Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Computación

Desarrollo de una Aplicación para el Tratamiento Automático de Encuestas

Autor:
D. Raúl Hernansanz Quevedo

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Computación

Desarrollo de una Aplicación para el Tratamiento Automático de Encuestas

Autor:
D. Raúl Hernansanz Quevedo

Tutores:
D. Carlos Vivaracho Pascual
Dña. María Aránzazu Simón Hurtado

Agradecimientos

El llegar hasta aquí y terminar este trabajo no ha sido esfuerzo de uno solo y, aunque se merezcan mucho más, me gustaría al menos dar las gracias al resto de personas que lo han hecho posible.

Gracias a mi madre, Ana, por estar siempre ahí, por apoyarme en todo momento y por todo el cariño que me has dado durante estos cinco años, o bueno, durante estos 23 años.

Gracias a mis tíos, Gemma y Javier, que aunque estén un poco más lejos, siempre están ahí para apoyarme y ayudarme.

Gracias a esas dos personas que han hecho de esta etapa universitaria, una experiencia inolvidable. A Adrián por amenizar (casi todas) las clases y, con su sabiduría, solucionar hasta los peores problemas, y a Irene, por su paciencia, apoyo y por esa sonrisa que iluminaba los días más oscuros. Gracias una vez más y deciros que espero poder dároslas otra vez dentro de cinco, diez o quince años más.

Gracias también a mis tutores, Mª Aránzazu Simón Hurtado y Carlos Vivaracho Pascual por el tiempo y apoyo que me han prestado durante la realización de este trabajo e incluso durante el resto de la carrera.

Por último, quiero agradecer también a esas personas que, a pesar de no estar, sé que me están dando ánimos como los que más. Papá, abuelos, va por vosotros.

Índice general

Índice de figuras

Capítulo 1

Introducción

Este proyecto titulado "Desarrollo de una aplicación para el tratamiento automático de encuestas" ha sido desarrollado por Raúl Hernansanz Quevedo bajo la tutela de Mª Aránzazu Simón Hurtado y Carlos Vivaracho Pascual.

En este apartado se hablará de la motivación de este proyecto, de los objetivos propuestos para todo el trabajo y de la estructura de la memoria.

1.1. Motivación

Actualmente existen muchos profesionales que se dedican, entre otras cosas, a la realización de gráficos para mostrar diferentes datos de encuestas. Sin embargo, en la mayor parte de los casos, estos gráficos se realizan expresamente para una encuesta concreta y empleando Excel[1]. Esta forma de proceder no es para nada reutilizable, lo cual provoca una inversión de tiempo elevada para redactar cada informe.

En este trabajo se pretende solventar, o al menos dar unos primeros pasos hacia una solución que permita a esos profesionales, tener un programa con el que poder realizar las visualizaciones que deseen para cualquier encuesta, simplemente conociendo el nombre de las variables que componen dicha encuesta.

Además de esto, y para que la forma de utilizar esa solución resulte más intuitiva, se ha planteado la creación de una interfaz de usuario que permita que esta aplicación pueda ser empleada por cualquier tipo de usuario.

Por todo esto, durante este trabajo de fin de grado se desarrollará la interfaz antes mencionada, que apoye a la aplicación de R [2]. Para ello se realizará todo un proceso de planificación e ingeniería del software que haga que este desarrollo se produzca de la manera más eficiente posible.

1.2. Objetivos

Los objetivos a conseguir con este proyecto pueden resumirse en un objetivo principal, el cual a su vez puede ser desglosado en una serie de objetivos específicos, todos ellos necesarios para cumplir el principal.

- **Objetivo Principal:** Desarrollar la interfaz de usuario para la aplicación de tratamiento automático de R con la cual, introduciendo una serie de parámetros, se produce una llamada a la aplicación de R, de la que se habla en la memoria adjunta [3] que genera un informe con una serie de visualizaciones que indique el usuario.
- **Objetivos Específicos:**
  1. Permitir al usuario indicar las visualizaciones que desea obtener, así como otra serie de parámetros.
  2. Se podrá eliminar la visualización que desee el usuario de la lista.
  3. Permitir al usuario crear un diccionario[1], añadiendo y eliminando variables de él.
  4. La interfaz deberá ser fácil de usar por cualquier persona.

1.3. Estructura de la memoria

Se describirán brevemente todos los apartados por los que estará formada la memoria.

1. **Introducción:** Se introducirá el trabajo, indicando los objetivos de los que consta el mismo.
2. **Organización y proceso de gestión:** Se hablará principalmente del modelo de proceso elegido para el proyecto.
3. **Conceptos previos:** Tratará de enunciar brevemente en qué consiste el desarrollo software y, además, se indicarán las herramientas a utilizar.
4. **Planificación:** Se mostrarán los requisitos a cumplir, y la planificación de todo el proyecto.
5. **Análisis:** Contendrá toda la parte de análisis de la aplicación.
6. **Diseño:** Indicará todos los diagramas necesarios de la parte diseño, así como toda lo relacionado con la encuesta de usabilidad de la interfaz y el diseño final de la misma. También se mostrará el funcionamiento de todas las clases que conforman la aplicación.

[1]Se trata de un documento que contiene las diferentes variables, junto con los códigos que componen cada variable y una descripción de cada código. Por ejemplo, Variable: Sexo. Código: 1. Descripción: Mujer

7. **Implementación:** Se hablará brevemente de los paquetes utilizados para la implementación y de las pruebas realizadas.

8. **Conclusiones y trabajo futuro:** Se exponen las conclusiones del proyecto, así como algunas posibles mejoras que pueden ser implementadas en el futuro.

9. **Bibliografía**

10. **Apéndice A. Manual de instalación de R:** Se explica cómo instalar la herramienta R, para poder utilizar la aplicación.

11. **Apéndice B. Manual de uso de la interfaz:** Instrucciones para poder utilizar la aplicación.

12. **Apéndice C. Contenido del CD:** Contenido del CD

Capítulo 2

Organización y proceso de gestión

A lo largo de esta sección se especificarán los objetivos principales del proyecto, así como las posibles restricciones que pueden darse. También se detallará el modelo de proceso elegido para la realización del trabajo y se enumerarán tanto los posibles riesgos como las medidas que se tomarán en el caso de que esos riesgos se den.

2.1. Objetivos y prioridades de gestión

Para la realización de este proyecto, debido a la restricción de que todo el desarrollo debe ser realizado por una sola persona, las tareas de gestión se vuelven más complicadas que de costumbre, sin embargo, se van a intentar mantener los mismos objetivos que si fuera un proyecto normal, es decir:

- Productividad, se intentarán cumplir los plazos establecidos de la mejor forma posible para poder finalizar el proyecto en el tiempo estipulado.
- Reducción de los riesgos, se creará un plan de contingencia para todos los posibles riesgos que se crea que pueden surgir a lo largo de toda la realización del trabajo.
- Calidad, el objetivo principal es completar el proyecto, y que éste tenga la mayor calidad posible, procurando ir puliendo los fallos que puedan ir surgiendo durante su realización.

2.2. Restricciones

Las restricciones fundamentales de este proyecto que serán enumeradas a continuación, son debidas a factores que limitarán la realización del mismo.

- Tiempo: El tiempo para la realización de este trabajo es limitado y llega hasta principios de Julio.

- Recursos: Dado que se trata de un Trabajo Fin de Grado, no se dispone de una gran cantidad de recursos.
- Conocimientos limitados: Existen muchos detalles dentro del alcance del proyecto de los que apenas se poseen conocimientos, y por tanto esto implica la inversión de mayor cantidad de tiempo en adquirirlos.
- Equipo de trabajo: Como ya se ha mencionado anteriormente, al ser un Trabajo Fin de Grado, debe ser realizado únicamente por una persona.

2.3. Modelo de proceso

Si se habla de desarrollo de software, se suele trabajar con el esquema tradicional de ciclo vida, ya que ha demostrado que es de bastante utilidad en numerosas situaciones. Sin embargo, su problema principal es que está enfocado a proyectos de gran tamaño y por tanto no parece ser el adecuado para el proyecto que se tiene entre manos.

En el caso de proyectos de menor tamaño, las metodologías ágiles aportan una significativa simplificación y aun así no se pierde calidad. La estrella, por así decirlo, de estas metodologías ágiles es SCRUM[4], el cuál, además, es el que más se ha utilizado durante la carrera. A pesar de esto, debido a las restricciones mencionadas en la sección 2.2, no se puede aplicar SCRUM de forma pura y, por tanto, se aplicará una adaptación del mismo, la cual se detallará a continuación.

2.3.1. Marco general de SCRUM

Los tres roles principales de esta metodología son los que siguen.

- Product Owner: Se trata del representante de los *stakeholders* o interesados del proyecto, es decir, el cliente.
- Scrum Master: Es la persona que más conocimiento posee de SCRUM y, por tanto, será el que se encargue de que se cumpla.
- Equipo de desarrollo: Los desarrolladores implicados en el proyecto.

Al hablar de SCRUM debemos conocer y especificar una serie de eventos.

- *Sprint*: Son los periodos de trabajo en los que será dividido el proyecto, los cuales normalmente poseen una duración concreta y al final de los que se debe tener parte de la funcionalidad del proyecto. Hay que tener en cuenta que se debe planificar cada sprint antes de comenzarlo.
- *Scrum meeting*: Se tratan de reuniones diarias entre los miembros del equipo.

- *Sprint review*: Es una reunión que se celebra al final un sprint en la que se echa la vista atrás para comprobar si se han cumplido los objetivos planeados, así como para hablar sobre la funcionalidad cubierta.
- *Sprint retrospective*: También se trata de una reunión que se produce al final de un sprint en la que se analizan los objetivos no cumplidos y se proponen medidas para poder mejorar en el siguiente sprint.

2.3.2. Adaptación de SCRUM

Tras este pequeño resumen de en qué consiste SCRUM, se van a especificar los cambios a aplicar de cara al proyecto.

En el caso de los roles, las personas que los van a llevar a cabo son las siguientes.

- Product Owner: Los tutores del trabajo, tanto los de la parte del Grado en Estadística como los del Grado en Informática.
- Scrum Master y equipo de desarrollo: Raúl Hernansanz Quevedo.

La primera modificación a mencionar es la de que la duración de los sprints no estará fijada de antemano, de manera que cada uno de los que se realicen podrá tener una longitud variable. Debido a este motivo, se pasará a referirse a ellos como *iteraciones* en lugar de sprints. Esto es así debido al reducido equipo de desarrollo, así como a la existencia de otras tareas, entre las que se encuentran las Prácticas en empresa así como una asignatura del Grado. Sin embargo, todas las iteraciones realizadas finalizarán con una reunión con el cliente, ya sea de manera presencial o mediante un correo en el que se adjunte todo lo que se haya realizado en la iteración para que el cliente pueda validarlo y aportar su visión de la situación.

A pesar de esto, durante los cuatro meses de duración del proyecto, se prevé la realización de ocho iteraciones, de aproximadamente dos semanas de duración. Aun así, como ya se ha mencionado, tanto la cantidad de iteraciones como la duración de las mismas podrá cambiar a lo largo de la realización del trabajo. La finalización de cada iteración conlleva una serie de tareas de las que ya se ha hablado en la sección 2.3.1.

- Realización del *sprint review*, aunque en el caso de esta adaptación, englobará también el *sprint retrospective*, de manera que analizará tanto la funcionalidad cubierta como la no cubierta. Además, como se ha decidido que no se realizarán sprints, sino iteraciones, se hará referencia a él de manera distinta.
- Reunión con los clientes para comprobar la funcionalidad añadida así como para detectar posibles fallos o correcciones.
- Planificación de la siguiente iteración, en base a las tareas no completadas, así como a las que restan.

Por último, otra de las cosas que variarán con respecto a SCRUM puro es que no se realizarán *Scrum meeting* debido a que no es necesaria la coordinación entre los miembros del equipo.

La estructura que se seguirá en este documento para especificar la planificación y el seguimiento de las diferentes iteraciones será secuencial, de manera que habrá una sección para cada iteración y ésta estará dividida en dos partes, una para la planificación y otra para el seguimiento.

2.4. Gestión de riesgos

La gestión de riesgos es una de las áreas más importantes que debe haber en la estructura de un proyecto, según propone el PMBOK [5]. El riesgo de no cumplir con los objetivos es más elevado al inicio del proyecto, debido al nivel de incertidumbre. La certeza de terminar con éxito aumenta gradualmente a medida que avanza el proyecto.

A continuación se elaborará una lista de los posibles riesgos globales a los que se enfrenta el proyecto. Además de esto, al final del apartado de seguimiento se detallarán riesgos concretos que han aparecido en el desarrollo y los planes de contingencia aplicados a cada uno. Para cada riesgo identificado, se incluirá la siguiente plantilla.

Identificador	Identificador único del riesgo
Descripción	Descripción del riesgo
Probabilidad	Estimación de la probabilidad de ocurrencia del riesgo
Consecuencias	Consecuencias que puede tener la ocurrencia del riesgo
Impacto	Nivel de impacto que tendría en el proyecto el riesgo en cuestión
Plan de contingencia	Forma de actuar en el caso en que se produzca el riesgo
Comentarios	Comentarios sobre el riesgo

Tabla 2.1: Plantilla de riesgos

Una vez especificada la plantilla, se procederá a enumerar los riesgos globales del proyecto.

Identificador	1
Descripción	Retrasos excesivos en las iteraciones
Probabilidad	Baja
Consecuencias	Los plazos estimados no se completan
Impacto	Muy elevado
Plan de contingencia	El plan de actuación en este caso es muy sencillo, habrá que aumentar el ritmo de producción, aumentando el número de horas diario invertido.
Comentarios	

Identificador	2
Descripción	Baja motivación
Probabilidad	Baja
Consecuencias	El ritmo de producción bajaría de forma considerable
Impacto	Muy elevado
Plan de contingencia	Para este riesgo no hay un plan de actuación como tal, pero para intentar minimizar el impacto se hablaría con los tutores para intentar buscar una solución. En el caso de que haya muchas holguras en los plazos se podría plantear un pequeño descanso para aumentar la motivación.
Comentarios	

Identificador	3
Descripción	Enfermedad de un miembro del equipo
Probabilidad	Media
Consecuencias	No se podría avanzar nada durante ese tiempo
Impacto	Elevado
Plan de contingencia	
Comentarios	En este caso, dado que sólo hay un miembro en el equipo, si la enfermedad le impide realizar tareas, habría que ajustar la planificación para contemplar ese tiempo sin avanzar.

Identificador	4
Descripción	Poca disponibilidad de los clientes
Probabilidad	Baja
Consecuencias	No se podrían solucionar posibles problemas o dudas
Impacto	Elevado
Plan de contingencia	El miembro del equipo de desarrollo se deberá adaptar tanto como pueda a los horarios de los clientes y, en el caso de que aun así sea imposible tener reuniones con ellos, se deberán tratar todos los temas importantes a través del correo electrónico.
Comentarios	

Identificador	5
Descripción	Demasiada complejidad de los requisitos
Probabilidad	Media
Consecuencias	Posible reducción de los requisitos
Impacto	Elevado
Plan de contingencia	La forma de mitigar este riesgo es hablando con los clientes e intentando llegar a un acuerdo para reducir la complejidad de las tareas sin comprometer demasiado la funcionalidad deseada.
Comentarios	

Identificador	6
Descripción	Tiempo insuficiente
Probabilidad	Media
Consecuencias	Posible reducción de los requisitos
Impacto	Elevado
Plan de contingencia	Una vez más, la única solución posible es, ó aumentar las horas invertidas ó tratar con los clientes para reducir los objetivos del proyecto.
Comentarios	

Identificador	7
Descripción	Requisitos mal especificados o ambiguos
Probabilidad	Baja
Consecuencias	No se satisfacen las necesidades de los clientes
Impacto	Muy elevado
Plan de contingencia	Por último, la forma de reducir el impacto de este riesgo es mantener una reunión con los clientes para solventar los posibles problemas.
Comentarios	

2.4.1. Matriz de impacto / probabilidad

En la siguiente tabla se resumirán los riesgos descritos en la sección 1.4. y además se indicará el impacto junto con una estimación de la probabilidad de ocurrencia de cada riesgo.

Identificador	Riesgo	Impacto	Probabilidad
1	Retrasos en las iteraciones	3	20 %
2	Baja motivación	2	5 %
3	Enfermedad de un miembro del equipo	1	15 %
4	Poca disponibilidad de los clientes	3	15 %
5	Complejidad elevada	4	10 %
6	Tiempo insuficiente	4	5 %
7	Requisitos mal especificados	3	5 %

Tabla 2.2: Tabla de riesgos

A partir de la tabla anterior se obtiene la siguiente matriz de riesgos.

Impacto/probabilidad	Muy Alta	Alta	Media	Baja	Muy baja
Catastrófico				5	6
Crítico		1	4		7
Marginal					2
Despreciable			3		

Figura 2.1: Matriz de impacto / probabilidad

Como puede apreciarse en la figura 2.1, el riesgo número 1, *Retrasos en las iteraciones* es el que más hay que vigilar. Tampoco se pueden perder de vista el riesgo 5, *Complejidad elevada*, aunque tenga una probabilidad de ocurrencia baja, ni el riesgo 4, *Poca disponibilidad de los clientes* ya que este último también tiene un gran impacto y una probabilidad no despreciable. Todos los demás riesgos, aunque puedan tener consecuencias graves, las probabilidades de ocurrencia son muy bajas y, por tanto, no hay que prestarles demasiada atención, pero sin llegar a olvidarse de ellos.

Capítulo 3

Conceptos previos

3.1. Desarrollo software

Esta breve sección estará dedicada a comentar todas las fases que se van a tratar en este documento, y que son necesarias en cualquier proyecto de desarrollo software. El proceso está dividido en cinco partes, las cuales aparecen listadas a continuación, además, en la figura 3.1 se puede ver de manera gráfica el proceso descrito.

- **Análisis y planificación:** Esta fase inicial consiste en la identificación de todos los requisitos que deberá tener la aplicación, así como planificar cómo va a ser desarrollada la misma. Aunque, al aplicar una variación de SCRUM, la parte de planificación irá evolucionando con el proyecto.
- **Diseño:** Es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería. Se trataría del proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistemas con los suficientes detalles como para permitir su realización física. El objetivo del diseñador es producir un modelo que será construido mas adelante.
- **Implementación:** Es la parte en la que se programa la aplicación atendiendo tanto a los requisitos como al diseño creados en las fases anteriores.
- **Pruebas:** En esta fase se prueba todo lo desarrollado en la fase anterior buscando posibles fallos o problemas.
- **Mantenimiento:** Esta fase no se tendrá en cuenta en este proyecto.

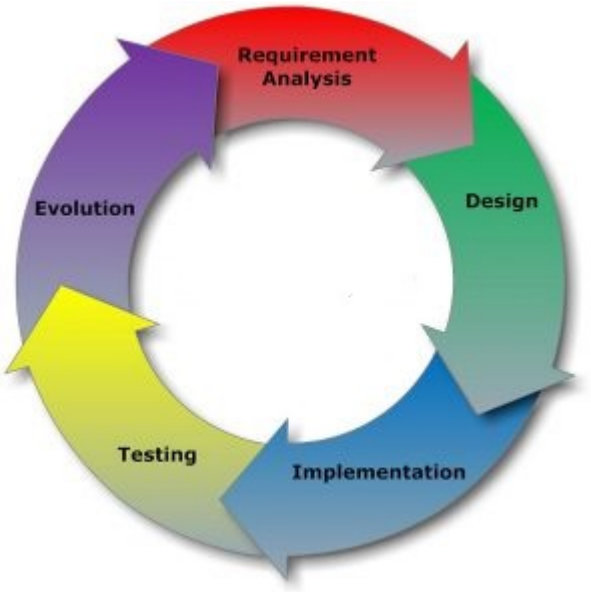

Figura 3.1: Proceso del desarrollo software

3.2. Herramientas y lenguajes utilizados

A continuación se hablará de las herramientas que se han utilizado para llevar a cabo este proyecto, así como de los lenguajes empleados para la implementación.

3.2.1. Para la interfaz de usuario

La creación de la interfaz ha sido realizada en el entorno de desarrollo integrado libre, *NetBeans*[6], y ha sido desarrollada totalmente en el lenguaje *Java*[7]. El motivo principal ha sido la facilidad que ofrece esa herramienta para crear interfaces gráficas, proporcionando al usuario una serie de elementos, como pueden ser botones, paneles de texto, etc. que hacen que la creación de una interfaz se torne algo relativamente sencillo.

Además, se ha utilizado *Java*, a pesar de existir otras posibilidades cuya conexión con *R* es más directa, ya que ha sido el lenguaje de programación que más se ha estudiado durante la carrera y, debido a las limitaciones de tiempo existentes, no era posible aprender un nuevo lenguaje para desarrollar la interfaz.

3.2.2. Para el programa principal

A pesar de que de esto se hablará más en detalle en la memoria anexa [3], indicar que se ha empleado el lenguaje *R* para la programación de la aplicación principal, utilizando una serie de librerías que también se detallan en dicha memoria.

3.2.3. Para la memoria

Para crear la memoria principal han sido necesarios los siguientes programas.

- **Overleaf**[8]: Se trata de un editor de LaTeX[9] en línea. El motivo de su uso es la facilidad de poder seguir editando la memoria desde cualquier parte.
- **Astah**[10]: Ha sido empleado para la creación de todos los diagramas de la parte de diseño.
- **Microsoft Excel**: Se ha utilizado para la creación de algunas tablas de manera más sencilla.
- **LightScreen**[11]: Utilizado para tomar las capturas de pantallas que incluye el documento.

Capítulo 4

Planificación

En esta sección se van a indicar todos los requisitos existentes en este proyecto, así como el *Product Backlog* y la planificación de cada una de las iteraciones, la cuál se irá actualizando a lo largo del desarrollo del proyecto ya que, dado que se está empleando una modificación de SCRUM, la planificación de cada iteración ocurre antes del comienzo de la misma, y no al principio.

4.1. Requisitos

A continuación se enumeran las características que debe presentar la aplicación. Algo que merece la pena mencionar es que esta es una primera visión de la aplicación y que, dado el carácter iterativo del proyecto, se espera que surjan más durante su desarrollo y, probablemente, que aumenten las especificaciones de los que sí se hayan contemplado.

El modelo que será utilizado para describir los requisitos es el siguiente.

Identificador	**Nombre del riesgo**
Descripción	Descripción del requisito
Comentarios	Comentarios respecto del requisito

4.1.1. Requisitos funcionales

1	**Lectura de datos**
Descripción	La aplicación deberá poder leer archivos en formato CSV, TSV, TXT y XLSX
Comentarios	-

2	Lectura de datos
Descripción	La aplicación deberá poder leer archivos en formato texto que utilicen como carácter de separación el espacio o el tabulador.
Comentarios	-

3	Lectura de datos
Descripción	La aplicación deberá poder leer archivos en formato XLSX
Comentarios	-

4	Tablas
Descripción	La aplicación deberá poder crear tablas que impliquen dos o tres variables.
Comentarios	-

5	Formato de datos
Descripción	La aplicación deberá permitir al usuario indicar en qué formato están los datos que va a utilizar.
Comentarios	Podrá elegir entre los que están disponibles

6	Estructura de tablas
Descripción	La aplicación deberá permitir al usuario indicar qué tablas desea, así como las variables implicadas en dicha tabla y su estructura.
Comentarios	-

7	Gráficos de barras
Descripción	La aplicación deberá poder crear gráficos de barras horizontales.
Comentarios	-

8	Gráficos de barras
Descripción	La aplicación deberá poder crear gráficos de barras verticales.
Comentarios	-

9	Gráficos de sectores
Descripción	La aplicación deberá poder crear gráficos de sectores.
Comentarios	-

11	Gestión de gráficos
Descripción	La aplicación deberá permitir al usuario indicar qué gráficos desea, así como las variables implicadas en dicho gráfico.
Comentarios	-

12	Diccionario
Descripción	La aplicación deberá permitir al usuario crear un diccionario para la encuesta con la que trabajará
Comentarios	Esto será opcional y servirá para sustituir los valores en las tablas y gráficos.

13	Rutas
Descripción	La aplicación deberá permitir al usuario indicar dónde se encuentra el archivo de datos y dónde desea obtener el resultado
Comentarios	También podrá indicar el nombre del archivo resultado.

14	Eliminación
Descripción	La aplicación deberá permitir al usuario eliminar visualizaciones antes de ejecutar el programa R
Comentarios	Una vez ha indicado las visualizaciones que desea, debe poder eliminar la que quiera que no se cree, por si se equivoca o cambia de idea.

15	Eliminación
Descripción	La aplicación deberá permitir al usuario editar el diccionario ya creado en cualquier momento.
Comentarios	De esta forma podrá eliminar o añadir las variables que desee cuando quiera.

16	Formato de destino
Descripción	La aplicación deberá permitir al usuario indicar en qué formato desea obtener las visualizaciones
Comentarios	El usuario podrá elegir entre los formatos disponibles.

17	Modificaciones
Descripción	La aplicación deberá permitir al usuario indicar nuevos tipos de visualizaciones así como nuevos formatos de salida a través de un fichero de texto.
Comentarios	Podrá añadir nuevos elementos, pero tendrá que modificar el código de R

4.1.2. Requisitos no funcionales

18	**Lenguaje**
Descripción	La aplicación principal deberá estar programada en lenguaje R.
Comentarios	La forma de implementar dicha aplicación estará detallada en la memoria anexa [3]

19	**Interfaz**
Descripción	La aplicación deberá contar con una interfaz que permita indicar todos los datos necesarios así como ejecutar la aplicación sin necesidad de ver el código.
Comentarios	-

20	**Sistema**
Descripción	El sistema funcionará en el sistema operativo Windows.
Comentarios	-

21	**Manual de usuario**
Descripción	Se deberá proporcionar un manual de usuario que explique cómo funciona la aplicación.
Comentarios	-

4.2. Product Backlog

En esta sección se van a listar las tareas por las que está compuesto el proyecto. La forma de indicarlas será de una manera global, es decir, cada tarea mostrada en la tabla 4.1 podría ser desglosada en varias subtareas con menor alcance. Estas subtareas estarán indicadas en la planificación de cada iteración junto con el elemento del *Product Backlog* al que hacen referencia.

Backlog Item	**Horas**
Creación de la interfaz de usuario	120
Creación de un boceto y un prototipo de la interfaz de usuario	5
Desarrollo y aplicación de una serie de pruebas de cara a probar la usabilidad de la interfaz de usuario	25
Planteamiento y creación de los diagramas de diseño de la interfaz.	50

Backlog Item	Horas
Creación del módulo de lectura de los diferentes formatos de los archivos de datos	15
Creación del módulo para la realización de tablas de 2 o 3 variables	15
Creación del módulo para la realización de gráficos de barras verticales u horizontales	15
Creación del módulo para la realización de gráficos de sectores	15
Creación del módulo que exporte los resultados a distintos formatos	45
Análisis de requisitos	20
Gestión (reuniones, etc.)	8
Pruebas integradas	12
Creación del manual de usuario	5
Creación de la memoria del proyecto	100
Total	450

Tabla 4.1: Product Backlog

4.3. Calendario de iteraciones

Como la metodología de trabajo elegida ha sido SCRUM, no será necesario realizar un diagrama de Gantt que ilustre la planificación global del proyecto, puesto que, en SCRUM, la planificación se va realizando al finalizar cada iteración. No obstante, sí que se cree conveniente plantear un calendario global inicial para ilustrar cómo van a estar distribuidas las iteraciones, aunque dicho calendario podría cambiar dependiendo de las necesidades que vayan surgiendo.

Figura 4.1: Calendario global

4.4. Costes

Por último, antes de comenzar a detallar la planificación real de cada iteración, así como su desarrollo, se va a detallar el coste total del desarrollo de esta aplicación.

4.4.1. Coste laboral

Se va a partir de la base de que los que trabajan en el proyecto lo hacen en calidad de empleados de una empresa. Se debe tener en cuenta que el número de horas de trabajo efectivo de un trabajador ronda unas 1.500 horas al año, siendo un trabajo normal de 8 horas al día, 5 días a la semana. Como estamos considerando que se trata de una empresa, además del sueldo base habría que contar con la seguridad social, gastos médicos etc. Vamos a suponer que esto es un 30 % más sobre el salario base. Pongamos, por ejemplo, que el trabajador tiene un sueldo bajo de aprendiz de 1.200€al mes y que, los tutores, al tener más experiencia, cobran el doble, 2.400€al mes. Es decir, cada trabajador le cuesta a la empresa al año lo siguiente.

- Trabajador: 14 pagas x 1.200€x 1.3 = 21.840€
- Profesores: 14 pagas x 2.400€x 1.3 = 43.680€

Entonces, si cada uno de ellos trabaja 1.500 horas, el coste por hora de cada trabajador sería:

- Trabajador: 21.840€/ 1.500 horas = 14.56€/hora
- Profesores: 43.680€/ 1.500 horas = 29.12€/hora

Finalmente, llevando esos costes al caso que nos atañe, se debe pensar que para la realización del Trabajo Fin de Grado se dedican 450 horas, además, supongamos que al final del proyecto, se habrán invertido unas 8 horas de tutorías.

- Trabajador: 14.56€/hora x 450 horas = 6.552€
- Profesores: 29.12€/hora x 8 horas x 3 personas = 698.88€

4.4.2. Coste material

En este apartado se tendrán en cuenta los costes del material necesario para la ejecución del proyecto.

Será necesario un ordenador, el cual tiene una vida útil de 4 años, y se va a utilizar durante unos 5 meses. El coste de un ordenador normal es de unos 1.300€por tanto, para este proyecto, el coste es de:

Ordenador: 1.300€x 5 meses / 48 meses = 125€

Además, se utilizarán varios programas, sin embargo, son todos gratuitos, por lo que, en cuanto al software, solo habrá que tener en cuenta la licencia de Windows 10, la cual cuesta unos 140€.

4.4.3. Gastos de viajes

A pesar de que yo viajo cada fin de semana al pueblo donde resido, solamente se van a tener en cuenta los viajes realizados para asistir a tutorías, que hemos dicho que iban a ser 8 horas, a una hora por cada tutoría, eso son 16 viajes. Entre Valladolid e Íscar hay 40 kilómetros, suponiendo que la empresa paga 0.30€por cada kilómetros se tiene el siguiente coste total.

Viajes: 8 viajes x 40 km x 0.30€/km = 96€

4.4.4. Costes indirectos y beneficio empresarial

Generalmente se tienen que tener en cuenta los costes indirectos, para ello, se considerará un porcentaje pequeño del coste total que se prevé cueste el proyecto, en este caso serán de un 5 %.

Por último, hay que considerar que una empresa que produzca software desea obtener beneficio, por este motivo, supondremos que la empresa desea ganar un 50 % del coste total del proyecto.

4.4.5. Presupuesto detallado

Teniendo en cuenta todo lo anterior, el presupuesto final que la empresa debería cobrar a los clientes por realizar este proyecto tendría una cuantía de 11.988,71€.

Presupuesto detallado		
Motivo	Cantidad	Total
Coste laboral:		
Trabajadores	6.552€	
Tutores	698.88€	
Coste material:		
Ordenador	125€	
Licencias	140€	
Viajes:	96€	7.611,88€
Costes indirectos y beneficio:		
5%	5% de 7.611,88 €	7.992,474€
50%	50% de 7.992,474€	11.988,711€
Total		11.988,711€

Figura 4.2: Presupuesto detallado

Capítulo 5

Seguimiento detallado

5.1. Primera iteración

5.1.1. Planificación de la primera iteración

Esta primera iteración, se centra en su mayor parte en la realización de varios apartados de la memoria principal, así como en otro de los elementos del *Product Backlog*. Estos dos elementos se encuentran indicados en las figuras 5.1 y 5.2.

Tarea		
Número: 1	**Estimación:** 100	
Prioridad: Alta	**Riesgo:** Medio	**Iteración:** -
Nombre de la tarea: Creación de la memoria del proyecto		
Descripción: Creación de la memoria final del proyecto.		
Validación: Los clientes están satisfechos con los apartados escritos.		

Figura 5.1: Primera tarea

De esta tarea, se pretende escribir los siguientes apartados.

- Organización y proceso de gestión, y todas las sub-secciones correspondientes.
- Planificación, y todas las sub-secciones correspondientes, hasta la planificación de la segunda iteración.
- Diseño conceptual de la aplicación.

Tarea		
Número: 2	**Estimación:** 20	
Prioridad: Alta	**Riesgo:** Medio	**Iteración:** 1
Nombre de la tarea: Análisis de requisitos		
Descripción: Se tratará de encontrar todos los requisitos necesarios para la aplicación a desarrollar.		
Validación: Los clientes encuentran estos requisitos suficientes.		

Figura 5.2: Segunda tarea

El motivo de que esta primera iteración esté centrada en la realización del informe es que el cliente no tiene disponibilidad debido a su trabajo y, por tanto, se ha decidido que la mejor opción es avanzar el informe hasta que se produzca una reunión con él para tratar una serie de temas de la implementación.

5.1.2. Desarrollo de la primera iteración

En esta sección se detallará brevemente cómo ha ido el desarrollo de esta iteración, los posibles contratiempos surgidos, etc.

Esta primera iteración ha finalizado de manera satisfactoria, pudiéndose realizar todas las tareas previstas. Además, se ha tenido una reunión con un profesor del Grado en Ingeniería Informática[12] para tratar temas relacionados con la organización y el proceso de gestión.

5.2. Segunda iteración

5.2.1. Planificación de la segunda iteración

En esta segunda iteración, se va a continuar trabajando en temas necesarios para comenzar con el desarrollo de la aplicación, además de que se tiene prevista una reunión con el tutor del Grado en Estadística[13] para tratar temas del programa de R y, una vez realizada, se comenzará con la implementación de esa parte del programa.

Continuando con la memoria, se pretende escribir los siguientes puntos.

- Planificación de la segunda iteración.
- Análisis de paquetes R para el tratamiento de encuestas.

Adicionalmente a esto, se indican a continuación los elementos del *Product Backlog* que serán realizados en esta iteración.

Tarea		
Número: 3	**Estimación:** 15	
Prioridad: Alta	**Riesgo:** Medio	**Iteración:** 2
Nombre de la tarea: Creación del módulo de lectura de datos		
Descripción: Se creará el módulo de R que permita la lectura de diferentes formatos de archivo de datos.		
Validación: Los archivos de datos son leídos correctamente.		

Figura 5.3: Tercera tarea

Tarea		
Número: 4	**Estimación:** 50	
Prioridad: Alta	**Riesgo:** Medio	**Iteración:** 2
Nombre de la tarea: Planteamiento de los diagramas de diseño		
Descripción: Se plantearán los diagramas de diseño necesarios para la aplicación.		
Validación: La validación será realizada a través dde un profesor del Grado.		

Figura 5.4: Cuarta tarea

5.2.2. Desarrollo de la segunda iteración

La segunda iteración realizada también ha salido bastante bien, sin embargo, no se ha podido finalizar la tarea de diseño y, por tanto se continuará en posteriores iteraciones. En la reunión con el tutor se han aclarado bastantes aspectos relacionados con diversos temas del proyecto, además se ha mostrado el avance del mismo para obtener la aprobación del cliente. Gracias a esta reunión se podrá seguir avanzando de manera correcta en futuras iteraciones.

5.3. Tercera iteración

5.3.1. Planificación de la tercera iteración

La tercera iteración, a diferencia de las dos primeras, estará más enfocada al apartado del desarrollo de la aplicación, no obstante, también se avanzará la memoria del proyecto. Por este motivo, una vez más, se avanzará en algunos elementos del *Product Backlog* además de otras tareas.

En cuanto a la memoria correspondiente, se avanzará en los siguientes puntos.

- Planificación y desarrollo de la tercera iteración.
- Documentación de los módulos desarrollados de la aplicación.
- Reunión con los tutores de informática.

Adicionalmente a esto, se pretende avanzar en algunos elementos del *Product Backlog* que se indicarán a continuación en la figura 5.5 y en la figura 5.6.

Tarea		
Número: 5	**Estimación:** 15	
Prioridad: Alta	**Riesgo:** Medio	**Iteración:** 3
Nombre de la tarea: Creación del módulo para realizar las tablas		
Descripción: Se creará el módulo de R para la creación de las tablas.		
Validación: Las tablas se crean de manera correcta.		

Figura 5.5: Quinta tarea

Esta vez, la iteración está centrada en el desarrollo, esto es así gracias a la reunión descrita en la sección anterior, en la que se clarificaron varios aspectos del proyecto. Además, de esto, se concertará una reunión con los tutores del Grado en Ingeniería Informática para tratar otra serie de temas relevantes para el proyecto de los que se hablará más adelante.

Tarea		
Número: 6	**Estimación:** 15	
Prioridad: Alta	**Riesgo:** Medio	**Iteración:** 3
Nombre de la tarea: Módulo para realizar gráficos de barras		
Descripción: Se creará el módulo de R para la creación de gráficos de barras tanto verticales como horizontales.		
Validación:Los gráficos de barras se crean de manera correcta.		

Figura 5.6: Sexta tarea

5.3.2. Desarrollo de la tercera iteración

En esta sección, como en el caso anterior, se comentará de forma breve cómo se ha desarrollado esta tercera iteración.

A lo largo de las dos semanas de iteración se ha producido un contratiempo personal que ha producido un poco de retraso en la planificación. Sin embargo, se ha completado todo lo planeado de manera correcta.

Una vez más, al igual que en la iteración anterior, se ha concertado una reunión, esta vez con los tutores de informática en la cual se han resuelto temas relacionados con la interfaz de usuario que se va a desarrollar. Tras la finalización de esta iteración, se mostrará el avance conseguido al cliente para que dé su aprobación o proponga modificaciones.

5.4. Cuarta iteración

5.4.1. Planificación de la cuarta iteración

Tras la reunión con los tutores, se va a comenzar con la realización de la interfaz además, se continuará con la tarea relacionada con el diseño que no se pudo completar en la segunda iteración. Por último, se mantendrá una reunión con un profesor de informática para solucionar las dudas relacionadas con el diseño.

Las tareas con las que se va a avanzar se enumerarán a continuación.

- Planificación y desarrollo de la cuarta iteración.
- Reunión con un profesor del Grado Ingeniería Informática de cara a clarificar el diseño de la interfaz.

Adicionalmente a esto, se pretende avanzar en varios elementos del *Product Backlog* que se indicarán a continuación en la figura 5.7 y en la figura 5.8.

Tarea		
Número: 7	**Estimación:** 5	
Prioridad: Alta	**Riesgo:** Bajo	**Iteración:** 4
Nombre de la tarea: Creación del boceto y prototipo		
Descripción: Se creará un boceto en sucio y, tras él, un prototipo de la interfaz de usuario.		
Validación: Los tutores dan el visto bueno a la interfaz.		

Figura 5.7: Séptima tarea

Tarea		
Número: 8	**Estimación:** 15	
Prioridad: Alta	**Riesgo:** Medio	**Iteración:** 4
Nombre de la tarea: Módulo para la creación de gráficos de sectores		
Descripción: Se creará el módulo de R que realice los gráficos de sectores.		
Validación: Los gráficos son creados correctamente.		

Figura 5.8: Octava tarea

5.4.2. Desarrollo de la cuarta iteración

Esta iteración se ha desarrollado sin complicaciones, se han completado las tareas previstas para la misma y se ha enviado todo lo desarrollado al tutor de Estadística para que hiciera sugerencias de posibles cambios para implementar. Dichos cambios ya han sido aplicados al programa principal y son funcionales.

5.5. Quinta iteración

5.5.1. Planificación de la quinta iteración

La cuarta iteración se centrará en el desarrollo de la interfaz gráfica, como esta iteración coincide con las vacaciones de Semana Santa, la carga será inferior a las demás y no se podrá establecer una reunión con ningún tutor. A continuación se enumerarán las tareas que se realizarán en esta iteración.

- Planificación y desarrollo de la cuarta iteración

Adicionalmente a esto, se pretende avanzar en un elemento del *Product Backlog* que se indicarán a continuación en la figura 5.9 y en la figura 5.10.

Tarea		
Número: 9	**Estimación:** 20	
Prioridad: Media	**Riesgo:** Bajo	**Iteración:** 5
Nombre de la tarea: Pruebas de usabilidad		
Descripción: Se realizarán una serie de pruebas de usabilidad con el prototipo de la interfaz creado.		
Validación: No requiere validación.		

Figura 5.9: Novena tarea

5.5.2. Desarrollo de la quinta iteración

Debido a que esta iteración coincidía con las vacaciones de Semana Santa, las tareas previstas no han podido ser completadas. La tarea perteneciente a la figura 5.9 ha sido dividida en dos partes, de cara a realizar las pruebas tanto en el prototipo, como en la interfaz funcional, la parte del prototipo ha sido completada y será detallada más adelante en este mismo informe, mostrando el modelo de cuestionario y los resultados obtenidos, así como el boceto inicial de la aplicación y el resultado final. En cuanto a la tarea vista

Tarea		
Número: 10	**Estimación:** 120	
Prioridad: Alta	**Riesgo:** Alto	**Iteración:** 5
Nombre de la tarea: Desarrollo de la interfaz.		
Descripción: Implementación de la interfaz de usuario para la aplicación.		
Validación: Todo lo implementado funciona correctamente.		

Figura 5.10: Décima tarea

en la figura 5.10, en consecuencia de su alto coste, no se ha podido finalizar y, por tanto, se continuará con ella durante la próxima iteración intentando obtener una versión de la interfaz completamente funcional en la cual solamente haya que aplicar las posibles mejoras y sugerencias propuestas por los tutores.

5.6. Sexta iteración

5.6.1. Planificación de la sexta iteración

Esta iteración estará centrada en completar la interfaz así como en continuar con el informe final, también se concertará una reunión con el tutor del Grado en Estadística para analizar lo ya desarrollado y obtener sugerencias de cómo continuar. Como siempre, se detallan a continuación todas las tareas que se espera desarrollar a lo largo de esta iteración.

- Planificación y desarrollo de la quinta iteración
- Reunión con el tutor del Grado en Estadística

Adicionalmente a esto, se pretende avanzar en dos elementos del *Product Backlog* que se indicarán a continuación en la figura 5.9 y en la figura 5.10, incluidas en la planificación de la iteración anterior.

Además, se documentará todo lo realizado de la aplicación en R, avanzando, de esta forma, con la tarea vista en la figura 5.1.

5.6.2. Desarrollo de la sexta iteración

Esta iteración no ha salido como se esperaba, sobre todo por el hecho de que no se ha conseguido establecer la reunión con el tutor de Estadística. A pesar de este contratiempo, se ha centrado la iteración en completar las tareas que quedaron pendientes de las

iteraciones anteriores, las cuales se mostraban en las figuras 5.9 y en la 5.10. La segunda mencionada, que tiene que ver con la construcción de la interfaz de usuario, ha sido completada con la versión inicial a falta de una revisión con los tutores para validación. La primera de ellas, referente a las pruebas de usabilidad, también ha sido completada y se detallará en secciones posteriores de este mismo documento.

5.7. Séptima iteración

5.7.1. Planificación de la séptima iteración

Esta séptima iteración estará centrada, una vez más, en la parte de la implementación, tanto de cara a la interfaz, como de cara a la aplicación de R, además de esto, se van a celebrar reuniones con los tutores de informática, de cara a revisar todo el trabajo realizado así como para buscar posibles mejoras y solventar problemas.

En cuanto a la reunión que no pudo celebrarse la iteración pasada con el tutor del Grado en Estadística, se tratará de realizarla en esta séptima iteración. Por tanto, a continuación se indican todas las tareas a realizar.

- Planificación y desarrollo de la séptima iteración
- Reunión con el tutor del Grado en Estadística
- Reunión con los tutores del Grado en Ingeniería Informática

Adicionalmente a esto, y como también se ha mencionado, se continuará avanzando en dos elementos del *Product Backlog* que pueden verse en la figura 5.10, incluida en la planificación de la iteración anterior y, en la figura 5.11.

Tarea		
Número: 11	**Estimación:** 45	
Prioridad: Alta	**Riesgo:** Alto	**Iteración:** 7
Nombre de la tarea: Módulo de exportación de resultados		
Descripción: Implementación del módulo que exporte los resultados a distintos formatos.		
Validación: Los documentos resultantes son creados de manera correcta.		

Figura 5.11: Undécima tarea

5.7.2. Desarrollo de la séptima iteración

Como se indicaba en la planificación de la iteración, se ha dedicado todo el tiempo de la misma a la implementación.

En lo que respecta a las reuniones con los diferentes tutores, ambas han podido celebrarse y, de ellas, han surgido una serie de cambios y mejoras que serán implementados en posteriores iteraciones.

5.8. Octava iteración

5.8.1. Planificación de la octava iteración

En este caso, la iteración estará centrada en la realización de la memoria, además de aplicar los cambios comentados en las reuniones de la iteración anterior.

Por tanto, se enumerarán a continuación las tareas a desarrollar durante esta iteración.

- Planificación y desarrollo de la octava iteración.
- Reunión con profesor del Grado en Ingeniería Informática para solucionar dudas.

Adicionalmente a esto, y como también se ha mencionado, se continuará avanzando en un elemento del *Product Backlog* que puede verse en la figura 5.1, incluida en la planificación de la iteración número uno. Además, se aplicarán todos los cambios a la interfaz solicitados en la reunión con los tutores realizada en la iteración anterior, es decir, desarrollar aun más el elemento de la figura 5.10.

5.8.2. Desarrollo de la octava iteración

La octava iteración no se ha ajustado totalmente a lo planeado en la sección anterior, sin embargo, su desarrollo ha sido bastante aceptable, pues se han aplicado todos los cambios pensados a la interfaz de manera correcta y se ha avanzado de manera considerable con la memoria. El problema principal de la iteración ha sido que la reunión con el profesor del Grado ha sido pospuesta y se llevará a cabo en la próxima iteración.

5.9. Novena iteración

5.9.1. Planificación de la novena iteración

Esta iteración, según la planificación inicial, sería la última, no obstante, en función de como se desarrolle, podría ser necesario incluir alguna más. En cuanto a las tareas planificadas para realizar, se listan a continuación.

- Planificación y desarrollo de la novena iteración.
- Reunión con profesor del Grado en Ingeniería Informática para solucionar dudas.
- Reunión con los tutores de ambos Grados.

Adicionalmente a esto, se continuará con la tarea mostrada en la figura 5.1 y, se trabajará en varios elementos del *Product Backlog* muy relacionados con la tarea mencionada. Dichos elementos son los que se muestran en las figuras 5.12 y 5.13.

Tarea		
Número: 12	**Estimación:** 12	
Prioridad: Alta	**Riesgo:** Alto	**Iteración:** 9
Nombre de la tarea: Pruebas unitarias e integradas		
Descripción: Realización de las pruebas unitarias, tanto del programa en R como de la interfaz y de las pruebas de integración de ambos.		
Validación: La aplicación pasa todas las pruebas		

Figura 5.12: Décimo segunda tarea

Tarea		
Número: 13	**Estimación:** 5	
Prioridad: Alta	**Riesgo:** Bajo	**Iteración:** 9
Nombre de la tarea: Manual de usuario		
Descripción: Creación del manual de usuario de la aplicación.		
Validación: El manual es fácil de entender y de utilizar.		

Figura 5.13: Décimo tercera tarea

5.9.2. Desarrollo de la novena iteración

Esta iteración ha supuesto algunos cambios importantes en la planificación inicial. En vista del poco tiempo restante hasta la fecha máxima de presentación del trabajo, se ha decidido retrasar dicha presentación hasta la convocatoria extraordinaria teniendo, de esta forma, más tiempo para terminar de completar todas las tareas necesarias de la mejor manera posible. Las reuniones con los profesores y tutores han sido satisfactorias, y se han clarificado todas las dudas que se tenían pendientes. No obstante, los tres elementos del *Product Backlog* pensados para esta iteración no han podido ser completados y se pospondrán a futuras iteraciones.

5.10. Décima iteración

5.10.1. Planificación de la décima iteración

Esta iteración está pensada para completar todas las tareas restantes que se tienen, es decir, la idea de esta iteración es completar el trabajo, sin tener en cuenta posibles correcciones planteadas por los tutores sobre la memoria, ni la preparación de la presentación para la exposición. Por tanto, las tareas indicadas en la iteración anterior, pasan a formar parte de estas. Dichas tareas pueden verse en las figuras 5.1, 5.12 y 5.13.

5.10.2. Desarrollo de la décima iteración

La décima iteración ha sido finalizada de forma satisfactoria, habiendo completado todas las tareas planteadas. En la semana restante antes de la entrega del proyecto, se realizaran las posibles correcciones planteadas por los tutores así como añadir la bibliografía a la memoria final.

5.11. Riesgos presentados

A lo largo del desarrollo completo del proyecto, se han producido una serie de problemas que han tenido un impacto en la planificación inicial. Dichos riesgos pueden ser resumidos en dos.

- Problemas con el desarrollo de la aplicación, causados por errores puntuales o por falta de conocimiento para realizar ciertas tareas.
- Problemas con la memoria, los cuales han sido debidos a una mala estimación de las horas necesarias para escribir la misma.

La principal forma de suplir estos problemas, ha sido dedicar horas extra diarias a la realización de tareas, así como ampliar las 9 iteraciones inicialmente planeadas con una décima de dos semanas de duración, que ha permitido finalizar el proyecto, consiguiendo además una semana de margen para poder realizar posibles correcciones.

Capítulo 6

Análisis

Antes de dar paso al diseño de la aplicación, se va a realizar la parte de análisis. En ella se mostrarán todos los diagramas de modelado necesarios, así como una descripción de los casos de uso que debe abordar la aplicación.

6.1. Modelado conceptual

En esa sección se tratará de explicar, de forma conceptual, en qué consiste la aplicación, y cuál es su objetivo. La forma de alcanzar ese objetivo, es decir, el funcionamiento interno, se detallará en secciones posteriores de este mismo documento.

La idea general de la aplicación es el tratamiento automático de encuestas, es decir, el usuario provee a la aplicación de un archivo de datos procedentes de una encuesta, y solicita alguna de las visualizaciones ofertadas por la aplicación. De esta manera, la aplicación realizará esas visualizaciones y creará un documento con ellas en el formato que indique el usuario. Esto puede verse en forma de esquema en la figura 6.1

Para este proyecto, se ha decidido dividir la aplicación en dos partes diferenciadas, la primera de ellas se trata de la interfaz de usuario, que es de la que se va a hablar en este documento. La segunda parte es un programa en *R* que creará las visualizaciones y obtendrá el documento solicitado pero, esta segunda parte se explicará en el documento anexo[3].

En lo que respecta a la parte de la interfaz, su función es proveer al programa en R de todo lo necesario para poder trabajar, esto es.

- Ruta en la que se encuentra R. Parámetro obligatorio.
- Ruta en la que se encuentra el fichero de datos. Parámetro obligatorio.
- Separador del fichero de datos. Parámetro obligatorio.
- Formato de destino del informe. Parámetro obligatorio.

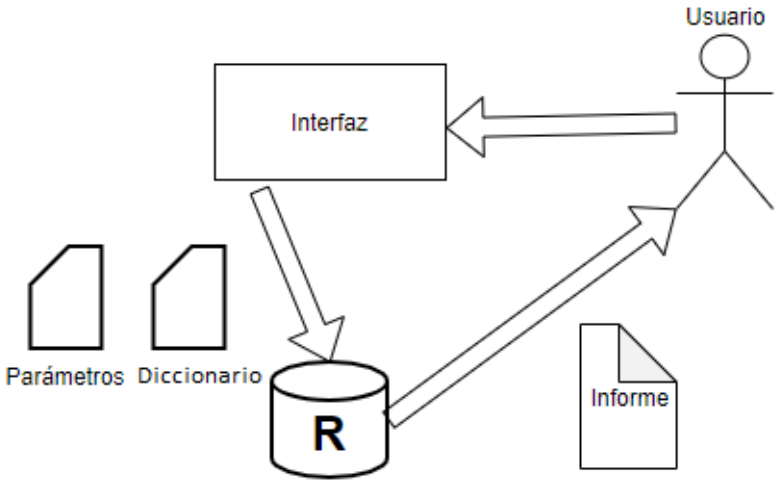

Figura 6.1: Diagrama de la aplicación

- Ruta en la que se guardará el informe final. Parámetro optativo.
- Visualizaciones que se desean. Obligatorio indicar al menos una.
- Diccionario de datos para la encuesta. Parámetro optativo.
- Nombre que tendrá el informe resultado. Parámetro obligatorio.
- Nombre del factor de elevación de la encuesta. Parámetro obligatorio.

Para ello, la interfaz recibirá un archivo de texto en el que el usuario indicará los formatos de salida, los separadores y los tipos de visualizaciones. El motivo de esto es que el usuario pueda ampliar la funcionalidad de la aplicación sin necesidad de acceder al código fuente, teniendo en cuenta que deberá modificar el programa en R para contemplar los nuevos parámetros que cree.

Una vez se han indicado todos los parámetros obligatorios y pulsado aceptar, la interfaz creará dos ficheros de texto, uno para el diccionario y otro para el resto de parámetros y llamará a la aplicación de R para que cree las visualizaciones utilizando esos ficheros.

De manera interna, la interfaz está compuesta por distintas ventanas con la composición que se ve en la figura 6.2. Así pues, la mayor parte de la funcionalidad se encontraría en las ventanas principal y de diccionario, siendo las de portada y espera meramente decorativas, y la de warning tendrá la función de informar al usuario en el caso de que el archivo que desea crear ya exista en la ubicación que haya indicado, dándole la opción de reemplazarlo o de cancelar la operación.

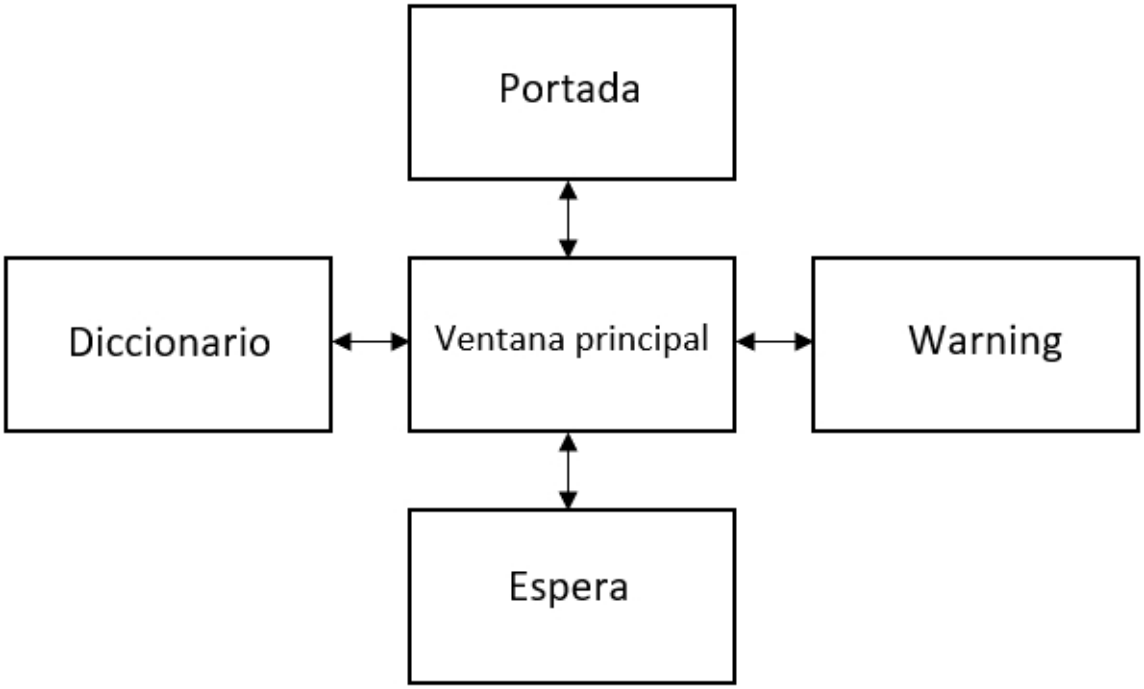

Figura 6.2: Esquema interfaz de usuario

6.2. Modelo de casos de uso

A partir de la especificación de requisitos, se han obtenido los casos de uso del sistema. Como el objetivo de esta parte del proyecto es la creación de la interfaz, solo se distingue un actor que puede realizar los casos de uso.

- **Usuario:** usuario que utilizará la interfaz.

A continuación, se van a detallar los casos de uso que se han encontrado y que pueden verse en la figura 6.3.

- **Crear archivo de parámetros:** se introducen todos los datos necesarios para la aplicación.
- **Crear visualización:** se añade una nueva visualización a la lista.
- **Eliminar visualización:** se elimina la visualización que desee el usuario.
- **Crear diccionario:** se crea un diccionario asociado.
- **Añadir variable:** se añade una variable al diccionario.

Figura 6.3: Diagrama de modelo de casos de uso

- **Eliminar variable:** se elimina la variable del diccionario que desee el usuario.
- **Editar diccionario:** se edita el diccionario para añadir o eliminar variables.

6.3. Modelo de dominio

En esta sección se describirá el modelado de dominio de la interfaz. A continuación, en la figura 6.4, se muestra el esquema del modelo en UML[14] (realizado con Astah) con las entidades y las relaciones entre ellas, incluyendo multiplicidades. También se muestran los atributos de cada entidad.

El modelo contiene las siguientes entidades:

- **FuenteDeDatos:** representa la clase principal del sistema, contiene todos los elementos necesarios para el funcionamiento de la aplicación. Está relacionado con la clase Diccionario, y también con la clase Visualización.
- **Diccionario:** representa el diccionario asociado a los datos. Está relacionada con la clase FuenteDeDatos y con la clase Entrada.
- **Entrada:** representa cada componente por el que está formado el diccionario.
- **Visualización:** representa las visualizaciones requeridas por el usuario. Está relacionada con la clase FuenteDeDatos.

En lo que respecta a las relaciones, simplemente mencionar que la que existe entre las entidades FuenteDeDatos y Diccionario, tiene multiplicidades 1 a 0..1 ya que no es necesaria la creación de un diccionario pero, en el caso de crearlo, solo se podría crear uno. Además, la relación entre la entidad Diccionario y la entidad Entrada, indica que para que exista un diccionario, tiene que existir al menos una entrada en el mismo.

La otra relación existente, se trata de la que hay entre las entidades FuenteDeDatos y Visualización, con multiplicidades 1 a 1..*. Es este caso, para que la entidad Datos esté completa, debe contar al menos con una visualización, pero puede tener todas las que se quiera.

6.4. Descripción de los casos de uso

En esta sección se detallarán todas las secuencias de cada uno de los casos de uso mostrados en la sección 4.1

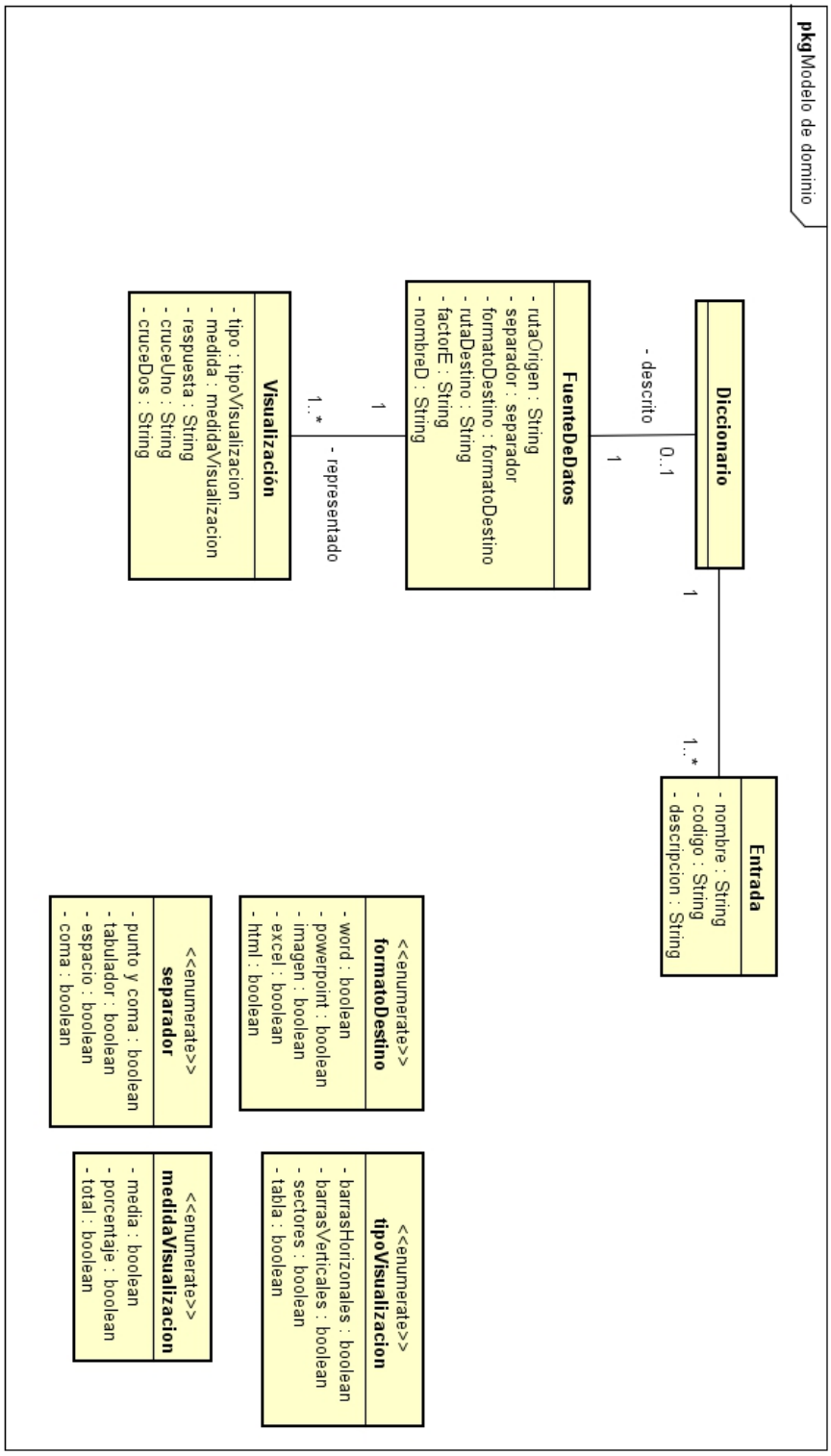

Figura 6.4: Diagrama de modelo de dominio

ITEM	VALUE
UseCase	Crear diccionario
Summary	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor Usuario crea un diccionario.
Actor	Usuario
Precondition	No existe diccionario ya creado
Postcondition	El diccionario se crea de manera satisfactoria
Base Sequence	1- El actor Usuario indica que desea crear un diccionario 2- El sistema muestra la vista de creación del diccionario 3- El actor Usuario introduce una nueva variable con su código y descripción 4- El sistema muestra la nueva variable creada 5- El actor Usuario acepta la creación del diccionario
Branch Sequence	3a- El actor Usuario añade más variables al diccionario 3b- El actor Usuario elimina alguna variable del diccionario
Exception Sequence	
Sub UseCase	
Note	

Figura 6.5: Descripción del caso de uso Crear Diccionario

6.4.1. Caso de uso Crear Diccionario

El primer caso de uso que se describirá será el de crear diccionario, su secuencia se encuentra detallada en la figura 6.5.

6.4.2. Caso de uso Añadir variable

Este caso de uso es una parte incluida dentro del caso de uso descrito en la sección anterior, aun así, se indicará su secuencia en la figura 6.6.

ITEM	VALUE
UseCase	Añadir variable
Summary	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor Usuario trate de añadir una variable al diccionario
Actor	Usuario
Precondition	No existe
Postcondition	La variable se añade satisfactoriamente
Base Sequence	1- El actor Usuario accede a la edición de diccionario 2- El sistema muestra la ventana de edición de diccionario 3- El actor usuario rellena los campos necesarios para crear la nueva variable y pulsa el botón añadir 4- El sistema actualiza las variables creadas con la nueva
Branch Sequence	
Exception Sequence	3a- Alguno de los campos indicados por el usuario no cumple los requisitos
Sub UseCase	
Note	En el caso de que no se cumplan los requisitos, el sistema indicará qué campo es el erroneo, y el usuario tendrá que volver a introducir los campos

Figura 6.6: Descripción del caso de uso Añadir Variable

6.4.3. Caso de uso Eliminar variable

Se muestra su secuencia en la figura 6.7.

6.4.4. Caso de uso Editar diccionario

Se muestra su secuencia en la figura 6.8.

ITEM	VALUE
UseCase	Eliminar variable
Summary	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor Usuario trate de eliminar una variable al diccionario
Actor	Usuario
Precondition	Dicha variable existe anteriormente
Postcondition	La variable se elimina satisfactoriamente
Base Sequence	1- El actor Usuario accede a la edición de diccionario 2- El sistema muestra la ventana de edición de diccionario con las variables y a creadas 3- El actor usuario hace doble click sobre la variable que desea eliminar 4- El sistema actualiza las variables creadas con la nueva
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Figura 6.7: Descripción del caso de uso Eliminar Variable

ITEM	VALUE
UseCase	Editar diccionario
Summary	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor Usuario desee editar el diccionario
Actor	Usuario
Precondition	Ya hay un diccionario creado
Postcondition	El diccionario se modifica tal y como ha pedido el usuario
Base Sequence	1- El actor Usuario accede a la edición de diccionario 2- El sistema muestra la ventana de edición de diccionario con las variables ya creadas 3- El caso de uso continúa o bien por el caso de uso de eliminar variable o bien por añadir variable
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Figura 6.8: Descripción del caso de uso Editar Diccionario

6.4.5. Caso de uso Crear Archivo de Parámetros

Este es el caso de uso principal de la aplicación, en la figura 6.9 se detalla su secuencia.

ITEM	VALUE
UseCase	Crear archivo parámetros
Summary	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor Usuario cree el archivo de parámetros
Actor	Usuario
Precondition	No existe
Postcondition	El archivo de parámetros se crea de manera satisfactoria
Base Sequence	1- El actor usuario indica que desea introducir la ruta del archivo de origen 2- El sistema muestra el explorador de archivos 3- El actor usuario selecciona el archivo de datos 4- El sistema muestra la ruta en la que se encuentra dicho archivo 5- El actor usuario selecciona el formato de destino y el separador del archivo 6- El actor usuario indica que desea introducir la ruta de destino 7- El sistema muestra el explorador de archivos 8- El actor usuario selecciona una ruta 9- El sistema muestra la ruta seleccionada 10- El actor usuario crea alguna visualización 11- El actor usuario presiona Aceptar
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	El paso 12 será detallado en el caso de uso "Crear visualización"

Figura 6.9: Descripción del caso de uso Crear Archivo de Parámetros

6.4.6. Caso de uso Crear Visualización

Este caso de uso va ligado al anterior, sin embargo, se considera que tiene suficiente complejidad para indicar su secuencia, la cual se muestra en la figura 6.10.

6.4.7. Caso de uso Eliminar Visualización

Este caso de uso va un poco ligado al de la figura 6.9, a pesar de ello, se va a indicar su secuencia en la figura 6.11

ITEM	VALUE
UseCase	Crear visualización
Summary	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor Usuario cree una visualización
Actor	Usuario
Precondition	No existe
Postcondition	La visualización se añade correctamente
Base Sequence	1- El actor usuario selecciona el tipo de visualización. 2- El actor usuario selecciona la medida representada. 3- El actor usuario indica la variable respuesta. 4- El actor usuario indica la variable de cruce 1. 5- El actor usuario añade la visualización detallada. 6- El sistema muestra en pantalla la visualización creada y limpia los campos de las variables.
Branch Sequence	4a- El actor usuario indica la variable de cruce 2.
Exception Sequence	
Sub UseCase	
Note	

Figura 6.10: Descripción del caso de uso Crear Visualización

ITEM	VALUE
UseCase	Eliminar visualizacion
Summary	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor Usuario desee eliminar una visualización
Actor	Usuario
Precondition	Ya existe alguna visualización creada
Postcondition	La visualización se elimina correctamente
Base Sequence	1- El actor usuario accede a la ventana principal 2- El sistema muestra la ventana principal con las visualizaciones ya creadas 3- El actor usuario hace doble click sobre la visualización que desea eliminar 4- El sistema actualiza la lista de visualizaciones eliminando la indicada
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Figura 6.11: Descripción del caso de uso Eliminar Visualización

Capítulo 7

Diseño

Para el correcto desarrollo de una aplicación, es esencial un buen diseño software. Por esto, van a ser aplicadas distintas técnicas y principios, seleccionados cuidadosamente, con el propósito de definir un sistema con el suficiente detalle como para permitir su construcción física (su implementación). Si se toma como punto de partida un diseño bien hecho, la implementación será más rápida, eficaz y eficiente, buscando siempre respetar los principios de diseño.

Durante esta sección se van a tratar los siguientes contenidos. En primer lugar se hablará sobre la arquitectura de software abordada de forma global, tras esto, se detallará el proceso seguido para crear la interfaz, desde el boceto inicial, hasta el resultado final, pasando por algunas pruebas con usuarios externos, seguidamente se mostrará la realización en diseño para dos de los casos de uso indicados en la sección anterior y, por último, se hablará del diseño detallado de las clases que conforman la interfaz.

7.1. Arquitectura de la aplicación

La arquitectura elegida para la implementación de la aplicación es una arquitectura basada en el conocido patrón Modelo-Vista-Controlador Jerárquica en su variante pasiva[15]. Como solo se trata de una interfaz, la arquitectura es muy sencilla, solo necesitan del patrón antes mencionado para toda la implementación. Se trata de la variante jerárquica pues existe una vista principal que indicará cuándo se deben crear el resto.

Este patrón permite abordar la complejidad y aplicar el Principio de Responsabilidad Única (SRP)[16] para separar *cómo* se presenta la información al usuario, de las operaciones que se realizan sobre la interfaz. El modo en el que la interfaz de usuario realiza las peticiones del usuario se considera independiente de la interfaz gráfica de usuario y, por tanto, se separan en dos clases diferentes.

La variante elegida será la pasiva, donde el Modelo no establece ningún tipo de comunicación indirecta con la Vista. En la vertiente activa, es posible que la Vista deba

actualizarse en tiempo real en el momento de actualización del Modelo. Esto está fundamentalmente pensado para aquellas aplicaciones en las que múltiples personas se encuentran trabajando globalmente. En el caso que nos ocupa, esto no se va a dar. A continuación se explican de manera breve los tres complementes de este patrón.

- Modelo: contiene toda la información final con la que trabaja la aplicación.
- Vista: encargada de mostrar la información al usuario.
- Controlador: se encarga de manejar las comunicaciones entre la vista y el modelo.

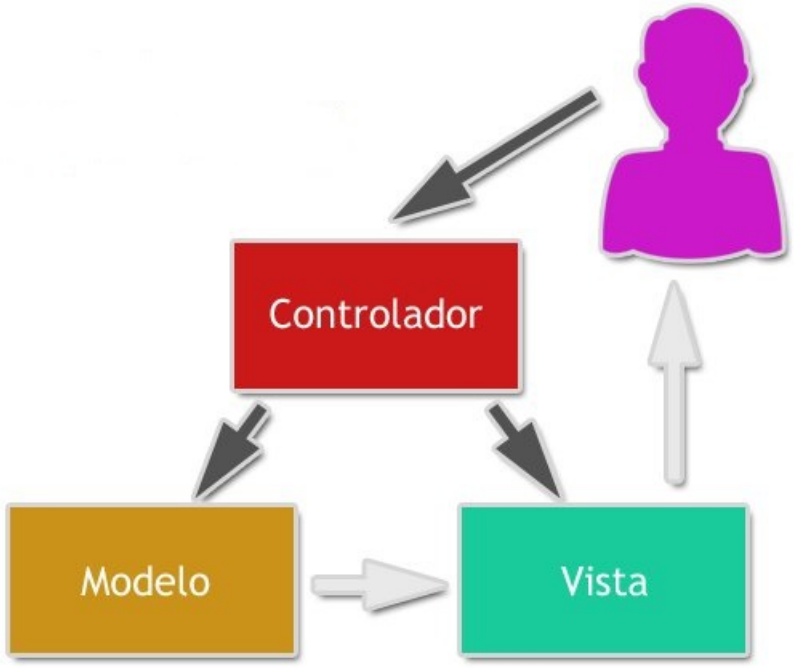

Figura 7.1: Modelo Vista Controlador

En la imagen 7.1, se muestra visualmente el funcionamiento de este modelo, su explicación es muy sencilla. El usuario realiza las peticiones al controlador, este a su vez, realiza las acciones pertinentes ya sea actualizando el modelo y/o la vista y, en el caso de actualizar el modelo, la vista se actualiza con los nuevos datos del modelo, mostrándoselos al usuario.

La arquitectura global de la aplicación puede encontrarse en la figura 7.2.

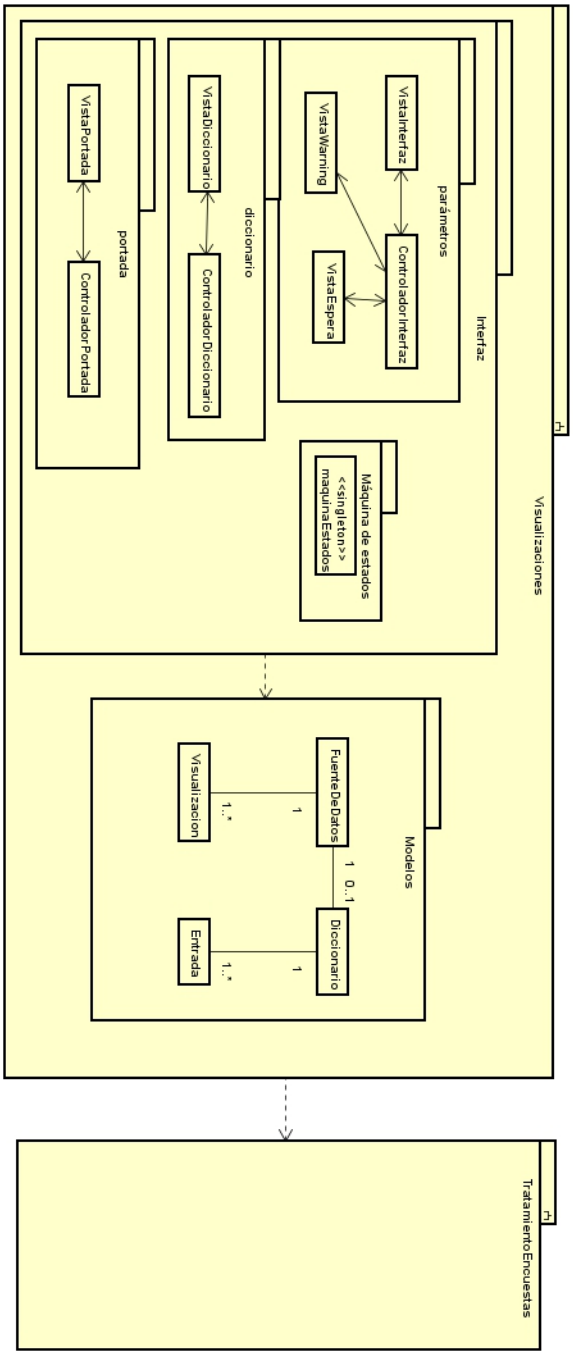

Figura 7.2: Arquitectura de la Aplicación

7.2. Realización en diseño de los casos de uso

En esta sección se van a describir, de forma detallada, tres de los casos de uso listados en la sección anterior.

7.2.1. Caso de uso Añadir variable

En primer lugar, se mostrará la realización del caso de uso en la fase de análisis. A este nivel no nos preocupamos de los detalles técnicos de la solución final ni de las clases que la componen. La figura 7.3 muestra el diagrama de secuencia para este caso de uso.

Una vez que los detalles de análisis están cubiertos, se procede a la realización en diseño de este caso de uso. La diferencia con el anterior diagrama de secuencia, será que el que se produzca en esta fase contendrá detalles sobre las clases que se crearán en nuestra aplicación, así como será dependiente del lenguaje de programación utilizado (en este caso, Java). Las figuras 7.4, 7.5 y 7.6 muestran el diagrama de secuencia producido en diseño. Dada su complejidad, ha sido dividido en tres trozos, donde cada uno es la continuación natural del anterior.

El actor usuario pulsa sobre la vista de la Portada, la cual manda un evento a su controlador que le pide a la máquina de estados que muestre la ventana principal de la interfaz. Una vez ahí, el actor usuario presiona sobre el botón de crear/editar diccionario, una vez hecho esto, la vista actual le manda un evento a su controlador, que a su vez le pide a la máquina de estados que cambie a la ventana de creación de diccionario.

Una vez se crea la vista de creación del diccionario, y su controlador asociado, dicho controlador creará un objeto diccionario vacío y lo almacenará de forma local. A continuación, solicitará a la clase Fuente de Datos, el diccionario, si existiera, que dicha clase tiene almacenado. Una vez obtenido, el controlador actualizará el diccionario que había creado inicialmente con los valores que contenga el obtenido de la Fuente de Datos.

Tras esto, el actor usuario pulsa el botón añadir variable, la vista envía un evento a su controlador, el cual obtiene los tres campos en cuestión, nombre, código y descripción de su vista asociada, comprueba si son correctos y los añade a su diccionario local.

Cuando el usuario ha finalizado de introducir variables, presiona el botón aceptar, que desencadena que la vista envíe un evento a su controlador, el cual actualiza el diccionario de la clase Fuente de Datos y le pide a la máquina de estados que vuelva a mostrar la ventana principal de la interfaz.

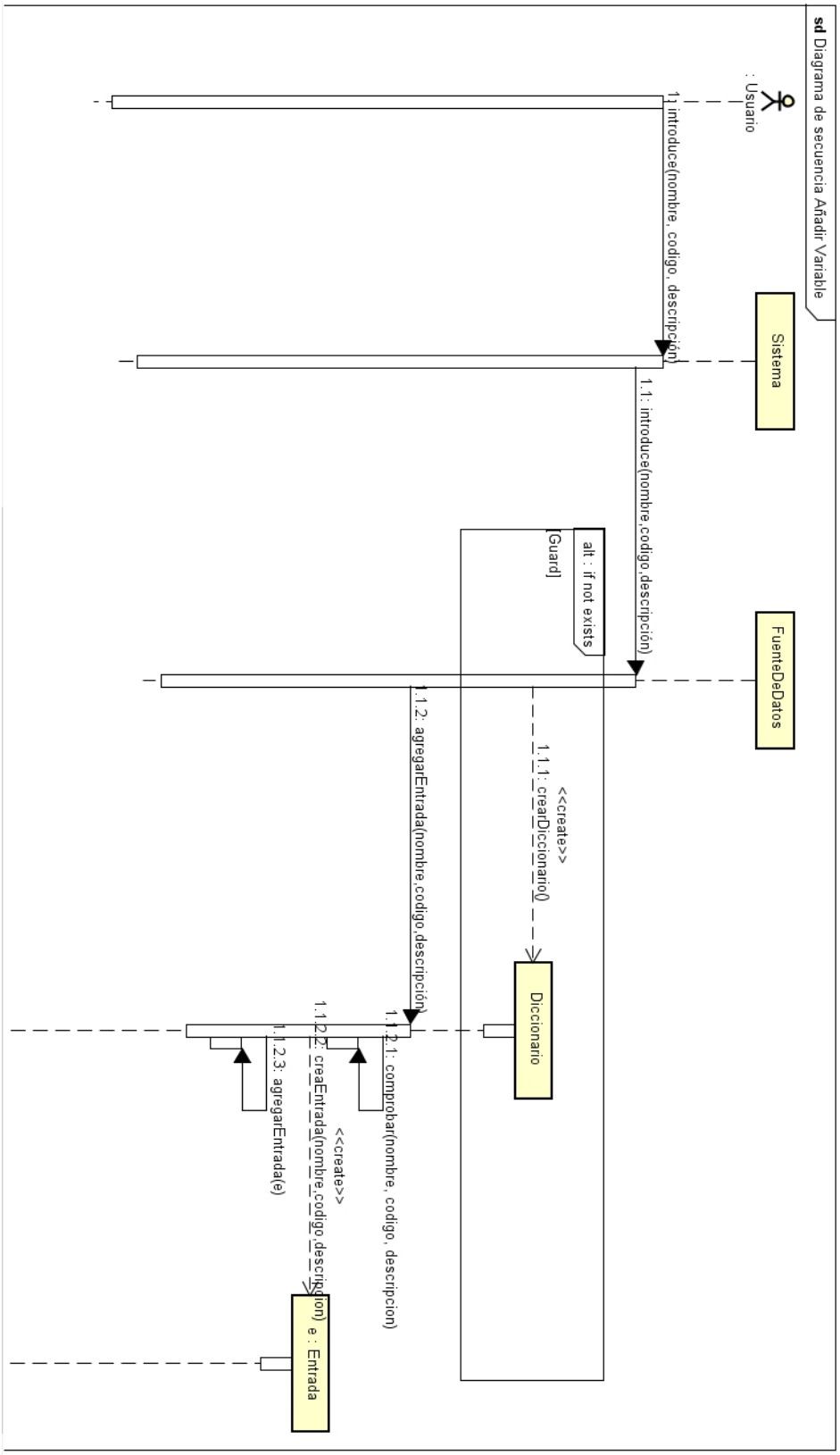

Figura 7.3: Realización en análisis del CU Añadir Variable

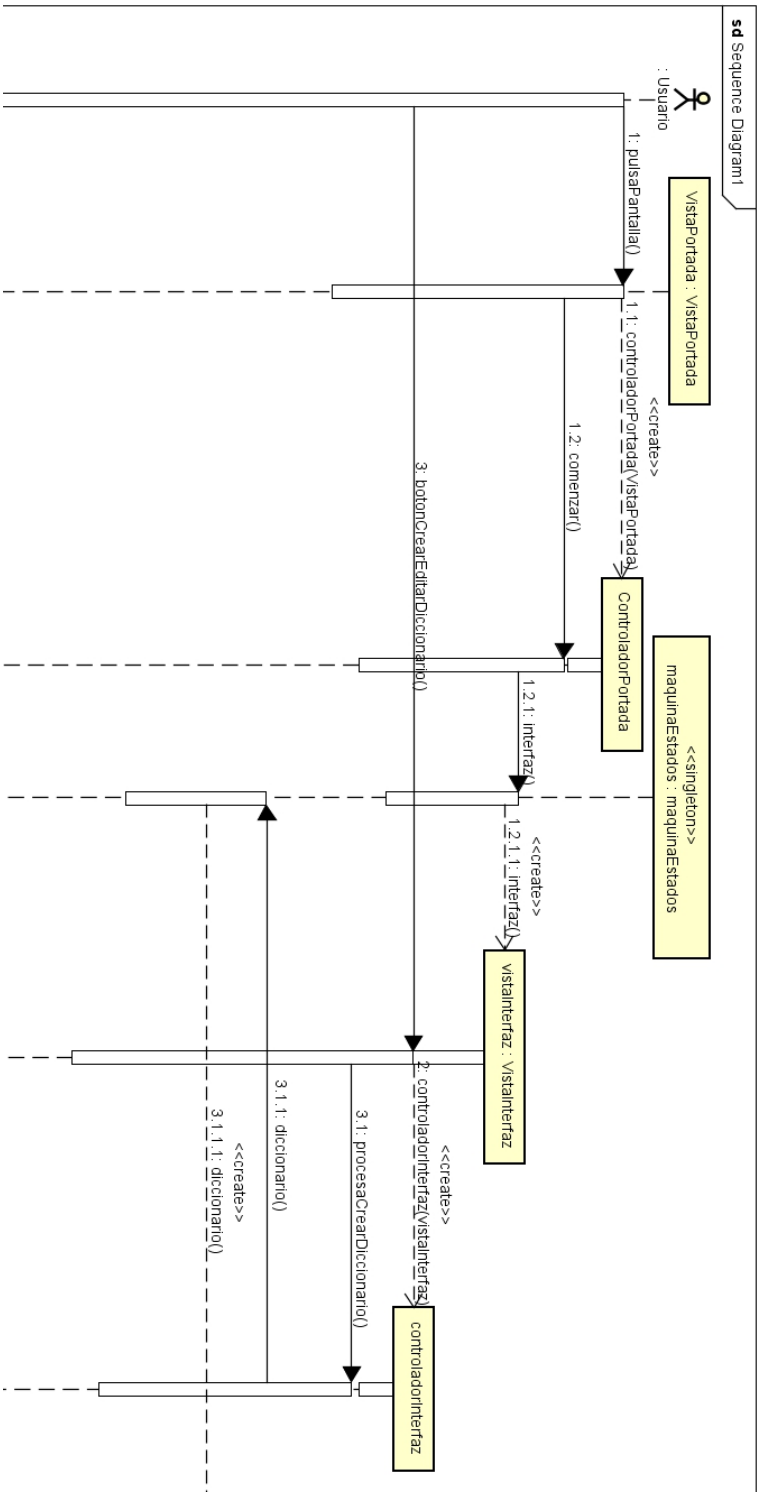

Figura 7.4: Realización en diseño del CU Añadir Variable parte 1

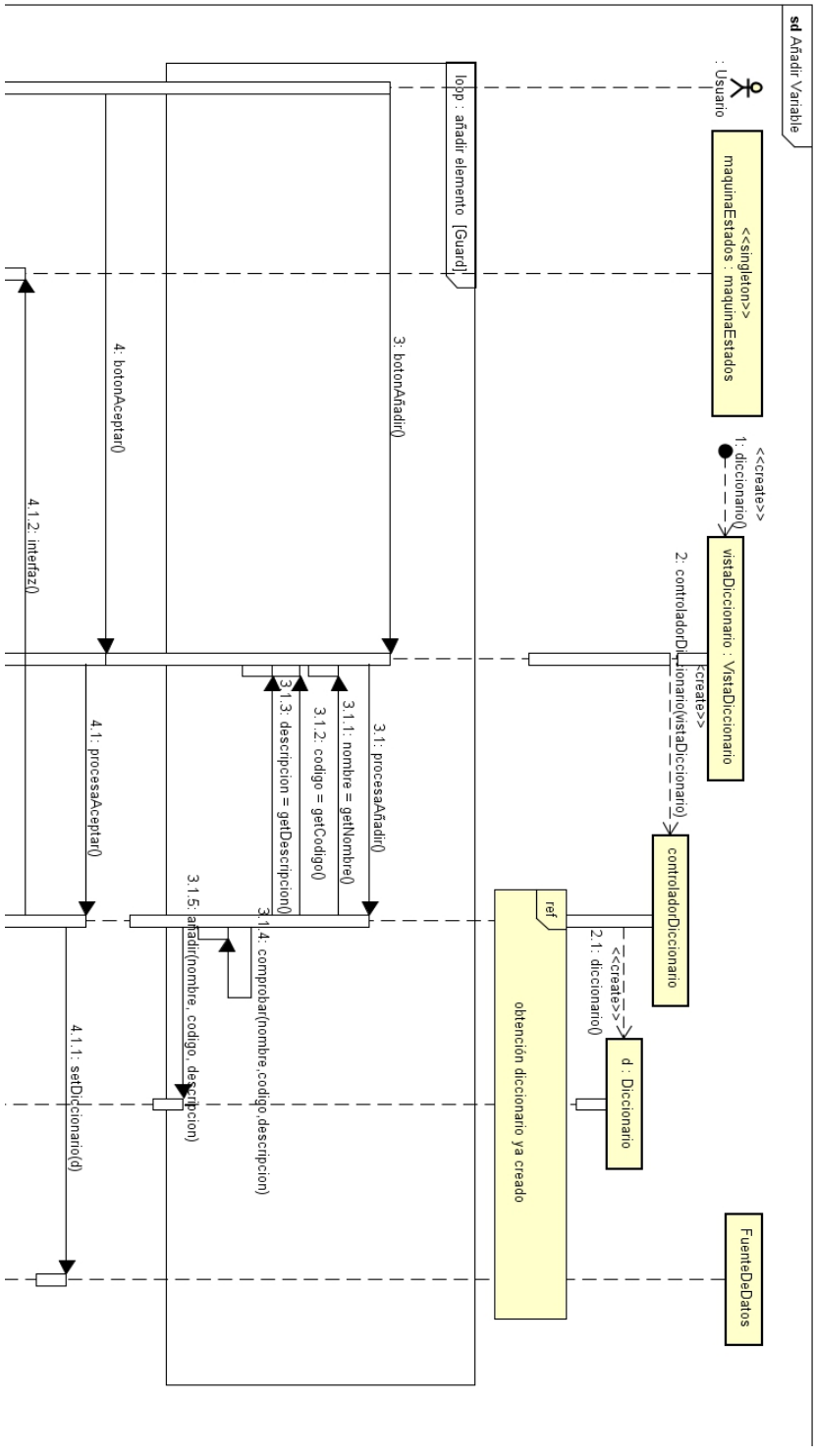

Figura 7.5: Realización en diseño del CU Añadir Variable parte 2

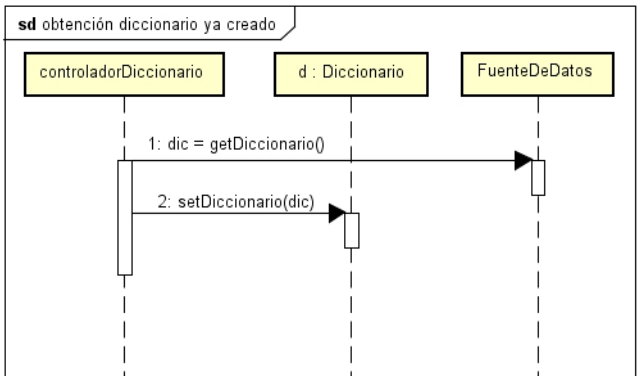

Figura 7.6: Realización en diseño del CU Añadir Variable parte 3

7.2.2. Caso de uso Crear Archivo de Parámetros

En primer lugar, se mostrará la realización del caso de uso en la fase de análisis. A este nivel no nos preocupamos de los detalles técnicos de la solución final ni de las clases que la componen. La figura 7.7 muestra el diagrama de secuencia para este caso de uso.

Una vez que los detalles de análisis están cubiertos, se procede a la realización en diseño de este caso de uso. La diferencia con el anterior diagrama de secuencia, será que el que se produzca en esta fase contendrá detalles sobre las clases que se crearán en nuestra aplicación, así como será dependiente del lenguaje de programación utilizado (en este caso, Java). Las figuras 7.8, 7.9 y 7.10 muestran el diagrama de secuencia producido en diseño. Dada su complejidad, ha sido dividido en tres trozos, donde cada uno es la continuación natural del anterior.

El actor usuario pulsa sobre la vista de la Portada, la cual manda un evento a su controlador que le pide a la máquina de estados que muestre la ventana principal de la interfaz. Una vez ahí, el actor usuario debe comenzar el caso de uso "Añadir visualización", ya que es necesario crear al menos una.

El actor usuario presionará el botón de añadir visualización, momento en el cual la vista le enviará un evento a su controlador, y este obtendrá todos los parámetros necesarios, es decir, el tipo, la medida, y las variables, creará una visualización con ellos, se la pasará a la clase Fuente de Datos, después pedirá a esa misma clase la lista de visualizaciones que tiene guardadas, y se la pasará a su vista asociada para que se actualice con ellas. Esto se repetirá tantas veces como el usuario desee.

Una vez finalizado ese caso de uso, se continúa con el de "Crear archivo de Parámetros", el actor usuario puede proceder de dos maneras. La primera es que pulse el botón crear/editar diccionario, momento en el cual se desencadenará el caso de uso "Crear o editar diccionario". Este caso de uso no se va a detallar debido a su parecido con el que se ha explicado en la sección anterior. La segunda forma de proceder es que presione el botón aceptar.

Si el actor usuario pulsa el botón aceptar, la vista enviará un evento a su controlador, el cual obtendrá todos los parámetros necesarios, véase la ruta de R, la ruta del archivo de datos, el separador, el formato de destino, la ruta de destino, el factor de elevación y el nombre de destino, actualizará la clase Fuente de datos con ellos y, tras comprobar que todos son correctos, creará los ficheros de texto con todos esos datos para, finalmente, realizar una llamada al programa de R utilizando la ruta en la que se encuentra, es decir, la ruta de R.

La variable conocida como factor de elevación hace referencia a una variable calculada en base al diseño de la encuesta y que permite realizar los cálculos de forma sencilla, sin tener que trabajar con dicho diseño, ya que es el factor que hay que aplicar a cada observación para poder obtener lo que se quiera. Más información de esta variable puede ser obtenida en la memoria anexa[3].

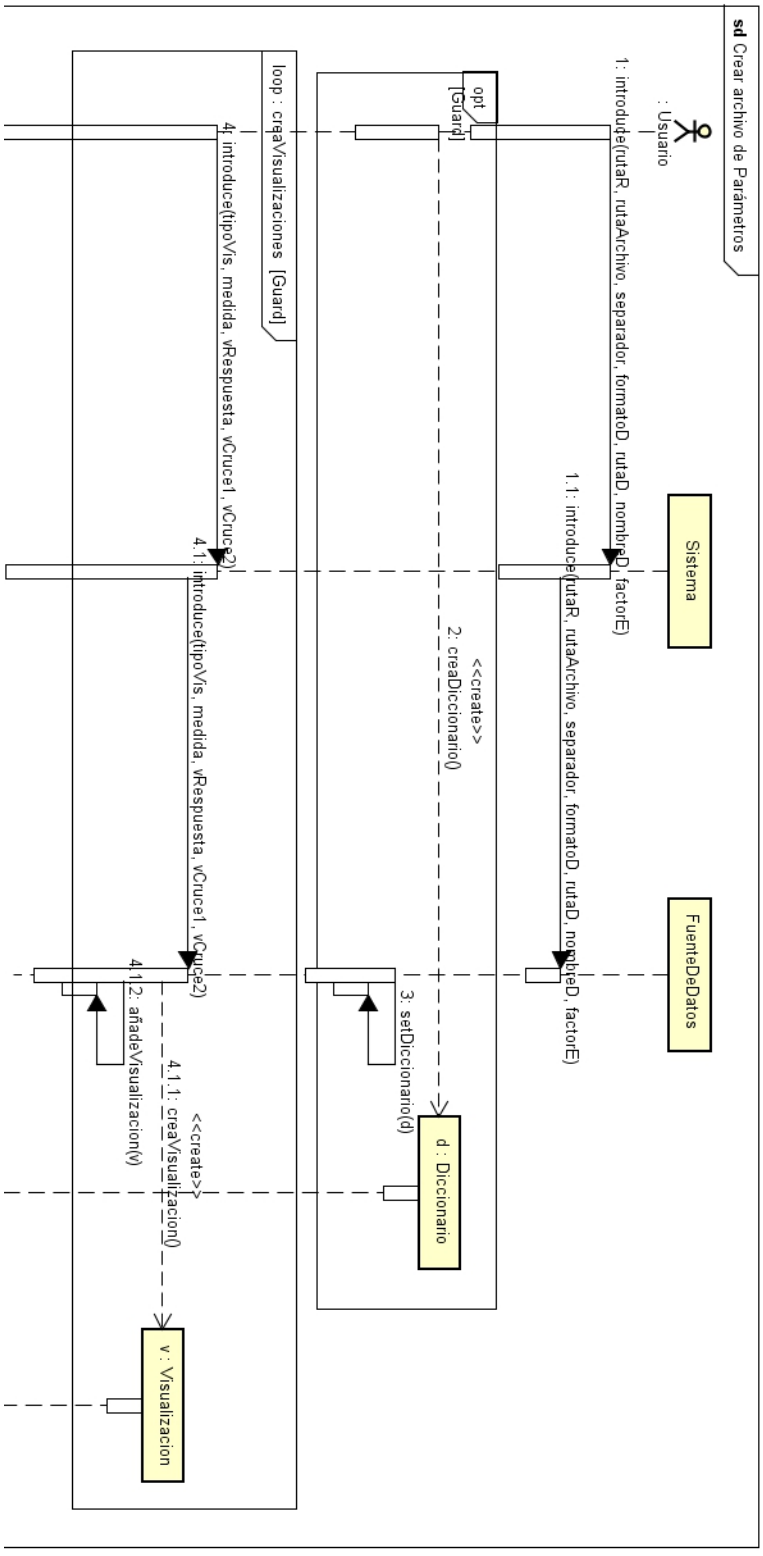

Figura 7.7: Realización en análisis del CU Crear Archivo de parámetros

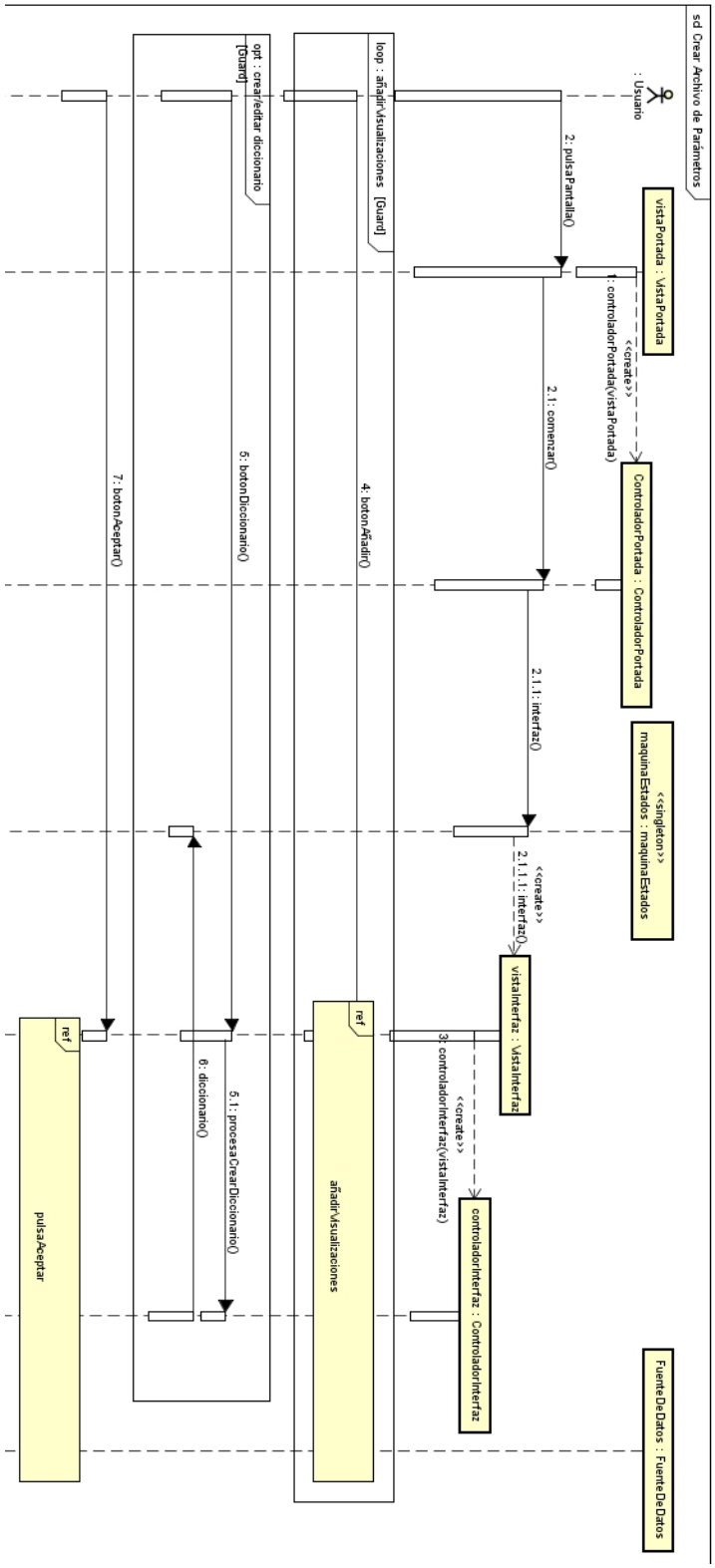

Figura 7.8: Realización en diseño del CU Crear Archivo de parámetros parte 1

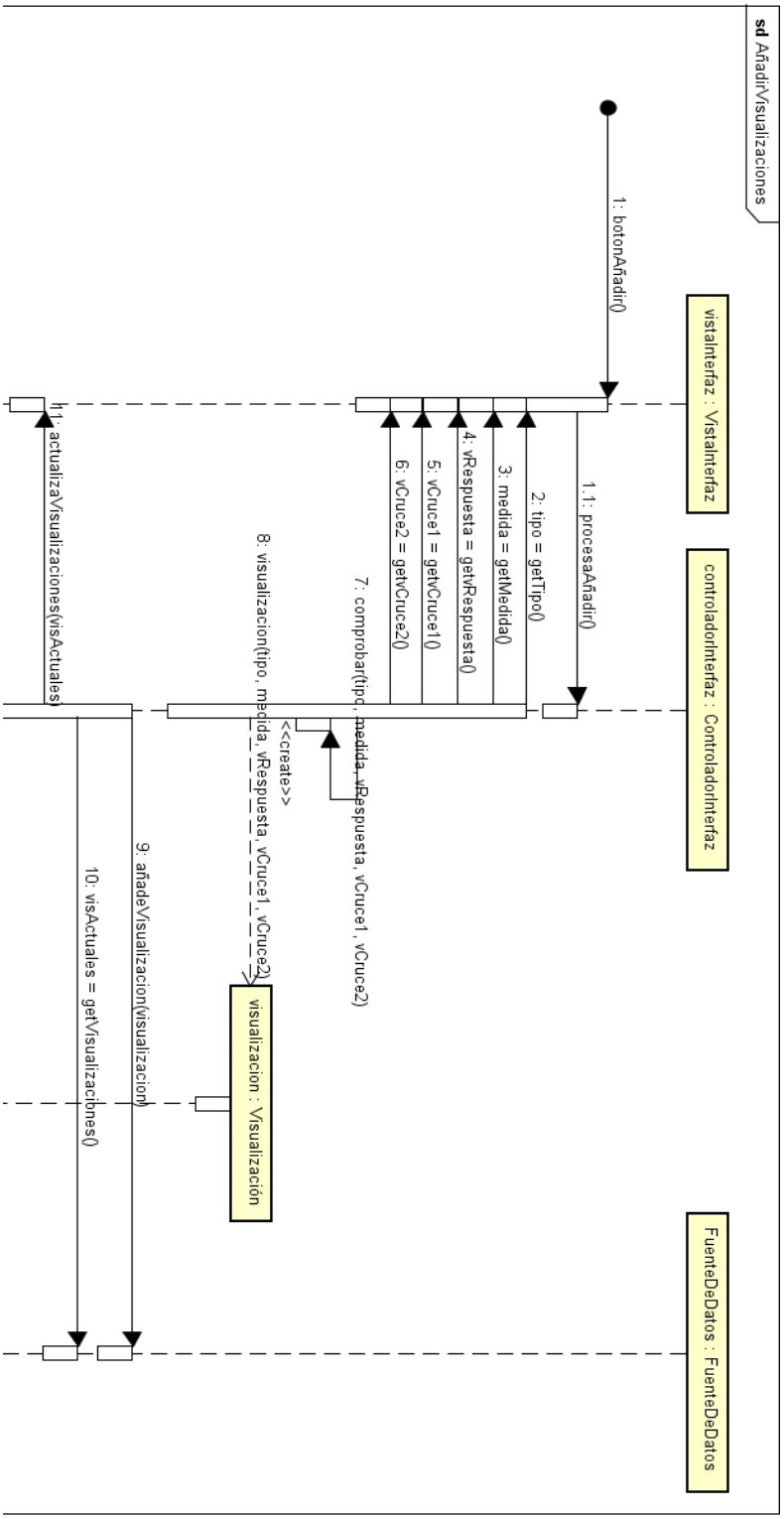

Figura 7.9: Realización en diseño del CU Crear Archivo de parámetros parte 2

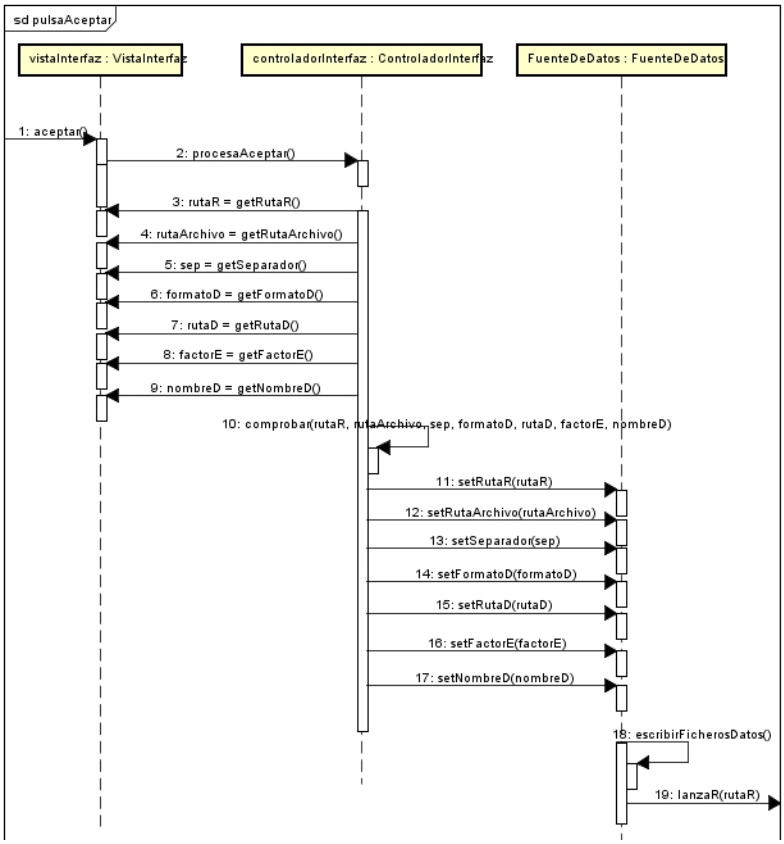

Figura 7.10: Realización en diseño del CU Crear Archivo de parámetros parte 3

7.3. Creación de la interfaz

A lo largo de esta sección se detallará en profundidad el proceso seguido para crear la interfaz, comenzando por los primeros bocetos, con los que se realizó una encuesta de usabilidad con diferentes usuarios para encontrar posibles fallos y problemas, así como para detectar aspectos bloqueantes. Después se mostrará el diseño final creado, junto con una nueva encuesta de usabilidad, ya realizada con la aplicación funcional, buscando encontrar lo mismo que con la anterior.

7.3.1. Primer boceto

Desde un principio se tenía más o menos claro toda la funcionalidad que debía poseer la interfaz, la cantidad de ventanas distintas que la compondrían y lo que tenía que haber en cada una. Con todo esto en mente, se creó el primer boceto de la aplicación, que más tarde se utilizaría para la realización de una encuesta de usabilidad para comprobar qué carencias tenía el diseño o qué cosas estaban bien pensadas.

Además de buscar defectos, se pretendía medir el tiempo que podía tardar un usuario cualquiera en realizar una serie de tareas que englobaran toda la funcionalidad de la que disponía la interfaz, incluso se esperaba encontrar posibles adiciones para completar aun más la misma.

A continuación, en las figuras 7.11 y 7.12 se muestra el primer boceto, ya pasado a limpio, que se utilizó para la encuesta antes mencionada.

La figura 7.11 muestra la ventana principal del programa. Como se ha explicado anteriormente cuenta con dos partes diferenciadas claramente.

- La zona de los parámetros en la cual se introducen una serie de valores listados a continuación.
    - Archivo: Ruta donde se encuentra el archivo de datos, al pulsar en los tres puntos situados a la derecha de este campo, se abrirá el explorador de archivos para seleccionarlo.
    - Separador del archivo: Lista desplegable para seleccionar el separador con el que cuenta el archivo de datos.
    - Formato de destino: Lista desplegable para seleccionar el formato en el que el usuario desea obtener las visualizaciones.
    - Diccionario de datos: Se trata de un botón que sirve para cambiar a la vista de creación del diccionario, junto a él existe una etiqueta que indicará si ya ha sido creado un diccionario o no.
    - Ruta de destino: Ruta donde queremos obtener el archivo con las visualizaciones, al igual que en el primer elemento, los tres puntos en la parte derecha del campo muestran el explorador de archivos, aunque en este caso solo se puede indicar un directorio.

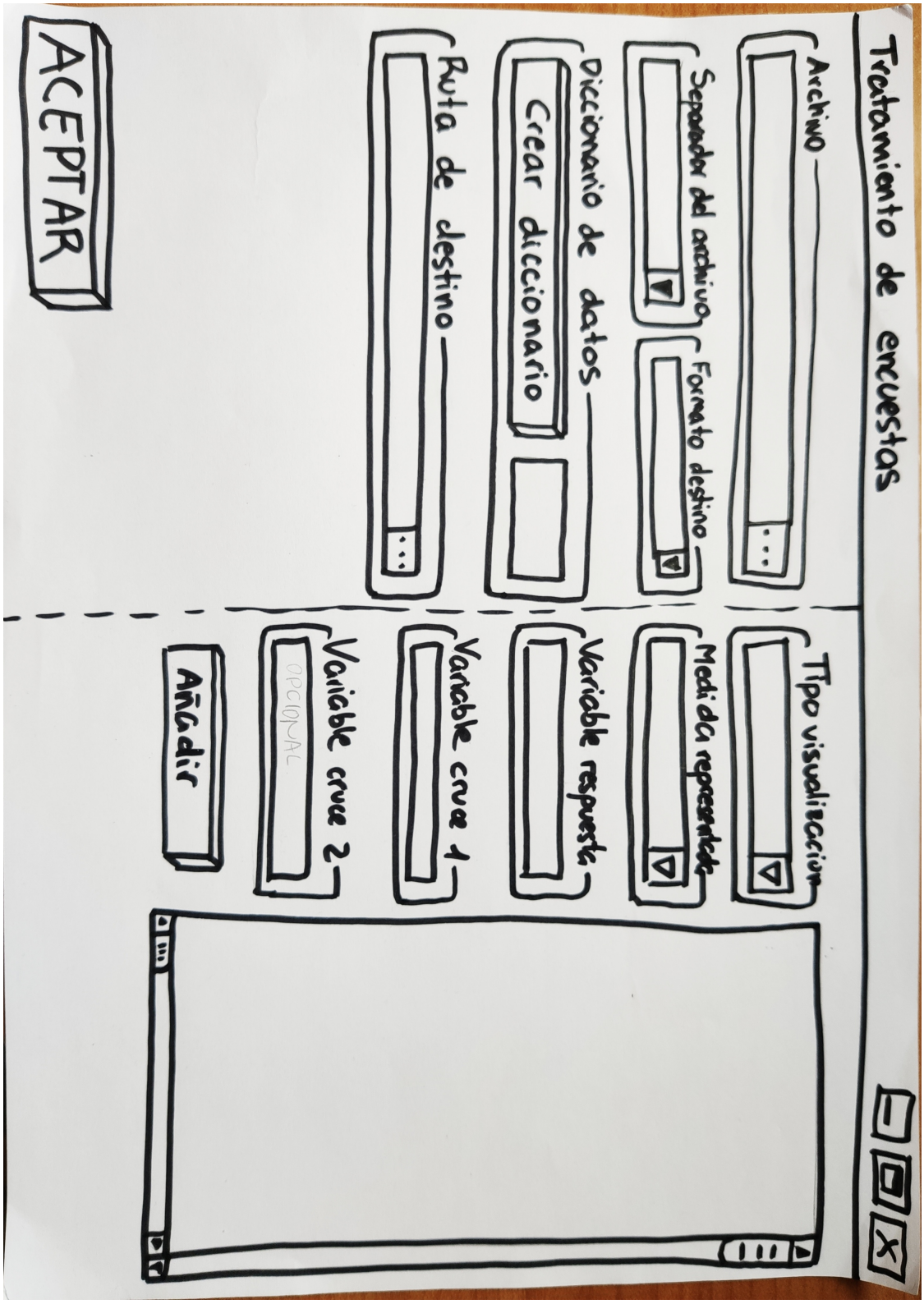

Figura 7.11: Primer boceto de la venta principal

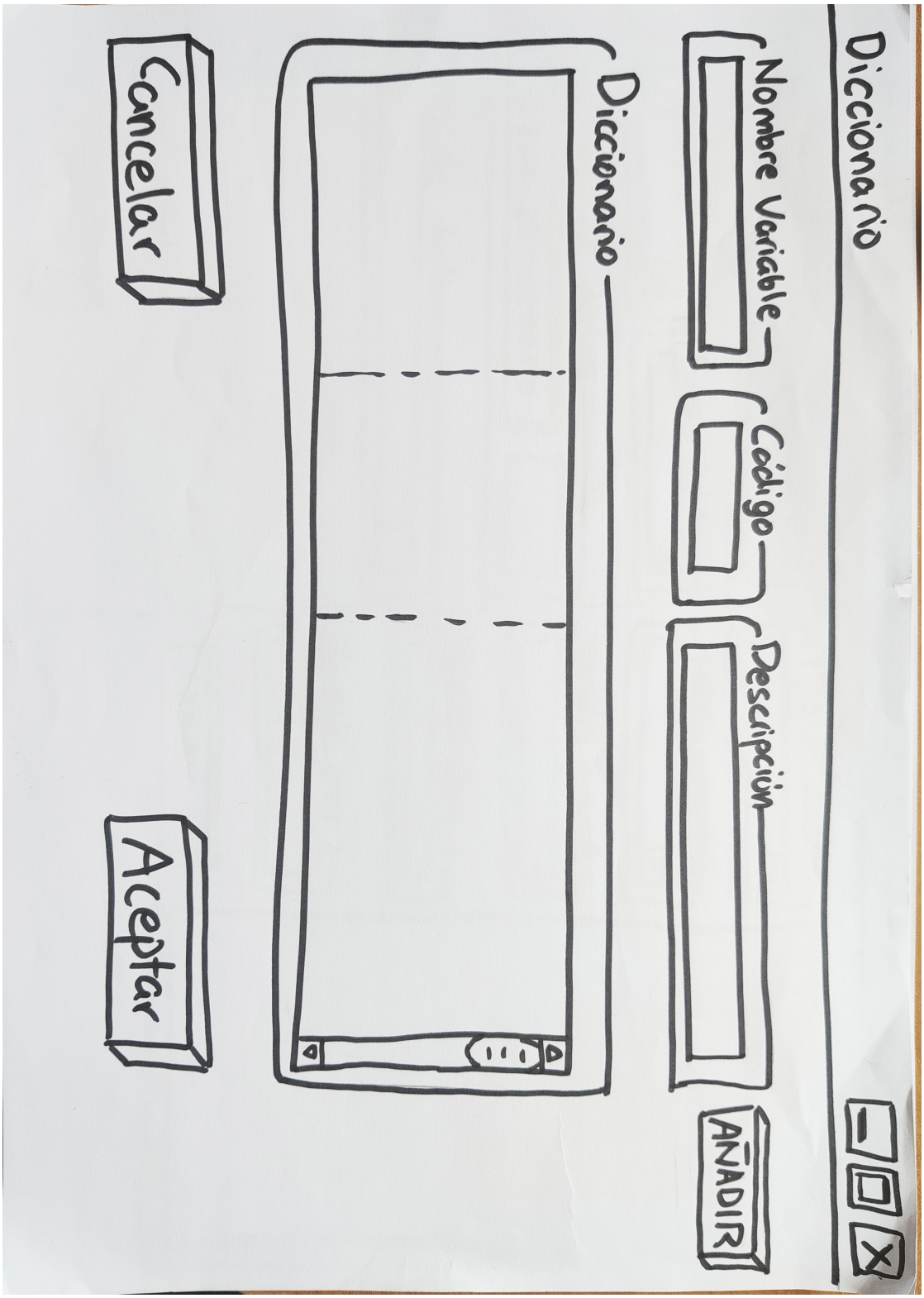

Figura 7.12: Primer boceto de la venta de creación del diccionario

    - Botón aceptar: Una vez indicados todos los parámetros necesarios, al pulsar este botón se crearán los ficheros de texto, los cuales leerá la aplicación principal y creará las visualizaciones indicadas.

- La zona de las visualizaciones. En ella el usuario podrá elegir qué tipo de visualización desea, así como algunas características de la misma. Además, existirá un panel de texto que mostrará las visualizaciones que el usuario haya indicado.

    - Tipo de visualización: Lista desplegable en la que se indicará el tipo de la visualización a elegir entre diferentes gráficos o tablas.
    - Medida representada: Lista desplegable para indicar la medida que representará el gráfico o tabla.
    - Variable respuesta: Campo para indicar la variable respuesta de la visualización.
    - Variable cruce 1: Campo para introducir la primera variable de cruce.
    - Variable cruce 2: Campo para indicar la segunda variable de cruce, este campo es opcional.
    - Botón añadir: Una vez indicadas todas las características de la visualización, este botón la añadirá a la lista y la mostrará en el panel derecho.

La figura 7.12 muestra la ventana de creación del diccionario, esta parte es totalmente opcional, y el usuario no tiene por qué rellenarla. En la parte superior de la ventana se encuentran los siguientes componentes.

- Nombre variable: El nombre de la variable escrito igual que aparezca en el archivo de datos.
- Código: Diferentes códigos que tiene dicha variable.
- Descripción: Significado del código introducido para la variable indicada.
- Botón añadir: Al pulsar este botón, si se han rellenado correctamente los valores anteriores, se añadirá la variable a la lista de las ya creadas y se mostrará en el panel situado en la parte inferior, además, se vaciarán los campos de "Código" y "Descripción" ya que una variable suele tener varios códigos distintos.

En la parte inferior de la ventana se encuentra el panel en el que se mostrarán las variables creadas y los típicos botones de "Aceptar" y "Cancelar". El primero de ellos guarda los cambios realizados en el diccionario y vuelve a la ventana principal, el segundo cancela los cambios y también vuelve a la ventana principal.

7.3.2. Encuesta inicial de usabilidad

Una vez se tenía el boceto inicial de la aplicación, se procedió a realizar una encuesta de usabilidad con diferentes personas, en la cual, dichas personas debían realizar una tarea sin ayuda. La idea es que se pueda realizar la prueba a cualquier persona, ya sea alguien con experiencia tecnológica, o sin ella. Con esta encuesta se pretendía refinar la interfaz, modificando los aspectos más confusos o los que llevaran más tiempo, además de recopilar las posibles ideas que se les ocurrieran a los participantes.

Al comenzar el ejercicio se les leía el siguiente párrafo a los usuarios.

La aplicación que vas a utilizar sirve para crear, de manera automática, una serie de visualizaciones, como pueden ser diferentes tipos de gráficos, así como tablas, partiendo de archivos de datos de encuestas. En ella se deben indicar una serie de parámetros básicos sobre el fichero en el que están los datos y también, las visualizaciones que deseas. A continuación, se te propondrá una tarea a realizar, paso por paso, para comprobar la usabilidad de la interfaz de usuario diseñada para la aplicación. En ella, se te irá explicando poco a poco qué hacer en cada momento y se tomará nota de las partes que te cuesten más. Al finalizar la tarea, se te harán una serie de preguntas para valorar tu opinión y realizar las modificaciones pertinentes en dicha interfaz.

Una vez dicho esto, se comenzaba inmediatamente a describir la prueba, la idea es que el usuario fuera realizando la tarea a medida que se le iba leyendo el enunciado.

7.3.3. Primera prueba

Primero se expondrá el "enunciado" de esta prueba y, después, se adjuntará una imagen de los resultados obtenidos.

Has obtenido un archivo de datos perteneciente a una encuesta sobre salarios en España y deseas crear dos tipos de gráficos y una tabla con el mismo.

Lo primero que tienes que hacer es indicar dónde se encuentra el archivo de datos a la aplicación. En este caso el archivo de datos es una hoja de Excel, en la que el separador de cada columna es el ***punto y coma*** *y, además, lo que tu deseas es obtener las visualizaciones en formato* ***CSV****.*

Como quieres que los nombres de los campos sean los adecuados, vas a crear un diccionario de datos para esta encuesta.

Sabes que solo necesitas 2 variables diferentes con 2 niveles cada una. La primera variable es ***TSexo****, y tiene dos códigos, el primero es* ***H****, que significa* ***Hombre****, el segundo es* ***M****, que significa* ***Mujer****. La segunda variable es* ***TNacionalidad****, también con dos códigos, el primero es* ***1*** *que significa* ***Español****, el segundo es* ***2****, que significa* ***Alemán****.*

Una vez aceptas la creación del diccionario, vuelves a la pantalla principal para indicar las visualizaciones que deseas.

Lo primero que quieres es un ***gráfico de barras horizontales****, que represente la* ***media*** *de la variable respuesta* ***SALARIO*** *frente al* ***SEXO*** *y a la* ***NACIONALIDAD****.*

*El segundo gráfico se trata de un **gráfico de sectores** que represente lo mismo. Por último, quieres una **tabla** en la que se mida el **total** de la variable **SALARIO** frente al **SEXO**. Una vez tienes las 3 visualizaciones, indicas la ruta de destino en la que quieres el CSV y terminas.*

Número de usuario	1	2	3	4
Problemas surgidos	Problema al identigicar la variable respuesta	Ninguno	Problema al saber en qué campo introducir cada variable en la parte de las visualizaciones	Confusión entre 'añadir' y 'aceptar' en la creación del diccionario
Tiempo tardado	4 minutos y 30 segundos	3 minutos y 10 segundo	4 minutos y 40 segundos	4 minutos y 15 segundos
Sugerencias de mejora	Ninguna	Indicar mejor la separación entre la zona de los parámetros y la de las visualizaciones	Ninguna	Intentar que sea más clara la diferencia entre 'añadir' y 'aceptar'

Número de usuario	5	6	7
Problemas surgidos	Ninguno	Se pulsó aceptar sin añadir ninguna visualización	Ninguno
Tiempo tardado	4 minutos y 18 segundos	4 minutos y 2 segundos	3 minutos y 1 segundo
Sugerencias de mejora	Añadir un botón para eliminar una visualización	Indicar mejor la separación entre la zona de los parámetros y la de las visualizaciones	Añadir botón de eliminar tanto en el diccionario como en las visualizaciones

Figura 7.13: Resultados de la primera encuesta

Como se puede ver en la figura 7.13, tanto los problemas surgidos como las sugerencias de mejora son cosas con poco impacto en el desarrollo. Esto puede ser debido a que la tarea a la que esta destinada la aplicación es sencilla e incluso gente poco familiarizada con la tecnología no ha tenido problemas a la hora de utilizar la "aplicación". Cabe destacar que este primer "test" no ha sido realizado a los clientes principales de la aplicación debido a que se ha preferido esperar a tener la interfaz funcional desarrollada para que pudieran probarla directamente y encontrar fallos o mejoras directamente con toda la funcionalidad implementada.

Una vez realizada la primera encuesta, se comenzó con el desarrollo de la aplicación. A continuación, se mostrará la primera versión de la interfaz sin entrar en su funcionamiento, el cual ya se detallará en secciones posteriores. Después de esto, se incluirá una nueva tabla con los resultados de la segunda encuesta de usabilidad y funcionalidad y, a raíz de ella, se detallarán los problemas surgidos y las modificaciones realizadas, teniendo en cuenta que esta vez sí que se incluirá a los clientes principales en la encuesta.

7.3.4. Primera versión de la interfaz

En las imágenes 7.14 y 7.15 se muestra la primera versión de la interfaz ya implementada.

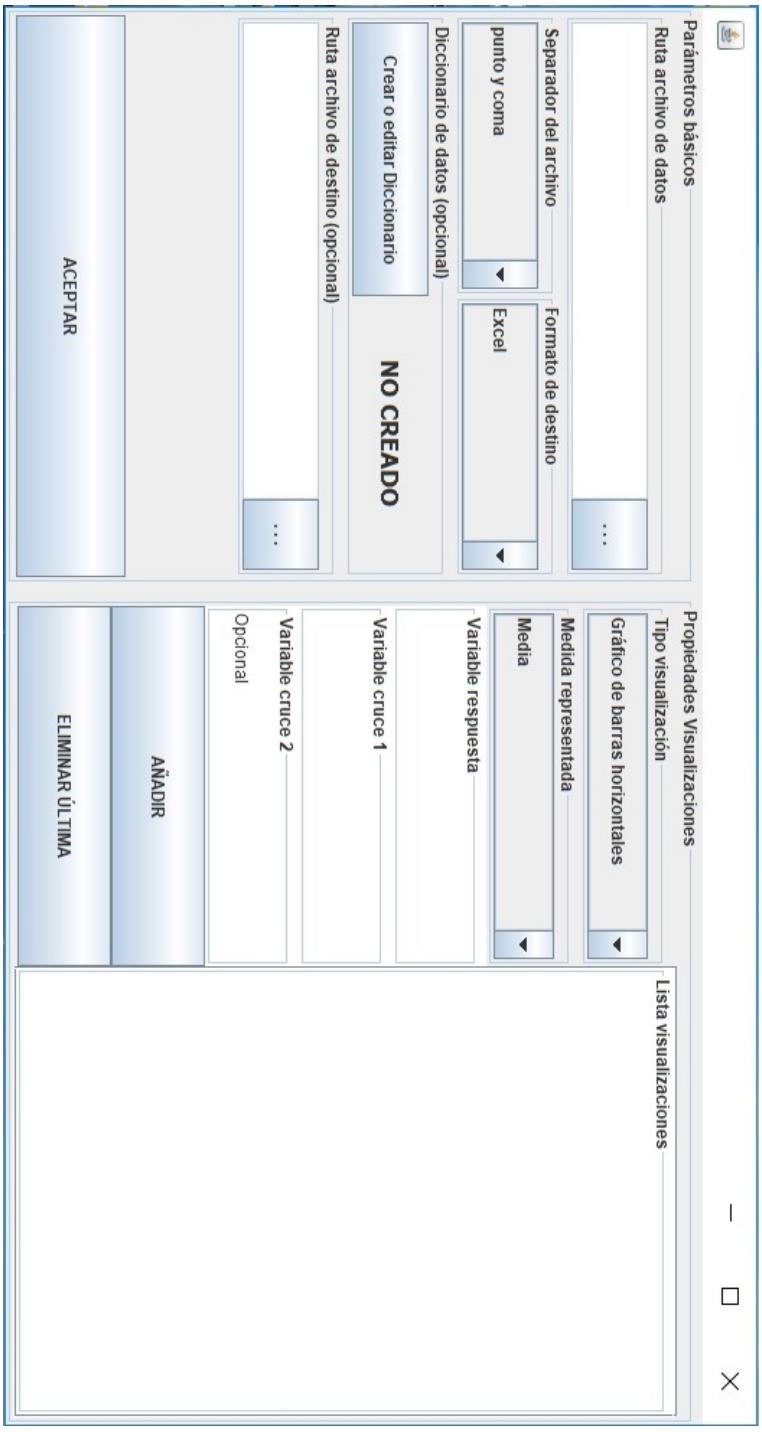

Figura 7.14: Primera versión de la ventana principal

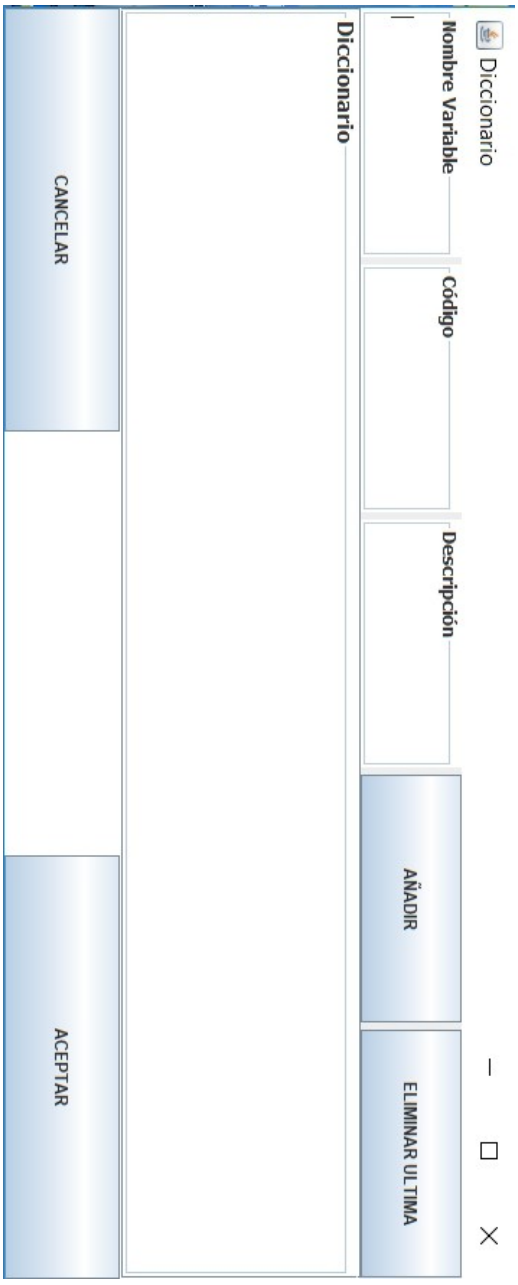

Figura 7.15: Primera versión de la ventana de creación del diccionario

Como puede verse, la mayor parte de las modificaciones han sido aplicadas a esta primera versión. Se han separado mediante marcos las zonas izquierda y derecha de la ventana principal y, además, se han añadido botones para eliminar tanto visualizaciones como variables en el diccionario. Una vez mostrada la interfaz, se va a explicar la realización de la segunda encuesta en la que, esta vez, se tenían que realizar dos tareas, la primera es la misma que ya se ha expuesto y, la segunda, intentaba centrarse en la nueva funcionalidad de eliminar visualizaciones y variables.

7.3.5. Segunda prueba y encuesta de usabilidad

Como ya se ha indicado, esta prueba era la misma que la anterior, pero realizada sobre la interfaz ya implementada, por lo que no se volverá a indicar el enunciado, pero sí los resultados, los cuales se muestran en la figura 7.16.

Número de usuario	1	2	3	4
Problemas surgidos	El diccionario no se creaba correctamente dentro de la interfaz	Ninguno	El fichero del diccionario no se creaba bien	Problema al crear visualizaciones
Tiempo tardado	4 minutos y 10 segundos	3 minutos y 1 segundo	3 minutos y 40 segundos	4 minutos y 15 segundos
Sugerencias de mejora	Que el botón cancelar del diccionario anule los cambios realizados	Ninguna	Modificar la apariencia, con colores etc.	Ninguna
Errores encontrados	Problemas con el diccionario	Ninguno	El botón aceptar no creaba bien el archivo del diccionario	Al crear una visualización, se modificaba la ya creada

Número de usuario	5	6	7
Problemas surgidos	Ninguno	Ninguno	Ninguno
Tiempo tardado	3 minutos	2 minutos y 40 segundos	2 minutos y 50 segundos
Sugerencias de mejora	Ninguna	Añadir colores	Ninguna
Errores encontrados	Ninguno	Ninguno	Ninguno

Figura 7.16: Resultados de la segunda encuesta

Con estos resultados, se ve que a partir de la cuarta persona que probó la aplicación, no se produjeron más errores, esto es así porque a medida que aparecían los errores, se iban solucionando. A continuación se detallan los errores surgidos y cómo han sido solventados.

- Problemas con el diccionario: Los primeros problemas surgidos estaban relacionados con la forma de crear el diccionario, y consistían en que, al crear una variable, y darle a aceptar, esta variable se guardaba correctamente, sin embargo, si se editaba el diccionario y se eliminaba dicha variable, la interfaz seguía indicando que el diccionario estaba creado, aunque no hubiera ninguna variable añadida.
- Problemas al crear el archivo del diccionario: A raíz de solventar el primer problema, se tocó parte del código relacionado con la exportación de los datos a los ficheros, y esto provocó que el fichero del diccionario no se creara correctamente.
- Problemas con las visualizaciones: Al crear una visualización, se modificaban todas las ya creadas.

Tras exponer los errores, se indicarán brevemente las soluciones que fueron aplicadas a los mismos.

- Problema del diccionario: Se ha modificado la forma de guardar las variables en el modelo y la forma en la que se creaban en el controlador, de esta forma todo funcionaba correctamente.
- Problemas de creación de ficheros: Se retocó la función que obtenía los datos del modelo.
- Problemas con las visualizaciones: Las variables en la clase estaban marcadas como estáticas y, por tanto, solo almacenaban el último valor indicado.

7.3.6. Tercera prueba y encuesta de usabilidad

Una vez se solucionaron todos los problemas encontrados tras la anterior prueba, se procedió con un último ejercicio de cara a probar la funcionalidad nueva de eliminar visualización, el enunciado de la prueba es el siguiente.

Esta segunda tarea será similar a la primera, pero centrada en probar el funcionamiento de los botones de eliminar.

Una vez más comienzas por indicar dónde se encuentra el archivo de datos, pero esta vez el separador de las columnas es el tabulador y el formato de destino es pdf. Seguidamente piensas en crear un diccionario. Indicas la variable ESTUDIOS, con código 1 y descripción "Básicos" y el código 2 y descripción "Avanzados". Aceptas la creación, pero después piensas que no necesitas el diccionario, por tanto lo editas, eliminas ambas variables y una vez más aceptas.

En cuanto a las visualizaciones, para esta segunda tarea necesitas un gráfico de líneas que mida la media de la variable PATRIMONIO frente a la variable ESTUDIOS. Además, quieres una tabla que muestre el porcentaje de la variable PARADOS frente a las variables SEXO y NACIONALIDAD.

Sin embargo, al crear esa tabla, te das cuenta de que está mal, por tanto la eliminas, y la vuelves a crear, esta vez midiendo solamente el numero de PARADOS frente al SEXO.

En este caso el ejercicio salió de manera correcta y no se adjuntarán los resultados ya que no aportan nada relevante.

Tras finalizar con las pruebas, se realizó una reunión con los tutores del Grado en Ingeniería Informática para que indicaran sugerencias, las cuales se listan en la siguiente sección.

7.3.7. Sugerencias de los tutores

A pesar de no realizarles las pruebas como tal, se les explicó la funcionalidad implementada hasta el momento y, en base a eso, ellos aportaron sus ideas.

- Organizar los ficheros de salida en diferentes directorios o, en caso de no ser necesarios, eliminarlos.
- Modificar el código para obtener varios datos de manera dinámica a través de un fichero de texto, como los formatos de salida, los tipos de gráfico, etc. Lo que se busca con esto es intentar, en la medida de lo posible, que se puedan añadir nuevas funcionalidades sin tener que tocar el código fuente de Java. Sin embargo, en el caso en el que se añada un nuevo formato de destino no contemplado anteriormente, habría que modificar el código ya que para comprobar si el informe a crear, ya existe en la ubicación indicada, es necesario conocer la extensión del mismo y, actualmente, solo hay contempladas algunas.
- Añadir un campo para que el usuario pueda indicar el nombre del fichero donde obtendrá las visualizaciones.
- Al pulsar "Aceptar" se notificará al usuario si existe un fichero con el mismo nombre que el que va a crear en la misma ubicación, para no perder información.
- Poder eliminar el elemento deseado en vez del último.

Tras esto, y antes de implementar los cambios, se ha realizado una reunión con el tutor del Grado en Estadística para obtener sus sugerencias también.

En este caso, la mayor parte de los temas tratados con el tutor, ha sido referentes a la aplicación en R, por lo que no se indicarán aquí. Sin embargo, sí que se han encontrado problemas a la interfaz, así como pequeñas modificaciones que se deben añadir.

- Para realizar la conexión entre Java y R, así como para ejecutar el programa principal, se debe tener R instalado y, además, hay que indicarle a la aplicación la ruta en la que se encuentra el archivo ejecutable de R. Para tratar esto, se creará una vista inicial, en la que se muestre la imágen de la universidad, el nombre de la aplicación, y en la que haya que indicar la ruta de R.

- A la hora de crear visualizaciones, es necesario el campo "Factor de elevación" por lo que se debe añadir un nuevo campo para indicar el nombre **exacto** de dicho campo para la encuesta a tratar.
- En el caso de los gráficos de sectores, solo se debería poder indicar **una** variable de cruce, por lo que si el usuario indica que desea este tipo de visualización, se deberá deshabilitar el campo correspondiente a la segunda variable de cruce.
- Añadir dos nuevos formatos de obtención de resultados, en este caso, **html y pptx**.
- Debido a las complicaciones que supone la creación de un único archivo pdf con todas las visualizaciones, de momento se eliminará de los formatos de salida actuales.

En las imágenes 7.17, 7.18, 7.20, 7.21 y 7.19 se muestra el resultado obtenido tras aplicar todas las modificaciones sugeridas por los tutores de ambos grados.

Figura 7.17: Interfaz final, portada

Cabe destacar la adición de tres nuevas ventanas, la portada, la de alerta y la de espera, de las cuales, la primera se abrirá cuando inicie la aplicación, la segunda, en el caso en el que el archivo solicitado ya exista en la ubicación que indique el usuario y la última, tras pulsar aceptar, para dar retroalimentación al usuario.

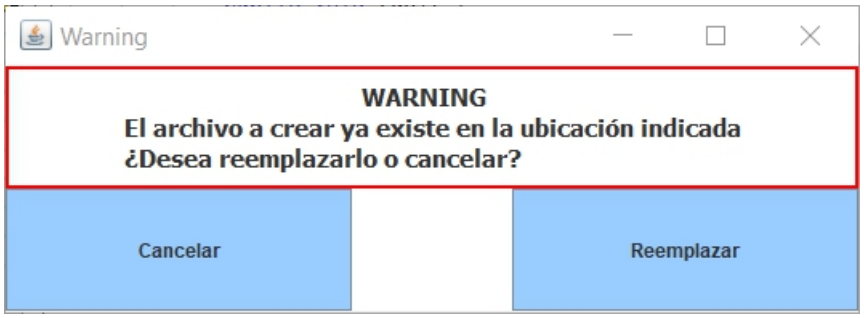

Figura 7.18: Interfaz final, ventana alerta

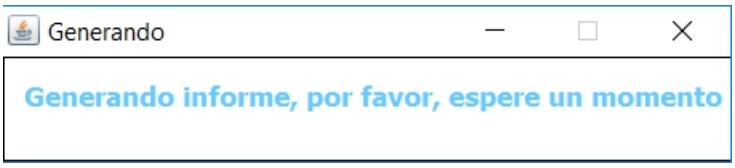

Figura 7.19: Interfaz final, ventana espera

Otra de las novedades es la desaparición de los botones de eliminar variable o visualización, que han sido sustituidos por una etiqueta que indica que haciendo doble click, se elimina el elemento deseado. Además de esto, se han añadido algunos nuevos campos en la ventana principal y se han modificado los colores de la aplicación completa. El resto de cambios indicados anteriormente también han sido incluidos y se explicará su funcionamiento en posteriores secciones.

7.4. Diseño detallado

A continuación se van detallar todas las clases que conforman la aplicación final, explicando cual es su función y, en algunos casos, mostrando un ejemplo de salida esperada. La sección se va a estructurar por paquetes, de la misma forma que se ha desarrollado la interfaz.

7.4.1. Clases comunes

El primer paquete que se va a comentar, contiene las clases que pueden ser usadas por cualquier otra, en este caso son las tres siguientes.

7.4.1.1.0 Diccionario

Esta clase tiene los siguientes tres atributos.

- **Nombres**: ArrayList de String que contiene los nombres de las variables añadidas al diccionario.

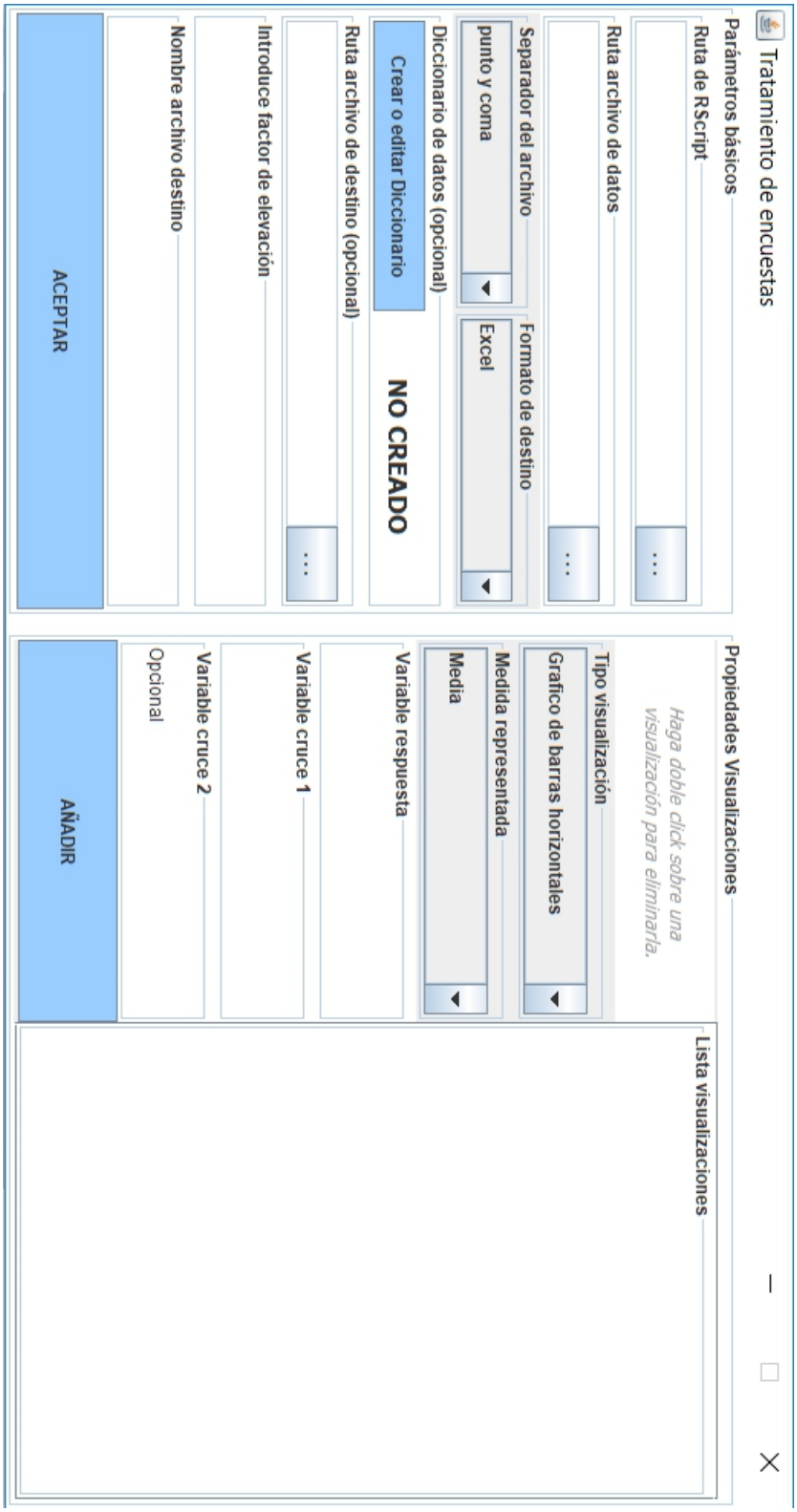

Figura 7.20: Interfaz final, ventana principal

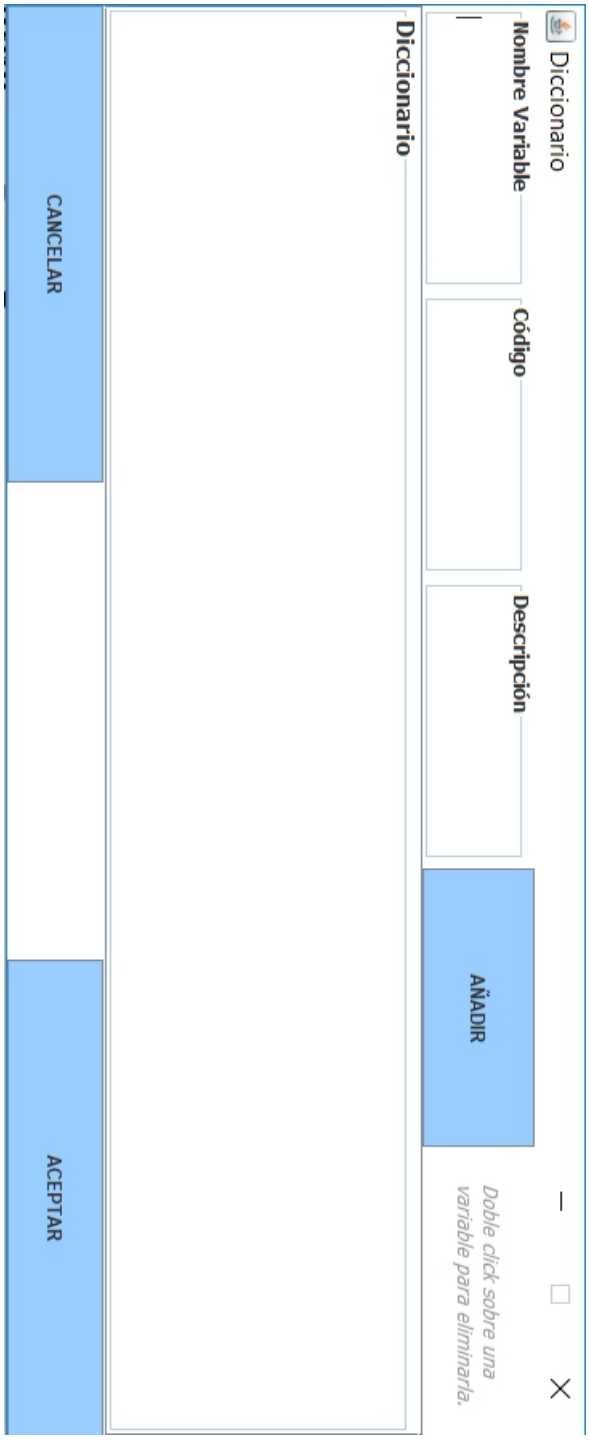

Figura 7.21: Interfaz final, ventana diccionario

- **Códigos**: ArrayList de String que contiene los códigos de las variables añadidas al diccionario.
- **Descripciones**: ArrayList de String que contiene las descripciones de las variables añadidas al diccionario.

En cuanto a las funciones que la forman, se listan a continuación.

- **Diccionario**: constructor por defecto de la clase.
- **Getters**: los correspondientes getters para cada atributo.
- **Setters**: en este caso los setters son funciones que añaden elementos a los ArrayList.

7.4.1.2.0 Visualización

Esta clase tiene los siguientes cinco atributos.

- **Tipo**: String que contiene el tipo de la visualización creada.
- **Medida**: String que contiene la medida que va a representar la visualización creada.
- **vRespuesta**: String que contiene la variable respuesta representada en la visualización.
- **vCruceUno**: String que contiene la primera variable de cruce representada en la visualización.
- **vCruceDos**: String que contiene la segunda variable de cruce representada en la visualización.

En cuanto a las funciones que la forman, son las que siguen.

- **Visualización**: constructor por defecto de la clase.
- **Getters**: los correspondientes getters para cada atributo.

7.4.1.3.0 Imagen

Se trata de una clase sin atributos, cuya única función es insertar la imagen de la escuela en la portada de la aplicación. Está formada por una única función que obtiene la imagen, adapta sus dimensiones y la pinta.

7.4.2. Modelo

El siguiente paquete a comentar es el que contiene al modelo, en este caso está formado por una única clase, la cual se detalla a continuación.

7.4.2.1.0 FuenteDeDatos

Se trata de una de las clases más importantes de la aplicación, debido a que es la que contiene todos los atributos que serán exportados a los ficheros de texto que leerá la aplicación de R para realizar los informes. Dichos atributos son los siguientes.

- **rutaOrigen**: String que contiene la ruta donde se encuentra el archivo de datos de la encuesta.
- **separador**: String que contiene el separador que tiene el archivo de datos.
- **formatoDestino**: String que contiene el formato en el que el usuario desea obtener el informe final.
- **rutaDestino**: String que contiene la ruta en la que el usuario desea alojar el informe final.
- **visualizaciones**: ArrayList de Visualizaciones formato por todas las visualizaciones que el usuario desea.
- **diccionario**: Diccionario formado por todas las variables que el usuario ha introducido.
- **existeDiccionario**: Boolean que indicará a la aplicación de R si se ha creado un diccionario o no.
- **rutaR**: String que contiene la ruta en la que se encuentra el ejecutable de R.
- **factorE**: String que contiene el nombre exacto del factor de elevación asociado a la encuesta con la que se está trabajando.
- **nombreD**: String que contiene el nombre que el usuario quiere dar al informe final.

En cuanto a las funciones que conforman la clase, son las listadas a continuación.

- **FuenteDeDatos**: constructor por defecto de la clase.
- **Getters**: los correspondientes getters para cada atributo de la clase.
- **Setters**: los correspondientes setters para cada atributo.
- **añadeVisualizaciones**: añade la visualización recibida como parámetro al ArrayList de visualizaciones.
- **eliminaVisualizaciones**: elimina la visualización con el índice recibido como parámetro del ArrayList de visualizaciones.

7.4.3. Main

Se trata del paquete que contiene el lanzador de la aplicación. También está formado por una única clase, indicada a continuación.

7.4.3.1.0 Main

Clase encargada de lanzar la aplicación. Esta compuesta por un único atributo.

- **maquinaEstados**: la máquina de estados que se encargará de cambiar entre las diferentes ventanas de la aplicación.

También está formada por una única función.

- **main**: Esta función crea la máquina de estados y la lanza.

7.4.4. Máquina de estados

Este paquete contiene la máquina de estados, la encargada de cambiar entre las diferentes ventanas de la aplicación.

7.4.4.1.0 MaquinaEstados

Se trata de la clase encargada de ir alternando entre las diferentes vistas de la interfaz, así como de crear el modelo inicialmente. Se compone de tres atributos.

- **mimodelo**: modelo de la aplicación.
- **currentState**: JFrame que contiene el estado actual, es decir, vista que esta mostrando en cada momento la interfaz.
- **war**: JFrame que contiene la ventana de warning que se mostrará en determinadas situaciones.
- **war**: JFrame que contiene la ventana de espera que se mostrará en determinadas situaciones.

Las funciones que conforman esta clase son las siguientes.

- **start**: función que crea la vista inicial de la aplicación.
- **diccionario**: función que destruye la vista que se esté mostrando en un determinado momento y crea la ventana del diccionario.

- **interfaz**: función que destruye la vista que se esté mostrando en un determinado momento y crea la ventana principal de la interfaz.
- **portada**: función que destruye la vista que se esté mostrando en un determinado momento y crea la ventana de la portada de la aplicación.
- **warningON**: función que muestra la ventana de warning.
- **warningOFF**: función que destruye la ventana de warning que se esté mostrando.
- **getModelo**: función que devuelve el modelo actual de la aplicación con sus atributos.
- **generando**: función que mostrará la ventana de espera y, tras 25 segundos, la destruirá.

7.4.5. InterfazPortada

Se trata del primer paquete que contiene una de las ventanas de la aplicación, se compone de dos clases, la vista y su controlador.

7.4.5.1.0 VistaPortada

Primer JFrame Form que se va a describir en la sección, encargado de crear la Portada de la aplicación. Contiene un único atributo.

- micontrol: controlador de esta vista que se encargará de realizar cualquier operación que desee el usuario, así como de actualizar el modelo en el caso de que fuera necesario.

En cuanto a las funciones, son las que siguen.

- **VistaPortada**: contructor de la clase, que inicializa los componentes de la ventana, crea su controlador y llama al método encargado de adaptar la imágen.
- **añadeImagen**: función que recibe la imagen a insertar y la incrusta en el panel principal.

Además de las funciones normales, existen unas especiales, encargadas de gestionar los eventos que suceden en la vista, dichas funciones no se van a detallar puesto que no se cree necesario.

7.4.5.2.0 ControladorPortada

Controlador de la vista descrita anteriormente, funciona como el nexo entre la vista y el modelo y, además, se encarga de gestionar las operaciones realizadas por el usuario sobre la vista. Está compuesto de un solo atributo.

- **mivista**: vista que gestiona el controlador.

Las funciones que lo componen son las siguientes.

- **ControladorPortada**: constructor de la clase.
- **devuelveImagen**: función que se encarga de crear la imagen deseada y decirle a la vista que la inserte.
- **comenzar**: función que le pide a la máquina de estados que cree la ventana principal de la aplicación.

7.4.6. InterfazDiccionario

Contiene los elementos necesarios para mostrar y manejar la ventana de creación del diccionario de variables, también está compuesto de dos clases, la vista y el controlador.

7.4.6.1.0 VistaDiccionario

Se trata del JFrame Form encargado de crear la ventana del Diccionario, al igual que la anterior vista, contiene un único atributo.

- **micontrol**: controlador de esta vista que se encargará de realizar cualquier operación que desee el usuario, así como de actualizar el modelo en el caso de que fuera necesario.

En lo que respecta a las funciones que la componen, se listan a continuación.

- **VistaDiccionario**: constructor de la clase, que inicializa los componentes de la ventana, crea su controlador y llama al método encargado de actualizar la vista en el caso de que ya haya alguna variable creada anteriormente.
- **Getters**: los correspondientes getters encargados de recuperar la información de los campos de texto de la ventana.
- **nuevoRegistro**: función encargada de limpiar los campos de texto para que sea más sencillo introducir una nueva variable. Solamente se limpian los códigos y las descripciones, esto es así ya que una misma variable puede tener más de un código. Esta función solo se ejecuta cuando el usuario pulsa el botón de añadir nueva variable.

- **limpiaNombre**: función que limpia el campo de texto correspondiente con el nombre de la variable y restaura el color de la fuente a negro, solo se ejecuta cuando el usuario hace click en el campo.
- **limpiaCodigo**: función que limpia el campo de texto correspondiente con el código de la variable y restaura el color de la fuente a negro, solo se ejecuta cuando el usuario hace click en el campo.
- **limpiaDescripcion**: función que limpia el campo de texto correspondiente con la descripción de la variable y restaura el color de la fuente a negro, solo se ejecuta cuando el usuario hace click en el campo.
- **errorNombre**: función que se lanza cuando se produce un error en el nombre de la variable, muestra la palabra "Error" en el campo de texto y cambia el color de la fuente a rojo.
- **errorCodigo**: función que se lanza cuando se produce un error en el código de la variable, muestra la palabra "Error" en el campo de texto y cambia el color de la fuente a rojo.
- **errorDescripción**: función que se lanza cuando se produce un error en el código de la variable, muestra la palabra "Error" en el campo de texto y cambia el color de la fuente a rojo.
- **actualizaDiccionario**: función encargada de actualizar el campo de texto de la ventana con todos los elementos ya incluidos en el diccionario del modelo, separados por una línea discontinua.

Además de estas funciones, existen otras ocho funciones encargadas de capturar determinados eventos y realizar las acciones pertinentes. La única de ellas que es más relevante es la función de eliminar variables, la cual debe capturar la posición relativa del ratón cuando el usuario hace click sobre la ventana de texto y, en el caso de que se produzcan dos clicks seguidos, avisar al controlador de que realice las acciones oportunas.

7.4.6.2.0 ControladorDiccionario

Controlador asociado a la vista del Diccionario, encargado de gestionar las operaciones realizadas por el usuario sobre la vista. Está compuesto por dos atributos.

- **mivista**: vista gestionada por el controlador.
- **dic**: diccionario local en el que se añadirán todas las variables que desee el usuario y, que al terminar, se traspasará al modelo.

Las funciones incluidas en esta clase están descritas a continuación.

- **ControladorDiccionario**: constructor de la clase.
- **procesaCancelar**: función encargada de limpiar el diccionario local y pedir a la máquina de estados que vuelva a la ventana principal.
- **procesaAñadir**: función que comprueba si existen errores en los campos de texto y, en caso de que no, añade los campos al diccionario local y le pide a su vista que se actualice. Los errores contemplados son los siguientes.
  - Que el nombre de la variable tenga una longitud menor que 3. Tras observar diferentes encuestas, se ha concluido que 3 es el tamaño mínimo que puede tener el nombre de una variable.
  - Que el código introducido no este en blanco.
  - Que la descripción de la variable tenga una longitud menor que 3. El motivo es el mismo que en el campo de nombre.
- **limpiaNombre**: función encargada de pedirle a la vista que limpie el campo del nombre de la variable.
- **limpiaCodigo**: función encargada de pedirle a la vista que limpie el campo del código de la variable.
- **limpiaDescripcion**: función encargada de pedirle a la vista que limpie el campo de la descripción de la variable.
- **procesaAceptar**: función que se ejecuta cuando el usuario pulsa el botón de aceptar y que se encarga de comprobar si el diccionario local no está vacío y, en caso de que no lo esté, actualiza el modelo con él y le pide a la máquina de estados que vuelva a la ventana principal.
- **actualizar**: función que se ejecuta cada vez que se abre la ventana del diccionario y que se encarga de traer el diccionario que exista en el modelo, actualizar el local con él y decirle a la vista que se actualice con las variables que existan.
- **eliminaPorPos**: función que se ejecuta cuando el usuario hace doble click en una variable existente para eliminarla y que se encarga de, en función del índice que calcula la función auxiliar "obtenerIndice", se elimine la variable deseada y se actualice la vista.
- **obtenerIndice**: función auxiliar que obtiene el índice de la variable que desea eliminar el usuario en base a la posición relativa del ratón. Se ha tomado como base que cada variable mostrada en la ventana tiene una altura de 44 píxeles, a excepción del primer elemento que tendrá una altura de 55 píxeles debido a la estructura del panel de texto. De esta forma, un ejemplo de salida esperada de esta función se muestra en la figura 7.22. La forma de trabajar de la función es la siguiente.

```
Procedimiento ObtenerIndice (y)
        indice ← 0
        bandera ← false
        Si (y > 134) entonces
                ref ← 134
                Mientras (!bandera) hacer
                        Si(y<=ref) entonces
                                bandera ← true
                        Si no
                                indice += 1
                                ref += 126
                        Fin si
                Fin mientras
        Fin si
        Devolver indice
Fin procedimiento
```

Coordenada y	Salida esperada	Salida obtenida
96	1	1
150	3	3
20	0	0

Figura 7.22: Ejemplo salida esperada función “obtenerIndice”

7.4.7. InterfazUsuario

Se trata del último paquete de la aplicación, contiene los elementos necesarios para mostrar y manejar la ventana de creación del diccionario de variables, en este caso, está compuesto de tres clases, dos vistas y un controlador.

7.4.7.1.0 VistaInterfaz

Se trata del JFrame Form encargado de crear la ventana principal de la aplicación, contiene un único atributo.

- **micontrol**: controlador de esta vista que se encargará de realizar cualquier operación que desee el usuario, así como de actualizar el modelo en el caso de que fuera necesario.

En lo que respecta a las funciones que la componen, se listan a continuación.

- **VistaInterfaz**: constructor de la clase, que inicializa los componentes de la ventana, crea su controlador y llama a los métodos encargados de actualizar la vista con las correspondientes visualizaciones.
- **Getters**: los correspondientes getters encargados de recuperar la información de los campos de texto de la ventana.
- **Setters**: los correspondientes setters para algunos de los campos de texto. En esta ventana están incluidos por si el usuario introduce algún campo, y cambia a la ventana del diccionario; de esta manera, cuando vuelva a la ventana principal, esos campos introducidos siguen estando.
- **actualizaVisualizaciones**: función encargada de actualizar el campo de texto en el que se listan las visualizaciones, con las que el usuario haya creado ya.
- **limpiaCruce2**: función que limpia el campo de texto de la variable de cruce 2.
- **limpiaCruce1**: función que limpia el campo de texto de la variable de cruce 1.
- **limpiaRespuesta**: función que limpia el campo de texto de la variable respuesta.
- **limpiaElevacion**: función que limpia el campo de texto de la variable de elevación.
- **errorRespuesta**: función que escribe “Falta la variable respuesta” en el campo correspondiente y modifica el color de la fuente a rojo.
- **errorCruce1**: función que escribe “Falta la variable de cruce” en el campo correspondiente y modifica el color de la fuente a rojo.

- **errorElevacion**: función que escribe "Introduce el factor de elevacion" en el campo correspondiente y modifica el color de la fuente a rojo.
- **errorVisualizaciones**: función que escribe "Introduce alguna visualización" en el campo correspondiente y modifica el color de la fuente a rojo.
- **errorRutaOrigen**: función que escribe "Introduce ruta de origen" en el campo correspondiente y modifica el color de la fuente a rojo.
- **hayDiccionario**: función que modifica la etiqueta que indica que sí existe un diccionario creado.
- **deshabilitaCruce**: función que deshabilita el campo para introducir la segunda variable de cruce en algunos tipos de visualización.
- **habilitaCruce**: función que habilita el campo para introducir la segunda variable de cruce en algunos tipos de visualización.
- **componentesFichero**: función que recibe como parámetros los formatos, separadores y tipos de visualización escritos en el archivo de texto y actualiza la vista con ellos.

Además de estas funciones, existen otras catorce funciones encargadas de capturar determinados eventos y realizar las acciones pertinentes. La única de ellas que es más relevante es la función de eliminar visualizaciones, la cual debe capturar la posición relativa del ratón cuando el usuario hace click sobre la ventana de texto y, en el caso de que se produzcan dos clicks seguidos, avisar al controlador de que realice las acciones oportunas.

7.4.7.2.0 VistaWarning

Se trata del JFrame Form encargado de crear la ventana de alerta de la aplicación, no posee ningún atributo.

En cuanto a las funciones, solo existen tres, el constructor y las encargadas de capturar los eventos correspondientes a los dos botones por los que está formada y pedirle al controlador que realice las acciones pertinentes.

7.4.7.3.0 VistaGenerando

Se trata del JFrame Form encargado de crear la ventana de espera de la aplicación, no posee ningún atributo ni ninguna función.

7.4.7.4.0 ControladorInterfaz

Controlador asociado a la vista principal de la aplicación, encargado de gestionar las operaciones realizadas por el usuario sobre la vista. Está compuesto por dos atributos.

- **mivista**: vista gestionada por el controlador.
- **rutaCompleta**: ruta que se utilizará para guardar el archivo resultante que obtenga la aplicación.

Las funciones incluidas en esta clase están descritas a continuación.

- **ControladorInterfaz**: constructor de la clase.
- **procesaSeparador**: función que obtiene el separador que ha introducido el usuario en la vista y se lo pasa al modelo.
- **procesaFormatoDestino**: función que obtiene el formato de destino que ha introducido el usuario en la vista y se lo pasa al modelo.
- **procesaCreaDiccionario**: función que pide a la máquina de estados cambiar a la ventana de creación del diccionario.
- **actualizar**: función que, en función de la visualización seleccionada, actualiza las medidas que pueden ser elegidas o las variables de cruce que pueden ser introducidas. También comprueba si existe diccionario creado para actualizar la etiqueta que lo indica.
- **procesaRutaOrigen**: función que abre el explorador de archivos y obtiene la ruta del fichero de datos seleccionado por el usuario, pasándosela a la vista para que actualice la etiqueta correspondiente, y al modelo para que la almacene.
- **procesaRutaDestino**: función que abre el explorador de archivos y obtiene la ruta seleccionada por el usuario, pasándosela a la vista para que actualice la etiqueta correspondiente, y al modelo para que la almacene.
- **procesaRutaR**: función que abre el explorador de archivos y obtiene la ruta del ejecutable de R seleccionada por el usuario, pasándosela a la vista para que actualice la etiqueta correspondiente, y al modelo para que la almacene.
- **procesaAñadir**: función que comprobará si existen errores en los campos relacionados con la visualización que el usuario desea añadir y, en caso de que no existan, crea una nueva visualización con los parámetros indicados y se la pasa al modelo para que la añada a su ArrayList y a la vista para que actualice el campo de texto.
- **limpiaCruce2**: función que comprueba si el campo de texto de la segunda variable de cruce no ha sido modificado, en cuyo caso le pide a la vista que lo limpie.

- **limpiaRespuesta**: función que comprueba si el campo de texto de la variable respuesta tiene su valor de error y, en ese caso, le pide a la vista que lo limpie.
- **limpiaCruce1**: función que comprueba si el campo de texto de la primera variable de cruce tiene su valor de error y, en ese caso, le pide a la vista que lo limpie.
- **limpiaElevacion**: función que comprueba si el campo de texto del factor de elevación tiene su valor de error y, en ese caso, le pide a la vista que lo limpie.
- **eliminaPorPos**: función que se ejecuta cuando el usuario hace doble click en una visualización existente para eliminarla y que se encarga de, en función del índice que calcula la función auxiliar "obtenerIndice", se elimine la visualización deseada y se actualice la vista.
- **obtenerIndice**: función auxiliar que obtiene el índice de la visualización que desea eliminar el usuario en base a la posición relativa del ratón. Se ha tomado como base que cada visualización mostrada en la ventana tiene una altura de 126 píxeles, a excepción del primer elemento que tendrá una altura de 134 píxeles debido a la estructura del panel de texto.
- **leefichero**: función encargada de leer el fichero de parámetros y pasárselo a la vista para que actualice las opciones seleccionables.
- **actualizate**: función que obtiene todos los elementos de la ventana principal ya existentes en el modelo, pasándoselos a la vista para que se actualice con toda la información actual.
- **cancelarOperacion**: una de las dos funciones referentes a la ventana de alerta, la cual le pide a la máquina de estados que destruya dicha ventana.
- **reemplazarArchivo**: segunda de las funciones referentes a la ventana de alerta, la cual se ejecuta si el usuario desea reemplazar el archivo que exista en la ubicación de destino, simplemente elimina dicho archivo para que el nuevo pueda ser creado, y cierra la aplicación.
- **procesaAceptar**: se trata de la función más compleja de la aplicación, realiza las siguientes tareas.

    1. Comprueba si existen errores en todos los campos obligatorios, en cuyo caso, le pide a la vista que lo muestre. Si no existe error, la función continúa.
    2. Comprueba si existe el primer fichero que va a crear, denominado "principal.txt" en la ruta por defecto. En caso de que no exista, lo crea, escribiendo en él toda la información obtenida de la ventana principal. Si el fichero existe, simplemente lo reemplaza.
    3. Comprueba si existe el segundo fichero que va a crear, denominado "diccionario.txt" en la ruta por defecto, en caso de que no exista, lo crea, escribiendo

en él toda la información obtenida de la ventana del diccionario. Si el fichero existe, simplemente lo reemplaza.

4. Comprueba si ya existe un fichero, en la ruta indicada, con el mismo nombre que el que el usuario desea crear. En el caso de que sí que exista, aparece la ventana de alerta. En caso contrario, se realiza la llamada al programa de R con normalidad, mostrándose la ventana de espera durante 25 segundos y volviendo una vez más a la ventana principal.

- **generando**: función que le solicita a la máquina de estados que muestre la ventana de espera.

Capítulo 8

Implementación y pruebas

8.1. Implementación

8.1.1. Paquetes utilizados

Para poder implementar las clases detalladas en la sección anterior, ha sido utilizado el entorno de desarrollo **Netbeans** y el lenguaje de programación **Java**. Junto con ellos, la librería gráfica **Java Swing**, la cual viene integrada con Netbeans, ha sido empleada para poder crear la interfaz.

Estos tres elementos han sido básicos para la implementación de toda la interfaz, no obstante, se han necesitado otra serie de librerías de java que serán detalladas a continuación.

- **Paquete java.util[17]:** Se trata de una librería que contiene una gran cantidad de clases e interfaces. Dentro de ella, solo han sido necesarias dos APIs.
  - *ArrayList:* Esta API es la más usada en toda la interfaz, y es necesaria para crear objetos de ese tipo.
  - *Logging:* Ha sido usada para capturar las excepciones que se puedan producir.
- **Paquete java.awt[18]:** Esta librería contiene algunas clases para crear GUIs. Esta vez, han sido necesarias cinco clases.
  - *Point:* Utilizada para almacenar la posición del ratón cuando el usuario quiera eliminar una visualización o una variable.
  - *Color:* Se ha empleado para cambiar el color de algunos componentes de la interfaz cuando se producen errores.
  - *Component y FileDialog:* Usadas en las clases para abrir el explorador de archivos.

  - *event:* Utilizada para el temporizador que cerrará la ventana de espera.

- **Paquete java.io[19]:** Una librería muy útil para recibir datos o exportarlos. En este caso, se ha usado para leer el fichero de texto con los parámetros y para crear los dos ficheros resultado.

- **Paquete java.swing[20]:** Como se ha dicho al principio, se ha usado para la creación de los elementos gráficos de la interfaz, sin embargo, han sido necesarias algunas clases a mayores.

  - *JFrame:* Se ha usado para poder almacenar el estado actual en la máquina de estados.
  - *JEditorPane:* Utilizado para poder acceder al explorador de archivos.
  - *Timer:* Utilizado para crear el temporizador.

8.1.2. Dificultades en la implementación

Durante toda la creación de la interfaz, han existido funcionalidades más complicadas de implementar que el resto. A continuación se van a exponer todas ellas y la forma en la que se han abordado.

- La eliminación de visualizaciones y variables a petición del usuario es una de las cosas que no se habían realizado anteriormente de la forma planteada y, por este motivo, se ha tenido que realizar un esfuerzo a mayores investigando la forma de realizarlo. La forma ideal planteada de eliminar una visualización o una variable es haciendo click sobre ella en el panel de texto en el que se muestra, el problema de esto saber cómo capturar la posición relativa del ratón en el momento de hacer click.

  Tras investigar un poco, se encontró que Java ya tenía un método implementado para ello y, haciendo uso de él, se conseguían las coordenadas X e Y del ratón cuando el usuario hacía click sobre una visualización o variable, teniendo eso, y dado que lo que ocupa cada visualización o variable dentro de su panel de texto es fijo, no fue demasiado complicado obtener el índice al que correspondía dicha coordenada Y. Pudiendo eliminar, de esta forma, el elemento que el usuario deseaba. Una vez finalizado esto, me di cuenta de algo obvio, el usuario podía pulsar sin querer en algún lugar del panel de texto. La solución adoptada fue que, para eliminar un elemento, se debía hacer **doble** click sobre él.

- Otra de las cosas que no se habían realizado nunca era la conexión entre Java y R. Como no se conocía una forma sencilla de comunicar la interfaz con la aplicación de R, y dado que R puede ser ejecutado desde línea de comandos, ese fue el camino adoptado. Desde Java no es complicado lanzar un comando desde cmd, no obstante, R necesitaba recibir todos los parámetros que usuario había introducido en la interfaz, para esto, se decidió que la interfaz creara dos documentos de texto con los

parámetros que la aplicación de R leería y, de esta forma, que todo funcionara sin problemas. En la figura 8.1 se muestra cómo se ha realizado la llamada a R. Como puede verse, el comando está formado por dos partes. En la primera se introduce la ruta de R y, en la segunda, la ruta en la que se encontraría el programa en R. Hay que tener en cuenta que lo que va dentro del "source", está escrito como un comando de R, por lo que las barras está al revés de lo normal.

```
String cmd = "\""+FuenteDeDatos.getRutaR()+"\" -e \"source('./tfg.r')\"";
Runtime.getRuntime().exec(cmd);
generando();
```

Figura 8.1: Conexión entre Java y R

8.2. Pruebas

A lo largo de esta sección se van a describir las pruebas realizadas sobre la interfaz de usuario creada para este proyecto. Como el proyecto está dividido en dos partes, la interfaz y la aplicación, pero en esta memoria solo se trata la primera, todas las pruebas serán unitarias, centradas en probar dicha interfaz. No obstante, serán necesarias algunas pruebas de integración con el objetivo de comprobar si la interfaz realiza bien la llamada al programa en R, y se crea, de forma satisfactoria, el informe final. En cuanto a las pruebas de la aplicación de R, estarán especificadas en la memoria adjunta[3].

Cabe destacar, antes de detallar las pruebas, que todas ellas han sido realizadas sobre la última versión de la aplicación.

Todas las pruebas realizadas seguirán la plantilla mostrada en la figura 8.2.

Identificador	El identificador de la prueba, de la forma P-00
Tipo	Si la prueba es unitaria o de integración
Descripción	Descripción de la prueba realizada
Salida esperada	Salida o resultado esperado de la prueba
Estado	Estado de la prueba, puede ser: - Superada - No superada

Figura 8.2: Plantilla para las pruebas

Identificador	P-01
Tipo	Prueba unitaria
Descripción	El explorador de archivos funciona correctamente en todos los apartados en los que se utiliza.
Salida esperada	Tras seleccionar las diferentes rutas, éstas se muestran en la interfaz.
Estado	Superada

Figura 8.3: Prueba unitaria 1

Identificador	P-02
Tipo	Prueba unitaria
Descripción	La aplicación permite crear un diccionario de datos.
Salida esperada	El cambio entre ventanas se produce de forma correcta y al añadir alguna variable al diccionario, la etiqueta de la ventana principal cambia.
Estado	Superada

Figura 8.4: Prueba unitaria 2

Identificador	P-03
Tipo	Prueba unitaria
Descripción	La opción de añadir variable en el diccionario funciona correctamente.
Salida esperada	La ventana de creación de diccionario se actualiza con la nueva variable creada tras pulsar el botón de añadir
Estado	Superada

Figura 8.5: Prueba unitaria 3

Identificador	P-04
Tipo	Prueba unitaria
Descripción	La opción de eliminar variable en el diccionario funciona correctamente.
Salida esperada	Al hacer doble click sobre una variable, dicha variable se elimina correctamente y se actualiza la vista reflejando los cambios.
Estado	Superada

Figura 8.6: Prueba unitaria 4

Identificador	P-05
Tipo	Prueba unitaria
Descripción	Tras añadir o eliminar variables en el diccionario, se pulsa cancelar para descartar los cambios.
Salida esperada	El diccionario almacenado no cambia.
Estado	Superada

Figura 8.7: Prueba unitaria 5

Identificador	P-06
Tipo	Prueba unitaria
Descripción	Tras introducir los datos y pulsar Añadir, se añaden las visualizaciones correctamente y se actualiza la ventana.
Salida esperada	La vista se actualiza reflejando la nueva visualización y además, también se actualiza el modelo.
Estado	Superada

Figura 8.8: Prueba unitaria 6

Identificador	P-07
Tipo	Prueba unitaria
Descripción	Tras hacer doble click sobre una visualización, se elimina correctamente
Salida esperada	La vista se actualiza correctamente y se elimina dicha visualización del modelo.
Estado	Superada

Figura 8.9: Prueba unitaria 7

Identificador	P-08
Tipo	Prueba unitaria
Descripción	Si se selecciona la visualización “gráfico de sectores”, se deshabilita la segunda variable de cruce.
Salida esperada	La segunda variable de cruce se deshabilita y no se permite modificarla.
Estado	Superada

Figura 8.10: Prueba unitaria 8

Identificador	P-09
Tipo	Prueba unitaria
Descripción	Los ficheros de texto se crean correctamente tras pulsar Aceptar.
Salida esperada	Ambos ficheros se crean en la ubicación correcta y con el contenido correcto.
Estado	Superada

Figura 8.11: Prueba unitaria 9

Identificador	P-11
Tipo	Prueba unitaria
Descripción	En el caso en el que el archivo a crear ya exista, aparece la ventana de aviso
Salida esperada	La ventana de aviso aparece correctamente al presionar Aceptar.
Estado	Superada

Figura 8.12: Prueba unitaria 10

Identificador	P-10
Tipo	Prueba integración
Descripción	La llamada a la aplicación de R se produce de forma correcta.
Salida esperada	Se crea el informe elegido, en la ubicación seleccionada y con el contenido correcto.
Estado	Superada

Figura 8.13: Prueba de integración 1

Identificador	P-12
Tipo	Prueba integración
Descripción	Si al aparecer la ventana de aviso, se selecciona reemplazar, la llamada a R ocurre correctamente.
Salida esperada	El programa en R se ejecuta correctamente y se sustituye el informe final por el nuevo.
Estado	Superada

Figura 8.14: Prueba de integración 2

Capítulo 9

Conclusiones y trabajo futuro

9.1. Conclusiones

Al finalizar este trabajo fin de grado, se puede concluir que se han alcanzado los objetivos planteados en su inicio. Se ha desarrollado una interfaz de usuario para una aplicación que genera informes partiendo de datos de encuestas. En ella el usuario puede introducir todos los parámetros necesarios así como visualizar los que ya haya introducido de una manera simple y concisa. Puede crear visualizaciones o eliminarlas si lo desea. También tiene a su disposición una opción para crear un diccionario, añadiendo o eliminando variables del mismo. Todo esto está complementado con el hecho de que la interfaz sea fácil de usar, debido a la encuesta realizada a diferentes perfiles de usuarios con el objetivo de refinar y simplificar su diseño.

Además de la parte de la programación, a lo largo de este proyecto se han abordado las cinco fases del desarrollo software. Se ha analizado el problema planteado inicialmente, obteniendo de esta forma los requisitos que debía cumplir la aplicación, así como otros elementos importantes. Una vez finalizada la fase de análisis, había que enfrentarse a la de planificación, la cual ha resultado más complicada, pues a pesar de la cantidad de trabajos que se han ido realizando a lo largo de la carrera, nunca había hecho falta planificar, de forma tan exhaustiva, cómo se iba a desarrollar algo. Sin embargo, al ser algo, relativamente nuevo, ha servido para perfeccionar esas habilidades, muy necesarias de cara al mundo laboral. También se ha entrado en la fase de diseño que, por la experiencia vivida a lo largo de las prácticas en empresa, es fundamental a la hora de desarrollar un proyecto.

Por todo esto, no es solo el hecho de haber programado una interfaz y haberla documentado. Con este proyecto fin de grado se ha podido comprender mejor cómo va a funcionar, aunque evidentemente no en su totalidad, el mundo laboral que nos espera a los ingenieros informáticos, y todas las tareas que deberemos realizar en el futuro.

9.2. Líneas de trabajo futuro

Se ha desarrollado toda la interfaz de manera que fuera lo más modular posible, ya que de esta manera, es muy sencillo implementar nueva funcionalidad sin modificar la ya existente. A continuación se enumeran algunas ideas que podrían llevarse a cabo en el futuro.

- Mejoras visuales en la interfaz, haciendo más vistosa la misma e incluso más usable.
- Contemplar en la interfaz las posibles ampliaciones realizadas en la aplicación de R, de las cuales se hablará en la memoria adjunta[3].
- Modificar la aplicación para que no sea necesario crear ficheros de texto para comunicarse con R, una forma de realizar esto, habría sido implementar la interfaz con Shiny[21], la cual es una librería de R para crear aplicaciones web. De esta forma, no solo habría resultado más fácil la comunicación de la interfaz con la aplicación, sino que habrían podido realizarse gráficos más interactivos, además de que no sería necesario depender de un archivo jar para ejecutar dicha interfaz. Los motivos de no haber utilizado esta librería era el desconocimiento total de la misma y el tiempo limitado del que se disponía para la realización del trabajo fin de grado.

Capítulo 10

Bibliografía

[1] Microsoft Office, diversas herramientas de procesado de textos, etc. Fecha de último acceso, 8 de Julio de 2019. Página web https://products.office.com/es-es/home

[2] The R Project for Statistical Computing, entorno de programación, fecha de último acceso, 9 de Julio de 2019. Página web https://www.r-project.org/

[3] Raúl Hernansanz Quevedo, 06 de junio de 2019, "Tratamiento automático de encuestas con R", memoria del Trabajo de Fin de Grado de Estadística.

[4] Pablo de la Fuente López, Yania Crespo Gonzalez-Carvajal, Curso 2018/2019, asignatura Planificación y Diseño de Sistemas Computacionales, Universidad de Valladolid.

[5] Project Management Institute, 2008, "A guide to the project management body of knowledge (PMBOK Guide)", Ed. Newton Sqare. Enlace web: https://www.pmi.org/pmbok-guide-standards/foundational/pmbok

[6] Apache Netbeans, entorno de desarrollo compatible con diferentes lenguajes, Fecha de último acceso, 8 de Julio de 2019. Página web https://netbeans.org/

[7] Lenguaje de programación, último acceso, 8 de Julio de 2019, información recuperada de https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci %C3 %B3n)

[8] Editor de Latex online, último acceso, 8 de Julio de 2019. Página web https://es.overleaf.com

[9] Sistema de composición de textos, último acceso, 8 de Julio de 2019. Información al respecto en https://es.wikibooks.org/wiki/Manual_de_LaTeX

[10] UML and Data Modeling and diagraming tool, último acceso, 8 de Julio de 2019. Página web http://astah.net/

[11] Herramienta para realizar capturas de pantalla, último acceso, 8 de Julio de 2019. Página web https://lightscreen.com.ar/

[12] Escuela de Ingeniería Informática de Valladolid, último acceso, 8 de Julio de 2019. Enlace web https://www.inf.uva.es/

[13] Departamento de Estadística e Investigación Operativa de Valladolid, último acceso, 8 de Julio de 2019. Enlace web http://www.eio.uva.es/

[14] Digital Guide, 26 de octubre de 2018, recuperado de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/uml-lenguaje-unificado-de-modelado-orientado-a-objetos/

[15] Miriam García, 5 de octubre de 2018, recuperado de https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve

[16] Antonio Leiva, último acceso el 14 de Marzo de 2019, recuperado de https://devexperto.com/principio-responsabilidad-unica/

[17] Paquete java.util. Información al respecto en la documentación de Oracle. Fecha último acceso, 8 de Julio de 2019. Página web https://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html

[18] Paquete java.awt. Información al respecto en la documentación de Oracle. Fecha último acceso, 8 de Julio de 2019. Página web https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html

[19] Paquete java.io. Información al respecto en la documentación de Oracle. Fecha último acceso, 8 de Julio de 2019. Página web https://docs.oracle.com/javase/7/docs/api/java/io/package-summary.html

[20] Paquete java.swing. Información al respecto en la documentación de Oracle. Fecha último acceso, 8 de Julio de 2019. Página web https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

[21] Paquete de R para construir aplicaciones web, último acceso el 23 de Junio de 2019. Página web https://shiny.rstudio.com/

Anexos

Apéndice A

Manual de Instalación de R

En esta sección se van a explicar, de forma detallada, todos los pasos necesarios para realizar la instalación de la herramienta R, necesaria para poder utilizar la aplicación desarrollada en este proyecto.

Antes de comenzar con las instrucciones para instalar R, cabe mencionar que, aunque dicha herramienta funciona en diversos sistemas operativos, como **Windows**, **Linux** o **Mac**, la aplicación desarrollada está pensada para Windows, por lo que este manual será para instalar R en este sistema operativo.

El primer paso será hacer doble click sobre el acceso directo llamado R que está incluido en la carpeta de la aplicación. Esto nos llevará a la página principal de R, donde deberemos presionar en el enlace marcado en rojo en la figura A.1, una vez hecho esto, tendremos que entrar en el siguiente enlace, también marcado en rojo en la figura A.2 y, finalmente, en el enlace de descarga de la figura A.3.

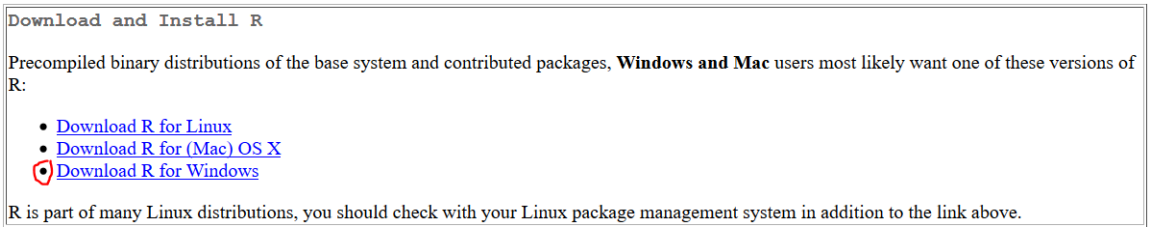

Figura A.1: Página web de R (parte 1)

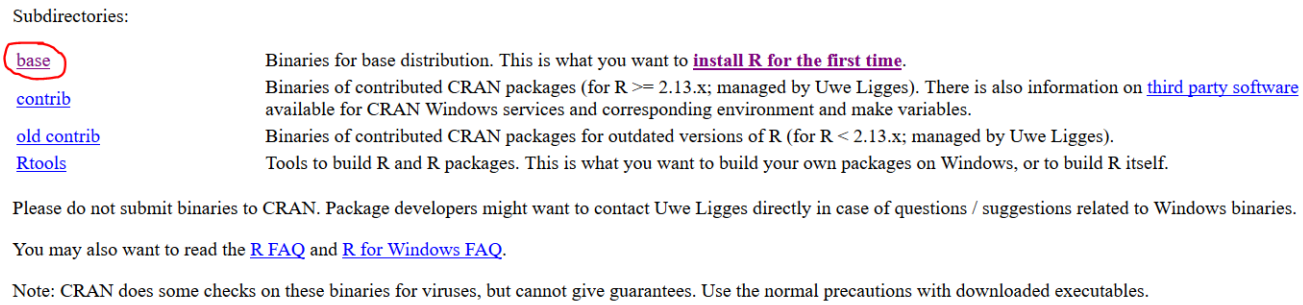

Figura A.2: Página web de R (parte 2)

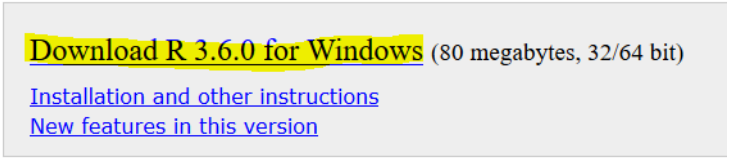

Figura A.3: Página web de R (parte 3)

Seguidamente habrá que ejecutar el archivo que hemos descargado y seguir las instrucciones que se describirán a continuación.

Para comenzar seleccionaremos el idioma en el que deseamos que esté la instalación, en mi caso será Español, una vez elegido, pulsaremos en **Aceptar**. Aparecerá entonces el acuerdo de licencia de la herramienta y tendremos que pulsar en **Siguiente**.

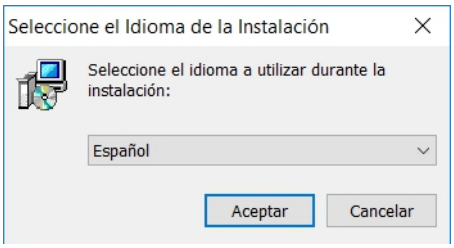

Figura A.4: Instalación de R (parte 1)

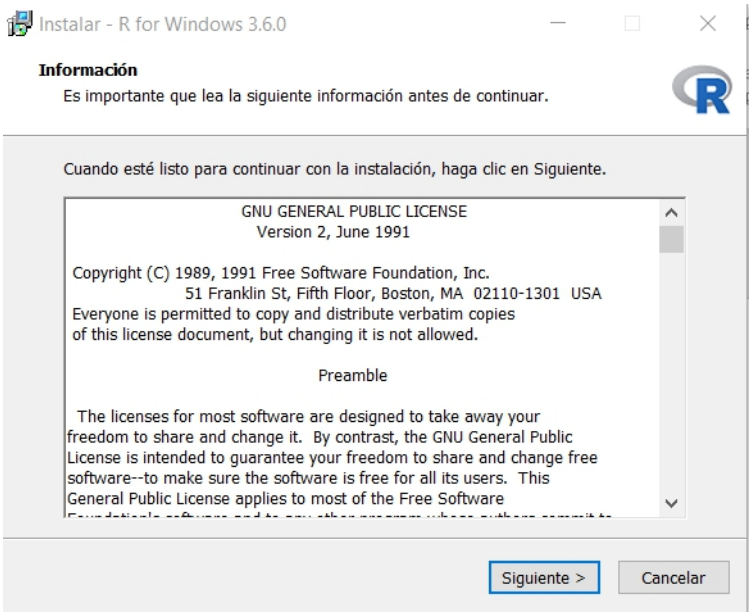

Figura A.5: Instalación de R (parte 2)

A continuación deberemos indicar la ruta en la que deseamos instalar R, esta ruta habrá que tenerla presente puesto que la aplicación desarrollada hará uso de ella, pero eso ya se explicará más adelante en esta misma sección.

Tras introducir la ruta que más nos interese, presionaremos en **Siguiente**. En la siguiente ventana, habrá que indicar los componentes que deseamos instalar. En este caso recomiendo instalar todos, para evitar posibles problemas. Con esto presente, pulsaremos en **Siguiente** tal y como aparece en la figura A.7.

Ahora pincharemos en **Siguiente** en las dos ventanas que siguen a la de los componentes, es decir, figuras A.8 y A.9 hasta que el instalador pida seleccionar las tareas

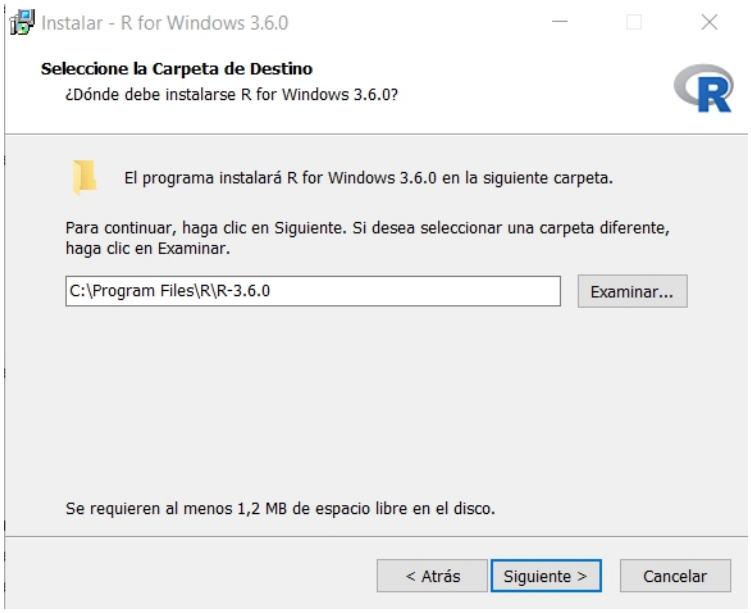

Figura A.6: Instalación de R (parte 3)

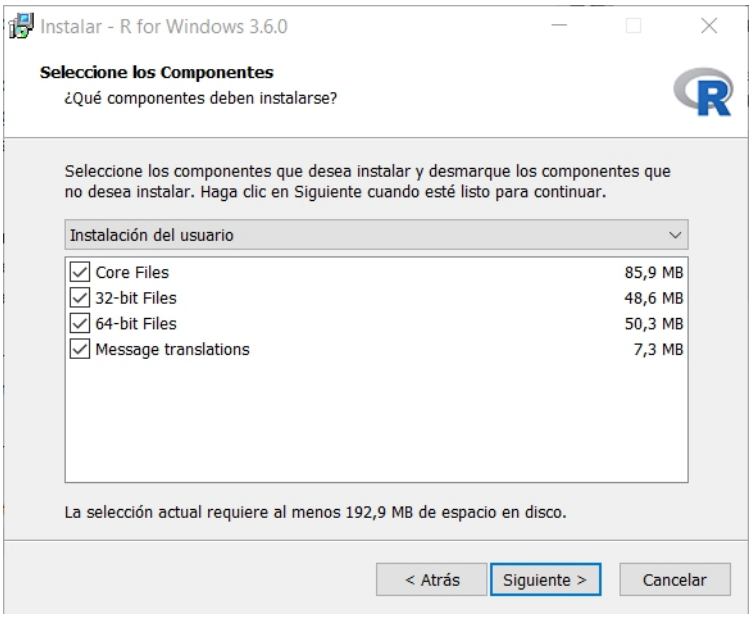

Figura A.7: Instalación de R (parte 4)

adicionales, como en la figura A.10. En este caso, además de las que aparecen marcadas por defecto, también seleccionaré la de crear un acceso directo en el escritorio. Tras esto, pulsaremos **Siguiente** y comenzará el proceso de instalación. Cuando termine, pulsaremos **Finalizar** y habrá terminado la instalación.

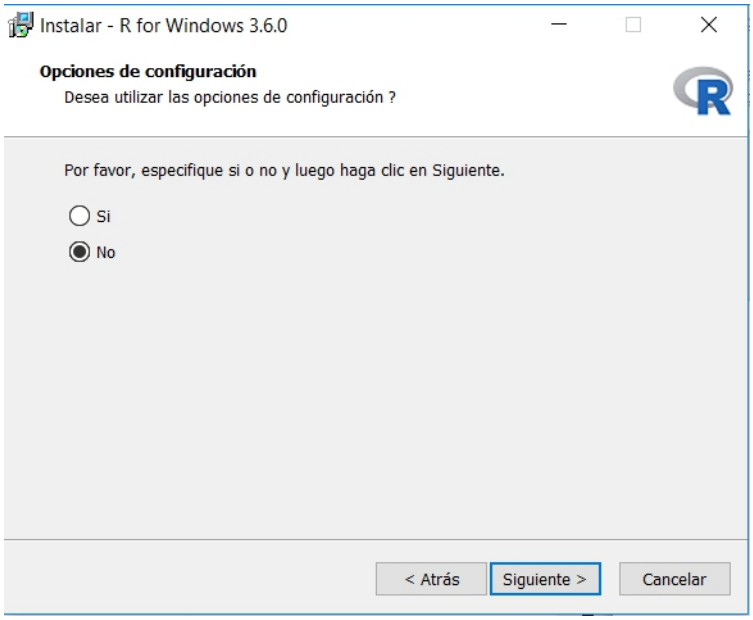

Figura A.8: Instalación de R (parte 5)

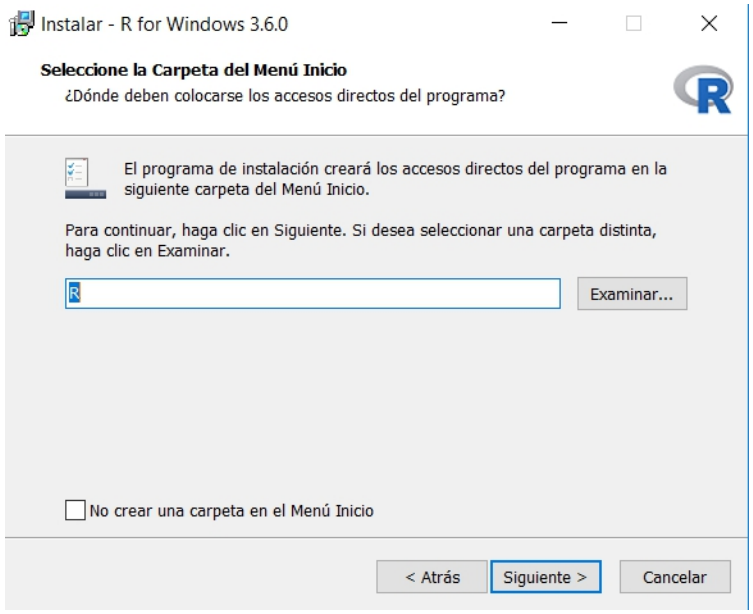

Figura A.9: Instalación de R (parte 6)

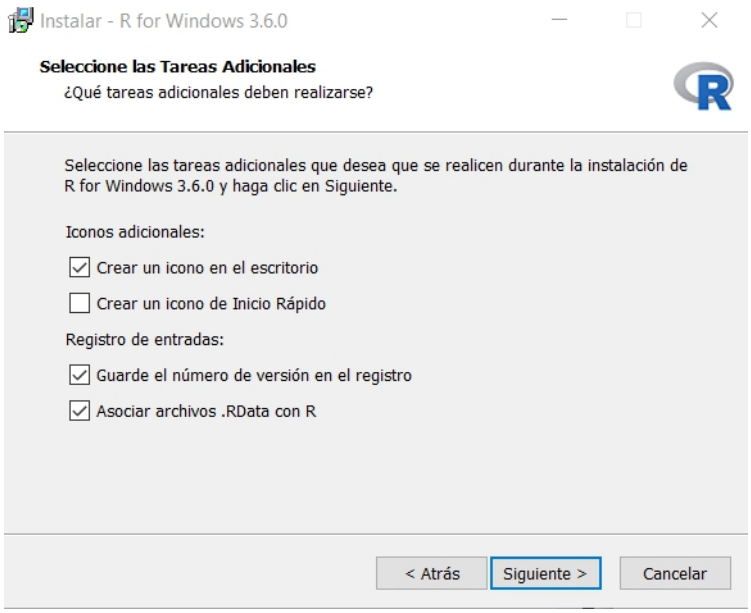

Figura A.10: Instalación de R (parte 7)

Apéndice B

Manual de uso de la Interfaz

Esta sección estará dedicada a explicar cómo debe ser usada la aplicación desarrollada para este proyecto. Como es una aplicación hecha con java, dicho programa deberá estar instalado para que pueda funcionar.

Teniendo en cuenta que se debe haber instalado R primeramente, se supondrá que ya se ha extraído el archivo comprimido en el que se encuentra la aplicación. En la carpeta principal, mostrada en la figura B.1 se pueden encontrar los siguientes archivos y directorios.

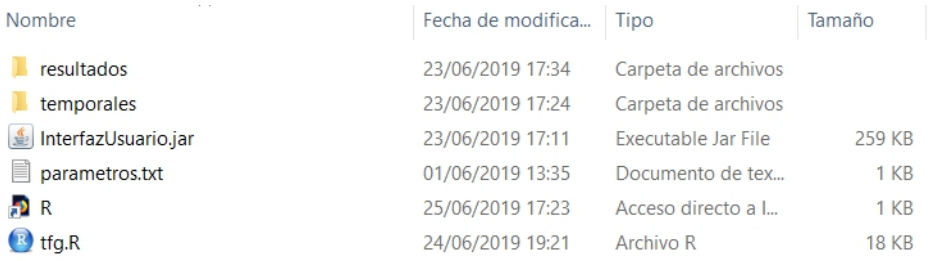

Figura B.1: Contenidos de la carpeta de la aplicación

- **tfg.R:** Archivo R que contiene el código fuente de la aplicación. Este archivo solo deberá ser abierto en el caso en el que se desee añadir o modificar algún módulo de la misma.

- **R:** Acceso directo a la web de descarga de la herramienta R.

- **parametros.txt:** Archivo de texto en el que el usuario podrá indicar algunos parámetros que utilizará la interfaz. Se debe tener en cuenta que no solo bastará añadir parámetros a este archivo, sino que será necesario modificar el código fuente de la aplicación R para contemplar los nuevos parámetros.

- **InterfazUsuario.jar:** Archivo java ejecutable que lanzará la interfaz de la aplicación.

- **temporales:** Directorio en el que se guardarán los archivos de texto que creará la interfaz de usuario y que son necesarios para que la aplicación R funcione. No es necesario que el usuario acceda a este directorio.

- **resultados:** Directorio en el que se guardarán las imágenes de los gráficos que cree el usuario, estas imágenes se incluirán más tarde en los informes indicados.

Para iniciar la aplicación se deberá hacer doble click sobre el archivo jar llamado InterfazUsuario, apareciendo de esta forma la ventana inicial de la aplicación, la cual se muestra en la figura B.2.

Figura B.2: Interfaz de usuario (parte 1)

Lo primero será hacer click en cualquier parte de la imagen para comenzar a utilizar la aplicación. De esta forma se abrira la ventana principal de la aplicación, mostrada en la figura B.3, la forma de utilizar la interfaz es muy sencilla, el primer paso sera introducir la ruta donde se encuentra el archivo de RScript, para ello pulsaremos en el botón con los 3 puntos situado en la parte derecha de ese mismo campo y buscaremos la ruta en la que tenemos instalado R; una vez en ella, accederemos a la carpeta **bin** marcada en la figura B.4.

Tras esto, habrá que entrar a la carpeta **x64**, también marcada en la figura B.5. Por último seleccionaremos el archivo **Rscript.exe** indicado en la figura B.6.

Una vez seleccionado el ejecutable de R, introduciremos el resto de parámetros comenzando por la ubicación del archivo de datos, el cual lo podremos introducir de la misma forma que la ruta de R. Después habrá que indicar el separador que emplea dicho archivo y el formato en el que deseamos el informe final.

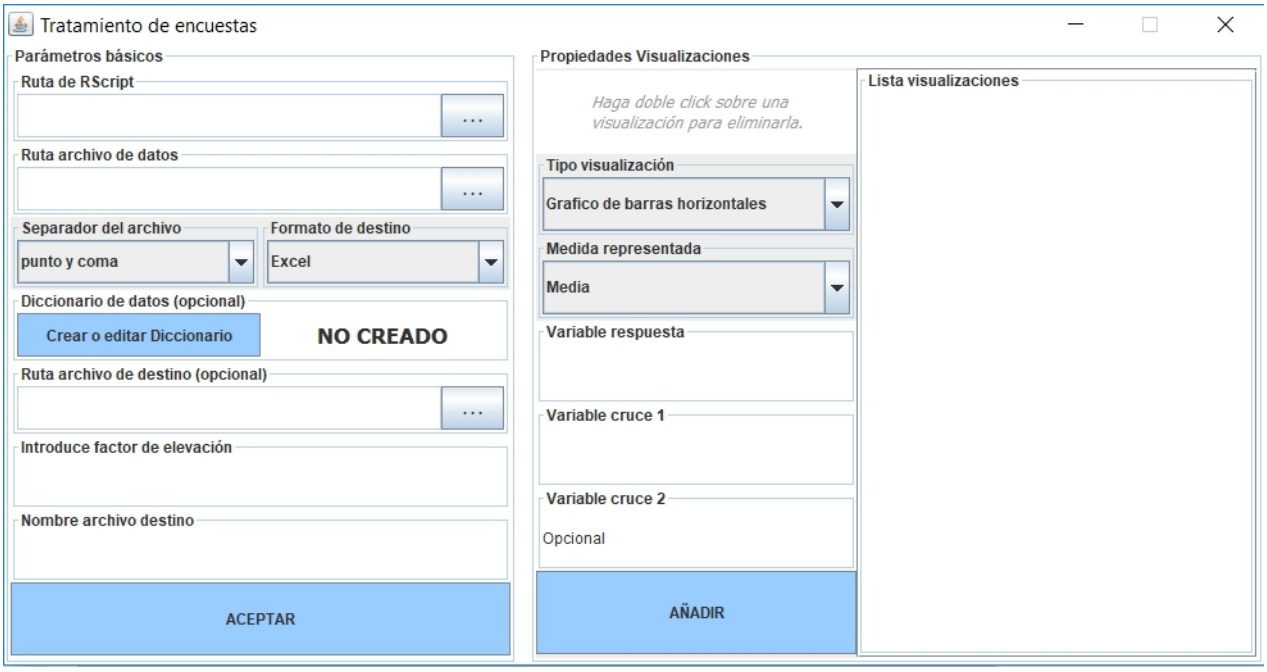

Figura B.3: Interfaz de usuario (parte 2)

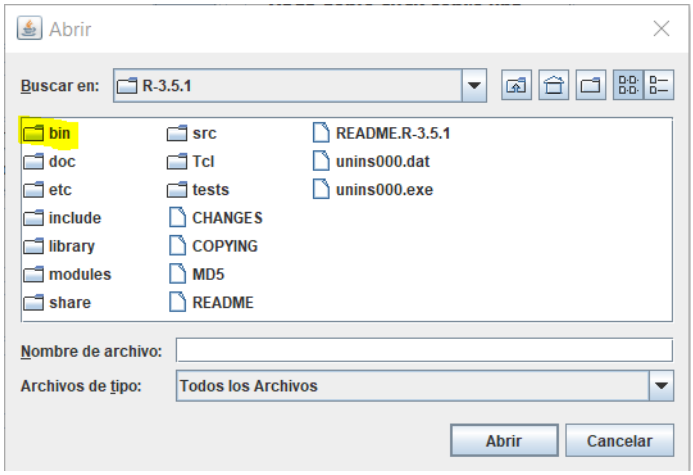

Figura B.4: Ruta de R (parte 1)

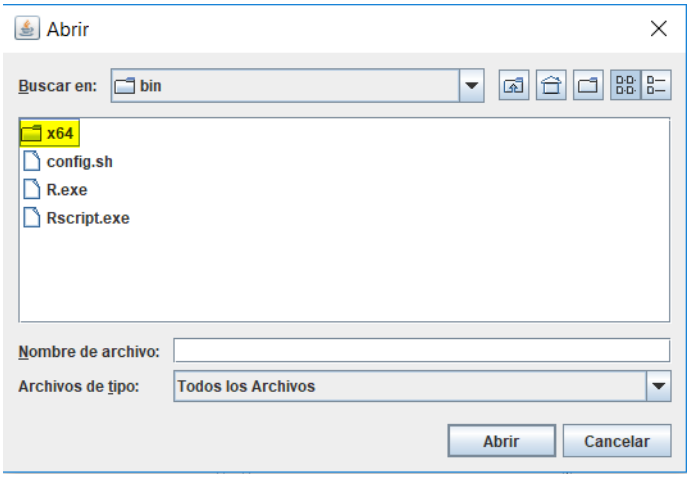

Figura B.5: Ruta de R (parte 2)

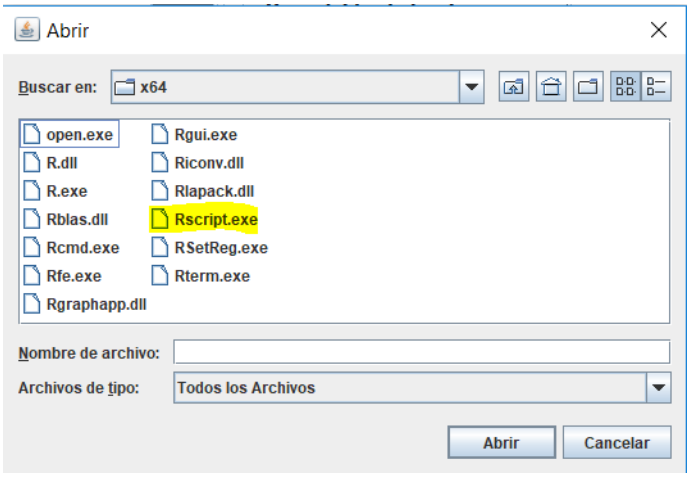

Figura B.6: Ruta de R (parte 3)

Tras esto, podremos crear un diccionario, cuya utilidad ya ha sido explicada en diferentes puntos de este mismo documento. Para hacerlo, lo primero será pulsar en el botón "Crear o editar Diccionario", mostrándose así la ventana de la figura B.7, una vez ahí, podremos crear o eliminar variables a nuestro gusto. Para crearlas, solo debemos introducir su nombre, código y descripción y pulsar sobre "Añadir", de esta forma se mostrará la variable creada en el campo de texto inferior, para ilustrar esto, se ha creado la variable "ejemplo". En el caso de que se desee eliminar dicha variable, solo habrá que hacer doble click sobre cualquier parte de ella en el panel de texto.

Figura B.7: Creación del diccionario

Hay que tener en cuenta que cuando se esté creando el diccionario, los cambios no se guardarán si el usuario no presiona el botón **Aceptar**. Cuando el diccionario esté creado o en el caso en el que no se desee crear, habrá que introducir la ruta donde queremos los resultados, si esta ruta no se indica, por defecto estará en la misma ubicación que la carpeta de la aplicación.

Lo siguiente será escribir el nombre **exacto** del factor de elevación que se encuentra en la encuesta con la que vamos a trabajar. El último parámetro a indicar, será el nombre que el usuario desea dar al informe resultado.

Cuando todos los parámetros estén indicados, habrá que introducir las visualizaciones que se deseen. Esto se hará en la parte derecha de la ventana principal, mostrada en la figura B.8, donde habrá que introducir el tipo de visualización deseada, la medida que queremos representar, la variable respuesta y, al menos, la primera variable de cruce. **Todas** las variables introducidas deben estar escritas de manera **exacta** a como están escritas en la encuesta con la que estemos trabajando, incluyendo si están o no en mayúsculas.

Una vez se ha rellenado todo, pulsaremos "Añadir" y el resultado de la visualización aparecerá en el panel de texto derecho. Para este ejemplo, se ha solicitado un Gráfico de barras horizontales, que represente la media de la variable *REJEMPLO* frente a la variable *CEJEMPLO1*, como no se ha introducido una segunda variable de cruce, ese campo aparece con un guión. La forma de eliminar una visualización es la misma que la

utilizada con las variables del diccionario, bastará con hacer doble click en cualquier parte de dicha visualización en el panel de texto.

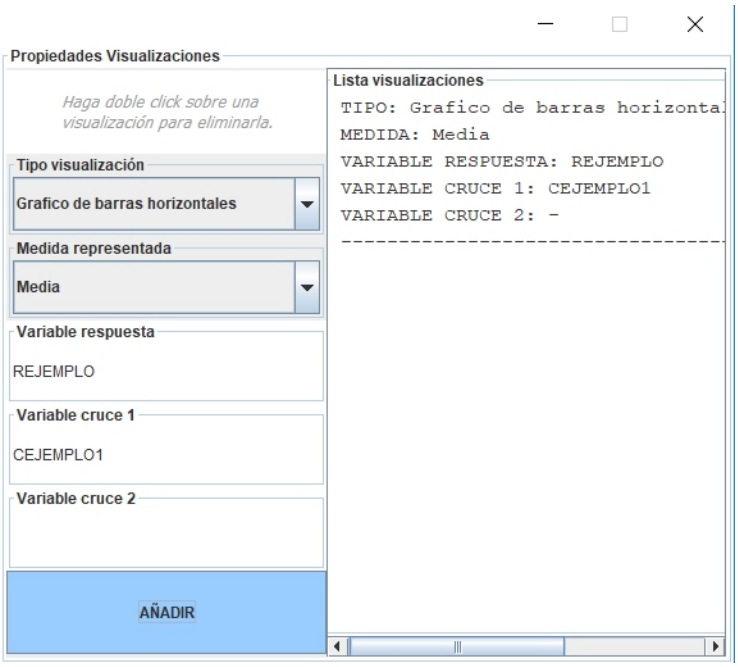

Figura B.8: Creación de visualizaciones

Finalmente, cuando todo esté introducido, pulsaremos el botón **Aceptar** y, tras un poco de tiempo, se creará el informe en la ubicación indicada. En el caso de que no se cree, se deberán comprobar las variables introducidas por si hubiera algún error.

Algo a tener en cuenta es que, si cuando presionamos **Aceptar**, el archivo que deseamos ya existe en la ubicación indicada, aparecerá una ventana de alerta, mostrada en la figura B.9, en este caso existirán dos opciones, la primera es reemplazar el archivo pulsando **Reemplazar** y la segunda es cancelar la operación pulsando **Cancelar**, si hacemos esto último, volveremos a la ventana de la interfaz.

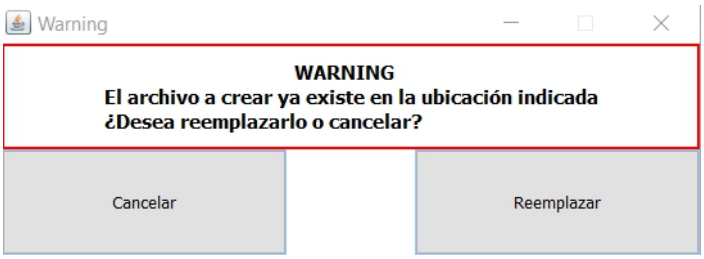

Figura B.9: Ruta de R (parte 4)

Una **observación importante** es que, a la hora de depositar la carpeta que contiene todos los elementos de la aplicación, se **debe** hacer en un lugar con una ruta absoluta no muy larga, y sin caracteres raros, debido a la codificación de los ficheros de texto generados y de limitaciones de lectura en rutas demasiado extensas.

Una vez explicado el funcionamiento general de la aplicación, se van indicar algunas cosas que se pueden realizar para probarla. Empleando el archivo de Excel incluido entre el contenido del CD, se tendría que introducir, como factor de elevación, la variable **"FACTOTAL"** y, como separador del archivo, el **tabulador**. En la figura B.10 se muestran algunas variables que se pueden introducir como variables de cruce para las visualizaciones. Como variable respuesta, se introducirá **"SALBRUTO"**. Para mostrar un ejemplo, en la imagen B.11 se han introducidos todos los parámetros necesarios para crear un gráfico de barras horizontales.

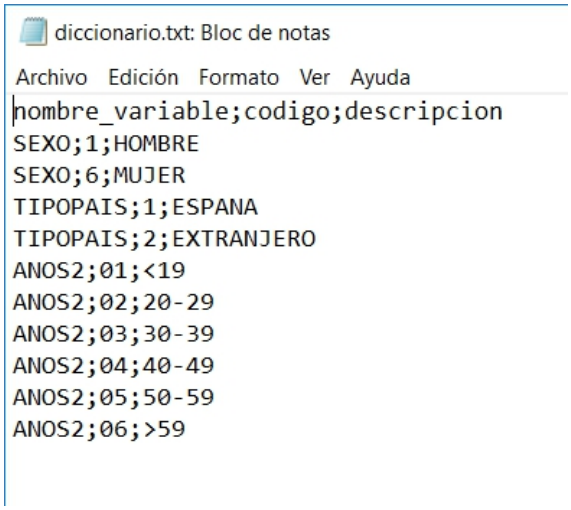

Figura B.10: Ejemplo de variables

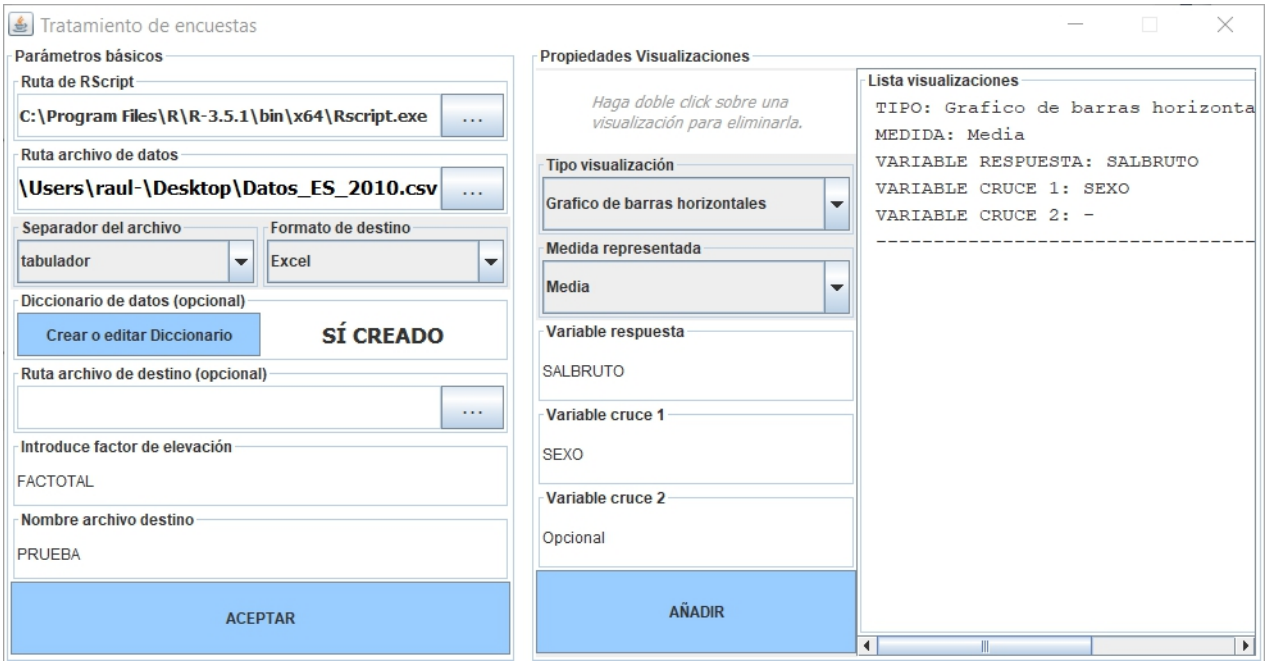

Figura B.11: Ejemplo de parámetros

Apéndice C

Contenido del CD

- /
  - memoria.pdf — Memoria del Trabajo de Fin de Grado de Ingeniería Informática.
  - InterfazUsuario.zip — Proyecto exportado mediante NetBeans, que contiene el código fuente de la interfaz, debido a las rutas, no se debe ejecutar la aplicación desde aquí.
  - Datos_ES_2010.csv — Fichero Excel con datos de ejemplo para probar la aplicación.
  - Programa
    - resultados — Carpeta donde se almacenarán las imágenes de las visualizaciones de forma temporal.
    - temporales — Carpeta donde se crearán los archivos de texto que leerá la aplicación de R.
    - InterfazUsuario.jar — Ejecutable que lanza la aplicación.
    - parametros.txt — Documento de texto en el que se pueden introducir diversos parámetros.
    - R — Acceso directo a la página de descarga de R.
    - tfg.R — Código fuente de la aplicación R.