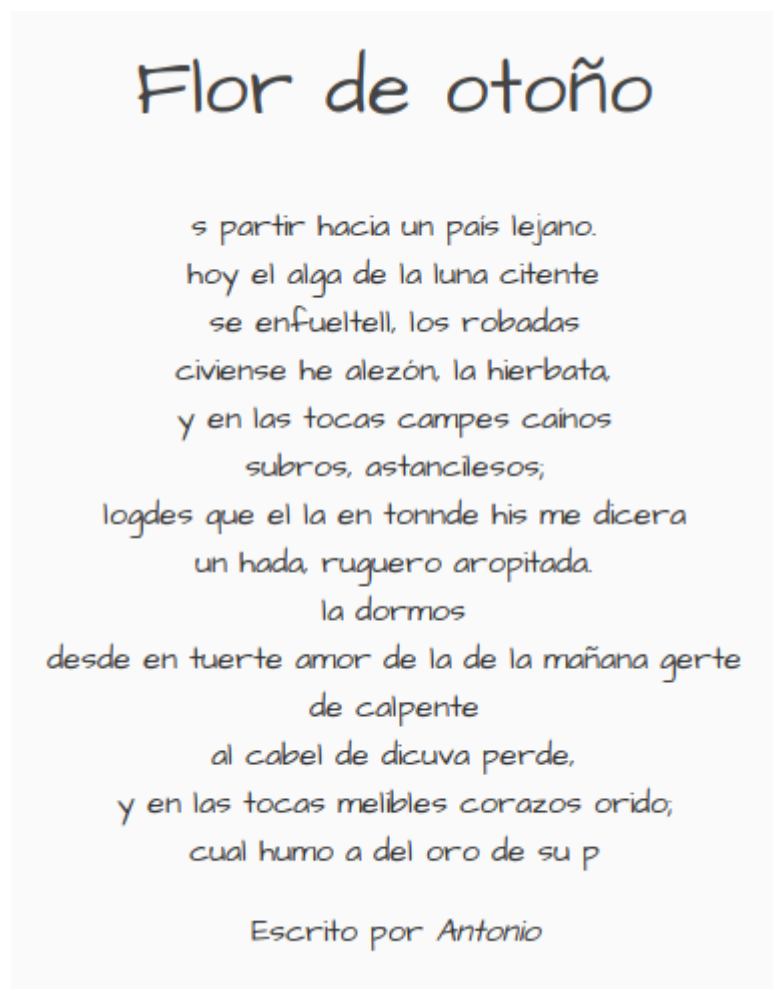


Imagen 6.1 Ejemplo de la red neuronal de caracteres. Elaboración propia.

En base a estos resultados se concluyó que lo mejor para la aplicación sería la implementación de una red neuronal de palabras. La cual fuese capaz de predecir palabras y no caracteres, intentando así mantener una mejor concordancia entre las mismas.

### **- Red neuronal de palabras**

La red neuronal de palabras será testada en sus parámetros principales, a fin de encontrar mejoras significativas en los resultados obtenidos de ella.

La variación de los parámetros se ha realizado variando cada uno de los parámetros que hemos considerado más importantes por separado.

### **-Valores de los parámetros de nuestra red neuronal de palabras**

Una vez definidos los distintos parámetros presentes en la red neuronal veamos a ver uno por uno, cuales se han decidido tocar y cuales no, además de una justificación de dicha decisión. Para ello, se generarán tres poemas de cada modelo diferente y se compararán entre si, dando importancia especialmente a la cohesión textual y a la fluidez en la lectura.

#### **1- Epochs**

El número inicial de epochs que se ha utilizado en la red neuronal es de 100. Posteriormente se ha incrementado dicho número a 1000. Gracias a este incremento se ha experimentado una ligera mejora de los resultados.

Si bien la variación en el número de epochs ha sido un factor que ha condicionado el resultado de las ejecuciones. Se ha de hacer mención a otro valor que ha sido variado también de forma experimental juntos al número de epochs, y al que no se ha nombrado hasta ahora, la diversidad.

El valor de diversidad hace referencia al grado de cercanía en la selección de términos a usar durante la generación del texto. Es decir, valores bajos de diversidad se utilizarán palabras que

habitualmente se encontrarán mas cercanas entre si en probabilidad de aparición, mientras que los valores de diversidad altos usarán palabras mas lejanas entre si.

En este caso se han establecido diversas ejecuciones bajo los siguientes parámetros de diversidad y número de epochs:

Primera ejecución: número de epochs: 100, diversidad: 0.2, 0.5, 1.0 y 1.2

Segunda ejecución: número de epochs: 100, diversidad: 0.5, 0.8, 1.0 y 1.2

Tercera ejecución: número de epochs: 100, diversidad: 0.8, 1.0 y 1.2

Cuarta ejecución: número de epochs: 100, diversidad: 1.0 y 1.2

Quinta ejecución: número de epochs: 1000, diversidad: 0.2, 0.5, 1.0 y 1.2

Sexta ejecución: número de epochs: 1000, diversidad: 0.5, 0.8, 1.0 y 1.2

Septima ejecución: número de epochs: 1000, diversidad: 0.8, 1.0 y 1.2

Octava ejecución: número de epochs: 1000, diversidad: 1.0 y 1.2

Tras esta serie de ejecuciones se ha comprobado que la sexta ejecución realizada a 1000 epochs y con diversidad de 0.5, 0.8, 1.0 y 1.2, es la que muestra los mejores resultados. Esto es así ya que al tener un mayor número de epochs, una diversidad variable entre un rango bajo y un rango alto de la misma, escoge mejores palabras para concatenar entre sí.

Es por esto que se han elegido los parámetros de la sexta ejecución para en base a ella, realizar y testear el resto de cambios aplicables a la web.

## 2- Batch size

El batch size ha sido testeado inicialmente con un valor de 128. Sin embargo con el fin de realizar diversas pruebas con otros valores de batch size, se ha decidido testear valores de 64 y 256 de batch size.

Tras comparar los resultados obtenidos, se ha observado que no se produce ningún tipo de mejora en cuanto a la fluidez lectora de los poemas generados con batch size 64. Mientras que sí se han experimentado cambios positivos en los poemas generados con batch size 256. Por lo que se

puede determinar que experimentalmente los resultados obtenidos con batch size 256 son mejores que con batch size 64 y 128. Por otra parte este resultado era esperado, ya que al igual que el aumento en el número de epochs, al aumentar el tamaño del lote, incrementamos la información procesada por epoch. Así mismo el tiempo de procesamiento con 256 es considerablemente mayor que con 128.

Es por ello que para el Batch size se optará por usar 256 en vez de 128.

### 3- Max len

El valor de max len original ha sido de 20. Es decir, la secuencia de ejemplo seleccionada ha sido siempre de 20 palabras.

Este valor no se ha tenido en cuenta como demasiado importante ya que aun cuando un ajuste del mismo a la baja permitiría ahorrar en número de palabras usadas por la red. Y un ajuste al alza permitiría aumentar ligeramente la variabilidad de las cadenas de secuencia de palabras. El estudio de este valor se ha considerado demasiado costoso en relación al tiempo de ejecución y posterior análisis que se invertiría en dar con el número mas óptimo.

Es por este motivo por el cual el valor 20 para Max len permanece inalterado en todas las ejecuciones.

### 4- Número de neuronas

El número de neuronas utilizado inicialmente (128) es el número de neuronas por capa que se ha comprobado como efectivo para la mayoría de desarrolladores de redes neuronales de generación de textos en internet. Es por ello por lo que se comenzó con dicho número.

Sin embargo, se ha experimentado también con valores de 64 y 256 en el número de neuronas, a fin de comprobar el funcionamiento bajo estos parámetros. La premisa es que a menor número de neuronas, el rendimiento caerá. Mientras que a mayor número de ellas el rendimiento aumentará, tal y como sucede con epochs y batch size. La única incógnita es si el valor 256 será demasiado elevado como para general buenos resultados o caerá en la problemática de tener un



número de neuronas demasiado elevado.

Tras el testeo se comprueba que el uso de 64 neuronas por capa resulta ser perjudicial frente al uso de 128, provocando una menor cohesión en el texto. Por su parte el uso de 256 neuronas resulta en una complicación no contemplada inicialmente. Y es que tal cantidad de neuronas resultan en un volumen de parámetros a mantener en memoria demasiado elevado para el sistema. Haciendo que la ejecución se aborte en mitad del proceso.

Es por ello que para este parámetro se optará por mantener el valor inicial de 128.

#### 5- Return sequences

El número de return sequences establecido en 0.1, es un parámetro cuya variación se ha considerado poco relevante. Puesto que al igual que en el caso de Max len, este número se ha de estudiar considerablemente a fondo invirtiendo un gran esfuerzo computacional y temporal en ello.

Adicionalmente el valor actual de 0.1 es un valor habitualmente usado por los desarrolladores, por lo que a falta de un testeo más intenso se considerará como un valor sino óptimo, al menos adecuado.

#### 6- Dropout

Por contra el valor de Dropout establecido inicialmente en 0.1, si adquiere relevancia al tener una capacidad de diversificar la memoria que se transmite a la siguiente capa como entrada.

En este caso se ha optado por tomar valores bajos ya que un Dropout alto podría hacer perder la memoria de tal forma que la red LSTM careciese de sentido.

El testeo se realizará por tanto con los valores 0.1 y 0.2.

Los resultados obtenidos con el valor de 0.2 en Dropout no son satisfactorios en relación al Dropout de 0.1. Por lo que se optará por seguir utilizando 0.1 de valor Dropout.

Cabe destacar que en los artículos y referencias tomadas de Internet como base para modificar el Dropout, el rango de Dropout satisfactorio oscila entre el 0.1 y el 0.3, siendo este último el valor que mejores resultados suele dar. Es por ello que este parámetro será tenido en cuenta para futuros y más profundos estudios.

Así mismo se puede reseñar que el Dropout se utiliza doblemente en la propia red neuronal siendo utilizado la segunda vez con un valor de 0.5.

#### 7- Recurrent Dropout

Al igual que en el caso del return sequences el parámetro recurrent Dropout, se ha tomado poco relevante por su alto costo para determinar su valor óptimo. Y por ser usado habitualmente con el valor actual (0.1).

#### 8- Dense

El valor inicial utilizado en la densidad es de 64. Esto supone que en el caso de nuestra red de 128 neuronas, la densidad es del 50%. Dicha densidad es adecuada en tanto en cuanto no queremos hacer que la transmisión de conocimiento entre neuronas se establezca en el 100%. El motivo será el de conservar una mayor diversidad, puesto que una densidad del 100% provocaría una unificación de soluciones de forma más rápida sin dejar que el sistema explore otros valores.

Por lo tanto la densidad elegida será de 64.

#### 9- Activation

Partiremos de la función de activación inicial ReLu, utilizada hasta ahora y realizaremos comparaciones con otra serie de funciones de activación exitosas ya implementadas en Keras, como son las funciones Sigmoid y Tanh.

Hemos de recordar que al finalizar el proceso se añade una activación softmax. Esta activación no será testeada ya que el peso de las activaciones recae fundamentalmente en el lugar de la activación actual o ReLu. Adicionalmente se suele utilizar la función softmax en el proceso final

de activación de redes neuronales.

Al ser funciones ya definidas en Keras procederemos a ver sus diferencias antes de evaluar resultados.

#### Relu (Rectified Linear Unit):

La unidad lineal rectificadora Relu tiene una gran ventaja sobre Sigmoid y Tanh ya que nunca se satura con un valor alto de  $x$ . La principal desventaja es que su media no es cero debido a lo cual la función se convierte en cero, haciendo el aprendizaje demasiado lento (Machine Intelligence, 2017).

Esta función genera bastante buenos resultados debido a que no considera los valores negativos al catalogarlo internamente como cero.

#### Sigmoid

La función Sigmoid toma cualquier número real de rango y devuelve el valor de salida que cae en el rango de 0 a 1. Basado en la convención podemos esperar el valor de salida en el rango de -1 a 1.

La principal desventaja de Sigmoid es que deja de aprender por el gran valor de  $x$  o en otras palabras la función se satura por el gran valor de  $x$  (Machine Intelligence, 2017).

#### Tanh (Tangente Hiperbólica)

La función Tanh, también conocida como función tangente hiperbólica, es un re-escalamiento del sigmoid logístico, de modo que sus salidas van de -1 a 1.

El rango de salida  $(-1, +1)$  tiende a ser más conveniente para las redes neuronales. Estas funciones son propensas a alcanzar un punto desde el cual el gradiente de las funciones no cambia, detiene el aprendizaje o se satura fácilmente por un gran valor de  $x$  (Machine Intelligence, 2017).

En base a estas definiciones, se espera que la mejor función de activación resulte ser Tanh o

Relu debido a sus condiciones de baja o nula saturación.

Una vez realizado el testeo, se ha comprobado que los resultados generados por las funciones de activación Sigmoid y Relu, son similares. Sin embargo los resultados obtenidos por la función Tanh dan unos resultados de mayor fluidez.

En base a esto, se escogerá la función de activación Tanh para obtener nuevos modelos.

No obstante se ha comprobado experimentalmente durante el entrenamiento de diferentes corpus, que en ciertos casos la red neuronal con activación Tanh se satura y aborta la ejecución. En todos estos casos se ha procedido a hacer uso de la función de activación ReLu, la cual no ha dado problemas, generando los modelos satisfactoriamente.

#### 10- Optimizer

El optimizador utilizado inicialmente ha sido el optimizador Adam. Al igual que en el caso de las funciones de activación. Keras también proporciona diversos optimizadores en base al enfoque de la red y de su uso.

En este caso Adam es un optimizador que ha probado dar muy buenos resultados para la generación de textos a través de redes neuronales. Por lo tanto y pese a haber numerosos optimizadores, se ha optado por seguir usando Adam.

#### 11- Loss

Por último se encuentra el valor de la pérdida. Este valor no es un valor a tener en cuenta, ya que las ejecuciones de la red neuronal, han sido establecidas bajo un criterio de parada por número de epochs y no por valor de pérdida.

Es decir, durante las ejecuciones se puede establecer otro criterio de parada que no sea un número concreto de ejecuciones como ha sido el caso hasta ahora. Sino que en base a la pérdida acumulada tras cada ejecución o epoch. Si esta baja de un cierto umbral se considera que el modelo es ya lo suficientemente bueno como para detener la ejecución y guardar el modelo.

Aunque no será el caso de uso del valor de pérdida, este valor nos puede dar una idea de cuan buenos son nuestros modelos obtenidos en base a al pérdida de información.

Keras nos proporciona diversos métodos de evaluación de la pérdida, en nuestro caso utilizaremos la pérdida categórica de entropía cruzada, o categorical crossentropy. Este tipo de evaluación de la pérdida arroja al cabo de 1000 epochs un valor que se mantiene constante alrededor de 0.12.

### **-Parámetros finales**

Por tanto en base a los estudios realizados en cuanto al valor más adecuado de los parámetros podemos resumir que los valores más adecuados encontrados para la generación de modelos serán los siguientes:

- 1- Epochs → 1000 y diversidad → 0.5, 0.8, 1.0, 1.2
- 2- Batch size → 256
- 3- Max len → 20
- 4- Número de neuronas → 128
- 5- Return sequences → 0.1
- 6- Dropout → 0.1
- 7- Recurrent Dropout → 0.1
- 8- Dense → 64
- 9- Activation → Tanh (en caso de saturación se usará ReLu)
- 10- Optimizer → Adam
- 11- Loss → Categorical Crossentropy

### **6.3 Ejemplos de salida**

Vamos a ver tres ejemplos de salida en base al poema generado.



Imagen 6.2. Ejemplo de la red neuronal de palabras. Elaboración propia.

Como se puede observar se ha conseguido aportar una mayor cohesión y claridad en el texto, lo cual permite leer con mayor fluidez el mismo. Hay que destacar también el hecho de que los poemas carecen de ninguna forma métrica concreta. Esto se puede entender como una poesía de verso libre donde las rimas tampoco son frecuentes entre unos versos y otros.

#### 6.4 Plan de pruebas de la web y de la base de datos

En este punto de la memoria se van a realizar una serie de aclaraciones sobre el dispositivo utilizado para realizar todo el proyecto. Además se hará una validación de los diferentes casos de uso establecidos en las fases iniciales del proyecto a fin de determinar si se ha cumplido con todos ellos y por lo tanto con los objetivos de la aplicación.

**6.4.1 Condiciones del equipo de pruebas**

El equipo de pruebas ha dispuesto de los siguientes elementos hardware y software:

- 1- Sistema operativo “Ubuntu 18.04 Bionic Beaver”
- 2- CPU Intel Core I-7 cuarta generación
- 3- Ram 16 Gb ddr3 última generación
- 4- Python versión 3.6.8
- 5- Mongo DB versión 3.6.3
- 6- Pymongo versión 3.8.0
- 7- Navegador Mozilla Firefox versión 65.0.2 Quantum

**6.4.2 Pruebas para cada caso de uso**

A lo largo de la validación se han establecido dos pruebas de la web. La primera en la cual aún no se había añadido la base de datos a la web y la segunda con la base de datos ya incluida en la web. Veamos la validación de los casos de uso mediante la siguiente tabla a lo largo de las dos pruebas.

Caso de uso	Prueba 1	Prueba 2
CDU-01 - Navegar por las vistas de la interfaz	✓	✓
CDU-02 - Seleccionar el país del poeta	✓	✓
CDU-03 - Seleccionar poeta	✓	✓
CDU-04 - Escribir título, y semilla	✓	✓
CDU-05 -	✓	✓

Generar poema		
CDU-06		
-	✓	✓
Imprimir poema		
CDU-07		
-		✓
Buscar por campos clave		
CDU-08		
-		✓
Consultar listado de poemas		
CDU-09		
-		✓
Comparar redes neuronales		

Tabla 6.1 Validación de los casos de uso. Elaboración propia.

Como podemos observar tras la segunda prueba, podemos asegurar que cada caso de uso se ha validado en la aplicación. Este hecho nos permitirá ratificar nuestra aplicación.

### 6.5 Experiencia de usuario

A continuación se van a detallar los aspectos fundamentales de la experiencia de usuario recogida a lo largo de la evolución del proyecto.

Se han evaluado las dos versiones de la web, ya que la primera versión operativa de la misma fue expuesta abiertamente al público la noche de los investigadores del 28 de Septiembre de 2018. En ella se mostraban los resultados que se podían obtener con un modelo del autor Antonio Machado entrenado por una red neuronal de caracteres.

En la primera versión de la web se han evaluado tres características fundamentales:

- 1- El manejo por la interfaz y su facilidad de aprendizaje.
- 2- La calidad de los resultados obtenidos en relación a lo esperado.
- 3- El interés despertado por el proyecto y la idea asociada al mismo. Es decir, la generación automática de poesía.



Durante el testeo de dicha versión de la web al que tuvieron acceso varias decenas de personas, se detectaron y registraron las siguientes reacciones y comportamientos, las cuales se detallarán en función de los apartados definidos anteriormente.

1- El manejo de la interfaz resulta simple e intuitivo. No hay que poseer ningún conocimiento específico para el uso de la aplicación. El diseño es simple y amable de cara al usuario, proporcionando una reacción favorable en este aspecto.

2- Los resultados generados por la web causan diversas reacciones de incompreensión, extrañeza, risa o curiosidad. Los usuarios más mayores reaccionan de forma decepcionada ante las poesías generadas, mientras los usuarios más jóvenes reaccionan con curiosidad y risas al leer los textos.

Un aspecto a resaltar es que los usuarios más jóvenes y los adultos sin hijos valoran de forma más positiva y agradable el trabajo llevado a cabo en la aplicación sin importar su grado de satisfacción con el resultado obtenido. Mientras que los adultos con hijos se han mostrado mucho mas exigentes.

3- En general, todo el público que se acercó al stand donde se encontraba expuesta la aplicación, la testeó y se mostró ampliamente interesado con la idea y el proyecto desarrollado.

Debido a falta de tiempo para subir la aplicación final a un servidor, no se ha podido realizar una encuesta a un número razonable de personas sobre el funcionamiento de la segunda versión operativa. Si se ha podido realizar una encuesta a ocho personas sobre diferentes aspectos del uso de la web tanto desde el ordenador personal en el que se ha desarrollado, como desde dos ordenadores más en los que esta web se ha instalado y ejecutado. De esta forma se ha podido validar también el uso correcto de las instrucciones de instalación y uso de la aplicación.

En esta ocasión se han evaluado los mismos aspectos de la web que en la versión anterior con la característica adicional de realizar una valoración de las características adicionales incorporadas, como han sido el uso de una base de datos, las consultas a dicha base de datos con y sin parámetros, la traducción al inglés y la posibilidad de testear diferentes redes neuronales.

- 1- El manejo por la interfaz y su facilidad de aprendizaje.
- 2- La calidad de los resultados obtenidos en relación a lo esperado.
- 3- El interés despertado por el proyecto y la idea asociada al mismo. Es decir, la generación automática de poesía.
- 4- Opinión sobre las funcionalidades extra que aporta la aplicación.

Veamos la evaluación de dichos puntos en detalle:

1- El manejo ha resultado nuevamente muy satisfactorio. El diseño ha sido elegante, completo y detallado. No presenta una gran sobrecarga de contenido y es muy intuitivo. La reacción hacia este aspecto es totalmente favorable.

2- La calidad de los resultados ha sido aceptable. Se puede leer con cierta soltura el texto generado. Aunque si bien la generación del texto ha sido un punto favorable, la calidad del texto como poesía es algo más baja. Se aprecia de forma difusa la estructura poética debido a la larga longitud de los versos y las rimas de verso libre mezcladas con rimas de última sílaba. En general, el resultado aunque bueno, no es todo lo positivo que se espera de una poesía.

3- El interés despertado por el proyecto resulta si cabe más alto que el percibido en la anterior exposición al público. Esto se debe principalmente a la adición de características a la web y la mejor generación de textos que si bien distan de ser las poesías originales, son mucho más amables para el usuario que las pertenecientes a la versión anterior.

4- Las funcionalidades extra han resultado ser muy positivas. Ha gustado especialmente el poder elegir entre diferentes autores, el comparar redes neuronales y el poder guardar las poesías generadas. Sin embargo algunos usuarios han realizado algún tipo de sugerencia o han echado en falta algunos elementos adicionales de la web, estos han sido la posibilidad de hacer de la web un foro comunicativo entre usuarios con un perfil registrado y la posibilidad de compartir mediante correo o redes sociales el poema generado; sin necesidad de tener que descargarlo para compartirlo.

Además como apartado adicional se ha mencionado que si bien la web es intuitiva para quienes están familiarizados con la poesía y con el mundo digital y de redes neuronales, para aquellos que no lo están, el manejo de dicha web es algo mas costoso ya que no son capaces de entender los

conceptos asociados a la misma, por lo que una buena sugerencia de mejora sería hacerla web más accesible y entendible a todos aquellos usuarios que no tengan conocimientos informáticos o literarios.

### 6.6 Gestión de errores

A continuación se muestra una tabla en la que se listan los errores encontrado de uso durante la ejecución y se muestra en que prueba se corrigen dichos errores.

Error	Prueba 1	Prueba 2
El texto se muestra mal formateado	✓	✓
Al elegir diferentes autores la aplicación falla		✓
Si cierras la aplicación mientras se genera el poema, este se pierde		✓

Tabla 6.2 Gestión de errores. Elaboración propia.

### 6.7 Seguridad

Aunque el funcionamiento de la aplicación es el correcto. No hay que pasar por alto que en casos de tener páginas web con acceso a bases de datos siempre existe el riesgo de sufrir un ataque que comprometa la seguridad de la web.

En nuestro caso contamos con una serie de ventajas:

1- La información que se guarda es de carácter anónimo y no es sensible.

Este hecho es bastante relevante ya que al no guardar información sensible de ninguna persona. La intencionalidad asociada a un robo de datos es prácticamente nula.

2- La limitación en el número de consultas:

Al poseer la base de datos limitada a un número de consultas de treinta, evitamos una sobrecarga excesiva en caso de establecer una serie de peticiones consecutivas al servidor. No solo desde un mismo terminal sino desde numerosos terminales al mismo tiempo, limitando así los riesgos de un ataque DDoS.

No obstante tanto los ataques a nivel de consulta masiva aún podrían producirse. Otras medidas a realizar a la hora de implementar este sistema web en internet, serían escalar el número de servidores que gestionasen la aplicación. Otra medida, sería la de realizar desvíos de todas las conexiones a la capa de red de tal forma que se pudiese filtrar el numero de peticiones realizadas por fracción de segundo desde un mismo terminal y cortar la conexión de forma temporal en caso de detectar un número de peticiones masivo. Cabe recordar que los ataques DdoS, son principalmente inundaciones de peticiones GET y POST de tipo HTTP pertenecientes a la capa de red, es por ello importante filtrar en esta capa.

-Otro tipo de ataques que podrían realizar sobre la aplicación.

Los ataques por lo general suelen estar relacionados con las bases de datos y a su vez con las operaciones básicas CRUD que estas realizan.

En el caso de esta aplicación además de las operaciones de consulta, se realizan operaciones de inserción en la base de datos.

Esta inserción se realiza de forma indirecta, es decir, a través del paso de parámetros de un formulario al controlador del servidor mediante POST. Una vez allí, el controlador envía la petición para generar el nuevo poema y posteriormente lo manda insertar en la base de datos. Esto evita parcialmente la problemática de realizar inserciones masivas de forma manual.

Sin embargo esta tarea se podría realizar de forma automática mediante el uso de un bot o un script de ejecución que rellenas el formulario con información aleatoria, y mandase la petición. Para evitar este suceso, además de realizar la desviación de peticiones por la capa de red, se podría implementar un sistema de detección de bots o bien utilizar uno ya existente como captcha.

Otro posible ataque a tener en cuenta es la inyección de comandos NoSQL, realizados a través de los campos del formulario de generación de poemas. Mediante un ataque de este tipo, se podrían insertar, eliminar o alterar los datos residentes en la base de datos. Para evitar este suceso, una opción es filtrar los campos del formulario en el apartado del modelo, de forma que los posibles comandos maliciosos, no se ejecuten dentro de la base de datos.

En resumen el apartado de la seguridad aunque de menor relevancia en este proyecto, nunca se debe pasar por alto debido al gran riesgo que supone mantener una base de datos en una aplicación web abierta a cualquier usuario de internet. Será siempre de gran importancia supervisar su correcto funcionamiento y la integridad y seguridad de los datos que en ella se encuentren almacenados en todo momento.

## 7. Manual de usuario

### 7.1 Instalación y puesta en marcha

Para la instalación de la aplicación web ejecutaremos un script, el cual garantice que tendremos disponibles todos los recursos de lenguajes y tecnologías necesarios instalados en nuestro ordenador para hacer que este actúe como máquina servidora.

Para ello simplemente daremos permisos de ejecución al archivo `installer`. Y a continuación procederemos a dar algo de tiempo para que se instalen todos aquellos paquetes y librerías necesarias y no disponibles en nuestra máquina. Esto, puede tardar varios minutos en caso de no poseer ninguno de los requerimientos tecnológicos que la aplicación demanda. También se puede abrir una consola de ejecución y ejecutar uno a uno las distintas dependencias que vienen en el script. Para ejecutar el script, solo habrá que escribir en una terminal, y desde el directorio donde se encuentre el script, el siguiente comando: `$bash installer.txt`

Es importante resaltar dos aspectos adicionales. Una vez instalados los recursos necesarios, no podremos ejecutar directamente la aplicación.

Para ello habrá que abrir una consola de ejecución y acceder a mongoDB, escribiendo en consola la orden `"mongo"`. A continuación se deberá crear la base de datos del mismo nombre que la que se usará desde la web, en este caso llamada `"poemdb"`, para ello se deberá escribir en consola la orden `"use poemdb"`. Por último se insertará un único elemento a fin de que la base de datos de mongo no destruya la nueva base de datos por encontrarse vacía. Para realizar esta tarea, escribiremos en consola `"db.poemdb.insert({"hola": "adios"})"`. Este ejemplo será borrado si se quiere tras realizar la importación de la base de datos o tras generar un poema con la aplicación.

Ahora si podremos realizar inserciones en la base de datos. Si se quiere importar la antigua base de datos, se deberá hacer lo siguiente.

En la aplicación en el archivo `routes.py`, descomentaremos la función `"processDBImport()"` y recargaremos la página web. Tras esto ya habremos importado la base de datos, por lo que podremos volver a comentar la función.

Si se desea realizar una exportación de los datos, en el mismo archivo de `routes.py`, bastará con descomentar la `processDBExport()` y recargar la página. Posteriormente es recomendable volver a comentar dicha función.

Finalmente si se desea eliminar la primera inserción bastará con escribir en consola de mongo la siguiente orden `db.poemdb.remove({"hola" : "adios"})`.

Una vez completada la instalación de las diferentes tecnologías y recursos necesarios para lanzar la aplicación, nos moveremos desde terminal hasta la carpeta donde resida el proyecto y ejecutaremos el script que hará que la aplicación comience a funcionar.

## **7.2 Navegar por la web**

### **7.2.1 Introducción**

En este apartado vamos a desarrollar el manual de uso de la aplicación web, centrándonos en todos y cada uno de sus diferentes aspectos, usos y funciones desde la interfaz de usuario o navegador web. Este manual esta destinado al usuario final de la aplicación, de tal modo que en caso de necesitarlo, pueda aprender a manejar la web de forma fácil y sencilla simplemente siguiendo los pasos aquí detallados.

### **7.2.2 Terminología**

A continuación vamos a detallar y a definir la terminología usada en este proyecto. A fin de aclarar los significados dados a palabras de uso específico y poco usuales a los ciudadanos que no están familiarizados con las nuevas tecnologías y el uso de Internet.

#### **1- Registro**

Entenderemos por registro toda aquella entrada u objeto de la base de datos que haya sido insertado en ella y con el que podemos operar, es decir, trabajar con él.

## 2- Vista

Las vistas serán cada una de las diferentes pestañas o secciones de nuestra aplicación. A través de las cuales nos podremos mover libremente gracias a la barra superior de navegación. Además cada vista realizará una tarea única e independiente.

## 3- Campo

Los campos serán todos y cada uno de los apartados susceptibles de ser alterados por el usuario. En nuestro caso tendremos campos de texto en los cuales el usuario podrá escribir aquello que desee, y campos de selección simple, en los cuales de entre una lista de opciones podrán seleccionar una única opción.

## 4- Consulta

Las consultas harán referencia a las vistas o acciones que hagan una extracción de datos procedente de la base de datos, que luego serán expuestas en su propia vista.

## 5- Pdf

Formato de texto estándar para lectura y guardado de la información capaz de ser procesado por casi todos los dispositivos del mercado actual.



### 7.2.3 Secciones de la aplicación

En este apartado, vamos a ver cada una de las diferentes vistas y sus detalles particulares.

#### 1- Menú inicial

Acciones de la barra de navegación que nos devuelven siempre al menú de inicio.

Esta vista nos permitirá generar el poema que deseemos

Esta vista nos llevará al menú inicial en el idioma especificado



En este área se mostrará la información relativa a la web

Haciendo clic en esta acción iremos a la vista que nos muestre todos los poemas de la web

Esta acción nos redirige al testeo de redes neuronales

Mediante esta búsqueda podremos encontrar poemas por las palabras deseadas

Imagen 7.1. Menú inicial. Elaboración propia.

1- Tanto el icono de iPoetry como botón de home, nos llevan a la misma sección de la web, el menú inicial.

2- Para activar la búsqueda por campos clave, hemos de colocar el ratón sobre la misma, lo cual desplegará un campo de texto donde pondremos la palabra o palabras a buscar. Tras esto, haremos clic sobre el icono de búsqueda o presionaremos “Enter” en nuestro teclado.

Vista al completo del menú inicial de la web

## 2- Generar poemas

The screenshot shows the iPoetry website interface. At the top is a navigation bar with links: HOME, GENERAR POEMA, POEMAS PUBLICADOS, TESTEA LA RED, and ENGLISH. A search icon is on the right. The main heading is "¿Qué es iPoetry?" followed by a subtext: "iPoetry es una aplicación capaz de generar a través del uso de una inteligencia artificial un poema de características y estilo similares a las del autor seleccionado". Below this is a question: "¿Te gustaría servir de inspiración para generar poesía?". The form contains several elements: two dropdown menus labeled "Selecciona el país:" and "Selecciona el autor:" with "--País--" and "--Autor--" respectively; a section titled "Elige tu propio nombre de autor:" with a text input field "Nombre del autor"; another section titled "Elige el título del poema:" with a text input field "Título del poema"; and a section titled "Introduce una frase de semilla para el poema:" with a text input field "Introduce una frase (máximo 40 caracteres)". A blue button labeled "Generar Poema" is at the bottom of the form. Red arrows point from text labels to these elements: "Desplegable para seleccionar el país" to the first dropdown, "Desplegable para seleccionar el autor" to the second dropdown, "Campos de texto opcionales" to the three text input fields, and "Botón que activará la acción de generar el poema" to the "Generar Poema" button.

Imagen 7.2. Generar poemas. Elaboración propia.

1- Para el correcto funcionamiento de los desplegables, primero se ha de seleccionar el primero (país) y después se ha de seleccionar el segundo (autor).

2- Los campos opcionales no tienen por qué ser rellenados.

3- Para generar poema, hacer clic en generar poema o presionar "Enter".

Vista al completo del  
formulario para generar  
poemas

## 3- Imprimir poema



IPOETRY HOME GENERAR POEMA POEMAS PUBLICADOS TESTEA LA RED ENGLISH

### La espada del gato

de septiembre besa y se lleva algunos amarillos secos jugando entre el campo del monte lleva la luna un fuego si os no responden cuando el viento de la tierra el el terciopelo en la angustia bandurria al fondo de un surco

campo del monte moscos oia el romance pasan sobre la estrella de la mar fue en la tierra la vida de un cazador nidos y todo más verdadero del mundo mio castilla

campo del negro y el viento se abría el viento la estrella de la tierra el viento y brilla renacimiento el camino los torres

campo del monte lleva la revuelta corre un mago vuelve la pura como una piedra gacelas la tierra de soria y florecia tienen ojos de invierno la tierra de valencia cuando la arena hierba la blanca verdes

Generado por IPOetry  
Semilla dada por Jeanne

Poema escrito por IPOetry, una inteligencia artificial sencilla basada en redes neuronales que ha aprendido de leer: palabra a palabra, la obra del autor que has seleccionado ( Antonio Machado ).

Trabajo de fin de grado, mayo 2019 - Universidad de Jaén, Escuela Politécnica Superior de Jaén

**UJa** Universidad de Jaén

Descargar en formato PDF

Botón para descargar el poema en formato PDF

Imagen 7.3. Imprimir poema. Elaboración propia.

1- Para imprimir el poema, hacer clic en descargar en formato PDF o presionar “Enter”.

Vista al completo del poema generado y su vista de impresión

## 4- Consultar toda la base de datos

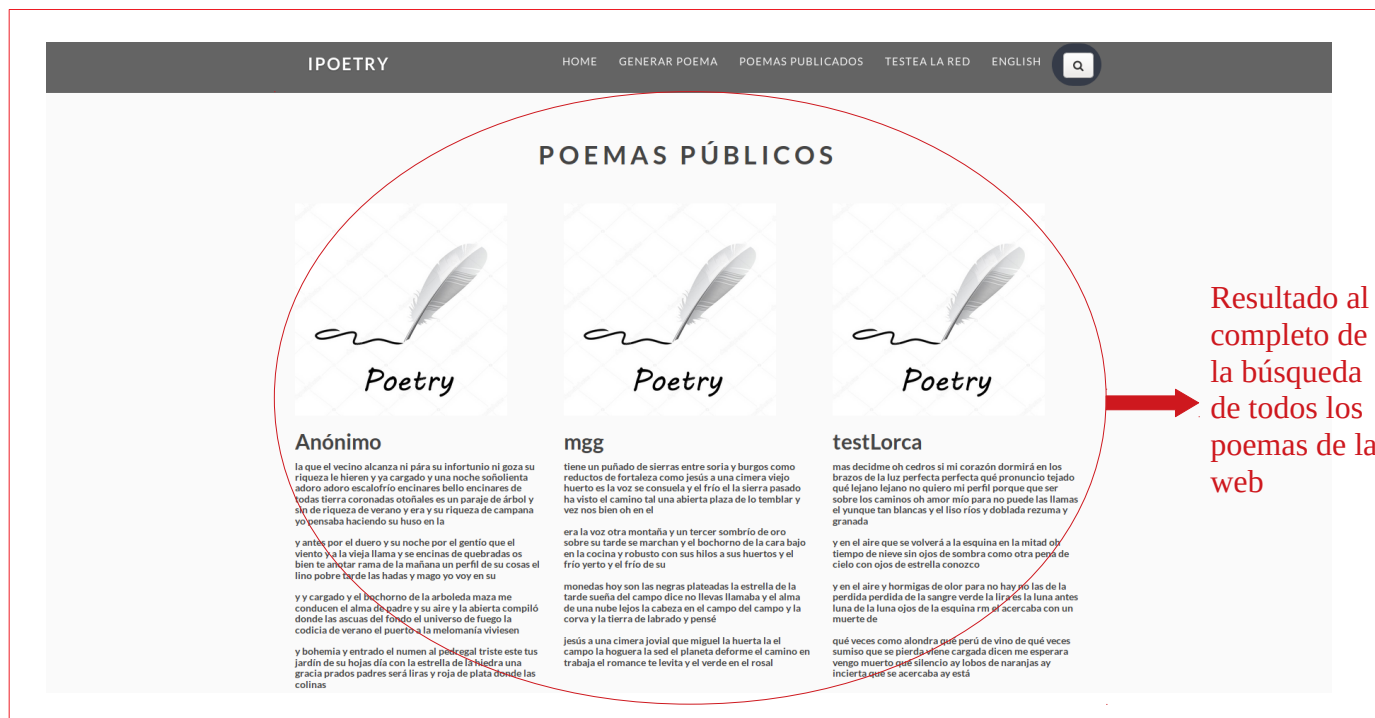


Imagen 7.4. Consultar la base de datos. Elaboración propia.

Vista al completo de la  
búsqueda general sobre la  
web

## 5- Realizar consultas específicas



Resultado de  
la búsqueda  
por palabras  
clave en la  
web

Imagen 7.5. Realizar consultas específicas. Elaboración propia.

Vista al completo de la  
búsqueda por palabras  
clave

## 6 - Testea las redes neuronales



Imagen 7.6. Testeando las redes neuronales. Elaboración propia.

Vista comparativa sobre  
diferentes redes  
neuronales

## 8. Conclusiones

### 8.1 Reflexiones finales

El proyecto que se ha desarrollado a lo largo de esta memoria de trabajo de fin de grado, ha sido uno de los mayores retos a nivel de despliegue de conocimientos y aprendizaje que he vivido a lo largo de este período universitario. Este trabajo realmente ha supuesto un gran esfuerzo, pues he tenido que superar numerosas complicaciones derivadas de aprender a gestionar tiempos, aprender a discernir entre lo posible y lo imposible, a aprender a mantener una visión global del proyecto, así como de su proceso de evolución a lo largo de su desarrollo.

Este proceso ha sido especialmente nuevo para mí, pues he tenido que tomar decisiones constantes, sabiendo que estas requerían tiempo y esfuerzo. Además esta toma de decisiones han implicado valorar de forma permanente la relevancia de los cambios que he debido ir realizando una y otra vez. Estos cambios se han planteado constantes con tal de facilitar la interacción con la interfaz, con tal de simplificar el aprendizaje del uso de la web, e incluso, con tal de introducir el máximo número de opciones adicionales a la web, sin descentrarnos en todo momento del objetivo principal: El estudio de generación de poesía en base a una red neuronal.

A lo largo del trascurso de este proyecto he tenido que adquirir todo tipo de destrezas relativas a la programación general, las cuales me han permitido descubrir un mundo nuevo de tecnologías y aplicaciones. Todo esto me ha hecho reflexionar ampliamente sobre el propósito relativo de la informática; el cual hasta ahora entendía como la búsqueda de soluciones a grandes problemas y el cual ahora entendería como la búsqueda de soluciones a todos y cada uno de los problemas que se nos pueden presentar diariamente mediante el uso de las tecnologías disponibles.

Así pues es importante darnos cuenta de la importancia e influencia de la informática en el mundo actual, viéndola como un motor de cambio que permite a través de aportar una serie de herramientas que nos permitan dar soluciones para todo.

## 8.2 Revisión de lo aprendido

En general lo aprendido tiene relación principalmente con las tecnologías web y las redes neuronales como método de aprendizaje computacional.

Este proyecto me ha permitido hacer hincapié en una parte del mundo de la informática que hasta ahora había permanecido desconocido para mí. He encontrado apasionante no solo aprender las tecnologías utilizadas actualmente para el desarrollo de una página web de forma que esta sea ágil y eficiente, sino también el hecho de la integración de las redes neuronales en dicha página web.

Particularmente este proyecto ha servido para aprender a manejar en otras tecnologías back end como Python y para adaptar e interconectar las tecnologías front end que he estado usando a lo largo de la carrera para el desarrollo web, a diversos usos más allá de la creación de una página web.

## 8.3 Posibles mejoras

Aunque la aplicación cumple todos los requisitos propuestos inicialmente en el proyecto, esto no quiere decir que dicha aplicación no esté sujeta a cambios o mejoras del sistema capaces de añadir otras funcionalidades que permitan construir una aplicación con un mayor número de elementos útiles de cara al usuario. A fin y al cabo, el desarrollo de la misma ha supuesto cambios y modificaciones constantes de dichos elementos. Algunos de estas posibles mejoras serán las siguientes:

- 1- Indexación de información personal sobre los autores con los que generamos cada poema.

Disponer de otra vista dentro de la web capaz de mostrar fragmentos de los versos de los poetas, así como su estilo de escritura o su contexto histórico. Esta tarea aportaría información muy útil para los usuarios, ya que podrían elegir en base a los fragmentos de verso extraídos, que poeta les gustaría utilizar más para generar su poema. Por ejemplo para no tener que registrar o guardar una cantidad de información bastante grande, podemos optar por utilizar datos enlazados y consultarlos mediante lenguaje SPARQL. Una ontología interesante para realizar dicha tarea sería wikidata o dbpedia.



## 2- Mejorar la indentación métrica.

Es decir, poder adaptar la salida del poema a un formato más equivalente al formato métrico seleccionado por el usuario. Por ejemplo si el usuario eligiera un formato de soneto, el esquema métrico que debería tratar de simular el poema sería: ABBA ABBA CDC CDC con 11 sílabas métricas cada verso.

## 3- Realización de un estudio avanzado de las palabras usadas en el poema y su acentuación fonética.

La realización de dicho estudio se llevaría acabo haciendo un análisis fonético que evalúe donde recaería la sílaba tónica de cada palabra. Además se analizaría el ritmo con el que se pronuncia cada palabra del verso original, a fin de establecer palabras afines o similares en ritmo y en esquema fonético. De este modo podríamos introducir una mayor variabilidad en el vocabulario a usar y hacer con ello mejores versos.

## 4- Mayor personalización de los poemas generados.

Los poemas actuales tienen una baja personalización, actualmente solo se puede escoger el autor, la semilla a colocar al principio del poema, el nombre de escritor que quiere usar cada usuario para sí mismo y para el título del poema.

Sería interesante por lo tanto incluir algunos aspectos más de personalización como por ejemplo seleccionar la temática del poema, seleccionar la posición donde irá la semilla del poema, o poder seleccionar también el tipo de métrica del poema.

Realizar todo esto implicaría un tratamiento más específico del modelo generado por la red neuronal, que aun no siendo el tema a investigar podría generar mejores resultados al colocar el poema de forma más visual de cara al usuario.

#### 5- Creación de un foro interactivo.

Muchas de las web existentes dotadas de este tipo de uso son a su vez foros de intercambio de likes, opiniones o contenido generado en la propia web. Por lo tanto una posible buena idea sería la de optar por dar opción a registrarse en la web y poder así interactuar con la comunidad creada entorno a los amantes de la poesía automática o de la poesía en general.

#### 6- Poder imprimir o borrar poemas generados por un usuario no anónimo.

Si se desarrolla un foro dentro de la aplicación web, podemos dotar además a cada usuario de la posibilidad de imprimir o borrar los poemas que ellos mismos han generado con su perfil. De este modo cada usuario podrá en todo momento disfrutar de sus poemas y descargarlos más de una vez.

#### 7- Gestión de hilos para el procesado de varias peticiones en el mismo tiempo.

Actualmente uno de los problemas de eficiencia de la web reside en que la web trabaja de forma secuencial en base a las peticiones que recibe, pudiendo no soportar un mayor número de procesos que el número de hilos que componen dicha web. Una posible solución sería una gestión de procesos por hilos de forma que si el servidor se encontrase ocupado en un momento determinado, pudiera seguir recibiendo peticiones y procesarlas. Además si el procesador alcanzase un número de peticiones límite podríamos hacer que se almacenasen dichas peticiones en base a una cola por prioridad como FIFO (First In First Out), la cual atendiese por orden de llegada las diferentes peticiones al servidor.

## 9. Bibliografía

### Anexo I (Índice de tablas)

Tabla 2.1. Funcionalidades. ....	21
Tabla 2.2. Requisito funcional 1. ....	22
Tabla 2.3. Requisito funcional 2. ....	23
Tabla 2.4. Requisito funcional 3. ....	23
Tabla 2.5. Requisito funcional 4. ....	23
Tabla 2.6. Requisito funcional 5. ....	24
Tabla 2.7. Requisito funcional 6. ....	24
Tabla 2.8. Requisito funcionales vs objetivos del sistema. ....	25
Tabla 2.9. Casos de uso. ....	26
Tabla 2.10. CDU-01 Navegar por las vistas de la interfaz. ....	28
Tabla 2.11. CDU-02. Seleccionar el país del poeta. ....	29
Tabla 2.12. CDU-03. Seleccionar al poeta. ....	30
Tabla 2.13. CDU-04. Escribir título y semilla. ....	31
Tabla 2.14. CDU-05. Generar poema. ....	32
Tabla 2.15. CDU-06. Imprimir poema. ....	33
Tabla 2.16. CDU-07. Búsqueda por campos clave. ....	34
Tabla 2.17. CDU-08. Consultar listado de poemas. ....	35
Tabla 2.18. CDU-09. Comparar redes neuronales. ....	36
Tabla 2.19. Matriz de trazabilidad. ....	37
Tabla 3.1. Planificación temporal del proyecto. ....	39
Tabla 3.2. Costes humanos. ....	45
Tabla 3.3. Especificaciones ordenador de desarrollo. ....	46
Tabla 3.4. Costes Hardware. ....	47
Tabla 3.5. Costes Software. ....	47
Tabla 3.6. Costes adicionales. ....	47
Tabla 3.7. Total de costes. ....	48
Tabla 4.1. Descripción de la herramienta “Icono home”. ....	64
Tabla 4.2. Descripción de la herramienta “Icono generar poema”. ....	65
Tabla 4.3. Descripción de la herramienta “Icono testear redes”. ....	65
Tabla 4.4. Descripción de la herramienta “Icono poemas publicados”. ....	66

Tabla 4.5. Descripción de la herramienta “Menú principal en inglés”. .....	66
Tabla 4.6. Descripción de la herramienta “Barra de búsqueda”. .....	67
Tabla 4.7. Descripción de la herramienta “Generar poema”. .....	67
Tabla 4.8. Descripción de la herramienta “Imprimir poema”. .....	68
Tabla 4.9. Tabla de datos de poema. ....	72
Tabla 4.10. Volumen inicial de datos almacenados. ....	75
Tabla 4.11. Crecimiento anual estimado de los datos almacenados. ....	76
Tabla 6.1 Validación de los casos de uso. ....	110
Tabla 6.2 Gestión de errores. ....	114

**Anexo II (Índice de imágenes)**

Imagen 2.1. Red neuronal convolucional. ....	14
Imagen 2.2. Estructura general de una red neuronal. ....	17
Imagen 2.3. Funcionamiento de una red neuronal recurrente. ....	18
Imagen 2.4. Diagrama de casos de uso. ....	27
Imagen 3.1 Diagrama de Gantt. ....	41
Imagen 4.1. Estructura patrón MVC. ....	51
Imagen 4.2. Menú inicial versión uno. ....	56
Imagen 4.3. Menú inicial versión dos. ....	56
Imagen 4.4. Menú inicial versión tres. ....	57
Imagen 4.5. Poema recién generado versión uno. ....	58
Imagen 4.6. Poema recién generado versión dos. ....	58
Imagen 4.7. Poema recién generado versión tres. ....	59
Imagen 4.8. Información poeta versión uno. ....	60
Imagen 4.9. Información poeta versión dos. ....	60
Imagen 4.10. Consulta parametrizada versión uno. ....	61
Imagen 4.11. Consulta parametrizada versión tres. ....	62
Imagen 4.12. Consulta últimos poemas versión uno. ....	62
Imagen 4.13. Consulta últimos poemas versión tres. ....	63
Imagen 4.14. Comparativa de redes neuronales versión tres. ....	64
Imagen 4.15. Redes neuronales simples. ....	77
Imagen 4.16. Redes neuronales recurrentes LSTM. ....	78
Imagen 4.17. Redes neuronales recurrentes GRU. ....	78
Imagen 4.18. Red neuronal LSTM en detalle. ....	79
Imagen 5.1. Controlador principal. ....	83
Imagen 5.2. Controlador encargado de mostrar el formulario. ....	83
Imagen 5.3. Controlador encargado de crear el poema. ....	84
Imagen 5.4. Controlador encargado de indexar poemas. ....	85
Imagen 5.5. Controlador encargado de indexar poemas por palabras clave. ....	85
Imagen 5.6. Controlador encargado de comparar poemas de diversas redes. ....	86
Imagen 6.1. Ejemplo de la red neuronal de caracteres. ....	100

Imagen 6.2. Ejemplo de la red neuronal de palabras. ....	109
Imagen 7.1. Menú inicial. ....	120
Imagen 7.2. Generar poemas. ....	121
Imagen 7.3. Imprimir poema. ....	122
Imagen 7.4. Consultar la base de datos. ....	123
Imagen 7.5. Realizar consultas específicas. ....	124
Imagen 7.6. Testeando las redes neuronales. ....	125

**Anexo III (Referencias)**

- Phadkay, V. (Ed.). (2017). Deep Learning with Keras. Birmingham, UK: Editorial Packt Publishing Ltd.
- Ghazvininejad M., Shi X., Choi Y. y Knight K. (2016). Generating Topical Poetry. *Information Sciences Institute & Computer Science Department University of Southern California*, 9.
- Rajeswar S., Subramanian S., Dutil F., Pal C. y Courville A. (2017). Adversarial Generation of Natural Language. *MILA, Université de Montréal. École Polytechnique de Montréal*, 11,
- Yan R., Li C., Hu X. y Zhang M. (2016). Chinese Couplet Generation with Neural Network Structures. *Institute of Computer Science and Technology, Peking University, Beijing 100871, China*, 11.
- Yan R. (2016). i, Poet: Automatic Poetry Composition through Recurrent Neural Networks with Iterative Polishing Schema. *Department of Computer Science, Peking University*, 7.
- Wen T., Gasic M., Mrksic N., Su P., Vandyke D. y Young S. (2015). Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. *Cambridge University Engineering Department, Trumpington Street, Cambridge, UK*, 11.
- Habernal I. y Gurevych I. (2016). Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM. *Ubiquitous Knowledge Processing Lab (UKP) Department of Computer Science, Technische Universitat Darmstadt*, 11.
- Recopilación de poemas de poesía castellana. (2001) . Los Poetas. *Los-poetas.com*.  
Recuperado de: <http://www.los-poetas.com/> [Último acceso: 17 6 2019]
- ¿Que es tokenizar? (2012). Procesamiento del lenguaje natural. *pdln.blogspot.com*.  
Recuperado de: <http://pdln.blogspot.com/2012/10/normalizacion-del-texto-tokenizacion.html>  
[Último acceso: 17 6 2019]

Redes neuronales recurrentes. (2017). Red Neuronal Recurrente – RNN. [www.diegocalvo.es](http://www.diegocalvo.es).

Recuperado de: <http://www.diegocalvo.es/red-neuronal-recurrente/>

[Último acceso: 17 6 2019]

Funcionamiento de una red LSTM. (2016). Understanding LSTM and its diagrams. *medium.com*.

Recuperado de: <https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714> [Último acceso: 17 6 2019]

Datos del instituto Cervantes. (2017). El Instituto Cervantes presenta hoy el anuario «El español en el mundo 2017». *www.cervantes.es*.

Recuperado de: [https://www.cervantes.es/sobre\\_instituto\\_cervantes/prensa/2017/noticias/Presentaci%C3%B3n-Anuario-2017.htm](https://www.cervantes.es/sobre_instituto_cervantes/prensa/2017/noticias/Presentaci%C3%B3n-Anuario-2017.htm) [Último acceso: 17 6 2019]

Sueldo de un desarrollador web junior. (2018). El sueldo de un desarrollador web con un año de experiencia, superior a la media salarial española. *www.observatoriorh.com*

Recuperado de: <https://www.observatoriorh.com/mercado-de-trabajo/sueldo-desarrollador-web-ano-experiencia-superior-media-salarial-espanola.html> [Último acceso: 17 6 2019]

Sueldo de un jefe de proyectos. (2019). Salarios para empleos de Jefe de proyecto en España. *www.indeed.es* Recuperado de: <https://www.indeed.es/salaries/Jefe-de-proyecto-Salaries>

[Último acceso: 17 6 2019]

Sueldo de un analista programador. (2019). Salarios para empleos de Analista programador/a en España. *www.indeed.es* Recuperado de: <https://www.indeed.es/salaries/Analista-programador/a-Salaries> [Último acceso: 17 6 2019]

Sueldo de un programador web senior. (2019). Salarios para empleos de Programador/a senior en España. *www.indeed.es* Recuperado de: <https://www.indeed.es/salaries/Programador/a-senior-Salaries> [Último acceso: 17 6 2019]

Parámetros disponibles en Keras. (2019). Recurrent Layers – Keras Documentation. *keras.io*

Recuperado de: <https://keras.io/layers/recurrent/> [Último acceso: 17 6 2019]



Cómo crear una red neuronal recurrente básica. (2019). Keras LSTM tutorial – How to easily build a powerful deep learning language model. *adventuresinmachinelearning.com*

Recuperado de: <https://adventuresinmachinelearning.com/keras-lstm-tutorial/>

[Último acceso: 17 6 2019]

Valores para crear una red neuronal recurrente básica. (2018). Una sencilla Red Neuronal en Python con Keras y Tensorflow. *aprendemachinelearning.com*

Recuperado de: <http://www.aprendemachinelearning.com/una-sencilla-red-neuronal-en-python-con-keras-y-tensorflow/> [Último acceso: 17 6 2019]

Que función tiene el parámetro Recurrent Dropout. (2017). Keras: the difference between LSTM dropout and LSTM recurrent dropout. *stackoverflow.com*

Recuperado de: <https://stackoverflow.com/questions/44924690/keras-the-difference-between-lstm-dropout-and-lstm-recurrent-dropout> [Último acceso: 17 6 2019]

Que función tiene el parámetro Dropout. (2017). How to Use Dropout with LSTM Networks for Time Series Forecasting. *machinelearningmastery.com*

Recuperado de: <https://machinelearningmastery.com/use-dropout-lstm-networks-time-series-forecasting/> [Último acceso: 17 6 2019]

Tipos de activaciones. (2017). Different types of Activation functions in Deep Learning. *machineintelligence.com* Recuperado de: <http://www.machineintelligence.com/different-types-of-activation-functions-in-keras/> [Último acceso: 17 6 2019]

Funciones de los parámetros de una red LSTM. (2019). Glosario sobre aprendizaje automático. *developers.google.com* Recuperado de: [https://developers.google.com/machine-learning/crash-course/glossary?hl=es-419#gradient\\_descent](https://developers.google.com/machine-learning/crash-course/glossary?hl=es-419#gradient_descent) [Último acceso: 17 6 2019]

Por qué usar el optimizador Adam. (2018). Adam—latest trends in deep learning optimization. *towardsdatascience.com* Recuperado de: <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c> [Último acceso: 17 6 2019]