Universidad de Valladolid

E. T. S. DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería Informática

Aplicación Web para la visualización y consulta del Padrón Municipal de Valladolid

---

Alumno: Santiago de la Parte Flórez

Tutor: Pablo de la Fuente Redondo
Cotutor: Javier David Fernández García

Quiero agradecer a mis tutores Pablo y Javier el trabajo realizado y su implicación durante el desarrollo de este Proyecto, así como a mi familia el apoyo y la paciencia prestados.

RESUMEN

Cada vez son más las empresas públicas y privadas que de un modo u otro liberan información y la ponen al alcance del público a través de Internet. El Padrón Municipal de Valladolid contiene información y datos estadísticos que pueden ser interesantes para el público, lo que lo hace susceptible de ser publicado para que los ciudadanos lo puedan consultar.

El Trabajo Fin de Grado desarrollado plantea una aplicación Web para la consulta y visualización del Padrón Municipal de Valladolid por parte de todo aquel que lo desee. La presente memoria cubre todos los aspectos del desarrollo del Proyecto, desde los objetivos y motivaciones iniciales a la planificación, desarrollo, pruebas y conclusiones.

ABSTRACT

More and more public and private companies are releasing information in some way, making it available to people through the Internet. The Municipal Register of Valladolid contains statistic data and information which can be interesting to people. That makes it susceptible to be released and accessed as a result by citizen.

The developed Final Degree Project raises a web application to access and display data from the Municipal Register of Valladolid by anyone who wishes it. This memory covers all Project development issues, from objectives and motivation to planning, development, testing and conclusions.

ÍNDICE GENERAL DE LA MEMORIA

ÍNDICE DE FIGURAS

I. INTRODUCCIÓN

1.1 MOTIVACIÓN

Cada vez más instituciones públicas deciden permitir un acceso total a determinados datos, de forma que estén disponibles de forma libre a todo el mundo, sin restricciones de copyright, patentes u otros mecanismos de control [36].

Una de estas instituciones es el Ayuntamiento de Valladolid [11]. En la actualidad se pueden consultar distintos datos estadísticos sobre su población, territorio, clima, economía, trabajo o vivienda entre otras. Sin embargo el formato en el que se encuentra esta información no resulta el más adecuado para su consulta y actualización, al tratarse de hojas de cálculo estáticas.

Resulta por lo tanto interesante buscar un medio que por una parte permita a los ciudadanos consultar esta información de la manera más sencilla y rápida posible, y por otra permita la actualización y mantenimiento de los datos de forma completamente independiente a la representación gráfica de los mismos.

1.2 OBJETIVOS

El principal objetivo del Trabajo Fin de Grado consiste en proporcionar a los ciudadanos el acceso de los datos abiertos relativos al Padrón Municipal de Valladolid a través de una aplicación web capaz de recibir consultas y mostrar gráficamente sus resultados.

Como se ha sugerido en la motivación, se plantea la creación de una aplicación que permita la consulta de los datos abiertos del Ayuntamiento de Valladolid de forma gráfica y accesible al ciudadano medio, que no tiene porqué poseer conocimientos avanzados de consulta de datos persistentes y/o representación gráfica de los mismos.

A la hora de trabajar con un gran volumen de datos como pueden ser los de un padrón, debe buscarse una forma de representación de los mismos lo más simple posible. Un usuario que busque información, por ejemplo, sobre la edad de las mujeres nacidas en Madrid que vivan en un determinado distrito de la ciudad no está interesado en conocer los detalles de cada una de las personas que cumplen esos parámetros, sino que estará más interesado en obtener una relación del número y porcentaje de la población seleccionada, y agrupada de forma lógica.

Por lo tanto, la aplicación deberá funcionar también como un filtro, ofreciendo al usuario los datos que le interesen de manera general, mostrándolos agrupados de forma que de un vistazo rápido pueda obtener la información que requiera.

Se plantea entonces un nuevo objetivo, consistente en el desarrollo de una aplicación capaz de mostrar la información solicitada de forma clara y simple, que no muestre datos demasiado puntuales pero que tampoco se quede corta a la hora de desplegar. Se debe buscar por lo tanto y para cada tipo de distribución unos rangos de agrupación correctos, así como una representación gráfica acorde a la información a desplegar.

El último objetivo es lograr desarrollar una aplicación fácil de usar por un usuario inexperto. El despliegue de la aplicación debe ser lo más sencillo y rápido posible. Además deberá buscarse la forma de minimizar el tiempo de aprendizaje que un usuario no experto necesite para manejar la aplicación correctamente

1.3 PUNTO DE PARTIDA

Para lograr llevar a buen término los objetivos planteados, se parte en primer lugar de los datos del Padrón de 2011. En un principio se esperaba obtener los datos en documentos RDF que pudieran ser consultados a través de una aplicación externa y de forma remota. La disponibilidad de estos datos estaba supeditada a otro Proyecto que se ha retrasado; por lo que finalmente se obtuvieron los datos del Padrón de una base de datos relacional, que inicialmente iba a emplearse de forma temporal para poder construir un prototipo de la aplicación hasta que el acceso al RDF estuviera disponible.

La base de datos tuvo que ser modificada y optimizada en algunos aspectos, si bien estas modificaciones no fueron profundas.

Para alcanzar los objetivos planteados en el punto anterior se empleó el lenguaje de programación Java, que dispone de APIs (bajo licencia LGPL [19]) para representar datos gráficamente, no siendo por lo tanto necesario su desarrollo específico dentro del Trabajo Fin de Grado.

Los motivos de la elección de la citada tecnología se pueden ver detallados en el apartado *Contexto tecnológico* de esta memoria.

1.4 ESTRUCTURA DE LA MEMORIA

La presente memoria está estructurada en capítulos que abordan distintos aspectos del Trabajo Fin de Grado.

El capítulo actual (Introducción) sirve de prefacio, describiendo la motivación y los objetivos del Proyecto, así como la estructura de la memoria.

El segundo capítulo describe la planificación del Proyecto, y los motivos que llevaron a realizar una replanificación del mismo.

En el tercer capítulo se detalla el contexto del Trabajo Fin de Grado, diferenciando entre el contexto de la aplicación (información sobre los datos abiertos y las características del padrón municipal) y el contexto tecnológico (aproximación a las tecnologías Java, JFreeChart, PHP y los motivos de su elección).

El cuarto capítulo está enfocado al análisis. En él se detallan los requisitos, casos de uso y diagrama de clases inicial de la aplicación.

El quinto capítulo describe el diseño de la aplicación, entrando en el detalle de la configuración y estructura de la base de datos proporcionada y las modificaciones realizadas sobre ella, el diseño de la interfaz gráfica y la definición de las clases desarrolladas en el software y sus relaciones.

El sexto capítulo corresponde a la implementación de la aplicación, centrándose principalmente en la comunicación entre los distintos elementos software (*applet*, script PHP y base de datos).

El séptimo capítulo se centra en las pruebas realizadas sobre la aplicación y los resultados de las mismas, incluyendo el diagnóstico y corrección de las pruebas fallidas. Las pruebas que se pueden encontrar en el capítulo corresponden con una selección del total, pudiéndose consultar la batería de pruebas completa en el CD anexo a esta memoria.

El octavo capítulo contiene las conclusiones obtenidas al término del Proyecto, así como las posibilidades de trabajo futuro que derivan del mismo.

El noveno capítulo contiene las referencias de la bibliografía consultada durante el desarrollo del Proyecto ordenada alfabéticamente y por categorías.

Finalmente se encuentran tres anexos, con las instrucciones de instalación de la aplicación, su manual de uso y el contenido del CD anexo a esta memoria.

II. PLANIFICACIÓN

2.1 INTRODUCCIÓN

La planificación del proyecto tiene dos funciones: clarificar el orden de las tareas y estimar el tiempo necesario para llevarlas a cabo. Para realizar una buena planificación, los objetivos han de estar correctamente descritos. Deben estimarse los tiempos, identificar los hitos, encadenar las actividades necesarias para alcanzar los hitos ya definidos y finalmente realizar una planificación temporal. La planificación no ha de ser definitiva. A lo largo del desarrollo del proyecto será necesario reajustar y modificar los planes iniciales en función de cómo se va desarrollando el trabajo [1].

Para representar la planificación se usarán *diagramas de Gantt*, que sirven para representar de una manera simple e intuitiva las distintas fases, tareas y actividades de que consta el proyecto en una línea temporal.

La planificación del Proyecto se llevó a cabo siguiendo la metodología del Proceso Unificado (RUP) [21].

2.2 PLANIFICACIÓN INICIAL

Para realizar la planificación inicial, se determinaron los objetivos principales del proyecto y sus hitos asociados, dividiéndolos en actividades y estimando los tiempos necesarios para su consecución.

La metodología del Proceso Unificado (RUP) sugiere una división inicial del trabajo en cuatro fases: Inicio, Elaboración, Construcción y Transición. Estas cuatro fases engloban todo el Proyecto, desde su planteamiento inicial hasta su lanzamiento y despliegue.

En la primera fase inicio, se establece el alcance del proyecto, estableciendo su ámbito y límites y definiendo los casos de uso y críticos y los escenarios principales. En esta primera fase se realiza también un estudio sobre la bibliografía y tecnologías a emplear, y se realiza la planificación del proyecto así como el estudio de los riesgos potenciales del mismo.

La segunda fase del RUP es la fase de elaboración. En ella se analiza el dominio del problema, se describe la Arquitectura del Software y se desarrolla un plan de proyecto para la fase de construcción.

En la fase de construcción se desarrolla el producto, completando los diagramas y casos de uso planteados en las fases anteriores, así como llevando a cabo la implementación. Esta fase será iterativa, definiendo reuniones periódicas en las que se presentará y probará un nuevo prototipo de la aplicación, sobre el que se podrán plantear modificaciones y se determinarán los añadidos que deberán estar presentes en el siguiente prototipo.

La última fase del RUP es la fase de transición. En ella se debe conseguir que el usuario acepte el producto como completo y consistente. Al final de esta fase se obtendrá la versión final de la aplicación, con una funcionalidad completa y lista para usar y entregar al cliente.

A continuación se alcanza la etapa de pruebas, muy relacionada con la de revisión. En estas dos etapas se prueba el sistema y en función de los resultados se revisa y modifica. Por último llega la etapa de completar la memoria, ordenándola y desarrollándola para describir íntegramente y de forma clara y concisa las distintas etapas del proyecto. Se estimó una duración total de siete meses para la consecución de todas las etapas.

Esta planificación puede verse a continuación detallada y representada en diagramas de Gantt[1]:

[1] Los diagramas de Gantt han sido diseñados con la aplicación *Microsoft Project 2007* [55].

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio del Proyecto	0 días	mié 16/05/12	mié 16/05/12	
2	**Inicio**	**36 días**	**mié 16/05/12**	**mié 04/07/12**	
3	Contexto Proyecto	4 días	mié 16/05/12	lun 21/05/12	
4	Estudio Tecnologías	7 días	lun 21/05/12	mar 29/05/12	
5	Planificación Inicial	26 días	mié 30/05/12	mié 04/07/12	3;4
6	Reunión de Control	0 días	vie 01/06/12	vie 01/06/12	
7	Reunión de Control	0 días	vie 15/06/12	vie 15/06/12	
8	Reunión de Control	0 días	lun 02/07/12	lun 02/07/12	
9	**Elaboración**	**20 días**	**jue 05/07/12**	**mié 01/08/12**	**5**
10	Análisis (inicial)	15 días	jue 05/07/12	mié 25/07/12	
11	Diseño (inicial)	20 días	jue 05/07/12	mié 01/08/12	
12	Reunión de Control	0 días	lun 16/07/12	lun 16/07/12	
13	Reunión de Control	0 días	mar 31/07/12	mar 31/07/12	
14	**Construcción**	**37 días**	**jue 02/08/12**	**vie 21/09/12**	**11**
15	Análisis	10 días	jue 02/08/12	mié 15/08/12	
16	Diseño	20 días	jue 02/08/12	mié 29/08/12	
17	Implementación	30 días	lun 13/08/12	vie 21/09/12	
18	Reunión de Control	0 días	lun 03/09/12	lun 03/09/12	
19	Reunión de Control	0 días	lun 17/09/12	lun 17/09/12	
20	**Transición**	**15 días**	**lun 24/09/12**	**vie 12/10/12**	**17**
21	Pruebas	15 días	lun 24/09/12	vie 12/10/12	
22	Modificaciones	13 días	mié 26/09/12	vie 12/10/12	
23	Reunión de Control	0 días	lun 01/10/12	lun 01/10/12	
24	Documentación (inicio)	53 días	mié 01/08/12	vie 12/10/12	
25	Reunión de Control	0 días	lun 15/10/12	lun 15/10/12	
26	**Final**	**20 días**	**lun 15/10/12**	**vie 09/11/12**	**22;21;24**
27	Completar documentación	10 días	lun 15/10/12	vie 26/10/12	
28	Preparar presentación	10 días	lun 29/10/12	vie 09/11/12	27
29	Reunión de Control	0 días	jue 01/11/12	jue 01/11/12	
30	Final del Proyecto	0 días	vie 09/11/12	vie 09/11/12	

Figura 2.1: División de actividades de la planificación inicial

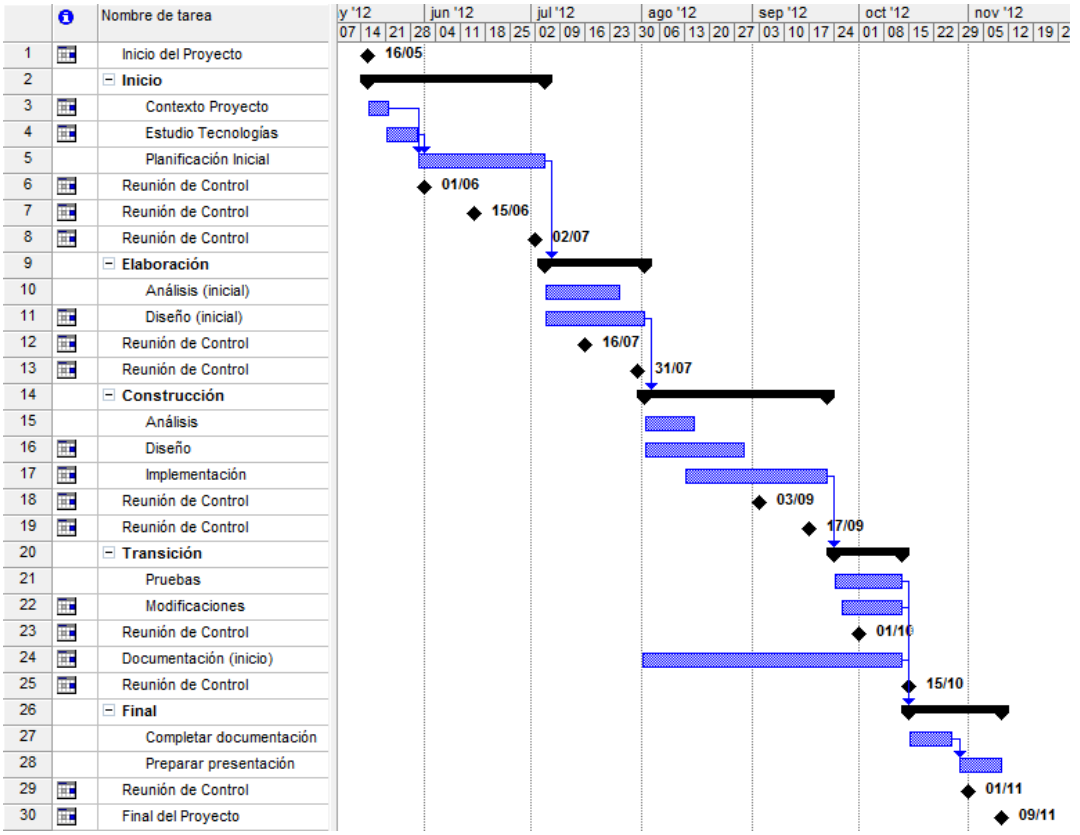

Figura 2.2: Diagrama de Gantt de la planificación inicial

2.3 RIESGOS

Dentro de la primera fase de la planificación se establecieron los riesgos que podrían penalizar el desarrollo del proyecto. Estos riesgos se detallan a continuación, y están incluidos en este apartado de la memoria porque resultaron en la necesidad de modificar la planificación original, como se puede ver en el siguiente apartado.

El principal riesgo fue la no disponibilidad de los datos del Padrón a través de una aplicación externa. La transformación de los datos a un formato adecuado y su carga en la aplicación dependían de otro proyecto paralelo al Trabajo Fin de Grado. Finalmente el riesgo potencial se hizo realidad, por lo que fue necesaria una modificación de la planificación inicial, que puede consultarse en el siguiente subapartado.

Otros riesgos a tener en cuenta fueron los posibles retrasos de reuniones debidos a causas de fuerza mayor. La forma de abordar esta posibilidad consistió en la planificación de cada reunión al término de la anterior, manteniendo en todo momento contacto por correo electrónico al fin de encontrar una solución en el menor tiempo posible en caso de un problema de esta índole. El contar

con dos tutores en lugar de uno solo facilitó también la situación, ya que las reuniones urgentes podrían llevarse a cabo incluso en el caso de no estar disponible alguno de ellos.

2.4 REPLANIFICACIÓN

La idea inicial del Trabajo Fin de Grado era contar con una aplicación externa que permitiese el acceso a los datos. Este planteamiento se tuvo en cuenta dentro de la planificación inicial, pero finalmente no pudo llevarse a cabo por estar supeditado al desarrollo de otro proyecto externo que no se completó a tiempo.

Debido a ello fue necesaria una modificación de la planificación original, incluyendo dentro de una de las iteraciones nuevas etapas de investigación bibliográfica y estudio de tecnologías, etapas que a priori debían haber estado terminadas durante las dos primeras fases.

Esta replanificación puede verse a continuación representada en diagramas de Gantt:

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Inicio del Proyecto	0 días	mié 16/05/12	mié 16/05/12	
2	**Inicio**	**36 días**	**mié 16/05/12**	**mié 04/07/12**	
3	Contexto Proyecto	4 días	mié 16/05/12	lun 21/05/12	
4	Estudio Tecnologías	7 días	lun 21/05/12	mar 29/05/12	
5	Planificación Inicial	26 días	mié 30/05/12	mié 04/07/12	3;4
6	Reunión de Control	0 días	vie 01/06/12	vie 01/06/12	
7	Reunión de Control	0 días	vie 15/06/12	vie 15/06/12	
8	Reunión de Control	0 días	lun 02/07/12	lun 02/07/12	
9	**Elaboración**	**20 días**	**jue 05/07/12**	**mié 01/08/12**	**5**
10	Análisis (inicial)	15 días	jue 05/07/12	mié 25/07/12	
11	Diseño (inicial)	20 días	jue 05/07/12	mié 01/08/12	
12	Reunión de Control	0 días	lun 16/07/12	lun 16/07/12	
13	Reunión de Control	0 días	mar 31/07/12	mar 31/07/12	
14	**Construcción**	**37 días**	**jue 02/08/12**	**vie 21/09/12**	**11**
15	Análisis	10 días	jue 02/08/12	mié 15/08/12	
16	Diseño	20 días	jue 02/08/12	mié 29/08/12	
17	Implementación	30 días	lun 13/08/12	vie 21/09/12	
18	Reunión de Control	0 días	lun 03/09/12	lun 03/09/12	
19	Reunión de Control	0 días	lun 17/09/12	lun 17/09/12	
20	**Tareas Modificación**	**20 días**	**vie 21/09/12**	**vie 19/10/12**	**17**
21	Investigación bibliográfica	2 días	lun 24/09/12	mar 25/09/12	
22	Estudio tecnologías	2 días	mar 25/09/12	mié 26/09/12	
23	Análisis	3 días	mié 26/09/12	vie 28/09/12	
24	Reunión de Control	0 días	vie 21/09/12	vie 21/09/12	
25	Diseño	5 días	lun 01/10/12	vie 05/10/12	23
26	Implementación	10 días	lun 08/10/12	vie 19/10/12	25
27	Reunión de Control	0 días	lun 15/10/12	lun 15/10/12	
28	**Transición**	**15 días**	**lun 22/10/12**	**vie 09/11/12**	**26**
29	Reunión de Control	0 días	jue 01/11/12	jue 01/11/12	
30	Pruebas	15 días	lun 22/10/12	vie 09/11/12	
31	Modificaciones	13 días	lun 22/10/12	mié 07/11/12	
32	Documentación (inicio)	73 días	mié 01/08/12	vie 09/11/12	
33	**Final**	**20 días**	**lun 12/11/12**	**vie 07/12/12**	**31;30;32**
34	Reunión de Control	0 días	jue 15/11/12	jue 15/11/12	
35	Completar documentación	10 días	lun 12/11/12	vie 23/11/12	
36	Preparar presentación	10 días	lun 26/11/12	vie 07/12/12	35
37	Reunión de Control	0 días	vie 30/11/12	vie 30/11/12	
38	Final del Proyecto	0 días	vie 07/12/12	vie 07/12/12	

Figura 2.3: División de actividades de la replanificación

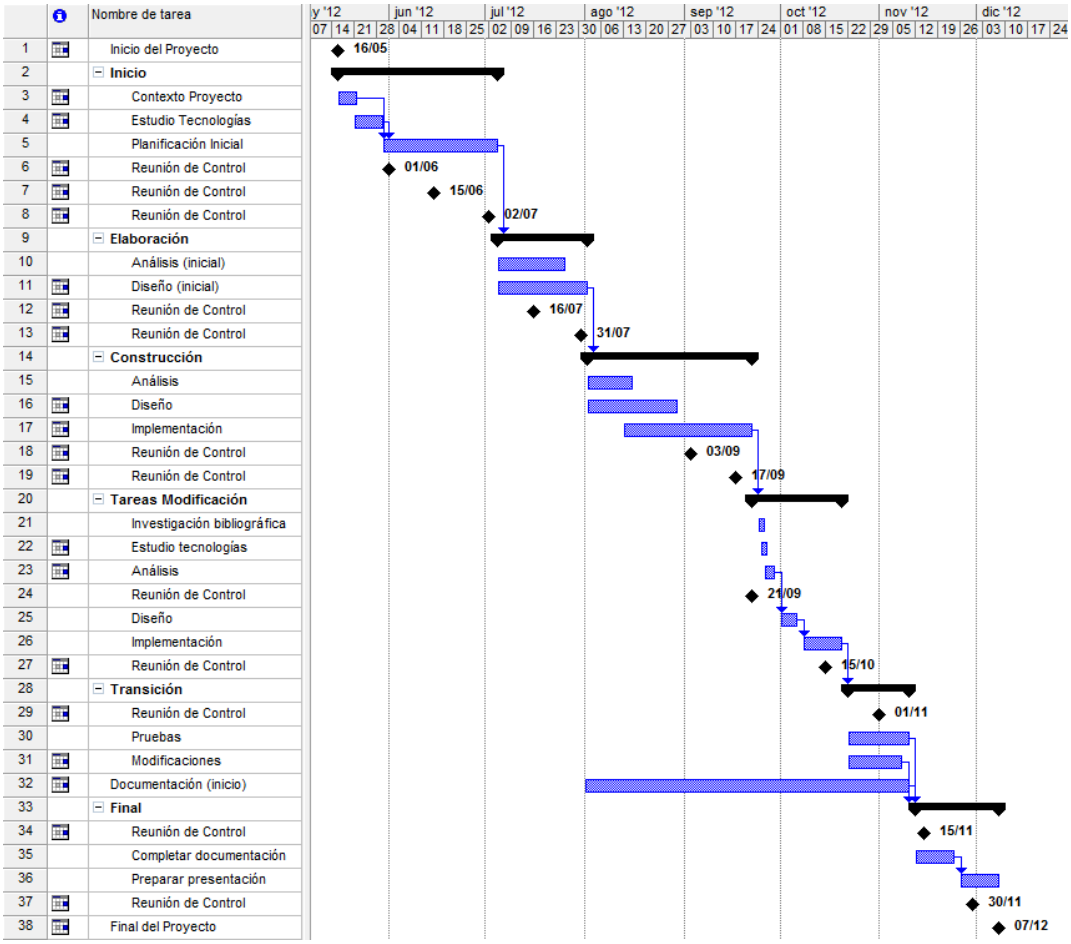

Figura 2.4: Diagrama de Gantt de la replanificación

2.5 CONCLUSIONES SOBRE LA PLANIFICACIÓN

Se puede determinar que la planificación planteada fue adecuada, ya que se logró la consecución de los objetivos en el plazo previsto. Los riesgos también fueron planteados correctamente, y fueron solventados sin problemas.

III. CONTEXTO

Este capítulo describe el contexto de la aplicación. Se diferencia el contexto temático del tecnológico. El apartado contexto temático examina el entorno en que se ubica la aplicación (datos abiertos gubernamentales, información sobre los datos incluidos en el padrón municipal…). El contexto tecnológico introduce las tecnologías empleadas en el desarrollo del Trabajo Fin de Grado, explicando los motivos de su elección.

3.1 CONTEXTO DE LA APLICACIÓN

Antes de abordar el desarrollo de la aplicación es importante poder ubicarla en un contexto conocido. El objetivo principal de la aplicación es permitir la consulta y visualización de los datos del Padrón Municipal de Valladolid, por lo que en este capítulo se realizará una aproximación a los datos abiertos gubernamentales (qué características tienen, por qué motivos se publican o qué proyectos actuales están involucrados en ellos); así como una investigación sobre el padrón municipal para saber en qué consiste, y conocer qué datos incluye y cuál es la mejor forma de representarlos.

3.1.1 DATOS ABIERTOS

Muchas organizaciones recogen grandes cantidades de datos que luego procesan de un modo u otro. Datos abiertos (*open data*) es una iniciativa que persigue que determinados datos de estas organizaciones estén disponibles de forma libre a todo aquel interesado en consultarlos.

Conocimiento abierto

El origen de los datos abiertos se encuentra dentro del conocimiento abierto (*open knowledge*). La definición de conocimiento abierto sirve como punto de partida para numerosas iniciativas como el software libre [50] o el código abierto [41].

Una obra es abierta si cumple los siguientes puntos [39]:

- Disponible íntegramente y con un coste de reproducción razonable.
- Licencia redistribuible y reutilizable.
- El acceso, redistribución y reutilización no debe presentar restricciones tecnológicas.
- La licencia puede exigir el reconocimiento de contribuyentes y creadores de la obra para su redistribución.
- La licencia puede requerir un nombre o número de versión diferente para la distribución de la obra modificada.
- No debe discriminar a ninguna persona o grupo de personas.
- No debe restringir su uso en un ámbito de trabajo específico.
- Los derechos de la obra deben extenderse a cualquier persona a quien le sea redistribuida.
- Los derechos de la obra no deben depender de su pertenencia a un paquete.
- La licencia no debe restringir la distribución de otras obras.

Datos abiertos gubernamentales

Resultan especialmente interesantes los datos abiertos gubernamentales. El volumen de datos manejado por los distintos gobiernos es muy elevado, siendo además una gran parte de esta información es además de carácter público. La posibilidad de que cualquier persona pueda acceder directamente y en cualquier momento a la misma resulta realmente atractiva.

Los datos abiertos gubernamentales generan una gran cantidad de valor en distintos ámbitos, como por ejemplo mejorando la transparencia y el control democrático, fomentando la participación ciudadana, aumentando la innovación o mejorando la eficiencia y eficacia de los servicios públicos entre otras [37].

Datos abiertos en España

Existen numerosos ejemplos de datos abiertos en España. Muchos de ellos pueden consultarse en el catálogo disponible en [38]. Cabe destacar fuentes de datos como el Portal de Datos Abiertos de la Junta de Castilla y León [28], donde podemos encontrar contenidos como el modelo de ordenación del territorio, mapas topográficos, municipios con riesgo potencial de inundaciones, alertas de productos de consumo, servicios públicos o encuestas de población activa entre otros.

Otra fuente interesante es el Catálogo de Información Pública de la Administración General del Estado [4], donde podemos encontrar datos climatológicos, ruedas de prensa del Consejo de Ministros, revistas, publicaciones y artículos diversos, así como registros sobre numerosos ámbitos.

Además de estos ejemplos pueden destacarse los datos abiertos de otras entidades como la Biblioteca Nacional de España [13], el Centro Tecnológico de la Información y la Comunicación [16], el Ayuntamiento de Valladolid [12] o la Fundación Ciudadana Civio [17], responsable de *España en llamas* [15], donde se puede consultar información sobre los distintos incendios ocurridos en España a lo largo del tiempo.

La Web Semántica

La Web Semántica es una Web extendida, dotada de mayor significado a fin de mejorar la experiencia del usuario ofreciendo la respuesta a sus consultas de una forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Para lograrlo, esta Web extendida y basada en el significado se apoya en lenguajes universales.

La Web Semántica ayuda a resolver los problemas de la sobrecarga de información y la heterogeneidad de las fuentes de información problemas presentes en la Web carente de semántica. Gracias a la semántica en la Web, el software es capaz de procesar su contenido, razonar con este, combinarlo y realizar deducciones lógicas para resolver problemas cotidianos automáticamente [54].

Linked Data

La Web Semántica vincula los distintos datos distribuidos en la Web, mediante referencias (al igual que lo hacen los enlaces de las páginas web). No aborda por lo tanto la simple publicación de datos en la Web, sino que hace hincapié en vincularlos a otros, de forma que personas y máquinas puedan explorarlos, pudiendo pasar de unos bloques de información a otros siguiendo un camino de relaciones. Esta iniciativa se conoce como *Linked Data* (Datos Enlazados) [31].

La Web Semántica se construye mediante documentos, al igual que la Web no semántica. A diferencia de la web del hipertexto, donde los enlaces son relaciones entre puntos de documentos HTML, los datos de la Web Semántica enlazan elementos arbitrarios que se describen en RDF.

Por ejemplo, suponiendo que un directorio de empresas publica información especializada relativa a las organizaciones, como su tamaño o área profesional, es posible que desee indicar también información sobre la localización. Ya que en la web existen sitios con grandes bases de datos geográficas, con información pormenorizada sobre las localizaciones, el directorio de empresas puede hacer referencia a los datos geográficos que están dispuestos por esa fuente externa. De esta forma, los datos iniciales de la organización se enriquecen con información que ofrecen los expertos en el ámbito geográfico [32].

La consecución de este objetivo es lo que persigue la iniciativa *Linked Data*, llevada a cabo por el *World Wide Web Consortium* (W3C), una comunidad internacional que desarrolla estándares abiertos que aseguren el crecimiento a largo plazo de la Web [52].

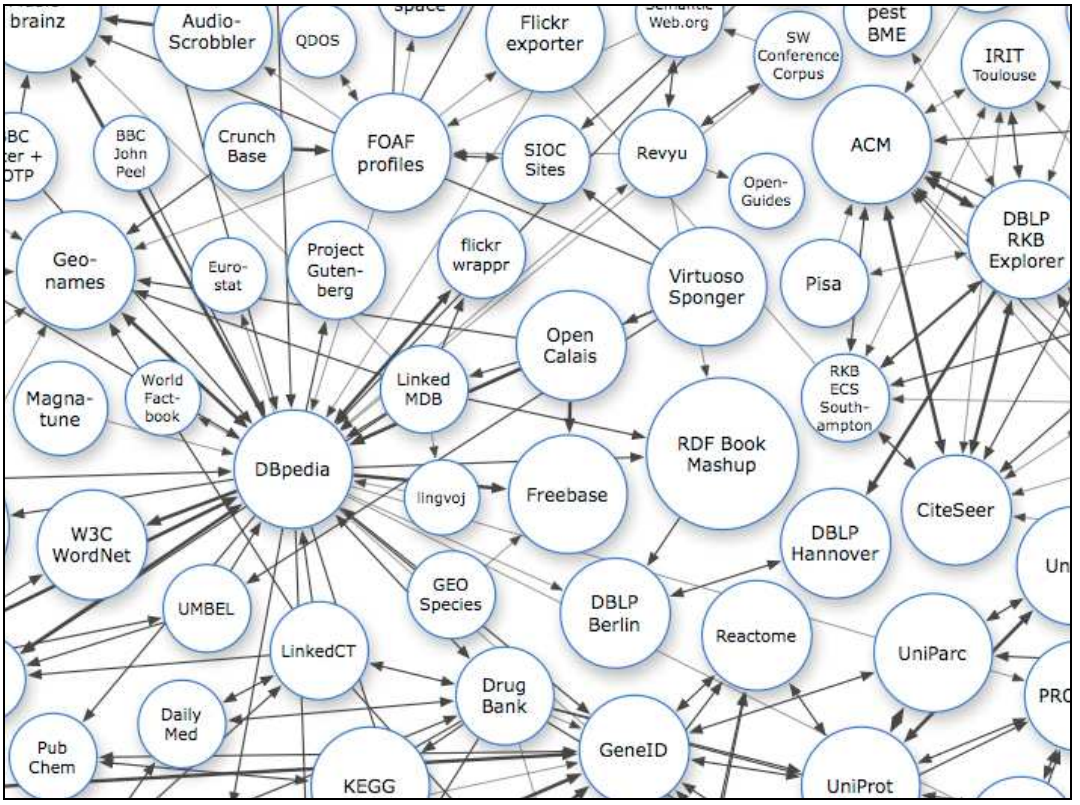

Figura 3.1: *Linked Open Data* [31]

Las cuatro reglas de Tim Berners-Lee

Tim Berners-Lee [51] es el creador de la *World Wide Web (WWW)* como iniciativa para compartir información a través de Internet y a nivel mundial, y actual director de la *World Wide Web Foundation* [53]. Dentro de su interés por la publicación de datos abiertos gubernamentales, Berners-Lee definió cuatro reglas o principios que caracterizan los datos abiertos [33]:

1. Utilizar URIs (*Uniform Resource Identifier*, Identificador Uniforme de Recursos en español) como nombres para identificar recursos.

2. Emplear el HTTP (*Hypertext Transfer Protocol*, Protocolo de Transferencia de Hipertexto en español) para buscar y localizar las URIs.

3. Proporcionar información útil al buscar las URIs, empleando para ello estándares como RDF.

4. Incluir enlaces a otras URIs, de forma que se pueda ampliar la información.

Estos cuatro principios se llaman reglas aunque realmente representan "normas de comportamiento". Romperlas no destruye nada, pero impiden la interconexión de los datos que las incumplen.

Datos abiertos de cinco estrellas

Tim Berners-Lee sugirió en 2010 un sistema de evaluación para datos abiertos basado en estrellas. Su intención es animar a las organizaciones (especialmente a las gubernamentales) a publicar datos abiertos de buena calidad. Para ello definió la siguiente forma de evaluación, de forma que para alcanzar una nueva estrella deben cumplirse las condiciones de todas las estrellas previas, además de la nueva condición.

El sistema de evaluación se muestra a continuación [53]:

★	Disponible en la Web, en cualquier formato, pero bajo una licencia abierta.
★★	Disponible como datos estructurados legibles por máquinas (por ejemplo, una hoja de cálculo en lugar de una imagen escaneada de una tabla).
★★★	Disponible en un formato no propietario (por ejemplo, en CSV en lugar del formato de *Microsoft Excel*.
★★★★	Uso de los estándares abiertos del W3C (RDF y SPARQL) para identificar recursos.
★★★★★	Enlazar los datos con otras fuentes para proporcionar un contexto.

3.1.2 PADRÓN MUNICIPAL

El Padrón municipal es el registro administrativo donde constan los vecinos del municipio. Sus datos constituyen prueba de residencia en el municipio y de domicilio habitual en el mismo. Todo habitante en España está obligado a inscribirse en el Padrón municipal del municipio en que resida. En caso de residir en varios, deberá inscribirse en el que habite durante más tiempo en el año [42].

Datos obligatorios

Los datos que deben ser incluidos obligatoriamente en el Padrón municipal para cada habitante son los siguientes:

- Nombre y apellidos.
- Sexo.
- Domicilio habitual.
- Nacionalidad.
- Fecha y lugar de nacimiento.
- Número del Documento Nacional de Identidad (DNI) o del Documento que lo sustituya en caso de ser un extranjero.

Explotación estadística del Padrón

La explotación estadística del Padrón, realizada a fecha de 1 de enero de cada año, se realiza a partir del fichero derivado de la base padronal del Instituto Nacional de Estadística (INE) del que se obtiene la propuesta de las cifras oficiales, depurándose las variables básicas que contiene el Padrón susceptibles de explotación estadística. Se realizan cruces por: lugar de residencia, sexo, edad, nacionalidad y lugar de nacimiento hasta un nivel de desagregación municipal y por sexo, nacionalidad y edad hasta el nivel de sección censal [43].

El estudio de los datos recopilados se publica en forma de nota de prensa por parte del INE. En la referencia [44] se puede consultar como ejemplo el *Avance de la Explotación Estadística del Padrón a 1 de enero de 2012*.

Censo de Población y Padrón Municipal

El Censo de Población y el Padrón Municipal son dos operaciones que tienden a confundirse dado que permiten determinar el número de habitantes y algunas características de los mismos. Sin embargo, su finalidad es muy distinta.

Las diferencias entre el Padrón Municipal y el Censo de Población son las siguientes [45]:

- La formación, mantenimiento y gestión del Padrón Municipal corresponde a cada uno de los ayuntamientos existentes en España, mientras que en el caso del Censo de Población, corresponde al INE.

- Los datos que figuran en el Padrón están regulados por la Ley de Bases de Régimen Local [29] y la Ley de Protección de Datos de Carácter Personal [30], conteniendo datos muy limitados. Sin embargo el Censo no tiene limitación en las variables a recoger por tratarse de una operación estadística.

- El Padrón es un registro actualizado permanentemente, del que se obtienen las cifras de población anualmente. El Censo de Población se realiza cada diez años.

3.2 CONTEXTO TECNOLÓGICO

En este apartado se explicarán las tecnologías empleadas para el desarrollo del Trabajo Fin de Grado y las razones que motivaron su elección.

3.2.1 ESTUDIO DE POSIBLES TECNOLOGÍAS

Desde un primer momento se plantearon los siguientes requisitos a la hora de escoger un lenguaje de programación en el que desarrollar la aplicación:

- **Accesible desde la Web.** La aplicación debe ser una aplicación web, accesible desde cualquier ordenador con conexión a Internet.

- **Representación gráfica de datos estadísticos.** La información mostrada por la aplicación serán sobre todo representaciones gráficas de datos. La tecnología debe ser capaz no solo de representar esta información correctamente, sino que además es importante contar con APIs o bibliotecas ya desarrolladas para no emplear recursos en un desarrollo propio de forma innecesaria. Sería deseable que el usuario pudiese interactuar con la gráfica, haciendo zoom o pudiendo obtener información adicional por ejemplo.

- **Selección sobre mapas.** El usuario debe poder visualizar un mapa sobre el que seleccionar zonas para realizar consultas sobre ellas.

- **Entorno gráfico.** La aplicación debe contar con un entorno gráfico adecuado, en el que se ubicarán los distintos elementos (menús, seleccionables, gráficas…).

- **Tecnología conocida.** Es importante que la tecnología sea conocida por el desarrollador a fin de no gastar tiempo en aprender una nueva.

- **Facilidad de instalación.** La aplicación debe poder funcionar sin requerir una instalación compleja en el cliente.

Para cubrir estas necesidades se plantean dos posibilidades: Una aplicación del lado del servidor (desarrollada en lenguajes como ASP o PHP) o una aplicación embebida en una página web y que se ejecute en el cliente (como Flash, o un *applet* Java). A continuación se discuten los pros y contras de cada una de estas opciones.

Aplicación del lado del servidor

Un lenguaje del lado del servidor es aquel capaz de generar una página web dinámicamente en el servidor y a continuación enviarla al cliente. Dicha página se construye a como resultado de la ejecución de un *script* al que el cliente no tiene acceso.

En el caso que nos ocupa, el cliente introduciría la información a consultar en unos formularios HTML. Los parámetros llegarían al servidor donde se procesarían y se generaría una nueva vista HTML con los resultados de la consulta, que se mostraría en el cliente (navegador).

La principal ventaja de los lenguajes del lado del servidor es que para ejecutarse basta con tener el intérprete del mismo instalado en el servidor que aloje la aplicación, con lo que la aplicación se puede visualizar en el cliente sin la necesidad de instalar software adicional.

Con lenguajes del lado del servidor como ASP.NET [10] o PHP [46] podría desarrollarse una aplicación web capaz de representar datos en forma de gráficas gracias a bibliotecas externas. Estas gráficas pueden ser estáticas (imágenes) o interactivas (basadas en la tecnología Flash) [47]. Una biblioteca PHP para representación gráfica en imágenes estáticas es *JpGraph* [27]. Un ejemplo de biblioteca capaz de generar gráficas en Flash para el mismo lenguaje es *Open Flash Chart* [40].

Esto supone un primer problema. Es deseable que las gráficas sean interactivas, y existen para ello bibliotecas como la citada *Open Flash Chart*. Sin embargo esta opción introduce una tecnología del lado del cliente a mayores: Flash [5]. Este hecho elimina la principal ventaja de los lenguajes del servidor: su posibilidad de ejecutarse sin tener que instalar software adicional en el cliente. Además requeriría el estudio y aprendizaje de Flash para poder desarrollar la aplicación. Una alternativa consistiría en emplear gráficas estáticas en forma de imágenes generadas a partir de ciertos parámetros. El problema de este tipo de representación es la pérdida de interactividad sobre las gráficas, característica que desde un principio resulta deseable.

La representación de mapas en los que el usuario pueda seleccionar zonas es posible en lenguajes del servidor. Para ello se pueden utilizar APIs externas como *Google Maps API* [20]. El problema de la API de Google es su dependencia, quedando parte de la aplicación a expensas de la disponibilidad de un tercero.

También es posible utilizar mapas dinámicos en Flash, como por ejemplo *Carto:net* [14]. Esto presenta los mismos inconvenientes que el uso de Flash para representar gráficas, ya contemplados con anterioridad.

Dentro de las tecnologías disponibles de lenguajes del lado del servidor, son conocidas por el desarrollador PHP y ASP. Siendo ASP.NET una evolución de este último lenguaje, su aprendizaje no supondría un esfuerzo excesivo.

Aplicación del lado del cliente

Una aplicación del lado del cliente se descarga y se ejecuta en el ordenador del cliente. El mayor inconveniente de este tipo de aplicaciones es que requieren la instalación de software en la máquina que las ejecuta. También consumen recursos de dicha máquina, por lo que si la aplicación es muy pesada puede que no funcione correctamente. Flash o los *applet* de Java [6] son ejemplos de este tipo de lenguajes.

La representación de datos estadísticos en gráficas en Flash no resulta trivial, y debería desarrollarse específicamente para la aplicación. Sin embargo, existe un API para Java capaz de cubrir estas necesidades: *JFreeChart* [24]. Esta API permite generar gráficas tanto estáticas como interactivas de forma sencilla, lo que ahorraría el esfuerzo de tener que realizar un desarrollo propio.

A la hora de representar mapas, Java dispone de varias APIs que permiten trabajar con Sistemas de Información Geográficos (GIS), como puede ser *GeoTools* [18]. El problema de este tipo de bibliotecas es que su uso es excesivamente complejo para las funciones, relativamente triviales, que se requieren en el Trabajo Fin de Grado (visualización y selección de áreas). El API de Google también puede enlazarse con Java. Como ya se indicó en el apartado anterior, el problema de usar esta API reside en la dependencia de un servicio externo.

Java es una tecnología ya conocida por el desarrollador del Trabajo Fin de Grado, por lo que no sería necesario emplear recursos en su aprendizaje. Por el contrarío, Flash es una tecnología con la que nunca ha trabajado.

Conclusiones

Tras analizar los pros y contras de cada una de las posibilidades, se decidió desarrollar la aplicación como un *applet* Java embebido en una página HTML. Las razones que impulsaron esta decisión son las siguientes:

- La tecnología es conocida por el desarrollador con anterioridad al inicio del Trabajo Fin de Grado.
- La API *JFreeChart* cubre de forma completa las necesidades de representación de gráficas requeridas para la aplicación.
- Si bien el empleo de GIS y de la API de Google Maps para trabajar con mapas posee varios inconvenientes, existen alternativas para dibujar mapas como parte de la interfaz gráfica de la aplicación.

- La tecnología Java es muy accesible para cualquier usuario y está muy extendida en la actualidad. Esto minimiza el problema de la instalación de software en los clientes, al tratarse de un recurso disponible con facilidad.

En los siguientes apartados se detallarán las características de Java y de la API *JFreeChart* que se empleará para crear las gráficas.

3.2.2 JAVA

Java es un lenguaje de programación de alto nivel orientado a objetos. Parte de la sintaxis de Java se basa en la de otros lenguajes de programación como C, Cobol o Visual Basic. Los principales cambios que aporta sobre estos lenguajes son un modelo de objetos más simple y la eliminación de herramientas de bajo nivel como pueden ser la manipulación de punteros o la gestión de memoria.

Una de las grandes ventajas de Java es que es un lenguaje completamente multiplataforma. Un programa escrito en Java podrá ejecutarse en cualquier sistema que cuente una Máquina Virtual Java (*Java Virtual Machine*, JVM). Esta característica es importante dentro del contexto del Trabajo Fin de Grado, ya que la aplicación podrá ejecutarse a través de la JVM y de un navegador web, independientemente del sistema operativo que se esté utilizando [22].

Interfaz gráfica

Java posee varias bibliotecas gráficas, como Swing o AWT. El diseño de interfaces gráficas en Java empleando estas bibliotecas y con ayuda de un entorno de desarrollo integrado (IDE, del inglés *Integrated Development Environment*) adecuado resulta bastante sencillo. Estas bibliotecas se pueden utilizar para diseñar interfaces tanto de aplicaciones de escritorio como de *applets*.

Applets

Un *applet* es un programa escrito en Java que puede incluirse en una página HTML. Cuando se visualiza una página que contiene un *applet*, su código se transfiere al cliente y se ejecuta dentro de la Máquina Virtual Java.

Para introducir un *applet* en una página HTML se emplean la etiqueta <applet>. En HTML5 se esta etiqueta no está soportada, y se debe emplear la etiqueta <object> en su lugar. A continuación se muestran dos ejemplos de la integración de un *applet* en una página HTML y HTML5 (Figuras 3.2 y 3.3).

Existen restricciones de seguridad que se aplican a los *applet* y no a los programas de escritorio de Java. Esta seguridad extra se debe a la forma que tienen los *applet* de llegar hasta el usuario. Es más fácil introducir un *applet* con código malicioso en una web para que se ejecute sin conocimiento del cliente que escribir ese código en una aplicación de escritorio que requiera ser

instalad y ejecutada en local. Para evitar estos problemas de seguridad, los *applet* Java cuentan con restricciones extra.

```
<html>
  <head>
    <title>Ejemplo applet HTML</title>
  </head>
  <body>
    <applet
      code=pakage.Applet.class
      archive=container.jar
      width="300"
      height="550"
      alt="No se puede mostrar el applet">
    </applet>
  </body>
</html>
```

Figura 3.2: *Applet* en HTML

```
<html>
  <head>
    <title>Ejemplo applet HTML5</title>
  </head>
  <body>
    <object
      type="application/x-java-applet"
      height="300"
      width="550">
    <param
      name="code"
      value="pakage.Applet"/>
    <param
      name="archive"
      value="container.jar"/>
    No se puede mostrar el applet
    </object>
  </body>
</html>
```

Figura 3.3: *Applet* en HTML5

Antes de introducir estas restricciones, cabe destacar que podemos encontrar dos tipos de *applet*: las *applet* firmadas y las *applet* sin firmar. Algunas restricciones aplicadas a las *applet* sin firmar dejan de estar presentes en las que se encuentran firmadas. Las operaciones que un *applet* no puede hacer si no está firmado son las siguientes [7]:

- Acceder a recursos del cliente como ficheros locales, impresoras, portapapeles o ficheros ejecutables.
- Conectar u obtener recursos de un tercer servidor (distinto del cliente y del servidor desde el que se ha descargado el *applet*.
- Cargar bibliotecas nativas.
- Modificar el *SecurityManager*.
- Crear un *ClassLoader*.
- Acceder a ciertas propiedades del sistema.

Si el *applet* está firmado, antes de su ejecución se mostrará la firma al cliente, poniendo en su conocimiento las operaciones que el código podrá llevar a cabo en su sistema. Si el cliente se fía del emisor de la firma, podrá aceptar la aplicación, concediendo la posibilidad de llevar a cabo las acciones no permitidas habitualmente. Si por el contrario el cliente no confía en el emisor de la firma, podrá rechazarla para minimizar el riesgo de ejecución de código malicioso en su máquina.

La comunicación de los *applet* con los servidores de base de datos presenta también restricciones importantes a tener en cuenta en el Proyecto. Un *applet* Java no puede acceder a un fichero Access (formato en el que se encuentra la base de datos original suministrada para el Proyecto) alojado en un servidor. Se permite, no obstante, la conexión a una base de datos alojada en un servidor a través del driver JDBC correspondiente [8].

Como se puede ver en el capítulo de Análisis, la base de datos MySQL suministrada para el desarrollo del Proyecto sólo permite conexiones locales, esto es, realizadas desde el propio servidor. Al ejecutarse en el cliente, un *applet* Java no podría acceder a ella, por lo que para solventar este problema se recurrió a un script PHP alojado en el mismo servidor que MySQL y desde el que se descargará el *applet*.

3.2.3 JFreeChart

JFreeChart es una API Java para la utilización de gráficas. Las gráficas generadas con la API se pueden exportar como componentes *Swing* de Java (lo que permite su integración en una interfaz gráfica), imágenes (en formatos como JPG o PNG) o ficheros PDF o SGV. La biblioteca se distribuye como código abierto bajo licencia LGPL [24].

La API permite generar gráficas de sectores, de barras, de dispersión, de puntos, pirámides de población e histogramas entre otras. En la Figura 3.4 se pueden observar algunas de las gráficas generadas con *JFreeChart*:

Figura 3.4: Gráficas generadas con *JFreeChart*

`JFreeChart`, `Dataset` y `ChartPanel`

Una gráfica se representa en *JFreeChart* como un objeto de la clase `JFreeChart`, instanciado a partir de la clase `ChartFactory`. Para crear una instancia de `JFreeChart` es necesario un conjunto o juego de datos. Este conjunto de datos es un objeto de una de las clases que implementan la interfaz `Dataset`. Dependiendo del tipo de gráfica que se quiera generar, se necesitará una determinada clase derivada de `Dataset`.

Una vez instanciada la clase `JFreeChart` a partir de un `Dataset`, se pueden modificar sus propiedades o las de sus objetos para configurar la gráfica como sea necesario (leyendas, colores, márgenes…). A partir de la instancia de `JFreeChart` se puede obtener una imagen estática de la gráfica, o puede añadirse a un objeto `ChartPanel`.

`ChartPanel` es un componente Swing ideado para contener y mostrar una gráfica en la interfaz. La clase `ChartPanel` hereda de `JPanel`, por lo que mantiene todos los métodos de esta clase. Requiere un objeto `JFreeChart` para ser instanciada. Una vez situado el objeto `ChartPanel` en una interfaz Swing, permite opciones al usuario como guardar la gráfica como imagen, hacer zoom u obtener información adicional (por ejemplo, situando el puntero sobre un sector en una gráfica de sectores permite mostrar el nombre del elemento y el porcentaje que representa). Estas opciones pueden configurarse para permitir al usuario emplearlas o no.

Documentación de JFreeChart

Para poder trabajar con la API *JFreeChart* se ofrece su documentación completa en formato Javadoc [25], una comunidad de foros en la que resolver dudas [26] y una Guía para Desarrolladores [2] que puede adquirirse en la propia web de la API [24].

3.2.4 PHP

PHP (*PHP [Personal Home Page]: Hypertext Pre-processor*) es un lenguaje de scripting de código abierto, especialmente adecuado para el desarrollo Web y que puede ser incrustado en HTML. El código PHP se ejecuta en el servidor, genera HTML y lo envía al cliente [47].

PHP permite la comunicación con servidores de bases de datos como MySQL. Una forma de establecer la conexión, así como de enviar y recibir datos del servidor MySQL es el uso de los Objetos de Datos PHP (PDO, *PHP Data Objects*).

PDO define una interfaz para el acceso a bases de datos desde PHP. Para acceder a una determinada base de datos se requiere además de la extensión PDO un controlador específico para el SGBD. Proporciona una capa de abstracción de acceso a datos. Las funciones para realizar consultas y obtener datos son independientes del SGBD empleado [48].

Existen otras alternativas para conectar PHP con una base de datos. Sin embargo se ha optado por PDO en lugar de la API MySQL por ser independiente del SGBD (permitiendo la reutilización del código en caso de utilizar un sistema gestor diferente).

IV. ANÁLISIS

4.1 INTRODUCCIÓN

Como ya se explicó en el primer capítulo, el objetivo del Proyecto es el desarrollo de una aplicación web para la visualización y consulta del padrón municipal.

En todo proceso software se hace necesaria la definición de los servicios, funciones y características requeridas en el sistema, así como las restricciones asociadas al mismo. Esta definición se realiza en la fase de análisis, donde se genera un modelo del sistema basado en *qué* funciones realiza, pero dejando el *cómo* las realiza para la fase de diseño.

En este capítulo se describirá el entorno del software, se analizarán los recursos aprovechables ya existentes, y finalmente se detallarán los requisitos y la funcionalidad final de la aplicación.

4.2 ENTORNO

El Proyecto consiste en la creación de una aplicación web, que debe estar alojada en un servidor web. La aplicación accederá a datos persistentes almacenados en una Base de Datos. Dicha Base de Datos deberá estar también alojada en un servidor accesible desde el Applet.

Para la ejecución de la aplicación en el cliente, deberá estar instalado y habilitado en el navegador utilizado el entorno de ejecución de Java.

La aplicación ofrecerá distintas posibilidades de consulta, visualización y comparación de los datos del padrón municipal alojados en la citada base de datos.

4.3 FUNCIONALIDAD REQUERIDA

Los requisitos son una descripción de las necesidades o deseos de un producto. La meta primaria de la fase de requisitos es identificar y documentar lo que en realidad se necesita, en una forma que claramente se lo comunique al cliente y a los miembros del equipo de desarrollo. Se han de definir de manera inequívoca, de modo que se detecten los riesgos y no se produzcan sorpresas a la hora de entregar el producto [3].

4.3.1 REQUISITOS FUNCIONALES

Los requisitos funcionales establecen el comportamiento del sistema.

El sistema deberá permitir:

- Realizar consultas sobre el sexo, nivel de estudios, procedencia y edad de la población de Valladolid en su conjunto *(FRQ-001).*
- Comparar el resultado de una consulta sobre la población de Valladolid con el resultado de una consulta sobre un Distrito de Valladolid a la que se le apliquen los mismos parámetros *(FRQ-002).*
- Realizar consultas sobre el sexo, nivel de estudios, procedencia (por nacionalidad y por comunidad autónoma) y edad de la población de un Distrito de Valladolid, incluyendo o no las Secciones que formen parte del Distrito *(FRQ-003).*
- Realizar consultas sobre el sexo, nivel de estudios, procedencia (por nacionalidad y por comunidad autónoma) y edad de la población de una Zona Estadística de Valladolid *(FRQ-004).*
- Mostrar como resultado de una consulta una representación gráfica, el total de la población seleccionada, el porcentaje que representa esa población con respecto al total de habitantes y los parámetros seleccionados para llevar a cabo la consulta *(FRQ-005).*

- Informar de forma general (sin parámetros) los datos sobre el sexo, nivel de estudios, procedencia y edad de la población de un Distrito de Valladolid *(FRQ-006).*

- Comparar los datos de un Distrito de Valladolid con los de otro Distrito diferente. *(FRQ-007).*

- Informar de forma general (sin parámetros) los datos sobre el sexo, nivel de estudios, procedencia (por nacionalidad y por comunidad autónoma) y edad de la población de una Zona Estadística de Valladolid *(FRQ-008).*

- Comparar los datos de una Zona Estadística de Valladolid con los de otra Zona Estadística. *(FRQ-009).*

- Seleccionar el Distrito a consultar sobre un mapa de Valladolid que incluya información sobre los límites de las distintas divisiones *(FRQ-010).*

- Seleccionar la Zona Estadística a consultar sobre un mapa de Valladolid que incluya información sobre los límites de las distintas divisiones *(FRQ-011).*

- Visualizar la información relativa al sexo en forma de gráfica circular *(FRQ-012).*

- Visualizar la información relativa a la procedencia de individuos de origen español en forma de gráfica de barras *(FRQ-013).*

- Visualizar la información relativa a la procedencia de individuos de origen extranjero en forma de gráfica de barras *(FRQ-014).*

- Visualizar la información relativa al nivel de estudios en forma de gráfica de barras *(FRQ-015).*

- Visualizar la información relativa a la edad en forma de pirámide de población *(FRQ-016).*

- Visualizar las instrucciones básicas del manejo de la aplicación desde de la misma aplicación *(FRQ-017).*

- Ofrecer información sobre el número de individuos que compone cada elemento gráfico mediante el posicionamiento del puntero del ratón sobre el elemento deseado *(FRQ-018).*

- Facilitar la visualización de los datos mostrados en las gráficas mediante una función de zoom *(FRQ-019).*

4.3.2 REQUISITOS NO FUNCIONALES

Los requisitos no funcionales especifican los criterios que pueden usarse para juzgar las propiedades y restricciones de un sistema en lugar de sus comportamientos específicos.

El sistema deberá:

- Ser independiente del Sistema Operativo empleado para acceder a la aplicación *(NFR-001).*
- Funcionar correctamente con la última versión de Java *(NFR-002).*
- Ser independiente del navegador empleado para acceder a la aplicación, siempre que éste permita la ejecución de Applets Java *(NFR-003).*
- Facilitar al usuario el manejo de la aplicación mediante el uso de las metáforas adecuadas *(NFR-004).*
- Impedir al usuario la realización de operaciones no permitidas *(NFR-005).*
- Obtener la información persistente a través de un servidor MySQL que sólo acepta conexiones locales *(NFR-006).*

4.3.3 MODELO FURPS+

Al obtener y capturar los requerimientos de un sistema es difícil saber si ya se tienen contemplados todos, así como poder agruparlos en categorías que simplifiquen su administración. Típicamente se han manejado dos grupos de requisitos: funcionales y no funcionales. Sin embargo esta categorización puede quedarse corta [9].

El modelo FURPS+, desarrollado por Robert Grady, trata de completar dicha categorización. El acrónimo FURPS+ representa las siguientes categorías:

- Funcionalidad (*Functionality*): Esta categoría está desarrollada en los requisitos funcionales.
- Usabilidad (*Usability*): Métodos entendibles y auto descriptivos. Conocimiento intuitivo de las funciones del sistema.
- Fiabilidad (*Reliability*): Capacidad de recuperación frente a errores.
- Rendimiento (*Performance*): El sistema deberá tener un tiempo de respuesta aceptable tras la solicitud de una acción.
- Soporte (*Supportability*): Se podrá utilizar el sistema en cualquier sistema operativo y con cualquier SGBD.

- El signo '+' hace referencia a otros requisitos adicionales como restricciones de diseño (diseño de la base de datos empleada para albergar el padrón municipal), implementación (desarrollado en Java y PHP), interfaz (comunicación del usuario con la aplicación) y físicas (restricciones hardware).

Se ha usado el modelo FURPS+ en el proyecto para agrupar los requisitos a un nivel mayor del que ofrecen los grupos de requisitos funcionales y no funcionales y poder identificarlos y definirlos con mayor facilidad.

4.3.4 ROLES

Se ha definido un único rol para el manejo de la aplicación. Este rol es el del Usuario, que podrá consultar, visualizar y comparar la información del Padrón Municipal, pero que en ningún momento podrá modificar o eliminar.

El único acceso permitido a los datos será desde la propia aplicación, relegando las tareas de mantenimiento de la Base de Datos y la información contenida en ella al administrador de la misma, que operará de forma independiente de la aplicación.

4.3.5 CASOS DE USO

Tras detallar los requisitos, se muestra a continuación se muestra el diagrama de casos de uso de la aplicación[1] (Figura 4.1) así como los distintos casos de uso, mostrando el nombre, descripción, entrada, precondiciones y postcondiciones.

[1] Los diagramas UML han sido diseñados con la aplicación *UMLet* [61].

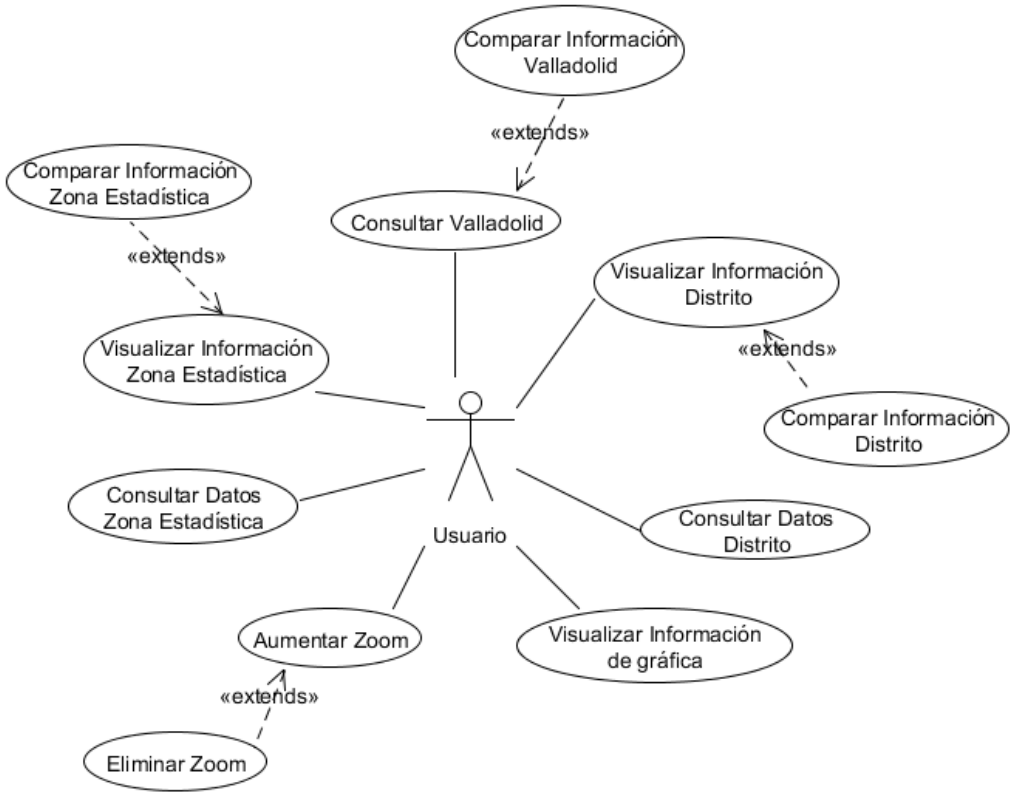

Figura 4.1: Diagrama de Casos de Uso

UC-001	**Consultar Valladolid**	
Descripción	Realizar una consulta a nivel de Valladolid.	
Entrada	-	
Precondiciones	Se ha seleccionado la pestaña “Valladolid (sin división)”.	
Secuencia	*Paso*	*Acción*
	1	El actor selecciona el tipo de dato a representar.
	2	El sistema deshabilita el parámetro del tipo de dato seleccionado y habilita el botón “Consultar”.
	3	El actor selecciona los parámetros de la consulta y pulsa “Consultar”.
	4	El sistema realiza la consulta.
	5	El sistema muestra el resultado de la consulta y habilita los desplegables “Comparar con Distrito o Zona Estadística”.
Postcondiciones	• Se muestra el resultado de la consulta. • Se habilitan los cuatro desplegables para comparar con Distrito o Zona Estadística.	
Excepciones	*Paso*	*Causa*
	4	Se produce un error al establecer la comunicación con la base de datos.
Notas	Se permite realizar consultas sobre: • Sexo • Procedencia (según comunidad autónoma) • Procedencia (según nacionalidad) • Nivel de estudios • Edad	

UC-002	**Comparar información Valladolid**	
Descripción	Comparar una consulta de Valladolid con un Distrito o Zona Estadística.	
Entrada	• Resultado de una consulta sobre Valladolid.	
Precondiciones	Se ha llevado a cabo satisfactoriamente el caso de uso *UC-001*.	
Secuencia	*Paso*	*Acción*
	1	El actor selecciona el Distrito o Zona Estadística con la que comparar.
	2	El sistema realiza la consulta.
	3	El sistema muestra el resultado de la consulta.
Postcondiciones	• Se muestra el resultado de la consulta.	
Excepciones	*Paso*	*Causa*
	2	Se produce un error al establecer la comunicación con la base de datos.

UC-003	**Visualizar información Distrito**	
Descripción	Visualizar la información general sobre un tipo de distribución de un Distrito.	
Entrada	-	
Precondiciones	Se ha seleccionado la pestaña “Información General” dentro de “División por Distrito”.	
Secuencia	*Paso*	*Acción*
	1	El actor selecciona un Distrito en el mapa.
	2	El sistema habilita el desplegable “Distribución”.
	3	El actor selecciona el tipo de distribución.
	4	El sistema realiza la consulta.
	5	El sistema muestra el resultado de la consulta y habilita el desplegable “Comparar con”.
Postcondiciones	• Se muestra la información general del tipo de distribución seleccionado para el Distrito. • Se habilita el desplegable “Comparar con”.	
Excepciones	*Paso*	*Causa*
	4	Se produce un error al establecer la comunicación con la base de datos.

UC-004	**Comparar información Distrito**	
Descripción	Comparar la información ya mostrada de un Distrito con otro Distrito distinto.	
Entrada	• Información de una distribución para un Distrito.	
Precondiciones	Se ha llevado a cabo satisfactoriamente el caso de uso *UC-003*.	
Secuencia	*Paso*	*Acción*
	1	El actor selecciona el Distrito con el que comparar.
	2	El sistema realiza la consulta.
	3	El sistema muestra el resultado de la consulta.
Postcondiciones	• Se muestra la información general del tipo de distribución seleccionado para el Distrito.	
Excepciones	*Paso*	*Causa*
	2	Se produce un error al establecer la comunicación con la base de datos.

UC-005	**Consultar datos Distrito**	
Descripción	Realizar una consulta sobre un Distrito.	
Entrada	-	
Precondiciones	Se ha seleccionado la pestaña "Consulta" dentro de "División por Distrito".	
Secuencia	*Paso*	*Acción*
	1	El actor selecciona un Distrito en el mapa y el tipo de dato a representar.
	2	El sistema deshabilita el parámetro del tipo de dato seleccionado y habilita el botón "Consultar" y el desplegable "Sección".
	3	El actor selecciona los parámetros de la consulta y pulsa "Consultar".
	4	El sistema realiza la consulta.
	5	El sistema muestra el resultado de la consulta.
Postcondiciones	• Se muestra el resultado de la consulta.	
Excepciones	*Paso*	*Causa*
	4	Se produce un error al establecer la comunicación con la base de datos
Notas	Se permite realizar consultas sobre: • Sexo • Procedencia (según comunidad autónoma) • Procedencia (según nacionalidad) • Nivel de estudios • Edad	

UC-006	**Visualizar información Zona Estadística**	
Descripción	Visualizar la información general sobre un tipo de distribución de una Zona Estadística.	
Entrada	-	
Precondiciones	Se ha seleccionado la pestaña "Información General" dentro de "División por Zona Estadística".	
Secuencia	*Paso*	*Acción*
	1	El actor selecciona una Zona Estadística en el mapa.
	2	El sistema habilita el desplegable "Distribución".
	3	El actor selecciona el tipo de distribución.
	4	El sistema realiza la consulta.
	5	El sistema muestra el resultado de la consulta y habilita el desplegable "Comparar con".
Postcondiciones	• Se muestra la información general del tipo de distribución seleccionado para la Zona Estadística. • Se habilita el desplegable "Comparar con".	
Excepciones	*Paso*	*Causa*
	4	Se produce un error al establecer la comunicación con la base de datos.

UC-007	**Comparar información Zona Estadística**	
Descripción	Comparar la información ya mostrada de una Zona Estadística con otra Zona Estadística distinta.	
Entrada	• Información de una distribución para una Zona Estadística.	
Precondiciones	Se ha llevado a cabo satisfactoriamente el caso de uso *UC-006*.	
Secuencia	*Paso*	*Acción*
	1	El actor selecciona la Zona Estadística con la que comparar.
	2	El sistema realiza la consulta.
	3	El sistema muestra el resultado de la consulta.
Postcondiciones	• Se muestra la información general del tipo de distribución seleccionado para la Zona Estadística.	
Excepciones	*Paso*	*Causa*
	2	Se produce un error al establecer la comunicación con la base de datos.

UC-008	**Consultar datos Zona Estadística**	
Descripción	Realizar una consulta sobre una Zona Estadística.	
Entrada	-	
Precondiciones	Se ha seleccionado la pestaña “Consulta” dentro de “División por Zona Estadística”.	
Secuencia	*Paso*	*Acción*
	1	El actor selecciona una Zona Estadística en el mapa y el tipo de dato a representar.
	2	El sistema deshabilita el parámetro del tipo de dato seleccionado y habilita el botón “Consultar”.
	3	El actor selecciona los parámetros de la consulta y pulsa “Consultar”.
	4	El sistema realiza la consulta.
	5	El sistema muestra el resultado de la consulta.
Postcondiciones	• Se muestra el resultado de la consulta.	
Excepciones	*Paso*	*Causa*
	4	Se produce un error al establecer la comunicación con la base de datos
Notas	Se permite realizar consultas sobre: • Sexo • Procedencia (según comunidad autónoma) • Procedencia (según nacionalidad) • Nivel de estudios • Edad	

UC-009	**Aumentar zoom**	
Descripción	Aumentar zoom sobre una gráfica.	
Entrada	-	
Precondiciones	• El sistema muestra una gráfica.	
Secuencia	*Paso*	*Acción*
	1	El actor hace clic sobre la gráfica y arrastra el ratón hacia abajo (gráfica de barras) o hacia la derecha (pirámide de población).
	2	El sistema aumenta el zoom sobre la zona seleccionada en el arrastre.
Postcondiciones	• Se muestra una parte de la gráfica a mayor tamaño.	
Excepciones	-	

UC-010	**Eliminar zoom**	
Descripción	Eliminar zoom sobre una gráfica de barras (distribución de procedencia o nivel de estudios).	
Entrada	-	
Precondiciones	• Se ha llevado a cabo satisfactoriamente el caso de uso *UC-011*.	
Secuencia	*Paso*	*Acción*
	1	El actor hace clic sobre la gráfica y arrastra el ratón arriba (gráfica de barras) o hacia abajo (pirámide de población).
	2	El sistema muestra la gráfica original.
Postcondiciones	• Se muestra la gráfica original.	
Excepciones	-	

UC-011	**Visualizar información de gráfica**	
Descripción	Visualizar la información relativa a un elemento de una gráfica (barra o porción).	
Entrada	-	
Precondiciones	• El sistema muestra una gráfica.	
Secuencia	*Paso*	*Acción*
	1	El actor posiciona el puntero del ratón sobre una sección o barra de una gráfica.
	2	El sistema muestra información sobre el dato seleccionado y el número de individuos que contiene.
Postcondiciones	• Se muestra la información solicitada.	
Excepciones	-	

4.4 MODELO DEL DOMINIO INICIAL

Tras el análisis del software disponible, la especificación de los requisitos y el estudio de los casos de uso se puede desarrollar el diagrama de clases inicial, que proporcionará una idea general de las clases que manejará el sistema:

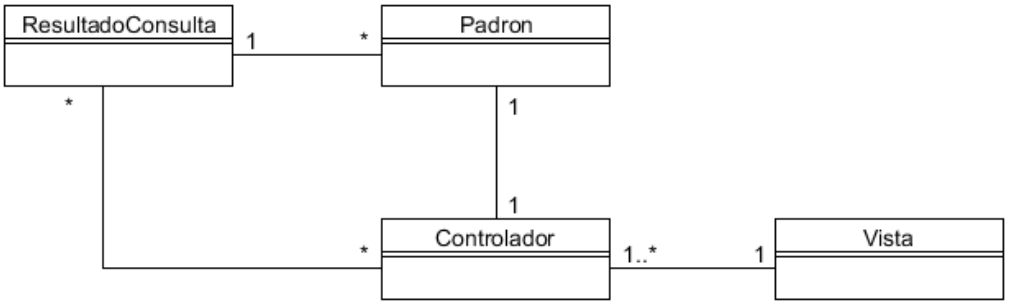

Figura 4.2: Diagrama de Clases inicial

Las clases `Vista`, representa la interfaz de usuario. Finalmente estará conformada por varias clases, pero a estas alturas del desarrollo se define como una única entidad. Será en el capítulo de Diseño cuando se entre en detalle sobre los elementos de la Vista.

La clase `Controlador` se encarga de recoger los eventos de `Vista`. En función de dichos eventos realizará cambios sobre la propia `Vista`, accediendo y solicitando información a la clase `Padron` si es necesario.

`Padron` recibirá las peticiones de datos por parte de `Controlador`, solicitará a su vez la información al script PHP (no contemplado en el diagrama) y creará una instancia de `ResultadoConsulta`, con contenido como la gráfica solicitada, el total de población representada y su porcentaje respecto al total. Esta instancia será devuelta a `Controlador` para que realice los cambios necesarios sobre `Vista`.

V. DISEÑO

5.1 INTRODUCCIÓN

En el presente capítulo se detallará el diseño realizado para la aplicación. El diseño ha de cumplir los requisitos detallados en el capítulo anterior, e incluir la descripción de las clases que contiene la aplicación, su relación, las operaciones que el usuario puede llevar a cabo, la estructura de la base de datos… Para ello se utilizarán diagramas UML [3].

También se detallarán y explicarán las decisiones tomadas durante el desarrollo y cómo han influido éstas en el resultado final.

5.2 ESTUDIO DE LA BASE DE DATOS PROPORCIONADA

Para el desarrollo del Proyecto fue proporcionada una base de datos con el contenido del padrón municipal de 2011. Esta base de datos se iba a utilizar inicialmente para probar la aplicación con datos reales. Sin embargo, y como se ha podido ver en el capítulo “Planificación”, los objetivos

del Proyecto se vieron modificados y se sustituyó la idea inicial de acceso a los datos vía *open data* por una base de datos relacional.

Esta base de datos tuvo que ser analizada y modificada para integrarse en la versión final del Proyecto. La sección actual presenta el análisis de la base de datos original. Las modificaciones llevadas a cabo finalmente pueden leerse en una sección posterior de este mismo capítulo.

5.2.1 ANÁLISIS DE LA BASE DE DATOS

La base de datos suministrada para el desarrollo del Proyecto era un fichero Microsoft Access 2000 [57]. Esta base de datos presenta algunos problemas como se podrá comprobar a continuación. Es posible que se trate de un volcado parcial y no de la base de datos empleada en la explotación real del padrón, pero se trata de la única fuente de información proporcionada para el Proyecto. Se describen a continuación las tablas de que consta la base de datos, su contenido y los problemas observados en un primer momento:

`Padron_habitantes_1_1_2012`

COLUMNA	TIPO	NOTAS	CLAVE
COD NUCLEO FAMILIAR	INTEGER		
DISTRITO	INTEGER		
SECCION	INTEGER		
AÑO NACI	INTEGER		
MES NACI	INTEGER		
PROVIN NACI	INTEGER		
COD SEXO	INTEGER		
EDAD	INTEGER		
COD NACIONALIDAD	INTEGER		
COD NIVEL ESTUDIOS	INTEGER		

Figura 5.1: Tabla "`Padron_habitantes_1_1_2012`"

`Padron_habitantes_1_1_2012` es la tabla principal de la base de datos. Contiene 314936 entradas, una por cada habitante registrado en el padrón municipal. La descripción de sus campos es la siguiente:

1. `COD NUCLEO FAMILIAR`: Identificador del núcleo familiar al que pertenece el individuo. Se entiende por núcleo familiar vivienda física en la que habitan una o más personas.
2. `DISTRITO`: Distrito al que pertenece el individuo.
3. `SECCIÓN`: Sección dentro del Distrito a la que pertenece el individuo.
4. `AÑO NACI`: Año de nacimiento del individuo.
5. `MES NACI`: Mes de nacimiento del individuo.
6. `PROVIN NACI`: Código de la provincia de nacimiento del individuo (ver tabla `CODIGO_PROVINCIA_NACIMIENTO`).

7. COD SEXO: Código del sexo del individuo (ver tabla CODIGO_SEXO).
8. EDAD: Edad del individuo.
9. COD NACIONALIDAD: Código de la nacionalidad del individuo (ver tabla CODIGO_PAIS_NACIONA_NACIMI(continente)).
10. COD NIVEL ESTUDIOS: Código del nivel de estudios del individuo (ver tabla CODIGO_NIVEL ESTUDIOS).

Sorprende no encontrar clave primaria con la que identificar unívocamente a cada individuo. A la vista de los campos y las inexistentes condiciones, dos individuos que pertenezcan al mismo núcleo familiar, tengan los mismos estudios, sean del mismo sexo, hayan nacido el mismo día tendrán entradas idénticas en la tabla.

Resalta también la ausencia de claves foráneas que relacionen la tabla con el resto de tablas del sistema, pese a que la tabla incluye campos identificadores de otras tablas (códigos de sexo, estudios, provincia y nacionalidad).

CODIGO_SEXO

COLUMNA	TIPO	NOTAS	CLAVE
IdDirección	INTEGER	No nulo	Clave Primaria
Nombre	VARCHAR(50)	No nulo	

Figura 5.2: Tabla "CODIGO_SEXO"

IdDirección	Nombre
1	varon
6	mujer

Figura 5.3: Contenido "CODIGO_SEXO"

CODIGO_SEXO contiene los identificadores del sexo de los individuos del padrón. Incluye por lo tanto una entrada para varones y otra para mujeres. Al contrario que Padron_habitantes_1_1_2012, codigo_sexo cuenta con clave primaria.

CODIGO_PROVINCIA_NACIMIENTO

COLUMNA	TIPO	NOTAS	CLAVE
Código	DOUBLE	Indexado	
Literal	VARCHAR(255)	No nulo	

Figura 5.4: Tabla "CODIGO_SEXO"

CODIGO_PROVINCIA_NACIMIENTO contiene los identificadores de las distintas provincias españolas en las que han nacido los individuos. Cuenta con un total de 53 entradas,

siendo los códigos del 1 al 50 para las distintas provincias (ordenadas alfabéticamente), 51 para Ceuta, 52 para Melilla y el último registro (`Código` 66) para Extranjero.

De nuevo, la tabla no tiene claves. La descripción de las provincias se encuentra en castellano en la mayoría de los casos, si bien algunas entradas están escritas en la lengua co-oficial de la provincia y otras en ambos idiomas (separados por una barra). Resulta por lo tanto un poco confuso el idioma a emplear a la hora de realizar consultas sobre la tabla.

CODIGO_PAIS_NACIONA_NACIMI(continente)

COLUMNA	TIPO	NOTAS	CLAVE
Id	INTEGER	Autonumérico	Clave Primaria
CODIGO	INTEGER		
COD_CONTINENTE	INTEGER		
CONTINENTE	VARCHAR(255)		
PAIS	VARCHAR(255)		

Figura 5.5: Tabla "CODIGO_PAIS_NACIONA_NACIMI(continente)"

`CODIGO_PAIS_NACIONA_NACIMI(continente)` contiene los identificadores de los países y continentes de nacimiento de los individuos. Está poblada por 196 registros. La descripción de sus campos es la siguiente:

11. `Id`: Identificador de la tabla.
12. `CODIGO`: Código del país.
13. `COD_CONTINENTE`: Código del continente.
14. `CONTINENTE`: Nombre del continente.
15. `PAIS`: Nombre del país.

La tabla presenta algunos problemas de diseño, como el uso del campo Id como clave primaria de forma innecesaria (el campo `CODIGO` es referenciado desde la tabla `Padron_habitantes_1_1_2012` y no contiene duplicados en esta, por lo que podría funcionar como clave primaria) y la inclusión de los campos `COD_CONTINENTE` y `CONTINENTE`. El primero tendría sentido si referenciase a otra tabla que incluyese el nombre del continente, y la inclusión del segundo tendría lógica sin estar acompañado del campo `COD_CONTINENTE`. La inclusión de ambos campos dentro de la misma tabla no resulta recomendable por poder llevar a inconsistencias.

CODIGO_NIVEL ESTUDIOS

COLUMNA	TIPO	NOTAS	CLAVE
CNES	INTEGER		
DESCRIPCION	VARCHAR(150)		

Figura 5.6: Tabla "CODIGO_NIVEL ESTUDIOS"

CNES	DESCRIPCION
0	NO APLICABLE POR SER MENOR DE 10 AÑOS
10	NO SABE LEER NI ESCRIBIR
11	NO SABE LEER NI ESCRIBIR
20	TITULCIÓN INFERIOR A GRADUADO ESCOLAR
21	SIN ESTUDIOS
22	ENSEÑANZA PRIMARIA INCOMPLETA. CINCO CURSOS DE EGB O EQUIVALENTE O CERTIFICADO DE ESCOLARIDAD O EQUIVALENTE
30	GRADUADO ESCOLAR O EQUIVALENTE
31	BACHILLER ELEMENTAL. GRADUADO ESCOLAR. EGB COMPLETA. PRIMARIA COMPLETA. ESO.
32	FORMACIÓN PROFESIONAL DE PRIMER GRADO. FORMACIÓN PROFESIONAL DE GRADO MEDIO. OFICIALÍA INDUSTRIAL.
40	BACHILLER, FORMACIÓN PROFESIONAL DE SEGUNDO GRADO O TÍTULOS EQUIVALENTES O SUPERIORES.
41	FORMACIÓN PROFESIONAL DE SEGUNDO GRADO. FORMACIÓN PROFESIONAL DE GRADO SUPERIOR. MAESTRÍA INDUSTRIAL.
42	BACHILLER SUPERIOR. BUP. BACHILLER LOGSE.
43	OTRAS TITULACIONES MEDIAS (AUXILIAR DE CLÍNICA, SECRETARIADO, PROGRAMADOR DE INFORMÁTICA. AUXILIAR DE VUELO.
44	DIPLOMADOS EN ESCUELAS UNIVERSITARIAS (EMPRESARIALES, PROFESORADO DE EGB, ATS Y SIMILARES)
45	ARQUITECTO O INGENIERO TÉCNICO
46	LICENCIADO UNIVERSITARIO. ARQUITECTO O INGENIERO SUPERIOR
47	TITULADOS DE ESTUDIOS SUPERIORES NO UNIVERSITARIOS
48	DOCTORADO Y ESTUDIOS DE POSTGRADO O ESPECIALIZACIÓN PARA LICENCIADOS
99	DESCONOCIDO

Figura 5.7: Contenido de "CODIGO_NIVEL ESTUDIOS"

CODIGO_NIVEL ESTUDIOS contiene los identificadores de los niveles de estudios de los distintos individuos. Consta de dos campos, CNES como identificador y DESCRIPCION con la descripción del nivel de estudios correspondiente. Nuevamente la tabla carece de claves.

Atendiendo al contenido de la tabla, cabe destacar que hay niveles de estudios repetidos (como "NO SABE LEER NI ESCRIBIR") con identificadores diferentes y descripciones que se prestan a confusión (existen las entradas "GRADUADO ESCOLAR O EQUIVALENTE" y "BACHILLER ELEMENTAL. GRADUADO ESCOLAR. EGB COMPLETA. PRIMARIA

COMPLETA. ESO.": ¿a cuál de las dos pertenecería un individuo con el título de "Graduado Escolar"?).

Si además consultamos el contenido de la tabla Padron_habitantes_1_1_2012, encontramos individuos de edad inferior a 10 años con un nivel de estudios distinto a "NO APLICABLE POR SER MENOR DE 10 AÑOS".

5.2.2 PROBLEMAS DETECTADOS EN LA BASE DE DATOS

Tras el estudio de la base de datos proporcionada, se detallan a continuación los principales problemas detectados tanto en su diseño como en su contenido. Estos problemas se tendrán en consideración a la hora de incluir modificaciones sobre la base de datos de cara a la versión que utilizará la aplicación.

5.2.2.1 Problemas de diseño y estructura

Los principales problemas del diseño de la base de datos son los siguientes:

16. Ausencia de claves primarias en las tablas "Padron_habitantes_1_1_2012", "CODIGO_PROVINCIA_NACIMIENTO", "CODIGO_SEXO" y "CODIGO_NIVEL ESTUDIOS".

17. Ausencia de claves foráneas en "Padron_habitantes_1_1_2012".

18. Nombres de tablas y campos no recomendables (caracteres no alfanuméricos, alternancia de mayúsculas y minúsculas…).

19. La tabla "CODIGO_PAIS_NACIONA_NACIMI(continente)" contiene dos campos ("COD_CONTINENTE" y "CONTINENTE") que presentan nueve posibles pares de valores. El primero de ellos es un código para identificar el continente y el segundo el nombre del continente. No resulta adecuado conservar ambos campos, ya que pueden dar lugar a inconsistencias.

20. Campo clave "Id" en la tabla "CODIGO_PAIS_NACIONA_NACIMI(continente)" innecesario, ya que el campo "CODIGO" funciona como un campo único (aunque no esté definido como tal).

5.2.2.2 Problemas de contenido

Los principales problemas del contenido de la base de datos son los siguientes:

21. Datos "no estandarizados" (distintos idiomas, erratas, faltas de ortografía, empleo de mayúsculas y minúsculas indistintamente…).

22. Los niveles de estudios (tabla "`CODIGO_NIVEL ESTUDIOS`") se prestan a confusión (estudios repetidos en distintos niveles, entradas repetidas con distinto identificador, falta de consecuencia con la tabla "`Padron_habitantes_1_1_2012`" en el caso de la categoría "`NO APLICABLE POR SER MENOR DE 10 AÑOS`"…).

23. Se echa en falta la relación entre Distrito-Sección y Zona Estadística. No es problema de la base de datos, pero a la hora de utilizarla en el Proyecto es un punto que debe ser corregido.

5.3. MODIFICACIONES SOBRE LA BASE DE DATOS

Para poder acceder a la base de datos desde la aplicación y obtener respuesta en un tiempo razonable se hace necesario introducir modificaciones en la misma. Los cambios a introducir en la base de datos son los siguientes:

- **SGBD alojado en un servidor.** Debido a las restricciones que presentan los Applet Java (que se ejecutan en el cliente y no en el servidor), no resulta posible realizar conexiones a servidores de bases de datos remotos. Para subsanar este problema se decidió emplear un script PHP que actuase como puente entre la aplicación y el almacenamiento persistente. Con el fin de mejorar el rendimiento de la base de datos se optó por transferirla a un servidor MySQL.

- **Claves primarias.** Muchas tablas carecían de claves primarias, necesarias para tener una base de datos consistente y mejorar el tiempo de acceso a los datos.

- **Claves foráneas.** La tabla "`Padron_habitantes_1_1_2012`" utiliza referencias a otras tablas. Sin embargo estas relaciones no están definidas en el diseño de la base de datos. Para aumentar la eficiencia y la consistencia del sistema es necesario emplear claves foráneas que relacionen los campos de "`Padron_habitantes_1_1_2012`" con los de las tablas referenciadas.

- **Índices.** Para mejorar los tiempos de respuesta a la hora de realizar consultas sobre la base de datos resulta interesante definir índices que mejoran la eficiencia de las mismas.

- **Nuevas tablas.** Para cubrir los requisitos planteados en el capítulo de Análisis se hace necesaria la inclusión de nuevas tablas para almacenar datos no contemplados en la base de datos original.

- **Nuevos campos en tablas existentes.** Por el mismo motivo será necesario incluir nuevos campos en las tablas existentes, a fin de relacionarlas con las nuevas tablas y aplicar un formato estándar al contenido de la base de datos.

Se ha decidido no eliminar ningún campo de las tablas existentes en la base de datos. Los motivos de la determinación de respetar la base de datos original en medida de lo posible responden a facilitar la actualización de los datos en padrones futuros. Estas actualizaciones deberían afectar exclusivamente a la tabla "Padron_habitantes_1_1_2012", por lo que para ser llevadas a cabo sólo sería necesario conocer el nuevo nombre de la tabla y sus campos.

5.3.1 MODIFICACIÓN DE Padron_habitantes_1_1_2012

Las modificaciones realizadas sobre la tabla "Padron_habitantes_1_1_2012" son las siguientes:

- **Nuevo nombre para la tabla.** Se renombró la tabla por "PADRON_HABITANTES".

- **Nuevos nombres para los campos.** Los campos se renombraron de la siguiente forma:

Nombre original	Nuevo nombre
COD NUCLEO FAMILIAR	NUCLEO_FAMILIAR
DISTRITO	DISTRITO
SECCION	SECCION
AÑO NACI	ANHO_NACIMIENTO
MES NACI	MES_NACIMIENTO
PROVIN NACI	ID_PROVINCIA_NAC
COD SEXO	ID_SEXO
EDAD	EDAD
COD NACIONALIDAD	ID_NACIONALIDAD
COD NIVEL ESTUDIOS	ID_ESTUDIOS

Figura 5.8: Renombrado de "PADRON_HABITANTES"

- **Nuevos campos.** Se añadió el campo "ID" como clave primaria.

- **Diseño de la tabla.** Se muestra a continuación el tipo de cada columna, si es clave o no y sus propiedades.

COLUMNA	TIPO	NOTAS	CLAVE
ID	INT(4)	Autoinc. No nulo	Clave Primaria
NUCLEO_FAMILIAR	INT(4)	No nulo	
DISTRITO	INT(1)	Índice No nulo	Clave Foránea (CODIGO_DTO_SECCION.DISTRITO)
SECCION	INT(1)	Índice No nulo	Clave Foránea (CODIGO_DTO_SECCION.SECCION)
ANHO_NACIMIENTO	INT(2)	No nulo	
MES_NACIMIENTO	INT(1)	No nulo	
ID_PROVINCIA_NAC	INT(1)	Índice No nulo	Clave Foránea (CODIGO_PROVINCIA.ID)
ID_SEXO	INT(1)	No nulo	Clave Foránea (CODIGO_SEXO.ID)
EDAD	INT(1)	Índice No nulo	
ID_NACIONALIDAD	INT(2)	Índice No nulo	Clave Foránea (CODIGO_PAIS.ID)
ID_ESTUDIOS	INT(1)	Índice No nulo	Clave Foránea (CODIGO_ESTUDIOS.ID)

Figura 5.9: Estructura de "PADRON_HABITANTES"

5.3.2 MODIFICACIÓN DE CODIGO_SEXO

Las modificaciones realizadas sobre la tabla "CODIGO_SEXO" son las siguientes:

- **Nuevos nombres para los campos.** Los campos se renombraron de la siguiente forma:

Nombre original	Nuevo nombre
IdDirección	ID
Nombre	SEXO

Figura 5.10: Renombrado de "CODIGO_SEXO"

- **Diseño de la tabla.** Se muestra a continuación el tipo de cada columna, si es clave o no y sus propiedades.

COLUMNA	TIPO	NOTAS	CLAVE
ID	INT(1)	No nulo	Clave Primaria
SEXO	VARCHAR(5)	Índice No nulo	

Figura 5.11: Estructura de "CODIGO_SEXO"

5.3.3 MODIFICACIÓN DE CODIGO_PROVINCIA_NACIMIENTO

Las modificaciones realizadas sobre la tabla "CODIGO_PROVINCIA_NACIMIENTO" son las siguientes:

- **Nuevo nombre para la tabla.** Se renombró la tabla por "CODIGO_PROVINCIA".
- **Nuevos nombres para los campos.** Los campos se renombraron de la siguiente forma:

Nombre original	Nuevo nombre
Código	ID
Literal	PROVINCIA

Figura 5.12: Renombrado de "CODIGO_PROVINCIA"

- **Nuevos campos.** Se añadió el campo "ID_COM_AUTON" para identificar la comunidad autónoma de la que forma parte la provincia.
- **Diseño de la tabla.** Se muestra a continuación el tipo de cada columna, si es clave o no y sus propiedades.

COLUMNA	TIPO	NOTAS	CLAVE
ID	INT(1)	No nulo	Clave Primaria
PROVINCIA	VARCHAR(22)	No nulo	
ID_COM_AUTON	INT(1)	Índice No nulo	Clave Foránea (CODIGO_COM_AUTON.ID)

Figura 5.13: Estructura de "CODIGO_PROVINCIA"

5.3.4 MODIFICACIÓN DE CODIGO_PAIS_NACIONA_NACIMI(continente)

Las modificaciones realizadas sobre la tabla "CODIGO_PAIS_NACIONA_NACIMI (continente)" son las siguientes:

- **Nuevo nombre para la tabla.** Se renombró la tabla por "CODIGO_PAIS".
- **Nuevos nombres para los campos.** Los campos se renombraron de la siguiente forma:

Nombre original	Nuevo nombre
CODIGO	ID
COD_CONTINENTE	ID_CONTINENTE
CONTINENTE	CONTINENTE
PAIS	PAIS

Figura 5.14: Renombrado de "CODIGO_PAIS"

- **Nuevos campos.** Se añadió el campo "COD_CONTINENTE" para estandarizar el nombre del continente de cara a las consultas, utilizándose una abreviatura de cuatro letras en mayúsculas para identificar el continente.

- **Diseño de la tabla.** Se muestra a continuación el tipo de cada columna, si es clave o no y sus propiedades.

COLUMNA	TIPO	NOTAS	CLAVE
ID	INT(2)	No nulo	Clave Primaria
ID_CONTINENTE	INT(1)	No nulo	
CONTINENTE	VARCHAR(17)	No nulo	
PAIS	VARCHAR(31)	Índice No nulo	
COD_CONTINENTE	VARCHAR(4)	Índice No nulo	

Figura 5.15: Estructura de "CODIGO_PAIS"

5.3.5 MODIFICACIÓN DE CODIGO_NIVEL ESTUDIOS

Las modificaciones realizadas sobre la tabla "CODIGO_NIVEL ESTUDIOS" son las siguientes:

- **Nuevo nombre para la tabla.** Se renombró la tabla por "CODIGO_ESTUDIOS".

- **Nuevos nombres para los campos.** Los campos se renombraron de la siguiente forma:

Nombre original	Nuevo nombre
CNES	ID
DESCRIPCION	ESTUDIOS

Figura 5.16: Renombrado de "CODIGO_ESTUDIOS"

- **Nuevos campos.** Se añadieron los campos "ESTUDIOS_INE" y "CODIGO_INE" para, respectivamente, englobar los estudios dentro de un marco estandarizado e identificar mediante un código de cinco letras en mayúscula el nivel de estudios.

- **Diseño de la tabla.** Se muestra a continuación el tipo de cada columna, si es clave o no y sus propiedades.

COLUMNA	TIPO	NOTAS	CLAVE
ID	INT(1)	No nulo	Clave Primaria
ESTUDIOS	VARHCAR(150)	No nulo	
ESTUDIOS_INE	VARCHAR(39)	No nulo	
CODIGO_INE	VARCHAR(5)	No nulo Índice	

Figura 5.17: Estructura de "CODIGO_ESTUDIOS"

5.3.6 TABLA CODIGO_DTO_SECCION

Para establecer la relación entre el Distrito y Sección con la Zona Estadística, a fin de permitir consultas sobre éste último tipo de distribución territorial se hizo necesaria una nueva tabla ("CODIGO_DTO_SECCION") que se relacionara con las ya existentes para poder ubicar a cada individuo en una determinada zona estadística.

La estructura de la tabla "CODIGO_DTO_SECCION" es la siguiente:

COLUMNA	TIPO	NOTAS	CLAVE
ZONA_ESTADISTICA	VARCHAR(3)	Índice No nulo	
DISTRITO	INT(1)	No nulo	Clave Primaria
SECCION	INT(1)	No nulo	Clave Primaria

Figura 5.18: Estructura de "CODIGO_DTO_SECCION"

Una Zona Estadística engloba una o más Secciones de uno o más Distritos. Por lo tanto para cada Zona Estadística habrá tantas entradas como Secciones la conformen. Cada Sección depende de un Distrito, por lo que cada tupla de la tabla indicará qué Sección (campo "SECCION") de qué Distrito (campo "DISTRITO") pertenece a una determinada Zona Estadística (campo "ZONA_ESTADISTICA").

5.3.7 TABLA CODIGO_COM_AUTON

Para determinar a qué Comunidad Autónoma pertenece cada Provincia se hizo necesaria la creación de la tabla "CODIGO_COM_AUTON". Es necesario conocer está relación puesto que en los requisitos se definió la necesidad de permitir las consultas por y agrupadas según Comunidad Autónoma.

La estructura de la tabla "CODIGO_COM_AUTON" es la siguiente:

COLUMNA	TIPO	NOTAS	CLAVE
ID	INT(1)	No nulo	Clave Primaria
COM_AUTON	VARCHAR(22)	No nulo	
COD_COM_AUTON	VARCHAR(3)	Índice No nulo	

Figura 5.19: Estructura de "CODIGO_COM_AUTON"

Una Provincia pertenece a una Comunidad Autónoma. El campo "ID" (clave foránea de la tabla "CODIGO_PROVINCIA") permite identificar a la Comunidad Autónoma. El campo "COM_AUTON" contiene el nombre de la Comunidad y el campo "COD_COM_AUTON" es una clave de tres letras para facilitar las consultas.

5.4 DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO

En este apartado se detallan las consideraciones tenidas en cuenta para el diseño de la interfaz gráfica de usuario de la aplicación.

Distribución por pestañas

Se decidió emplear un sistema de pestañas para seleccionar los distintos tipos de distribución territorial sobre los que consultar. Como alternativa se puede usar un menú o desplegable, pero se eligió el sistema de pestañas por los siguientes motivos:

- **Espacio.** El listado de pestañas ocupa poco tamaño en relación a otros elementos de la interfaz.

- **Familiaridad.** El sistema de pestañas está muy extendido en las aplicaciones actuales, por lo que el usuario está familiarizado con él. Un ejemplo son la mayoría de los navegadores actuales, a través de los cuales se puede acceder a la aplicación.

- **Doble función.** Las pestañas cumplen una doble función: permiten al usuario elegir otra visualización y a la vez le permiten conocer en qué visualización se encuentra.

En la aplicación se utiliza el sistema de pestañas para seleccionar las distintas opciones de visualización principal:

- **Valladolid (sin división).** Consultas sobre el municipio de Valladolid (sin división territorial) y comparativas con Distrito y Zona Estadística.

- **División por Distrito.** Consultas sobre Distritos (y Secciones de los Distritos) y comparativas con otros Distritos.

- **División por Zona Estadística.** Consultas sobre Zonas Estadísticas y comparativas con otras Zonas.

- **Información – Ayuda.** Información sobre el funcionamiento de la aplicación y los parámetros con los que filtrar las consultas.

El sistema de pestañas se utiliza también dentro de las pestañas de "División por Distrito" y "División por Zona Estadística" para visualizar las opciones de "Información General" y "Consulta".

Mapas para la selección de Distritos y Zonas Estadísticas

Para seleccionar el Distrito o Zona Estadística se decidió utilizar mapas con las divisiones marcadas de diferentes colores. Su uso resulta adecuado por tratarse de una metáfora directa, y permite indicar conocer al usuario tanto las distintas divisiones territoriales como la división seleccionada (si la hay) y la división que se seleccionaría en caso de hacer clic sobre el mapa.

Para ello los mapas cuentan con un código de colores. Cada división (Distrito o Zona Estadística dependiendo de la pestaña seleccionada) está representada por un color. Además la división sobre la que se encuentra el ratón se muestra en un color especial, así como la división seleccionada.

Desplegables y botones

Las distintas opciones (como el tipo de distribución en la pestaña "Información General" de "División por Distrito" o "División por Zona Estadística", o los filtros aplicables a las consultas) se muestran como desplegables.

Su función puede ir más allá de seleccionar una opción aplicable en un momento posterior. El comportamiento de los desplegables de la interfaz distingue entre los siguientes casos:

- **La acción requiere la selección de un único desplegable.** Si la acción a ejecutar sólo requiere la utilización de un único desplegable (por ejemplo al seleccionar el Distrito con el que comparar un resultado) la propia selección de la opción en el desplegable desencadena la acción. Esto funciona así para simplificar las acciones a llevar a cabo por el usuario.

- **La acción requiere la selección de uno o más desplegables.** Si la acción a ejecutar requiere la utilización de uno o más desplegables (por ejemplo al seleccionar los parámetros de una consulta) la selección de la opción no desencadena la acción, que deberá solicitarse a través de un botón cuando el usuario haya concluido la selección de las opciones.

Por ejemplo, si el usuario quiere consultar información sobre el nivel de estudios de los varones entre 30 y 34 años, deberá solicitar la consulta tras seleccionar el tipo de distribución, el sexo y la edad. No resultaría conveniente actualizar la gráfica con cada selección. Consumiría recursos de la máquina en la que se ejecuta la aplicación y del servidor, lo que repercutiría directamente en el tiempo que el usuario emplearía en obtener el resultado de la consulta deseada.

Evitar los mensajes de error

Una buena interfaz gráfica debe evitar en medida de la posible lanzar mensajes de error al usuario. En su lugar, debe impedirle realizar acciones que fuesen a concluir en un error. Impedir al usuario llevar a cabo opciones no permitidas permite ahorrar tiempo al usuario y facilitar su comprensión de la aplicación.

En el diseño de la interfaz de la aplicación se ha seguido esta política, de forma que se desactivan los desplegables y botones en aquellos momentos en que falten condiciones para solicitar la acción.

Por ejemplo, a la hora de consultar información sobre un Distrito, las opciones del desplegable para elegir Sección se muestran en función del Distrito seleccionado. Además el desplegable se encuentra deshabilitado mientras no haya ningún Distrito seleccionado en el mapa.

Siguiendo con el ejemplo anterior, el botón "Consultar" estará deshabilitado hasta que el usuario no haya seleccionado los dos elementos necesarios para llevar a cabo la consulta, que son la selección del territorio en el mapa y el tipo de distribución.

Mantener estas opciones habilitadas constantemente supondría tener que lanzar un mensaje de error en caso de que el usuario seleccionase un número de Sección inexistente en el Distrito o si pulsase el botón Consultar sin haber seleccionado un tipo de distribución.

5.5 DIAGRAMA DE CLASES DE LA APLICACIÓN

Se decidió emplear el patrón de diseño en tres capas para el desarrollo de la aplicación. Este patrón permite dividir la funcionalidad en tres capas [3]:

- **Capa de Presentación.** Reúne los aspectos software relativos a la interfaz con la que interactuaran los usuarios (humanos). Incluye la disposición y el formato de los distintos elementos gráficos, como pueden ser los menús, pestañas, desplegables…

- **Capa de Dominio.** Recoge los componentes software que contienen las tareas y reglas que rigen el proceso. También se conoce como capa de Lógica o de Negocio.

- **Capa de Almacenamiento.** Contiene los componentes software que manejan los datos persistentes (típicamente bases de datos) a los que accede el sistema.

Es habitual que la clase que instancia a las demás sea el Controlador, ubicada en la capa de Dominio. En el Proyecto la clase que arranca e instancia a `Controlador` es `Ventana`. Esto es así porque un *applet* se lanza desde el método init(), que se encuentra en la propia clase `Applet` (o en este caso en la clase que la extiende). Además de lanzar el sistema, esta clase actúa como contenedor de los elementos de la interfaz, lo que la sitúa dentro de la capa de Presentación. Construir la clase `Controlador` extendiendo a `Applet` aunaría en una misma clase el controlador de Dominio y la clase principal de la Vista, lo que iría en contra de la filosofía de la

división en capas. A continuación se muestra en la Figura 5.20 el diagrama de clases que de la aplicación.

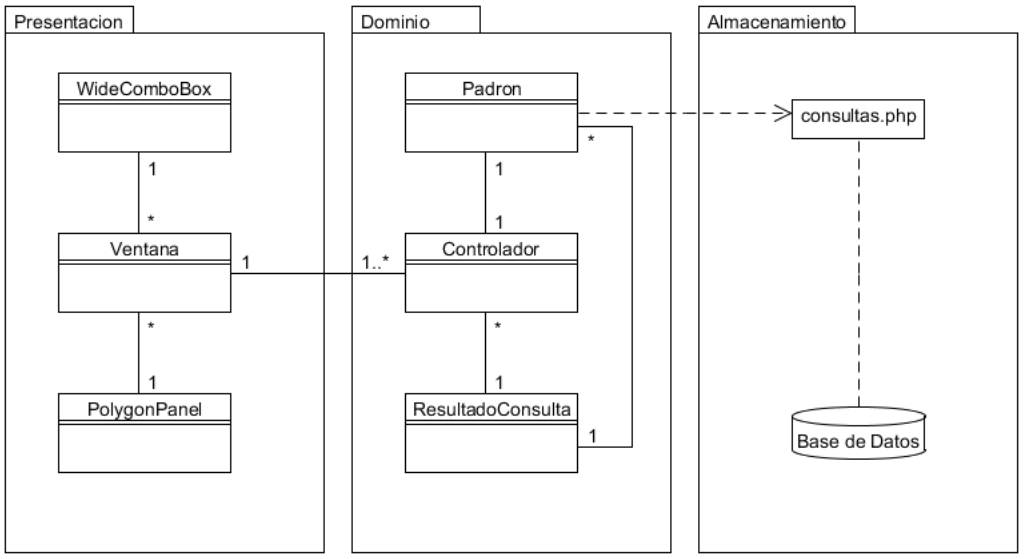

Figura 5.20: Diagrama de Clases de la aplicación

Con el fin de mejorar la legibilidad de la memoria, el detalle de los métodos y atributos de las clases, así como algunos comentarios sobre las mismas, se muestran de forma individual a continuación:

`Ventana`

La clase `Ventana` extiende la clase `JApplet`. Es la encargada de lanzar la aplicación, crear una instancia de `Controlador`, así como las distintas instancias de sus componentes y registrarlos en `Controlador` para que pueda recoger los eventos que se disparen dependiendo de las acciones del usuario.

En el diagrama se han resumido y simplificado los objetos de la clase `Ventana`. Al contener la interfaz gráfica, instancia una distintos desplegables (`JComboBox` y `WideComboBox`), etiquetas (`JLabel`), botones (`JButton`), pestañas (`JPanel`), contenedores de pestañas (`JTabbedPane`), paneles de gráficas (`ChartPanel`), paneles de polígonos (`PolygonPanel`) e imágenes de los mapas (`Image`).

Los métodos de `Ventana` están orientados al arranque de la aplicación, así como a atender las peticiones de `Controlador` dependiendo de las selecciones del usuario. De nuevo se han resumido y agrupado los que son similares.

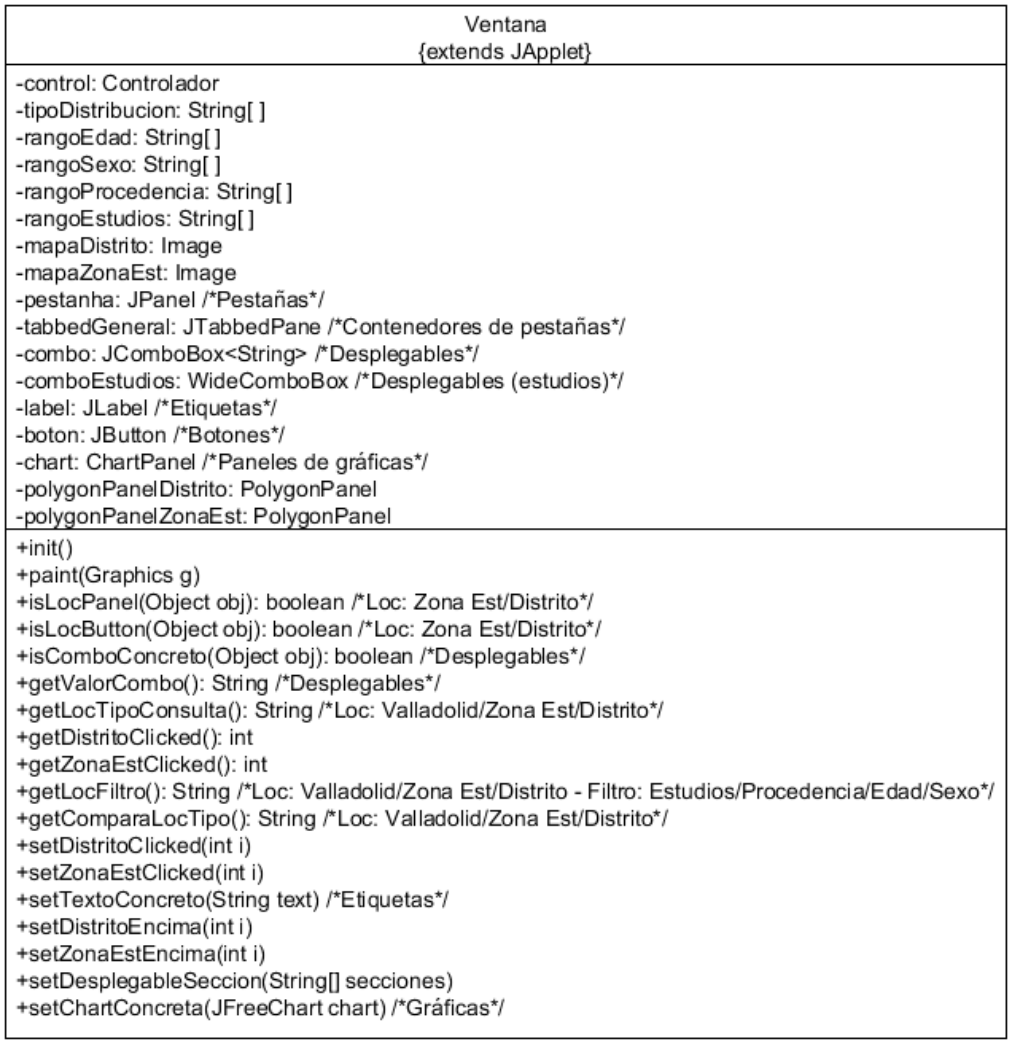

Figura 5.21: Clase `Ventana`

`PolygonPanel`

La clase `PolygonPanel` extiende la clase `JPanel`. Es la encargada representar los mapas de polígonos que el usuario podrá recorrer y seleccionar en la interfaz. Almacena un

conjunto de polígonos y de colores que asignará a cada polígono. Controla qué polígono ha sido pulsado y sobre qué polígono se encuentra el puntero del ratón a fin de colorearlo adecuadamente.

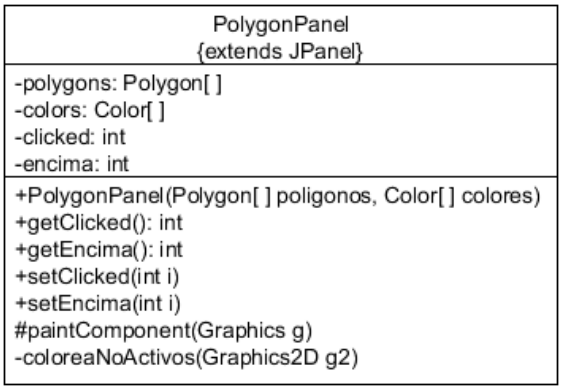

Figura 5.22: Clase `PolygonPanel`

`WideComboBox`

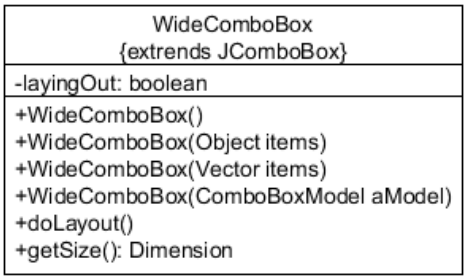

Figura 5.23: Clase `WideComboBox`

La clase WideComboBox fue obtenida de [23]. Extiende la clase JComboBox, y su uso surge de la necesidad de mostrar correctamente el contenido del desplegable relativo al nivel de estudios.

El problema con este desplegable radica en que la longitud de alguna de sus opciones es superior a la anchura del elemento. Como consecuencia, en un `JComboBox` las opciones que exceden el tamaño del componente se muestran cortadas. Para solucionar este problema se define la clase `WideComboBox`. En la Figura 5.24 se puede observar la diferencia entre el uso de un `JComboBox` y de un `WideComboBox` a la hora de mostrar opciones de longitud mayor a la anchura del desplegable.

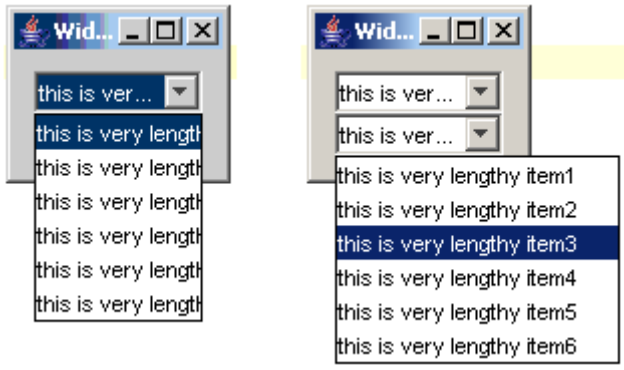

Figura 5.24: Comparativa `JComboBox` y `WideComboBox` [23]

`Controlador`

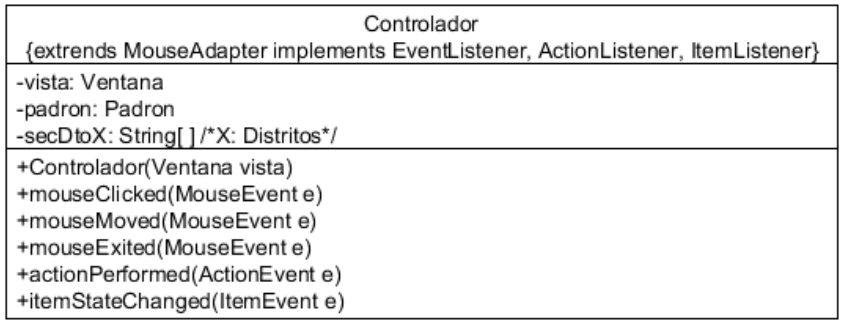

Figura 5.25: Clase `Controlador`

Controlador es la clase encargada de recibir los eventos que genere el usuario al manipular la interfaz. En función de los eventos recibidos, Controlador solicitará información a `Padron` y/o modificará los distintos elementos de `Vista`.

Extiende la clase `MouseAdapter` e implementa las interfaces `EventListener`, `ActionListener` e `ItemListener`.

`Padron`

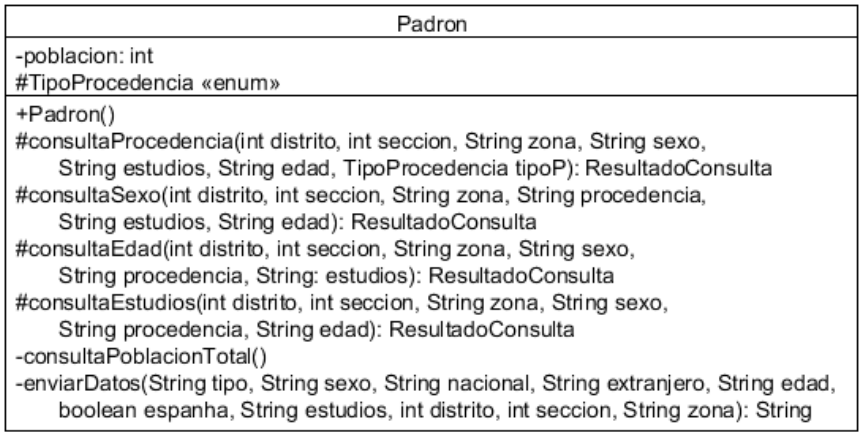

Figura 5.26: Clase `Padron`

`Padron` contiene los métodos necesarios para devolver resultados de consultas (mediante instancias de `ResultadoConsulta`) sobre el padrón municipal. A partir de los métodos de consulta (uno para cada tipo de distribución) se comunica con el script `consultas.php` alojado en el servidor y con la información recibida construye las gráficas y añade la información necesaria al resultado.

`ResultadoConsulta`

ResultadoConsulta representa el resultado de una consulta sobre el padrón municipal. Contiene una gráfica JFreeChart, el total de la población y el número de individuos y porcentaje de población representado en la gráfica.

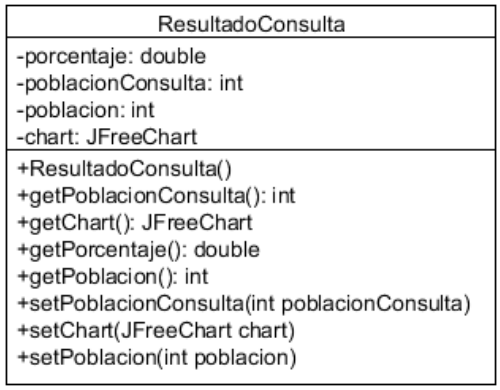

Figura 5.27: Clase `ResultadoConsulta`

5.6 INTERACCIÓN ENTRE LOS OBJETOS DE LA APLICACIÓN

A continuación se muestran los diagramas de secuencia de los casos de uso que se han considerado más relevantes.

5.6.1 CONSULTAR "VALLADOLID (SIN DIVISIÓN)"

Tras la solicitud del usuario de una consulta sobre Valladolid (sin atender a divisiones por distrito o zona estadística), la clase `Controlador` recibe el evento disparado por `Ventana`, confirma qué botón ha disparado el evento y solicita a `Ventana` el valor de los distintos parámetros de la consulta para enviárselos a `Padron`.

Estos parámetros han sido fijados previamente por el usuario a través de los desplegables (`JComboBox`) que alberga `Ventana`. No figuran en el diagrama de secuencia porque no provocan ninguna acción por parte de `Ventana` ni de ninguna otra clase desarrollada en el Proyecto. Es la clase `JComboBox` la que realiza las operaciones necesarias de forma transparente. Cuando Controlador solicita el valor de los parámetros a `Ventana`, esta transmite la petición a la instancia de `JComboBox` correspondiente.

`Padron` envía la información al script PHP y recibe el resultado de la misma. Crea una instancia de `ResultadoConsulta` y da valor a los atributos `chart`, `poblacionTotal` y `poblacionConsulta`. Finalmente devuelve la instancia de `ResultadoConsulta` a `Controlador`.

`Controlador` consulta los atributos de `ResultadoConsulta` y los asigna en `Ventana`. Por último, solicita a `Ventana` que restablezca las posibles comparativas realizadas a partir de una consulta previa sobre Valladolid y actualiza la interfaz.

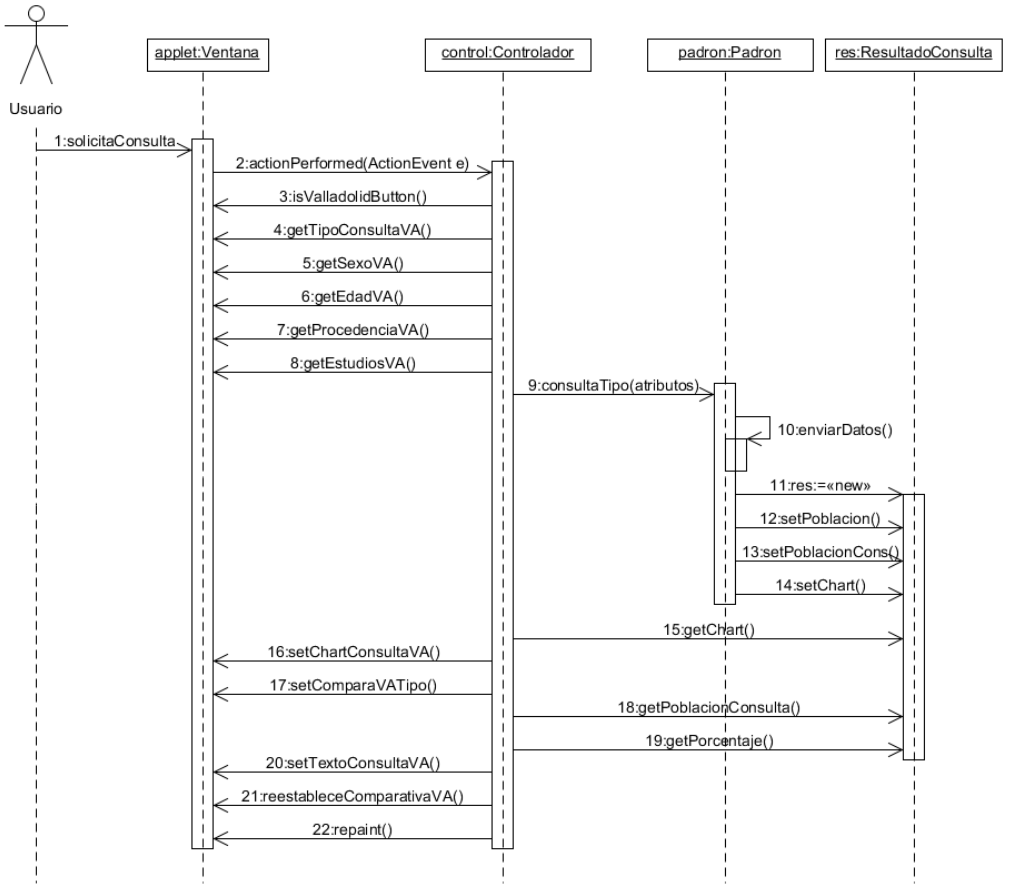

Figura 5.28: Diagrama de secuencia “Consultar Valladolid (sin división)”

5.6.2 CONSULTAR “DIVISIÓN POR DISTRITO”

Tras la solicitud del usuario de una consulta sobre un distrito de Valladolid, la clase `Controlador` recibe el evento disparado por `Ventana`, confirma qué botón ha disparado el evento y solicita a `Ventana` el valor de los distintos parámetros de la consulta para enviárselos a `Padron`.

Estos parámetros han sido fijados previamente por el usuario a través de los desplegables (`JComboBox`) que alberga `Ventana`. No figuran en el diagrama de secuencia porque no provocan ninguna acción por parte de `Ventana` ni de ninguna otra clase desarrollada en el Proyecto. Es la clase `JComboBox` la que realiza las operaciones necesarias de forma transparente. Cuando Controlador solicita el valor de los parámetros a `Ventana`, esta transmite la petición a la instancia de `JComboBox` correspondiente.

`Padron` envía la información al script PHP y recibe el resultado de la misma. Crea una instancia de `ResultadoConsulta` y da valor a los atributos `chart`, `poblacionTotal` y `poblacionConsulta`. Finalmente devuelve la instancia de `ResultadoConsulta` a `Controlador`.

`Controlador` consulta los atributos de `ResultadoConsulta`, los asigna en `Ventana`, y solicita la actualización de la interfaz.

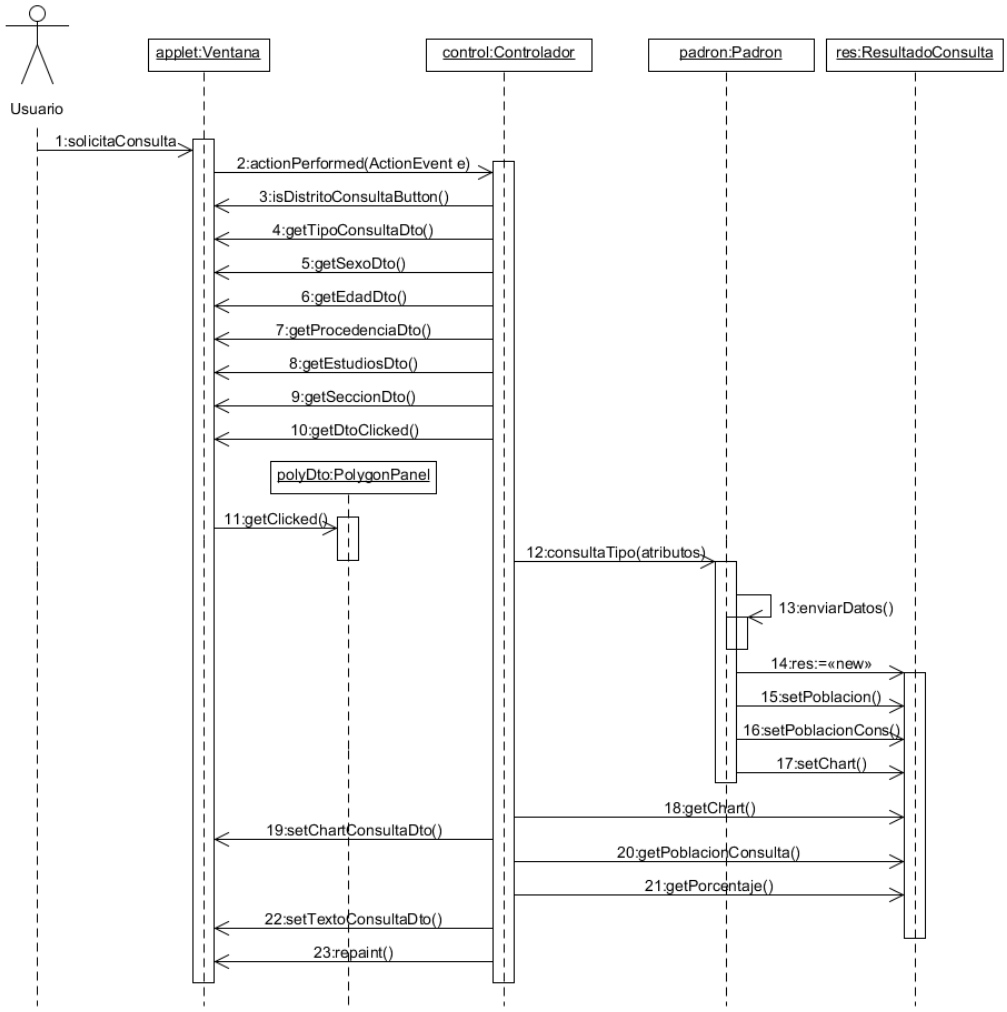

Figura 5.29: Diagrama de secuencia “Consultar división por Distrito”

5.6.3 CONSULTAR “DIVISIÓN POR ZONA ESTADÍSTICA”

La consulta sobre una zona estadística es similar a la consulta sobre un distrito, radicando las diferencias en los atributos consultados y configurados en `Ventana` y en la ausencia de secciones en la división por zona estadística.

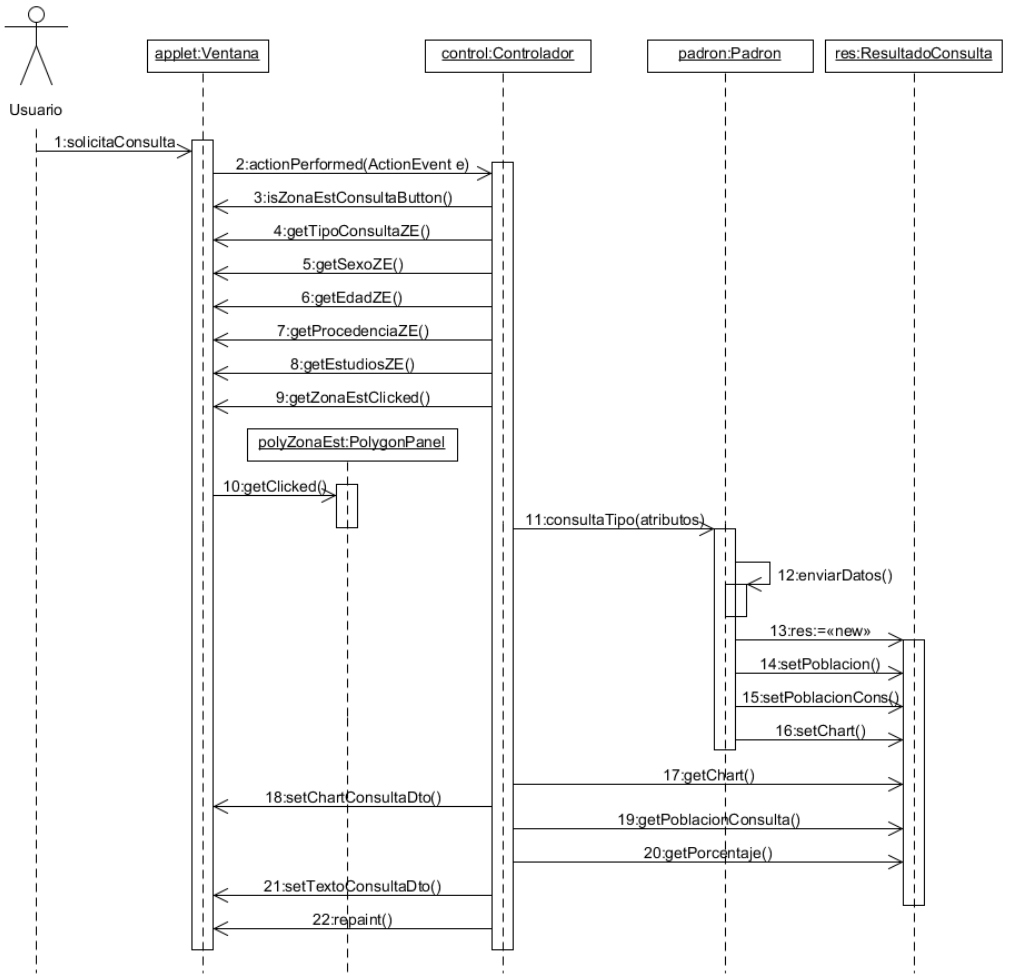

Figura 5.30: Diagrama de secuencia “Consultar división por Zona Estadística”

5.6.4 COMPARAR CONSULTA VALLADOLID CON DISTRITO

La extrapolación de una consulta realizada sobre Valladolid a un distrito sólo puede realizarse si la interfaz está mostrando en ese momento el resultado de una consulta. El resultado de

la comparativa se puede mostrar en dos posiciones (panel superior y panel inferior), pero la ejecución es idéntica en ambos casos salvo por el panel y etiqueta objetivos sobre los que se incluirá la información.

Tras la solicitud del usuario de una comparativa entre una consulta sobre Valladolid y el resultado de esa misma consulta sobre un distrito, la clase `Controlador` recibe el evento disparado por `Ventana`, confirma qué desplegable ha disparado el evento y solicita a `Ventana` el parámetro consultado sobre Valladolid.

`Padron` envía la información al script PHP y recibe el resultado de la misma. Crea una instancia de `ResultadoConsulta` y da valor a los atributos `chart`, `poblacionTotal` y `poblacionConsulta`. Finalmente devuelve la instancia de `ResultadoConsulta` a `Controlador`.

`Controlador` consulta los atributos de `ResultadoConsulta`, los asigna en `Ventana`, y solicita la actualización de la interfaz.

Las sentencias 3, 12 y 15 del diagrama de la Figura 5.31 se han definido como `isComboComparaPosDtoVA()`, `setChartPosVA()` y `setTextoPosVA()`. Si la solicitud se realiza en el desplegable superior, las sentencias serán `isComboComparaPosDtoVA()`, `setChartSupVA()` y `setTextoSupVA()`; siendo `isComboComparaPosDtoVA()`, `setChartInfVA()` y `setTextoInfVA()` en caso contrario.

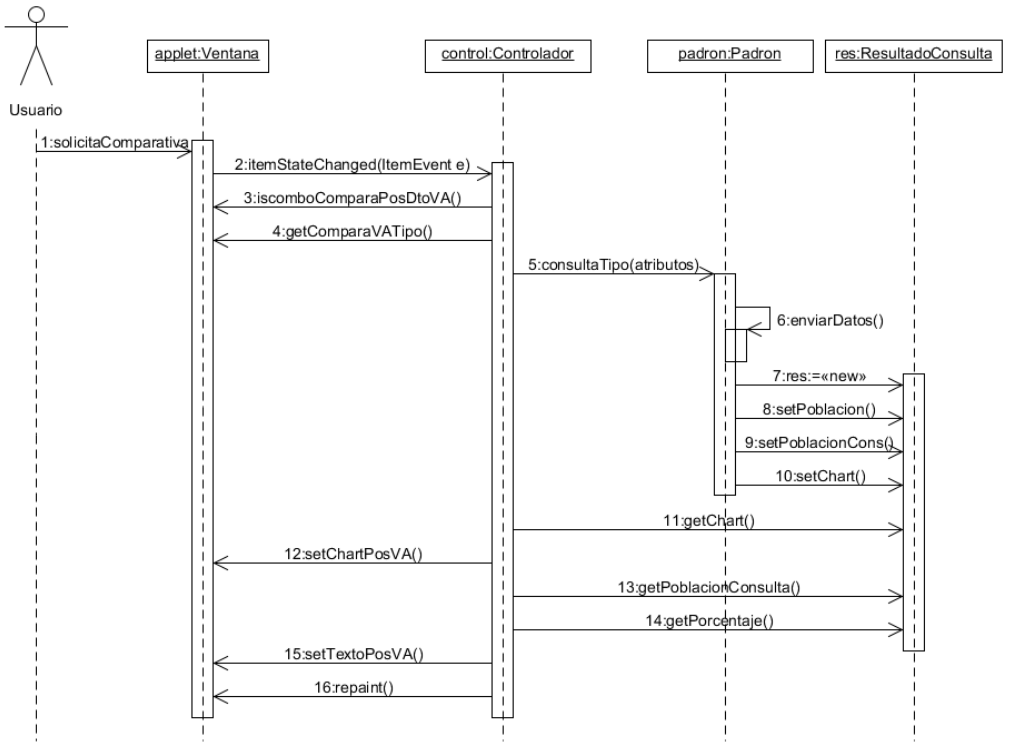

Figura 5.31: Diagrama de secuencia “Comparar consulta Valladolid con Distrito”

5.6.5 COMPARAR CONSULTA VALLADOLID CON ZONA ESTADÍSTICA

La comparación del resultado de una consulta sobre Valladolid con una zona estadística es idéntica a la comparación con un distrito, con la diferencia de que el desplegable que dispara la acción será diferente y contendrá un listado de las distintas zonas estadísticas. De esta forma, sobre la Figura 5.31, la sentencia 3 será (dependiendo del desplegable) `isComboComparaSupZEVA()` o `isComboComparaInfZEVA()`.

5.6.6 CONSULTAR INFORMACIÓN GENERAL DISTRITO

Tras la solicitud del usuario de la información general de un tipo de distribución en un Distrito, la clase `Controlador` recibe el evento disparado por `Ventana`, confirma qué desplegable ha disparado el evento y solicita a `Ventana` el tipo de distribución y el distrito seleccionado.

`Padron` envía la información al script PHP y recibe el resultado de la misma. Crea una instancia de `ResultadoConsulta` y da valor a los atributos `chart`, `poblacionTotal` y `poblacionConsulta`. Finalmente devuelve la instancia de `ResultadoConsulta` a `Controlador`.

`Controlador` consulta los atributos de `ResultadoConsulta` y los asigna en `Ventana`. Asigna también los valores de la distribución y distrito actuales, y elimina el gráfico y texto del apartado de la comparativa (ya que dejará de servir como comparativa con la consulta actual, que será diferente). Finalmente, solicita la actualización de la interfaz.

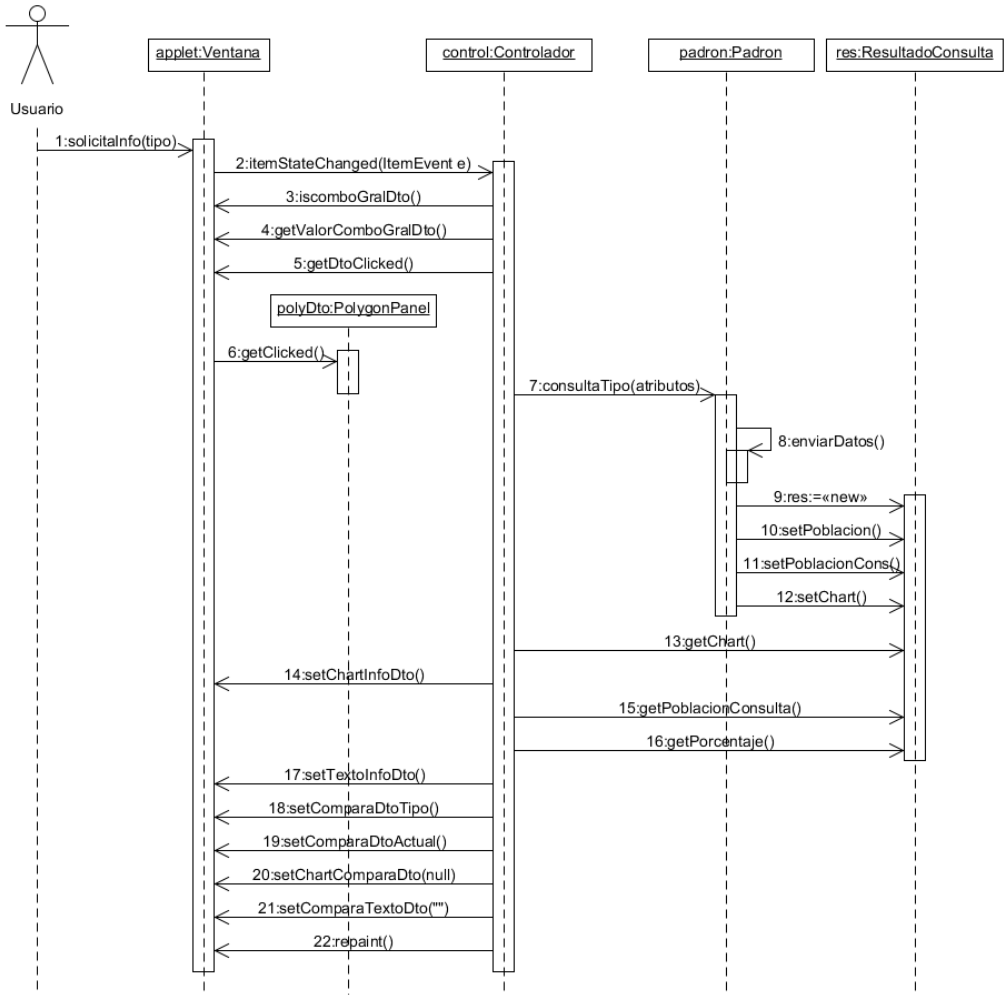

Figura 5.32: Diagrama de secuencia “Consultar información general Distrito”

5.6.7 COMPARAR INFORMACIÓN GENERAL DISTRITO

Tras la solicitud del usuario de la comparativa entre la información general de un tipo de distribución en un Distrito con otro, la clase `Controlador` recibe el evento disparado por `Ventana`, confirma qué desplegable ha disparado el evento y solicita a `Ventana` el tipo de distribución y el distrito seleccionado.

`Padron` envía la información al script PHP y recibe el resultado de la misma. Crea una instancia de `ResultadoConsulta` y da valor a los atributos `chart`, `poblacionTotal` y `poblacionConsulta`. Finalmente devuelve la instancia de `ResultadoConsulta` a `Controlador`.

`Controlador` consulta los atributos de `ResultadoConsulta`, los asigna en `Ventana` y solicita la actualización de la interfaz.

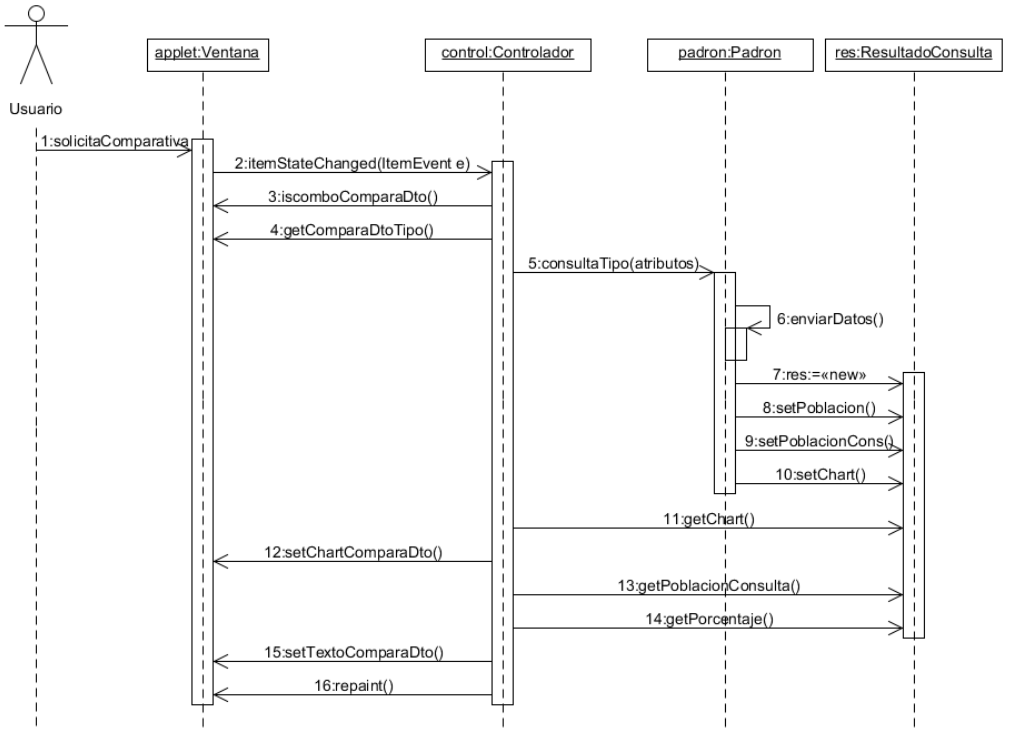

Figura 5.33: Diagrama de secuencia “Comparar información general Distrito”

5.6.8 CONSULTAR INFORMACIÓN GENERAL ZONA ESTADÍSTICA

Tras la solicitud del usuario de la información general de un tipo de distribución en una Zona Estadística, la clase `Controlador` recibe el evento disparado por `Ventana`, confirma qué desplegable ha disparado el evento y solicita a `Ventana` el tipo de distribución y el distrito seleccionado.

`Padron` envía la información al script PHP y recibe el resultado de la misma. Crea una instancia de `ResultadoConsulta` y da valor a los atributos `chart`, `poblacionTotal` y `poblacionConsulta`. Finalmente devuelve la instancia de `ResultadoConsulta` a `Controlador`.

`Controlador` consulta los atributos de `ResultadoConsulta` y los asigna en `Ventana`. Asigna también los valores de la distribución y zona estadística actuales, y elimina el gráfico y texto del apartado de la comparativa (ya que dejará de servir como comparativa con la consulta actual, que será diferente). Finalmente, solicita la actualización de la interfaz.

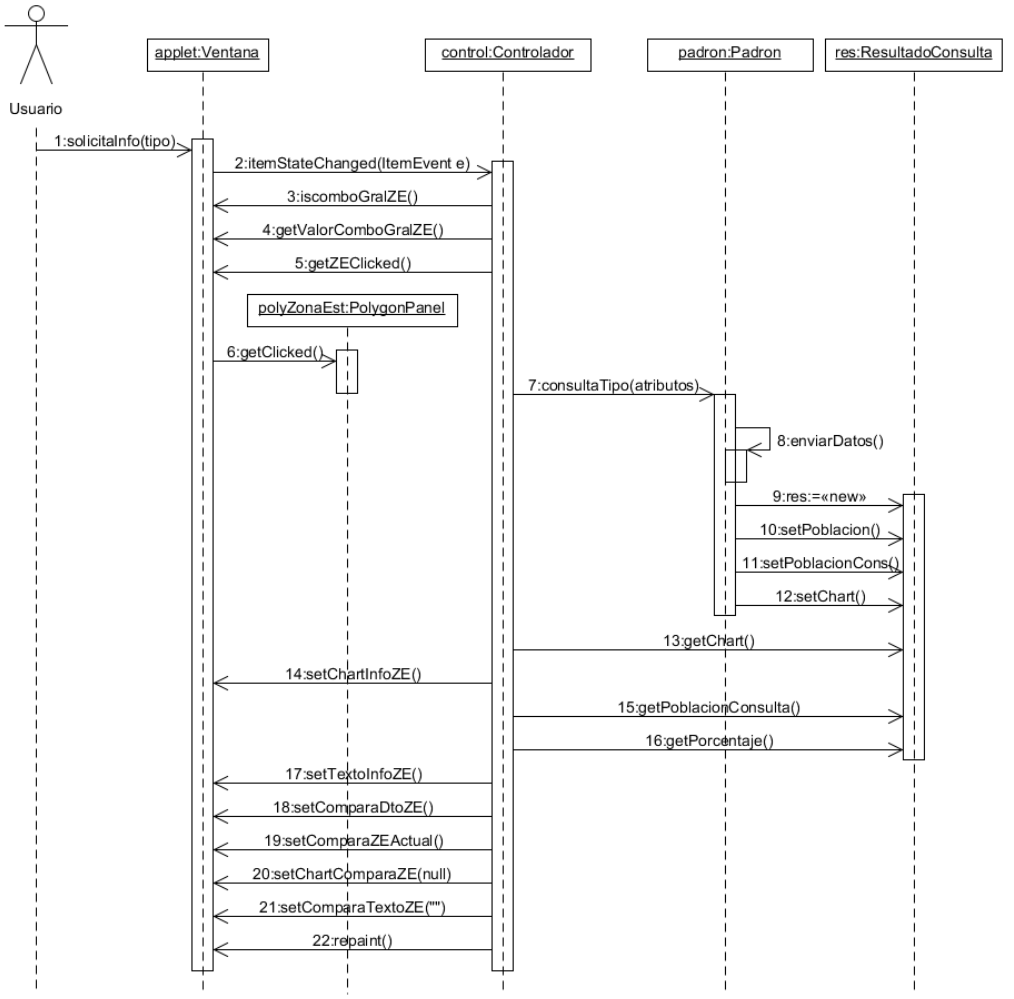

Figura 5.34: Diagrama de secuencia “Consultar información general Zona Estadística”

5.6.7 COMPARAR INFORMACIÓN GENERAL ZONA ESTADÍSTICA

Tras la solicitud del usuario de la comparativa entre la información general de un tipo de distribución en una Zona Estadística con otra, la clase `Controlador` recibe el evento disparado por `Ventana`, confirma qué desplegable ha disparado el evento y solicita a `Ventana` el tipo de distribución y el distrito seleccionado.

`Padron` envía la información al script PHP y recibe el resultado de la misma. Crea una instancia de `ResultadoConsulta` y da valor a los atributos `chart`, `poblacionTotal` y `poblacionConsulta`. Finalmente devuelve la instancia de `ResultadoConsulta` a `Controlador`.

`Controlador` consulta los atributos de `ResultadoConsulta`, los asigna en `Ventana` y solicita la actualización de la interfaz.

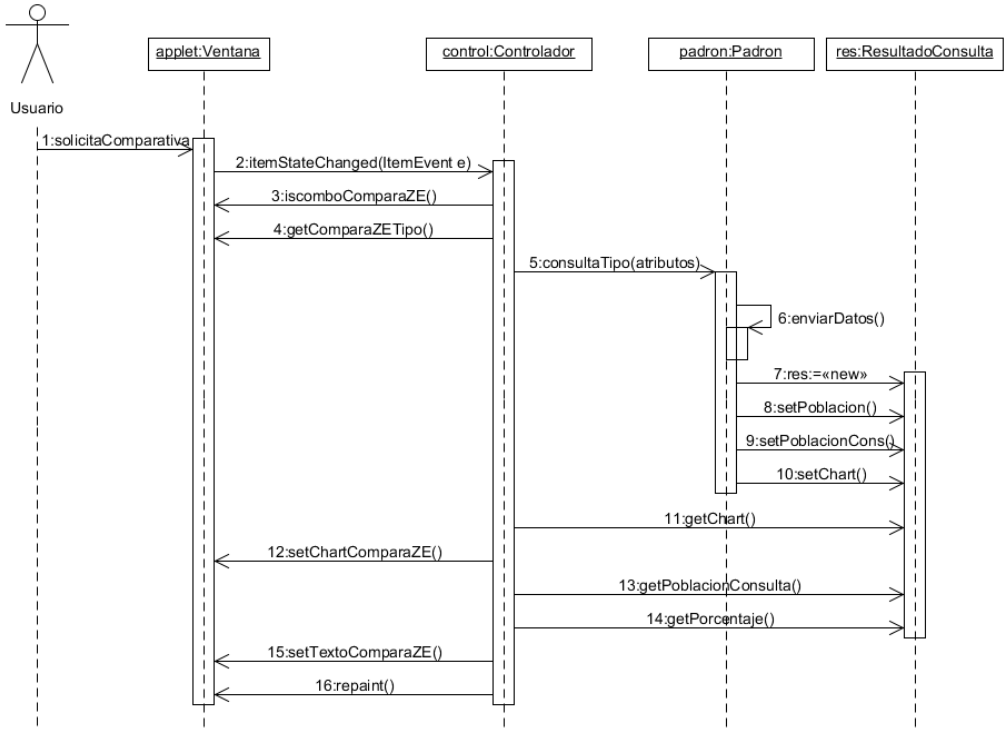

Figura 5.35: Diagrama de secuencia “Comparar información general Distrito”

VI. IMPLEMENTACIÓN

6.1 INTRODUCCIÓN

En el presente capítulo se describirán los detalles de la implementación de la aplicación, haciendo especial hincapié en la comunicación con la base de datos (a través de un script desarrollado en PHP) y en las consultas SQL que se llevarán a cabo para extraer la información y generar las gráficas.

6.2 COMUNICACIÓN CON LA BASE DE DATOS

Como se ha comentado anteriormente, la idea inicial del Proyecto era acceder a los datos sobre el padrón municipal a través de una aplicación externa. Debido al retraso del desarrollo paralelo de dicha aplicación, se optó por obtener la información de una base de datos relacional, suministrada por el Ayuntamiento.

En los capítulos de Análisis y Diseño se ha podido ver el estudio de la base de datos proporcionada. Para alojar la base de datos se utilizó un servidor MySQL proporcionado por el

Grupo de Investigación GRINBD del Departamento de Informática de la Universidad de Valladolid. En este mismo servidor se aloja la web y la aplicación.

El servidor MySQL presenta una limitación: sólo acepta conexiones locales (llevadas a cabo desde el propio servidor). Java posee APIs para realizar conexiones entre sus aplicaciones y los distintos Sistemas Gestores de Bases de Datos. Sin embargo las *Applet* Java se ejecutan en el cliente, por lo que resulta imposible realizar la conexión con el servidor facilitado.

Para subsanar este problema se empleó un script PHP alojado en el servidor del SGBD (de forma que la conexión fuese local) y con el que el *applet* se podría comunicar. El acceso al contenido de la base de datos se llevaría a cabo tal y como muestra la Figura 6.1.

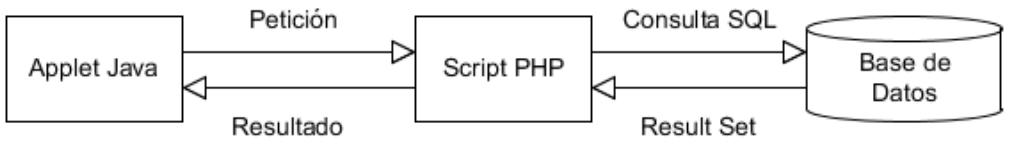

Figura 6.1: Comunicación *Applet* – PHP – Base de Datos

6.2.1 COMUNICACIÓN *APPLET* – PHP

Es posible establecer una conexión entre un *applet* y una página PHP, enviando datos en forma *POST* y recibiendo la salida del script en forma de cadena.

Para ello se utilizan los objetos de las clases Java `URLEncoder`, `URL` y `URLConnection`. La primera codifica una URL (*Uniform Resource Locator*, Localizador Uniforme de Recursos) en un formato adecuado, la segunda define la URL y la tercera establece la conexión con el recurso definido en la URL. Tras establecer la conexión, se pueden enviar (vía *POST*) y recibir datos de la página PHP suministrada como URL [49].

El código que realiza este proceso puede observarse en la Figura 6.2. En primer lugar se proporciona el formato de codificación de la URL (que será UTF-8), se instancia la URL y se establece la conexión, configurando el envío de de solicitudes vía *POST*. A continuación se envía la cadena con los datos de la consulta.

En este momento el script PHP realizará las operaciones pertinentes y devolverá el resultado como una página HTML (cadena de texto). Esta cadena tendrá un formato determinado que el *applet* podrá interpretar para obtener el resultado de la consulta y crear la gráfica solicitada.

```
URLEncoder.encode(dataString, "UTF-8");
URL url = new URL(phpURL);
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter wr = new
      OutputStreamWriter(conn.getOutputStream());
wr.write(dataString);
wr.flush();
wr.close();
// El script PHP procesa la información
BufferedReader rd = new BufferedReader(new
      InputStreamReader(conn.getInputStream()));
String linea, resultado = "";
while (( linea = rd.readLine()) != null)
      resultado+=linea;
rd.close();
```

Figura 6.2: Conexión *Applet* - PHP

6.2.2 COMUNICACIÓN PHP – BASE DE DATOS

Para establecer la conexión entre PHP y el servidor MySQL se utiliza la extensión de PHP PDO (*PHP Data Objets*, Objetos de Datos PHP) [48]. Otras posibilidades para realizar la conexión y comunicación entre PHP y MySQL son las extensiones Mysql [34] y Mysqli [35]. Se optó sin embargo por PDO por presentar mejoras sustanciales frente a la API Mysql (como por ejemplo el uso de sentencias preparadas) y debido a la no disponibilidad de Mysqli en el servidor facilitado para alojar el script PHP.

La conexión con el Sistema Gestor de Bases de Datos MySQL puede observarse en la Figura 6.3. Para realizarla basta con facilitar el servidor, nombre de la base de datos, usuario y contraseña; así como controlar si la conexión se ha establecido correctamente antes de continuar con la ejecución del script.

```
$conn = new PDO(
      "mysql:host=".DBHOST.";dbname=".DBNAME,
      DBUSER,
      DBPASS);
if (!$conn) {
      echo "Conexión no establecida";
      die();
}
```

Figura 6.3: Conexión PHP – MySQL

A continuación se construye la sentencia SQL que contendrá la consulta requerida sobre la base de datos. Esta consulta se construirá dependiendo de las variables recibidas en el *POST*, con el fin de simplificar la consulta en medida de lo posible y obtener una respuesta rápida del sistema.

Se pueden agrupar las consultas conforme a tres tipos: consultas sobre el total de la población, consulta sobre la edad y consulta de otro tipo. Las Figuras 6.4, 6.5 y 6.6 muestran ejemplos de estos tres tipos (en el script se utilizan sentencias preparadas, pero se muestran las consultas completas para facilitar su lectura).

La agrupación se debe al número de columnas que devuelven las consultas y deben ser enviadas al *applet*. La consulta sobre el total de la población (Figura 6.4) consta de un único campo (total), la consulta sobre la edad (Figura 6.5) consta de tres (edad, sexo y total) y el resto de consultas (Figura 6.6), de dos (atributo consultado y total).

```
SELECT COUNT(*) AS TOTAL
FROM PADRON_HABITANTES
```

Figura 6.4: Ejemplo de consulta el total de la población

```
SELECT PAD.EDAD AS RESULTADO, SEX.SEXO AS SEXO,
      COUNT(*) AS TOTAL
FROM (((PADRON_HABITANTES PAD
      INNER JOIN CODIGO_SEXO SEX ON PAD.ID_SEXO = SEX.ID)
      INNER JOIN (CODIGO_PROVINCIA PRV
            INNER JOIN CODIGO_COM_AUTON CAU
                  ON PRV.ID_COM_AUTON = CAU.ID)
            ON PAD.ID_PROVINCIA_NAC = PRV.ID)
      INNER JOIN CODIGO_ESTUDIOS EST
            ON PAD.ID_ESTUDIOS = EST.ID)
WHERE PAD.DISTRITO = 8
AND PAD.SECCION = 2
AND EST.CODIGO_INE ='UNIVE'
AND CAU.COD_COM_AUTON = 'CYL'
GROUP BY RESULTADO, SEXO
```

Figura 6.5: Ejemplo de consulta sobre la edad

```
SELECT EST.ESTUDIOS_INE AS RESULTADO, COUNT(*) AS TOTAL
FROM (((PADRON_HABITANTES PAD
      INNER JOIN CODIGO_ESTUDIOS EST
            ON PAD.ID_ESTUDIOS = EST.ID)
      INNER JOIN CODIGO_SEXO SEX ON PAD.ID_SEXO = SEX.ID)
      INNER JOIN (CODIGO_PROVINCIA PRV
            INNER JOIN CODIGO_COM_AUTON CAU
                  ON PRV.ID_COM_AUTON = CAU.ID)
            ON PAD.ID_PROVINCIA_NAC = PRV.ID)
WHERE PAD.DISTRITO = 8
AND PAD.SECCION = 2
AND PAD.EDAD BETWEEN 15 AND 19
AND SEX.SEXO = 'varon'
AND CAU.COD_COM_AUTON = 'CYL'
GROUP BY RESULTADO
```

Figura 6.6: Ejemplo de consulta sobre el nivel de estudios

Tras construir la consulta en función de la información solicitada, y almacenar los valores de los distintos campos, se procede a preparar y ejecutar la sentencia sobre la conexión establecida previamente. Este proceso puede verse en la Figura 6.7

```
$stmt = $conn->prepare($sql);
$stmt->execute($vars);
```

Figura 6.7: Ejecución de la consulta

En este momento el SGBD ejecutará la consulta y devolverá el resultado de la misma en un conjunto de columnas. A partir de esta salida, el script elaborará una cadena de texto que mostrará por pantalla y será recibida por el *applet*, que tras el envío de los parámetros queda a la espera de una respuesta.

Para codificar el resultado en una cadena de caracteres se determinó que el carácter '#' serviría para separar un campo de otro, y el carácter '@' para indicar el fin de una fila (ver Figura 6.8). En la Figura 6.9 se puede ver la salida de la consulta de la Figura 6.6 codificada de esta forma.

```
switch ($tipoConsulta) {
case "count":
      while($row = $stmt->fetch(PDO::FETCH_ASSOC)){
            echo $row['TOTAL']."@";
      }
      break;
case "edad":
      while($row = $stmt->fetch(PDO::FETCH_ASSOC)){
            echo $row['RESULTADO']."#".$row['SEXO']."#".
                  $row['TOTAL']."@";
      }
      break;
default:
      while($row = $stmt->fetch(PDO::FETCH_ASSOC)){
            echo $row['RESULTADO']."#".$row['TOTAL']."@";
      }
      break;
}
```

Figura 6.8: Construcción de la salida

```
ESTUDIOS PRIMARIOS#3@SIN ESTUDIOS#16@
```

Figura 6.9: Resultado de la consulta de ejemplo sobre el nivel de estudios

VII. PRUEBAS

7.1 INTRODUCCIÓN

Tras el desarrollo de la aplicación, es necesario probar que el funcionamiento obtenido corresponde con el deseado. Para ello se deben diseñar y aplicar una serie de pruebas, que permitan localizar los posibles fallos para proceder a su corrección. Las pruebas aplicadas sobre este trabajo están orientadas a detectar fallos en cada método ofrecido al cliente por la API y en el cumplimiento de los requisitos especificados.

Se han utilizado pruebas de caja negra de análisis de valores límite y conjetura de errores. Para llevar un orden a la hora de hacer las pruebas de cada método, se han clasificado en pruebas sobre la interfaz gráfica (actualizaciones, respuesta a eventos…) y pruebas sobre las consultas. Cada prueba consta de un identificador y un título, una descripción, la salida esperada, la salida observada y en ocasiones comentarios adicionales.

Para la realización de las siguientes pruebas se parte de una instalación completa y correcta de los componentes del sistema.

El capítulo actual contiene una selección de las pruebas realizadas sobre el software desarrollado. El resto de pruebas puede consultarse en el CD del Trabajo Fin de Grado.

7.2 PRUEBAS SOBRE LA INTERFAZ

Las pruebas sobre la interfaz se han dividido en "Pruebas Generales" y en función de la pestaña seleccionada: "Valladolid (sin división)", "División por Distrito" y "División por Zona Estadística".

7.2.1 PRUEBAS GENERALES

Título	**[CP-1] Selección de pestaña "Valladolid (sin división)"**
Descripción	Seleccionar la pestaña "Valladolid (sin división)".
Entrada	Clic en la pestaña "Valladolid (sin división)".
Resultado esperado	Se muestra el contenido de la pestaña "Valladolid (sin división)".
Resultado obtenido	Resultado esperado.

Figura 7.1: [CP-1] Selección de pestaña "Valladolid (sin división)"

Se han llevado a cabo pruebas similares sobre el resto de pestañas principales y sub-pestañas.

Título	**[CP-5] Aumentar zoom sobre gráfica de barras**
Descripción	Aumentar el zoom sobre una gráfica de barras.
Entrada	Clic y arrastrar el ratón hacia abajo en una gráfica de barras.
Resultado esperado	El zoom aumenta mostrando la zona seleccionada en el arrastre.
Resultado obtenido	Resultado esperado.

Figura 7.2: [CP-5] Aumentar zoom sobre gráfica de barras

Título	**[CP-6] Eliminar zoom sobre gráfica de barras**
Descripción	Eliminar el zoom de una gráfica de barras.
Entrada	Clic y arrastrar el ratón hacia arriba en una gráfica de barras.
Resultado esperado	Se muestra la gráfica original.
Resultado obtenido	Resultado esperado.

Figura 7.3: [CP-6] Eliminar zoom sobre gráfica de barras

Título	**[CP-9] Mostrar detalle de gráfica**
Descripción	Mostrar más información sobre un elemento de una gráfica.
Entrada	Posicionar el ratón sobre un elemento de una gráfica.
Resultado esperado	Se muestra información adicional sobre el elemento elegido.
Resultado obtenido	Resultado esperado.
Comentarios	Esta prueba se realizó para cada tipo de gráfica: sectores, barras y pirámide de población.

Figura 7.4: [CP-9] Mostrar detalle de gráfica

7.2.2 PRUEBAS SOBRE LA PESTAÑA "Valladolid (sin división)"

Título	**[CP-10] Selección de distribución "Sexo" [Valladolid (sin división)]**
Descripción	Seleccionar la distribución "Sexo" en la pestaña "Valladolid (sin división)".
Entrada	Opción "Sexo" en el desplegable "Datos a representar:".
Resultado esperado	Se habilita el botón "Consultar" y se deshabilita el desplegable "Sexo".
Resultado obtenido	Resultado esperado.

Figura 7.5: [CP-10] Selección de distribución "Sexo" [Valladolid (sin división)]

Se han llevado a cabo pruebas similares sobre el resto de opciones de los desplegables que habilitan y deshabilitan otros elementos de la interfaz, tanto en la pestaña "Valladolid (sin división)" como en las pestañas "División por Distrito" y "División por Zona Estadística".

Título	**[CP-16] Clic en "Consultar" [Valladolid (sin división)]**
Descripción	Realizar una consulta haciendo clic en el botón "Consultar" de la pestaña "Valladolid (sin división)".
Entrada	Clic en "Consultar".
Resultado esperado	Se muestra el resultado de la consulta en forma de gráfica y texto; se eliminan las gráficas de comparativas previas (si las hay), se habilitan los desplegables para realizar comparativas.
Resultado obtenido	Resultado esperado.

Figura 7.6: [CP-16] Clic en "Consultar" [Valladolid (sin división)]

Título	**[CP-17] Comparativa con Distrito o Zona Estadística [Valladolid (sin división)]**
Descripción	Comparar una consulta realizada sobre Valladolid (sin división) con la aplicación de los mismos parámetros sobre un Distrito o Zona Estadística.
Entrada	Selección de Distrito o Zona Estadística en uno de los desplegables.
Resultado esperado	Se muestra el resultado de la comparativa en forma de gráfica y texto en la zona correspondiente de la interfaz.
Resultado obtenido	Resultado esperado.
Comentarios	Esta prueba se realizó con cuatro variantes: Distrito y Zona Estadística en parte superior e inferior.

Figura 7.7: [CP-17] Comparativa con Distrito o Zona Estadística [Valladolid (sin división)]

7.2.3 PRUEBAS SOBRE LA PESTAÑA "División por Distrito"

Título	**[CP-18] Navegación sobre el mapa [División por Distrito]**
Descripción	Navegar sobre el mapa de la pestaña "División por Distrito".
Entrada	Desplazamiento del ratón sobre los elementos del mapa.
Resultado esperado	El elemento sobre el que se encuentra el ratón se pone de color verde y en la parte superior aparece una descripción del distrito. Si el ratón se posiciona sobre una zona sin elementos, se muestra el mensaje "Seleccione un Distrito" en la parte superior.
Resultado obtenido	Resultado esperado.

Figura 7.8: [CP-18] Navegación sobre el mapa [División por Distrito]

Título	**[CP-19] Selección de Distrito [División por Distrito]**
Descripción	Seleccionar un Distrito en el mapa.
Entrada	Clic sobre un Distrito en el mapa.
Resultado esperado	El Distrito se pone de color rojo. En la parte superior queda fija la descripción del Distrito mientras el ratón no se sitúe sobre otro Distrito. Se habilita el desplegable "Distribución:" de la pestaña "Información General". Se habilita el desplegable "Sección" de la pestaña "Consulta", mostrando las Secciones propias del Distrito.
Resultado obtenido	Se detectó que sacando el puntero del ratón del mapa desde un Distrito no seleccionado pegado al borde del mismo, y teniendo otro Distrito seleccionado, la descripción de la parte superior queda fija en la del último Distrito en el que se situó el ratón en lugar de mostrar la del seleccionado.

Figura 7.9: [CP-19] Selección de Distrito [División por Distrito]

Diagnóstico y corrección

El problema radica en los eventos disparados desde Ventana y capturados en Controlador. Había un problema en el control de la salida del ratón del mapa en el caso de que un Distrito estuviese seleccionado.

Se corrigió implementando una modificación en el código para subsanar el problema.

Título	**[CP-21] Información General de Distrito [División por Distrito]**
Descripción	Obtener la información general de un tipo de distribución para un Distrito.
Entrada	Seleccionar un tipo de distribución en el desplegable “Distribución:”.
Resultado esperado	Se muestra el resultado de la consulta en forma de gráfica y texto en el espacio habilitado para ello. Se habilita el desplegable “Comparar con:” con un Distrito por opción excluyendo el Distrito seleccionado.
Resultado obtenido	Resultado esperado.

Figura 7.10: [CP-21] Información General de Distrito [División por Distrito]

7.2.4 PRUEBAS SOBRE LA PESTAÑA “División por Zona Estadística”

Las pruebas ejecutadas sobre la pestaña “División por Zona Estadística” son similares a las realizadas sobre “División por Distrito”, pero aplicadas sobre la pestaña correspondiente.

7.3 PRUEBAS SOBRE LAS CONSULTAS

Las pruebas sobre las consultas se han dividido en “Pruebas Generales” y en función de la pestaña seleccionada: “Valladolid (sin división)”, “División por Distrito” y “División por Zona Estadística”.

Su funcionamiento se ha comprobado realizando la misma consulta a través de la aplicación y directamente sobre la base de datos empleando SQL.

7.3.1 PRUEBAS GENERALES

Título	**[CP-42] Consulta vacía**
Descripción	Realizar una consulta que no devuelva resultados, en la que ningún miembro del Padrón cumpla los parámetros establecidos para ella.
Entrada	Consulta con parámetros no cumplidos por ningún elemento de la población.
Resultado esperado	Se muestra el resultado de la consulta en forma de gráfica (vacía) y texto.
Resultado obtenido	Resultado esperado.

Figura 7.11: [CP-42] Consulta vacía

7.3.2 PRUEBAS SOBRE LA PESTAÑA "Valladolid (sin división)"

Título	**[CP-43] Consulta sobre "Sexo" sin parametrizar [Valladolid (sin división)]**
Descripción	Consulta sobre "Sexo" sin añadir parámetros adicionales.
Entrada	Clic en "Consultar" con la opción "Sexo" seleccionada en el desplegable "Datos a representar:" y la opción "Todos" seleccionada en el resto de desplegables.
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.

Figura 7.12: [CP-43] Consulta sobre "Sexo" sin parametrizar [Valladolid (sin división)]

Título	**[CP-44] Consulta sobre "Sexo" parametrizada por "Procedencia" [Valladolid (sin división)]**
Descripción	Consulta sobre "Sexo" añadiendo una "Procedencia" como parámetro.
Entrada	Clic en "Consultar" con la opción "Sexo" seleccionada en el desplegable "Datos a representar:", una opción distinta de "Todos" en el desplegable "Procedencia" y la opción "Todos" seleccionada en el resto de desplegables.
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.

Figura 7.13: [CP-44] Consulta sobre "Sexo" parametrizada por "Procedencia" [Valladolid (sin división)]

Título	**[CP-47] Consulta sobre "Sexo" parametrizada por varios parámetros [Valladolid (sin división)]**
Descripción	Consulta sobre "Sexo" añadiendo varios parámetros.
Entrada	Clic en "Consultar" con la opción "Sexo" seleccionada en el desplegable "Datos a representar:", y una opción distinta de "Todos" en dos o más desplegables.
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.
Comentarios	Esta prueba se realizó con distintas combinaciones de dos y tres parámetros, obteniendo en todos los casos el mismo resultado.

Figura 7.14: [CP-47] Consulta sobre "Sexo" parametrizada por varios parámetros [Valladolid (sin división)]

Se han llevado a cabo pruebas similares sobre cada tipo de distribución (Sexo, Procedencia, Estudios y Edad) parametrizada por todos los filtros posibles (Sexo, Procedencia (Nac.), Procedencia (Extr.), Estudios y Edad); así como por combinaciones de estos.

Título	**[CP-68] Comparativa de consulta sobre "Sexo" con un Distrito [Valladolid (sin división)]**
Descripción	Extrapolación de una consulta realizada sobre "Sexo" a un Distrito.
Entrada	Selección de un Distrito en uno de los desplegables.
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.
Comentarios	Esta prueba se realizó sobre el resultado de las consultas realizadas en las pruebas [CP-43] a [CP-47] con el fin de cubrir el mayor número de casos posible.

Figura 7.15: [CP-68] Comparativa de consulta sobre "Sexo" con un Distrito [Valladolid (sin división)]

Título	**[CP-73] Comparativa de consulta sobre "Sexo" con una Zona Estadística [Valladolid (sin división)]**
Descripción	Extrapolación de una consulta realizada sobre "Sexo" a una Zona Estadística.
Entrada	Selección de una Zona Estadística en uno de los desplegables.
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.
Comentarios	Esta prueba se realizó sobre el resultado de las consultas realizadas en las pruebas [CP-43] a [CP-47] con el fin de cubrir el mayor número de casos posible.

Figura 7.16: [CP-73] Comparativa de consulta sobre "Sexo" con una Zona Estadística [Valladolid (sin división)]

Se han llevado a cabo pruebas similares sobre cada prueba realizada sobre los distintos tipos de distribución (con todas las opciones de parámetros), realizando la comparación tanto con Distritos como con Zonas Estadísticas.

7.3.3 PRUEBAS SOBRE LA PESTAÑA "División por Distrito"

Título	**[CP-78] Información General "Sexo" [División por Distrito]**
Descripción	Obtención de Información General sobre "Sexo" para un Distrito.
Entrada	Selección de la distribución "Sexo" en la pestaña "Información General".
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.

Figura 7.17: [CP-78] Información General "Sexo" [División por Distrito]

Se han llevado a cabo pruebas similares sobre la Información General de cada tipo de distribución.

Título	**[CP-83] Comparativa de Información General con otro Distrito [División por Distrito]**
Descripción	Extrapolación de una consulta realizada sobre "Estudios" a una Zona Estadística.
Entrada	Selección de una Zona Estadística en uno de los desplegables.
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.
Comentarios	Esta prueba se realizó sobre el resultado de las consultas realizadas en las pruebas [CP-78] a [CP-82].

Figura 7.18: [CP-83] Comparativa de Información General con otro Distrito [División por Distrito]

Título	**[CP-84] Consulta sobre "Sexo" sin parametrizar [División por Distrito]**
Descripción	Consulta sobre "Sexo" sin añadir parámetros adicionales.
Entrada	Clic en "Consultar" con la opción "Sexo" seleccionada en el desplegable "Datos a representar:" y la opción "Todos" seleccionada en el resto de desplegables.
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.

Figura 7.19: [CP-84] Consulta sobre "Sexo" sin parametrizar [División por Distrito]

Título	[CP-85] Consulta sobre "Sexo" parametrizada por "Sección" [División por Distrito]
Descripción	Consulta sobre "Sexo" añadiendo una "Sección" como parámetro.
Entrada	Clic en "Consultar" con la opción "Sexo" seleccionada en el desplegable "Datos a representar:", una opción distinta de "Todas" en el desplegable "Sección" y la opción "Todos" seleccionada en el resto de desplegables.
Resultado esperado	Se obtiene el mismo resultado que a través de la consulta SQL.
Resultado obtenido	Resultado esperado.

Figura 7.20: [CP-85] Consulta sobre "Sexo" parametrizada por "Sección" [División por Distrito]

Se han llevado a cabo pruebas similares sobre cada tipo de distribución (Sexo, Procedencia, Estudios y Edad) parametrizada por todos los filtros posibles (Sección, Sexo, Procedencia (Nac.), Procedencia (Extr.), Estudios y Edad); así como por combinaciones de estos, tal y como se hizo con la pestaña "Valladolid (sin división)".

7.3.4 PRUEBAS SOBRE LA PESTAÑA "División por Zona Estadística"

Las pruebas ejecutadas sobre la pestaña "División por Zona Estadística" son similares a las realizadas sobre "División por Distrito", pero aplicadas sobre la pestaña correspondiente. Al no existir división dentro de las Zonas Estadísticas, las pruebas realizadas en "División por Distrito" con el parámetro "Sección" no se llevaron cabo sobre la división por Zona Estadística.

7.4 PRUEBAS DE PORTABILIDAD

La aplicación debe funcionar de forma independiente del navegador utilizado. Las pruebas [CP-1] a [CP-144] se ejecutaron originalmente desde Mozilla Firefox [60]. Para probar la independencia entre navegadores se ejecutaron de nuevo en los otros dos navegadores más utilizados: Google Chrome [55] e Internet Explorer [59].

Título	[CP-145] Portabilidad entre navegadores: Google Chrome
Descripción	Prueba de la aplicación en el navegador *Google Chrome*.
Entrada	Pruebas [CP-1] a [CP-144].
Resultado esperado	Mismo resultado que en las pruebas originales.
Resultado obtenido	Resultado esperado.

Figura 7.21: [CP-145] Portabilidad entre navegadores: Google Chrome

Título	**[CP-146] Portabilidad entre navegadores: Internet Explorer**
Descripción	Prueba de la aplicación en el navegador *Google Chrome*.
Entrada	Pruebas [CP-1] a [CP-144].
Resultado esperado	Mismo resultado que en las pruebas originales.
Resultado obtenido	Resultado esperado.

Figura 7.22: [CP-146] Portabilidad entre navegadores: Internet Explorer

Al realizar las pruebas de portabilidad se observó que en algunas máquinas las pruebas [CP-10] a [CP-15], [CP-23] a [CP-28] y [CP-35] a [CP-38] (consistentes en habilitar y deshabilitar elementos de la interfaz en función del elemento seleccionado en un desplegable) fallaban. Este problema no parecía depender del navegador o SO empleado, ya que se probó en distintas máquinas con el mismo SO y navegador web, funcionando según lo esperado en algunas de ellas y fallando en otras.

Se tomaron acciones correctoras para que no se produjese en ningún caso, añadiendo los desplegables problemáticos al Controlador de forma que se capturasen los eventos y se forzase la actualización de la Vista.

VIII. CONCLUSIONES

8.1 CONCLUSIONES

El principal objetivo de este Proyecto es el desarrollo e implementación de una aplicación web (disponible a través de Internet) para poder realizar consultas y visualizar gráficamente el contenido del Padrón Municipal de Valladolid. Para ello se elaboró una interfaz basada en APIs Java existentes para la representación gráfica de datos. Los datos debían obtenerse en un principio de una fuente externa de datos abiertos, estimándose inicialmente que su formato sería RDF. Debido a un retraso en el Proyecto que debía proveer esos datos se hizo necesario utilizar los datos proporcionados inicialmente en un fichero Access. De cara a la versión final el diseño de la base de datos se modificó y optimizó, pero el conjunto de datos permaneció inalterado.

El Proyecto contó con tres frentes principales: encontrar una forma satisfactoria de visualizar los distintos tipos de datos del Padrón Municipal, desarrollar una aplicación web para mostrar las gráficas y permitir al usuario interactuar con el contenido del Padrón y obtener los datos de una fuente de datos abiertos. Como ya se ha comentado, esta última parte no pudo llevarse a cabo, siendo sustituida por la optimización de una base de datos relacional.

Los tres frentes se cubrieron satisfactoriamente, por lo que se puede concluir que se alcanzó el objetivo didáctico del Trabajo Fin de Grado, consistente en la aplicación de los conocimientos adquiridos durante la carrera en un Proyecto real.

Durante el estudio de las tecnologías se estudió no solo la posibilidades de desarrollo de las mismas sino su la viabilidad de las mismas en base a distintos criterios. Las tecnologías escogidas cumplieron su cometido sin problemas, lo cual no quiere decir que las otras opciones descartadas fuesen menos válidas. El desarrollo pudo ser llevado a cabo sin problemas empleando *applets* Java y PHP, lo que permitió un acercamiento y profundización en tecnologías que se emplean en la actualidad en Proyectos tecnológicos empresariales.

Las opciones de visualización escogidas para los distintos datos del Padrón resultaron también correctas, siendo en cualquier caso válidas otras posibilidades como por ejemplo el empleo de gráficos de sectores para la representación del nivel de estudios o la procedencia de los individuos registrados en el Padrón.

Por último, la necesidad de disponer de una fuente de datos alternativa a la planteada inicialmente supuso alterar el plan de desarrollo previsto inicialmente. Esta situación, lejos de ser la ideal, supone un caso frecuente durante el desarrollo de software para su explotación comercial. Por lo tanto durante el desarrollo del Proyecto se ha cubierto una posibilidad interesante, como es la replanificación y adaptación a un escenario distinto al inicial. La necesidad de adaptar un recurso que inicialmente sería de prueba para su utilización en una versión final se puede valorar positivamente, al ser una acción requerida en ocasiones y que supone uno de los riesgos potenciales de muchos proyectos software.

Personalmente, el haber desarrollado el Proyecto de forma individual me ha permitido conocer y trabajar en todas las etapas del desarrollo de la aplicación (determinación de objetivos, planificación, estudio viabilidad, fases de análisis, diseño e implementación, pruebas, documentación…), permitiendo una aproximación más al desarrollo de software, complementada con el Proyecto Fin de Carrera realizado durante la Ingeniería Técnica. También me ha servido para acercarme de nuevo a tecnologías como Java o PHP, profundizando y descubriendo nuevas posibilidades en ellas, como la comunicación entre ambas o el desarrollo de *applets* en Java.

8.2 TRABAJO FUTURO

Uno de los principales atractivos de este Trabajo Fin de Grado era el acercamiento a los datos abiertos y la profundización en una de sus principales formas de difusión como es el estándar RDF. Lamentablemente no pudo realizarse, por lo que queda planteado como posible trabajo futuro.

La aplicación entregada puede servir de base para el desarrollo de una aplicación similar que pueda obtener los datos del Padrón de una fuente de datos abiertos como puede ser un repositorio de documentos RDF.

Para llevar a cabo esta adaptación bastaría inicialmente con modificar la clase que accede a los datos, sustituyendo las operaciones actuales por las necesarias para consultar, recibir y estructurar la información de otra fuente.

Esta tarea no debería suponer un gran coste, al estar la interfaz completamente desarrollada y existir APIs para Java centradas en la comunicación con XML en general y RDF en particular.

Otra posibilidad de trabajo futuro podría consistir en la revisión y estandarización de los datos contenidos en la base de datos relacional, al ser en algunos casos incompletos o contradictorios. Esta tarea sin embargo debería recaer en la persona o personas encargadas de su mantenimiento más que barajarse como posibilidad de futuro Proyecto software.

IX. BIBLIOGRAFÍA

Referencias bibliográficas

[1] Dawson, Christian W. & Martín, Gregorio. "El Proyecto Fin de Carrera en Ingeniería Informática: Una Guía para el Estudiante". Prentice-Hall, 2002

[2] Gilbert, David. "The JFreeChart Class Library - Developer Guide". Object Refinery Limited, 2004

[3] Larman, Craig. "UML y Patrones. Introducción al Análisis y Diseño Orientado a Objetos y al Proceso Unificado". Prentice-Hall, 2004

Fuentes Web

[4] Catálogo de Información Pública de la Administración General del Estado [en línea]. Disponible en http://datos.gob.es/datos/ [Última consulta 01/11/2012]

[5] Adobe Flash Professional CS6 [en línea]. Disponible en http://www.adobe.com/es/products/flash.html [Última consulta 28/08/2012]

[6] Applets Java [en línea]. Disponible en http://java.sun.com/applets/ [Última consulta 29/08/2012]

[7] What Applets Can and Cannot Do [en línea]. Disponible en http://docs.oracle.com/javase/tutorial/deployment/applet/security.html [Última consulta 13/10/2012]

[8] JDBC in Applets [en línea]. Disponible en http://docs.oracle.com/cd/B14117_01/java.101/b10979/applet.htm [Última consulta 13/10/2012]

[9] Capturing Architectural Requirements [en línea]. Disponible en http://www.ibm.com/developerworks/rational/library/4706.html [Última consulta 06/10/2012]

[10] ASP.NET [en línea]. Disponible en http://www.asp.net/ [Última consulta 28/08/2012]

[11] Ayuntamiento de Valladolid [en línea]. Disponible en http://www.valladolid.es/ [Última consulta 27/08/2012]

[12] Datos Abiertos - Ayuntamiento de Valladolid [en línea]. Disponible en http://www.valladolid.es/es/servicios/open-data-datos-abiertos [Última consulta 01/11/2012]

[13] Datos enlazados en la BNE [en línea]. Disponible en http://www.bne.es/es/Inicio/Perfiles/Bibliotecarios/DatosEnlazados/index.html [Última consulta 01/11/2012]

[14] Carto:net [en línea]. Disponible en http://www.carto.net/ [Última consulta 28/08/2012]

[15] España en llamas [en línea]. Disponible en http://www.espanaenllamas.es/ [Última consulta 01/11/2012]

[16] Fundación CTIC [en línea]. Disponible en http://www.fundacionctic.org/ [Última consulta 01/11/2012]

[17] Fundación Ciudadana Civio [en línea]. Disponible en http://www.civio.es/ [Última consulta 01/11/2012]

[18] GeoTools [en línea]. Disponible en http://www.geotools.org/ [Última consulta 29/08/2012]

[19] GNU Lesser General Public License [en línea]. Disponible en http://www.gnu.org/licenses/lgpl.html [Última consulta 27/08/2012]

[20] Google Maps API [en línea]. Disponible en https://developers.google.com/maps/?hl=es [Última consulta 28/08/2012]

[21] IBM Rational Unified Process [en línea]. Disponible en http://www-01.ibm.com/software/awdtools/rup/ [Última consulta 27/08/2012]

[22] Introducción a Java [en línea]. Disponible en http://www.programacion.com/articulo/introduccion_a_java_80 [Última consulta 30/08/2012]

[23] Santhosh Kumar's Weblog - Make JComboBox popup wide enough [en línea]. Disponible en http://www.jroller.com/santhosh/entry/make_jcombobox_popup_wide_enough [Última consulta 14/10/2012]

[24] JFreeChart [en línea]. Disponible en http://www.jfree.org/jfreechart/ [Última consulta 29/08/2012]

[25] Javadoc JFreeChart [en línea]. Disponible en http://www.jfree.org/jfreechart/api/javadoc/index.html [Última consulta 30/08/2012]

[26] JFreeChart Forum [en línea]. Disponible en http://www.jfree.org/phpBB2/viewforum.php?f=3 [Última consulta 30/08/2012]

[27] JpGraph [en línea]. Disponible en http://jpgraph.net/index.php/ [Última consulta 28/08/2012]

[28] Portal de Datos Abiertos de la Junta de Castilla y León [en línea]. Disponible en http://www.datosabiertos.jcyl.es/ [Última consulta 01/11/2012]

[29] Ley de Bases del Régimen Local [en línea]. Disponible en http://www.boe.es/boe/dias/1985/04/03/pdfs/A08945-08964.pdf [Última consulta 27/08/2012]

[30] Ley de Protección de Datos de Carácter Personal [en línea]. Disponible en http://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf [Última consulta 27/08/2012]

[31] Linked Data - Connect Distributed Data across the Web [en línea]. Disponible en http://linkeddata.org/ [Última consulta 27/08/2012]

[32] Guía Breve de Linked Data [en línea]. Disponible en http://w3c.es/Divulgacion/GuiasBreves/LinkedData [Última consulta 27/08/2012]

[33] Design Issues: Linked Data [en línea]. Disponible en http://www.w3.org/DesignIssues/LinkedData.html [Última consulta 27/08/2012]

[34] Original MySQL API [en línea]. Disponible en http://www.php.net/manual/en/book.mysql.php [Última consulta 27/10/2012]

[35] MySQL Improved Extension [en línea]. Disponible en http://www.php.net/manual/en/book.mysqli.php [Última consulta 27/10/2012]

[36] The Open Data Foundation [en línea]. Disponible en http://www.opendatafoundation.org/ [Última consulta 27/08/2012]

[37] The Open Data Handbook [en línea]. Disponible en http://opendatahandbook.org/ [Última consulta 27/08/2012]

[38] Fuentes de Datos Abiertos en España [en línea]. Disponible en https://datospublicos.jottit.com/ [Última consulta 01/11/2012]

[39] Definición de Conocimiento Abierto - Open Definition [en línea]. Disponible en http://opendefinition.org/okd/espanol/ [Última consulta 01/11/2012]

[40] Open Flash Chart [en línea]. Disponible en http://teethgrinder.co.uk/open-flash-chart/ [Última consulta 28/08/2012]

[41] Mission - Open Source Initiative [en línea]. Disponible en http://opensource.org/ [Última consulta 01/11/2012]

[42] Padrón Municipal. Metodología [en línea]. Disponible en http://www.ine.es/metodologia/t20/t203024566.htm [Última consulta 27/08/2012]

[43] Padrón Municipal. Explotación Estadística [en línea]. Disponible en http://www.ine.es/jaxi/menu.do?type=pcaxis&path=%2Ft20%2Fe245%2F&file=inebase [Última consulta 27/08/2012]

[44] Avance de la Explotación Estadística del Padrón a 1 de enero de 2012 [en línea]. Disponible en http://www.ine.es/prensa/np710.pdf [Última consulta 27/08/2012]

[45] Revista Índice. Diferencias entre Censo de Población y Padrón Municipal [en línea]. Disponible en http://www.revistaindice.com/numero3/p12.pdf [Última consulta 27/08/2012]

[46] Top 10 de Librerías para Gráficos en PHP [en línea]. Disponible en http://tednologia.com/top-10-librerias-graficos-php/ [Última consulta 28/08/2012]

[47] PHP [en línea]. Disponible en http://www.php.net/ [Última consulta 28/08/2012]

[48] PHP Data Objects [en línea]. Disponible en http://www.php.net/manual/en/book.pdo.php [Última consulta 13/10/2012]

[49] Sending a POST Request Using a URL [en línea]. Disponible en http://www.exampledepot.com/egs/java.net/Post.html [Última consulta 21/10/2012]

[50] ¿Qué es Software Libre? [en línea]. Disponible en http://www.gnu.org/philosophy/free-sw.es.html [Última consulta 01/11/2012]

[51] Tim Berners-Lee [en línea]. Disponible en http://www.w3.org/People/Berners-Lee/ [Última consulta 27/08/2012]

[52] World Wide Web Consistorium [en línea]. Disponible en http://www.w3.org/ [Última consulta 27/08/2012]

[53] World Wide Web Foundation [en línea]. Disponible en http://www.webfoundation.org/ [Última consulta 27/08/2012]

[54] Guía Breve de la Web Semántica [en línea]. Disponible en http://w3c.es/Divulgacion/GuiasBreves/WebSemantica [Última consulta 27/08/2012]

Referencias de software empleado en el Proyecto

[55] Google Chrome [en línea]. Disponible en https://www.google.com/intl/es/chrome/browser/?hl=es [Última consulta 09/10/2012]

[56] Java [en línea]. Disponible en http://www.java.com/es/download/ [Última consulta 10/11/2012]

[57] Microsoft Access 2000 [en línea]. Disponible en http://support.microsoft.com/ph/2484 [Última consulta 09/10/2012]

[58] Microsoft Project 2007 [en línea]. Disponible en http://www.microsoft.com/project/en-us/project-2007.aspx [Última consulta 27/08/2012]

[59] Microsoft Internet Explorer [en línea]. Disponible en http://windows.microsoft.com/es-ES/internet-explorer/download-ie [Última consulta 09/10/2012]

[60] Mozilla Firefox [en línea]. Disponible en http://www.mozilla.org/es-ES/firefox/fx/ [Última consulta 09/10/2012]

[61] UMLet [en línea]. Disponible en http://www.umlet.com/ [Última consulta 04/11/2012]

ANEXO A. INSTALACIÓN DE LA APLICACIÓN

A.1 INTRODUCCIÓN

En este anexo se detalla en primer lugar la instalación de la aplicación para el cliente (usuario), donde se detallan los componentes necesarios para ejecutar la aplicación y en segundo lugar la instalación en el servidor, con instrucciones para configurar los parámetros de los componentes del sistema y el código SQL necesario para la creación de las tablas de la base de datos.

A.2 INSTALACIÓN DE LA APLICACIÓN (CLIENTE)

Los únicos requerimientos para poder acceder a la aplicación son haber Java instalado en el equipo, disponer de un navegador web compatible con Java y tener conexión a Internet.

Navegador

La mayoría de los SO actuales incluyen un navegador capaz de ejecutar *applets*. Para la poder trabajar con la aplicación se debe habilitar la ejecución de este tipo de código para el sitio web en el que se aloja la aplicación. Está opción solo se podrá llevar a cabo si Java está instalado en el equipo.

Java

Java puede descargarse de su página web [56]. La página detectará el SO y ofrecerá la versión de Java adecuada. Sin embargo se puede acceder a las opciones completas de descarga, donde se podrá escoger SO, versión de Java y tipo de instalación (online u offline). La instalación consta de una secuencia sencilla de pasos guiada a través de una interfaz.

Es recomendable activar las actualizaciones tras la instalación para disponer de la última versión del software en todo momento, que incluirá mejoras y corregirá errores y posibles problemas de seguridad.

A.3 INSTALACIÓN DE LA APLICACIÓN (SERVIDOR)

En este apartado se detalla la configuración de los parámetros de los distintos componentes que se deben ser configurados a la hora de instalar la aplicación en un nuevo servidor, así como la creación de las tablas de la base de datos.

Configuración de la página HTML

La configuración del HTML no tiene porqué variar, salvo que se ubiquen los paquetes Java (jar) en un directorio distinto. En ese caso debería modificarse el atributo `archive` de la etiqueta `applet` con la localización de los ficheros padron.jar, jfreechart-1.0.14.jar y jcommon-1.0.17.jar.

Parámetros de la clase `Padron`

Dentro de la clase `Padron` debe modificarse el parámetro `phpURL` de forma que contenga la URL para acceder a través de Internet al script PHP encargado de comunicarse con la base de datos.

Parámetros del script PHP

En el script PHP deben modificarse los siguientes parámetros:

24. `DBHOST`. URL del servidor que aloja la base de datos.
25. `DBNAME`. Nombre de la base de datos.
26. `DBUSER`. Nombre del usuario de la base de datos.
27. `DBPASS`. Contraseña del usuario de la base de datos.

Creación de tablas de la Base de Datos

El código SQL para la creación de las tablas que conforman la base de datos puede verse en las siguientes figuras. Se ha obviado en la memoria el contenido de las tablas debido a su extensión. El script SQL completo (incluyendo las sentencias para añadir los datos a la base de datos) se encuentra disponible en el CD.

```
CREATE TABLE IF NOT EXISTS `CODIGO_COM_AUTON` (
  `ID` int(1) NOT NULL,
  `COM_AUTON` varchar(22) character set utf8 collate
utf8_spanish_ci NOT NULL,
  `COD_COM_AUTON` varchar(3) character set utf8 collate
utf8_spanish_ci NOT NULL,
  PRIMARY KEY  (`ID`),
  KEY `COD_COM_AUTOM` (`COD_COM_AUTON`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figura A.1: Creación de la tabla `CODIGO_COM_AUTON`

```
CREATE TABLE IF NOT EXISTS `CODIGO_DTO_SECCION` (
  `ZONA_ESTADISTICA` varchar(3) NOT NULL,
  `DISTRITO` int(1) NOT NULL,
  `SECCION` int(1) NOT NULL,
  PRIMARY KEY  (`DISTRITO`,`SECCION`),
  KEY `DISTRITO` (`DISTRITO`),
  KEY `SECCION` (`SECCION`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figura A.2: Creación de la tabla `CODIGO_DTO_SECCION`

```
CREATE TABLE IF NOT EXISTS `CODIGO_ESTUDIOS` (
  `ID` int(1) NOT NULL,
  `ESTUDIOS` varchar(150) character set utf8 collate
utf8_spanish_ci NOT NULL,
  `ESTUDIOS_INE` varchar(39) character set utf8 collate
utf8_spanish_ci NOT NULL,
  `CODIGO_INE` varchar(5) character set utf8 collate
utf8_spanish_ci NOT NULL,
  PRIMARY KEY  (`ID`),
  KEY `CODIGO_INE` (`CODIGO_INE`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figura A.3: Creación de la tabla CODIGO_ESTUDIOS

```
CREATE TABLE IF NOT EXISTS `CODIGO_PAIS` (
  `ID` int(2) NOT NULL,
  `ID_CONTINENTE` int(1) NOT NULL,
  `CONTINENTE` varchar(17) character set utf8 collate
utf8_spanish_ci NOT NULL,
  `PAIS` varchar(31) character set utf8 collate utf8_spanish_ci
NOT NULL,
  `COD_CONTINENTE` varchar(4) character set utf8 collate
utf8_spanish_ci NOT NULL,
  PRIMARY KEY  (`ID`),
  KEY `PAIS` (`PAIS`),
  KEY `COD_CONTINENTE` (`COD_CONTINENTE`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figura A.4: Creación de la tabla CODIGO_PAIS

```
CREATE TABLE IF NOT EXISTS `CODIGO_PROVINCIA` (
  `ID` int(1) NOT NULL,
  `PROVINCIA` varchar(22) character set utf8 collate
utf8_spanish_ci NOT NULL,
  `ID_COM_AUTON` int(1) NOT NULL,
  PRIMARY KEY  (`ID`),
  KEY `ID_COM_AUTON` (`ID_COM_AUTON`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figura A.5: Creación de la tabla CODIGO_PROVINCIA

```
CREATE TABLE IF NOT EXISTS `CODIGO_SEXO` (
  `ID` int(1) NOT NULL default '0',
  `SEXO` varchar(5) character set utf8 collate utf8_spanish_ci
NOT NULL,
  PRIMARY KEY  (`ID`),
  KEY `SEXO` (`SEXO`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figura A.6: Creación de la tabla CODIGO_SEXO

```
CREATE TABLE IF NOT EXISTS `PADRON_HABITANTES` (
  `ID` int(4) NOT NULL auto_increment,
  `NUCLEO_FAMILIAR` int(4) NOT NULL,
  `DISTRITO` int(1) NOT NULL,
  `SECCION` int(1) NOT NULL,
  `ANHO_NACIMIENTO` int(2) NOT NULL,
  `MES_NACIMIENTO` int(1) NOT NULL,
  `ID_PROVINCIA_NAC` int(1) NOT NULL,
  `ID_SEXO` int(1) NOT NULL,
  `EDAD` int(1) NOT NULL,
  `ID_NACIONALIDAD` int(2) NOT NULL,
  `ID_ESTUDIOS` int(1) NOT NULL,
  PRIMARY KEY  (`ID`),
  KEY `DISTRITO` (`DISTRITO`),
  KEY `EDAD` (`EDAD`),
  KEY `SECCION` (`SECCION`),
  KEY `ID_PROVINCIA_NAC` (`ID_PROVINCIA_NAC`),
  KEY `ID_SEXO` (`ID_SEXO`),
  KEY `ID_ESTUDIOS` (`ID_ESTUDIOS`),
  KEY `ID_NACIONALIDAD` (`ID_NACIONALIDAD`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=314937 ;
```

Figura A.7: Creación de la tabla PADRON_HABITANTES

```
--
-- Filtros para la tabla `CODIGO_PROVINCIA`
--
ALTER TABLE `CODIGO_PROVINCIA`
  ADD CONSTRAINT `CODIGO_PROVINCIA_ibfk_1` FOREIGN KEY
(`ID_COM_AUTON`) REFERENCES `CODIGO_COM_AUTON` (`ID`) ON DELETE
CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `PADRON_HABITANTES`
--
ALTER TABLE `PADRON_HABITANTES`
  ADD CONSTRAINT `PADRON_HABITANTES_ibfk_9` FOREIGN KEY
(`ID_NACIONALIDAD`) REFERENCES `CODIGO_PAIS` (`ID`) ON DELETE
CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `PADRON_HABITANTES_ibfk_2` FOREIGN KEY
(`SECCION`) REFERENCES `CODIGO_DTO_SECCION` (`SECCION`) ON
DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `PADRON_HABITANTES_ibfk_3` FOREIGN KEY
(`ID_PROVINCIA_NAC`) REFERENCES `CODIGO_PROVINCIA` (`ID`) ON
DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `PADRON_HABITANTES_ibfk_4` FOREIGN KEY
(`ID_SEXO`) REFERENCES `CODIGO_SEXO` (`ID`) ON DELETE CASCADE
ON UPDATE CASCADE,
  ADD CONSTRAINT `PADRON_HABITANTES_ibfk_7` FOREIGN KEY
(`DISTRITO`) REFERENCES `CODIGO_DTO_SECCION` (`DISTRITO`) ON
DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `PADRON_HABITANTES_ibfk_8` FOREIGN KEY
(`ID_ESTUDIOS`) REFERENCES `CODIGO_ESTUDIOS` (`ID`) ON DELETE
CASCADE ON UPDATE CASCADE;
```

Figura A.8: Filtros para las tablas CODIGO_PROVINCIA y PADRON_HABITANTES

ANEXO B. MANUAL DE USO DE LA APLICACIÓN

B.1 MANUAL DE USO DE LA APLICACIÓN

La aplicación de halla dividida en cuatro secciones: Valladolid (sin división), División por Distrito, División por Zona Estadística e Información – Ayuda. A continuación se describen las opciones disponibles en cada uno de estos apartados.

B.1.1 PESTAÑA "Valladolid (sin división)"

Esta pestaña (Figura B.1) permite visualizar información y realizar consultas sobre el total de la población incluida en el Padrón Municipal, permitiendo también extrapolar estos resultados sobre las divisiones de la ciudad (Distritos y Zona Estadísticas).

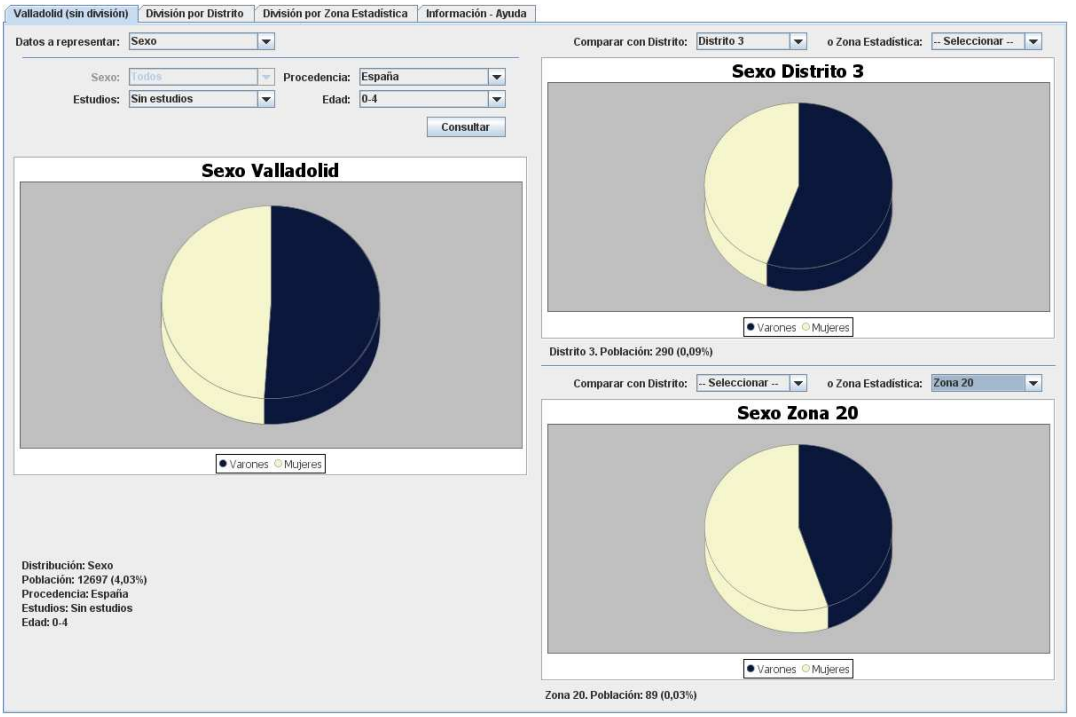

Figura B.1: Pestaña “Valladolid (sin división)”

Realizar una consulta

Los pasos para realizar una consulta son los siguientes:

1. Seleccionar la distribución a consultar en el desplegable “Datos a representar:”.

2. Definir uno o más filtros para la consulta. El filtro que coincida con la distribución seleccionada estará deshabilitado y no se podrá seleccionar.

3. Hacer clic en el botón “Consultar”.

En este momento el sistema realizará la consulta y mostrará como resultado una gráfica e información sobre la consulta realizada (tipo de distribución, población representada, porcentaje representado respecto al total y parámetros seleccionados para la consulta). Además pasarán a estar disponibles las comparativas con Distritos y Zonas Estadísticas.

Comparar con Distrito o Zona Estadística

El resultado de una consulta se puede comparar con el de esa misma consulta sobre uno o varios Distritos y Zonas Estadísticas. Para ello basta con seleccionar el Distrito o Zona Estadística en uno de los cuatro desplegables de la derecha, habilitados tras la realización de una consulta.

Como resultado se muestra la gráfica de la consulta sobre la división seleccionada junto al identificador de la división, número de individuos que cumplen la consulta y el porcentaje que representan respecto al total.

Las comparativas mostradas están siempre relacionadas con el resultado de la consulta actual. Si se realizara una nueva consulta, las comparativas desaparecerían, pudiéndose realizar nuevas extrapolaciones de la consulta más reciente.

B.1.2 PESTAÑA "División por Distrito"

Esta pestaña permite visualizar información y realizar consultas sobre un Distrito de Valladolid. Las opciones de visualización a nivel general y consulta se encuentran separadas en otras dos pestañas.

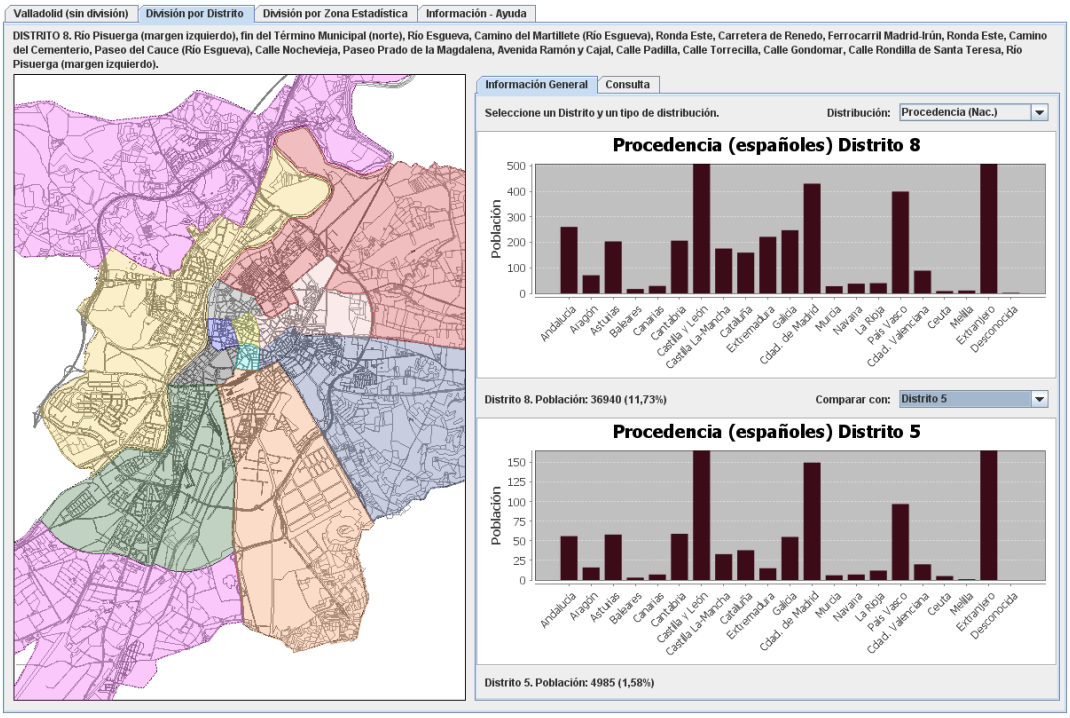

Figura B.2: Pestaña "División por Distrito – Información General"

Visualizar la Información General de un Distrito

La pestaña “Información General” permite obtener gráficas de las distintas distribuciones aplicadas sobre el total del Distrito (Figura B.2). Los pasos para visualizar esta información son los siguientes:

1. Seleccionar un Distrito. En la parte superior de la aplicación se muestran los límites de cada Distrito según se van recorriendo. En cuanto se seleccione uno, quedará fijo.

2. Seleccionar dentro de la pestaña “Información General” la distribución para la que se desea obtener la gráfica.

En este momento el sistema realizará la consulta y mostrará como resultado una gráfica e información sobre la consulta realizada (Distrito seleccionado, población representada y porcentaje representado respecto al total). Además pasará a estar disponible la comparativa con otro Distrito.

Comparar con otro Distrito

Para comparar la información desplegada con otro Distrito, basta con seleccionar el Distrito deseado en el desplegable inferior. El resultado de la consulta se mostrará en la parte inferior de forma similar a la consulta original.

La comparativa estará siempre relacionada con el resultado actual. Si se realizara una nueva consulta, la comparativa desaparecería, pudiéndose realizar una nueva sobre la consulta más reciente.

Realizar una consulta

La pestaña “Consulta” permite obtener gráficas de una consulta parametrizada sobre un Distrito (Figura B.3). Los pasos para realizar una consulta son los siguientes:

1. Seleccionar un Distrito. En la parte superior de la aplicación se muestran los límites de cada Distrito según se van recorriendo. En cuanto se seleccione uno, quedará fijo.

2. Seleccionar la distribución a consultar en el desplegable “Datos a representar:”.

3. Definir uno o más filtros para la consulta. El filtro que coincida con la distribución seleccionada estará deshabilitado y no se podrá seleccionar.

4. Hacer clic en el botón “Consultar”.

En este momento el sistema realizará la consulta y mostrará como resultado una gráfica e información sobre la consulta realizada: Distrito, Sección, tipo de distribución, población representada, porcentaje representado respecto al total y parámetros seleccionados para la consulta.

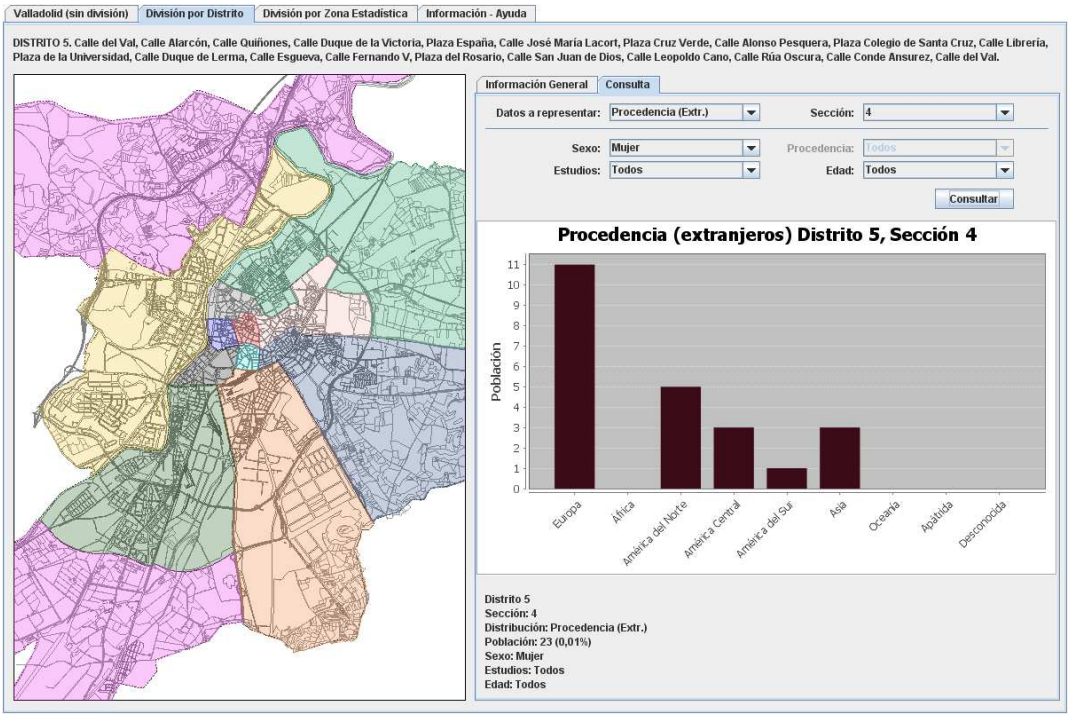

Figura B.3: Pestaña “División por Distrito – Consulta”

B.1.3 PESTAÑA “División por Zona Estadística”

El funcionamiento de esta pestaña “División por Zona Estadística” es similar al de la pestaña “División por Distrito”. En esta ocasión los elementos seleccionables en el mapa serán Zonas Estadísticas, y no habrá posibilidad de filtrar las consultas por Sección, puesto que en este caso no existe dicha subdivisión.

B.1.4 PESTAÑA “Información – Ayuda”

Esta pestaña muestra una descripción breve del funcionamiento de la aplicación; así como una descripción de los distintos parámetros seleccionables en cada filtro.

ANEXO C. CONTENIDO DEL CD

C.1 CONTENIDO DEL CD

El contenido del CD se encuentra estructurado de la siguiente forma:

- **TFG_PADRON**
  - **Memoria**
    - **memoria.pdf**
  - **Software**
    - **padron**
    - **Código fuente**
    - **padron.jar**
    - **creacion_bd.sql**
  - **Utilidades**
    - **JFreeChart**
    - **JCommon**
  - **Batería de Pruebas**
    - **Batería_de_pruebas.pdf**

A continuación se detalla el contenido de cada carpeta:

- **Memoria:** Contiene el documento memoria.pdf, con la presente memoria siguiendo la normativa ISO B5.
- **Software:** Contiene los siguientes archivos y carpetas:
    - **padron:** Carpeta que contiene las clases de que consta la aplicación.
    - **Código fuente:** Carpeta que contiene el código fuente de la aplicación.
    - **padrón.jar: F**ichero con la aplicación para ser llamado desde una página HTML.
    - **creacion_bd.sql:** Script de creación de la base de datos (tablas y contenido).
- **Utilidades:** Contiene las siguientes carpetas:
    - **JFreeChart:** Biblioteca JFreeChart, necesaria para el uso de la aplicación.
    - **JCommon:** Biblioteca JCommon, necesaria para el uso de la aplicación.
- **Batería de pruebas:** contiene el documento "batería de pruebas.pdf", con la batería de pruebas completa no incluida en la memoria.