UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

"¿De qué se habla en la correspondencia de Jovellanos? Análisis automático de textos adaptado al castellano del siglo XVIII"

DIRECTOR: Susana Irene Díaz Rodriguez

AUTOR: Oriol Invernón Llaneza

Vº Bº del Director del Proyecto

Agradecimientos

Esta sección no es en absoluto obligatoria, pero es el lugar correcto para dedicar el proyecto a las personas/instituciones/empresas/… que se desee.

Resumen

Este proyecto consta de dos partes diferenciadas, en la primera parte se ha desarrollado un programa en R para el análisis de la correspondencia conocida de Gaspar Melchor de Jovellanos. Para realizar este análisis se han utilizado técnicas de procesamiento de lenguaje natural adaptadas al castellano de la época en la que se escribieron dichas cartas para poder trabajar con ellas desde un punto de vista computacional. Posteriormente se han utilizado técnicas de aprendizaje automático, disponibles en diversas librerías de R, para identificar las distintas temáticas y agrupar los personajes (o el subconjunto de personajes) que mantienen correspondencia con Jovellanos según las mismas.

En la segunda parte se ha desarrollado una página web con varios grafos, realizados con D3.js, en los que se visualizan los datos de la correspondencia de Jovellanos, incluidos los conseguidos durante la primera parte. Además, se incluyen diferentes gráficas sobre los datos extraídos.

Palabras Clave

Historia, Correspondencia Jovellanos, Análisis lenguaje natural, Aprendizaje, Grafos, R, D3.js.

Abstract

History, Jovellanos' mail, Natural language analysis, Unsupervised learning, Force directed Graphs, R, D3.js.

Keywords

History, Jovellanos' mail, Natural language analysis, Unsupervised learning, Force directed Graphs, R, D3.js.

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Como se explicará en el apartado de Introducción, este proyecto se ha desarrollado con la motivación de trabajar un campo multidisciplinar que se explota poco y hacer más accesible la información de un personaje ilustre como Gaspar Melchor de Jovellanos.

Los objetivos son utilizar las técnicas de aprendizaje automático disponible para comprender que temas trataba Jovellanos en la correspondencia, sin tener que pasar por leer todas y cada una de las cartas. Además, se busca presentar esta información de forma interactiva, clara y dinámica en una sencilla página web.

El alcance del proyecto comprende:

- El estudio de la correspondencia mediante la realización o modificación de un lematizador y el análisis de los datos con algoritmos de aprendizaje no supervisado.
- La clasificación de las cartas según lo analizado con anterioridad.
- El desarrollo de una sencilla página web para albergar visualizaciones de los datos. Incluidos una serie de grafos interactivos y dinámicos.

1.2 Resumen de Todos los Aspectos

Este documento está dividido en los siguientes apartados: Introducción, Planificación del proyecto, Análisis del lenguaje, Desarrollo de la página web, Conclusiones y ampliaciones, Presupuesto, Referencias y Anexos.

En la introducción se tratará de forma un poco más detallada lo resumido en el apartado anterior, se comentarán los sistemas existentes con los que se comparten funcionalidades y se explicarán las distintas alternativas en enfoque y tecnologías que se barajaron a la hora de realizar el proyecto.

En el apartado de planificación se presentará un resumen del EDT (estructura de descomposición del trabajo) del proyecto, el cual se ha seguido para realizarlo y un resumen del presupuesto calculado en base a esta planificación. Este presupuesto será tratado con más detalle en el apartado del mismo nombre.

Los siguientes dos puntos representan el contenido del trabajo. En el primero, se describirá el proceso seguido para realizar el análisis de textos, incluyendo una explicación de los algoritmos utilizados y de los resultados obtenidos. En el segundo, se tratará la parte de desarrollo web siguiendo el siguiente esquema: Análisis, Diseño, Implementación, Pruebas y Manuales.

Por último, en las conclusiones y ampliaciones se tratará si los resultados obtenidos eran los esperados y si se han cumplido las expectativas. Además, se plantearán diferentes tareas de mejora y ampliación del trabajo.

1.3 Otros Apartados

Otros apartados que el alumno o el director del proyecto consideren relevantes.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

El proyecto consiste en dos partes diferenciadas: el análisis del texto de la correspondencia de Jovellanos, y una pequeña página web en la que se puedan visualizar tanto los datos obtenidos como la correspondencia en sí.

El principal motivo para desarrollar este proyecto es transmitir, de forma accesible, la información recogida en la correspondencia de un personaje histórico tan importante para la historia de Asturias como Gaspar Melchor de Jovellanos, sin que sea necesario leerse una a una las miles de cartas que la componen. Además, expande un campo multidisciplinar que, sobre todo en habla hispana, está sin explotar y resulta ampliamente beneficioso de cara a educación y cultura.

Personalmente, este proyecto da una oportunidad de trabajar con herramientas que no se enseñan en el grado y en un campo que se trata poco.

2.2 Objetivos del Proyecto

Los objetivos de este proyecto, explicados de forma resumida, son los siguientes:

1. Realizar un análisis de lenguaje natural adaptado a la correspondencia de Gaspar Melchor de Jovellanos y, por lo tanto, al lenguaje del siglo XVIII-XIX.
2. Utilizar diferentes métodos de aprendizaje automático para identificar los temas de conversación que aparecen en la correspondencia.
3. Generar diferentes visualizaciones de los datos obtenidos para la fácil interpretación de los resultados obtenidos, por ejemplo, *wordclouds*.
4. Diseñar una página web sencilla y clara para alojar las diferentes visualizaciones.
5. Generar varios grafos para visualizar tanto las conexiones entre personajes en la correspondencia como los datos obtenidos en el análisis.

2.3 Estudio de la Situación Actual

El sistema más similar al desarrollado en este trabajo es el proyecto Republic Of Letters de la universidad de Stanford. En esta web se alojan diferentes visualizaciones de la correspondencia de personajes históricos como Voltaire, Benjamin Franklin, Galileo Galilei, John Locke, etc… Los puntos en común con ese sistema son:

- Visualización de la correspondencia en un grafo sobre mapa. (Realizado en las prácticas de empresa por lo que no forma parte del proyecto actual)
- Visualización de los corresponsales mediante un grafo.
- Identificación de comunidades en dicho grafo.
- Diferentes gráficas sobre la correspondencia.

Respecto a estos puntos en común, se pretende mejorar:

- El rendimiento de los grafos y el grafo sobre mapa.
- La accesibilidad y claridad de los datos representados ya que el proyecto va dirigido a interesados en la historia, los cuales no tiene por qué tener conocimientos técnicos.

En cuanto a las diferencias con Republic Of Letters, la principal es la realización de un tratamiento del contenido de la correspondencia con la que se trata. Como se ha mencionado anteriormente, la primera parte de este proyecto consiste en el análisis del lenguaje encontrado en las cartas de Gaspar Melchor de Jovellanos y la extracción de datos como los temas tratados en las mismas. Además, se añaden las siguientes funcionalidades al grafo:

- Consultas a Wikipedia mediante el click en los personajes históricos que aparecen en el grafo y tienen página propia.
- Los nodos del grafo son móviles y seleccionables.
- Multiselección de nodos.
- Filtro por número total de cartas entre Jovellanos y los corresponsales.
- Grafos tanto del total de la correspondencia como solo de las cartas enviadas o recibidas.

Sobre las herramientas utilizadas, en cuanto al lenguaje para realizar la parte de análisis se escogió R ante Python ya que es el más utilizado para el análisis de datos y tiene acceso a un mayor número de librerías relacionadas con minería de textos, aprendizaje, etc… Para el sitio web se ha considerado Spring, pero se ha elegido realizarla directamente en HTML y JavaScript debido a la sencillez de cara al desarrollo, ya que la página solo debe alojar las visualizaciones. En cuanto al grafo se ha valorado desde utilizar directamente las visualizaciones de Neo4j hasta una multitud de diferentes herramientas para creación de gráficas tanto de pago (Highcharts, Anycharts, …) como gratuitas (Chart.js, Cytoscape.js, …). Finalmente, se ha elegido D3.js por ser la herramienta que mejor se ajusta a las necesidades del proyecto gracias a su capacidad de personalización, por su amplia y activa comunidad y por la gran variedad de visualizaciones disponibles.

2.3.1 Evaluación de Alternativas

Debido a la naturaleza de la primera parte de este proyecto, la única alternativa que se barajó para ella fue la posibilidad de utilizar Python en vez de R, como ya se ha explicado en el apartado anterior.

En cuanto a la segunda parte, se consideraron dos opciones más que serán explicadas a continuación.

2.3.1.1 Aplicación web con Spring

2.3.1.1.1 Descripción

La idea tras esta alternativa era desarrollar una aplicación web, con el mismo contenido que la página actual, bajo el framework Spring y con una base de datos NoSql como MongoDB (documental) o Neo4j (grafo).

2.3.1.1.2 Ventajas

- Considerable experiencia en Java.
- Experiencia con Spring y thymeleaf (ASW).
- Neo4J incluye visualización y funcionalidades para grafos.

2.3.1.1.3 Desventajas

- Mayor complejidad y tiempo de desarrollo.
- Las funcionalidades de Neo4j no encajan completamente con lo requerido.

2.3.1.2 Aplicación web con Angular

2.3.1.2.1 Descripción

Esta alternativa era desarrollar otra aplicación web como la anterior, pero con Angular.

2.3.1.2.2 Ventajas

- Oportunidad de aprender nuevas tecnologías.
- Cantidad de ejemplos con D3.js.

2.3.1.2.3 Desventajas

- Mayor complejidad y tiempo de desarrollo.
- Nula experiencia con el framework.

Capítulo 3. Planificación del Proyecto y Resumen de Presupuestos

3.1 Planificación

El proyecto se ha realizado, siguiendo un calendario de trabajo algo irregular, entre el 7 de marzo de 2018 hasta el 18 de mayo del mismo año. El total de horas trabajadas asciende a 240 horas.

La estructura del trabajo se ha dividido en dos partes principales (ambos hitos) que constituyen los entregables del proyecto: análisis de lenguaje y desarrollo web. Dentro del análisis del lenguaje se ha realizado una división en las siguientes actividades, las cuales son los hitos de esta parte:

- Recopilación de textos.
- Lematizador y DTM.
- Aprendizaje.
- Resultados.

En cuanto a la parte de la web, la división ha sido la siguiente:

- Recursos. (Hito)
- Cuerpo de la página. (Contiene un hito)
- Grafos y funciones. (Hito)
- Validación.

A continuación, se presenta el diagrama de Gant del proyecto:

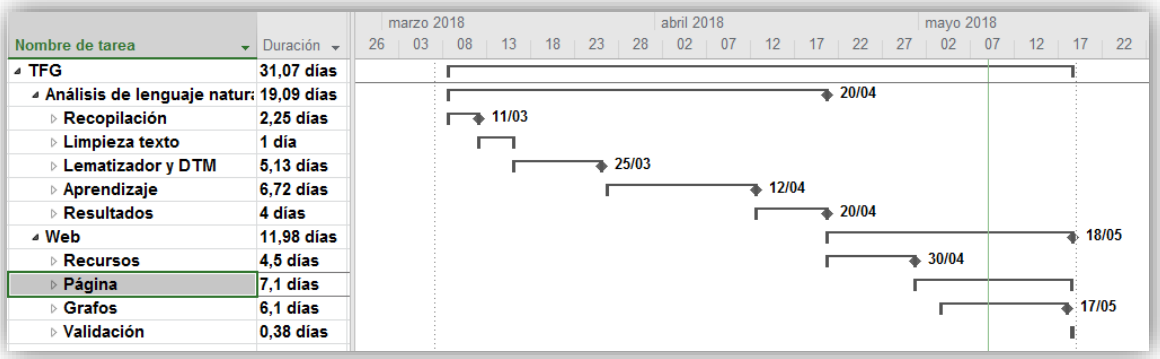

Ilustración 1. Diagrama de Gantt del proyecto.

3.2 Resumen del Presupuesto

El presupuesto para el cliente se ha resumido en las dos partes diferenciadas del proyecto: el análisis de lenguaje y el desarrollo de la web. El total asciende a **ONCE MIL QUINIENTOS OCHO EUROS CON SEIS CÉNTIMOS** (11.508,06 €).

Item	Concepto	Cantidad	Precio Unitario	TOTAL
0	*Análisis de lenguaje*	1,00	7.625,13 €	7.625,13 €
1	*Desarrollo de la web*	1,00	1.885,66 €	1.885,66 €
			Subtotal	9.510,79 €
			IVA (21%)	1.997,27 €
			TOTAL	**11.508,06 €**

Tabla 1. Presupuesto para el cliente.

El presupuesto completo y su desarrollo se encuentran explicados en el capítulo 10.

Capítulo 4. Análisis de lenguaje de la correspondencia

4.1 Resumen

Esta primera parte del trabajo consiste principalmente en la recopilación, limpieza y lematización del texto, por una parte, y la aplicación de diferentes algoritmos de aprendizaje automático por la otra. En este caso, se han utilizado algoritmos de agrupamiento (clustering), para juntar la correspondencia según el tema tratado en cada carta.

Todo el trabajo de esta parte ha sido realizado en R, el cual es un lenguaje especialmente diseñado para computación estadística. Como es un proyecto GNU, es software gratuito ya que se encuentra bajo la licencia pública general de GNU, por lo que está permitido su uso, estudio y modificación. Además, tiene acceso a multitud de librerías sobre minería de textos y aprendizaje, por eso ha sido elegido para este proyecto.

4.2 Minería de texto

El primer paso para realizar este análisis, como resulta lógico, es recoger todo el texto que se desea usar. En este caso, se ha recopilado la correspondencia de Gaspar Melchor de Jovellanos, que se encuentra en la web www.jovellanos2011.es, en un archivo CSV (valores separados por comas) con la siguiente estructura:

```
Id;Objeto;Fecha;Lugaremisión;Lugarrecepción;Escribea;Recibede;año;Textodelacarta

1;enlace; 7/3/1770;Sevilla;Madrid;Campomanes;;1770;"Muy señor mío…"
```

Ilustración 2. Estructura del archivo contenedor de la correspondencia.

Tras recoger todos los textos, el archivo tiene 2133 líneas, una por carta.

4.2.1 Limpieza del texto

A continuación, se realiza la limpieza del texto. Para ello se ha usado el paquete tm (*Text Mining Package*) dado que tiene todos métodos básicos que son necesarios para esta tarea.

El primer punto a tratar es la estructura de datos usada para gestionar documentos de texto, en el caso de *tm* es el *Corpus*, que se inicializa fácilmente leyendo el CSV. Los *Corpus* son colecciones de documentos (en este caso cada carta) que contienen lenguaje natural. A parte del propio texto incluye dos tipos de metadatos.

Una vez construido el *Corpus*, el texto se pasa a minúscula y se eliminan caracteres gráficos, espacios sobrantes, puntuación, números y palabras vacías. La puntuación, los números y los espacios sobrantes se eliminan de forma fácil mediante funciones propias de *tm*. Sin embargo, las funciones disponibles para los caracteres gráficos y las palabras vacías para castellano se quedan cortas, por lo que es necesario hacer listas propias de palabras y caracteres especiales.

Para las palabras vacías, las cuales se definen como aquellas que no tienen significado propio (artículos, pronombres, preposiciones, …), se ha creado un archivo propio de texto plano con una palabra por línea. Además de los tipos de palabras enumerados antes, al tratarse de cartas se han añadido las diferentes fórmulas de cortesía (servidor, amigo, don, …) y los números romanos.

En cuanto a los caracteres especiales, se ha creado una pequeña función que utiliza una expresión regular para borrar algunos de ellos. Debido a las restricciones en cuánto a la codificación del texto (todas las funciones trabajan con UTF-8), varios caracteres de este tipo han sido eliminados directamente del archivo CSV con todas las cartas.

```
cleanCorpus <- function(corpus){

      corpus <- tm_map(corpus, content_transformer(tolower))

      corpus <- tm_map(corpus, removeNumbers)

      corpus <- tm_map(corpus, removePunctuation)

      corpus <- tm_map(corpus, content_transformer(function(n) { n <-
            gsub("[¡¿'«»ª°°*\"]", "", n)}))

      corpus <- tm_map(corpus, removeWords, c(stopwords("spanish"),
            customStopwords, "al"))

      corpus <- tm_map(corpus, stripWhitespace)

      return(corpus)

}
```

Ilustración 3. Función de limpieza del corpus.

4.2.2 Lematizador y DocumentTermMatrix

La lematización es el proceso de agrupar palabras según su raíz para poder tratarlas como un mismo ítem. En este proyecto se han considerado varios lematizadores, los más importantes siendo: Snowball y el utilizado finalmente.

El lematizador Snowball es una implementación del algoritmo de Porter, creado originalmente para inglés por Martin Porter, que tiene soporte para el castellano entre otros múltiples idiomas. Durante las pruebas realizadas con una pequeña parte de la correspondencia (200 cartas aprox.), quedó claro que, tanto la implementación del paquete *SnowballC* como la de *tm*, tenían problemas con los pretéritos, los subjuntivos y demás formas más complejas. Posiblemente, estos problemas deriven del hecho de haber sido diseñado con el inglés, un idioma bastante más sencillo que el castellano, en mente.

Tras probar otros lematizadores, finalmente se modificó un lematizador de Emilio Torres Manzanera, profesor de estadística e investigación operativa en la Universidad de Oviedo. Este lematizador realiza los siguientes pasos, pasando de uno a otro si no resuelve la palabra:

1. Averigua si la palabra en cuestión es una palabra común.
2. Verifica si aparece en el diccionario.
3. Verifica si hay una palabra con el mismo lexema en el diccionario.
4. Por último, si no ha tenido éxito en los pasos anteriores, se conecta al lematizador GRAMPAL de la UAM (disponible en http://cartago.lllf.uam.es/grampal/grampal.cgi). En este penúltimo paso, se realizan dos peticiones por palabra a la página: una para obtener el código anti CSRF (*Cross-site request forgery*) y otra para lematizar la palabra.

Adicionalmente, se ha incluido una lista de reglas específicas para lematizar algunas palabras con sufijos, prefijos, etc… poco comunes o antiguos.

```
lematizadorGPAL <- function( word ){

        if(word == "") {

                return(NA)

        }

        base.url <- paste("http://cartago.lllf.uam.es/grampal/grampal.cgi?m=analiza&e=")

        csrf <- readLines( base.url, encoding = 'utf-8' )[[59]]

        csrf <- iconv( csrf, "utf-8" )

        csrf <- strsplit(csrf, "\"")[[1]][[6]] #get csrf code

        csrf <- paste(csrf, "&e=", sep="")

        csrf <- paste(csrf, word, sep="")

        word.url <- paste(
        http://cartago.lllf.uam.es/grampal/grampal.cgi?m=analiza&csrf=, csrf, sep = "")

        tmp <- readLines( word.url, encoding = 'utf-8' )

        […]

        if(tmp == "-") { return(NA) }

        return(tolower(tmp))

}
```

Ilustración 4. Extracto de la función que conecta con GRAMPAL.

Los resultados se trabajar con este son más que aceptables, siendo el mayor punto negativo el rendimiento ya que para 2133 cartas tarda algo más de una hora en ejecutarse.

Tras el proceso de lematización, se crea una instancia de la estructura de datos que se utilizará para el aprendizaje. La *DocumentTermMatrix* es una matriz cuyas filas son los documentos (en este caso cartas) y las columnas son las palabras, siendo cada celda las veces que aparece la palabra de la columna en el documento de la fila.

Ejemplo:

Frase 1: “De cada cual según sus capacidades”

Frase 2: “A cada cual según sus necesidades”

DTM	de	cada	cual	según	sus	capacidades	a	necesidades
Frase 1	1	1	1	1	1	1	0	0
Frase 2	0	1	1	1	1	0	1	1

Tabla 2. Ejemplo de DocumentTermMatrix.

Finalmente, con el contenido de la correspondencia se obtiene una matriz de 2133 filas y 14593 palabras.

4.3 Aprendizaje (Clustering)

Una vez obtenida la *DTM*, se procede a la clasificación de las cartas utilizando diferentes algoritmos de *clustering* (aprendizaje no supervisado). Además de los aquí tratados, también se ha utilizado *hdbscan* (extiende *DBSCAN* convirtiéndolo en jerárquico) y *sparcl* (Jerárquico disperso y k-means disperso), pero, como sus resultados no aportaban nada nuevo, se ha decidido no incluirlos en esta explicación.

4.3.1 K-means

K-means (James MacQueen, 1967) es un algoritmo que agrupa los datos en *K* grupos, siendo *K* un valor introducido de forma previa a la ejecución. El algoritmo trabaja de la siguiente manera:

1. Una vez introducidos *K* y los datos, estima las posiciones de los centroides (posición media de todos los puntos de la forma) de cada grupo. Esto se suele realizar aleatoriamente.
2. Seguidamente, cada punto (en nuestro caso cada carta) es asignado al *clúster* del centroide más cercano. Para ello se utiliza la distancia euclidiana al cuadrado. $\min_S \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - c_i||^2$, siendo c_i el centroide del grupo S_i y x cada uno de los puntos asignados a S_i.
3. Una vez asignados todos los puntos se actualizan los centroides, es decir, se calcula la posición media de los puntos de cada grupo que sustituyen a los centroides definidos aleatoriamente en el primer paso. La operación realizada es la siguiente: $c_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$, siendo c_i el centroide del grupo S_i y x cada uno de los puntos asignados a S_i.

El algoritmo repite los dos últimos pasos hasta que se cumpla algún criterio de parada, por ejemplo: los centroides no cambian o la distancia que se mueven está por debajo de un umbral, se alcanza un número límite de iteraciones, la suma de distancias es mínima, …

Como se puede deducir, la principal desventaja de k-means es tener que elegir el valor de *K* previamente. Sin embargo, aunque el valor exacto de *K* no se pueda calcular, existen diferentes técnicas para estimarlo.

En este proyecto se ha utilizado como métrica la suma de distancias dentro del clúster (*whitin clusters sum of squares*), la cual disminuye al aumentar *K* ya que, a mayor número de grupos, menos puntos por grupo y, por lo tanto, menos distancias. Para solucionar este inconveniente, se presenta dicha métrica contra *K*. La implementación utilizada ha sido la del método *fviz_nclust* del paquete *factoextra*:

```
fviz_nbclust(DTM, kmeans, method = "wss", k.max = 25)
```

Con esa línea de código, se genera una gráfica en la que se busca el “codo” donde el valor de la métrica pasa de decrecer rápidamente a un decrecimiento mucho más pausado.

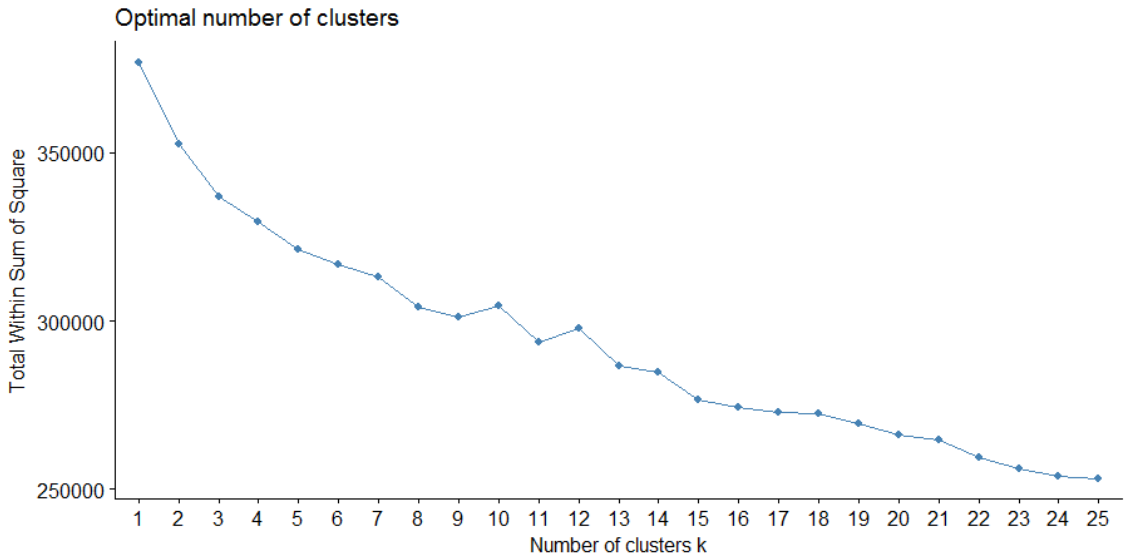

Ilustración 5. Resultado de fviz_nbclust para k-means.

Como se puede observar, este método no es válido para seleccionar K ya que no se aprecia ningún "codo" muy pronunciado. Como alternativa, se ha usado *NbClust*, el cual es un método que devuelve el número óptimo de agrupamientos, dentro de un rango dado, según el algoritmo y la métrica que se especifique. Debido a limitaciones de memoria en el equipo utilizado para los cálculos, estos se han realizado con una versión reducida de la *DocumentTermMatrix* donde se han eliminado las palabras que no aparecen en el 0.005% de las cartas (10 cartas). Los resultados obtenidos se presentan en la siguiente tabla:

	silhouette	frey	ball	pseudot2	gap	Media
Kmeans	3 (index=0.76)	5 (index=1.75)	3 (index=57060)	2 (index=-172)	2 (index=2.31)	3

Tabla 3. Resultados de NbClust para k-means.

Una vez obtenido el número óptimo aproximado de clústeres para K-means, se ha ejecutado el algoritmo y estos han sido los resultados:

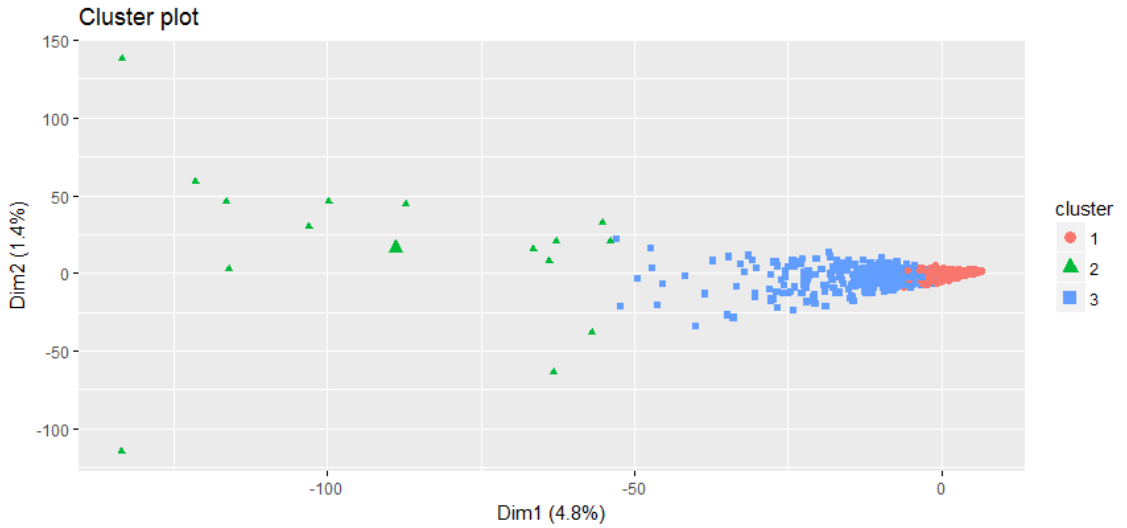

Ilustración 6. Gráfica de los resultados de k-means con el K "óptimo".

Solo con echar un vistazo a la gráfica se puede observar que los resultados no son buenos, ya que, el agrupamiento número 2 es muy ancho y tiene muy pocas observaciones, las cuales bien podrían ser ruido. Para salir de dudas, se ha utilizado un método de validación de clústeres (*silhouette*) y se ha hecho un resumen de los resultados para comprobar diferentes parámetros.

	Agrupamiento 1	Agrupamiento 2	Agrupamiento 3
Nº de observaciones	1656	15	462
Media del ancho *silhouette*	-4.80821	4.455305	-3.322321

Tabla 4. Resultado de silhouette para k-means con K igual a 3.

Este índice calculado por *silhouette*, se calcula de la siguiente manera:

1. Para cada punto *p* cualquiera, calcula la disimilitud media entre él y todos los demás puntos de su propio agrupamiento (*a*).
2. Realiza el mismo cálculo con los puntos de los clústeres a los que no pertenece y se queda con el valor mínimo (*b*), el cual corresponde a la diferencia entre *p* y el clúster más cercano al que no pertenece (clúster vecino).
3. Por último, calcula el ancho *silhouette* siguiendo la siguiente formula: $S = \frac{(b-a)}{\max(a,b)}$

De los resultados obtenidos con *silhouette* se deduce que los resultados del *clustering* son malos ya que los valores del ancho están muy alejados del valor óptimo. Además, con la distribución de observaciones por agrupamiento y la gráfica, se pueden identificar fallos del *clustering* a simple vista, por ejemplo, la separación entre los puntos del grupo 2 indica que al menos parte de ellos deberían ser considerados ruido.

4.3.2 Densidad

Los algoritmos de agrupamiento basados en densidad funcionan localizando zonas con densidad alta (vecindarios) separadas de otras, de densidad similar, por zonas de baja densidad. La implementación de un algoritmo basado en densidad que se ha utilizado en este proyecto es DBSCAN (Density-Based Spatial Clustering of Applications with Noise), presentado en el año 1996 por Martin Ester, Hans-peter Kriegel, Jörg Sander y Xiaowei Xu.

Para realizar el agrupamiento se requieren dos parámetros:

- **Ɛ (Eps):** el valor máximo del radio del vecindario.
- **MinPts:** el mínimo número de puntos de los que debe constar un vecindario.

Entonces, dados valores reales de Ɛ y *MinPts* tal que: Ɛ > 0 y *MinPts* > 0, queda definido el Ɛ-vecindario de un punto cualquiera *p* como el grupo de más de *MinPts* puntos, centrado en *p*, que están, como máximo, a una distancia Ɛ de *p*.

Dado un agrupamiento cualquiera basado en densidad, aparecen los siguientes tipos de punto:

- **Punto central:** es todo aquel punto que tiene más o igual puntos que el mínimo (*MinPts*) a una distancia Ɛ. En el caso anterior, *p* es un punto central.
- **Punto fronterizo:** es un punto *q* que no cumple la condición de los *MinPts* (no es punto central de un vecindario) pero forma parte del Ɛ-vecindario de otro punto *p*, es decir, *q* es directamente alcanzable desde *p*.
- **Ruido:** son todos aquellos puntos que no son fronterizos ni centrales, por lo tanto, no son asignados a ningún clúster y serán considerados *outliers* (valores atípicos).

Resulta importante remarcar la diferencia entre los posibles tipos de alcance entre puntos:

- Directamente alcanzable: un punto *q* es directamente alcanzable desde un punto *p* si este último es un punto central y *q* pertenece a su Ɛ-vecindario.
- Alcanzable: un punto *q* es alcanzable desde otro punto *p* si hay una serie de puntos centrales desde *p* a *q*.
- Conectado: dos puntos están conectados si ambos tienen un punto central alcanzable en común.

Una vez definidos los conceptos y parámetros, el proceso de funcionamiento de DBSCAN es el siguiente:

1. Escoger un punto *p*, que no ha sido asignado a un clúster o marcado como ruido, de forma aleatoria.
2. Encontrar todos los puntos que son alcanzables desde *p*, es decir, calcular el Ɛ-vecindario de *p*.
3. Comprobar si *p* es un punto central, teniendo en cuenta el valor previamente especificado de *MinPts*.
4. Si *p* es un punto central, marcarlo como tal y crear un agrupamiento a su alrededor. Si no lo es, no marcarlo como ruido.

5. Si se ha creado un clúster, expandirlo y añadir todos los puntos que sean directamente alcanzables desde *p*. Si se añade un punto marcado anteriormente como ruido, se añade y se marca como punto fronterizo.
6. Repetir hasta que todos los puntos están asignados a un clúster o marcados como ruido.

Utilizando el paquete "factoextra" de R, se puede usar DBSCAN con una sencilla línea:

dbscan(data, Ɛ, MinPts)

Para buscar el número adecuado de puntos mínimos y de Ɛ, se ha utilizado KNNdistplot para buscar "codos" de forma parecida a lo realizado con K-means. Aun así, solo se han conseguido dos tipos de resultado:

1. Un único clúster relativamente pequeño y una gran mayoría de ruido.
2. Un único clúster abarcando todas las observaciones.

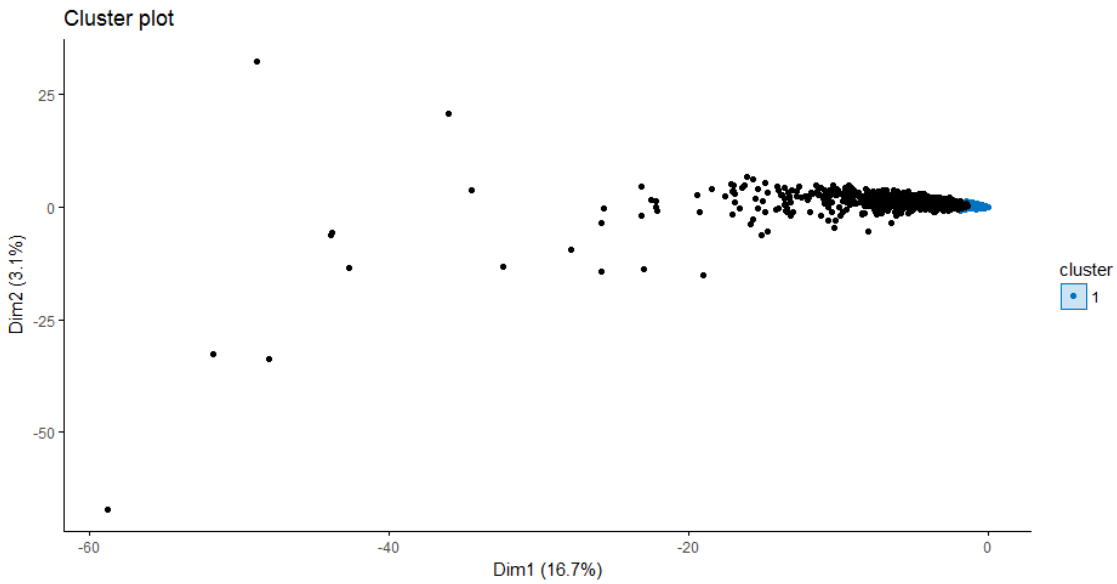

Ilustración 7. Gráfica de uno de los resultados dado por dbscan.

4.3.3 Jerárquico

El agrupamiento jerárquico es un método que busca organizar los *clústeres* de arriba hacia abajo, creando así una jerarquía de agrupamientos. En este proyecto se han considerado los siguientes algoritmos de agrupamiento jerárquicos: agnes y diana.

4.3.3.1 Agnes

Agnes es un algoritmo de agrupamiento jerárquico aglomerativo, también llamados "de abajo a arriba", lo que implica que:

1. Primero, asigna cada dato a su propio clúster.
2. Luego, calcula la distancia entre cada clúster.
3. Por último, junta los dos más similares (cercanos).
4. Los dos pasos anteriores se repiten hasta que solo quede un único clúster.

Para calcular dichas distancias existen varios métodos:

- **Enlace singular:** calcula la distancia de todos los pares de puntos entre los dos clústeres y toma la menor de todos los pares como distancia entre ellos. Utilizando este método se construyen agrupamientos más grandes.
- **Enlace completo:** realiza el mismo procedimiento que la anterior, pero toma la mayor distancia. Suele construir clústeres más compactos.
- **Enlace medio:** Igual que los dos anteriores, pero escoge la distancia media.
- **Enlace singular centroide:** Calcula la distancia entre los dos centroides.
- **Método de Ward de mínima varianza:** minimiza la varianza dentro del clúster mediante juntar en cada paso los dos agrupamientos que menos aumentan la varianza al unirse.

Estos métodos son comunes con Diana.

De forma idéntica al proceso realizado con K-means, se ha utilizado *NbClust* para calcular el número óptimo de agrupamientos con cada uno de los diferentes métodos explicados anteriormente.

	silhoutte	frey	ball	pseudot2	cindex	Media
Singular	3 (index= 0.85)	ERROR	3 (index=57314)	3 (index=3.14)	2 (0.124)	3
Completo	2 (index= 0.86)	8 (index=22.3)	3 (index=56571)	2 (index=3.15)	2 (0.129)	4
Medio	3 (index= 0.85)	ERROR	3 (index=56571)	2 (index=3.15)	2 (0.129)	3
Centroide	2 (index= 0.86)	ERROR	3 (index=56571)	2 (index=3.15)	2 (0.129)	2
Ward	2 (index= 0.34)	ERROR	3 (index=52684)	9 (index=-0.45)	9 (index=0.08)	6

Tabla 5. Resultados de NbClust para los métodos disponibles para Agnes.

Tras generar y observar detenidamente los dendogramas de estos métodos, se ha decido trabajar con el de Ward ya que el que genera un dendograma más claro y mejor distribuido. El siguiente paso ha sido cortar el árbol según el número de clústeres obtenido en la tabla superior, para ello se ha utilizado la función *cutree* con un valor *K* de 6. Los valores de *silhouette* para Ward con 6 agrupamientos son los siguientes:

	C1	C2	C3	C4	C5	C6
Nº de observaciones	1402	560	162	7	1	1
Media del ancho *silhouette*	0.4976	-0.2182	-0.2403	-0.1707	0	0

Tabla 6. Resultados de silhouette para Ward cortado en 6 clústeres.

Como se puede apreciar, aunque con seis clústeres los resultados del ancho son mucho mejores que los obtenidos anteriormente con k-means, no se puede concluir todavía que este agrupamiento es el óptimo ya que hay dos grupos con una sola observación. Seguidamente, se ha realizado el mismo proceso con *K* igual a 4.

	C1	C2	C3	C4
Nº de observaciones	1402	722	8	1
Media del ancho *silhouette*	0.5645	-0.2515	-0.1788792	0

Tabla 7. Resultados de silhouette para Ward cortado en 4 clústeres.

Al igual que en la primera prueba, aparece un clúster con solo una observación y otro con solo 8. Si se aumenta el número de clústeres, el algoritmo se sigue comportando de la misma manera, como se puede apreciar en la siguiente tabla.

	C1	C2	C3	C4	C5	C6	C7	C8	C9
Nº de observaciones	1402	560	155	7	1	1	4	1	2
Media del ancho *silhouette*	0.4976	-0.2182	-0.2403	-0.1707	0	0	-0.212	0	0.06

Tabla 8. Resultados de silhouette para Ward cortado en 9 clústeres.

Debido a esto, se concluye que la división real realizada por Agnes con el método de Ward genera dos grandes agrupamientos (1402 y 731, respectivamente) y, por lo tanto, se ha decidido no utilizar este algoritmo para la clasificación de las cartas, ya que, dividir 2133 de estas en solo dos grupos, es una simplificación excesiva en la que se pierde demasiada información.

	C1	C2
Nº de observaciones	1402	731
Media del ancho *silhouette*	0.5822	-0.27

Tabla 9. Resultados de silhouette para Ward cortado en 2 clústeres.

4.3.3.2 Diana

Al contrario que Agnes, Diana es un algoritmo de agrupamiento jerárquico divisivo (o "de arriba a abajo"), por lo que sigue los siguientes pasos:

1. Asigna todos los puntos al mismo agrupamiento.
2. Seguidamente, calcula la distancia entre los posibles clústeres.
3. Separa cada clúster en los dos que sean menos parecidos (mayor distancia).
4. Finalmente, repite los dos pasos anteriores hasta que hay un agrupamiento por dato.

Se ha trabajado con Diana a la vez que con Agnes, sin embargo se descartó su utilización tras generar los dendogramas.

4.3.4 TopicModeling

TopicModeling es un tipo de minado de texto que consiste en identificar temas (*topics*) en una colección de documentos, comprendiendo como tema un patrón recurrente de palabras concurrentes. En este caso, un tema bien detectado en una serie de cartas sobre poesía sería, por ejemplo, "verso, rima, estrofa". Por lo tanto, *TopicModeling* no deja de ser un método para agrupar datos.

Hay varias técnicas de *TopicModeling* disponibles, este trabajo se ha centrado en LDA (*Latent Dirichlet Allocation*), un modelo estadístico presentado en 2003 por David Blei, Andrew Ng y Michael I. Jordan. LDA se basa en dos consideraciones básicas:

- Cada documento una mezcla de temas presentes en toda la colección.
- Cada palabra de un documento es asignable a uno de los temas presentes en el documento.

Además, LDA asume que los documentos han sido creados de la siguiente manera (extraído del artículo en el cual se presentó):

1. Se decide el número de palabras que constituirán el documento (de acuerdo con una distribución de Poisson).
2. Se elige una mezcla de temas cada uno con una probabilidad. Estos son escogidos de la lista fija de temas con la que se realiza la colección, la cual sigue una distribución de Dirichlet).
3. Para cada palabra:
   3.1. Se escoge un tema de acuerdo con la distribución multinomial decidida en el segundo paso.
   3.2. Se genera la palabra en base a la distribución de palabras del tema.

Luego, LDA intenta reproducir estos pasos en orden inverso para encontrar esa lista de temas fijos con la que ha asumido que sean creado los documentos. Para este análisis, se ha utilizado LDA con el algoritmo de muestreo de Gibbs, el funcionamiento es el siguiente (explicación basada en una previa de Edwin Chen):

1. Itera por cada documento, asignando cada palabra a un tema de forma aleatoria.
2. Para mejorar este primer modelo aleatorio, visita cada palabra de cada documento y, por cada tema, realiza las siguientes operaciones (asumiendo que todas las asignaciones a temas son correctas menos la visitada actualmente):
   2.1. Calcula la proporción de palabras del documento que están asignadas a ese tema [P (topic | document)].
   2.2. Calcula la proporción de asignaciones de esa palabra a ese tema a través de todos los documentos que la contienen [P (word | topic)].
   2.3. Asigna la palabra a un nuevo tema en función de las probabilidades anteriores, lo que se traduce realmente como la probabilidad de que esa palabra fuese generada por ese tema siguiendo el modelo de creación de documentos expuesto anteriormente.
3. Repite los pasos anteriores un gran número de veces hasta alcanzar un estado estable en el que las asignaciones son bastante buenas.

4. Estima la mezcla de temas en cada documento contando la proporción de palabras asignadas a cada tema presentes en ese documento.

En este trabajo se ha recurrido a la implementación de LDA que aparece en el paquete "topicmodels" de R. A continuación, se presenta un ejemplo de cómo se ha ejecutado este algoritmo:

```
burnin <- 5000

iter <- 1700

thin <- 50

seed <-list(2018,11,93,101010,813)

nstart <- 5

best <- TRUE

k <- 8

ldaOut <-LDA(dtm.new, k, method="Gibbs", control=list(nstart=nstart, seed =
seed, best=best, burnin = burnin, iter = iter, thin=thin))
```

Ilustración 8. Ejemplo de utilización de LDA en R.

La función LDA devuelve un objeto que contiene los resultados para su posterior observación y tratamiento de forma sencilla. Los más relevantes para el análisis son:

- **terms:** vector con las palabras por tema.
- **documents:** vector con los documentos, en este caso las cartas.
- **topics:** asignaciones de temas a documentos.
- **gamma:** distribución de temas por documento.

Como se puede observar en el código anterior, al igual que K-means, LDA requiere saber de antemano el número de clústeres (temas). Resulta muy importante elegir un buen número de temas dado que con muy pocos se pierde información, debido a que los temas principales invisibilizan a los menos comunes, y con muchos se generan temas prácticamente iguales que no aportan nada.

Para intentar ajustar ese número de forma automática, se realizó una valoración cruzada (*k-fold cross validation*) sobre los datos de las cartas. Desafortunadamente, los resultados no fueron ya que daban un resultado extremadamente grande que no se ajustaba a la realidad de la correspondencia y, por lo tanto, se pasó a las pruebas "manuales". Se generaron modelos para una gran variedad de número de temas, pero solo se van a tratar los más relevantes: 5, como ejemplo de un modelo en el que se pierde información, y 7 como los mejores resultados.

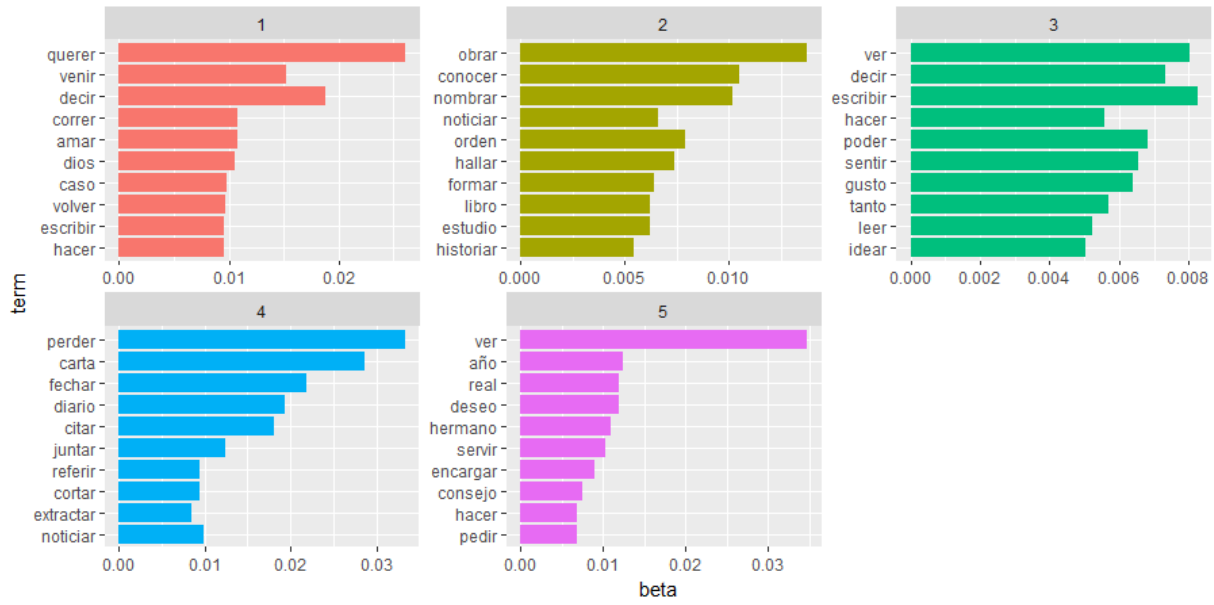

Ilustración 9. Top 10 palabras por tema, TopicModeling con 5 temas.

La gráfica anterior representa las diez palabras más comunes para cinco temas. En ella se puede observar fácilmente tanto de que trata cada tema como la información que se echa en falta al no haber suficientes temas. Por ejemplo, el tema número cuatro corresponde principalmente a las cartas de las que se conoce su existencia, pero cuyo texto se ha perdido. Por otro lado, la ausencia de un tema exclusivo para la poesía o para la guerra de independencia.

Perdida. En la carta de Jovellanos a Campomanes de 23 de julio de 1768 se refiere a otra que le ha escrito después de su llegada a Sevilla, por tanto entre el 29 de marzo y el 23 de julio de ese mismo año.

Ilustración 10. Ejemplo del contenido de una carta perdida (Jovellanos a Campomanes).

Al aumentar el número de temas, se aprecian unos resultados mucho mejores. Por ejemplo, con siete temas obtenemos lo siguiente:

Ilustración 11. Top 10 palabras por tema, TopicModeling con 7 temas.

Tan solo con echar un vistazo a esta gráfica ya podemos identificar los siguientes agrupamientos:

- **Tema 1 (Jovellanos, el político):** las cartas de este tema son de carácter formal como se ve en el uso de "guardar" y "años" debido a la expresión de cortesía "guarde a V.E. muchos años". Esta formalidad es debida al origen institucional de las cartas, mucha de esta correspondencia trata sobre reales decretos y reales ordenes, además de informes sobre los mismos. Asimismo, entran en este tema varias cartas intercambiadas con el Conde de Campomanes, de carácter muy formal debido a la posición de este sobre Jovellanos. Por último, ha asignado a este tema las cartas sobre el Instituto Asturiano de Náutica y Mineralogía, debido también a su carácter formal y político.

Al Conde de Lerena;

Excelentísimo señor: Cumpliendo con la Real Orden que V.E. se ha servido comunicarme con fecha de 7 de setiembre del año anterior, paso a sus manos el adjunto Informe a S.M. sobre la Representación que en 30 de abril del mismo había dirigido a S. R. P. el Director General de Minas, don Francisco de Angulo. El deseo de tomar una plena instrucción del objeto que trata, me ha hecho suspenderle hasta ahora, que con esta misma fecha dirijo a S.M. las resultas de mi principal comisión por la vía reservada de Marina, de quien dimana. Nuestro Señor guarde a V.E. muchos años.

Ilustración 12. Ejemplo de carta sobre política.

A Antonio Valdés y Bazán

Excelentísimo señor: Para enterar al público de la erección y estado del Real Instituto Asturiano he extendido la adjunta noticia, que paso a manos de V.E., suplicándole se digne obtener de S.M. el permiso de publicarla bajo el Augusto nombre del Príncipe de Asturias N.S.; y si S.M. condescendiese benignamente a ello, ruego también a V.E. se digne presentarla a los pies de S.A. y implorar su poderosa protección en favor del Instituto. Uno y otro espero de la bondad de V.E., porque, tratándose de un establecimiento que es obra suya y de su ardiente celo por el bien público, no puede V.E. dejar de mirarle como tal, ni negarle este nuevo testimonio de aquella paternal beneficiencia a cuyo influjo ha nacido y empieza a prosperar. Nuestro Señor guarde a V.E. muchos años Gaspar de Jovellanos.

Ilustración 13. Ejemplo de carte sobre el Real Instituto Asturiano.

- **Tema 2 (Jovellanos, círculo cercano, tono formal):** aunque este tema pueda parecer comodín si solo se miran las palabras, atendiendo a quienes son los corresponsales asignados, resulta fácil reconocer que la gran mayoría son algunos de los amigos y familiares más cercanos de Jovellanos. Los personajes con más apariciones son los siguientes:

Corresponsal	Número de cartas
Carlos González de Posada	30
Juan Meléndez Valdés	10
Lord Holland	8
Francisco de Paula Jovellanos	7
María Gertrudis del Busto y Miranda	6
Josefa Jovellanos	4
Tomás Menéndez Jove	3
Fray Diego Gonzalez	3
Baltasar González de Cienfuegos	3
Campomanes	2
Antonio Valdés y Bazán	2
Francolín de Solares Jove	2
Gregorio Jovellanos	2
Fray Matías Mariño	2
Pedro Manuel de Valdés Llanos	2

Tabla 10. Tabla de frecuencia de los personajes con más de una carta (Tema 2).

El más destacado es Carlos González de Posada, el más íntimo amigo de Jovellanos. Le sigue Meléndez Valdés, ministro de Marina y de los mejores amigos que tenía en Madrid (su hermano fue el segundo director del Instituto Asturiano). También destacan sus familiares, sobre todo su hermano De Paula y su hermana Josefa. La etiqueta de "tono formal" se debe a que, aunque en estas cartas trate con gente cercana, no lo hace en el mismo tono cercano e incluso cariñoso que utiliza en el tema 4.

- **Tema 3 (Correspondencia perdida):** cartas perdidas como en el modelo de 5 temas. Hay que tener en cuenta que no todas las cartas pérdidas son asignadas a este tema debido a que, por referencias a ellas e investigaciones de historiadores como Somoza, se sabe cuál era su contenido y, por lo tanto, si hay suficiente información, serán clasificadas en su tema correspondiente.

Destinatario: Prior de San Marcos de León.

"Perdida. Cit. en la carta del prior de 27 de julio siguiente. Le preguntaba por documentación antigua del archivo de S. Marcos y le pedía copia de una inscripción."

Ilustración 14. Carta perdida asignada al tema 5.

- **Tema 4 (Jovellanos, círculo íntimo):** al igual que el tema 2, este tema se caracteriza por la relación de los corresponsales con Jovellanos. Aquí aparecen también, Carlos González de Posada, Josefa Jovellanos, De Paula, Lord Holland, pero en mayor cantidad. Además, aparecen otros personajes íntimos como su herma Catalina de Sena, la cual no aparecía en el tema 2. Como se puede apreciar en las palabras más usadas y leyendo alguna de las cartas, este tema se caracteriza por la utilización de un lenguaje mucho más cercano y cariñoso que el segundo. A continuación, se presenta la tabla de frecuencias, es importante resaltar que no se han contado todas las cartas de Lord Holland ya que a partir de la carta 1800 (aproximadamente) las únicas cartas que aparecen en este tema son suyas (todas las que no fueron asignadas al tema de la guerra o al segundo tema).

Corresponsal	Número de cartas
Lord Holland	>35
Josefa Jovellanos	42
Carlos González de Posada	39
Baltasar González de Cienfuegos	16
Catalina de Sena (hermana)	15
Francisco de Paula Jovellanos	10
María Gertrudis del Busto y Miranda	8
Tomás de Veri	4
Juan Meléndez Valdés	3
Conde de Ayamans	3
Juan Agustín Ceán Bermúdez	3
Martín Fernández de Navarrete	3
Leandro Fernández de Moratín	3
Pedro Manuel de Valdés Llanos	3

Tabla 11. Tabla de frecuencia de los personajes con más de una carta (Tema 4).

- **Tema 5 (Jovellanos, el recopilador):** este es uno de los temas más complicados de entender a simple vista y requiere de la inspección de unas cuantas cartas asignadas a él para comprender el alcance del mismo. Aunque aparentemente heterogéneo y las cartas puedan diferir bastante más en su asunto que las de temas como poesía, las cartas siempre tienen elementos comunes como los siguientes:
    - **Religión:** incluye movimientos de personal dentro de la Iglesia (nombramientos) de los que se da noticia a Jovellanos, obras antiguas relacionadas con el culto, inscripciones, etc...
    - **Obras escritas:** incluye conversaciones sobre libros y autores históricos.
    - **Obras arquitectónicas:** en numerosas cartas asignadas a este tema se tratan construcciones generalmente de índole religiosa como templos y parroquias.
    - **Cobros de obras:** facturas de obras realizadas.

  La naturaleza de este tema tiene explicación. Tras ser "desterrado" a Asturias, en la década de 1790, Jovellanos se dedicó a viajar por Asturias, Cantabria, Burgos, Euskadi y León, visitando monasterios e iglesias y copiando escrituras antiguas, que luego enviaba a amigos o archivaba para sus trabajos históricos. Esto se puede comprobar fácilmente, revisando alguna de las cartas, por ejemplo, esta carta intercambiada con José Antonio Ruenes.

> "Muy señor mío y mi más estimado dueño: Quedará esta tarde efectuado el andamio que V.S. se sirve encargarme, y mañana haré copiar la consabida inscripción, […] de otras inscripciones de esta nación. Esta misma tarde paso a copiar la inscripción que dije a V.S. de Corao, pues la otra está bien cerca de Santa Cruz, en una casería, y es regular quiera verla V.S. más en su original que en copia, bien que elegirá lo que guste, pues la tendrá también. Ambas son sepulcrales, y ésta es muy parecida a otra que halló Sandoval junto a Burgos, […] A la hora que V.S. me dice procuraré hallarme en Santa Cruz, […] su más seguro servidor Josef Antonio Ru"

Ilustración 15. Ejemplo de carta asignada al tema 5 (Jovellanos, el recopilador).

- **Tema 6 (Jovellanos, el poeta):** uno de los temas que se echaba a faltar en el modelo de cinco temas, la poesía. Este tema no necesita explicación, solo hace falta ver las palabras más frecuentes o algún ejemplo como el siguiente para comprender el contenido.

"Muy señor mío y mi estimado paisano: Doy a usted muy finas y sinceras gracias por el romance […] el entusiasmo poético arrebataron su imaginación de usted y colocaron sus héroes entre los signos del Zodíaco; […] atribuir a los colores de la poesía, ya sabe usted que la poesía didáctica no concede tantas licencias. Pero si considero el romance como poeta, hallo en él mil gracias: muchos pensamientos sublimes y brillantes, muchos versos correctos y armoniosos, algunas ideas originales, y sobre todo un estilo fácil, noble y de bastante majestad. Seguramente usted podr[í]a hacer grandes cosas en poesía, […] cuyas obras creo que no desconocerá usted las hermosas Instituciones poéticas del padre Juvencio, […] El romance tiene sus defectos: algunos versos de mala medida, otros de no buen sonido, […] comunicar a usted mis ideas necesitaba de mucho tiempo y papel […] tener en mí un fino paisano y afecto servidor. Gaspar de Jovellanos."

Ilustración 16. Extracto de una carta en la que Jovellanos valora la obra de un conocido.

- **Tema 7 (Jovellanos durante la guerra):** corresponde a las cartas intercambiadas, principalmente, durante la Guerra de Independencia (la mayoría intercambiadas con Lord Holland) como se puede observar con la alta probabilidad de palabras como ejército, enemigo y España. Además, "juntar" y "cortar", aunque aparezcan en el infinitivo del verbo por el trabajo del lematizador, se deduce (y se puede comprobar fácilmente revisando las cartas asignadas al tema) que provienen de la Junta Suprema Central y de las Cortes de Cádiz, respectivamente.

"Mi muy amado Lord: Por fin usted llegó a Lisboa, como dije, antes que mis cartas a su mano, y así me lo confirma la del 5, escrita de Badajoz, que me entregó el calesero. […] ataque, y tal vez a esta hora estarán empeñados en él nuestros ejércitos. Hemos entrevisto a medias el plan, y aunque no entiendo la materia, me gusta poco. No me parece que hay bastante unión en los tres cuerpos, que deben obrar a mucha distancia contra un enemigo reunido. […]. Es el día de buen hado; hoy hemos celebrado en la capilla de San Fernando la batalla de Bailén. Asistieron el nuncio y los ministros de Inglaterra, Austria, Portugal y Provincias Unidas. Capmany está ya libre de la Gaceta y agregado a los trabajos de Cortes. Pero nos ocupan demasiado los negocios de la guerra y el temor de sus resultas; si malas, al pueblo, si buenas, al general victorioso. Amable Milady: me llama la hora de la Junta nocturna. Saludo a usted muy afectuosa y no menos respetuosamente, y soy de toda la compañía constante amigo."

Ilustración 17. Carta a Lord Holland durante la Guerra de Independencia.

Para intentar mejorar estos resultados, se realizaron cambios al tratamiento del texto para eliminar palabras que se creían vacías como los verbos "hacer", "ver", "decir", "poder", … Sin embargo, los resultados obtenidos fueron bastante peores que los anteriores, tanto con 7 como con 6 temas. Como se aprecia en la siguiente ilustración, los temas tan definidos que fueron obtenidos anteriormente se encuentran diluidos entre ellos.

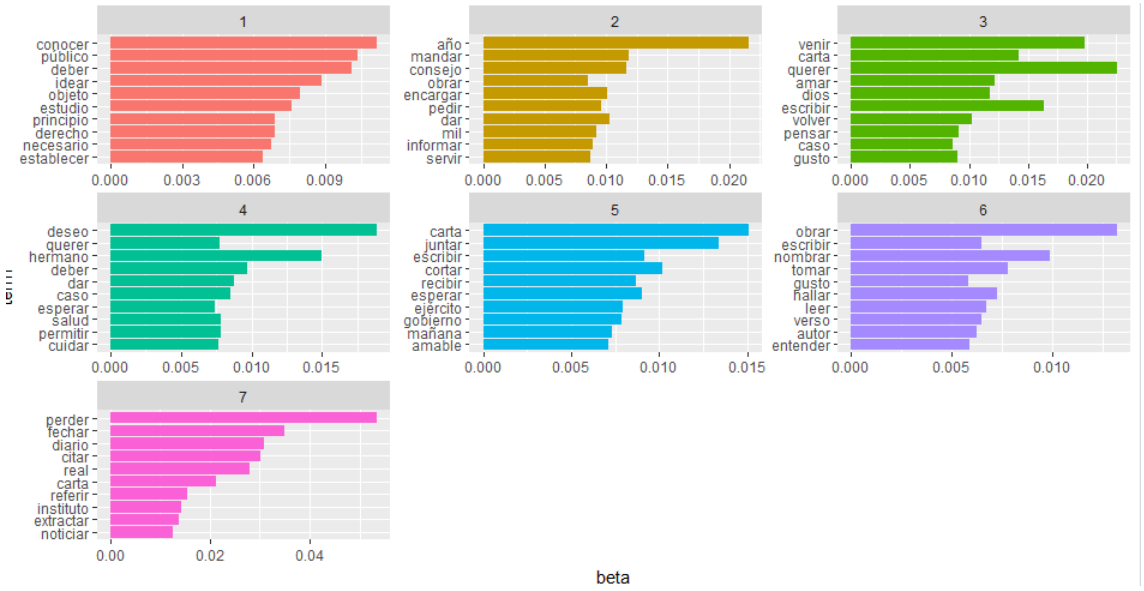

Ilustración 18. Top 10 palabras por tema, TopicModeling 7 temas y sin verbos frecuentes.

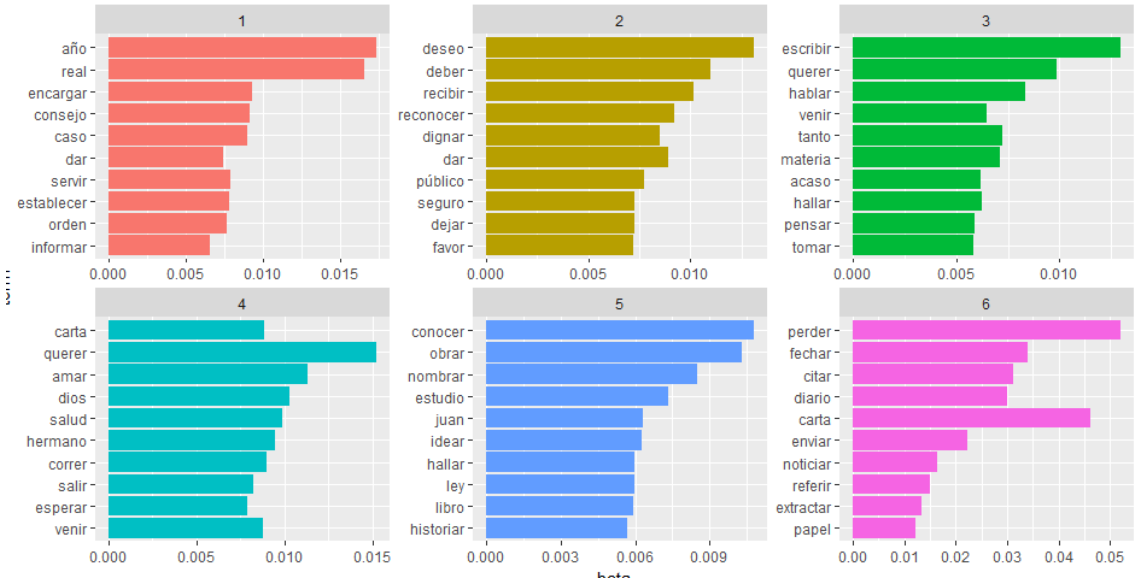

Ilustración 19. Top 10 palabras por tema, TopicModeling 6 temas y sin verbos frecuentes.

Tras esta explicación, se puede apreciar la calidad y precisión del algoritmo LDA cuando el texto está bien tratado y se escoge un número de temas correcto.

4.3.5 Conclusiones

Después de obtener y analizar todos los resultados, como se ha explicado en los apartados anteriores, se ha decidido utilizar los resultados de TopicModeling ya que se consideran los únicos precisos y que representan tanto los temas de conversación como las épocas de la vida de Gaspar Melchor de Jovellanos.

Capítulo 5. Desarrollo de la página web

5.1 Análisis y Diseño

Este apartado contendrá toda la especificación de requisitos y toda la documentación del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

5.1.1 Definición del Sistema

5.1.1.1 Determinación del Alcance del Sistema

Se trata de describir de nuevo el sistema, pero en lugar de repetir lo que ya hemos dicho de él, tenemos que constatar en este apartado hasta donde vamos a llegar en su construcción, es decir, qué límites vamos a poner en el desarrollo estableciendo qué se va a hacer y qué se va a omitir (en general, hasta donde se va a llegar). Podemos por tanto usar todo lo que hemos dicho en descripciones anteriores para ayudar a describir el alcance del sistema. Conviene dejar claro este apartado para así delimitar la labor de análisis y diseño que vamos a hacer a continuación y evitar así no describir aspectos que se han construido o describir cosas que finalmente no van a construirse.

En el caso de que quede claro implícitamente qué se va a hacer en el sistema, esta sección se puede omitir.

--

Como se ha tratado antes el alcance de la página web es relativamente sencillo. Se van a realizar una serie de grafos interactivos, tres en concreto (más dos realizados durante las prácticas de empresa y que, por lo tanto, no serán tratados aquí), con una serie de funcionalidades (explicadas en los requisitos) y una galería de imágenes para albergar otro tipo de visualizaciones estáticas (gráficas, wordclouds, …) generadas durante el análisis de lenguaje.

5.1.2 Requisitos del Sistema

5.1.2.1 Obtención de los Requisitos del Sistema

A continuación, se presenta la lista de requisitos recogidos del cliente.

Código	Nombre Requisito	Descripción del Requisito
R1	Alojamiento correspondencia	Se debe guardar la información de la correspondencia en documentos sencillos y legibles por humanos.
R2	Lectura datos	El sistema debe leer los datos de forma automática de los archivos de origen.
R3	Generar grafos	Se deben generar grafos para las cartas recibidas, enviadas y ambas juntas.
R3.1	Visibilidad grafos	El sistema no enseñará más de un grafo simultáneamente.
R3.2	Navegación grafos	El usuario deberá poder navegar entre grafos mediante pestañas.
R4	Funcionalidad grafos	Los grafos tienen que tener las funcionalidades expuestas en los siguientes apartados.
R4.1	Selección nodos	El usuario debe poder seleccionar nodos.
R4.1.1	Selección múltiple	El usuario debe poder seleccionar varios nodos a la vez.
R4.1.2	Arrastrar nodos	El usuario debe poder mover los nodos seleccionados.
R4.1.3	Estabilidad grafo	El grafo debe volver a una posición estable cuando el usuario suelte los nodos.
R4.1.4	Quitar selección nodos	El usuario debe poder eliminar la selección de los nodos seleccionados.
R4.2	Resaltar nodos	El usuario debe poder resaltar un nodo y sus vecinos.
R4.3	Zoom	El usuario debe poder hacer diferentes niveles de zoom sobre el grafo.
R4.4	Nombre nodos	El usuario debe ser capaz de ver los nombres de los personajes en sus nodos mediante pasar el ratón sobre el mismo y/o una versión resumida del nombre aparecerá encima de este.
R4.5	Wikipedia	El usuario deberá poder hacer una consulta a la Wikipedia (si la tiene en castellano) del personaje.
R4.5.1	Información Wikipedia	El sistema enseñará el contenido del resumen de la página de Wikipedia del personaje.
R4.5.2	Imagen Wikipedia	Además, enseñará también la foto si esta existiese.
R4.6	Filtrar por número de cartas	El usuario deberá poder filtrar a los personajes según el volumen de cartas intercambiadas.
R4.7	Búsqueda personajes	El usuario deberá poder buscar personajes por su nombre completo.
R4.7.1	Resultado búsqueda	El sistema resaltará el nodo al cual corresponda el nombre introducido.
R4.8	Filtro contenido	El usuario deberá poder filtrar a los personajes según los temas identificados en el análisis.
R4.9	Filtro mujeres	El usuario deberá poder filtrar a los personajes

		según si son mujeres o no.
R4.10	Estilo arcos	El sistema deberá elegir el grosor de los arcos del grafo en función del volumen de cartas a representar.
R5	Ayuda usuario	El usuario deberá tener a su disposición la siguiente información:
R5.1	Leyenda	El usuario deberá poder ver la leyenda de colores del grafo en todo momento.
R5.2	Ayuda	El usuario podrá acceder a una sección de ayuda que explique las funcionalidades del grafo.
R6	Navegación página	El usuario deberá poder navegar por la página mediante un menú.
R7	Galería	El sistema ofrecerá un carrusel de imágenes con distintas gráficas generadas durante el análisis.
R7.1	Leyenda Galería	El usuario tendrá a su disposición una descripción de la imagen que esté observando.
R8	Compartir web	El usuario dispondrá de accesos directos para poder compartir la página web en redes sociales.

Tabla 12. Tabla de requisitos de la página web.

5.1.2.2 Identificación de Actores del Sistema

Debido a la naturaleza de la web como simple plataforma de visualización, solo se han identificado dos tipos de usuario: usuario anónimo y el administrador del sistema.

El usuario anónimo es toda aquella persona que se conecte a la web desde fuera. Este tipo de usuarios podrá disfrutar de todas las funcionalidades disponibles en el sistema y acceder a todas las visualizaciones. Como no es necesario ningún sistema de identificación, no aparece la figura del usuario registrado.

Por otro lado, el administrador del sistema tendría la única función de actualizar y mantener los documentos que conforman la base de datos si esto fuese necesario (añadir cartas, quitar cartas, arreglar erratas en nombres, localizaciones, etc…).

5.1.2.3 Especificación de Casos de Uso

Los casos de uso de este sistema consisten prácticamente en su totalidad de las diferentes funcionalidades del grafo. Por lo tanto, para poder representarlo adecuadamente, se han realizado dos diagramas: uno agrupando las funcionalidades del grafo y otro desglosándolas.

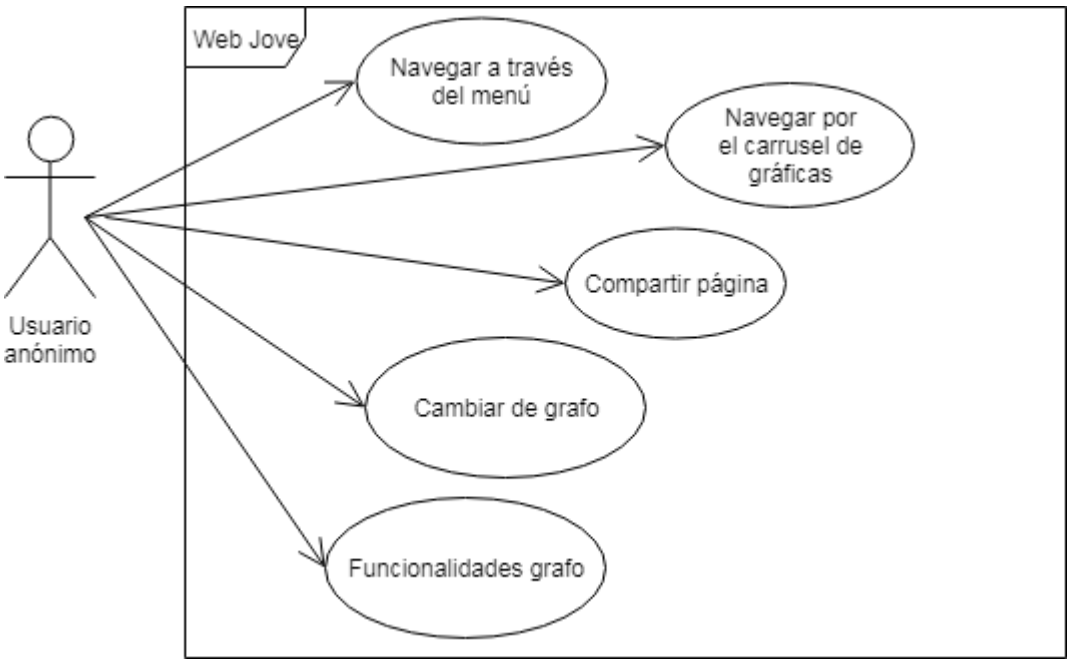

Ilustración 20. Esquema resumido de los casos de uso.

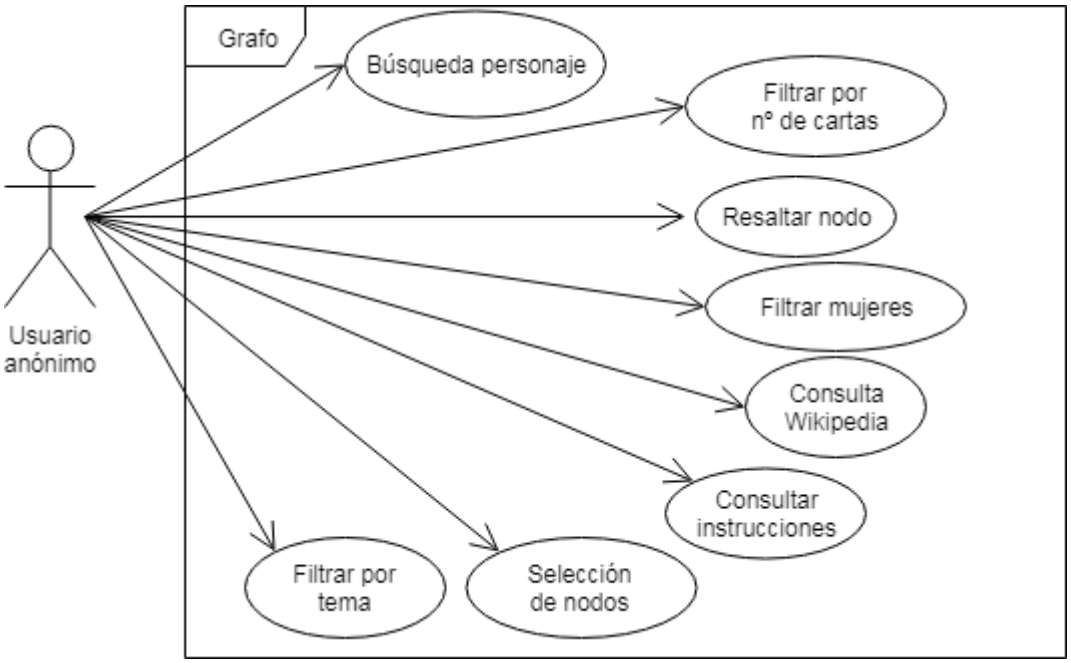

Ilustración 21. Esquema de los casos de uso del grafo.

A continuación, se van a describir estos casos de uso individualmente.

Nombre del Caso de Uso
Navegar a través del menú
Descripción
Una vez cargada la página, el usuario cambiará de sección mediante la selección de una de las opciones disponibles en el menú.

Tabla 13. Caso de uso de navegar a través del menú.

Nombre del Caso de Uso
Navegar por el carrusel
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de gráficas y utilizará el carrusel para cambiar la gráfica que se está visualizando.

Tabla 14. Caso de uso del carrusel.

Nombre del Caso de Uso
Cambiar de grafo
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de los grafos y utilizará las pestañas para cambiar el grafo que se está visualizando.

Tabla 15. Caso de uso de cambio de grafo.

Nombre del Caso de Uso
Compartir página
Descripción
Una vez cargada la página, el usuario navegará hasta el pie de la misma y utilizará uno de los botones de redes sociales para compartir el enlace de la página.

Tabla 16. Caso de uso de compartir en RRSS.

Nombre del Caso de Uso
Consulta de instrucciones
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de los grafos y consultará las instrucciones mediante el botón designado para ello.

Tabla 17. Caso de uso de consulta de instrucciones.

Nombre del Caso de Uso
Búsqueda de personaje
Descripción
Una vez cargada la página, el usuario navegará hasta la barra de búsqueda e introducirá el nombre del personaje que busca.

Tabla 18. Caso de uso de búsqueda de personajes.

Nombre del Caso de Uso
Filtrar por número de cartas
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de grafos y, mediante el deslizador, filtrará los personajes por el número total de cartas intercambiadas con Jovellanos.

Tabla 19. Caso de uso de filtro por número de cartas.

Nombre del Caso de Uso
Resaltar nodo
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de grafos y, mediante el doble click en un nodo, resaltará un personaje y su conexión a Jovellanos.

Tabla 20. Caso de uso de resaltar nodo.

Nombre del Caso de Uso
Filtrar mujeres
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de grafos y, mediante el desplegable, filtrará a los personajes por su sexo.

Tabla 21. Caso de uso de filtrar por sexo.

Nombre del Caso de Uso
Consulta Wikipedia
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de grafos y, mediante el click en un personaje, hará una consulta a la Wikipedia del personaje.

Tabla 22. Caso de uso de Wikipedia.

Nombre del Caso de Uso
Filtrar por tema
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de grafos y, mediante el desplegable, filtrará a los personajes por temas de conversación.

Tabla 23. Caso de uso de filtrar por tema.

Nombre del Caso de Uso
Seleccionar nodos
Descripción
Una vez cargada la página, el usuario navegará hasta la sección de grafos y, mediante el click, seleccionará a los personajes deseados. Posteriormente, los cambiará de posición.

Tabla 24. Caso de uso de seleccionar nodos.

5.1.3 Diseño de clases

5.1.3.1 Diagrama de Clases

Aunque posteriormente se decidió un enfoque menos ortodoxo, durante la fase de análisis se definieron unas clases que siguen el siguiente esquema:

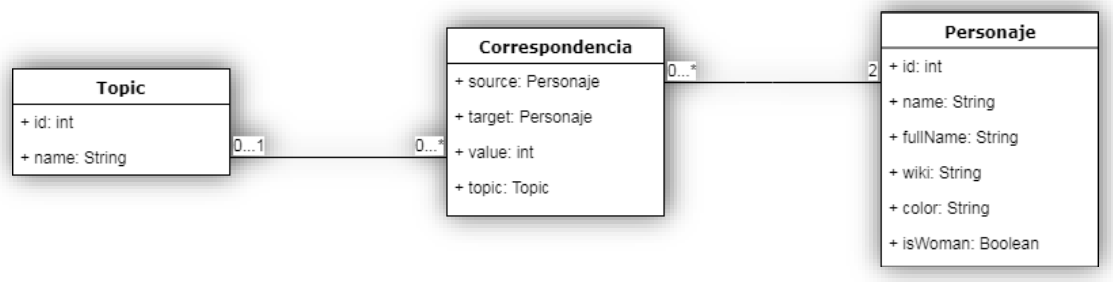

Ilustración 22. Diagrama de clases.

5.1.3.2 Descripción de las Clases

A continuación, se presentan las clases del diagrama anterior en forma de tabla:

Nombre de la Clase
Topic
Descripción
Representan las agrupaciones de cartas identificadas durante el proceso de análisis.
Atributos Propuestos
id: Identificador numérico único. **Name**: Título del tema.

Tabla 25. Clase Topic.

Nombre de la Clase
Personaje
Descripción
Representan los personajes históricos que aparecen en la correspondencia de Gaspar Melchor de Jovellanos.
Atributos Propuestos
id: Identificador numérico único. **Name**: Nombre abreviado del personaje histórico. **FullName:** Nombre completo del personaje. **Wiki: L**ink a la Wikipedia del personaje. **Color:** Cadena de texto correspondiente al nombre del color que tendrá el nodo del personaje. **isWoman:** Verdadero o falso según el personaje sea mujer o no.

Tabla 26. Clase Personaje.

Nombre de la Clase
Correspondencia
Descripción
Representa la agrupación de cartas enviadas por un personaje a otro.
Atributos Propuestos
Source: Identificador numérico único del personaje que actúa como emisor de las cartas. **Target**: Identificador numérico único del personaje que actúa como receptor de las cartas. **Value:** Número total de cartas enviadas por **Source** a **Target**. **Topic:** Cadena de texto con los títulos de los **Topic** que aparecen en estas cartas.

Tabla 27. Clase Correspondencia.

5.1.4 Diseño de la Base de Datos

5.1.4.1 Descripción de la base de datos usada

Para este sistema se ha decidido mantener el esquema del repositorio original donde se albergaba la correspondencia, pero enfocado en dos principios:

1. **Eficiencia:** los datos deben ser leídos de forma rápida, sencilla y directa por D3.js.
2. **Facilidad de uso y transparencia:** los datos deben ser leídos, entendidos y manipulables por un humano sin conocimientos técnicos de ningún tipo (incluido el uso de sistemas gestores de bases de datos).

Para ello se ha creado una base de datos documental propia basada en documentos JSON dado que D3.js posee una función para leer archivos en este formato directamente y, al ser texto plano, resultan legibles y fáciles de manejar una vez se comprende su estructura. En total se han utilizado cuatro documentos:

- **Enviados:** incluye dos *arrays*, los cuales incluyen las personas que recibieron cartas de Jovellanos (*nodes*) y los datos sobre la correspondencia enviada (*links*). Los *nodes* son objetos de la clase Personaje y los *links* son objetos de la clase Correspondencia, ambas definidas anteriormente.
- **Recibidos:** igual que enviados, pero con la correspondencia recibida por Jovellanos.
- **Jovellanos:** igual que las anteriores, pero incluye tanto la correspondencia enviada como la recibida.
- **Topics:** incluye un *array* con objetos de la clase Tema, los cuales son referenciados en los *links* de los documentos anteriores.

Estos cuatro documentos están albergados en un directorio desde el que D3 lee directamente los datos necesarios para generar los diferentes grafos.

5.1.4.2 Estructura de los documentos

Los documentos **Enviados**, **Recibidos** y **Jovellanos** siguen la siguiente estructura:

```
{
  "nodes": [
    {
      "id": 1,
      "name": "G.M. Jovellanos",
      "fname": "Gaspar Melchor de Jovellanos",
      "wiki": "Gaspar_Melchor_de_Jovellanos",
      "color": "blue",
      "woman": "false"
    }, …
  ],
 "links": [
   {
      "source": 1,
      "target": 2,
      "value": 166,
      "topics": "Política, Poesía"
   }, …
 ]
}
```

Ilustración 23. Estructura de los JSON de la correspondencia.

Por último, **Topics** tiene la siguiente estructura:

```
[
    {
        "id": 1,
        "name": "Política"
    }, …
]
```

Ilustración 24. Esquema del JSON de los temas (Topics).

5.1.5 Análisis de Casos de Uso y Escenarios

En esta sección se describirán los casos de uso identificados anteriormente de forma detallada, a través de sus escenarios. Los escenarios describen las interacciones entre los usuarios y el sistema e incluyen información acerca de los objetivos, expectativas, motivaciones, acciones y reacciones que se llevan a cabo. Los escenarios deben reflejar la forma en la que un sistema se comporta y se suelen describir en lenguaje coloquial, intentando no recurrir excesivamente a tecnicismos para poder ser entendidos por el usuario final. La intención de los mismos es definir el comportamiento deseado del *software* de manera que complementen a los requisitos funcionales antes descritos. Si se está usando una metodología de desarrollo ágil, los escenarios se detallan como acciones breves del usuario.

Es también frecuente incluir escenarios que describan acciones erróneas o equivocadas que el programa debe tener en cuenta, para asegurar un comportamiento adecuado y seguro en todos los posibles casos. A la hora de incluir estos escenarios en la descripción de un sistema se deben tener en cuenta aquellos casos en los que el sistema pueda tener un posible problema de seguridad o de fiabilidad. Por ejemplo, casos en los que el usuario introduzca información errónea, se produzca algún error al hacer algún cálculo o bien temas relacionados con algunos requisitos no funcionales.

Como ya se ha dicho, los escenarios se generan a partir de los casos de uso identificados. Como mínimo debe existir un escenario primario o principal que describa el flujo normal de los eventos que transcurran en el caso de uso, es decir, lo que debe ocurrir normalmente cuando este se ejecuta. Por ejemplo, para un caso de uso "*Registro en el sistema*", la secuencia de pasos asociada al escenario principal del mismo "*El usuario se da de alta correctamente*" sería:

1. El sistema muestra la pantalla de logon.
2. El usuario introduce su nombre y clave de usuario
3. El sistema valida la información introducida
4. El usuario entra correctamente en el sistema.

Además, también existirán escenarios que describan caminos secundarios o alternativos que son variantes del principal mostrado anteriormente. Por ejemplo, para el caso de uso anterior unos posibles escenarios secundarios serían:

- **Escenario Alternativo 1**: Alta errónea porque el usuario ya existe
- **Escenario Alternativo 2**: Alta errónea porque la contraseña no cumple las especificaciones requeridas.
- **Escenario Alternativo 3**: Alta errónea porque faltan campos obligatorios en el formulario.
- etc.

Los casos de uso es frecuente que se vuelvan complejos y generen un gran nº de escenarios secundarios (pueden tener un gran nº de pasos y cada uno de ellos genera varios escenarios secundarios). Por ello, se usan diagramas de actividad o robustez para poder representar mejor esa complejidad.

Por otro lado, los escenarios alternativos no deben contemplar los errores a nivel de sistema (fichero no encontrado, error de conexión), sino que suelen incluirse en su propia sección de "Excepciones" para tratar de no repetir la misma información de errores entre escenarios. Los escenarios alternativos son más secuencias de pasos alternativas a la "normal" o “principal”, pero que luego pueden estar relacionados con la misma. Por ejemplo, el escenario alternativo anterior “*Alta errónea porque faltan campos obligatorios en el formulario*" finalizará en "*Ir al paso 1 del escenario principal*" (volver a pedir los datos de login).

Bajo esta perspectiva, el número de escenarios que caben dentro de una aplicación medianamente compleja puede ser muy elevado. Entonces, ¿dónde poner el límite? ¿Cuáles deben ser representados? El analista debe ser capaz de seleccionar aquellos escenarios que aporten información útil al diseño. Por ejemplo, en toda aplicación web cabe considerar el escenario de "fallo de conexión". Sin embargo, si no se requiere un tratamiento específico para esta circunstancia, esa información es redundante dado que más que informar, despista al diseñador. Sin embargo, si queremos que, por ejemplo, para dar de alta un cargo en una actividad de la empresa ésta deba estar abierta, y en consecuencia que esto sea controlado por el sistema en desarrollo, sí deberemos considerar el escenario "Intento de asignación de cargo a actividad cerrada" dentro del caso de uso "Asignar cargo a actividad".

Una vez vista una pequeña descripción de los conceptos involucrados en esta sección, pasaremos a describir qué elementos deberían aparecer en la documentación final para contemplar todo lo dicho. Para ello se debe describir la secuencia de pasos de la que constan los escenarios y sus alternativas, clasificándolos según el caso de uso al que correspondan de una forma similar a la que se muestra en el ejemplo siguiente. Debe además tenerse en cuenta que no hay una plantilla estándar para describir los casos de uso y sus escenarios en una documentación, sino que es frecuente adaptar su descripción al proyecto que se está describiendo. A continuación se da un ejemplo de tabla para la descripción de los mismos, que puede adaptarse reduciéndose o ampliándose en función de las necesidades:

Nombre del caso de uso	
Precondiciones	Descripción de todas las condiciones que deben cumplirse para iniciar el caso de uso. Esto quiere decir que si el sistema no está en estado descrito por sus precondiciones, el comportamiento del caso de uso no está determinado.
Poscondiciones	Describe que cambios en el estado del software se producirán tras completar el caso de uso
Actores	Qué actores están involucrados en el caso de uso (quién lo inicia, quién lo termina)
Descripción	Se usará para capturar la esencia del caso de uso (su escenario principal), describiendo el contenido del mismo y sus operaciones
Variaciones (escenarios secundarios)	Aquí deben describirse todas las posibles variaciones contempladas sobre el escenario principal, es decir, la descripción de todos los escenarios secundarios identificados
Excepciones	Condiciones excepcionales o errores que puedan ocurrir en el escenario principal y/o los secundarios descritos antes
Notas	Cualquier aclaración necesaria que no se haya contemplado en los puntos anteriores

Para la documentación, los casos de uso complejos o de importancia elevada es mejor documentarlos con un diagrama de actividad o robustez además del texto que hagamos siguiendo la tabla anterior. De esta forma, se debería dividir la sección por cada caso de uso, y dentro de cada una de **ellas incluir primero un diagrama con la secuencia de pasos que contempla el mismo seguido de tablas como la mostrada antes, una por cada caso de uso**. A continuación se muestran unos ejemplos de esto para aclarar el contenido de esta sección.

--

En este apartado se van a analizar los casos de uso identificados en el apartado 5.1.2.3 de forma más extensa. Los casos identificados fueron los siguientes:

- Navegar a través del menú.
- Navegar por el carrusel.
- Compartir la página en RRSS.
- Cambiar el grafo visualizado.
- Selección de nodos.
- Resaltar nodos.
- Búsqueda de personajes.
- Filtrar por sexo.
- Filtrar por temas.
- Filtrar por número de cartas.
- Consulta Wikipedia.
- Consultar instrucciones.

5.1.5.1 Caso de Uso 1

Navegar a través del menú	
Precondiciones	La página debe estar completamente cargada.
Postcondiciones	La página estará centrada en la sección elegida por el usuario.
Actores	Usuario anónimo.
Descripción	El usuario: 1. Accederá a la página web. 2. Elegirá una sección en el menú de la misma. 3. El sistema moverá el enfoque de la página hasta quedar en la sección elegida por el usuario.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El usuario se encuentra exactamente sobre la sección que selecciona en el menú: ○ La página no cambiará el enfoque.
Excepciones	
Notas	

Tabla 28. Caso de uso navegación (extendido).

5.1.5.2 Caso de Uso 2

Navegación por el carrusel	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección del carrusel.
Postcondiciones	La imagen visible por el usuario cambiará.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección del carrusel. 3. El usuario selecciona cambiar la imagen visible. 4. El sistema cambia a la siguiente imagen.
Variaciones (escenarios secundarios)	
Excepciones	• **El sistema no dispone de más de una imagen:** al no disponer de más imágenes el sistema no podrá cambiar la que se está visualizando.
Notas	Cuando se haya navegado hasta la última imagen, si el usuario cambia la siguiente, el sistema pasará a la primera imagen.

Tabla 29. Caso de uso del carrusel (extendido).

5.1.5.3 Caso de Uso 3

Compartir en RRSS	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección el pie de página.
Postcondiciones	El sistema abrirá una pestaña con el mensaje preparado para compartir en la red social correspondiente.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la el pie de la página. 3. El usuario selecciona la red social deseada. 4. El sistema abre una pestaña nueva con un mensaje preparado para compartir en la red social elegida.
Variaciones (escenarios secundarios)	
Excepciones	• **El sistema no ha sido actualizado tras la actualiazación de la API de la red social:** al no haber actualizado los métodos para que funcionen con la nueva API de la red social, la pestaña abierta por el sistema contendrá un mensaje de error.
Notas	

Tabla 30. Caso de uso de compartir en RRSS (extendido).

5.1.5.4 Caso de Uso 4

Cambiar el grafo visualizado	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema mostrará el grafo seleccionado.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección de grafos. 3. El usuario selecciona la pestaña del grafo que quiere visualizar. 4. El sistema carga el grafo seleccionado.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El usuario selecciona el grafo que ya se está visualizando: ○ El sistema no hace nada.
Excepciones	
Notas	

Tabla 31. Caso de uso cambio de grafo (extendido).

5.1.5.5 Caso de Uso 5

Selección de nodos	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema marcará los nodos como seleccionados.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección de grafos. 3. El usuario selecciona uno o varios nodos. 4. El sistema marca los nodos como seleccionados. 5. Adicionalmente, el usuario podrá mover los nodos por el grafo mientras estén seleccionados. 6. Cuando el usuario suelte los nodos que está arrastrando, el sistema los devolverá a una posición estable.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** Si el usuario selecciona otros nodos que no estén seleccionados, quitará la selección de los originales. • **Escenario Alternativo 2:** Si el usuario utiliza la opción de añadir más nodos a la selección cuando realiza la operación descrita en el escenario alternativo anterior, solo se añadirán esos nodos a la selección (no se deseleccionarán los originales).
Excepciones	
Notas	Hay dos modos de selección: simple y múltiple.

Tabla 32. Caso de uso de selección de nodos (extendido).

5.1.5.6 Caso de Uso 6

Resaltar nodos	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema resaltará el nodo seleccionado y sus vecinos.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección de grafos. 3. El usuario elige el nodo que quiere resaltar mediante el doble click. 4. El sistema resalta el nodo seleccionado.
Variaciones (escenarios secundarios)	
Excepciones	
Notas	

Tabla 33. Caso de uso de resaltar nodos (extendido).

5.1.5.7 Caso de Uso 7

Búsqueda de personajes	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema resaltará el nodo correspondiente al personaje seleccionado.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la sección de grafos de la web. 2. El usuario selecciona la barra de búsqueda e introduce el nombre completo del personaje de la correspondencia que quiere encontrar. 3. El sistema resaltará el nodo correspondiente durante un tiempo determinado.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El usuario no introduce el nombre completo: ○ El sistema ofrecerá una lista de nombres que contienen parte del texto introducido por el usuario, donde él podrá escoger el nombre que buscaba sin tener que escribirlo.
Excepciones	• **El texto introducido no tiene relación con la lista de nombres disponible:** en este caso, el sistema no podrá ofrecer ninguna lista.
Notas	El sistema solo actúa cuando el valor introducido corresponde con uno de los nombres disponibles.

Tabla 34. Caso de uso de búsqueda de personajes (extendido).

5.1.5.8 Caso de Uso 8

Filtrar por sexo	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema carga un grafo filtrado por sexo.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección de grafos. 3. El usuario selecciona la opción de filtrar por solo mujeres. 4. El sistema carga un grafo con solo mujeres. 5. Si el usuario vuelve a seleccionar esa opción, el sistema cargará el grafo completo otra vez.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** si en el grafo no apareciese ninguna mujer, el sistema mostraría una alerta al usuario avisándole de lo ocurrido.
Excepciones	
Notas	Todos los filtros son acumulables.

Tabla 35. Caso de uso del filtro por sexo (extendido).

5.1.5.9 Caso de Uso 9

Filtrar por número de cartas	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema carga un grafo filtrado por número de cartas.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección de grafos. 3. El usuario selecciona un nuevo valor en el filtro por número de cartas. 4. El sistema carga un grafo con solo las relaciones entre personajes que poseen mayor cantidad de cartas que el especificado por el usuario.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** si en el grafo no apareciese ninguna relación con más cartas que el número especificado por el usuario, el sistema mostraría una alerta al usuario avisándole de lo ocurrido.
Excepciones	
Notas	Todos los filtros son acumulables.

Tabla 36. Caso de uso del filtro por cartas (extendido).

5.1.5.10 Caso de Uso 10

Filtrar por temas	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema carga un grafo filtrado por temas.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección de grafos. 3. El usuario selecciona un filtro por tema que aplicar. 4. El sistema carga un grafo con solo las relaciones entre personajes en las que se trata dicho tema.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** si en el grafo no apareciese ninguna relación con los filtros especificados por el usuario, el sistema mostraría una alerta al usuario avisándole de lo ocurrido.
Excepciones	
Notas	Todos los filtros son acumulables.

5.1.5.11 Caso de Uso 11

Consulta Wikipedia	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema carga una sección con información obtenida de la Wikipedia.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección de grafos. 3. El usuario realiza un único click en un nodo. 4. El sistema carga una sección con el resumen de la Wikipedia del personaje y la imagen asociada.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** si la página de Wikipedia del personaje no contiene una imagen, el sistema mostrará solo el texto. • **Escenario Alternativo 2:** si ya se ha realizado una consulta sobre otro personaje, la nueva consulta sobrescribirá la información previa. • **Escenario Alternativo 3:** si la última consulta realizada es sobre el mismo personaje, el sistema no hará nada dado que la información ya está siendo visualizada.
Excepciones	• **El personaje elegido no tiene Wikipedia en castellano:** si el personaje seleccionado por el usuario no tiene una página propia en Wikipedia, el sistema no hará nada.
Notas	La sección con la información de la consulta será cargada debajo del grafo.

Tabla 37. Caso de uso de consulta a Wikipedia (extendido).

5.1.5.12 Caso de Uso 12

Consulta de las instrucciones	
Precondiciones	La página debe estar completamente cargada y el usuario debe haber navegado hasta la sección de grafos.
Postcondiciones	El sistema mostrará un modal con la información requerida.
Actores	Usuario anónimo.
Descripción	1. El usuario accede a la web y esta carga completamente. 2. El usuario navega hasta la sección de grafos. 3. El usuario selecciona la opción de instrucciones. 4. El sistema carga modal con las instrucciones para utilizar el grafo.
Variaciones (escenarios secundarios)	
Excepciones	
Notas	

Tabla 38. Caso de uso de consulta de las instrucciones (extendido).

Podemos encontrar más información en: http://en.wikipedia.org/wiki/Scenario_(computing).

Por último, dado que los diagramas de robustez que se pueden usar para la descripción de la secuencia de pasos a dar en un escenario son un concepto relativamente nuevo y poco conocido, se ha incluido el siguiente anexo para su mejor comprensión.

Anexo: Diagramas de Robustez

El propósito principal de estos diagramas es poder analizar los pasos de cada caso de uso para validar que la lógica de negocio que modela es correcta, y que la terminología usada es consistente con la de otros casos de uso que se han analizado previamente. Por tanto, pueden usarse para comprobar si nuestros casos de uso son lo suficientemente robustos para representar a los requisitos del sistema construido. Otra ventaja de hacerlos es que permite identificar objetos o responsabilidades de objetos que son necesarias para soportar la lógica modelada por cada caso de uso, pero que se llaman fuera del mismo, sirviendo como puente hacia otros diagramas como diagramas de secuencia o diagramas de clase.

En un diagrama de robustez aparecen los siguientes conceptos:

- **Actores**: Los identificados en la sesión correspondiente.
- **Elementos limítrofes o *"Boundary Elements"*:** Representan elementos software como pantallas, informes, páginas HTML o interfaces del sistema con los que cada actor interactúa. También se les denomina "Elementos de Interface"
- **Elementos de Control o *"Control Elements"*:** Sirven como unión entre elementos *"Boundary"* y las entidades que veremos a continuación, implementando la lógica necesaria para gestionar los elementos y sus interacciones. También se les suele

denominar elementos de proceso o controladores. Es importante entender que es posible implementar controladores dentro del sistema con elementos que no sean objetos (si son muy simples, pueden representarse con métodos de una entidad o clase *"Boundary"* simplemente).

- **Entidades o *"Entity Elements"*:** Estos son los tipos de entidad que se encuentran en el modelo conceptual ("Estudiante", "Aula", etc.).
- **Otros casos de uso (opcional):** Dado que en muchas ocasiones unos casos de uso invocan a otros, puede ser necesario representar esto en los diagramas de robustez.

El uso de estos diagramas será el correcto si vemos que nos reportan las siguientes ventajas:

- **Control de corrección:** Ayudan a estar seguro de que la descripción de cada caso de uso y sus escenarios es correcta y que no describe comportamientos no razonables o imposibles. En ocasiones puede ser necesario cambiar los nombres usados en la descripción del caso de uso con los nombres que aparecen en los diagramas de robustez.
- **Control de completitud:** Ayuda a asegurarnos que los casos de uso encajan con las operaciones que pretendemos hacer.
- **Identificación de objetos:** Es posible que nos hayamos olvidado de identificar algunos objetos en las partes del análisis hechas anteriormente y estos diagramas nos ayudarán a saberlo. También pueden ayudarnos a identificar discrepancias y conflictos entre nombres que hayamos asignado a las entidades anteriormente. En caso de encontrar estos fallos, debemos modificar los diagramas de manera acorde.
- **Ayuda a una fase preliminar del diseño:** Estos diagramas suelen ser más sencillos y fáciles de leer que los de secuencia.

Para una realización correcta de estos diagramas debemos tener en cuenta:

- Que las entidades que representemos en este diagrama DEBEN aparecer en el diagrama de clases de entidades del análisis hecho anteriormente.
- Que los diagramas tienen que describir los procesos de los casos de uso/escenarios que hayamos identificado. En caso de encontrar discrepancias, se debe identificar cual es el error y arreglarlo en aquel diagrama que corresponda.

Podemos encontrar más información en:

- http://www.agilemodeling.com/artifacts/robustnessDiagram.htm
- http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/SoftwareEngineering/UCDOM_robustness.html (Especialmente recomendado si este tipo de diagramas no se ha visto anteriormente, detallando además posibles errores a la hora de construir estos diagramas)

Por último, debemos recordar que si los escenarios representan operaciones muy sencillas o triviales no es necesario hacer un diagrama para los mismos. Tampoco tiene mucho sentido desarrollar muchos diagramas casi iguales; si varias operaciones funcionan prácticamente de la misma forma, bastaría con indicar que el diagrama correspondiente a la operación X es muy similar al mostrado para la operación Y, indicando en texto las diferencias. Es también importante tener herramientas que nos permitan realizar fácilmente estos diagramas.

Actualmente herramientas como *DIA* (instalable mediante el "centro de *software"* de versiones recientes de Ubuntu) o *Enterprise Architect* son ejemplos de herramientas que lo permiten.

5.1.6 Análisis de Interfaces de Usuario

A la hora de diseñar un interfaz de usuario, debemos cumplir con las normas de comunicación persona-máquina existentes, procurando que el interfaz sea usable, permita manejar el programa de manera eficiente y que no sea propenso a provocar errores en los usuarios. Esto debe hacerse así porque los diseños obedecen al resultado de hacer un diseñado centrado en el usuario, que simplemente nos lleva a simular en la pantalla el trabajo que realiza sobre una mesa o, en general, su entorno de trabajo existente hasta el momento. Un enlace que puede ser de ayuda a la hora de tomar determinadas decisiones a la hora de construir el interface de la aplicación es el siguiente: http://www.ambysoft.com/essays/userInterfaceDesign.html.

5.1.6.1 Descripción de la Interfaz

En esta sección debemos crear la especificación de las interfaces entre el usuario y el sistema a construir, incluyendo todos los diferentes tipos de pantallas que van a existir, los cuadros de diálogo o los informes que le proporcionarán al usuario.

En este apartado también es importante identificar posibles grupos de usuarios para así aplicar las pantallas a dichos grupos, así como detallar otros aspectos, como lo que vamos a incluir en las pantallas para cumplir con normas de accesibilidad y usabilidad.

Para los distintos tipos de pantallas, se debe hacer un esquema que muestre la disposición de las mismas, que permita identificar donde irá cada elemento y las diferentes zonas de trabajo. Se muestra un ejemplo con este dibujo:

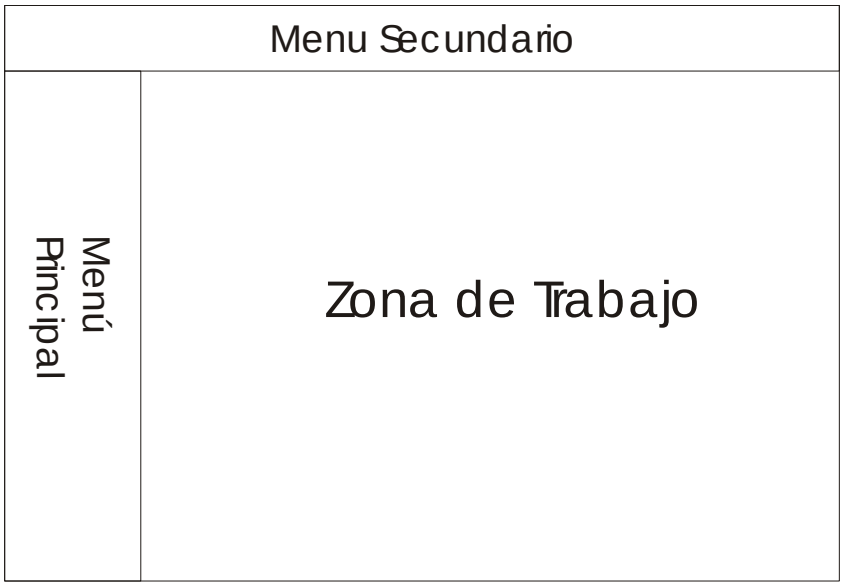

Figura 5.6. Boceto de una interfaz

Otra posible opción para este apartado es diseñar ya las pantallas definitivas sin funcionalidad, solo para ver como quedarán en el producto final (es decir, crear un **prototipo**), lo que tiene la ventaja de poder enseñarle al cliente el aspecto de la aplicación desde un primer momento.

El prototipo de interfaz que se planteó para esta web de una sola página está dividido en cuatro secciones claras: introducción, grafos, la zona de gráficas y el "acerca de". Como se verá más adelante, está estructura se ha mantenido prácticamente intacta en el diseño final.

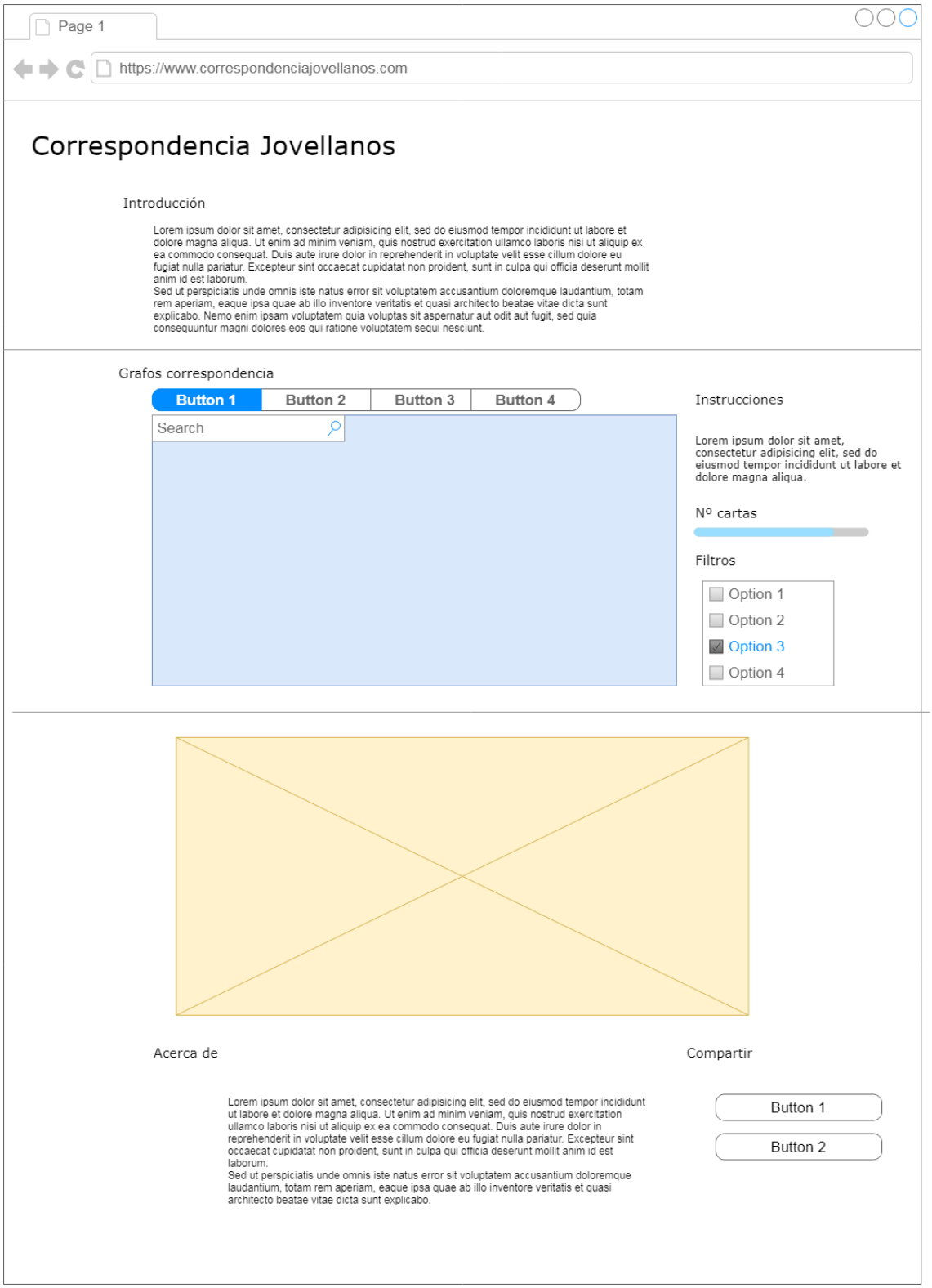

Ilustración 25. Prototipo de pantalla de la página web.

5.1.6.2 Descripción del Comportamiento de la Interfaz

Debido a la naturaleza de esta página, cuyo objetivo es únicamente albergar las visualizaciones de la correspondencia y el análisis realizado durante la primera parte del proyecto, las únicas entradas de datos por parte del usuario son las siguientes:

- **Barra de búsqueda de personajes:** este campo de tipo texto permite al usuario introducir el nombre de un personaje al que se desea buscar en el grafo. Se ha decidido acotar el texto posible a nombres existentes en la correspondencia, por lo que no es necesario ningún mensaje de error ya que no podrán buscarse personajes inexistentes.
- **Filtro número de cartas:** al ser un deslizador y no un campo donde el usuario pueda introducir cualquier valor libremente, no hace falta un mensaje de error ya que solo se podrá introducir valores preestablecidos.
- **Otros filtros:** como los filtros por sexo y tema son acumulativos, se ha tenido en cuenta la posibilidad de que no existan personajes con cierta combinación de filtros. Cuando esto ocurra, se avisará al usuario por medio de una alerta.

5.1.6.3 Diagrama de Navegabilidad

Dado que se ha diseñado el sistema como una web de una sola página, toda la navegación se realiza en la misma pantalla, con la excepción de dos diálogos con el usuario (instrucciones y la alerta mencionada previamente). Debido a esto, no se ha considerado útil la creación de un diagrama de navegabilidad.

5.1.6.5 Diseño de la Interfaz

El diseño final de la interfaz no ha sufrido demasiados cambios respecto al prototipo original como se puede observar en la página siguiente. La lista de diferentes elementos por secciones se presenta a continuación:

- **Sección superior:**
  - **Menú de navegación:** barra de navegación desde la cual el usuario podrá navegar directamente a los distintos apartados de la página. Además, esta barra es fija y es accesible en todo momento.
  - **Introducción:** pequeña introducción sobre el contenido de la página, incluye una imagen de Gaspar Melchor de Jovellanos.
- **Sección grafos:**
  - Incluye los filtros y la barra de búsqueda que ya se explicaron en el apartado anterior.
  - **Leyenda:** una leyenda siempre visible que describe la relación entre colores y número de cartas.
  - **Pestañas cambio de grafo:** identificadas como "Enviadas", "Recibidas" y "Completo", cargan el grafo correspondiente cuando el usuario las selecciona.
  - **Instrucciones:** botón que despliega un diálogo que describe, una a una, las funcionalidades del grafo y como utilizarlas.
  - **Sección Wikipedia:** sección oculta que, al realizar una consulta a Wikipedia, se hace visible con la imagen y el texto extraídos de la página del personaje seleccionado.
- **Sección gráficas:** carrusel de imágenes con distintas gráficas relacionadas con el análisis de la correspondencia.
- **Sección inferior:**
  - **Acerca de:** pequeño texto explicando cómo se ha realizado este proyecto.
  - **Pie de página:** incluye los botones para compartir la web en redes sociales, para ver el código en el sistema de control de versiones utilizado y un e-mail de contacto.

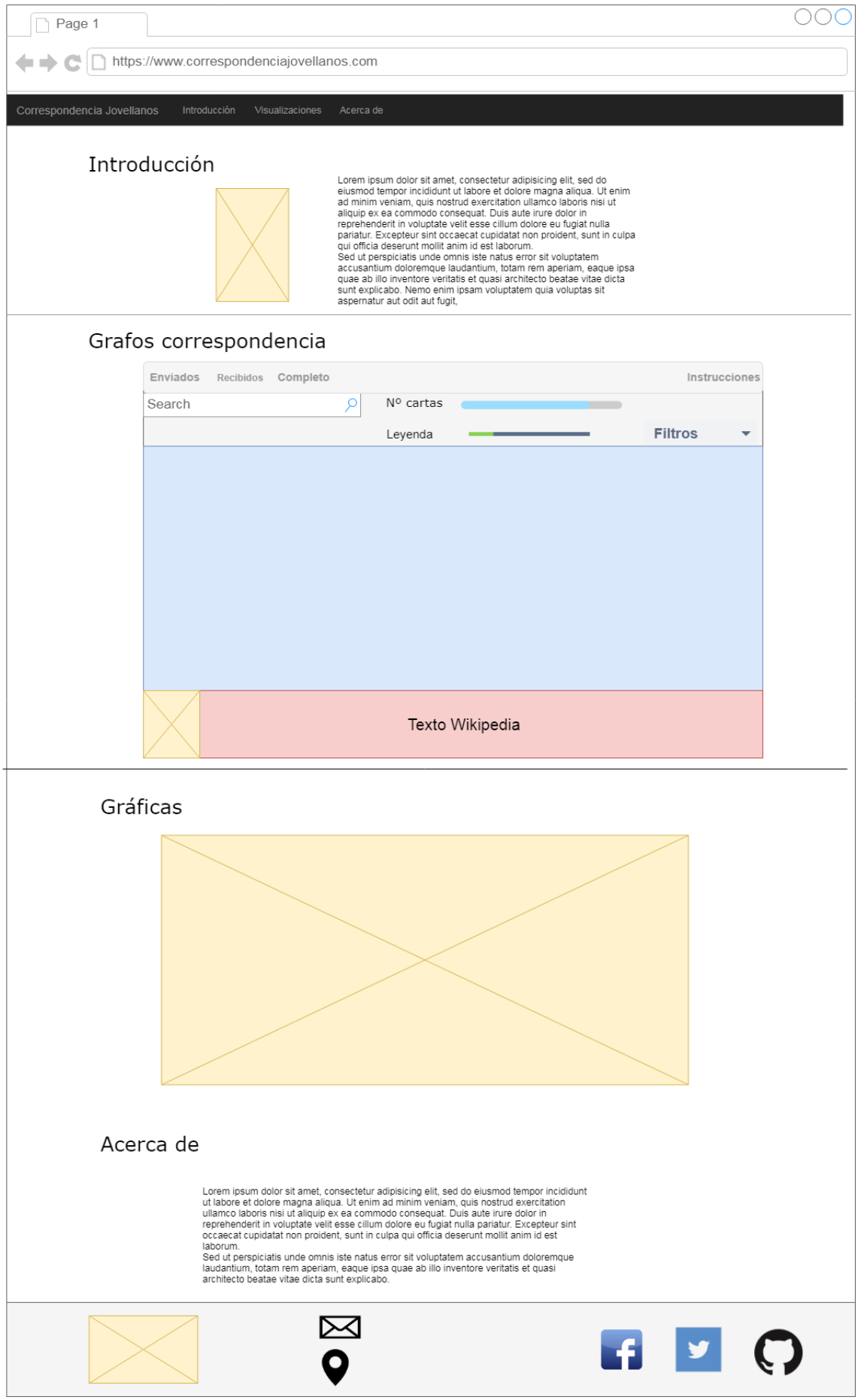

Ilustración 26. Diseño final de la interfaz.

5.1.7 Especificación del Plan de Pruebas

En esta sección crearemos y diseñaremos el plan de pruebas de la aplicación y sus funciones, así como todos los mecanismos que utilizaremos para detectar errores y corregirlos ya en la fase de implementación.

Las pruebas contemplarán aspectos tanto de funcionalidad de la aplicación como de aspectos de los usuarios o clientes de la misma.

Se contemplarán hasta cinco tipos de pruebas:

- **Pruebas Unitarias**: Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código, o en este caso una clase individual que cumple con una función concreta. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. A partir de los casos de uso, los escenarios y clases vistos anteriormente, debemos desarrollar pruebas unitarias que consideremos necesarias y especificar los resultados que se espera encontrar una vez ejecutada la operación sobre cada una de ellas. Es conveniente tabular estas pruebas para su aplicación posterior.
- **Pruebas de Integración**: Las pruebas de integración comprenden verificaciones asociadas a grupos de componentes, verificando que éstos funcionan correctamente cuando estos son ensamblados para cumplir con una función concreta. Para ello, cada escenario debe probarse con el mayor número de entradas posibles (y relevantes) que sea posible, incluyéndose entradas con datos correctos y con datos incorrectos para probar que el sistema reacciona correctamente ante errores de los usuarios. Para elaborar estas pruebas debemos tener en cuenta las características de la aplicación.
- **Pruebas del sistema**: Las pruebas del sistema son pruebas de integración del sistema construido completo, que permiten probar el conjunto de todo el sistema y que sus relaciones con otros sistemas que necesite son correctas, verificando así que todas sus especificaciones funcionales y técnicas se cumplen.
- **Pruebas de Usabilidad**: Este tipo de pruebas determinan la satisfacción del cliente con el producto final. Podemos especificar aquí cuales de estos aspectos son los más importantes en la aplicación a crear y establecer unas pautas generales por las cuales queremos medir en qué medida hemos conseguido estos aspectos. No se trata de diseñar ya los mecanismos de prueba de estos aspectos, ya que eso se hará posteriormente.
- **Pruebas de Código**: Para determinar la existencia de código muerto. Se recomienda hablar si incluir este tipo de pruebas o no con el director del proyecto.

--

Debido a la naturaleza del sistema, se ha decidido que solo se realizarán pruebas de usabilidad. Esto es debido a que no existen componentes que integrar y que la principal funcionalidad a probar, el grafo, se prueba de forma más completa trabajando directamente con el usuario, ya que, el principal objetivo de esta parte del proyecto es presentar los datos de la correspondencia y el análisis de forma clara, accesible y simple. Adicionalmente, se ha utilizado Codacy como sustituto de las pruebas de código por su capacidad de detectar errores, posibles errores, código repetido, código sin usar, etc…

Las pruebas han sido elaboradas a partir de los casos de uso y escenarios antes descritos, teniendo en cuenta el escenario principal del caso de uso y sus posibles alternativas y excepciones.

Caso de Uso 1: Añadir Usuario	
Prueba	**Resultado Esperado**
Añadir un usuario no existente	El sistema posee un usuario más
Prueba	**Resultado Esperado**
Añadir un usuario que ya existe	El sistema no posee un usuario más y se muestra un dialogo notificándolo
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

5.2 Diseño del Sistema

5.2.1 Arquitectura del Sistema

5.2.1.1 Diagramas de Paquetes

Aunque el concepto de paquete se puede aplicar a varios elementos del modelo, lo más típico en un PFC es agrupar clases con ellos. Los paquetes agruparán clases que estén relacionadas con una determinada funcionalidad y este diagrama debe mostrar también las relaciones existentes entre dichos paquetes (por ejemplo, dos paquetes estarán relacionados si algunas de sus clases se usan entre ellas o se envían mensajes). En esencia se trata de mostrar la organización lógica de la aplicación (en contraposición con la organización física, que aparecerá en los diagramas posteriores), presentando como se agrupan las clases de dicha aplicación en paquetes y la relación existente entre los mismos. Más información sobre estos diagramas se puede encontrar en:

http://www.agilemodeling.com/artifacts/packageDiagram.htm

http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_packagediagram.html

Los diagramas de paquetes son especialmente útiles cuando el diagrama de clases de todo el sistema es demasiado grande para visualizarse correctamente. Podemos entonces hacer un diagrama de paquetes (cada paquete contendrá N clases del sistema) y luego detallar individualmente cada uno de ellos donde corresponda.

5.2.1.1.1 Paquete 1

Descripción de los paquetes del diagrama (que tipo de elementos contiene, para qué sirven los mismos,…).

5.2.1.1.2 Paquete 2

5.2.1.2 Diagramas de Componentes

Los diagramas de componentes muestran los diferentes componentes de un sistema y sus dependencias. Un componente representa un módulo de código físico. Muchas veces se suele identificar un componente con un paquete, pero no siempre es así ya que los componentes representan el empaquetado físico del código. Por tanto, una misma clase puede estar presente en varios componentes, pero definida en un único paquete.

Es también una práctica común el incluir estos diagramas dentro de los de despliegue (que veremos a continuación) para que la distribución física de las distintas partes de la aplicación (y los componentes que las forman) en las distintas máquinas disponibles quede más clara.

Para tener más información sobre este tema, se pueden consultar los siguientes enlaces:

http://www.agilemodeling.com/artifacts/componentDiagram.htm

http://www.visual-paradigm.com/VPGallery/diagrams/Component.html

http://www.ibm.com/developerworks/rational/library/dec04/bell/

Si se integra con el diagrama siguiente (el de despliegue), en las descripciones que hagamos de los elementos debemos describir también los componentes. En caso contrario, describiremos los componentes en esta sección.

5.2.1.3 Diagramas de Despliegue

El sistema creado puede estar compuesto por varios procesos *software* y máquinas que colaboran para llevar a cabo la tarea encomendada, y esta sección es la indicada para representar estos procesos y máquinas así como la relación existente entre ellos. Debemos ofrecer un diagrama (un ejemplo extremadamente simple de cómo podemos hacerlo se muestra a continuación) y posteriormente una descripción de que es cada parte de la aplicación y su función (relacionándolo con las clases y paquetes vistos en el análisis).

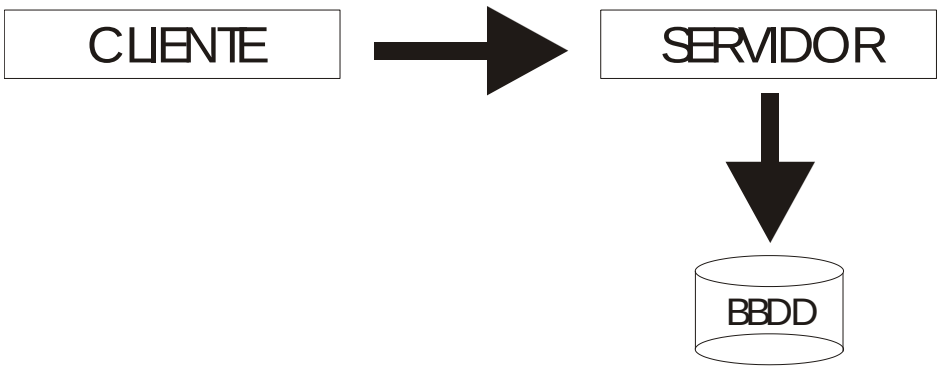

Figura 6.1. Ejemplo simple de arquitectura del sistema

Aunque el nivel de detalle puede variar en función del sistema diseñado, los elementos mostrados deben dejar clara la distribución de la aplicación. Podemos encontrar más información al respecto en los siguientes enlaces:

http://www.agilemodeling.com/artifacts/deploymentDiagram.htm

http://www.visual-paradigm.com/VPGallery/diagrams/Deployment.html

5.2.1.3.1 Elemento 1

Descripción de los elementos del diagrama anterior.

5.2.2 Diseño de Clases

En esta sección representaremos diagramas que muestren los paquetes (y la relación existente entre ellos) y las clases que formarán parte de la implementación final del sistema (incluyendo también las relaciones existentes entre ellas), explicando todo aquel dato acerca de la utilidad de los mismos y justificando todo aquello que consideremos necesario.

- Debemos extraer la información de métodos y atributos a incluir en cada clase (así como las relaciones existentes entre ellas) de los diagramas que hemos desarrollado en el punto anterior.
- También debemos mostrar las clases adicionales que pueden ser necesarias en función de lo que hemos desarrollado anteriormente.
- Debemos mostrar todas las asociaciones y agregaciones (en general, relaciones) que necesitemos, en función de dichos diagramas.
- El conjunto de atributos de las clases debemos crearlo según los métodos, relaciones y en general de las necesidades de cada clase del sistema.
- La jerarquía de clases debemos pensarla de acuerdo a las necesidades de cada subsistema y aplicación.
- Debemos aclarar todos aquellos aspectos que no queden claros en el diagrama, usando notas.
- En general, debemos tener en cuenta que el diseño es una evolución del análisis, por lo que las clases que incluyamos en el diseño deben corresponderse razonablemente con todo el trabajo hecho en esa parte del documento.

5.2.2.1 Diagrama de Clases

Es necesario mostrar por lo menos un diagrama global de clases. Si el diagrama de clases global fuese muy grande, podemos dividir el diagrama en varios diagramas más pequeños según la estructura de la aplicación, mostrando el principal abreviado tal y como se explico anteriormente en los diagramas de paquetes. Otra opción es hacer que la hoja que tenga el diagrama se imprima en otro formato que nos proporcione más espacio (en horizontal, A3,…).

5.2.3 Diagramas de Interacción y Estados

Esta sección se usará, entre otras cosas, para evolucionar y detallar los diagramas de robustez que hemos desarrollado en el análisis usando diagramas de interacción y de estados. La estructura a seguir es la de incluir el diagrama en sí (dibujo) y luego hacer una lista explicando cada uno de los pasos existentes en dicho diagrama que lo requieran. Los diagramas deben incluir nombres de clases, métodos y parámetros "reales", con la intención de que puedan trasladarse directamente a la implementación del sistema (el objetivo de un buen diseño). Por este motivo no debemos escatimar en detalles a la hora de desarrollarlos, ya que se supone que de estos diagramas podremos extraer directamente la implementación de la aplicación.

Cuando se menciona que los diagramas del diseño son una evolución de los del análisis significa que podemos aprovechar el trabajo ya realizado para su creación, adaptando la información en los diagramas correspondientes del análisis para desarrollar los nuevos diagramas (se debe lograr coherencia entre las entidades desarrolladas en ambas fases).

En cuanto al número de diagramas a contemplar, al igual que se dijo anteriormente no es necesario incluir en el diseño todos y cada uno de los posibles diagramas, sino que podremos ahorrar extender aquellos en los que:

- Las operaciones sean muy simples (en cuyo caso se puede optar por no decir nada al respecto o colocar simplemente un texto explicativo).
- La operación sea muy similar o idéntica a una que ya está desarrollada en esta sección. En este caso lo mejor es poner en el diagrama original a que otros casos se aplica o representa y no repetirlo.

5.2.3.1 Caso de Uso 1.1

5.2.3.1.1 Diagramas de Interacción (Comunicación y Secuencia)

Diagramas de secuencia de diseño correspondiente al escenario principal del caso de uso y a aquellos alternativos que merezcan ser desarrollados (estableciendo un paralelismo con la nomenclatura usada en el análisis). Podemos encontrar más información en:

http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/SoftwareEngineering/UCDOM_interaction.html

En el caso de que creamos necesario mostrar el comportamiento de varios objetos que colaboran para la consecución de un determinado fin común, debemos crear un diagrama de comunicación de objetos que represente este comportamiento. Podemos encontrar más información acerca de los mismos en esta dirección:

http://www.agilemodeling.com/artifacts/communicationDiagram.htm

5.2.3.1.2 Diagramas de Estados de las Clases

En caso necesario, si es conveniente dar más detalles de cómo se comporta un determinado escenario, podemos incluir diagramas de estados de las clases involucradas en el mismo para

indicar las distintas fases por las que una clase pasará durante el mismo. Podemos encontrar más información sobre estos diagramas en:

http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

5.2.3.2 Caso de Uso 1.2

5.2.4 Diagramas de Actividades

Si hay alguna operación o funcionalidad dentro del sistema que merezca la pena destacar se documentará mediante un diagrama de actividades. Puede haber tantos como consideréis necesarios. Más información acerca de estos diagramas se puede encontrar en estos enlaces:

http://dn.codegear.com/article/31863#activity-diagrams

http://www.agilemodeling.com/artifacts/activityDiagram.htm

5.2.5 Especificación Técnica del Plan de Pruebas

El proceso de pruebas se extiende durante todo el proceso de construcción del software y por tanto en esta sección debemos describir cómo vamos a aplicar las pruebas diseñadas y especificar más otro tipo de pruebas que vamos a realizar al software.

No debemos olvidarnos mencionar entonces bajo que máquina (incluyendo sistema operativo y entorno de desarrollo) se realizan las pruebas, y en qué orden se realizan las diferentes pruebas contempladas, entre otros aspectos.

NOTA: Es perfectamente posible que, aunque un sistema conste de múltiples procesos independientes que colaboran entre sí (por ejemplo un cliente y un servidor), las pruebas se hagan en sólo una máquina (Ej.: cliente y servidor que se conectan a *localhost*).

5.2.5.1 Pruebas Unitarias

Esta sección contendrá la descripción de la aplicación de las pruebas unitarias que hemos descrito anteriormente (qué datos se van a introducir finalmente sobre qué clases, ahora que conocemos finalmente todas las clases que se van a implementar gracias al diseño), indicando cuando las vamos a realizar y que medidas tomaremos en caso de encontrar fallos. También debemos hablar de si hemos dividido las pruebas entre los diferentes subsistemas de alguna forma.

En este apartado debemos también describir las posibles **pruebas de regresión** que queramos implementar. Estas pruebas de regresión se deben automatizar para poder pasarlas periódicamente cada vez que hagamos cambios importantes en el sistema.

Se recomienda encarecidamente el empleo de una herramienta de automatización de estas pruebas unitarias para facilitar mucho esta labor. Posibles herramientas son *JUnit* (Java) o *NUnit*. Realmente esta sería la forma más adecuada de afrontar esta tarea.

5.2.5.2 Pruebas de Integración y del Sistema

Las pruebas funcionales y del sistema que hemos especificado en el análisis deben aplicarse para garantizar que el sistema funciona correctamente. Debemos describir así cómo y cuándo vamos a aplicar este tipo de casos de prueba dentro del sistema, así como las entradas y salidas de estas pruebas. No debemos olvidar que la aplicación de las pruebas y qué ocurre cuando se pasan irá en una sección posterior.

Debemos recordar pues que, tanto en esta sección como en la anterior, ya tenemos las pruebas diseñadas en la fase de análisis, y que ahora se trataría de describir cómo y cuando las vamos a aplicar, qué datos vamos a introducir en cada caso y qué vamos a hacer en caso de acierto o fallo de las pruebas. No se trata de repetir lo mismo de nuevo.

A modo de guía, debemos pensar que lo que diseñemos aquí debe ser directamente aplicable sobre el código de la aplicación.

5.2.5.3 Pruebas de Usabilidad y Accesibilidad

Dado que es posible que el lector de este documento no esté familiarizado con este tipo de pruebas, haremos una descripción un poco más en detalle de en qué consisten. Este tipo de pruebas determinan la satisfacción del cliente con el producto final y por tanto debemos darle mucha importancia a las mismas, no dejándolas de lado en ningún caso.

Las pruebas de usabilidad tratan de evaluar la aplicación en su funcionamiento real. Debemos realizar un conjunto de pruebas para verificar que las distintas partes del programa se pueden usar adecuadamente. Los objetivos son:

- Mejorar la calidad de la interacción de los usuarios con nuestra aplicación.
- Reducir el tiempo necesario para hacer las distintas tareas en la aplicación y de esta forma aumentar la productividad.

Debemos tener los siguientes elementos en cuenta, para describirlos completamente en esta sección antes de hacer las pruebas en sí:

- **Usuarios**: Distintos grupos de usuarios de la aplicación sobre los que vamos a realizar las pruebas. Cada usuario tendrá un perfil de uso distinto y tenemos que tenerlo en cuenta a la hora de hacer las pruebas. Por lo general es suficiente con efectuar las pruebas con un número reducido de usuarios (de 3 a 5). También debemos dejar bien claro, en caso de que los usuarios lo deseen, la profesión y la edad de los mismos.
- **Lugar de realización**: En un laboratorio o bien en la propia casa u oficina del cliente.
- **Metodología**: Describir que vamos a hacer en estas pruebas (el conjunto de pasos a seguir en las mismas).

Aunque en la siguiente sección se dan detalles acerca de cómo hacer cuestionarios para efectuar estas pruebas, estos no son la única herramienta para ello. Existen otros tipos de herramientas como pruebas basadas en las opiniones dadas de forma oral por los usuarios, heurísticas, etc. Dependiendo fundamentalmente del producto y para quien este destinado. No obstante, los cuestionarios porque son una herramienta adecuada para gran parte de los casos y no exigen muchos recursos. Se recomienda consultar al director de proyecto acerca de la mejor forma de hacer este tipo de pruebas en el proyecto concreto a desarrollar.

5.2.5.3.1 Diseño de Cuestionarios

Detallamos a continuación algunas pautas a seguir en el diseño de los cuestionarios que podamos necesitar, si se estima oportuno el uso de esta herramienta. Una norma muy importante a cumplir es que él tiempo necesario para rellenar los cuestionarios no deben superar los 15 minutos.

5.2.5.3.1.1 Cuestionario de Evaluación

Debemos elaborar un cuestionario para que los usuarios lo hagan y determinar así distintos aspectos del mismo y de su interacción con la aplicación. Los puntos a tocar son (esto es un esquema que puede ampliarse si se desea para adaptarlo a la aplicación en sí):

- **1º**: **Preguntas de carácter general** a través de las cuales se determine el tipo de usuario y su nivel de conocimiento informático.

- **2º**: **Actividades guiadas** para hacer con nuestra aplicación.
- **3º**: **Batería de preguntas cortas** con los distintos aspectos de la aplicación que se pretendan evaluar.
- **4º**: **Observaciones**, para que el usuario aporte todo lo que considere oportuno de nuestra aplicación.

5.2.5.3.1.2 Cuestionario para el Responsable de las Pruebas

Además, debemos desarrollar un cuestionario para que el responsable de las pruebas pueda anotar distintos aspectos que observe durante la realización de las pruebas. Un posible desarrollo de todos estos aspectos se describe a continuación.

5.2.5.3.2 Actividades de las Pruebas de Usabilidad

5.2.5.3.2.1 Preguntas de carácter general

Se muestra un esbozo de un posible cuestionario, que debemos desarrollar y adaptar a nuestras necesidades:

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias

5.2.5.3.2.2 Actividades guiadas

Hacer un compendio de actividades que se puedan hacer con la aplicación para que nuestros usuarios las hagan y comenten los problemas y dificultades que según su opinión presenta la aplicación (si existiesen). Debemos recoger estas opiniones en el documento. Posibles actividades a probar son:

- Añadir un usuario a la aplicación
- Eliminar un artículo de la aplicación
- …

5.2.5.3.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Un posible cuestionario de preguntas cortas (a desarrollar más en cada proyecto) es el siguiente:

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe donde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
Observaciones				
Cualquier comentario del usuario				

Evidentemente debemos extender este cuestionario de ejemplo para adaptarlo a nuestras necesidades.

5.2.5.3.2.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	
Errores graves cometidos	
<Incluir otras cosas>	
…	
…	

Los resultados de las pruebas se deben analizar conjuntamente para así establecer una conclusión respecto a cuatro aspectos de usabilidad:

1. **Facilidad de aprendizaje**: Capacidad para aprender la funcionalidad de la aplicación desarrollada y desarrollar las tareas de manera adecuada, midiendo cuanto se tarda en hacer las distintas tareas.
2. **Eficiencia**: Cuanto mejora la labor de los usuarios por usar la aplicación respecto a lo que se hacía anteriormente.
3. **Errores**: Cuantos errores cometen los usuarios en las distintas tareas, lo que decrementa la usabilidad del mismo.
4. **Satisfacción del usuario**: Impresión general de los usuarios al usar la aplicación.

5.2.5.3.3 Pruebas de Accesibilidad

El grueso de las **pruebas de accesibilidad** será descrito en la parte de desarrollo de las pruebas. A nivel de diseño simplemente puede indicarse qué tipo de pruebas van a realizarse para el programa concreto, que normas *WCAG* van a seguirse u otras consideraciones generales acerca de las mismas, dentro de las posibles vías de actuación. *WAI* propone básicamente tres tipos de revisiones:

- **Revisión preliminar**: Identifica rápidamente problemas de accesibilidad de un sitio *Web*. Todos los procesos de esta revisión también tienen lugar en la siguiente revisión.
- **Evaluación de conformidad**: Permite indicar si un sitio web cumple con los estándares de accesibilidad (Ej.: *WCAG*). Puede encontrarse más información en: www.w3.org/WAI/eval, aunque en esta plantilla se intentarán dar pautas para hacer una evaluación adecuada.
- **Otras evaluaciones**: *WAI* también proporciona otra serie de documentos para evaluar aplicaciones en contextos específicos y para implicar a usuarios discapacitados en la evaluación. En el primer caso se tratan de documentos complementarios a los anteriores que describen consideraciones para la evaluación de sitios grandes y complejos durante el proceso de desarrollo y mantenimiento, o bien para sitios existentes cuyas páginas son generadas dinámicamente.

5.2.5.4 Pruebas de Rendimiento

El sistema desarrollado consumirá una determinada cantidad de recursos (memoria y tiempo de proceso) que debemos procurar que sean los menores posible. Por ello, la aplicación debe al menos medirse para ver cuántos recursos consume y se debe intentar eliminar posibles cuellos de botella en el rendimiento que se puedan presentar en uno o varios subsistemas de la misma. Debemos diseñar medios para hacer estas mediciones en nuestra aplicación.

Para esta tarea debemos pues medir el tiempo que tardan las operaciones más importantes que haga el sistema y comprobar si es posible mejorarlas de algún modo una vez que el sistema esté en funcionamiento. En este apartado debemos diseñar que actuaciones vamos a llevar a cabo para hacer estas pruebas y las actuaciones que vamos a llevar a cabo para mejorar el rendimiento en aquellos puntos en los que encontremos problemas. Una herramienta que podemos aplicar para este fin es la opción "*Tools – View Speed Report*" de las "*Web Developer Extensions*" de *Firefox*.

Una consideración a tener en cuenta es determinar qué actividades serán las más frecuentes y si su rendimiento es adecuado o no, ya que por esta vía conseguiremos una mejor optimización de la aplicación.

En el caso de que el programa deba tener algún requisito respecto al tiempo que debe tardar o a la memoria ocupada, debe tenerse especial cuidado en este aspecto.

Por último, también debemos diseñar pruebas de carga para determinar cómo responde el sistema con un alto número de usuarios o un gran volumen de datos. Para esta tarea puede ayudar una aplicación como *JMeter* (http://jakarta.apache.org/jmeter/).

Capítulo 6. Implementación del Sistema

6.1 Estándares y Normas Seguidos

Descripción breve de los estándares y normas que hayamos usado en nuestra aplicación a la hora de desarrollar su código y si nos hemos ocupado de validar que esos estándares se cumplan efectivamente.

En el desarrollo de este proyecto se han seguido las siguientes pautas:

- Validación (X)HTML5: el código HTML generado para el sistema se ha construido de acuerdo con las reglas marcadas para su validación para asegurar que sería interpretado de la misma forma por distintos navegadores. Para comprobar si se cumplía el estándar, se ha validado el HTML utilizando el MarkUp Validation Service ofrecido por W3C (*World Wide Web Consortium*). Adicionalmente, se ha validado satisfactoriamente el CSS (Hoja de estilo en cascada) tanto propio como el de las librerías.
- Estándar WCAG de accesibilidad 1.0 (AA): las guías de accesibilidad del contenido (WCAG) desarrolladas por W3C tienen como objetivo crear un estándar único para hacer la web más accesible a personas con discapacidad. Estas guías incluyen tanto el contenido natural (imágenes, texto, sonidos) como el código, la estructura y la presentación del mismo. En un principio, para este proyecto se puso como objetivo seguir las guías 2.0 nivel AA (aceptadas como estándar ISO/IEC), sin embargo, debido al código generado de forma automática por un par de las librerías utilizadas, no se ha podido llegar a validar completamente ese nivel. Aun así, como se han seguido dichas pautas desde el principio, ha sido posible validar el documento según las guías 1.0 nivel AA.
- Calidad de código (vía Codacy): aprovechando que el proyecto se ha albergado en el sistema de control de versiones Git ofrecido de forma gratuita por GitHub, este ha sido integrado con Codacy para mantener controlada y asegurar la calidad del código JavaScript. Se decidió utilizar esta herramienta por la facilidad para monitorizar en todo momento los posibles errores tanto de seguridad, *bugs*, código que no sigue las pautas de estilo, código sin usar, etc…

6.2 Lenguajes de Programación

Descripción de los lenguajes de programación usados, su versión y distribución, los módulos o complementos de los mismos empleados, etc.

--

A continuación, se presentan los diferentes lenguajes usados en ambas partes del proyecto.

6.2.1 Lenguajes utilizados para el análisis de lenguaje.

Para esta parte, como se mencionó anteriormente, solo se ha utilizado el lenguaje **R**. Este lenguaje diseñado especialmente para computación estadística como un proyecto GNU derivado del lenguaje S. R, ya de por sí, provee de multitud de herramientas para el análisis estadístico y la creación de gráficas, pero, además, existen infinidad de paquetes que extienden las funcionalidades del lenguaje. Los paquetes que han sido utilizados durante la realización de este trabajo son los siguientes: *NLP, TM, fastmatch, XML, stringr, cluster, fpc, dbscan, factoextra, tidytext, topicmodels, doParallel, dplyr, ggplot2, scales, RColorBrewer, wordcloud2, RTextTools y sparcl.*

Ilustración 27. Logo de R.

6.2.2 Lenguajes utilizados para la web

Para realizar la parte web del trabajo se ha utilizado JavaScript como lenguaje de programación, el lenguaje de marcado HTML, las hojas de estilo en cascada (CSS) y el formato de documentos JSON (JavaScript Object Notation).

JavaScript (última versión: ECMAScript 2016) es la piedra angular de esta parte del proyecto debido a que todas las librerías utilizadas están escritas en dicho lenguaje. JavaScript es un lenguaje interpretado de alto nivel, conocido por ser uno de los pilares de la *World Wide Web* junto a HTML y CSS. Otras características básicas de JavaScript son: tipado débil, orientado a objetos (basado en prototipos), dinámico, funcional, dirigido por eventos, …

Las librerías de JavaScript que han sido usadas son las siguientes:

- **D3.js (versión 4.13):** es la principal librería del proyecto ya que es la utilizada para generar los diferentes grafos. En general, esta librería está diseñada para tratar con documentos basados en datos y representarlos de forma dinámica o estática utilizando HTML, CSS y SVG. No solo incluye herramientas para visualización, sino que también tiene sus propios métodos para manipular el DOM (*Document Object Model*), siempre de una forma enfocada al tratamiento de datos. D3 tiene posibilidades prácticamente infinitas en cuanto a representación de datos de forma gráfica, se pueden realizar, por ejemplo: gráficos de burbujas, de barras, dendogramas, diagramas de voronoi, mapas normales y coropléticos, árboles, grafos dirigidos por fuerzas y un largo etcétera. En este proyecto se ha utilizado la versión 4.13 de enero de 2017 en vez de la más reciente (5.1) porque aún no había salido cuando comenzó el proyecto, sin embargo, 4.13 es una versión perfectamente estable y las diferencias entre ambas son poco significativas de cara a este proyecto. Aun así, se tratará en las ampliaciones la migración de esta versión a la más actual. Por último, D3.js se desarrolla bajo una licencia BSD 2, con una tercera clausula propia, por lo que es totalmente gratis y modificable si sus tres condiciones son cumplidas. Además, se ha utilizado una librería complementaria, desarrollada por el mismo creado bajo la misma licencia, para la funcionalidad de seleccionar varios nodos.

- **JQuery (versión 3.2.1):** esta librería gratuita de código abierto es la librería de JavaScript más utilizada actualmente. JQuery está enfocada a navegar y manipular HTML, manejar eventos, animaciones, etc… de una forma fácil y sencilla. Generalmente, está librería va a estar presente en cualquier proyecto web tanto de forma directa como indirecta, ya que puede ser requerida por otras librerías. JQuery se distribuye bajo una licencia MIT (*Massachusetts Institute of Technology*).

- **Bootstrap (versión 4.0):** es una librería, también de las más utilizadas, creada especialmente para diseñar y dar estilo a páginas y aplicaciones web, y, por lo tanto, contiene infinidad de plantillas HTML y CSS para botones, menús de navegación, imágenes, tipografías, formularios, tablas, modales, *tooltips*, *pop-ups*, etc... En este proyecto ha sido utilizada para dar estilo a la mayoría de componentes a excepción de los grafos. Al igual que JQuery, se distribuye bajo una licencia MIT. Además, es un ejemplo de una librería que trabaja con JQuery.

- **Bootstrap-multiselect:** esta es una librería que permite realizar menús desplegables con multiselección, de estilo similar a los dados por Bootstrap, de forma rápida. También requiere JQuery, como Bootstrap, y está licenciado bajo el mismo tipo de licencia que D3.js.

- **Bootstrap-slider (versión 10.0):** librería diseñada para crear *inputs* con forma de deslizador y un estilo similar al usado por Bootstrap. También requiere de JQuery y se encuentra bajo una licencia MIT.

Dejando atrás JavaScript, **HTML** (*Hypertext Markup Language*) es el lenguaje de marcado utilizado de forma estandarizada en la creación de aplicaciones y páginas web. La función de

este lenguaje es definir la estructura de la página web de forma que puedan ser renderizados por los diferentes navegadores web.

Por otro lado, si HTML describe la estructura, **CSS** es un lenguaje de hojas de estilo que describe la presentación de la página, diseñado originalmente para separar el contenido de la presentación.

Por último, **JSON** es un formato de archivos de texto que resultan fáciles de leer y escribir tanto para humanos como máquinas. Aunque fue diseñado originalmente para JavaScript, no depende del lenguaje y puede ser utilizado libremente. La estructura de un JSON se puede dividir en dos unidades básicas:

- Objetos: pares nombre/valor separados por comas entre llaves. Este tipo de estructura es leída por los distintos lenguajes, como JavaScript, como un objeto cuyos atributos son la primera parte de esos pares. Los valores pueden ser númericos, cadenas de texto, otros objetos, vectores, booleanos y *null*.

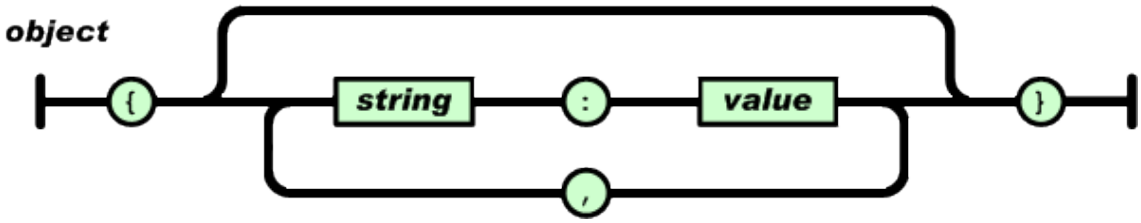

Ilustración 28. Estructura de un objeto de JavaScript en JSON.

- *Arrays*: lista de objetos separados por comas entre corchetes que los lenguajes interpretan como *arrays*.

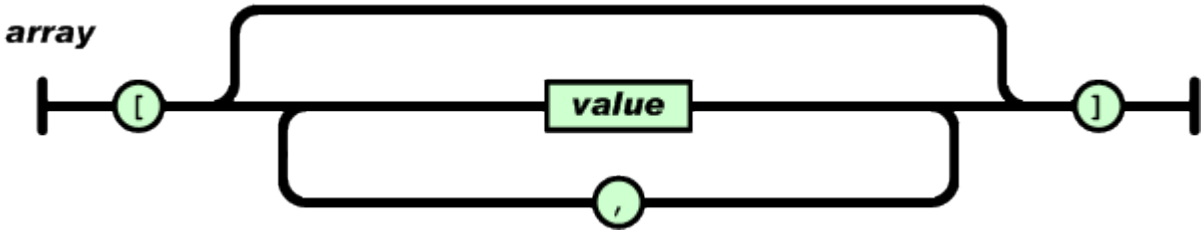

Ilustración 29. Estructura de un array en JSON.

En este proyecto se ha utilizado JSON para los documentos que albergan los datos de la correspondencia.

6.3 Herramientas y Programas Usados para el Desarrollo

A continuación, se presentan las herramientas y programas que se han utilizado durante el desarrollo del proyecto:

- **Visual Studio Code:** es un editor de código fuente adaptado para varios lenguajes desarrollado por Microsoft y distribuido de forma gratuita. Este editor tiene multitud de herramientas y características personalizables, por lo que es uno de los entornos de desarrollo más utilizado. En este proyecto ha sido utilizado para todo el desarrollo de la página web, esto incluye el tratamiento de los HTML, los CSS, los archivos de JavaScript y los JSON.

- **R Studio** es el entorno de desarrollo más utilizado para el lenguaje R, también es gratuito y de código abierto (aunque existe una versión de pago). Este entorno es muy completo y tiene multitud de herramientas, detección de errores, *debugger*, navegación entre funciones, ayuda integrada, etc… En este proyecto ha sido utilizado para toda la parte del análisis de lenguaje.

- Como apoyo a ambos programas se ha utilizado **Notepad++**, el cual es otro editor de texto y código fuente gratuito. Este editor, desarrollado en C++, tiene menos funcionalidades que Visual Studio Code, pero tiene soporte para R y opciones de codificación de textos que han resultado bastante útiles para el tratamiento de la correspondencia.

Adicionalmente, se ha utilizado el procesador morfológico de castellano GRAMPAL de la Universidad Autónoma de Madrid como complemento en el proceso de lematización del texto de la correspondencia.

Por último, también se ha utilizado Python para desplegar la web en un servidor local y realizar las pruebas. Esto se ha realizado con Python 2.7.15 y Python 3, mediante las clases *SimpleHTTPServer* y *http.server*, respectivamente.

6.4 Problemas Encontrados

Dentro de los problemas encontrados, se van a destacar el más relevante de cada parte:

- Durante el análisis de textos, los problemas más recurrentes han sido los derivados de que los paquetes de minado y tratamiento de texto estén originalmente diseñados para el idioma inglés. Sobresalen tres:
    - La necesidad de crear listas de palabras vacías y lematizador propios, ya que los de estos paquetes, al haber sido adaptados y no creados con el castellano en mente, no llegan a un mínimo de rendimiento aceptable.
    - Los problemas derivados de la codificación del texto y los caracteres especiales. Al estar diseñado para un idioma como el inglés, todos estos paquetes tienen graves problemas con caracteres como los acentos graves, los circunflejos, los apóstrofos, etc… lo que hizo que algunas cartas en francés fuesen modificadas o eliminadas. Además, algunas cartas incluían caracteres gráficos que no eran aceptados, por ejemplo, en una de las cartas aparecía una esquela y en ella una cruz (†), la cual tuvo que ser eliminada.
    - Al trabajar con una cantidad tan grande de datos, los cálculos realizados en R durante la fase de aprendizaje requieren una gran cantidad de memoria y capacidad de procesamiento. Debido a esto, se sufrieron multitud de errores por falta de espacio en la memoria RAM y bastante tiempo "muerto" durante las ejecuciones de los cálculos.
- Durante el desarrollo de la web, el mayor problema fue el tener que aprender y acostumbrarse a una herramienta con tantas posibilidades como es D3.js, sobre todo con los pocos conocimientos de JavaScript que se adquieren el grado. Estas circunstancias provocaron una dilatación del proceso de desarrollo debido al típico ciclo prueba-error del aprendizaje.

Capítulo 7. Desarrollo de las Pruebas

7.1 Pruebas de Integración y del Sistema

Ejecutamos las pruebas funcionales ya diseñadas anteriormente y anotamos el resultado que obtenemos, comparándolo con el que especificamos anteriormente. Podemos hacerlo a partir de una tabla modificada de la anterior como ésta (si es más sencillo, puede hacerse con pequeñas tablas independientes para cada caso):

Caso de Uso 1.1: Añadir Usuario	
Prueba	**Resultado Esperado**
Añadir un usuario no existente	El sistema posee un usuario más
	Resultado Obtenido
	El sistema efectivamente posee un usuario más
Prueba	**Resultado Esperado**
Añadir un usuario que ya existe	El sistema no posee un usuario más y se muestra un dialogo notificándolo
	Resultado Obtenido
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

7.2 Pruebas de Usabilidad y Accesibilidad

7.2.1 Pruebas de Usabilidad

A partir de los cuestionarios que se diseñaron anteriormente y de los procedimientos explicados, mostramos aquí el resultado de aplicarlos sobre todos los usuarios que hayamos empleado para estas pruebas.

Es muy importante a su vez indicar los cambios producidos en la interfaz a partir de las sugerencias recogidas por los usuarios. Una forma adecuada de representar esto sería poner la pantalla inicial y poner la pantalla después de los cambios efectuados.

Además de los cuestionarios diseñados anteriormente, podemos pasar esta guía de usabilidad desarrollada por Yusef Hassan Montero [Hassan08] (http://www.nosolousabilidad.com/articulos/heuristica.htm):

Criterios	¿Cumplido?
Generales	
¿Cuáles son los objetivos del sitio web? ¿Son concretos y bien definidos? ¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?	**SI**
¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?	
¿Muestra de forma precisa y completa qué contenidos o servicios ofrece realmente el sitio web? El diseño de la página de inicio debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio.	
¿La estructura general del sitio web está orientada al usuario? Los sitios web deben estructurarse pensando en el usuario, sus objetivos y necesidades. La estructura interna de la empresa u organización, cómo funciona o se organiza no interesan al usuario.	
¿El *look & feel* general se corresponde con los objetivos, características, contenidos y servicios del sitio web? Ciertas combinaciones de colores ofrecen imágenes más o menos formales, serias o profesionales.	
¿Es coherente el diseño general del sitio web? Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación.	
¿Es reconocible el diseño general del sitio web? Cuánto más se parezca el sitio web al resto de sitios web, más fácil será de usar.	
¿El sitio web se actualiza periódicamente? ¿Indica cuándo se actualiza? Las fechas que se muestren en la página deben corresponderse con actualizaciones, noticias, eventos...no con la fecha del sistema del usuario.	
Identidad e Información	
¿Se muestra claramente la identidad de la empresa-sitio a través de todas las páginas?	
El Logotipo, **¿es significativo, identificable y suficientemente visible?**	

El eslogan o *tagline*, **¿expresa realmente qué es la empresa y qué servicios ofrece?**	
¿Se ofrece algún enlace con información sobre la empresa, sitio web, 'webmaster',...?	
¿Se proporciona mecanismos para ponerse en contacto con la empresa? (email, teléfono, dirección postal, fax...)	
¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?	
En artículos, noticias, informes... **¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?**	
Lenguaje y Redacción	
¿El sitio web habla el mismo lenguaje que sus usuarios? Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario.	
¿Emplea un lenguaje claro y conciso?	
¿Es amigable, familiar y cercano? Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario.	
¿1 párrafo = 1 idea? Cada párrafo es un objeto informativo. Trasmita ideas, mensajes...Se deben evitar párrafos vacíos o varios mensajes en un mismo párrafo.	
Rotulado	
Los rótulos, **¿son significativos?** Ejemplo: evitar rótulos del tipo "haga clic aquí".	
¿Usa rótulos estándar? Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de...".	
¿Usa un único sistema de organización, bien definido y claro? No se deben mezclar diferentes. Los sistemas de organización son: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas.	
¿Utiliza un sistema de rotulado controlado y preciso? Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de"	
El título de las páginas, **¿Es correcto? ¿Ha sido planificado?** Relacionado con la capacidad para poder buscar y encontrar el sitio *web*.	
Estructura y Navegación	
La estructura de organización y navegación, **¿Es la más adecuada?** Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada,...	
En el caso de estructura jerárquica, **¿Mantiene un equilibrio entre Profundidad y Anchura?**	
En el caso de ser puramente hipertextual, **¿Están todos los clúster de nodos comunicados?** Aquí se mide la distancia entre nodos.	

¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos,...)? Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado, y ser reconocidos como una unidad	
En menús de navegación, **¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística?** No se deben superar los 7±2 elementos, ni los 2 o, como mucho, 3 términos por elemento.	
¿Es predecible la respuesta del sistema antes de hacer clic sobre el enlace? Relacionado con el nivel de significación del rótulo del enlace, aunque también con: el uso de globos de texto, información contextual, la barra de estado del navegador,...	
¿Se ha controlado que no haya enlaces que no llevan a ningún sitio? Enlaces que no llevan a ningún sitio: Los enlaces rotos, y los que enlazan con la misma página que se está visualizando (por ejemplo enlaces a la "home" desde la misma página de inicio)	
¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación? ...como *breadcrumbs*, enlaces a la página de inicio,...recuerde que el logo también es recomendable que enlace con la página de inicio.	
Las imágenes enlace, **¿se reconocen como clicables? ¿Incluyen un atributo 'title' describiendo la página de destino?** En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean.	
¿Se ha evitado la redundancia de enlaces?	
¿Se ha controlado que no haya páginas "huérfanas"? Páginas huérfanas: que aún siendo enlazadas desde otras páginas, éstas no enlacen con ninguna.	
Layout de la Página	
¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia? (como por ejemplo la zona central)	
¿Se ha evitado la sobrecarga informativa? Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Los grupos diferentes de objetos informativos de una página deben ser 7±2.	
¿Es una interfaz limpia, sin ruido visual?	
¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?	
¿Se hace un uso correcto del espacio visual de la página? Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia.	
¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página? (La jerarquía visual se utiliza para orientar al usuario)	
¿Se ha controlado la longitud de página? Se debe evitar en la medida de lo posible el *scrolling*. Si la página es muy extensa, se debe fraccionar.	
Búsqueda (si es necesario, por la extensión del sitio, incorporar un buscador interno)	

¿Se encuentra fácilmente accesible? Es decir: directamente desde la home, y a ser posible desde todas las páginas del sitio, y colocado en la zona superior de la página.	
¿Es fácilmente reconocible como tal?	
¿Permite la búsqueda avanzada? (siempre y cuando, por las características del sitio web, fuera de utilidad que la ofreciera)	
¿Muestra los resultados de la búsqueda de forma comprensible para el usuario?	
¿La caja de texto es lo suficientemente ancha?	
¿Asiste al usuario en caso de no poder ofrecer resultados para una consultada dada?	
Elementos Multimedia	
¿Las fotografías están bien recortadas? ¿Son comprensibles? ¿Se ha cuidado su resolución?	
¿Las metáforas visuales son reconocibles y comprensibles por cualquier usuario? (prestar especial atención a usuarios de otros países y culturas)	
¿El uso de imágenes o animaciones proporciona algún tipo de valor añadido?	
¿Se ha evitado el uso de animaciones cíclicas?	
Ayuda	
Si posee una sección de Ayuda, **¿Es verdaderamente necesaria?** Siempre que se pueda prescindir de ella simplificando los elementos de navegación e interacción, debe omitirse esta sección.	
En enlace a la sección de Ayuda, **¿Está colocado en una zona visible y "estándar"?** La zona de la página más normal para incluir el enlace a la sección de Ayuda, es la superior derecha.	
¿Se ofrece ayuda contextual en tareas complejas? (transferencias bancarias, formularios de registro...)	
Si posee *FAQs*, **¿Es correcta tanto la elección como la redacción de las preguntas? ¿Y las respuestas?**	
Accesibilidad (debería cubrirse con los test de Accesibilidad posteriores)	
¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?	
¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleadas facilitan la lectura?	
¿Existe un alto contraste entre el color de fuente y el fondo?	
¿Incluyen las imágenes atributos 'alt' que describan su contenido?	
¿Es compatible el sitio web con los diferentes navegadores? ¿Se visualiza correctamente con diferentes resoluciones de pantalla? Se debe prestar atención a: *JScript*, *CSS*, tablas, fuentes...	
¿Puede el usuario disfrutar de todos los contenidos del sitio web sin necesidad de tener que descargar e instalar *plugins* adicionales?	
¿Se ha controlado el peso de la página? Se deben optimizar las imágenes, controlar el tamaño del código *JScript*...	

¿Se puede imprimir la página sin problemas? Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible.	
Control y Retroalimentación	
¿Tiene el usuario todo el control sobre el interfaz? Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos...	
¿Se informa constantemente al usuario acerca de lo que está pasando? Si el usuario tiene que esperar hasta que se termine una operación, se debe mostrar un mensaje indicándoselo y que debe esperar, con el tiempo de espera estimado o una barra de progreso.	
¿Se informa al usuario de lo que ha pasado? Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.	
Cuando se produce un error, **¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?** Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.	
¿Posee el usuario libertad para actuar? NO restringir la libertad del usuario: Uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación, no impida al usuario poder usar el botón derecho de su ratón...	
¿Se ha controlado el tiempo de respuesta? Esto tiene que ver con el peso de cada página (accesibilidad) y tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder. El tiempo máximo que esperará un usuario son 10 segundos	
Aclaraciones	
¿Se ha evaluado adecuadamente la orientación del usuario? (Donde estoy, como volver, que he visitado, que va a pasar)[1]	
¿Se ha usado correctamente la publicidad?[2]	

Tras rellenar esta tabla se deben poner a continuación una enumeración de todas aquellas aclaraciones y/o observaciones que queramos hacer acerca de la misma.

7.2.2 Pruebas de Accesibilidad

A continuación se detallan las tareas que se recomiendan hacer para asegurarnos de que el programa creado cumple con los estándares de accesibilidad. Esta sección está muy enfocada

[1] La orientación del usuario se puede evaluar a través de varios criterios: elementos de navegación orientativos, caracterización de los enlaces e información contextual en elementos de interacción (estructura y navegación); distribución visual de la página (*layout*); coherencia del diseño (generales); nivel de significación de los rótulos (rotulado) y retroalimentación del usuarios (control y retroalimentación)

[2] Respecto a la publicidad (normalmente en forma de banners), se puede evaluar desde varios criterios: lenguaje (lenguaje y redacción), nivel de significación de los rótulos (rotulado), jerarquía informativa y ruido visual (*layout* de la página), pop-ups y banners intrusivos (control y retroalimentación)...

a proyectos web, pero algunas ideas pueden extrapolarse a programas de escritorio en caso necesario. Un proyecto debería hacer una evaluación de conformidad completa del mismo, pero se ha incluido un procedimiento más abreviado (revisión preliminar) para aquellos casos que por alguna razón se necesite pasar un procedimiento más rápido o menos exhaustivo. Una forma de proceder es hacer la revisión preliminar y, una vez superada arreglando todos los defectos encontrados, intentar ir a por la evaluación de conformidad completa, que permitirá reutilizar todo el estudio hecho en la revisión preliminar para hacerla.

7.2.2.1 Revisión Preliminar

Como paso previo a hacer los pasos descritos, es necesario enumerar el material utilizado para ello, como navegadores (*IE*, *Firefox*,…), lectores de pantalla o cualquier otra herramienta usada para hacer las pruebas.

Paso 1. Selección de un grupo de páginas representativo de la aplicación

Para pasar las pruebas de accesibilidad es necesario escoger una muestra de páginas de la aplicación representativa para así no tener que someter a toda la aplicación al proceso de revisión. Una muestra representativa está formada por:

- La página de entrada al sitio / principal / home
- Páginas con distinta organización y funcionalidad, como por ejemplo:
    - Páginas con tablas
    - Páginas con formularios
    - Páginas con diagramas o gráficos
    - Páginas con scripts o aplicaciones
    - Páginas con resultados generados dinámicamente (por ejemplo, con un gestor de contenidos)

Paso 2. Examinar las páginas usando un navegador gráfico

Esta actividad comprende las siguientes operaciones:

- Desactivar las imágenes y probar como queda el aspecto de la aplicación. Se puede hacer fácilmente mediante el menú "*Images*" de la "*Web Developer Extension*" de *Firefox* de la que se habló anteriormente.
- Apagar los altavoces (si la página tuviese alguna clase de narración oral o sonido)
- Cambiar el tamaño del texto y comprobar que sigue siendo usable (muchos navegadores permiten hacer esto fácilmente con *CTRL + Rueda del ratón*).
- Cambiar solamente el tamaño de letra de la página para ver cómo se comporta. En *Firefox* podemos ir a "Herramientas – Opciones – Contenido", aunque también podemos hacerlo cambiando la *CSS* de la propia página si el navegador no soporta esta opción. Si un tamaño de letra estándar es 16, se puede probar con un tamaño mínimo (9) y un máximo (72) para ver qué ocurre. Posteriormente podemos hacer pruebas con dos tamaños intermedios (32

y 48), para ver qué ocurriría con nuestra página si los usuarios necesitan un tamaño de letra muy superior al estándar.

- Cambiar la resolución y el tamaño de la ventana (verificando que no es necesario el *scroll* horizontal). Esto puede hacerse fácilmente con el *plugin* "*Web Developer Extension*" de *Firefox*. Este *plugin*, una vez instalado tiene una opción "*Resize*". Mediante su sub-opción "*Edit Resize Dimensions*" podemos introducir nuevas resoluciones que podemos usar para comprobar cómo se comporta nuestra aplicación ante ellas. Resoluciones interesantes a probar son: 800x600, 1024x768, 1152x864, 1280x720, 1280x768, 1280x960, 1280x1024 y 1600x1200. Las resoluciones a probar están condicionadas por la máxima resolución de nuestro monitor, por lo que se recomienda poner el mismo a la máxima resolución posible antes de hacer estas pruebas.
- Relacionado con lo anterior, también se recomienda, si es posible, comprobar cómo se ve nuestra aplicación en múltiples tamaños de pantalla (15 pulgadas, 17 pulgadas,…). Otra posible prueba es redimensionar la página múltiples veces (probando a hacerla cada vez más grande y cada vez más pequeña) para ver cómo se comporta su contenido ante esta situación. Si el aspecto de la página se estropea por ello, entonces habremos encontrado un problema.
- Probar a visualizar la página sin sus hojas de estilo *CSS* para asegurarse de que aún es legible y usable. Se puede hacer fácilmente mediante el menú "*CSS*" de la "*Web Developer Extension*" de *Firefox* de la que se habló anteriormente.
- Probar a visualizar la página usando una escala de grises. Para esto podemos usar la herramienta web *GrayBit* (http://graybit.com/main.php).
- Usar el teclado para navegar a través de los enlaces y controles de formularios, usando *Tab* para desplazarse.

Herramientas útiles para comprobar estos aspectos pueden ser:

- *AIS Toolbar* para *Internet Explorer* y *Opera*
- *WAVE Toolbar* para *Firefox*, *Internet Explorer* y *Netscape*
- *Web Developer Extension* para *Firefox (*recomendada*)*

Paso 3. Examinar las páginas usando uno o varios navegadores especializados

Es muy necesario comprobar cómo nuestras páginas se comportan ante diferentes clases de navegadores, como por ejemplo:

- Navegadores por voz como *Firevox* (http://firevox.clcworld.net/) (gratuito) o *JAWS* (http://www.freedomscientific.com/jaws-hq.asp). También puede encontrarse una lista de herramientas similares aquí: http://en.wikipedia.org/wiki/Comparison_of_screen_readers
- Navegadores de texto como *Lynx*. Si se trabaja en *Windows* es posible ejecutarlo usando *Cygwin*.

En estos navegadores es necesario verificar que la información transmitida por ambos tipos es similar a la mostrada en el navegador gráfico y asegurarse de que el orden en el que se transmite dicha información es coherente.

Por otro lado, también es conveniente probar la página con diferentes navegadores (*IE*, *Firefox*, *Opera*, *Chrome*,…) y con distintas versiones de cada navegador. Aunque podría hacerse mediante *VMware*, instalar un gran nº de navegadores y versiones de los mismos es muy costoso. No obstante, una herramienta que nos proporcionará fácilmente acceso a un elevadísimo número de navegadores distintos, con múltiples versiones de cada uno y con distintos sistemas operativos, es *BrowserShots* (http://browsershots.org/). En esta página *Web* podremos introducir la *URL* de nuestro sitio (en caso de que no la tengamos disponible se puede probar con la *IP* de la máquina de desarrollo, asegurándose de que es accesible). Esto invocará a un servicio *web* que distribuirá la petición a múltiples máquinas con distintos navegadores instalados, devolviéndonos las capturas de pantalla en todos y cada uno de esos navegadores de nuestro sitio web, transcurrido un tiempo de proceso de la petición (en función de la hora del día y de la carga de trabajo del servicio puede tardar bastante). Si no cerramos la página o cancelamos la petición y tenemos paciencia, una vez terminado el proceso tendremos imágenes de nuestra web en multitud de navegadores y SO diferentes fácilmente, algo que de otra manera nos supondría un coste muy elevado. Debido al coste en tiempo de esta herramienta, se recomienda hacerlo con la versión final del interfaz de la aplicación.

Paso 4. Utilizar herramientas automáticas de evaluación de accesibilidad

Este último paso consiste en pasar a la muestra de páginas seleccionadas herramientas de evaluación automática de la accesibilidad como:

- *TAW* (http://www.tawdis.net/taw3/cms/es)
- *HERA* (http://www.sidar.org/hera/index.php.en)
- *EvalAccess* (http://sipt07.si.ehu.es/evalaccess2/index.html)
- *WAVE* (http://wave.webaim.org/)

Debemos tener en cuenta:

- Que hay que pasar un mínimo de dos herramientas de esta clase.
- Que sólo pueden probar algunos aspectos de la accesibilidad. Una vez eliminados todos los errores de comprobación automática de la página, debemos hacer todo lo posible por eliminar todos los de comprobación manual (de todas las herramientas pasadas).

Paso 5. Resumen de resultados

Finalmente, se incluirá un pequeño informe final con los siguientes aspectos:

- Tipos de problemas encontrados, como se han resuelto y aspectos positivos de la página.
- Indicar como se detectó cada problema.

7.2.2.2 Evaluación de Conformidad

La evaluación de conformidad combina la evaluación manual de la página con la evaluación semiautomática. Se suele emplear cuando se desarrolla un sitio nuevo o bien cuando se evalúa un sitio existente. En el caso de un PFC, debemos intentar pasar una evaluación de conformidad completa al sitio para considerar que realmente ha hecho un esfuerzo adecuado por lograr un nivel de accesibilidad óptimo.

Paso 1. Determinar el alcance de la evaluación

Este paso contempla:

- Definir y divulgar el nivel deseado de conformidad *WCAG 1.0* (*A*, *AA*, *AAA*). Se recuerda que las web destinadas a la administración pública deben tener al menos un nivel AA, siendo este un nivel mínimo exigible en un PFC. También pueden emplearse las normas *WCAG 2.0*. El siguiente enlace proporciona información de cómo pasar de la versión 1.0 a la versión 2.0 de forma resumida: http://www.w3.org/WAI/WCAG20/from10/comparison/
- Seleccionar un conjunto representativo de páginas para la evaluación manual, siguiendo el criterio expuesto en la revisión preliminar.

Paso 2. Utilizar herramientas de evaluación automática de accesibilidad

Este paso contempla:

- Validar los lenguajes utilizados (*HTML*, *CSS*,…) con las herramientas adecuadas que existan para ello. La "*Web Developer Extension*" de *Firefox* posee opciones para ello.
- Usar al menos dos herramientas de evaluación automática en la muestra de páginas seleccionada (ver la revisión preliminar). También se debe usar al menos 1 herramienta en la totalidad del sitio.
- Anotar y resolver los problemas encontrados.

Paso 3. Evaluar manualmente la muestra de páginas

Para ello debemos:

- Utilizar el *checklist* de puntos de control del *WCAG* que aparecerá tras esta explicación, según versión. Como es mucho más probable que en los PFC se use la versión 1.0 de las normas, al final de esta sección se ha incluido este *checklist* adaptado a *Word* y listo para rellenarse:
  - *WCAG 1.0*: http://www.w3.org/TR/WCAG10/full-checklist.html
  - *WCAG 2.0*: http://www.w3.org/TR/2006/WD-WCAG20-20060427/appendixB.html
- Examinar las páginas con al menos 3 navegadores gráficos en diferentes versiones y en diferentes plataformas (para lo cual puede usarse el servicio *web Browsershots* ya visto). Tener en cuenta especialmente estos puntos, que son una lista extendida de los que se

enunciaron en la revisión preliminar. Para su comprobación podemos usar las mismas herramientas y procedimientos recomendados en dicha sección:

- Desactivar las imágenes y probar como queda el aspecto de la aplicación. Se puede hacer fácilmente mediante el menú "*Images*" de la "*Web Developer Extension*" de *Firefox* de la que se habló anteriormente.
- Apagar los altavoces (si la página tuviese alguna clase de narración oral o sonido)
- Cambiar el tamaño del texto y comprobar que sigue siendo usable (muchos navegadores permiten hacer esto fácilmente con *CTRL + Rueda del ratón*).
- Cambiar solamente el tamaño de letra de la página para ver cómo se comporta. En *Firefox* podemos ir a "Herramientas – Opciones – Contenido", aunque también podemos hacerlo cambiando la *CSS* de la propia página si el navegador no soporta esta opción. Si un tamaño de letra estándar es 16, se puede probar con un tamaño mínimo (9) y un máximo (72) para ver qué ocurre. Posteriormente podemos hacer pruebas con dos tamaños intermedios (32 y 48), para ver qué ocurriría con nuestra página si los usuarios necesitan un tamaño de letra muy superior al estándar.
- Cambiar la resolución y el tamaño de la ventana (verificando que no es necesario el *scroll* horizontal). Esto puede hacerse fácilmente con el *plugin* "*Web Developer Extension*" de *Firefox*. Este *plugin*, una vez instalado tiene una opción "*Resize*". Mediante su sub-opción "*Edit Resize Dimensions*" podemos introducir nuevas resoluciones que podemos usar para comprobar cómo se comporta nuestra aplicación ante ellas. Resoluciones interesantes a probar son: 800x600, 1024x768, 1152x864, 1280x720, 1280x768, 1280x960, 1280x1024 y 1600x1200. Las resoluciones a probar están condicionadas por la máxima resolución de nuestro monitor, por lo que se recomienda poner el mismo a la máxima resolución posible antes de hacer estas pruebas.
- Relacionado con lo anterior, también se recomienda, si es posible, comprobar cómo se ve nuestra aplicación en múltiples tamaños de pantalla (15 pulgadas, 17 pulgadas,…). Otra posible prueba es redimensionar la página múltiples veces (probando a hacerla cada vez más grande y cada vez más pequeña) para ver cómo se comporta su contenido ante esta situación. Si el aspecto de la página se estropea por ello, entonces habremos encontrado un problema.
- Probar a visualizar la página sin sus hojas de estilo *CSS* para asegurarse de que aún es legible y usable. Se puede hacer fácilmente mediante el menú "*CSS*" de la "*Web Developer Extension*" de *Firefox* de la que se habló anteriormente.
- Probar a visualizar la página usando una escala de grises. Para esto podemos usar la herramienta web *GrayBit* (http://graybit.com/main.php). En lo referente a colores, también debemos usar herramientas de verificación de color:
  - ***Color Contrast Analyzer*** (http://www.visionaustralia.org.au/info.aspx?page=628): permite verificar que los colores de fondo y texto de una imagen o página web contrastan lo suficiente para ser distinguibles por cualquier persona, según los algoritmos desarrollados por el W3C.
  - ***Vischeck*** (http://www.vischeck.com/): muestra como ven una página web personas con distintos tipos de daltonismo: deuteranopia, protanopia y tritanopia.
- Usar el teclado para navegar a través de los enlaces y controles de formularios, usando *Tab* para desplazarse.

  - Desactivar *scripts*, *applets*, *Flash*, etc. y comprobar que la página sigue siendo navegable. Para esto es especialmente útil el *plugin NoScript* de *Firefox*
- Examinar las páginas usando uno o varios navegadores especializados: al menos uno de texto y uno de voz (ver revisión preliminar).
- Leer y evaluar el contenido de las páginas, para buscar texto no claro o incongruente. Sobre todo debe primar que el texto de las páginas sea lo más claro posible.

Paso 4. Elaborar el informe de conclusiones

Finalmente se incluirá un informe de conclusiones que contemplará los siguientes aspectos:

- Resumen de los principales problemas encontrados y la solución dada a los mismos
- Resumen de los aspectos positivos que potencian la accesibilidad de la página
- Recomendación de fututas actividades para mejorar la accesibilidad de la página:
  - Reparación de barreras de accesibilidad encontradas y que no hayan podido ser solucionadas del todo o adecuadamente.
  - Ampliación de aspectos positivos de accesibilidad
  - Monitorización del sitio.

Por último, a modo de resumen se destacan aquí algunos aspectos muy importantes que no deben dejarse nunca de lado cuando se desarrolla una página, para minimizar los problemas cuando se haga una revisión de cualquier clase sobre la misma. No obstante, todo esto se contempla en el *checklist* del *WCAG*:

- Poner un texto alternativo para las imágenes
- Poner texto alternativo en los hipervínculos
- Revisar los formularios una vez creados (navegación, claridad)
- Marcos: títulos en los marcos y existencia de *tags <noframes>*
- Desactivar *scripts* y *Flash* y comprobar que la página no pierde funcionalidades
- Navegar sólo con el teclado y comprobar que se puede acceder a todo el sitio.
- Hacer especial hincapié en la revisión de la página de inicio (más importante).

7.2.2.3 Checklist del WCAG 1.0

La siguiente tabla es el *checklist* que el *WCAG* proporciona para verificar las pautas de accesibilidad de la aplicación *web*, pero adaptado a *Word* para poder ser editado fácilmente. Cada punto tiene antes un hiperenlace que va directamente a la web del *WCAG* para proporcionar explicaciones adicionales sobre el mismo (que pueden ir bien para arreglar los problemas detectados).

Puntos de verificación Prioridad 1:

En general (Prioridad 1)	Sí	No	N/A
1.1 Proporcione un texto equivalente para todo elemento no textual (Por ejemplo, a través de "alt", "longdesc" o en el contenido del		X	

elemento). *Esto incluye:* imágenes, representaciones gráficas del texto, mapas de imagen, animaciones (Por ejemplo, *GIFs* animados), "applets" y objetos programados, "ascii art", marcos, scripts, imágenes usadas como viñetas en las listas, espaciadores, botones gráficos, sonidos (ejecutados con o sin interacción del usuario), archivos exclusivamente auditivos, banda sonora del vídeo y vídeos.			
2.1 Asegúrese de que toda la información transmitida a través de los colores también esté disponible sin color, por ejemplo mediante el contexto o por marcadores.	**X**		
4.1 Identifique claramente los cambios en el idioma del texto del documento y en cualquier texto equivalente (por ejemplo, leyendas).			**X**
6.1 Organice el documento de forma que pueda ser leído sin hoja de estilo. Por ejemplo, cuando un documento HTML es interpretado sin asociarlo a una hoja de estilo, tiene que ser posible leerlo.			
6.2 Asegúrese de que los equivalentes de un contenido dinámico son actualizados cuando cambia el contenido dinámico.			
7.1 Hasta que las aplicaciones de usuario permitan controlarlo, evite provocar destellos en la pantalla.			
14.1 Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.			
Y si utiliza imágenes y mapas de imagen (Prioridad 1)	**Sí**	**No**	**N/A**
1.2 Proporcione vínculos redundantes en formato texto para cada zona activa de un mapa de imagen del servidor.			
9.1 Proporcione mapas de imagen controlados por el cliente en lugar de por el servidor, excepto donde las zonas sensibles no puedan ser definidas con una forma geométrica.			
Y si utiliza tablas (Prioridad 1)	**Sí**	**No**	**N/A**
5.1 En las tablas de datos, identifique los encabezamientos de fila y columna.			
5.2 Para las tablas de datos que tienen dos o más niveles lógicos de encabezamientos de fila o columna, utilice marcadores para asociar las celdas de encabezamiento y las celdas de datos.			
Y si utiliza marcos ("frames") (Prioridad 1)	**Sí**	**No**	**N/A**
12.1 Titule cada marco para facilitar su identificación y navegación.			
Y si utiliza "applets" y "scripts" (Prioridad 1)	**Sí**	**No**	**N/A**
6.3 Asegure que las páginas sigan siendo utilizables cuando se desconecten o no se soporten los scripts, *applets* u otros objetos programados. Si esto no es posible, proporcione información equivalente en una página alternativa accesible.			
Y si utiliza multimedia (Prioridad 1)	**Sí**	**No**	**N/A**
1.3 Hasta que las aplicaciones de usuario puedan leer en voz alta automáticamente el texto equivalente de la banda visual, proporcione una descripción auditiva de la información importante de la banda visual de una presentación multimedia.			
1.4 Para toda presentación multimedia dependiente del tiempo (por ejemplo, una película o animación) sincronice alternativas equivalentes (por ejemplo, subtítulos o descripciones de la banda visual) con la presentación.			
Y si todo lo demás falla (Prioridad 1)	**Sí**	**No**	**N/A**
11.4 Si, después de los mayores esfuerzos, no puede crear una página accesible, proporcione un vínculo a una página alternativa que use tecnologías *W3C*, sea accesible, tenga información (o funcionalidad)			

equivalente y sea actualizada tan a menudo como la página (original) inaccesible.			

Puntos de verificación Prioridad 2:

En general (Prioridad 2)	Sí	No	N/A
2.2 Asegúrese de que las combinaciones de los colores de fondo y primer plano tengan el suficiente contraste para que sean percibidas por personas con deficiencias de percepción de color o en pantallas en blanco y negro [Prioridad 2 para las imágenes. Prioridad 3 para los textos].			
3.1 Cuando exista un marcador apropiado, use marcadores en vez de imágenes para transmitir la información.			
3.2 Cree documentos que estén validados por las gramáticas formales publicadas.			
3.3 Utilice hojas de estilo para controlar la maquetación y la presentación.			
3.4 Utilice unidades relativas en lugar de absolutas al especificar los valores en los atributos de los marcadores de lenguaje y en los valores de las propiedades de las hojas de estilo.			
3.5 Utilice elementos de encabezado para transmitir la estructura lógica y utilícelos de acuerdo con la especificación.			
3.6 Marque correctamente las listas y los ítems de las listas.			
3.7 Marque las citas. No utilice el marcador de citas para efectos de formato tales como sangrías.			
6.5 Asegúrese de que los contenidos dinámicos son accesibles o proporcione una página o presentación alternativa.			
7.2 Hasta que las aplicaciones de usuario permitan controlarlo, evite el parpadeo del contenido (por ejemplo, cambio de presentación en periodos regulares, así como el encendido y apagado).			
7.4 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener las actualizaciones, no cree páginas que se actualicen automáticamente de forma periódica.			
7.5 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener el redireccionamiento automático, no utilice marcadores para redirigir las páginas automáticamente. En su lugar, configure el servidor para que ejecute esta posibilidad.			
10.1 Hasta que las aplicaciones de usuario permitan desconectar la apertura de nuevas ventanas, no provoque apariciones repentinas de nuevas ventanas y no cambie la ventana actual sin informar al usuario.			
11.1 Utilice tecnologías *W3C* cuando estén disponibles y sean apropiadas para la tarea y use las últimas versiones que sean soportadas.			
11.2 Evite características desaconsejadas por las tecnologías *W3C*.			
12.3 Divida los bloques largos de información en grupos más manejables cuando sea natural y apropiado.			
13.1 Identifique claramente el objetivo de cada vínculo.			
13.2 Proporcione metadatos para añadir información semántica a las páginas y sitios.			
13.3 Proporcione información sobre la maquetación general de un sitio (por ejemplo, mapa del sitio o tabla de contenidos).			
13.4 Utilice los mecanismos de navegación de forma coherente.			

Y si utiliza tablas (Prioridad 2)	**Sí**	**No**	**N/A**
5.3 No utilice tablas para maquetar, a menos que la tabla tenga sentido cuando se alinee. Por otro lado, si la tabla no tiene sentido, proporcione una alternativa equivalente (la cual debe ser una versión alineada).			
5.4 Si se utiliza una tabla para maquetar, no utilice marcadores estructurales para realizar un efecto visual de formato.			
Y si utiliza marcos ("frames") (Prioridad 2)	**Sí**	**No**	**N/A**
12.2 Describa el propósito de los marcos y cómo éstos se relacionan entre sí, si no resulta obvio solamente con el título del marco.			
Y si utiliza formularios (Prioridad 2)	**Sí**	**No**	**N/A**
10.2 Hasta que las aplicaciones de usuario soporten explícitamente la asociación entre control de formulario y etiqueta, para todos los controles de formularios con etiquetas asociadas implícitamente, asegúrese de que la etiqueta está colocada adecuadamente.			
12.4 Asocie explícitamente las etiquetas con sus controles.			
Y si utiliza "applets" y "scripts" (Prioridad 2)	**Sí**	**No**	**N/A**
6.4 Para los *scripts* y *applets*, asegúrese de que los manejadores de eventos sean independientes del dispositivo de entrada.			
7.3 Hasta que las aplicaciones de usuario permitan congelar el movimiento de los contenidos, evite los movimientos en las páginas.			
8.1 Haga los elementos de programación, tales como *scripts* y *applets*, directamente accesibles o compatibles con las ayudas técnicas [Prioridad 1 si la funcionalidad es importante y no se presenta en otro lugar; de otra manera, Prioridad 2].			
9.2 Asegúrese de que cualquier elemento que tiene su propia interfaz pueda manejarse de forma independiente del dispositivo.			
9.3 Para los "scripts", especifique manejadores de evento lógicos mejor que manejadores de eventos dependientes de dispositivos.			

Puntos de verificación Prioridad 3:

En general (Prioridad 3)	**Sí**	**No**	**N/A**
4.2 Especifique la expansión de cada abreviatura o acrónimo cuando aparezcan por primera vez en el documento.			
4.3 Identifique el idioma principal de un documento.			
9.4 Cree un orden lógico para navegar con el tabulador a través de vínculos, controles de formulario y objetos.			
9.5 Proporcione atajos de teclado para los vínculos más importantes (incluidos los de los mapas de imagen de cliente), los controles de formulario y los grupos de controles de formulario.			
10.5 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten claramente los vínculos contiguos, incluya caracteres imprimibles (rodeados de espacios), que no sirvan como vínculo, entre los vínculos contiguos.			
11.3 Proporcione la información de modo que los usuarios puedan recibir los documentos según sus preferencias (por ejemplo, idioma, tipo de contenido, etc.).			
13.5 Proporcione barras de navegación para destacar y dar acceso al mecanismo de navegación.			
13.6 Agrupe los vínculos relacionados, identifique el grupo (para las aplicaciones de usuario) y, hasta que las aplicaciones de usuario lo			

hagan, proporcione una manera de evitar el grupo.			
13.7 Si proporciona funciones de búsqueda, permita diferentes tipos de búsquedas para diversos niveles de habilidad y preferencias.			
13.8 Localice la información destacada al principio de los encabezamientos, párrafos, listas, etc.			
13.9 Proporcione información sobre las colecciones de documentos (por ejemplo, los documentos que comprendan múltiples páginas).			
13.10 Proporcione un medio para saltar sobre un *ASCII art* de varias líneas.			
14.2 Complemente el texto con presentaciones gráficas o auditivas cuando ello facilite la comprensión de la página.			
14.3 Cree un estilo de presentación que sea coherente para todas las páginas.			
Y si utiliza imágenes o mapas de imagen (Prioridad 3)	**Sí**	**No**	**N/A**
1.5 Hasta que las aplicaciones de usuario interpreten el texto equivalente para los vínculos de los mapas de imagen de cliente, proporcione vínculos de texto redundantes para cada zona activa del mapa de imagen de cliente.			
Y si utiliza tablas (Prioridad 3)	**Sí**	**No**	**N/A**
5.5 Proporcione resúmenes de las tablas.			
5.6 Proporcione abreviaturas para las etiquetas de encabezamiento.			
10.3 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten correctamente los textos contiguos, proporcione un texto lineal alternativo (en la página actual o en alguna otra) para *todas* las tablas que maquetan texto en paralelo, en columnas de palabras.			
Y si utiliza formularios (Prioridad 3)	**Sí**	**No**	**N/A**
10.4 Hasta que las aplicaciones de usuario manejen correctamente los controles vacíos, incluya caracteres por defecto en los cuadros de edición y áreas de texto.			

7.2.2.4 Accesibilidad con Dispositivos Móviles

En caso de que la página a analizar esté destinada a un dispositivo móvil (o tenga una parte o una versión destinada para los mismos), se proporcionan aquí enlaces a herramientas e información útil en estos casos:

- ***W3C mobileOK Checker***: http://validator.w3.org/mobile/
- ***TAW Ok Basic***: http://validadores.tawdis.net/mobileok/es
- ***Ready.mobi***: http://ready.mobi/launch.jsp?locale=en_EN

Otro aspecto a tener en cuenta con estos dispositivos es que si tenemos que evaluar la página en varios de ellos puede ser muy difícil conseguir el hardware necesario. No obstante, para esto nos pueden servir emuladores de los mismos, que permitan comprobar sobre un mismo PC y de forma rápida y sencilla como se ve nuestro sitio en distintos dispositivos móviles con gran fiabilidad. Podemos pues usarlos para enriquecer nuestras pruebas de usabilidad, mostrando cómo se ve nuestra aplicación en un dispositivo móvil. Ejemplos son:

- **.mobi**: http://mtld.mobi/emulator.php

- ***The Openwave Phone Simulator***: http://developer.openwave.com/dvl/tools_and_sdk/phone_simulator/

Una vez pasadas las pruebas indicadas en estos enlaces (para las que también pueden tomarse ideas de las pruebas de accesibilidad anteriores), se debe hacer un informe similar a los de la sección anterior indicando:

- Tipos de problemas encontrados, como se han resuelto y aspectos positivos de la página.
- Indicar como se detectó cada problema.

Capítulo 8. Manuales del Sistema

8.1 Manual de Instalación

Elaborar un manual que contemple todos los pasos necesarios para instalar nuestro sistema, incluyendo la instalación de otras herramientas o software cualquiera (sea o no comercial) necesario para que funcione. Debemos explicarlo todo paso a paso de forma clara y acompañarlo por capturas de pantalla adecuadas.

8.2 Manual de Ejecución

Este manual contemplará todos los pasos necesarios para el arranque de nuestro sistema, lo que es especialmente importante en caso de sistemas con clientes y servidores o distintos procesos que deban arrancarse independientemente.

Por otra parte, también debemos incluir procedimientos para parar adecuadamente la aplicación.

8.3 Manual de Usuario

El manual de usuario es algo muy importante debido a que es el documento que servirá a los usuarios de nuestro sistema para saber cómo funciona cada una de las partes de nuestra aplicación. Debemos pues describir cómo funcionan todas las opciones de las misma, que parámetros tiene, que cosas debemos hacer para que todas las operaciones funcionen correctamente y cualquier otro aspecto que consideremos oportuno para explicar el funcionamiento del sistema.

No debemos escatimar detalles en este manual ya que es la herramienta para que los usuarios comprendan nuestro sistema. También debemos hacer el mayor uso posible de capturas de pantalla para mejorar nuestras explicaciones.

8.4 Manual del Programador

En este manual debemos describir cualquier aspecto que pueda ayudar a otros programadores a ampliar, modificar o entender aspectos de la construcción de nuestra aplicación. Debemos por tanto hacer una descripción general de los distintos aspectos involucrados en la construcción del sistema que puedan ser más difíciles de entender y también describir los procedimientos necesarios para hacer ciertas ampliaciones que hayamos contemplado en el diseño del sistema (añadir nuevas entidades, nuevos atributos a entidades existentes, nuevos servicios que usen a los ya desarrollados, modificaciones en la interfaz, etc.).

Capítulo 9. Conclusiones y Ampliaciones

9.1 Conclusiones

Conclusiones del sistema: Qué hemos elaborado, si los resultados están dentro de lo esperado, si hemos cumplido las expectativas, justificación de haber escogido las mejores opciones para cada uno de los aspectos del sistema, etc.

9.2 Ampliaciones

Las ampliaciones y mejoras que más interesante resultaría realizar son las siguientes:

- **Mejorar el lematizador:** aunque el lematizador actual sea lo suficientemente bueno para realizar un análisis de esta correspondencia en concreto, se podrían buscar reglas generales para algunas de las palabras que se han ajustado con reglas específicas. De esta forma no solo se conseguiría un lematizador más universal si no que se mejoraría el rendimiento del mismo.
- **Refactorizar el código de la web:** con esta ampliación se pretender intentar mejorar el rendimiento de los grafos de D3.js ya que, al cargar tantos elementos como aparecen en la correspondencia de Gaspar Melchor de Jovellanos, los grafos sufren de subidas en el tiempo de respuesta. Además, se podrían añadir nuevas funcionalidades para apoyar la visualización de los datos, como ya hacen la barra de búsqueda, los filtros, etc.
- **Actualizar grafos:** esta ampliación consistiría en adaptar el código de los grafos a la última versión de D3.js, la cual salió a la luz cuando este proyecto ya había comenzado.
- **Actualizar las librerías:** con esta ampliación se pretende adaptar las librerías auxiliares utilizadas para que el código generado por ellas cumpla con las normas de accesibilidad WCAG 2.0 (AA).
- **Crear una red social completa:** esta ampliación consistiría en buscar la correspondencia de los interlocutores de Jovellanos para poder analizar con quien hablan aparte de con él y poder desarrollar una red más compleja.
- **Convertir la web en una herramienta:** aprovechando el trabajo realizado con D3.js, JSON y JavaScript para realizar los grafos, se podría, mediante un proyecto más extenso, desarrollar una aplicación web que genere los grafos de forma automática al proporcionarle los datos en un formato concreto. Esta aplicación sería muy útil para los historiadores que no tienen los conocimientos técnicos o el tiempo necesarios para realizar este tipo de visualizaciones.

Capítulo 10. Presupuesto

A continuación se presentan dos posibles alternativas para el desarrollo del presupuesto del proyecto. La primera de ellas (recomendada) contempla muchos más aspectos y es más completa en líneas generales que la segunda. Seleccionar una u otra depende del criterio del director y del tipo de proyecto. Se presentan ambas para que se seleccione la que se considere más adecuada en cada caso. Otra posible opción es tomar elementos de ambas opciones a conveniencia. **En cualquier caso, consultar al director de proyecto acerca de cuál de las dos es más adecuada para el proyecto desarrollado.**

--

A continuación, se procede a explicar el proceso de desarrollo del presupuesto interno y el presupuesto para el cliente.

10.1 Desarrollo de Presupuesto Interno

A la hora de elaborar un presupuesto hay que tener en cuenta dos aspectos importantes:

- ¿Cuánto cuesta desarrollar el proyecto? **(Presupuesto de Costes)**
- ¿Cuánto voy a cobrar por los trabajos? **(Presupuesto del Cliente)**

El alumno debe realizar ambos. Por lo que respecta al presupuesto del cliente, se hará constar en esta sección y deberá estar basado en el presupuesto de costes realizado. El presupuesto de costes puede realizarse en esta sección o figurar en un anexo específico, a criterio del director del proyecto.

Dicho presupuesto debe ser claro y exhaustivo, adelantándose a las dudas del cliente y presentando todos los elementos, de manera que no exista ningún "punto oscuro". Para ello deben constar únicamente ítems que entienda el cliente. Como ejemplo, el coste de amortización de los equipos de desarrollo no es un dato que deba constar en este presupuesto, sino en el de costes. Un curso de formación, el análisis del sistema, la instalación de un SGBD, etc. si son elementos que entregamos al cliente, sí deben constar en este apartado.

En general el presupuesto del cliente debe cubrir el presupuesto de costes de desarrollo para que el proyecto resulte rentable y debe tener un margen de beneficio con respecto a aquel.

El presupuesto del cliente no debe ser una tabla de valores monetarios vacía, sino que debe ser acompañado por una explicación de estos valores.

Las características fundamentales que debe cumplir son:

- **Completo:** Desde la presentación de nuestra empresa, hasta la solución que pensamos darle al proyecto del cliente, ejemplos de proyectos ya realizados, presupuesto cerrado indicando qué incluye y tan importante como esto, qué no incluye.

- **Claro:** Siempre hay que pensar que el cliente no sabe. Si sabe puede pensar que somos muy básicos, pero si no sabe y le pasamos un presupuesto excesivamente técnico, no le podremos hacer llegar nuestra idea de manera que él pueda visualizar el resultado.
- **Conciso:** No hay que andarse por las ramas. A nosotros nos interesa poder transmitir la mayor cantidad de ideas en el menor espacio posible. Para ello, es muy recomendable utilizar tablas y/o puntos.

A la hora de desarrollar el presupuesto, debemos pensar en hacer referencias a:

- **Metodología de trabajo:** Explicar al cliente como vamos a hacer las cosas, comentar por ejemplo que primero se va a diseñar la web o la base de datos, que hasta que no se apruebe (firmado), no se va a pasar a la siguiente fase.
- **Tiempos de ejecución:** Hay que cumplirlos. También hay que ajustar mucho e indicar cuáles son nuestras responsabilidades y cuáles son las suyas en cuanto a entregas de material, validaciones etc., y la influencia que tendrá en los tiempos, el retraso en las entregas y las aprobaciones.
- **Presupuesto detallado propiamente dicho:** Es muy útil tabular cada una de las secciones que van a componer el proyecto y desglosarlo por su coste.
- **Formas de pago (Si procede):** Indicando también los plazos que habrá que cumplir.

Todos estos apartados, pueden no llevar este orden, incluso estar mezclados, pero es importante tener transmitir todo esto claramente al cliente. Un ejemplo de un desglose de un presupuesto en una tabla se muestra a continuación:

--

Para realizar el cálculo de forma clara, transparente y trazable, se han seguido las siguientes fases:

1. División del proyecto en unidades de obra.
2. Cálculo de los precios básicos.
3. Cálculo de los costes unitarios.
4. Cálculo de los precios de las unidades de obra.
5. Cálculo del presupuesto interno.
6. Cálculo del presupuesto para el cliente.

Además, se ha tratado el proyecto como una colaboración entre varios profesionales, en vez de un pedido a una empresa más establecida, por lo tanto, los gastos indirectos son más reducidos que normalmente.

La división del proyecto en unidades de obra se ha realizado primero en las dos partes principales del proyecto: el análisis de lenguaje y el desarrollo de la página web. Seguidamente, se ha dividido el análisis en: recopilación de texto, limpieza del texto, lematizador y DTM, aprendizaje y resultados y asignación. Por último, se ha dividido el desarrollo en: recursos, página, grafos y validación.

El siguiente paso ha sido calcular los precios básicos del hardware y el personal involucrado. No se ha tenido en cuenta el software ajeno a las tareas de gestión del proyecto debido a que

todo el utilizado es software gratuito. Por su parte, el software de gestión está incluido en los costes indirectos ya que no es exclusivo del proyecto.

En cuanto al personal involucrado se han tenido en cuenta los siguientes profesionales: un experto/a en análisis del lenguaje, un desarrollador/a web y un historiador/a. En cuanto al material, el único hardware que se ha considerado son los ordenadores de los tres profesionales.

Teniendo esto en cuenta los precios básicos son los siguientes:

COD	UNIDAD	DESCRIPCION	PRECIO
RP1	h	Experto en análisis del lenguaje	25,20 €
RP2	h	Desarrollador web	16,40 €
RP3	h	Historiador	17,40 €

COD	UNIDAD	DESCRIPCION	PRECIO
RM1	h	Ordenador / monitor 21"	0,10 €

Para el cálculo de los costes unitarios, simplemente se ha sumado el coste del profesional y el de su equipo, como se puede ver en el siguiente ejemplo:

COD	UNIDAD	DESCRIPCION	PRECIO	HORAS	SUBTOTAL	IMPORTE
DWEB		**Coste del desarrollador web**				
RP2	h	Desarrollador web	16,40 €	1	16,40 €	16,40 €
RM1	h	Ordenador / monitor 21"	0,10 €	1	0,10 €	0,10 €
				Importe horario		**16,50 €**
				Importe diario		**132,00 €**

A continuación, se presentan los cálculos de coste para cada unidad de obra:

Precio nº 1	**Recopilación textos**		
Medición	**Concepto**	**Coste unitario**	**Total**
18	Experto en análisis de lenguaje	25,30 €	455,40 €
		Total	455,40 €

Precio nº 2	**Limpieza de texto**

Medición	Concepto	Coste unitario	Total
8	Experto en análisis de lenguaje	25,30 €	202,40 €
		Total	202,40 €

Precio nº 3	Lematizador y DTM		
Medición	Concepto	Coste unitario	Total
41	Experto en análisis de lenguaje	25,30 €	1.037,30 €
		Total	1.037,30 €

Precio nº 4	Aprendizaje		
Medición	Concepto	Coste unitario	Total
1	Historiador	17,50 €	17,50 €
83,2	Experto en análisis de lenguaje	25,30 €	2.104,96 €
		Total	2.122,46 €

Precio nº 5	Resultados y asignación		
Medición	Concepto	Coste unitario	Total
32	Historiador	17,50 €	560,00 €
32	Experto en análisis de lenguaje	25,30 €	809,60 €
		Total	1.369,60 €

Precio nº 6	Recursos		
Medición	Concepto	Coste unitario	Total
36	Desarrollador web	13,39 €	482,04 €
		Total	482,04 €

Precio nº 7	Página		
Medición	Concepto	Coste unitario	Total
16	Desarrollador web	13,39 €	214,24 €
		Total	214,24 €

Precio nº 8	Grafo

Medición	Concepto	Coste unitario	Total
40,8	Desarrollador web	13,39 €	546,31 €
		Total	546,31 €

Precio nº 9	Validación		
Medición	Concepto	Coste unitario	Total
3	Desarrollador web	13,39 €	40,17 €
		Total	40,17 €

Por último, se añaden los costes indirectos y los beneficios para calcular el presupuesto interno.

PRESUPUESTO INTERNO

Medición	Concepto	Coste Unitario	
1	Precio nº 1: Recopilación de textos	455,40 €	
1	Precio nº 2: Limpieza de textos	202,40 €	
1	Precio nº3: Lematizador y DTM	1.037,30 €	
1	Precio nº 4: Aprendizaje	2.122,46 €	
1	Precio nº 5: Resultados y asignación	1.369,60 €	
1	Precio nº6: Recursos	482,04 €	
1	Precio nº7: Página	214,24 €	
1	Precio nº8: Grafo	546,31 €	
1	Precio nº9: Validación	40,17 €	
		SUMA……	**6.469,92 €**
	(17% del costo del proyecto)	Costes indirectos	1.099,89 €
		PRESUPUESTO DE EJECUCION DE MATERIAL	**7.569,81 €**
		Beneficios (30%)	1.940,98 €
		PRESUPUESTO DE EJECUCIÓN	**9.510,79 €**

10.2 Presupuesto Cliente

Para el presupuesto del cliente, se han agrupado las unidades de trabajo en las dos grandes partes del proyecto: el análisis del lenguaje y la página web.

Concepto	Cantidad	Precio Unitario	Coste Total Concepto
Análisis del lenguaje			7.625,13 €
Desarrollo web			1.885,66 €
		Subtotal	9.510,79 €
		IVA (21%)	1.997,27 €
		TOTAL	**11.508,06 €**

Capítulo 11. Referencias Bibliográficas

11.1 Libros y Artículos

Libros y artículos usados de alguna forma durante el desarrollo del proyecto o su documentación.

Formato sugerido:

[<PrimerApellidoAutor><DosUltimosDigitosDelAño>] <Apellidos1, Nombre1; Apellidos2, Nombre2;…>. ”<Título del libro o Articulo>”. <Editorial o lugar de publicación>. <Año (4 cifras)>.

Ejemplo:

[Redondo07] Redondo L., J. Manuel; De Tal y Cual, Menganito. ”Ejemplo para la plantilla de PFC”. Universidad de Oviedo. 2007.

Si tenemos el ISBN, debemos también ponerlo al final.

--

1. **[Ester, Kriegel, Sander y Li, 1996]** “A density-based algorithm for discovering clusters in large spatial databases with noise”. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.1980. 2018
2. **[Grün y Hornik, 2011]** “topicmodels: An R Package for Fitting Topic Models”. Journal of Statistical Software. 2018
3. **[David M. Blei, Andrew Y. Ng y Michael I. Jordan, 2003]** “Latent Dirichlet Allocation”. http://jmlr.csail.mit.edu/papers/v3/blei03a.html. 2018

11.2 Referencias en Internet

Páginas Web consultadas para cualquier aspecto relacionado con el desarrollo del sistema o su documentación.

Formato sugerido:

[<PrimerApellidoAutor><DosUltimosDigitosDelAño>] <Apellidos1, Nombre1; Apellidos2, Nombre2;…>. "<Título de la página Web>". <URL>. <Año en el que se consultó (4 cifras)>.

Ejemplo:

[Redondo07] Redondo L., J. Manuel; De Tal y Cual, Menganito. "Título de la página Web de ejemplo". www.unaurlcualquiera.com. 2007.

Si tenemos más datos que permitan localizar la información dentro de la página, podemos ponerla donde consideremos oportuno.

Esta referencia es real (se usa dentro del documento) y debe dejarse aquí siempre que usemos el cuestionario que la menciona en la sección de usabilidad.

[Hassan08] Hassan Montero, Y. "Guía de Evaluación Heurística de Sitios Web". http://www.nosolousabilidad.com/articulos/heuristica.htm

1. **[Ingo Feinerer, 2017]** Documentación paquete *tm*. https://cran.r-project.org/web/packages/tm/tm.pdf. 2018
2. **[Ingo Feinerer, 2017]** "Introduction to the TM package. Text Mining in R". https://cran.r-project.org/web/packages/tm/tm.pdf. 2018
3. **[Bouchet-Valat, 2014] "**Snowball stemmers based on the C libstemmer UTF-8 library". https://cran.r-project.org/web/packages/SnowballC/SnowballC.pdf. 2018
4. **[Witten and Tibshirani, 2013]** "Perform sparse hierarchical clustering and sparse k-means clustering". https://cran.r-project.org/web/packages/sparcl/sparcl.pdf . 2018
5. **[Galiano]** "El algoritmo k-means aplicado a clasificación y procesamiento de imágenes". https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html. 2018
6. **[Trevino, 2016]** "Introduction to K-means clustering". https://www.datascience.com/blog/k-means-clustering. 2018
7. "Hierarchical clustering". http://www.saedsayad.com/clustering_hierarchical.htm. 2018
8. **[Statistical Tools for High-Throughput Data Analysis] "**Hierarchical Clustering Essentials - Unsupervised Machine Learning". http://www.sthda.com/english/wiki/print.php?id=237. 2018
9. **[Michael Greenacre]** "Hierarchical cluster analysis". http://www.econ.upf.edu/~michael/stanford/maeb7.pdf. 2018
10. **[Moise, Pournaras y Hellbing]** "Density-Based Clustering". https://www.ethz.ch/content/dam/ethz/special-interest/gess/computational-social-science-dam/documents/education/Spring2015/datascience/clustering2.pdf. 2018
11. **[Nandi, 2015]** "Density-Based Clustering". https://blog.dominodatalab.com/topology-and-density-based-clustering/. 2018

12. **[Statistical Tools for High-Throughput Data Analysis]** "DBSCAN: density-based clustering for discovering clusters in large datasets with noise - Unsupervised Machine Learning". http://www.sthda.com/english/wiki/print.php?id=246. 2018
13. **[Brett, 2012]** "Topic Modeling: A Basic Introduction". http://journalofdigitalhumanities.org/2-1/topic-modeling-a-basic-introduction-by-megan-r-brett/. 2018.
14. "Documentación sobre el método *silhouette*". http://stat.ethz.ch/R-manual/R-devel/library/cluster/html/silhouette.html. 2018
15.

Capítulo 12. Apéndices

12.1 Glosario y Diccionario de Datos

Por orden alfabético, todos los términos que se consideren importantes en la aplicación con una descripción breve de su significado dentro de la aplicación.

- **Término1**: Descripción del significado.
- **Término2**: Descripción del significado.

12.2Índice Alfabético

Este índice alfabético esta generado automáticamente por Word e incluirá todos los términos que nosotros marquemos adecuadamente con la herramienta que *Word* posee para ello (en *Word 2007*, seleccionamos la palabra o frase a indexar y vamos luego a *Referencias* - *Marcar Entrada*. En el cuadro que sale seleccionados luego "*Marcar*", o "*Marcar todas*" para que se busquen todas las apariciones de dicha palabra en el documento). No debemos pues incluir palabras "a mano" en él. Este índice tiene una serie de ejemplos para ilustrar como funciona. No debemos olvidarnos de usar la opción "Actualizar campos" al finalizar la documentación.

NOTA: Quitar esta explicación en la documentación final.

12.3 Código Fuente

El código fuente tiene que ir dividido por paquetes y por archivos, con un formato que haga que resulte legible, tal y como se muestra a continuación en este ejemplo. En la mayoría de las ocasiones, es posible que no sea conveniente colocar la totalidad del código fuente en esta sección, sino solo una parte, que contenga aquellas clases más significativas e importantes. En cualquier caso, conviene consultar al director del proyecto este aspecto. Para dar un formato al código apropiado, podemos probar a copiarlo directamente de nuestro editor y ver si el texto acarrea sus propiedades de estilo (no funcionará en todos los editores), o bien usar programas como *Scintilla* (*Windows*) (http://www.scintilla.org/) o el editor *Kate* (para *KDE*, *Linux*) (http://kate-editor.org/)

12.3.1 Funciones R

12.3.1.1 Función "lematizador":

```
lematizador <- function(word, all.words = FALSE, commonwords =  spcommonwors, dictionary
= spdictionary, morphemes = spmorphemes, ...) {

        word <- tolower(as.character(word))
        getcanonicalword <- function(words, database, all.words = FALSE ) {
                pos <- fmatch(words, database$word )
                pos <- pos[!is.na(pos)]
                if(all.words ) database$canonical[pos]
                else database$canonical[ pos[1] ]
        }

        canonical <-  getcanonicalword(word, commonwords, all.words)
        if(any(!is.na(canonical)) ) return(canonical)

        canonical <-  getcanonicalword(word, dictionary, all.words)
        if(any(!is.na(canonical)) ) return(canonical)
        nch <- nchar(word)

        listroots <- lapply(1:(nch-1), function(i, word, nch) {
                root <- substring(word,1,i)
                desinence <- substring(word,i+1,nch)
                c(root, desinence)
        }, word,nch)

        listroots <- as.data.frame(do.call(rbind, listroots))
        names(listroots) <- c("root","desinence")

        getderivational <- function(x, mylist) {
                pos <- fmatch(x, names(mylist))
                tmp <- mylist[[pos]]
                if(is.null(tmp) ) {NA}
                else {tmp}
        }

        derivational <- lapply(as.character(listroots$desinence), getderivational ,
                spmorphemes)
        names(derivational) <- listroots$root

        possiblewords <-  (unlist(lapply(names(derivational), function(x) paste(x,
                derivational[[x]], sep=""))))
        possiblewords <- possiblewords[ !duplicated(possiblewords)]

        canonical <- getcanonicalword(possiblewords, dictionary, all.words )
        if( any(!is.na(canonical)) ) return(canonical[!is.na(canonical)])
        return(NA)
}
```

12.3.1.2 Función "lematizadorGPAL":

```
lematizadorGPAL <- function( palabra ){

  if(palabra == "") {

    return(NA)

  }

  base.url <-
paste("http://cartago.lllf.uam.es/grampal/grampal.cgi?m=analiza&e=")

  csrf <- readLines( base.url, encoding = 'utf-8' )[[59]]

  csrf <- iconv( csrf, "utf-8" )

  csrf <- strsplit(csrf, "\"")[[1]][[6]] #get csrf code

  csrf <- paste(csrf, "&e=", sep="")

  csrf <- paste(csrf, palabra, sep="")


  word.url <- paste(
       "http://cartago.lllf.uam.es/grampal/grampal.cgi?m=analiza&csrf=",
       csrf,sep = "")

  tmp <- readLines( word.url, encoding = 'utf-8' )

  if(length(tmp) < 79) { return(NA) }

  tmp <- iconv( tmp[[79]], "utf-8" )

  aux <- strsplit(tmp, ">")

  if(length(aux[[1]]) < 3) { return(NA) }

  tmp <- strsplit(aux[[1]][[3]], " ")[[1]][[2]]

  if(tmp == "-") { return(NA) }

  return(tolower(tmp))

}
```

12.3.1.3 Función "stemCustom":

```
stemCustom <- function(x) {
  if(x=="") {
    return()
  }
  for(i in 1:length(x)) {
    l <- unlist(strsplit(x[[i]], " "))
    for(j in 1:length(l)){
      aux <- lematizador(l[[j]])
      #print(aux)
      if(!is.na(aux)) {
        l[[j]] <- aux
      } else {
        aux <- lematizadorGPAL(l[[j]])
        if(!is.na(aux)) {
          l[[j]] <- aux
        } else {
          l[[j]] <- checkWeirdWords(l[[j]])
        }
      }
    }
    x[[i]] <- paste(unlist(l), collapse=" ")
  }
  return(x)
}
```

12.3.1.4 Carga de cartas y limpieza del corpus

```
customStopwords <- read.table("stopwordsJovellanos.txt", header = TRUE)

customStopwords <- as.vector(customStopwords$WORDS)


csv <- read.csv("cartas\\Cartas-full.csv", sep =";", header = TRUE, encoding =
      "UTF-8")

ex <- VCorpus(VectorSource(csv$Textodelacarta))


cleanCorpus <- function(corpus){

  corpus <- tm_map(corpus, content_transformer(tolower)) #to minus

  corpus <- tm_map(corpus, removeNumbers) #numbers

  corpus <- tm_map(corpus, removePunctuation) #punt

  corpus <- tm_map(corpus, content_transformer(function(n) { n <-
       gsub("[¡¿'«»ª°°*\"]", "", n)}))

  corpus <- tm_map(corpus, removeWords, c(stopwords("spanish"),
       customStopwords, "al")) #stopwords

  corpus <- tm_map(corpus, stripWhitespace) #extra whitespace

  return(corpus)

}
```