Título: ¿Cuándo dos antepasados son la misma persona?

Alumno: Glen Patrick Caliba Ramos

Fecha: 05 de junio de 2012

Director: Enrique Mayol Sarroca

Departamento: ESSI

Titulación: Ingeniería informática

Centre: Facultad de Informática de Barcelona (FIB)

Universidad: Universidad Politécnica de Catalunya (UPC) BarcelonaTech

Índice de Contenidos

ÍNDICE DE CONTENIDOS 2

INTRODUCCIÓN 5

CONTEXTO DEL PROYECTO 6

OBJETIVO 9

PLANIFICACIÓN Y COSTES 11

PLANIFICACIÓN 11
FASE INICIAL 13
FASE DE DESARROLLO 15
FASE DE DOCUMENTACIÓN 17
VALORACIÓN ECONÓMICA 20

HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS 22

ARGOUML 22
NETBEANS 22
MAMP 23
MYSQL 23
PHP 23
CSS 24
HTML 24
GEDCOM 24

ESPECIFICACIÓN 26

ELEMENTOS A CONSIDERAR 26
GEDCOM 26
ATRIBUTOS QUE USAREMOS DE UNA PERSONA. 27
MODELO CONCEPTUAL 31
PRIMERA ALTERNATIVA 31
SEGUNDA ALTERNATIVA 34
MODELO CONCEPTUAL PARA EL CÁLCULO DE PROBABILIDADES 35
MODELO CONCEPTUAL FINAL 37
REGLAS 39
CLASE REGLA 39
CASOS DE USO 40
DIAGRAMA DE CASO DE USO 40
ESPECIFICACIÓN DE LOS CASOS DE USO 42

DISEÑO 48

ARQUITECTURA FÍSICA DEL SISTEMA 48
DESARROLLO DESDE CERO UTILIZANDO UN FRAMEWORK 48
ARQUITECTURA LÓGICA DEL SISTEMA 49
ARQUITECTURA BASADO EN EL PATRÓN MVC (MODELO VISTA CONTROLADOR) 49
ORIENTACIÓN A OBJETO 51
VISTA 52
MAPA DE NAVEGACIÓN 52

DISEÑO EXTERNO 54
CONTROLADOR 72
CLASE CONTROLLER 73
ALGORITMO PARA EL CALCULO DE PROBABILIDADES 74
MODELO 80
DISEÑO DE LA BASE DE DATOS 82
REGLAS 92
REPRESENTACIÓN DE UNA REGLA 92
COMPORTAMIENTO DE UNA REGLA. 92
VALORES DE UNA REGLA 93

IMPLEMENTACIÓN 95

DESCRIPCIÓN DEL ENTORNO DE DESARROLLO 95
ESTRUCTURA DEL FRAMEWORK 98
CONTROLADORES 99
MODELOS 101
VISTAS 104
REGLAS 106
INCLUSIÓN DE UNA REGLA 108
USO DE UNA REGLA 109

PRUEBAS 111

PRUEBAS FUNCIONALES 111
INSERCIÓN DE UNA NUEVA REGLA 112
MODIFICACIÓN DE UNA REGLA 112
ELIMINACIÓN DE UNA REGLA 112
CALCULO DE PROBABILIDADES 112
PRUEBAS DE USABILIDAD 113
PRUEBAS DE RENDIMIENTO 114

CONCLUSIONES Y TRABAJO FUTURO 116

CONCLUSIONES Y VALORACIÓN PERSONAL 116
TRABAJO FUTURO 117
NUEVAS FUNCIONALIDADES 117
MEJORAR EL CÁLCULO DE PROBABILIDADES 117
CONVERTIRLO EN HERRAMIENTA ONLINE 118

BIBLIOGRAFÍA Y REFERENCIAS 119

BIBLIOGRAFÍA 119
REFERENCIAS 119
ARCHIVOS Y DOCUMENTOS WEBS 119
FUENTES GENEALÓGICAS ONLINE 119

INSTALACIÓN Y EJECUCIÓN 122

INSTALACIÓN 122
MAMP (IOS) 122
INSTALACIÓN EN LINUX (UBUNTU) 122
CONFIGURAR EL SERVIDOR 125
MAMP Y LINUX 125
MAMP PRO 128
UTILIZACIÓN 130
GESTIONAR REGLAS 130

CALCULAR PROBABILIDAD 132
VER RESULTADOS 133

DIAGRAMAS UTILIZADOS 135

INSERCIÓN DE DATOS 143

Introducción

El proyecto que se presenta a continuación consiste en el desarrollo de una herramienta que permita calcular la probabilidad de que dos personas puedan ser la misma a partir de los datos que se tienen.

La información que se utiliza para este cálculo, son datos personales, relaciones entre personas o sobre eventos de su vida, como por ejemplo la fecha de nacimiento o de casamiento.

De este conjunto de datos se ha seleccionado los que interesa tener en cuenta y descartar los que no aportan información para el cálculo de la probabilidad. Para la herramienta se requiere una base de datos con esta información obtenidas de fuentes de interés genealógico.

La herramienta que se ha desarrollado esta dirigido principalmente para los geneólogos. Ellos obtienen datos de personas a partir de diferentes fuentes de interés genealógico. Es muy común que una misma persona pueda salir en diferentes documentos. Como consecuencia, después de recopilar información, se obtiene una base de datos en la que una misma persona puede salir varias veces identificada como diferentes personas y se necesita analizar estos individuos para poder tener información más completa de una persona en vez de tenerlo repartido entre diferentes personas que podrían ser la misma.

Por esta razón se ha pensado en hacer una herramienta para poder identificar una persona con otra y poder medir el grado de similitud de que dos personas sean la misma para después fusionar la información. Esta herramienta facilitaría el trabajo dentro del campo de la genealogía, pudiendo automatizar la tarea de comprobar si diferentes personas son la misma.

Contexto del proyecto

Actualmente se vive en una época, donde hay diferentes disciplinas en las cuales se trabajan con información de carácter personal. En las bases de datos donde se guardan esta información hay un gran numero de información personal. Hay muchas probabilidades que la información de una persona se pueda completar con la de otra, ya que pueden ser la misma pero con la diferencia de que la información pertenece a momentos o situaciones diferentes. En esta situación se ha tratado una misma persona como si fueran diferentes.

Un ejemplo del uso de estos datos es en el campo de la criminología o en la genealogía. En el caso de la criminología , los datos se utilizan para poder identificar un perfil determinado o identificar a personas que pueden tener la probabilidad de ser la misma a partir de los datos disponibles. De esta manera tienen un conjunto de personas que tienen datos en común como es el nombre procedencia o trabajos que a realizado.

Otro uso que se podría dar en la criminología es la de poder facilitar la tarea de fusionar varias bases de datos de personas que pueden tener diferentes departamentos. Un ejemplo del uso es a la hora de obtener expedientes de una persona a partir de diferentes bases de datos de varios departamento. Se tendría que mirar en cada una de las bases de datos coincidencia.

La genealogía se encarga de estudiar las relaciones de parentesco que hay entre varias personas. Un reflejo de este trabajo son los arboles genealógico, en los cuales se ve de forma gráfica las relaciones de parentesco que hay entre los individuos que salen. Para poder llevar a cabo su función, la genealogía necesita información de personas para poder concretar estas relaciones. Para esto tienen que buscar información de documentos de interés genealógico como el registro civil o partidas de nacimiento.

En la genealogía se usa la información de las personas para poder determinar la ascendencia o descendencia de una persona. Dentro de la genealogía se usa información procedente de los diferentes eventos que ha tenido una persona , como son nacimientos , bautismos o bodas. También se usa información de las relaciones familiares que tienen. A partir de toda esta información se consigue construir una genealogía.

Una dificultad que se puede tener al construir una genealogía es que hay la posibilidad de encontrar en varias fuentes documentales, información de una misma personas.

Como resultado de esta recopilación de datos personales de diferentes fuentes, se puede obtener una base de datos donde la información de una misma persona esta repartida entre diferentes individuos. Esto se debe a que se han tratado cada una de estos como si fueran distintos pero en realidad son el mismo.

Para poder completar la información de una persona se tiene que ir analizando cada persona con el resto que aparecen en la base datos. Con cada análisis se determina la probabilidad de que las personas que se esta comparando sean la misma usando el sentido común. A partir de este análisis hay dos opciones, juntar o no la información de estas dos personas.

Otra utilidad que se puede tener dentro de la genealogía es en la fusión de arboles genealógicos. Estos árboles son un reflejo de la base de datos que se tienen. Si se puede determinar la probabilidad de que dos personas sean la misma, se podría fusionar dos arboles genealógicos a partir de un punto en común (una persona). Como consecuencia, se pueden fusionar las bases de datos que representan los arboles fusionados.

El problema que se tiene al hacer el análisis es que se puede detectar contradicciones, inconsistencias, incoherencias, falta de datos, etc. Por otro lado, existe la posibilidad de que esta incoherencia de los datos haga pensar que dos descripciones de personas de documentos distintos (o del mismo) puedan referirse a una misma persona. También se puede dar el caso de que se detecte que parecen ser la misma persona,

pero a causa de un dato concreto, esta impida hacer tal afirmación, no pudiendo consensuar los datos (como es por ejemplo el sexo de una persona).

Por lo tanto, sería interesante tener una herramienta que permita analizar la probabilidad que diferentes personas puedan ser la misma, para poder tener una fuente donde la información de una persona este unificada y no este instanciada una persona en varias o también para poder fusionar base de datos en uno.

Objetivo

El objetivo principal que tiene este proyecto es la de desarrollar una herramienta que permite aplicar unas reglas en una base de datos. Al aplicar las reglas se obtiene el grado de similitud que pueden tener dos personas. A partir de estos cálculos el usuario podrá decidir si considera estas dos personas la misma o no.

Para poder conseguir este objetivo se han hecho las tareas que se describen a continuación:

Definir un modelo conceptual

Obtener un modelo conceptual que identifica todos los elementos que necesita la herramienta. Esto quiere decir identificar las clases que la aplicación usará para poder llevar a cabo las funcionalidades que se esperan.

Definir los atributos

Identificar lo atributos de una persona que son útiles a la hora de identificar la similitud entre dos personas. Para esto el atributo tiene que ser bastante identificativo o que junto a otro, ayude a diferenciar de forma efectiva personas diferentes.

Definir Reglas

Estas reglas son las que se utilizan para determinar la similitud. Cada una de las reglas tendrán en cuenta ciertos atributos para dar una probabilidad parcial.

Diseño externo de la interface

Diseñar las diferentes pantallas que tendrá la interface de la aplicación.

Implementación del funciones y clases

Implementar todas las funcionalidades y clases que se han definido previamente para que la herramienta pueda llevar a cabo su objetivo.

Definir un mapa navegacional de la herramienta

Este mapa muestra las relaciones entre las diferentes pantallas dependiendo del comportamiento del usuario. Este mapa puede facilitar la tarea de añadir nuevas funcionalidades en el futuro.

Comprobar el buen funcionamiento de la herramienta.

Hay que comprobar que las funcionalidades que se han implementado tengan el comportamiento esperado. Para esto se harán diferentes pruebas.

Documentación y extensibilidad del código

Este objetivo pretende facilitar la tarea de extender las funcionalidades de la herramienta. Para esto el código tienen que estar bien documentada y legible.

Planificación y Costes

En este apartado se comentan las cuestiones relativas a la planificación y al coste del proyecto..

Planificación

El proyecto esta pensado para que tenga una durada 15 semanas con un trabajo de 20 horas por semana (600 horas en total).
Durante el desarrollo es probable que la planificación haya sufrido cambios debido a imprevistos. A continuación se detalla la planificación real, exponiendo las diferencias con respecto a la planificación inicial.

La realización del proyecto se podría dividir en 3 grandes partes.

1. Fase inicial consiste en un análisis y planificación del proyecto. En esta fase se analiza que elementos se esperan de la herramienta.

2. Fase de desarrollo de la herramienta. En esta fase se implementa y testea las funcionalidades que se han determinado en la primera parte.

3. Documentación, donde se explicará que trabajo se ha hecho y las modificaciones se han hecho durante el desarrollo respecto a la planificación inicial.

ID	Tarea	Inicio	Fin	Horas
1	Fase Inicial	05/09/11	25/11/11	230
1.1	Lectura de artículos y proyectos anteriores	05/09/11	19/09/11	30
1.2	Especificación	19/09/11	25/11/11	200
2	Fase de Desarrollo	28/11/11	06/04/12	260
2.1	Configuración del entorno de trabajo	02/12/11	09/12/11	20
2.2	Implementación	09/12/11	15/03/12	155
2.3	Prueba	08/12/11	22/03/12	60
2.4	Revisión del código	22/03/12	06/04/12	25
3	Fase de Documentación	27/12/11	16/05/12	110
3.1	Manual para el usuario	22/03/12	03/04/12	20
3.2	Memoria	27/12/11	16/05/12	90
Total de Horas				600

Tabla Tareas (Visión general)

Fase Inicial

ID	Tarea	Inicio	Fin	Horas
1	Fase Inicial	05/09/11	25/11/11	230
1.1	Lectura de artículos y proyectos anteriores	05/09/11	19/09/11	30
1.2	Especificación	19/09/11	25/11/11	200

Tabla Tareas (Fase inicial)

Esta fase comenzó el día 6/09/11 hasta 25/11/11. La duración de esta fase fue de 230 horas. La dificultad de compaginar el proyecto con el trabajo profesional fue la principal razón de la duración de esta fase. Esta fase se divide en 2 partes.

Lectura de artículos y proyectos anteriores

Esta fase consistió en la lectura de los artículos y proyectos anteriores relacionados con este proyecto. Tuvo una duración de 30 horas.

ID	Tarea	Inicio	Fin	Horas
1.1	Lectura de artículos y proyectos anteriores	05/09/11	19/09/11	30
1.1.1	GEDCOM Data Model	05/09/11	06/09/11	5
1.1.2	The GEDCOM Standard Release 5.5	06/09/11	07/09/11	5
1.1.3	Modelización de fuentes documentales civiles de interés genealógico -Pedro Lucas Generelo Perulero	07/09/11	13/09/11	5
1.1.4	Integració d'eines pel buidatge de fonts documentals d'interès genealògic"	13/09/11	15/09/11	10
1.1.5	Modelització de fonts documentals d'interès genealògic	15/09/11	19/09/11	5

Tabla Tareas (Fase incial)

Creación de documentación previa.

En esta fase se han realizado documentación que ayudarían al desarrollo del proyecto. Tuvo una duración de 200 horas.

Esta parte consistía en hacer una especificación de la herramienta, elementos tales como la lista de atributos que se consideran en la herramienta, el modelo conceptual utilizado, las reglas y el algoritmo. Una de las dificultades de este proyecto se encuentra en esta fase. Esta dificultad consiste en crear un modelo que guarde la información genealógica de una persona.

ID	Tarea	Inicio	Fin	Horas
1.2	Especificación	19/09/11	25/11/11	200
1.2.1	Lista de atributos	19/09/11	05/10/11	50
1.2.2	Modelo de base de datos	05/10/11	24/10/11	80
1.2.3	Reglas	24/10/11	03/11/11	40
1.2.4	Algoritmo	03/11/11	25/11/11	30

Tabla Tareas (Fase documentación)

Fase de desarrollo

La fase de desarrollo comenzó del 28/11/11 hasta el 06/04/12. Esta ha tenido una duración de 220 horas.

La dificultad de compaginar trabajo profesional con el proyecto ha sido la principal causa de la desviación. Ha habido días en que no se ha podido dedicar al proyecto a causa de trabajo profesional. Para que no se desviara tanto se ha hecho algunas tareas en paralelo como es la fase de pruebas, en vez de hacerlo una vez finalizada la fase de implementación.

ID	Tarea	Inicio	Fin	Horas
2	Fase de Desarrollo	28/11/11	06/04/12	260
2.1	Configuración del entorno de trabajo	02/12/11	09/12/11	20
2.2	Implementación	09/12/11	15/03/12	155
2.3	Prueba	08/12/11	22/03/12	60
2.4	Revisión del código	22/03/12	06/04/12	25

Tabla Tareas (Fase de desarrollo)

Configuración del entorno de trabajo

En esta fase se ha configurado el entorno de trabajo que se utilizo durante la implementación. La tarea más difícil fue la de instalar librerías que se requería en la fase de implementación. Ha tenido una duración de 20 horas.

ID	Tarea	Inicio	Fin	Horas
2.1	Configuración del entorno de trabajo	02/12/11	09/12/11	20
2.1.1	Configuración de Apache	02/12/11	06/12/11	5
2.1.2	Configuración de Netbeans	06/12/11	09/12/11	5
2.1.3	Configuración de PHP	06/12/11	09/12/11	5
2.1.4	Configuración de Base de datos MySQL	06/12/11	07/12/11	5

Tabla Tareas (Fase desarrollo)

Implementación

Esta fase consistió en la implementación del cogido necesario para la herramienta. Consistió principalmente en la implementación de la base de datos, las reglas y por último el algoritmo de calculo. La implementación ha tenido una duración de 125 horas.

ID	Tarea	Inicio	Fin	Horas
2.2	Implementación	09/12/11	15/03/12	155
2.2.1	Implementación de Base de Datos	09/12/11	16/12/11	25
2.2.2	Implementación de Reglas	16/12/11	06/01/12	40
2.2.3	Implementación de Algoritmo de Calculo	06/01/12	15/03/12	90

Tabla Tareas (Fase desarrollo)

Testeo del código

Esta fase se ha hecho paralelamente a la fase de desarrollo. Mientras se ha implementado una funcionalidad o una clase nueva, se ha ido testeando las clases que se han hecho. Ha ido muy bien hacer pruebas mientras se implementaba, porque de

esta manera siempre había la seguridad que el comportamiento de una funcionalidad es la esperada para un determinado caso.

La fase de testeo de código ha tenido una duración de 50 horas.

ID	Tarea	Inicio	Fin	Horas
2.3	Prueba	08/12/11	22/03/12	60
2.3.1	Test de Persistencia	09/12/11	14/12/11	10
2.3.2	Test Unitario	16/12/11	23/12/11	20
2.3.3	Test Funcional	06/01/12	27/01/12	10
2.3.4	Test Global	15/03/12	22/03/12	20

Tabla Tareas (Fase desarrollo)

Revisión del código

La revisión del código ha tenido una duración de 25 horas. Esta fase se ha hecho desde que ha comenzado la fase de implementación. De esta manera había que dedicar menos tiempo a revisar el código después de la fase de implementación.

En esta fase se ha hecho un revisión del código. Se ha documentado todas las clases que se han implementado y eliminado todo aquel código que finalmente no se utiliza dentro de la aplicación.

ID	Tarea	Inicio	Fin	Horas
2.4	Revisión del código	22/03/12	06/04/12	25
2.4.1	Documentación del código	22/03/12	06/04/12	10
2.4.2	Eliminación de código muerto	22/03/12	02/04/12	15

Tabla Tareas (Fase desarrollo)

Fase de documentación

Esta fase comenzó el 27/12/11 y se termino el día 16/05/12. Esta tarea se ha hecho en paralelo con la fase de implementación.

Durante el proyecto se ha ido documentando todas las tareas realizadas. Esto ha facilitado la elaboración de la memoria.

ID	Tarea	Inicio	Fin	Horas
3	Fase de Documentación	27/12/11	16/05/12	150
3.1	Manual para el usuario	22/03/12	03/04/12	20
3.2	Memoria	27/12/11	16/05/12	130

Tabla Tareas (Fase documentación)

Elaboración de un manual

En esta sub-fase se ha realizado un manual para explicar al usuario final la instalación de la herramienta.

Memoria

Esta sub-fase considera todas las tareas relacionadas con la redacción de la memoria. Se ha ido juntando la documentación realizada durante el proyecto para después poderlo redactar y unificar la información.

ID	Tarea	Inicio	Fin	Horas
3.2	Memoria	27/12/11	16/05/12	130
3.2.1	Estructuración de la Memoria	27/12/11	29/12/11	10
3.2.2	Elaboración de la Memoria	30/12/11	14/05/12	100
3.2.3	Revisión de la Memoria	14/05/12	16/05/12	20

Tabla Tareas (Fase documentación)

Valoración Económica

El análisis económico del proyecto tiene en cuenta todos los costes de la carga de trabajo, el material utilizado (en este caso software y hardware) y la gestión del proyecto.

Coste de la carga de trabajo

Concepto	Horas	Coste Unitario	Coste Total
Fase Inicial	230	25	5750
Fase de Desarrollo	260	25	6500
Fase de Documentación	110	25	2750
Total	600		15000

Tabla de Coste de la carga de trabajo

Coste del hardware y software utilizado

El conjunto de software utilizado fueron :

1. Netbeans
2. MAMP
3. ArgUML

El conjunto de hardware utilizado:

1. Portátil valorado en 1800 €

Únicamente se tiene costes por el hardware. El portátil no es exclusivo al proyecto. El portátil tiene aproximadamente 4 años de vida y el proyecto ha tenido una duración de 600 horas. Se amortiza 600€ cada año de los cuales 200€ van dirigidos al proyecto

Coste total del proyecto.

Concepto	Coste Total
Proyecto	15000
Coste Hardware	200
Total	15200

Tabla de Coste total del proyecto

Herramientas y tecnologías utilizadas

Para el desarrollo de la herramienta se han usado diversas herramientas y tecnologías. En este apartado se muestran cuales han sido usadas durante el proyecto.

ArgoUML

Para poder hacer la especificación y la documentación relativa a la herramienta se ha usado la aplicación ArgoUML. La principal razón de esta elección es, que es una herramienta gratuita.

Además tiene las funcionalidades que se requiere para hacer la especificación. Esta herramienta se ha usado para el diseño el modelo conceptual y explicar los casos de uso que tiene la herramienta.

Netbeans

Para la implementación se ha usado la herramienta NetBeans. Al igual que ArgoUML, se ha elegido Netbeans porque, es una herramienta gratuita.

Otra razón fue porque permite el uso de varios lenguajes de programación, de las cuales están las que se necesitan PHP, CSS y HTML principalmente. También tiene funcionalidades que hacen de la programación una tarea más fácil. Un ejemplo son los autocompletados o las correcciones de sintaxis mientras se esta escribiendo el código.

MAMP

Como entorno de trabajo se ha usado el paquete MAMP.

Este paquete incluye PHP, MySQL y Apache. He escogido este paquete porque ha facilitado la tarea de configurar un entorno de trabajo, es gratuito y además tiene los elementos que se necesitan: Un servidor , apache; una base de datos, MySQL y también PHP.

MySQL

MySQL (http://www.mysql.com) es un sistema e gestión de base de datos relacional, multihilo y multiusuario. Esta disponible para plataformas UNIX, Microsoft Windows y Macintosh, entre otras. Su licencia es GNU GPL. En caso de uso privativo, es necesario la compra e una licencia específica.

Muchas grandes empresas como Google, Adobe, Facebook o Alcatel lo usan como base de datos para ahorrar tiempo y dinero en sus paginas webs y en sus procesos de negocio.

PHP

PHP (http://www.php.net) es un lenguaje de programación interpretado. Se ha convertido en una tecnología muy usada especialmente en la web aunque también se puede utilizar para crear aplicaciones de escritorio. Este lenguaje se puede usar con otras tecnologías como es HTML y CSS. Necesita un servidor que lo interprete para su uso en páginas webs.

La razón del porque se ha utilizado PHP, es porque facilita la creación de una interfaz grafica para el usuario final. También incluye funcionalidades para el uso del MySQL.

CSS

Es un lenguaje que permite definir la presentación de un documento estructurado escrito en HTML o XML. Al igual que HTML, su especificación es responsabilidad del Consorcio del World Wide Web.

HTML

Es un lenguaje de marcad utilizado par la creación de páginas webs. Es usado para describir su estructura y contenido en formato texto, así como para complementar el texto con imágenes, videos, vínculos, etc.

GEDCOM

GEDCOM es un formato de archivo de datos. Es un acrónimo de **GE**nealogical **D**ata **COM**munication (comunicación de datos genealógicos).
Estos archivos contienen información genealógica de individuos y metadatos que enlazan esos datos. Es comúnmente utilizado en la genealogía . Por esta razón es interesante que la herramienta lo tenga en cuenta.

GEDCOM esta formado por registros que representan, entre otros elementos, familias e individuos, información que necesita la herramienta. Cada registro se compone por varias líneas que contiene una abreviación y un valor.

```
0 HEAD
1 SOUR Reunion
2 VERS V8.0
2 CORP Leister Productions
1 DEST Reunion
1 DATE 11 FEB 2006
1 FILE test
1 GEDC
2 VERS 5.5
1 CHAR MACINTOSH
0 @I1@ INDI
1 NAME Bob /Cox/
1 SEX M
1 FAMS @F1@
1 CHAN
2 DATE 11 FEB 2006
```

Ejemplo de Archivo GEDCOM

Muchos programas permiten exportar de e importar a un formato GEDCOM. También hay webs donde se pueden encontrar información de interés genealógico. Usan este tipo de

archivo para exportar o importar información. Un ejemplo es la web de www.familysearch.com o http://www.ancestry.com. Estas webs se han usado como fuente de información para obtener datos de personas.

Webs que usan GEDCOM

Especificación

La fase de especificación es imprescindible para que un proyecto resulte exitoso. Por ello, se proporcionan diferentes modelos y diagramas para realizarla con el máximo de detalle posible. En algunos casos, dos o más tipos de diagramas muestran el mismo concepto de manera diferente, pudiendo llegar a ser, en ocasiones redundantes. Para evitar dicha redundancia, en este proyecto no se detallan los diagramas de secuencia para la especificación, pero si durante el diseño, porque el curso típico de acontecimiento de cada caso de uso se encuentra detallado en la descripción del propio caso de usos.

Elementos a considerar

En este apartado se comentan los aspectos que se han tenido en cuenta a la hora de hacer la especificación.

Gedcom

La parte más importante del modelo conceptual son los datos de una persona. Esta información está repartido entre diversas clases dentro del modelo conceptual. Para que el modelo se adapte al formato de GEDCOM, se separan el nombre de un atributo y el valor que tiene, como sucede en un archivo GEDCOM.

Por ejemplo, si se tiene dentro de la base de datos una persona con apellido López, en el modelo se dividirá en 2 instancias: una instancia donde se guarde el valor apellido y otra instancia donde se guarda el valor Lopez. De esta manera con un modelo GEDCOM se puede extraer y guardar la información fácilmente.

Atributos que usaremos de una persona.

Los atributos de una persona que se usan en la herramienta para hacer los cálculos son las siguientes:

apellido1
apellido2
dirrecion_domicilio
dirrecion_padron
edad
estado_civil
estado_vital
lugar_fallecido
fecha_bautismo
fecha_boda
fecha_censo
fecha_confirmacion
fecha_defuncion
fecha_enterramiento
fecha_nacimiento
fecha_obit
lugar_boda
lugar_enterramiento
lugar_nacimiento
nacionalidad
natural
nombre
oficio
relacion
sexo

Tabla de atributos utilizados

Se han escogido estos atributos porque determinan bastante a una persona y también son los atributos que salen en la mayoría de las fuentes de documentación de interés genealógico. No se descarta la posibilidad de añadir más atributos a considerar más adelante.

A continuación se muestra una tabla donde se describen cada uno de los atributos, indicando a que se refiere cada uno de los atributos.

Nombre del atributo	Descripción
apellido1	Es el primer apellido de una persona
apellido2	Representa el segundo apellido de una persona
dirrecion_domicilio	Dirección de la vivienda de una persona
dirrecion_padron	Dirección del padrón.
edad	Edad de una persona
estado_civil	El estado civil de una persona en un momento determinado
estado_vital	Estado vital de una persona en un momento determinado
lugar_fallecido	Lugar donde ha fallecido una persona
fecha_bautismo	Fecha de bautismo de una persona
fecha_boda	Fecha de bodas de una persona
fecha_censo	Fecha de censo de una persona
fecha_confirmacion	Fecha en el que se confirmo una persona

fecha_defuncion	Fecha en que una persona falleció
fecha_enterramiento	Fecha en que una persona fue enterrada
fecha_nacimiento	Fecha de nacimiento de una persona
fecha_obit	Fecha del obituario de una persona
lugar_boda	Lugar en que se celebro la boda de una persona
lugar_enterramiento	Lugar de enterramiento de una persona
lugar_nacimiento	Lugar donde nació una persona
nacionalidad	Nacionalidad de una persona
natural	Lugar de procedencia de una persona
nombre	Nombre de una persona
oficio	Trabajo de una persona
relacion	Relaciones familiares o de matrimonio que tiene una persona
sexo	Sexo de una persona

Tabla descriptiva de los atributos utilizados

De estos atributos hay algunos que solo pueden admitir un conjunto de valores. Estos atributos son estado_civil, estado_vital, relación y sexo.

Nombre	Valores posibles
estado_civil	soltero, casado, viudo, divorciado
estado_vital	vivo, muerto
relacion	padre_madre, hijo_hija, hermano_hermana, pareja
sexo	hombre , mujer

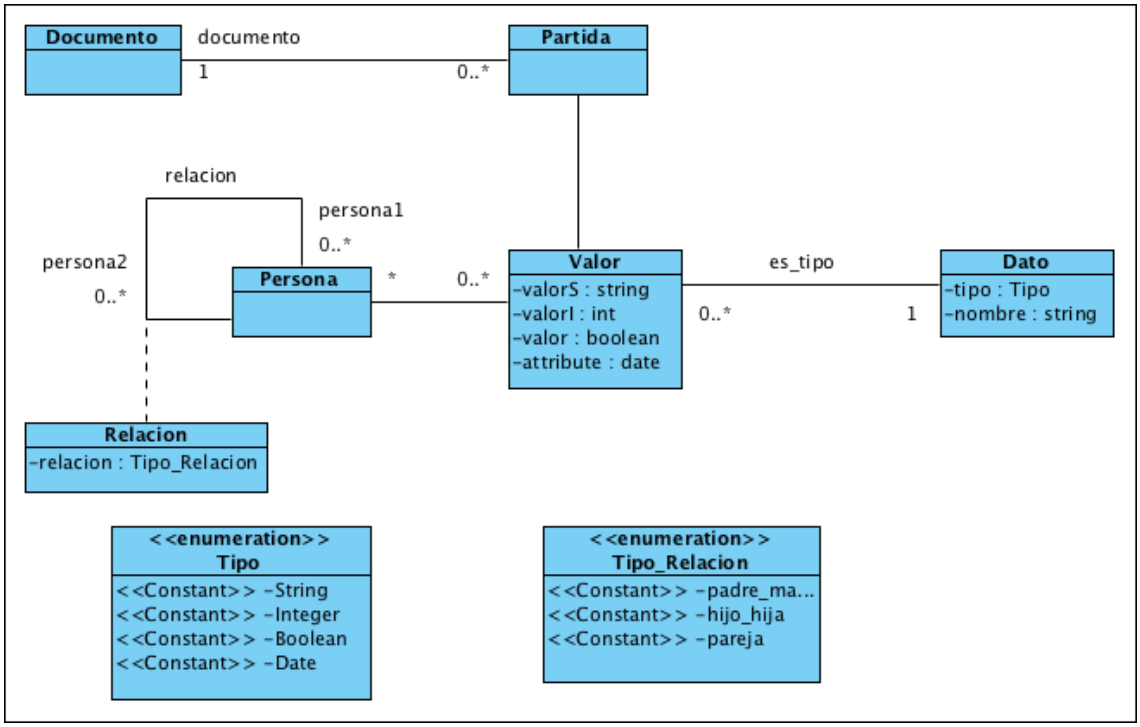
Valores posibles de ciertos atributos

Modelo Conceptual

Durante el proyecto se ha analizado diferentes modelos conceptuales para utilizar.
La característica principal que se esperaba del modelo a usar era la simplicidad. Esto facilita las consultas que hace la herramienta.
Otra característica que se requiere es que el modelo se pudiera ajustar al formato de los archivos GEDCOM[1]. Como se ha dicho anteriormente este formato es muy utilizado y es interesante tenerlo en cuenta.

A continuación se mostrarán las diferentes alternativas que se han analizado:

Primera Alternativa



Primera Alternativa de Modelo Conceptual

[1] GEDCOM es un formato de archivo de datos. Es un acrónimo de **GE**nealogical **D**ata **COM**munication (comunicación de datos genealógicos).

Restricciones de integridad

Regla 1:
Una instancia de Dato solo puede tener un atributo con valor no nulo.

Regla 2:
Una instancia de dato con tipo igual a String solo puede estar relacionado con instancias de Valor que tengan valorS diferente de nulo.

Regla 3:
Una instancia de dato con tipo igual a Integer solo puede estar relacionado con instancias de Atributos que tengan valorI diferente de nulo.

Regla 4:
Una instancia de dato con tipo igual a Boolean solo puede estar relacionado con instancias de Atributos que tengan valorB diferente de nulo.

Regla 5:
Una instancia de dato con tipo igual a Date solo puede estar relacionado con instancias de Atributos que tengan valorD diferente de nulo.

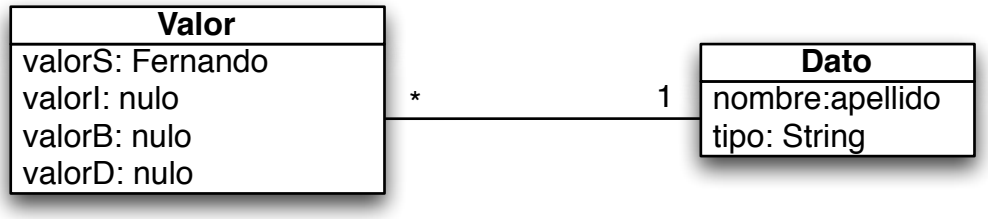
Regla 6:
Una persona no puede estar relacionada consigo mismo.

Explicación del modelo

Este modelo cumple con los requisitos que tenemos. Guarda los datos que necesitamos de una persona de forma simple y se adapta a la estructura de un archivo GEDCOM.

La clase Valor se encarga de guardar el valor de un atributo. La clase Dato representa los diferentes atributos que puede tener una persona como es el nombre, apellido1 o apellido2.

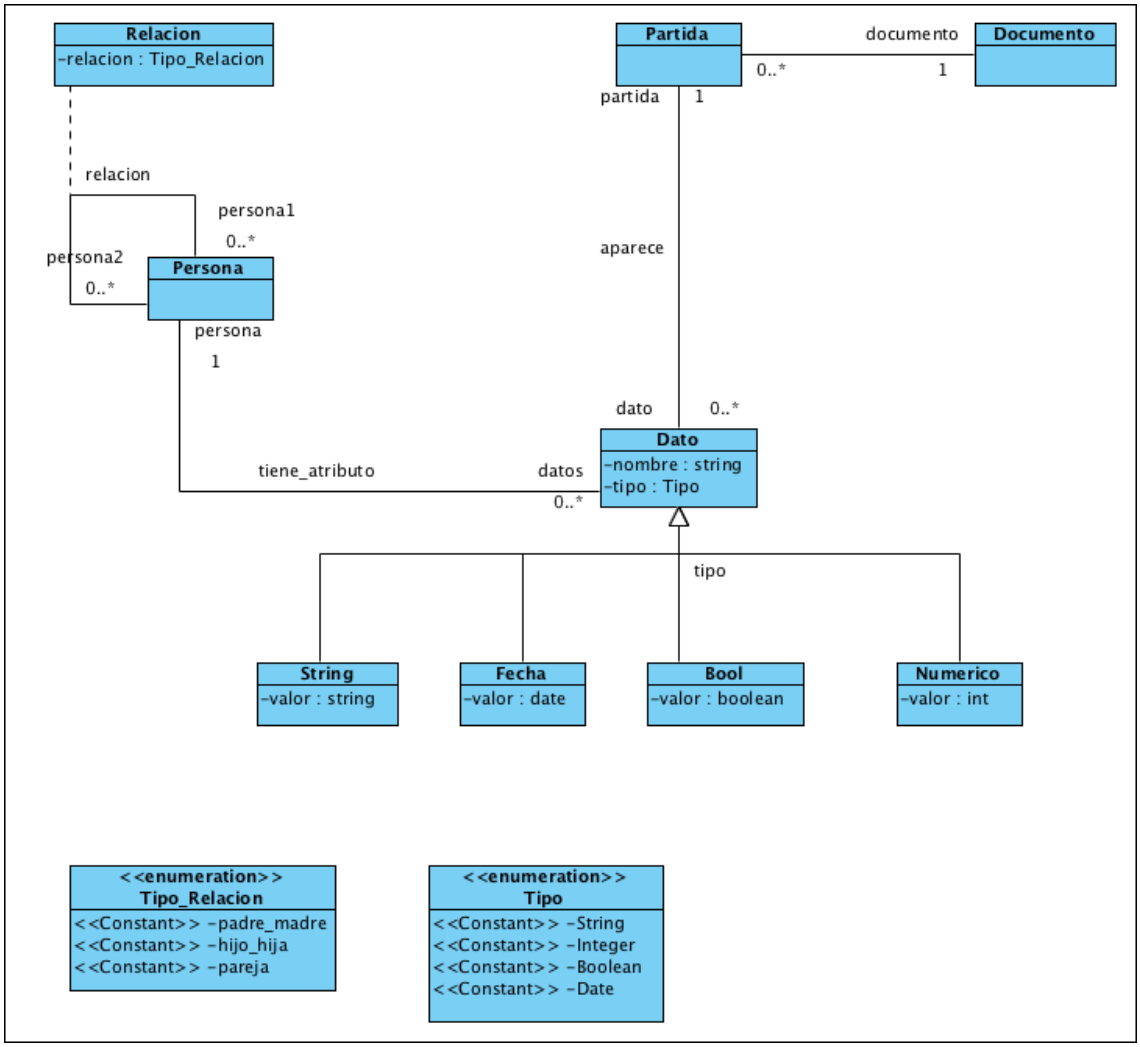
El problema que tiene este modelo es que en la clase Valor solo se usa una de los atributos para indicar el valor de un dato. Por ejemplo, en el caso que haya una persona de nombre Fernando, en la base de datos se instancia de la siguiente manera:



Ejemplo de instancia

Se observa que en la instancia de Valor todos los atributos menos valorS son igual a nulo. Esto hace que se ocupe espacio innecesario y para obtener el valor de un atributo se tiene que mirar el tipo, en vez de coger directamente de la relación.

Segunda Alternativa



Segunda Alternativa : Modelo Conceptual

Restricciones de integridad

Regla 1 :

Una misma persona no puede estar relacionado consigo mismo.

Regla 2 :

La subclase de dato se determinar por el valor tipo.

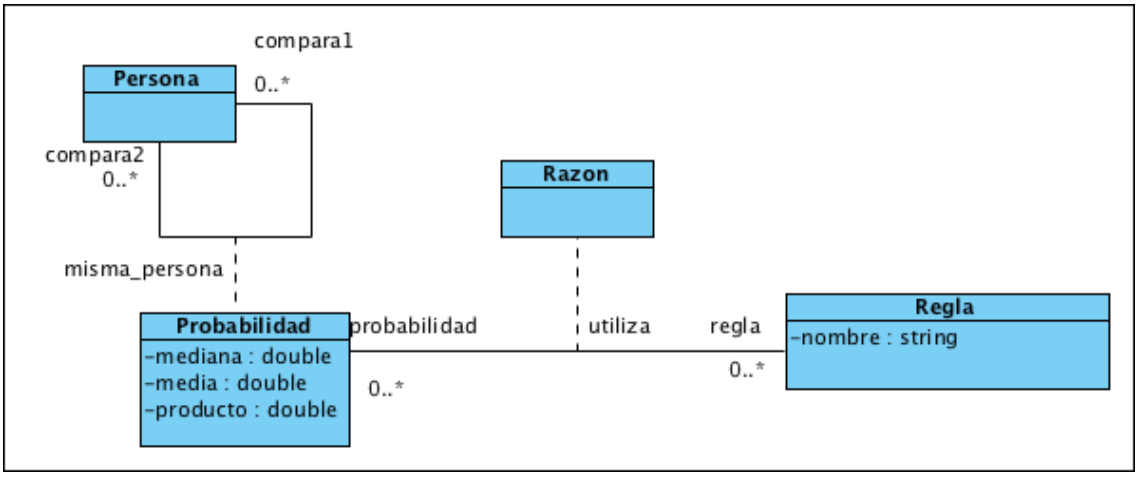
Explicación del modelo

Para solucionar el problema que tiene el modelo anterior, se ha optado por utilizar una jerarquía en la clase Atributo. De esta forma, dependiendo del tipo de dato que se tenga, se crea una instancia de una de las subclases.
Se tiene la seguridad que los atributos que contengan el valor de un atributo de una persona, no será nulo. Además se puede conseguir más fácilmente el valor de un atributo.

Una de las inconveniencias que tiene este modelo es que la jerarquía hace que el modelo sea más complicado. Sin embargo la jerarquía evita comprobar en todo momento que uno de los valores no sea nulo. Por lo tanto se decidió usar este modelo.

Modelo conceptual para el cálculo de probabilidades

Para la aplicación también se necesita clases que guarden la información que proporciona la herramienta después de hacer los cálculos.



Clases encargadas de guardar la información de los cálculos

Restricciones de integridad

Regla 1 :
Una misma persona no puede estar relacionado con ella misma .

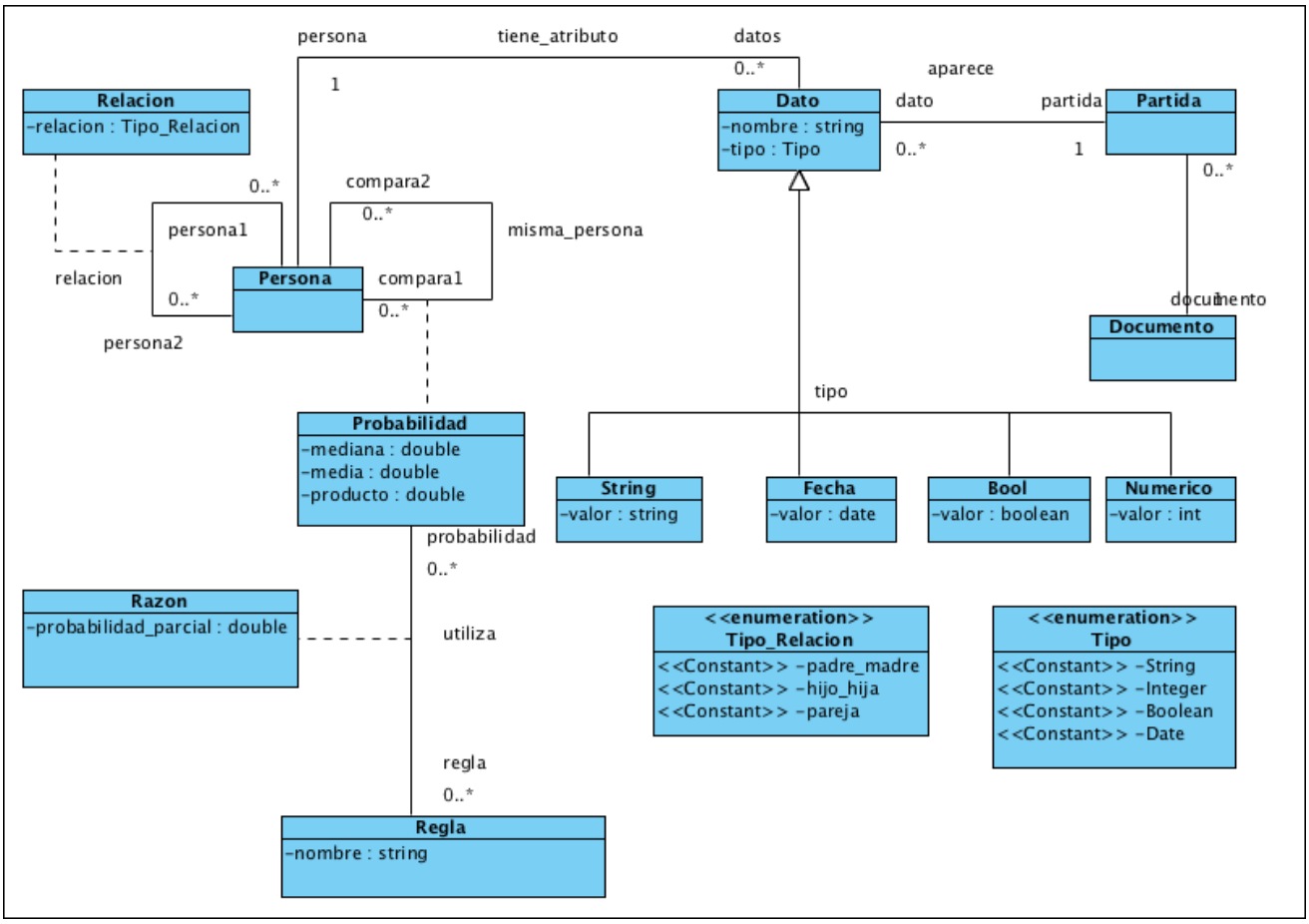
Explicación del modelo

Este modelo representa los datos que se obtiene después hacer los cálculos. La clase probabilidad contiene la probabilidad.

Las probabilidades parciales obtenidas de las reglas se guardan en la clase razón. De esta manera tendremos un registro de las probabilidades parciales obtenidas para poder analizar mejor los resultados calculados.

Modelo Conceptual Final

A continuación se muestra el modelo final, añadiendo las clases necesarias para guardar toda la información que la aplicación guarda una vez se ha hecho un calculo del grado de similitud que hay entre dos personas.



Modelo Final

Restricciones de integridad

Regla 1 :

Una misma persona no puede estar relacionado consigo misma.

Regla 2 :

La subclase de dato se determinar por el valor tipo

Regla 3:

Dos reglas no pueden tener el mismo nombre.

Regla 4:

Dos personas solo pueden tener un tipo de relación.

Reglas

Un de los elementos más importante dentro de la herramienta son las reglas. Estas se encargan de calcular la probabilidad de que dos personas sean la misma teniendo en cuenta únicamente algunos atributos.

A continuación se explica la especificación de las reglas. En los siguientes apartados, para simplificar la documentación, no se vuelven a explicar las restricciones.

Clase Regla

Las reglas están representadas por la clase Regla
La clase regla tiene un método que se encarga de analizar los atributos de estas dos para devolver una probabilidad.

Hay 3 tipos de valores posibles dependiendo de los casos que se den:

1. Probabilidad
   Este valor indica la probabilidad de que dos personas sean la misma teniendo en cuenta los atributos que utiliza.
2. Indeterminación
   Un valor que indique indeterminación. Esto sucede cuando no se puede determinar la probabilidad ya que faltan datos.
3. Imposibilidad
   Indica que la imposibilidad de que dos personas sean la misma.

En el sección de diseño se explica como se representan estos valores.

Casos de uso

Diagrama de caso de uso

Visión Global

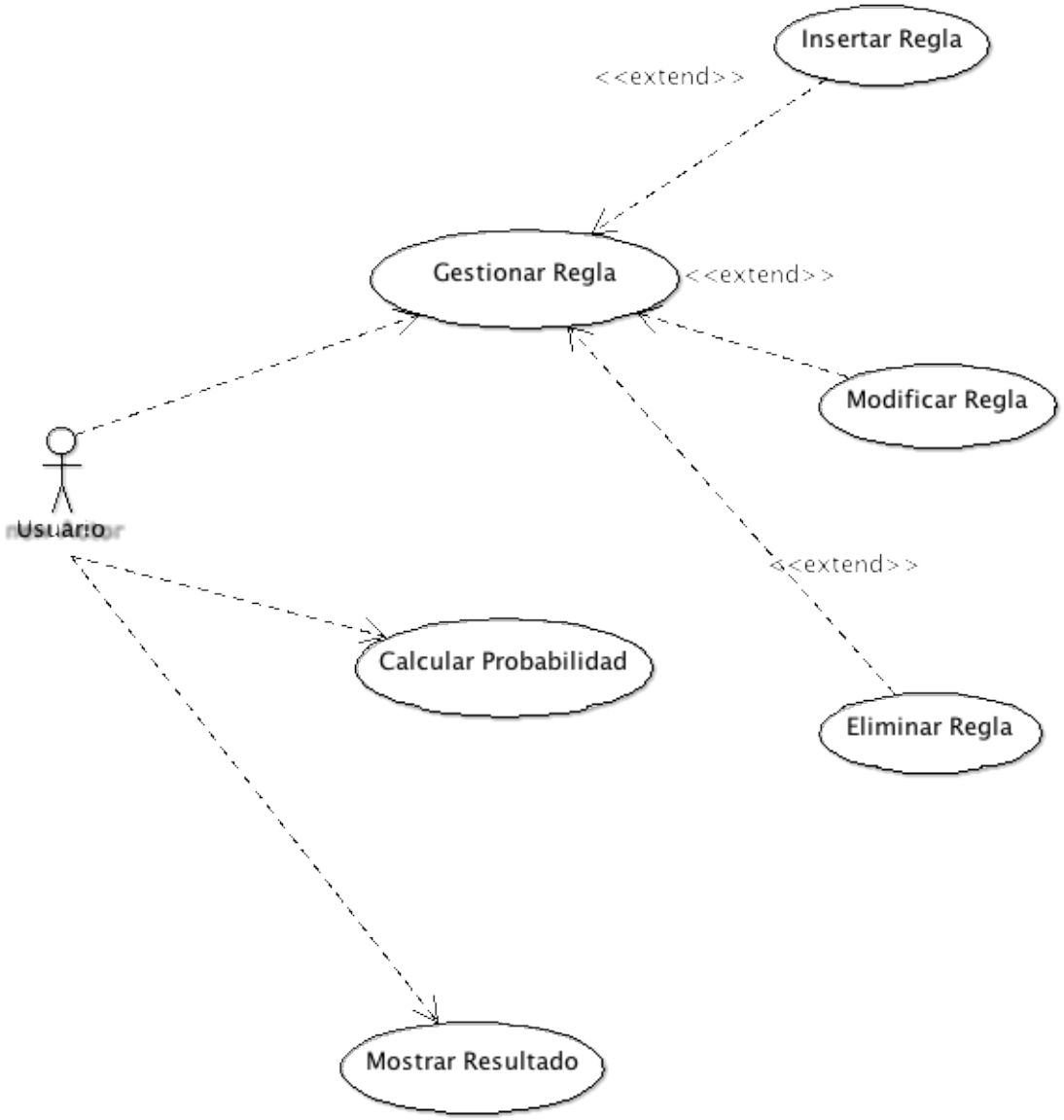


Diagrama de Casos de Uso

Un actor de un caso de uso es una entidad externa al sistema que participa en alguno de los escenarios de los casos de uso. De momento este actor será un genealogista. Por esta

razón, por temas de simplificación y claridad, en la especificación de los casos de uso se indica como usuario.

Especificación de los casos de uso

GESTIONAR REGLA
Actor Principal
Usuario
Precondición
-
Descripción
El actor quiere ver las reglas que existen en el sistema
Escenario Principal:
1. El sistema muestra una lista de todas las reglas que están en el sistema 2. El sistema da la opción de insertar una nueva regla o seleccionar una para eliminarla o modificarla.
Extensión
2a. El usuario quiere insertar una nueva regla. 2a1. <include insertar regla >. 2b. El usuario quiere modificar una regla. 2b1. <include modificar regla >. 2c. El usuario quiere eliminar una regla. 2c1. <include eliminar regla >.

Descripción del caso de uso Gestionar Regla

INSERTAR REGLA
Actor Principal
Usuario
Precondición
-
Descripción
El actor quiere insertar una nueva regla en el sistema.
Escenario Principal:
1. El usuario inserta los datos de la regla. 2. El sistema comprueba que los datos sean correctos. 3. El sistema muestra mensaje confirmando que la inserción ha sido correcta
Extensión
2a. Los datos son incorrectos. 2a1. El sistema muestra un mensaje de error.

Descripción del caso de uso Insertar Regla

MODIFICAR REGLA
Actor Principal
Usuario
Precondición
Existe en el sistema la regla que se quiere modificar.
Descripción
El actor quiere insertar una nueva regla en el sistema.
Escenario Principal:
1. El usuario inserta los datos de la regla a modificar 2. El sistema comprueba que los datos sean correctos. 3. El sistema muestra mensaje confirmando que la modificación ha sido correcta
Extensión
2a. Los datos son incorrectos. 2a1. El sistema muestra un mensaje de error.

Descripción del caso de uso Modificar Regla

ELIMINAR REGLA
Actor Principal
Usuario
Precondición
En el sistema existe una instancia de la regla que se quiere eliminar.
Descripción
El actor quiere eliminar una regla del sistema.
Escenario Principal:
1. El usuario selecciona la regla a eliminar 2. El sistema muestra mensaje confirmando que la eliminación ha sido correcta.
Extensión
-

Descripción del caso de uso Eliminar Regla

CALCULAR PROBABILIDAD
Actor Principal
Usuario
Precondición
-
Descripción
El actor quiere un calculo de probabilidad.
Escenario Principal:
1. El usuario introduce los parámetros que usará el algoritmo: el atributo con que se formarán las parejas y las reglas que se usarán. 2. El sistema inicia el calculo 3. El sistema muestra mensaje de que el calculo se ha hecho.
Extensión
2a. El usuario quiere ver los resultados. 2a1. <include Mostrar Resultados >.

Descripción del caso de uso Calcular Probabilidad

MOSTRAR RESULTADOS
Actor Principal
Usuario
Precondición
El sistema ha calculado previamente las probabilidades.
Descripción
El actor quiere ver los resultados del calculo de probabilidades.
Escenario Principal:
1. El sistema muestra una lista de parejas con su probabilidad global 2. El usuario clica en una pareja para ver las probabilidades calculadas por cada regla. 3. El sistema muestra una lista de las probabilidades parciales calculadas.
Extensión
-

Descripción del caso de uso Mostrar Resultados

Diseño

Hasta este momento se ha descrito qué hace la aplicación, pero no cómo lo hace.

Durante fase de diseño se han aplicado diferentes técnicas y principios que definen el sistema con los detalles necesarios para poder implementarlo. A partir del diseño se ha determinado la arquitectura, describiendo los subsistemas y componentes del sistema software y sus relaciones.

A continuación se describen los elementos más importantes relacionados con el diseño.

Arquitectura física del sistema

Para este proyecto, solo se necesita un equipo que funcione como servidor web. En la fase de desarrollo se ha utilizado un ordenador personal con un servidor web instalado donde también se aloja la base de datos.

El entorno de desarrollo consiste en un ordenador Intel Core i5 con 4Gb de RAM con Mac OS X 10.7.3 como sistema operativo. Apache como servidor web y MySQL como servidor de base de datos.

Desarrollo desde cero utilizando un framework

La aplicación se ha desarrollado desde cero pero con el uso de un framework.

El desarrollo desde cero consiste en construir la herramienta sin utilizar código existente. Esta opción tiene ventajas y desventajas:

Por una parte tiene la ventaja que permite una adaptación totalmente fiel al análisis y diseño realizado inicialmente. Además la curva de aprendizaje lo determina únicamente el lenguaje de programación utilizado. También tiene la ventaja de que no obliga seguir una nomenclatura ni convención.

En contra, tiene la desventaja que, desarrollar desde cero, implica una mayor inversión de tiempo para el desarrollo. Hay que implementar funcionalidades comunes a la mayoría de los proyectos. Esta opción es viable para proyectos pequeños, pero en un proyectos medianos o grandes no.

Por esta razón se combina el desarrollo desde cero utilizando un framework.

Los frameworks son programes o utilidades que tienen como propósito facilitar el desarrollo de proyectos. Suelen proporcionar una estructura de código y permiten organizar el desarrollo de manera clara y lógica. También ofrecen mecanismos para obtener funcionalidades básicas.

Tiene el principal inconveniente de imponer una estructura de programación y unos patrones de diseño concretos. Esto hace que haya diferencias con la análisis. Además de la curva de aprendizaje del lenguaje de programación, se le añade el aprendizaje del propio framework.

Utilizando un framework se consigue una programación más estructurada y no se invierte tanto tiempo en el desarrollo del proyecto. Se reutiliza código que se ha implementado previamente, una practica que se da actualmente en la empresas.

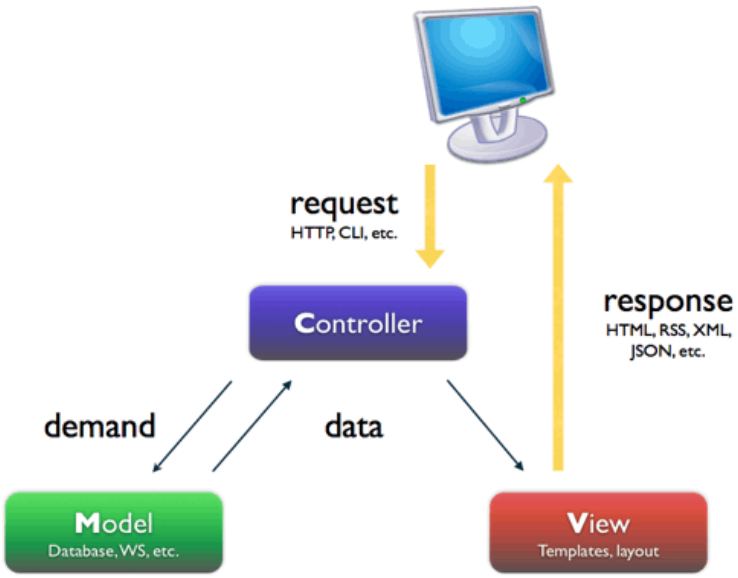
Arquitectura lógica del sistema

En este apartado se detallan los patrones y principios que se utilizan a la hora de diseñar el sistema como las tecnologías usadas, realizando una breve descripción de cada una. También se muestra y describe el diseño de cada una de las capas.

Arquitectura basado en el Patrón MVC (Modelo Vista Controlador)

Al utilizar un framework para el proyecto, se tiene que adaptar el análisis hecho inicialmente a causa de la estructura que obliga el framework. El framework elegido utiliza como base una arquitectura basada en el patrón modelo-vista-controlador (a partir de ahora patrón MVC).

Al Aplicar el patrón se divide la interacción entre el usuario y aplicación entre roles: modelo, vista y controlador.
La vista representa el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. El controlador responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y a la vista. El modelo es responsable de la lógica de negocio y del sistema de gestión de los datos.



Esquema del patrón MVC

El ciclo de interacción entre los tres componentes suelen ser la siguiente:

- El usuario provoca un evento
- El controlador recoge el evento, identifica el modelo a utilizar y ejecuta una acción sobre él.
- El modelo realiza la acción y devuelve el control al controlador.
- El controlador indica a la vista los datos a mostrar y delega en ella la presentación.
- La vista es responsable de mostrar los datos enviados por el controlador al usuario.
- El usuario retoma el control.

En el patrón MVC, el **controlador** puede únicamente comunicarse con **el modelo** y la **vista** y hace de enlace entre estas dos. Tiene la ventaja que cada parte es independiente. En el caso que se requiera hacer un cambio en alguna, esto no afecta a las otras.

Orientación a objeto

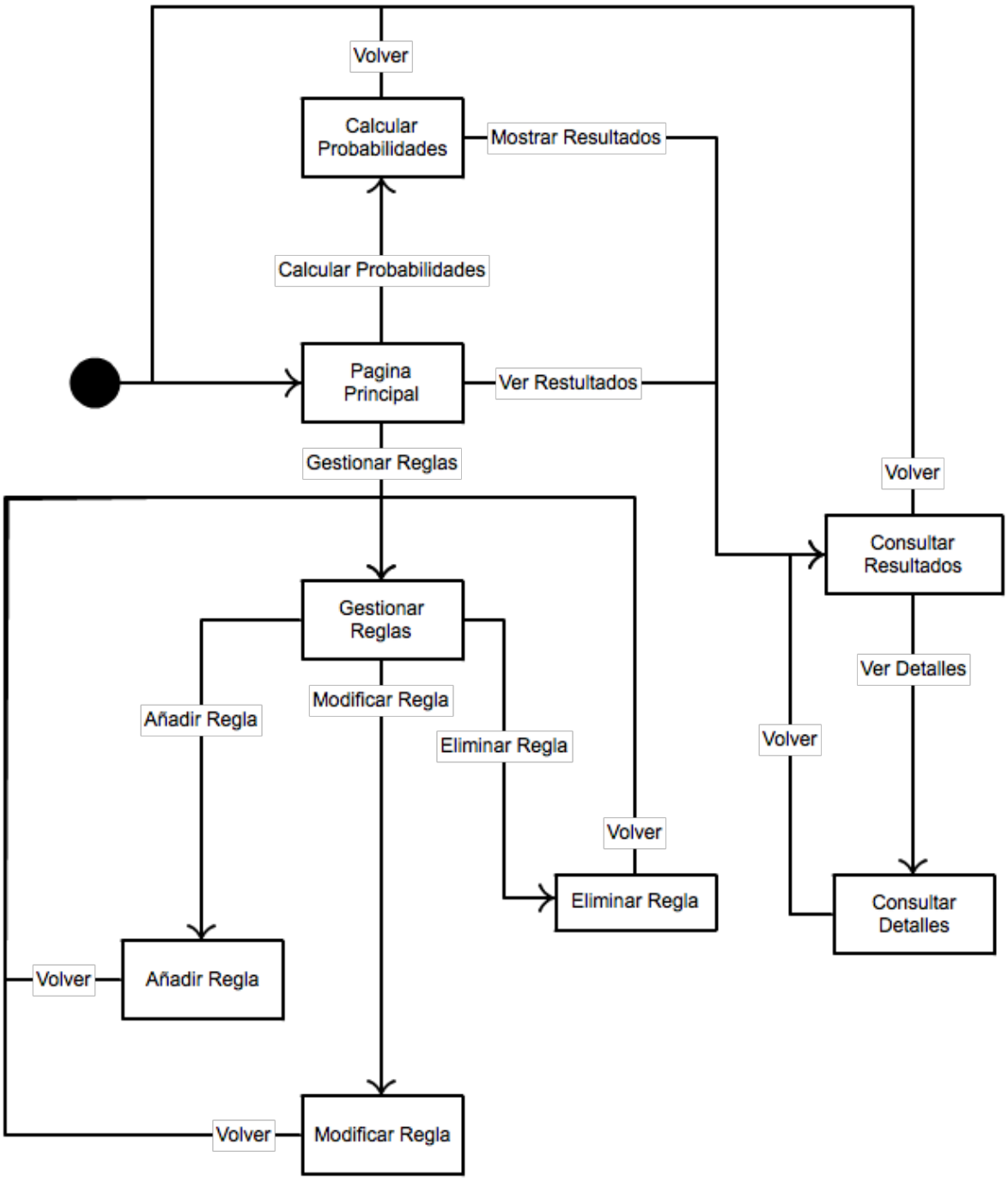
Se aplica este principio en el modelo y en el controlador. Aplicando este principio en el diseño, se obtiene un sistema software donde cada elemento que se ha descrito en la especificación pasa a ser un objeto. Cada uno tiene sus propias operaciones para manipular la información e interactúan entre ellos para cumplir las operaciones del sistema.

Vista

La vista se encarga de la presentación o interface de usuario.

Mapa de Navegación

Un mapa de navegación representa los diferentes caminos de navegación entre las diferentes pantallas de una aplicación. El mapa permite una visión clara y general sobre el funcionamiento de la interfaz y la interacción entre las diversas pantallas que forman la aplicación.



Mapa de Navegación

A continuación se muestra una tabla donde se puede ver que pantalla está asociada a un caso de uso.

Caso de Uso	Nombre de la Pantalla
Gestionar Reglas	Gestionar Reglas
Eliminar Regla	Eliminar Reglas
Añadir Regla	Añadir Regla
Modificar Regla	Modificar Regla
Calcular Probabilidad	Calcular Probabilidad
Mostrar Resultados	Consultar Resultados Ver detalles

Relación entre caso de uso y pantalla

Diseño Externo

El diseño externo define la interacción del usuario con el sistema software, en este caso la aplicación desarrollada. En esta sección se explicará el diseño externo que tiene la herramienta, mostrando la interacción del usuario con cada una de las pantallas que forman la aplicación.

Principios de Diseños

Como el objetivo principal es crear una herramienta, se ha dejado el aspecto gráfico en segundo plano. No por ello se ha dejado de utilizar algunos principios a la hora de diseñar la interfaz gráfica de las pantallas. El uso de estos principios tiene los siguientes objetivos.

1. Informar al usuario del estado del sistema .
2. Consistencia y estandarización.
3. Prevención de Errores.
4. Reconocimiento de acciones y opciones.
5. Flexibilidad y eficiencia en el uso.
6. Reconocimiento de errores, diagnóstico y recuperación.

Estos objetivos se reflejan en los siguientes.

El usuario puede saber dónde se encuentra, qué esta pasando, a dónde puede ir y qué puede hacer. En cada una de las pantallas hay un título que informa dónde esta el usuario

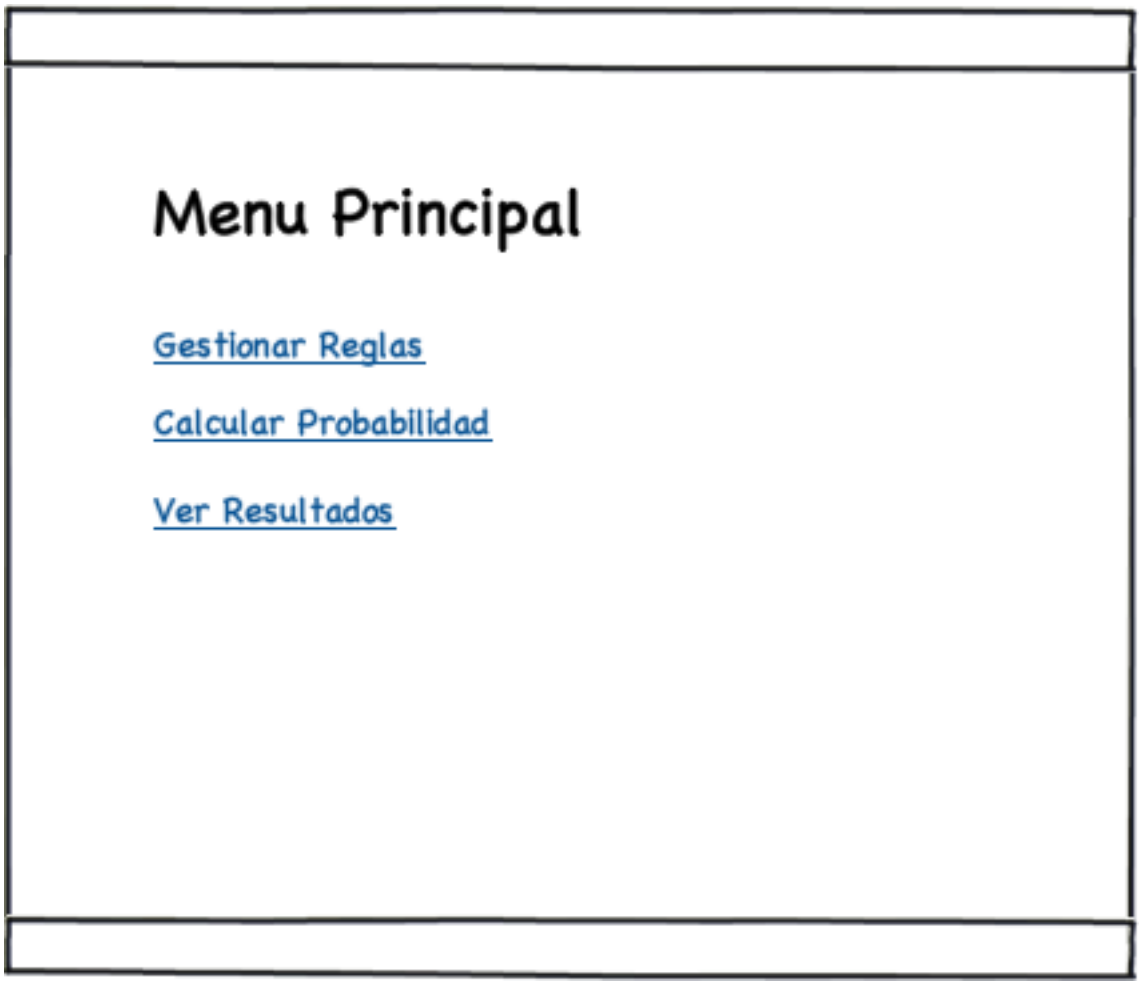
y qué esta pasando. También se han utilizado enlaces y botones que indican qué puede hacer y a dónde puede ir.

Para seguir la consistencia y estandarización se han utilizado botones y enlaces. Elementos como el checkbox o la lista desplegable son elementos que prevén errores del usuario. Se han utilizado tablas o listas para mostrar información.
De esta manera el usuario dispone la información de forma ordenada y clara. Los textos y el fondo tiene color que facilita la legibilidad de la información.

Los botones y enlaces informan de las acciones y opciones que dan. Cada botón tiene escrita un verbo para que el usuario pueda deducir el uso de cada uno de los botones o enlaces.
Cuando el usuario quiere cancelar una acción siempre se tiene la opción de volver al menú principal o en algunas acciones se le pide una confirmación.
En el caso de error se muestra en la pantalla un mensaje informando del error, las causas y como se puede resolver el error.

A continuación se muestran cada una de las pantallas utilizadas en el sistema.

Pantalla Principal



Pantalla del Menu Principal

Funcionalidad

Permite al usuario elegir entre las diferentes funcionalidades de la aplicación. Estas funcionalidades son la de gestión de reglas, calcular probabilidades y mostrar los resultados obtenidos anteriormente.

Proceso

1. El usuario clica uno de los links
2. El sistema comienza con las funcionalidades asociado al link elegido.

Botones/Enlaces

Gestionar Reglas: Lanza el caso de uso Gestionar Regla

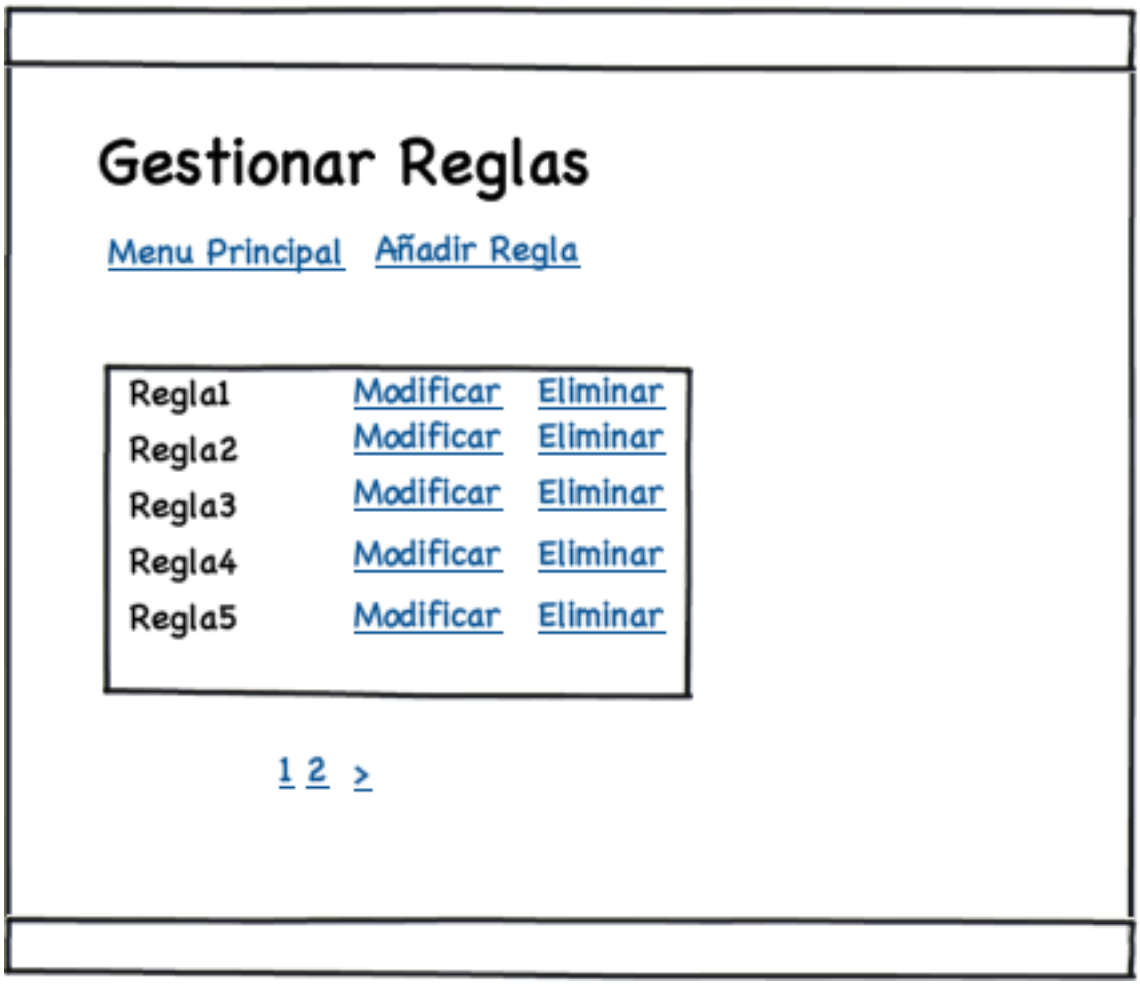
Calcular Probabilidad: Lanza el caso de uso Calcular Probabilidad

Ver Resultados: Lanza el caso de uso Mostrar Resultado

Posibles Errores

No hay errores posibles

Gestionar Reglas



Pantalla de Gestionar Reglas

Funcionalidad

La herramienta muestra todas las reglas existentes en el sistema y da la posibilidad al usuario de añadir, eliminar o modificar una regla. Las reglas están paginadas.

Proceso

1. El sistema muestra una lista de las reglas existentes en el sistema.
2. El sistema da la opción de añadir , eliminar modificar una regla o ver más reglas.
3. El usuario elige una opción.
4. El sistema ejecuta el caso de uso seleccionado para la regla escogida.

Botones/Enlaces

Eliminar: Lanza el caso de uso de eliminar regla.

Modificar: El sistema comienza el caso de uso modificar regla.

Añadir : El sistema comienza el caso de uso de añadir una regla.

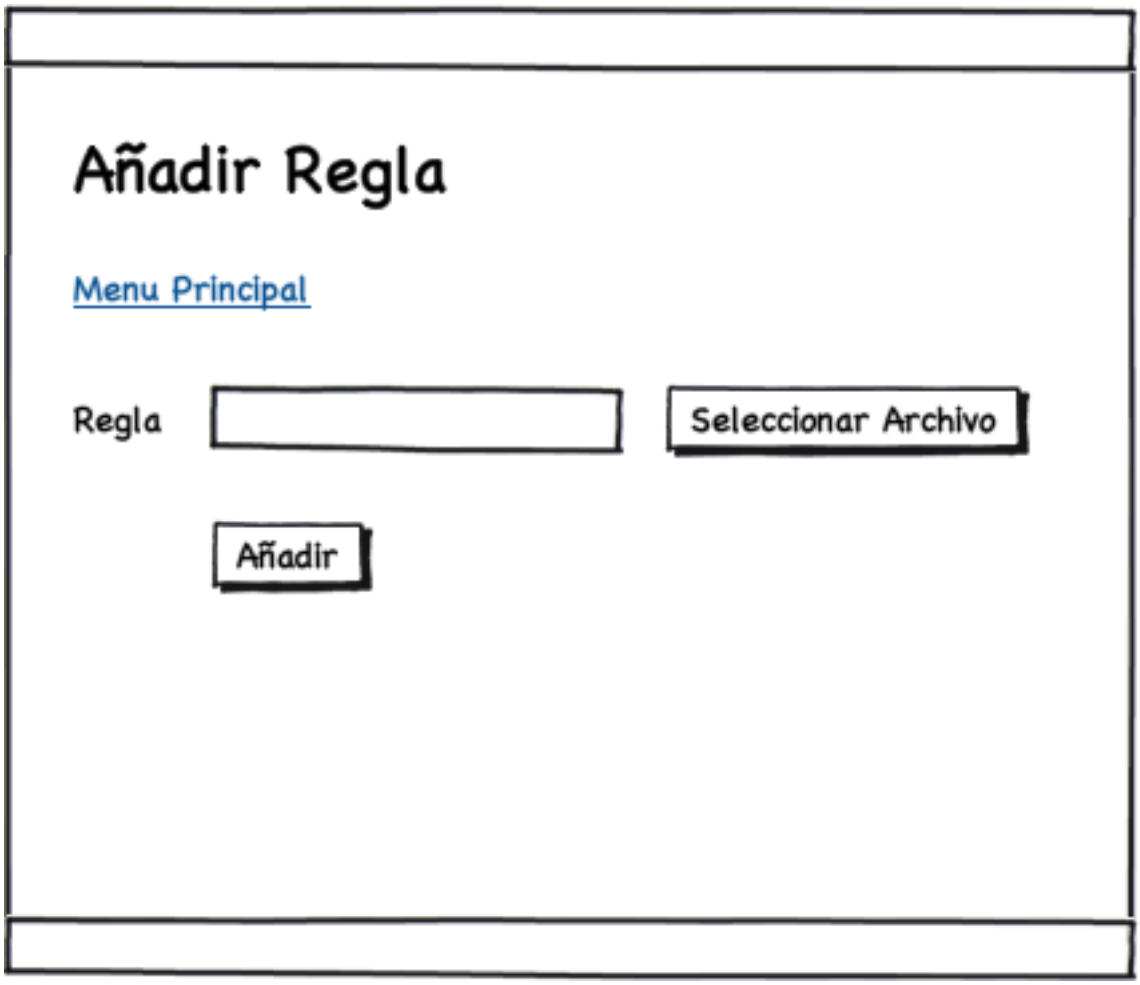
Menú Principal: El sistema vuelve al menú inicial

Números: Muestra más reglas (cada número representa una página)

Posibles Errores

No hay errores posibles.

Añadir Reglas



Pantalla de Añadir Regla

Funcionalidad

Añade una nueva regla en el sistema.

Proceso

1. El usuario indica el archivo donde se encuentra la regla
2. El usuario clica el botón añadir.
3. El sistema verifica el archivo.
4. El sistema añade en el sistema la nueva regla.

Botones/Enlaces

Seleccionar Archivo: Selecciona el archivo donde es la implementación de la regla a añadir.

Añadir: El sistema guarda el nombre de la regla.

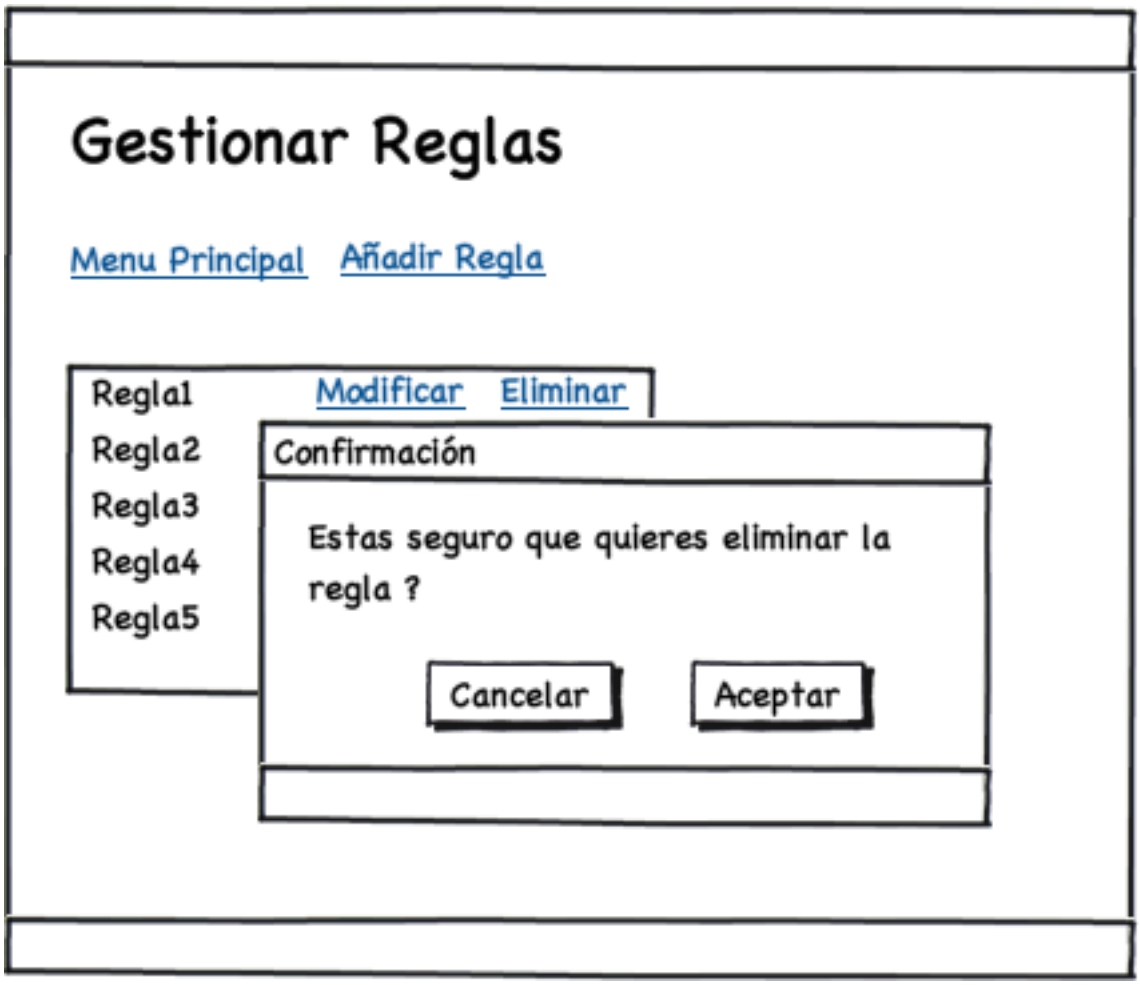
Menu Principal: El sistema cambia a la pantalla principal

Posibles Errores

La regla ya existe

El archivo no es válido

Eliminar Regla



Pantalla de Eliminar Regla

Funcionalidad

Elimina una regla seleccionada por el usuario.

Proceso

1. El usuario selecciona una regla para eliminar
2. El sistema pide confirmación al usuario
3. El usuario confirma que quiere eliminar la regla
4. El sistema eliminar la regla

Proceso alternativo

3a.1 El usuario cancela la eliminación.

3a.2 El sistema vuelve la pantalla de Gestión de Reglas

Botones/Enlaces

Cancelar: Cancela la eliminación.

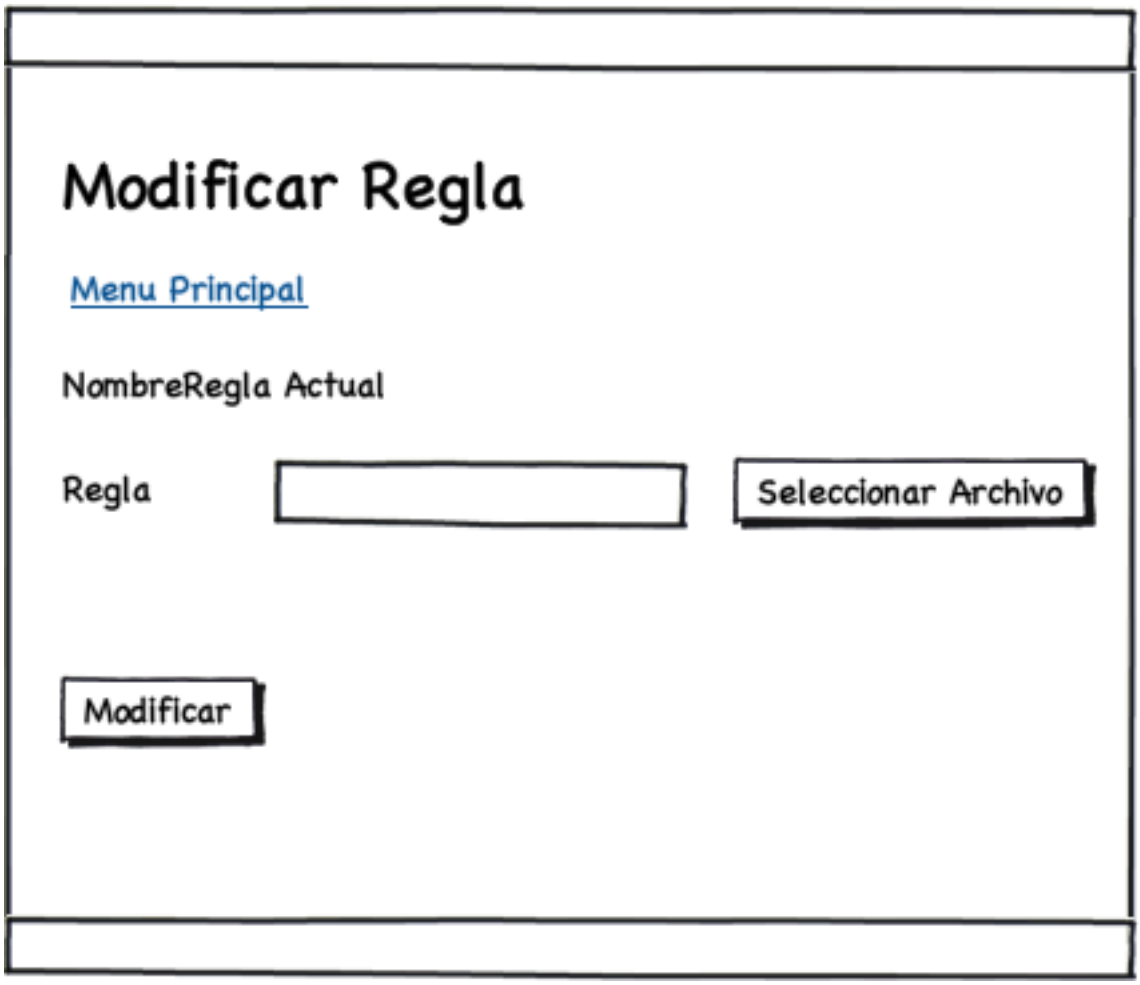
Aceptar: Confirma la cancelación

Menú Principal: El sistema cambia a la pantalla principal

Posibles Errores

No hay posibles errores

Modificar Regla



Pantalla de Modificar Regla

Funcionalidad

Modifica el código de una regla.

Proceso

1. El usuario indica el archivo donde se encuentra la nueva definición de la regla.
2. El usuario clica el botón Modificar.
3. El sistema modifica la definición de la regla

Botones/Enlaces

Seleccionar Archivo: Selecciona el archivo donde esta la definición de la regla.

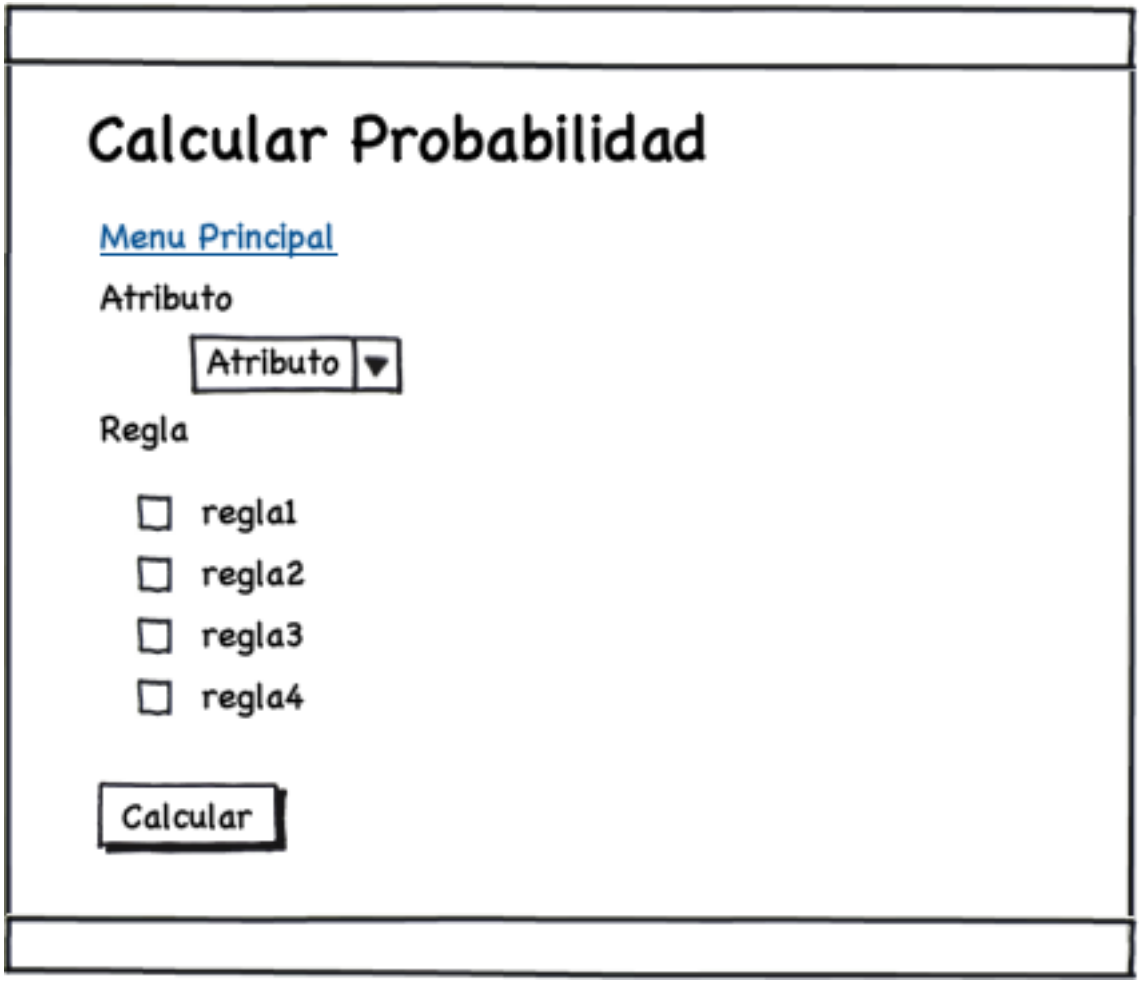
Modificar: El sistema modifica la definición de la regla.

Menu Principal: El sistema cambia a la pantalla principal.

Posibles Errores

El nombre de la regla elegida es igual a la de otra.

Calcular Probabilidades



Pantalla de Calcular Probabilidad

Funcionalidad

Busca posibles coincidencias dentro del conjunto de personas que se ha guardado en el sistema.

Proceso

1. El usuario indica el tipo de atributo que se usa para formar parejas (por ejemplo nombre o apellido)
2. El usuario indica que reglas se quieren aplicar
3. El usuario clica el botón de calcular
4. El sistema calcula las probabilidades
5. El sistema muestra los resultados al usuario.

Botones/Enlaces

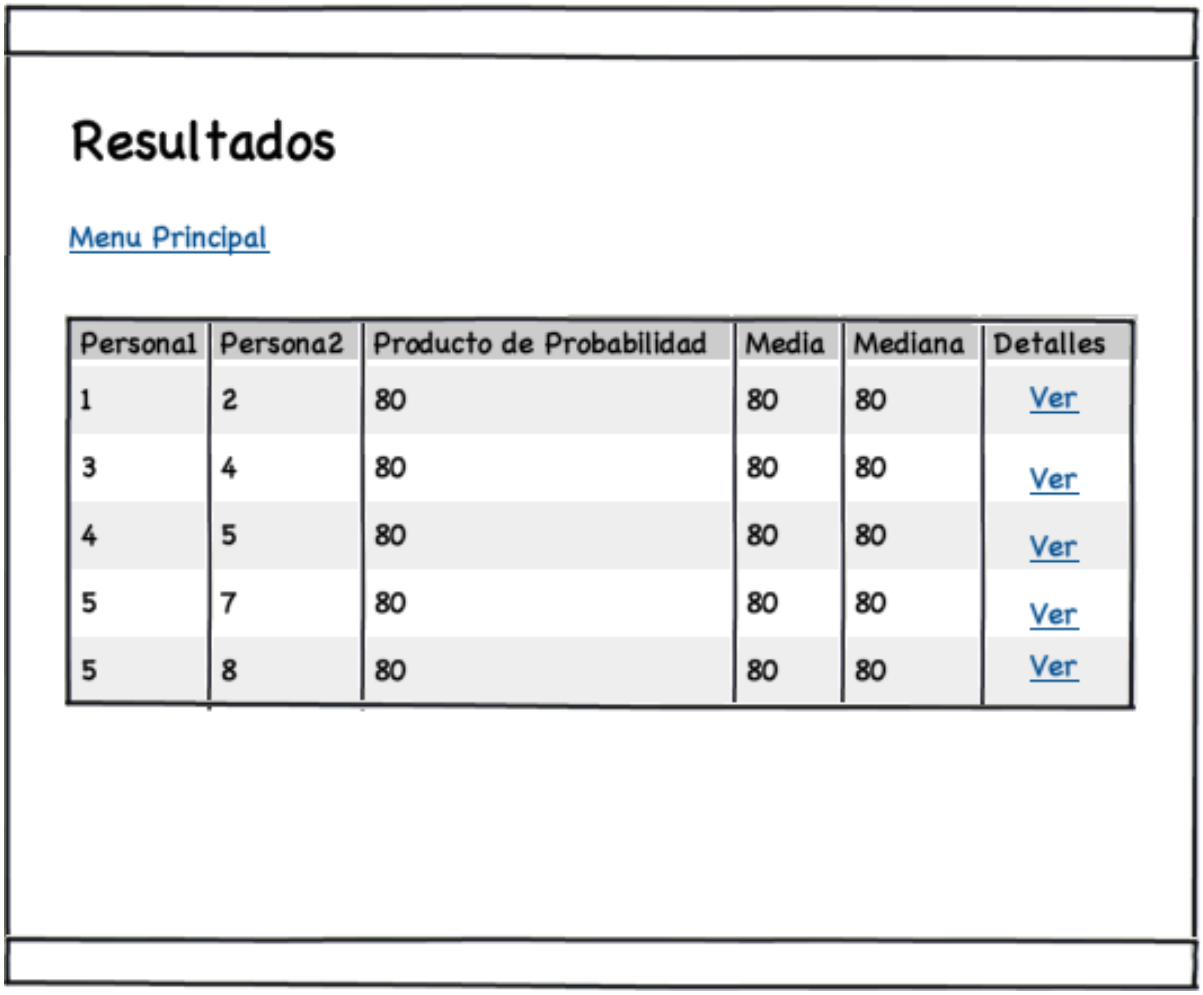
Calcular: Ejecuta el calculo.

Menu Principal: El sistema cambia a la pantalla principal

Posibles Errores

No hay errores posibles

Mostrar Resultados



Pantalla de Mostrar Resultados

Funcionalidad

Muestra los resultados obtenidos de un calculo

Proceso

1. El sistema muestra los resultados del calculo obtenido anteriormente.
2. El sistema da la opción al usuario de ver los detalles.

Botones/Enlaces

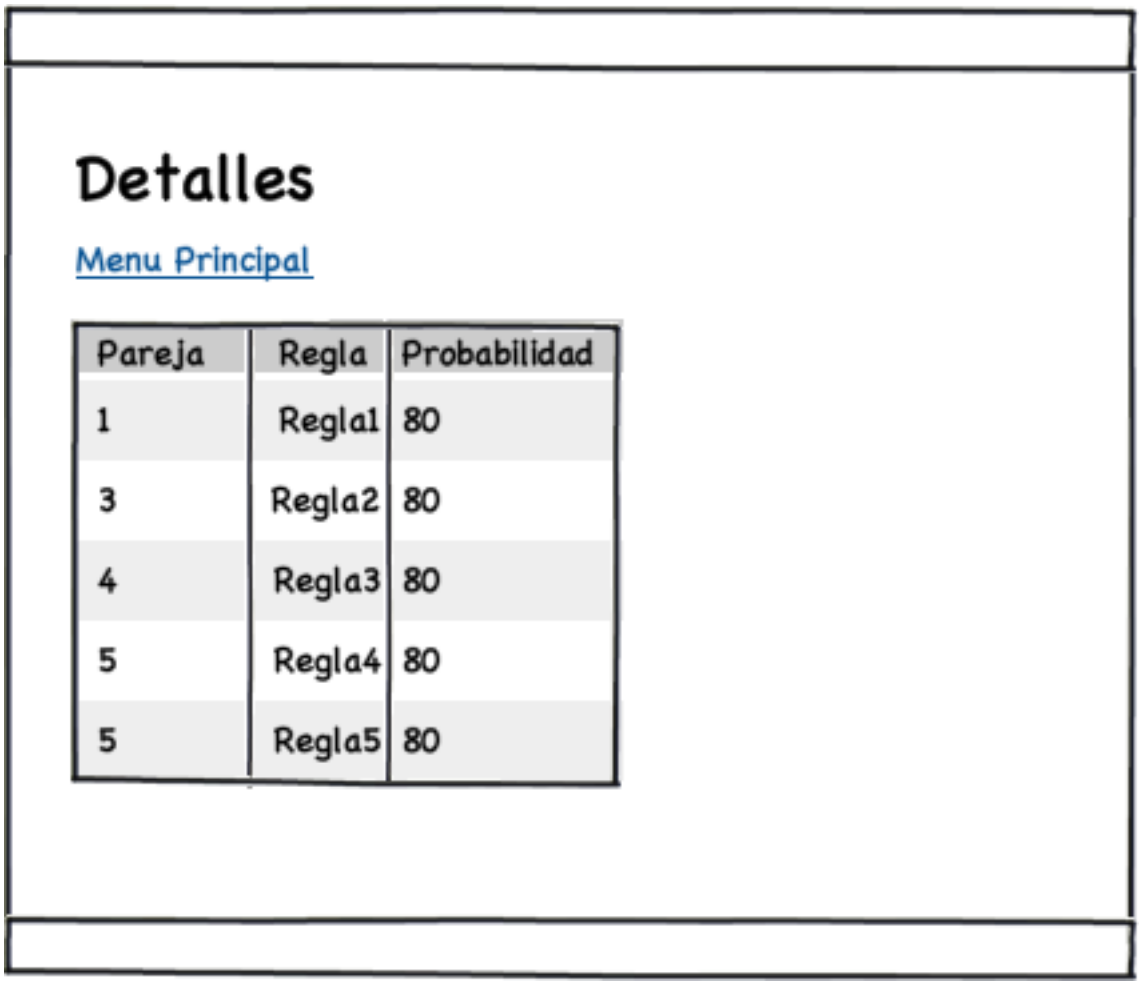
Ver: El sistema muestra los detalles del calculo.

Menu Principal: El sistema cambia a la pantalla principal

Posibles Errores

No hay posibles errores

Ver detalles



Pantalla de Ver Detalles

Funcionalidad

Permite consultar los detalles de cómo se ha hecho el cálculo. Estos detalles consisten en las probabilidades parciales que ha dado cada una de las reglas aplicadas a una pareja de persona.

Proceso

1. El sistema muestra los detalles de los cálculos hechos anteriormente.

Botones/Enlaces

Menu Principal: Vuelve a la pantalla de Mostrar Resultados

Posibles Errores

No hay errores posibles.

Controlador

El controlador se encarga de responder a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y a la vista.

Cada controlador se ha construido utilizando herencia. A continuación se muestra un diagrama con todos los controladores.

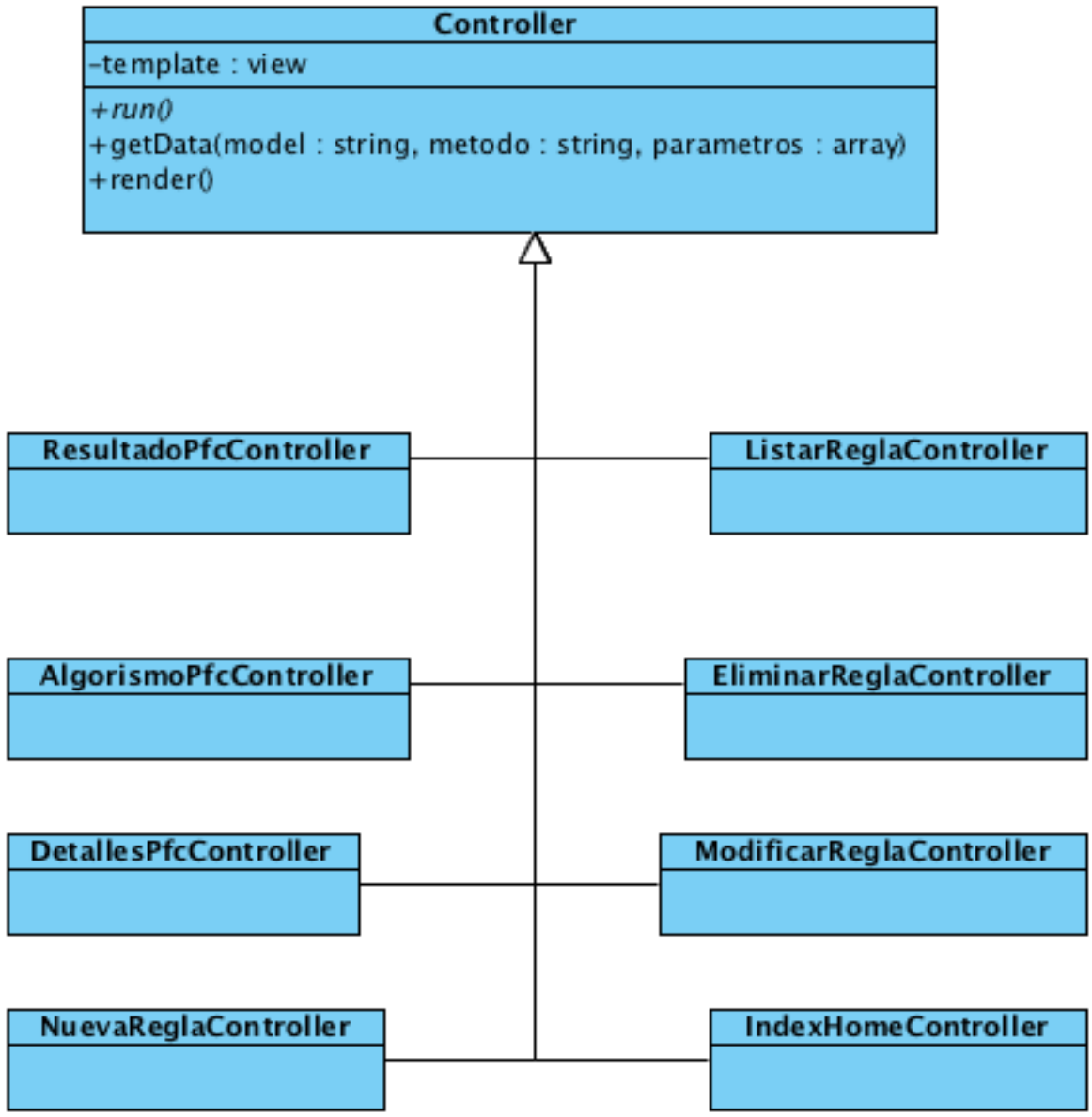
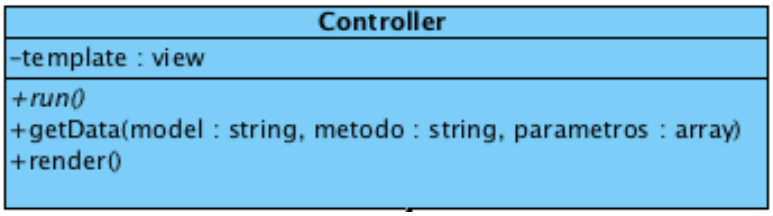


Diagrama de clases de tipo controlador

Clase Controller



Clase controller

De la clase Controller hay que destacar los siguientes elementos :

1. El atributos **template**, es la vista donde el controlador delega la presentación.
2. La función **getData**, que se encarga de llamar al modelo, indicando que método quiere llamar, a que modelo y los parámetros.
3. La función **run** ejecuta la función principal de la clase.
4. **render** muestra la vista relacionado con el controlador.

A continuación se muestran una relación entre cada controlador con los casos de usos de la aplicación.

Nombre de la Clase	Caso de uso
AlgorismoPfcController	Calcular Probabilidad
DetallesPfcController	Mostrar Resultado
ResultadoPfcController	
NuevaReglaController	Insertar Regla
ModificarReglaController	Modificar Regla
EliminarReglaController	Eliminar Regla
ListarReglaController	Gestionar Regla

Relación entre caso de uso y controlador

A continuación se explican los elementos más importante del controlador encargado de calcular las probabilidades de que dos personas sean la misma.

Algoritmo para el calculo de probabilidades

En esta sección se describen el algoritmo encargado de calcular el grado de similitud entre dos personas.

A continuación se muestran un diagrama que describe el funcionamiento del algoritmo de calculo de probabilidad. Para tener una visión general, se ha mostrado los elementos que actúan directamente con el usuario y que intervienen en el cálculo de probabilidades.

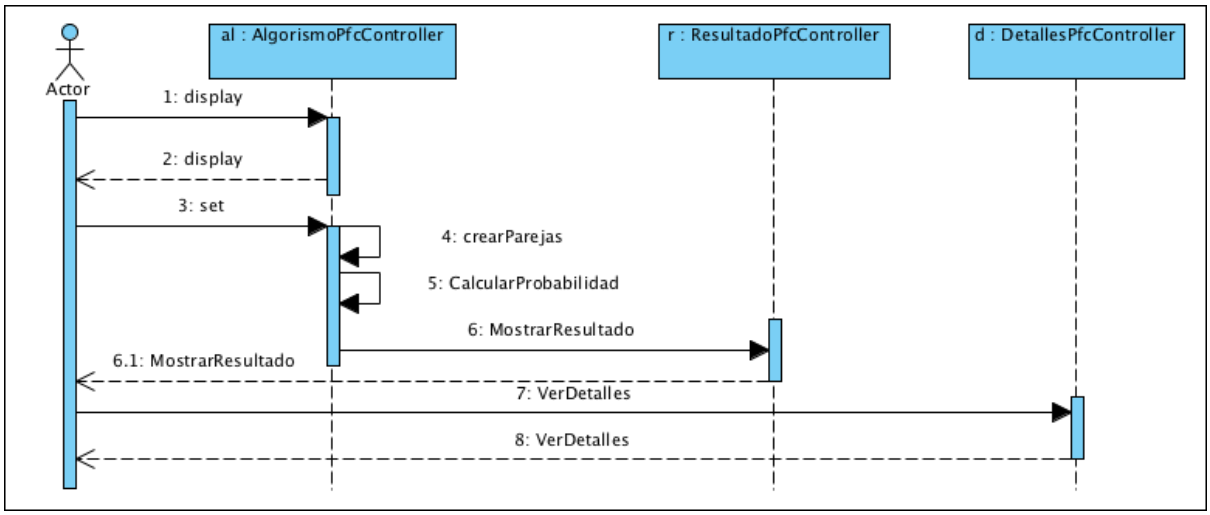


Diagrama de secuencia (Vista general)

A continuación se explica con más detalles las fases que tiene el algoritmo de calculo.

Configuración del algoritmo

Antes de comenzar los cálculos, el algoritmo hace una selección y un tratamiento previo de los datos que se utilizarán y prepara la base de datos para almacenar los resultados.
El usuario indica los siguientes parámetros:

- Las reglas utilizados.
- El atributo utilizado para crear parejas.

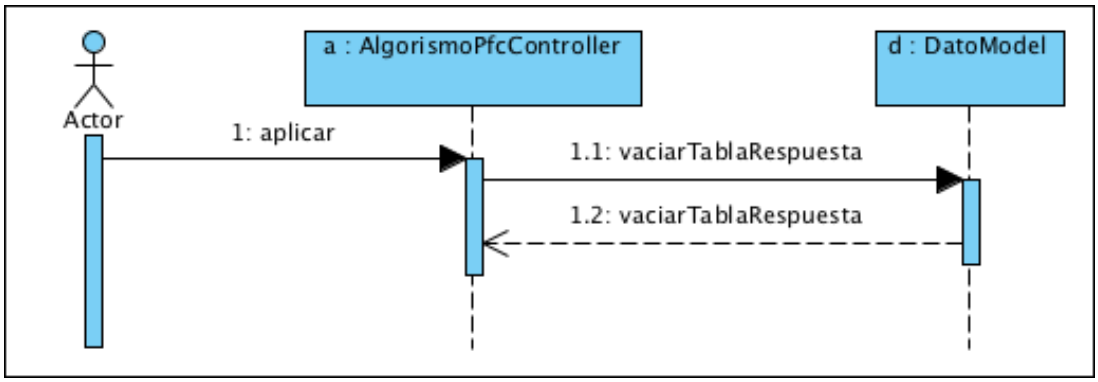


Diagrama de secuencia (configurar)

Creación de Parejas

Al crear las parejas, dos personas son emparejadas cuando se da una de las siguientes condiciones:

1. El valor del atributo elegido es igual en las dos personas
2. Como mínimo uno de los dos tiene valor nulo. Un valor nulo representa desconocimiento de ese atributo. Aun que ese valor sea nulo, no se descarta la posibilidad que la pareja represente a la misma persona. Con el resto de los atributos que se tiene, se podría determinar esta probabilidad.

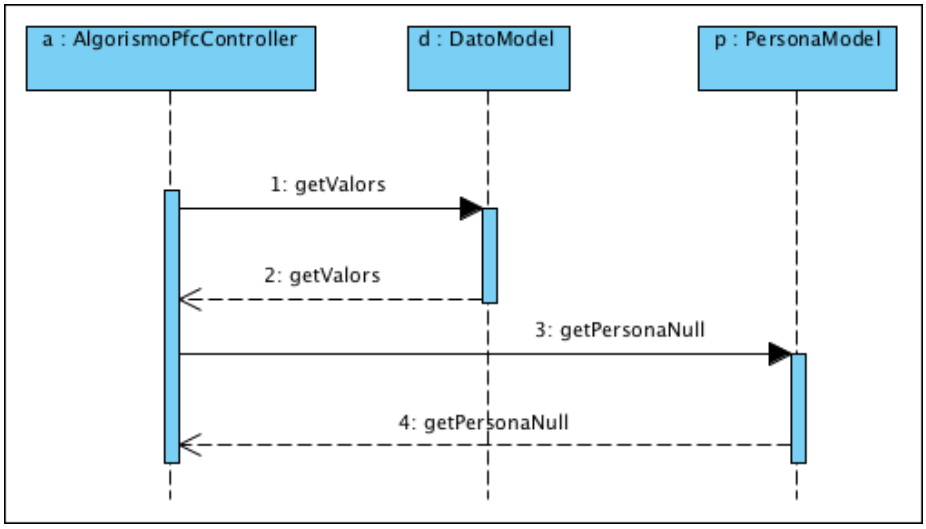


Diagrama de secuencia (Crear parejas)

Estos dos casos tienen en cuenta todas las pareja que tienen posibilidades de que sean la misma, descartando las que no.

Antes de emparejar se agrupan las personas que tengan el atributo elegido con el mismo valor. Con esto se pueden crear parejas entre cada persona de un mismo grupo. Las personas de un mismo grupo cumplen la condición de que tienen el mismo valor del atributo elegido. Esto facilita la tarea de emparejar las personas.

Para la segunda condición se juntan todas las personas con valor nulo con el resto de personas (tanto dentro del grupo como de los otros grupos formados).

Aplicación de Reglas

Se cogen todas las regles seleccionados anteriormente y se aplica a las diferentes parejas creadas. Las reglas se aplican en bloques de N parejas y se guardan los resultados en la base de datos. De esta manera, el uso de la memoria esta acotada.
Cuando se están aplicando las reglas a una pareja, su información esta guardada en memoria. En el momento de guardar los resultados en la base de datos, se libera memoria.

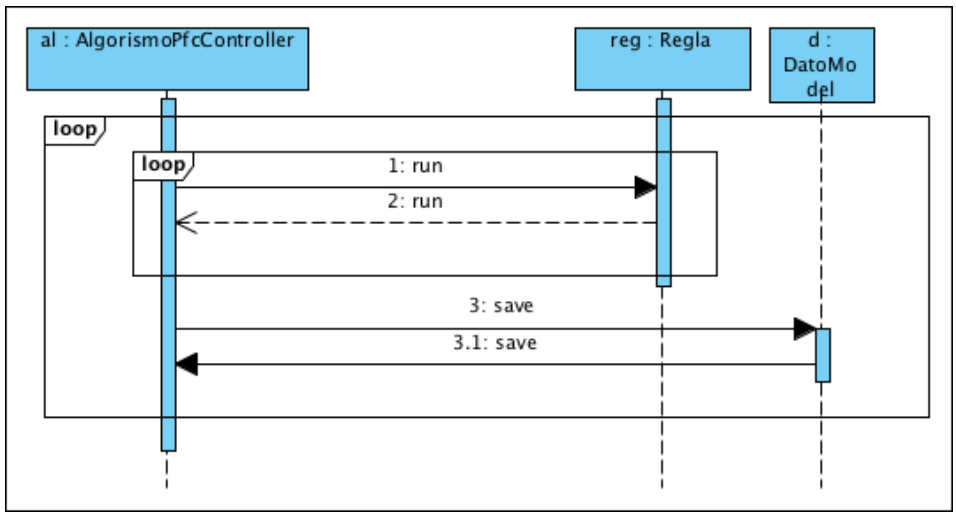


Diagrama de secuencia (Aplicación de Reglas)

Se va aplicando las reglas a una pareja hasta haber aplicado todas o cuando una de las reglas devuelva un valor que indique que dos personas no pueden ser la misma.
Si se han aplicado todas las reglas, se calcula la probabilidad de que sean la misma persona, en caso contrario la probabilidad será 0.

La probabilidad de que dos personas sean la misma son 3 valores:

- La mediana aritmética
- El producto de probabilidades.
- La media.

La razón de utilizar estos valores como probabilidad son las siguientes. La media y la mediana son medidas de tendencia central o de centralización. Con este valor se tiene un valor alrededor del cual tienden a concentrarse las probabilidades parciales obtenidas. La media tiene como característica que la suma de las distancias entre esta y las probabilidades parciales es la mínima. Como la media es muy sensible a los valores extremos, se utiliza también la mediana.

El producto de probabilidades representa la probabilidad teniendo en cuenta todas las probabilidades parciales. Se supone independencia entre las diferentes probabilidades parciales. Por contra si una probabilidad parcial es 0, el resultado del producto es 0. Por eso la media y la mediana también pueden ser valores que ayuden al usuario a decidir finalmente si dos personas pueden ser la misma.

Mostrar Resultados

La última fase consiste en mostrar al usuario los resultados obtenidos. Muestra para cada pareja la probabilidad de que sean la misma y los datos en que se ha basado los cálculos de las probabilidades.

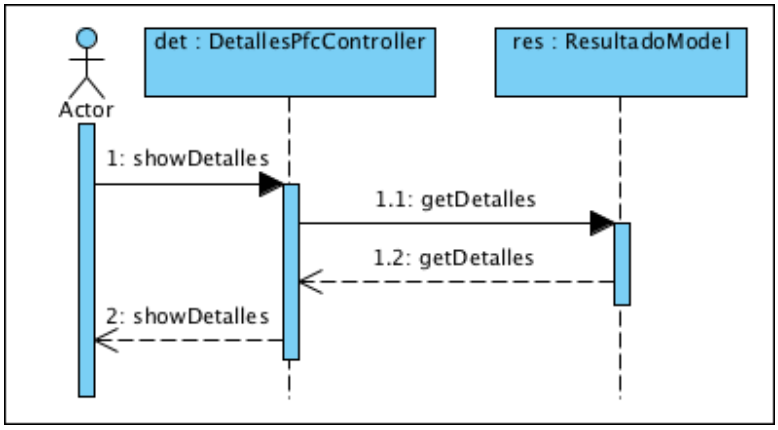


Diagrama de secuencia (Mostrar Resultados)

Para ver los detalles, el usuario selecciona un resultado de una pareja. En los detalles se muestran las probabilidades parciales que ha dado cada una de las reglas aplicadas.

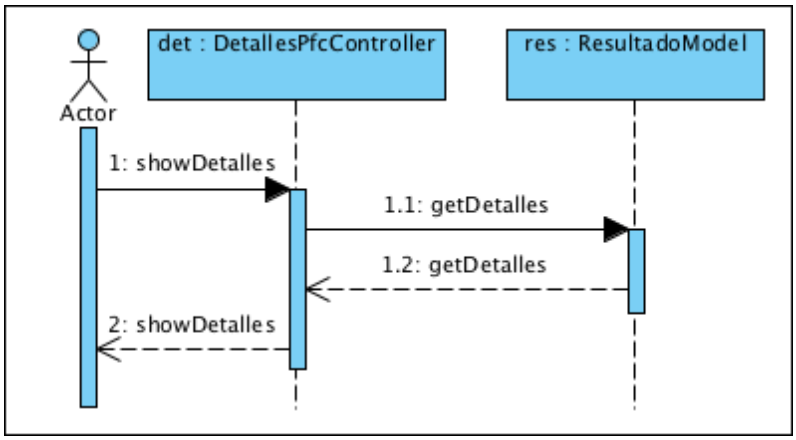


Diagrama de secuencia (Mostrar Detalles)

Modelo

El modelo se encarga de gestionar la lógica de negocio y acceder a la base de datos Cada modelo se ha construido utilizando herencia.

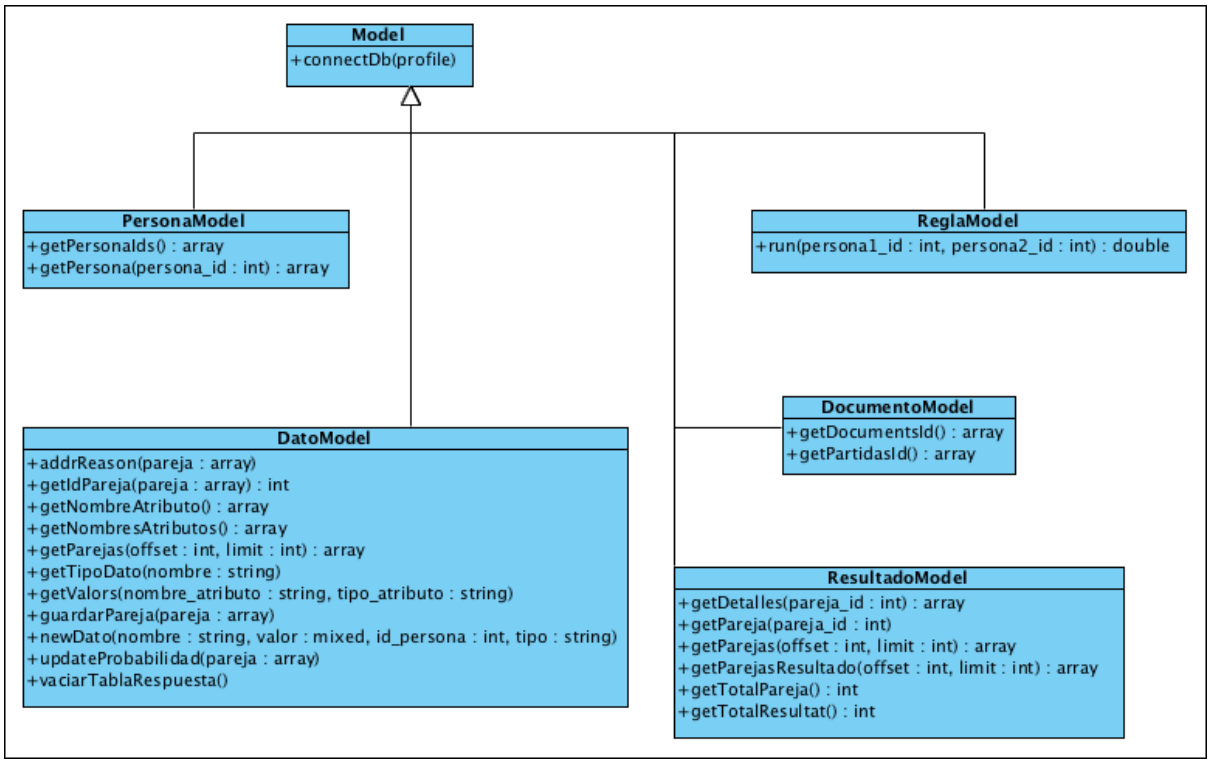


Diagrama de clases de tipo modelo

A continuación se van a describir los modelos creados.

DatoModel

Agrupa la lógica de negocio relacionada con los atributos que puede tener una persona.

DocumentoModel

Contiene la lógica de negoció relacionada con los diferentes documentos de donde pueden venir los datos de una persona.

Esta clase se puede utilizar en un futuro para mejorar las reglas. Sabiendo que un tipo de documento tiene unos atributos más fiables que en otro documento, podemos dar un peso a la regla.

Personamodel

Ayuda en el algoritmo para hacer las parejas para comparar posteriormente.

ReglaModel

Es la clase que representa a todas las reglas. Todas las reglas son subclases de esta.

ResultadoModel

Con esta clase se puede acceder a los resultados obtenidos en el calculo de probabilidades.

Diseño de la base de datos

Para la herramienta, se ha utilizado MySQL como sistema gestor de base de datos.
A continuación se describen todas las tablas utilizadas junto con sus campos y la equivalencia con el diagrama de clases del apartado de especificación.
Para obtener las tablas cada una de las clases del diagrama de clase se a convertido en una tabla. Las relaciones entre las tablas se han convertido en tablas o en claves foráneas dependiendo de la cardinalidad de la relación.

Persona

La tabla Persona almacena los identificadores de cada una de las personas.

Nombre	Tipo	Descripción	Modelo Conceptual
persona_id	entero	Identificador de la persona	-

Esquema de la tabla Persona

Probabilidad

La tabla Probabilidad guarda todos los resultados que ha dado un calculo de probabilidades.

Nombre	Tipo	Descripción	Modelo Conceptual
pareja_id	entero	identificador de la probabilidad	-
persona1_id	entero	clave foránea a la tabla persona	relación compara1
persona2_id	entero	clave foránea a la tabla persona	relación compara2
probabilidad	double	probabilidad de que dos personas sean la misma calculada a partir de	probabilidad.probabilidad

		producto de probabilidades parciales	
mediana	double	probabilidad de que dos personas sean la misma calculada a partir de la mediana de probabilidades parciales	probabilidad.mediana
media	double	probabilidad de que dos personas sean la misma calculada a partir de la mediana de probabilidades parciales	probabilidad.media

Esquema de la tabla Probabilidad

Razon

La tabla Razon almacena todos los resultados que ha dado todas regla aplicadas a cada una de las parejas.

Nombre	Tipo	Descripción	Modelo Conceptual
razon_id	entero	identificador de la razón	-
regla_id	entero	clave foránea a la tabla regla	relación : regla
distancia	double	probabilidad parcial de la regla utilizada	razón.probabilidad_parcial
pareja_id	entero	clave foránea a la tabla probabilidad	relación : probabilidad

Esquema de la tabla Razon

Relacion

La tabla Relacion guarda toda la información relativa a la relación entre las diferentes personas de la base de datos, si tienen algún vinculo familiar.

Nombre	Tipo	Descripción	Modelo Conceptual
persona1_id	entero	clave foránea a la tabla persona	relación persona1
persona2_id	entero	clave foránea a la tabla persona	relación persona2
relacion	entero	Tipo de relación (pareja, padre_madre, hijo_hija, hermano_hermana)	relación.relacion

Esquema de la tabla Relacion

Dato

La tabla dato indica la información disponible de cada una de las personas de la base de datos.

Nombre	Tipo	Descripción	Modelo Conceptual
nombre	cadena de texto	nombre del dato de una persona	dato.nombre
dato_id	entero	identificador del dato	-
tipo_dato	cadena de texto	tipo de dato de una persona (fecha, numero, palabra, booleano)	dato.tipo
partida_id	entero	clave foránea a la tabla partida	relación partida
persona_id	entero	clave foránea a la tabla persona	relación persona

Esquema de la tabla Dato

Date

Esta tabla guarda toda la información de una persona que sean fecha.

Nombre	Tipo	Descripción	Modelo Conceptual
dato_id	entero	clave foránea a la tabla dato e identificador de date	relación dato
valor	tiempo	valor del dato	date.valor

Esquema de la tabla Date

Boolean

La tabla Boolean guarda toda la información de una persona de tipo booleana.

Nombre	Tipo	Descripción	Modelo Conceptual
dato_id	entero	clave foránea a la tabla dato e identificador de boolean	relación dato
valor	booleano	valor del dato	boolean.valor

Esquema de la tabla Boolean

String

Esta tabla almacena toda la información de una persona. Esta información son de tipo texto.

Nombre	Tipo	Descripción	Modelo Conceptual
dato_id	entero	clave foránea a la tabla dato e identificador de string	relación dato
valor	cadena de texto	valor del dato	string.valor

Esquema de la tabla String

Numerico

La tabla Numerico guarda la información de una persona. La información que almacena son números.

Nombre	Tipo	Descripción	Modelo Conceptual
dato_id	entero	clave foránea a la tabla dato e identificador de entero	relación dato
valor	entero	valor del dato	numerico.valor

Esquema de la tabla Numerico

Documento

Esta tabla representa los documentos donde se han conseguido los datos de una persona

Nombre	Tipo	Descripción	Modelo Conceptual

documento_id	identificador de documento	-

Esquema de la tabla Documento

Partida

La tabla Partida guarda todas las partidas donde se han conseguido información de una persona.

Nombre	Tipo	Descripción	Modelo Conceptual
partida_id	entero	identificador de partida	-

Esquema de la tabla Partida

Regla

La tabla Regla almacena información de todas las reglas guardadas en el sistema.

Nombre	Tipo	Descripción	Modelo Conceptual
regla_id	entero	identificador de regla	-
nombre	cadena de texto	nombre de la regla	regla.nombre

Esquema de la tabla Regla

El siguiente esquema muestra la relación de las tablas.

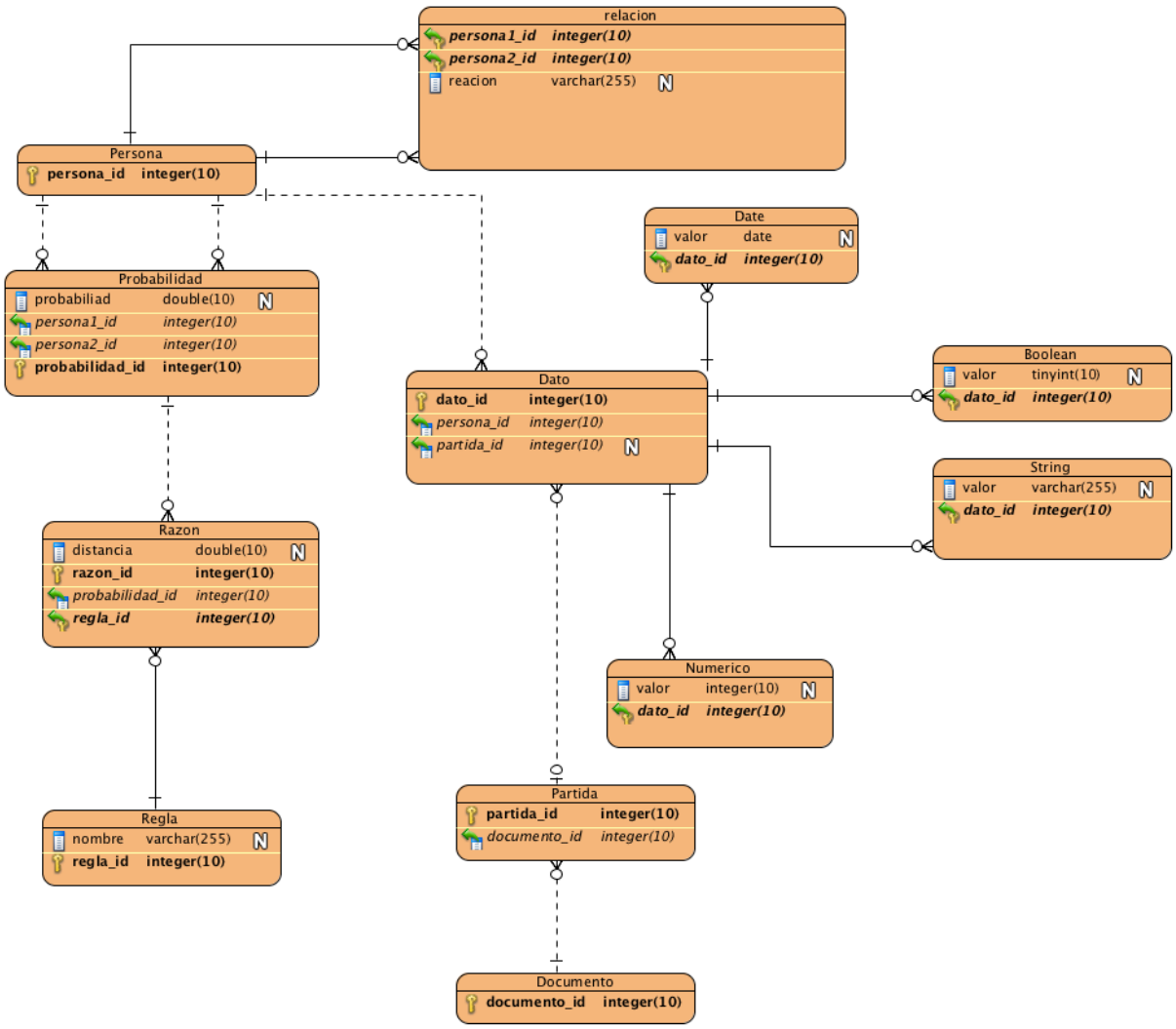


Diagrama de base de datos

Reglas

A continuación se explica el diseño una regla y su funcionamiento.

Representación de una regla

Las reglas están definidas en diferentes ficheros. La razón de esto es para que haya la posibilidad de poder intercambiar fácilmente las reglas y el usuario final pueda importar nuevas reglas definidas por otros usuarios o exportarlas para compartirlo con otros.

Las reglas sean incluidas dinámicamente para el calculo de las probabilidades. Al intentar utilizar una regla, el sistema comprueba que la regla ha sido incluida. En el caso contrario, lo incluye.

Comportamiento de una regla.

En una regla compara los atributo que se consideran. Las comparaciones se hacen usando las distancias para calcular las probabilidades. Una distancia mide lo lejos que puede estar dos valores de un mismo tipo de atributo. Esta idea esta inspirado en las distancias de Levenshtein[2].

Dependiendo del tipo de dato, la distancia se calcula de diferente manera, pero conservando la idea de Levenshtein:

- En el caso de los datos que sean palabras, se calcula la distancia de Levenshtein. Por ejemplo en el caso de los nombres Patrick y Patric hay una distancia de una ya que se tiene que hacer una modificación para que los dos nombres sean iguales. También se compara la fonética. En el caso de que fonéticamente no sean iguales, la distancia resultante será más grande. Para hacer esta comparación es usa el algoritmo soundex[3]. Si dos palabras se pronuncian igual fonéticamente soundex tendrá la misma codificación.

[2] Distancia de edición, o distancia entre palabras, al número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Estas operaciones pueden ser tanto inserción, eliminación o sustitución de un carácter.
[3] Es un algoritmo que codifica de la misma forma los nombres con la misma pronunciación.

- Para los valores numéricos, se ha optado por usar la misma idea pero la distancia se calcula por el valor absoluto de la diferencia entre dos atributos numéricos.

- En el caso de las fechas se hace la diferencia entre días, meses y años, cada uno con una ponderación.

Una vez calculado la distancia, se calcula una probabilidad teniendo en cuenta la distancia que ha dado la comparación entre dos atributos y una distancia máxima para acotar el resultado. Se ha decido para calcular el grado de similitud de la siguiente manera:

$$Similud = 1 – (distancia/distanciaMaxima)$$

Cuando hay más distancia entre los dos atributos, menos similares son estos dos atributos y por tanto las dos personas tiene menos posibilidad de ser la misma. Ha sido difícil para poder calcular esta probabilidad, ya que no es algo trivial. En un futuro se podría mejorar la forma de hacer el cálculo para que sea más preciso.

Hay casos en que una de las dos personas tiene, para un mismo atributo varios valores . En estos casos se aplica la siguiente lógica: Se compara un atributo con todas las de la otra persona y se guarda el valor mínimo. Posteriormente se calcula la media de los mínimos que se ha obtenido.

Una situación en el que este caso se puede dar es cuando dos personas tengan nombres parecidos o los mismos nombres pero en un orden diferente. Calculando de la manera descrita anteriormente, se tiene en cuenta esta situación y no influirán negativamente en el calculo de probabilidad. Esta idea solo es aplicable con atributos que sean datos personales y que puedan ser variables.

Valores de una regla

Una regla puede devolver, a parte de una probabilidad, un valor que indique indeterminación o un valor que descarte la posibilidad que dos personas no sean la misma.

Se descarta que dos personas puedan ser similares, cuando en una regla, los atributos que se están analizando entran en contradicción. La regla devolverá un valor negativo para indicar que se ha dado este caso.

Un ejemplo es una regla que tenga en cuenta la fecha de nacimiento y la fecha de fallecimiento. Si se encuentra que la fecha de nacimiento de una persona es posterior a la de fallecimiento de la otra, no es posible que estas dos personas puedan ser la misma.

En el caso de que los atributos que se usan en una regla sean desconocidos, no se considera la regla, para no perjudicar en la probabilidad. La regla devuelve un valor nulo para indicar que no se sabe la probabilidad por falta de datos. A la hora de calcular la media , mediana y producto de probabilidades este valor no se tiene en cuenta.

Implementación

En la fase de implementación se desarrolla todo lo detallado durante la especificación y diseño. Esta etapa es tan importante como cualquiera anterior, aunque a partir de este punto ya se obtiene un sistema.

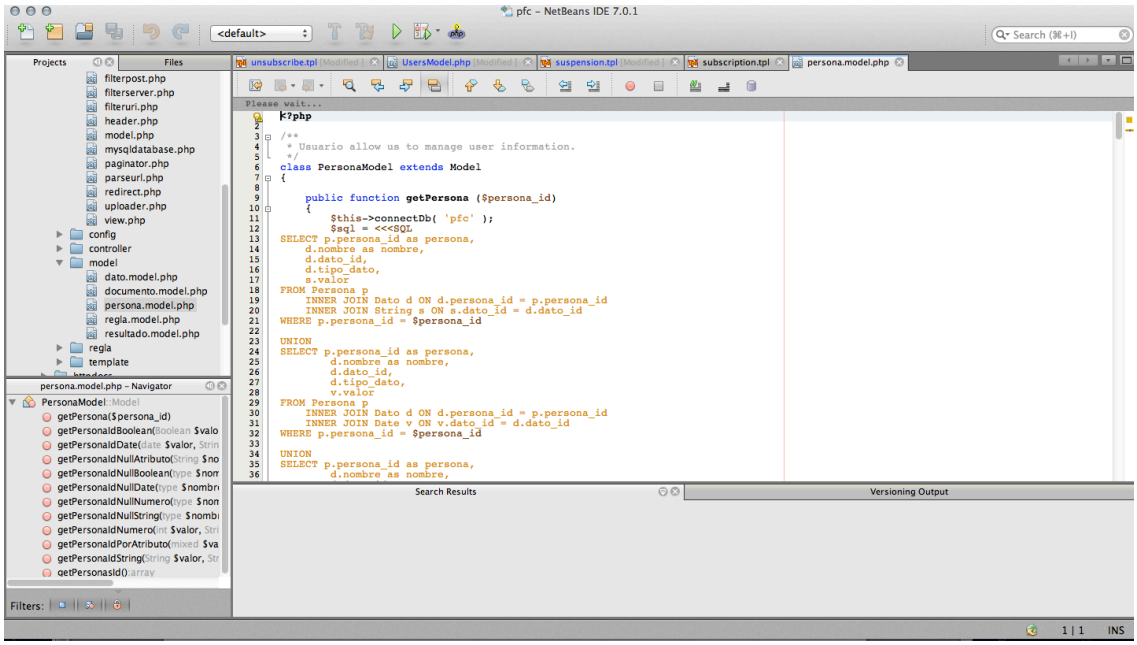
Descripción del entorno de desarrollo

En este apartado se pretende describir el entorno de desarrollo que se ha seguido para implementar el proyecto.

Para realizar la implementación de la herramienta detallada anteriormente sólo es necesario un editor de texto plano y un servidor.

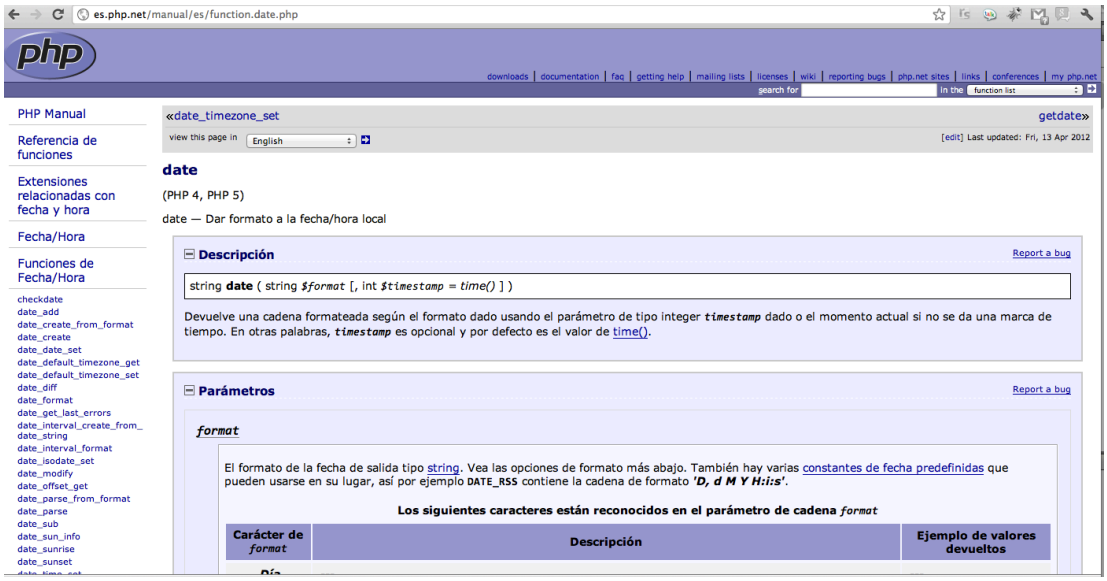
Sin embargo, en proyectos de mediana o gran envergadura es recomendable la utilización de herramientas más eficiente y potentes. En el caso de este proyecto, se ha utilizado un entorno de desarrollo integrado (IDE) que permite crear aplicaciones en el lenguaje PHP.

Se ha optado por el IDE NetBeans 7.0 (http://www.netbeans.org) con soporte para PHP. La razón de esta elección es la rapidez y facilidad de instalación y sus características avanzadas de autocompletado de código o estructuración de las clases.



Entorno de desarrollo Netbeans

Además se ha utilizado la documentación online de PHP (www.php.net) para consultar información sobre algunas funciones del lenguaje.

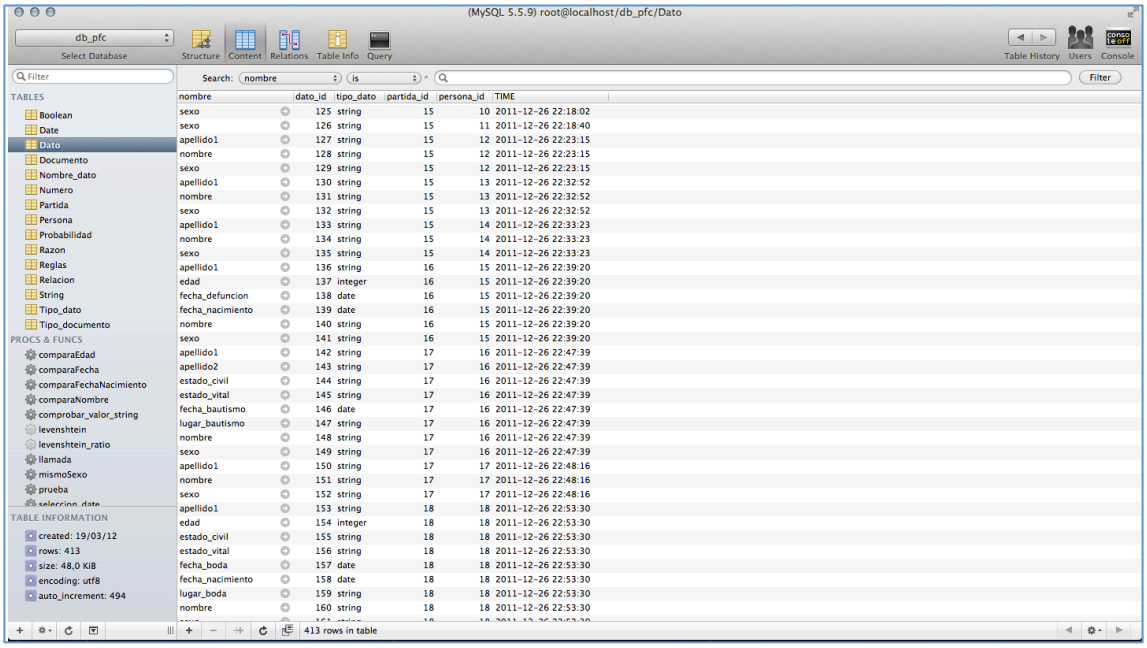


API de PHP

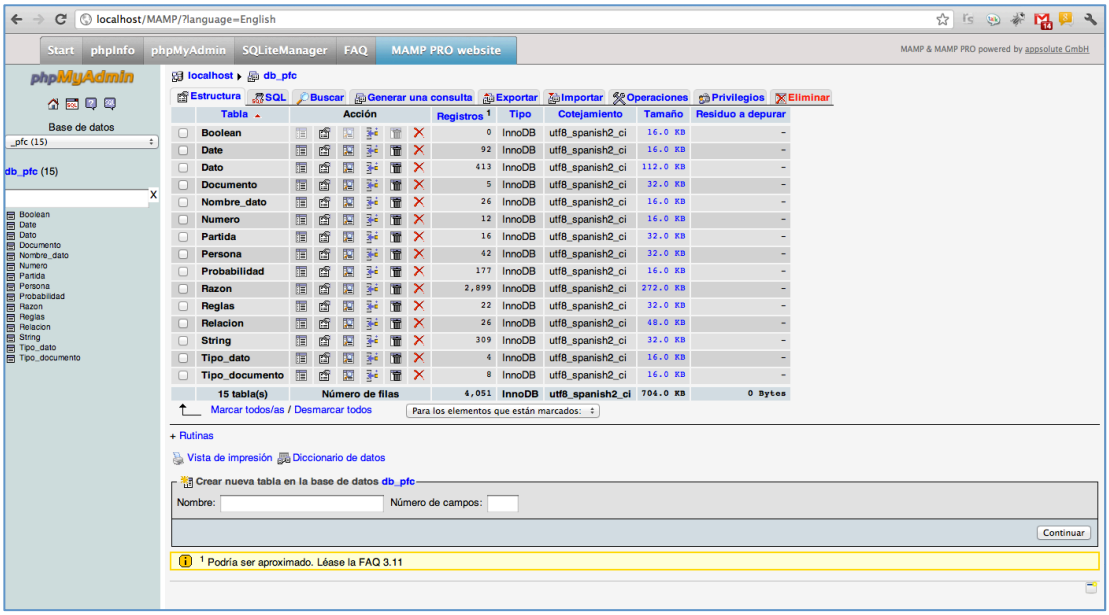
A medida que se ha ido desarrollando la herramienta, se ha ido comprobando su resultado en diferentes navegadores (Mozilla Firefox y Google Crome). Además en algunas casos se ha utilizado el depurador de código xdebug (http://www.xdebug.org) que ha permitido comprobar el correcto funcionamiento del código.

Además se ha utilizado un sistema de control de versiones, para poder tener una copia de seguridad de los avances que se han hecho durante la implementación o tener la posibilidad de recuperar un estado anterior del código.

También se ha utilizado la herramienta phpMyAdmin (http://www.phpmyadmin.net/home_page/index.php) y Sequel Pro (http://www.sequelpro.com/) para verificar los valores almacenados en la base de datos y para comprobar las consultas SQL que usa la herramienta.



Gestor de base de datos de Sequel



Gestor de base de datos de PHPMyAdmin

En los siguientes apartados se mostrarán los aspectos que se han considerado las más interesantes de la implementación de la herramienta.

Estructura del framework

La estructura del framework está formada por carpetas anidadas :

- app : carpeta donde se encuentra el código fuente de la herramienta
- httpdocs: directorio que ejecuta la herramienta.

En la carpeta app esta todo el código que implementa la herramienta. Tiene la siguiente estructura:

- app
    - class: carpeta que agrupa las clases necesarias para el funcionamiento del framework.
    - config: guarda la configuración para el funcionamiento del framework.
    - controller : agrupa todos los controladores que forma la herramienta.
    - model: contiene todos los modelos utilizados en la herramienta.
    - regla: guarda todas las reglas.
    - template: contiene todas las vistas utilizadas.

Controladores

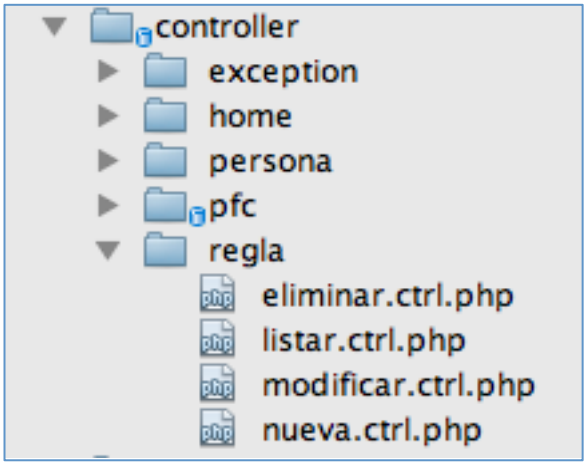
A la hora de implementar un nuevo controlador se tiene que guardar dentro de la carpeta controller y agrupados junto a otros controladores. El nombre del fichero donde se implementa sigue la siguiente nomenclatura:

<nombre_controlador>.ctrl.php

El nombre de la clase que implementa el controlador tiene que seguir la siguiente nomenclatura:

<grupo_controlador><nombre_controlador>Controller

De esta manera el controlador que modifica una regla esta dentro de la carpeta ***regla***, que agrupa todos los controladores relacionados con las reglas, el fichero se llama ***modificar.ctrl.php*** y la clase se llama ***ModificarReglaController***.



Directorio donde se guarda el controlador para modificar una regla

```php
<?php

class ModificarReglaController extends Controller
{
    public function run()
    {
        $message = '';
        $post = FilterPost::getInstance();
        $nombre_regla = $post->getText( 'nombre_regla', false );
        $regla_id = $this->uri->getNumber( 'regla_id', false );
        if ( $nombre_regla  && $regla_id )
        {
            $arguments = array( $regla_id, $nombre_regla );
            $res = $this->getData( 'ReglaModel', 'setNombre',$arguments );
            if ( !$res)
            {
                $message = 'No se ha modificado correctamente la regla';
            }
            else
            {
                $message = ' Se ha modificado correctamente la regla';
            }
        }
        $arguments = array ($regla_id);
        $nombre = $this->getData( 'ReglaModel', 'getReglaById',$arguments );

        $this->template->assign( 'message', $message );
        $this->template->assign( 'nombre', $nombre );
        $this->template->setTemplate( 'regla/modificar');

    }
}
?>
```

Código del controlador para modificar una regla

Modelos

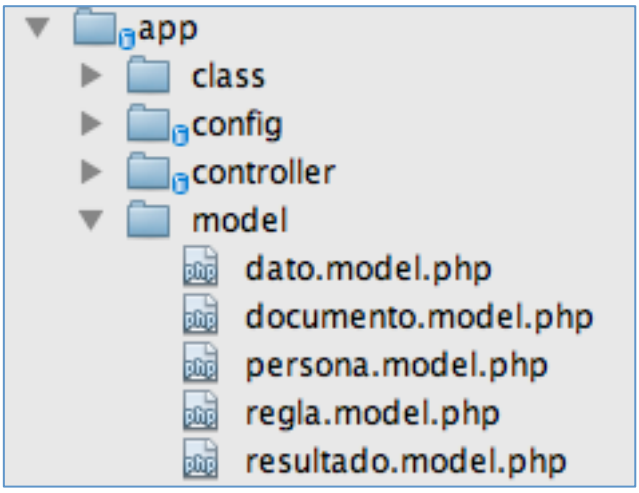
Para implementar un modelo, el fichero que lo implementa se guarda en la carpeta model y el nombre del fichero debe seguir la siguiente nomenclatura:

<nombre_modelo>.model.php

El nombre de la clase sigue la siguiente nomenclatura:

<nombre_modelo>Model

En el caso del modelo que gestiona las reglas, el fichero tiene como nombre regla.model.php y la clase que lo implementa se llama ReglaModel.



Directorio donde se guarda el modelo que gestiona las reglas

```php
<?php

class ReglaModel extends model
{
    public function getFecha( $personal_id , $nombre )
    {...}

    public function isDate( $fecha )
    {...}

    public function isYear( $fecha )
    {...}

    public function comparaFecha ( $fecha1 , $fecha2    )
    {...}

    public function comparaPalabras ( $palabra1 , $palabra2 )
    {...}

    public function getPalabra( $persona_id , $nombre )
    {...}

    public function getNumero ( $persona_id , $nombre )
    {...}

    protected function comparaListaNumero ( $numeros1 , $numeros2 )
    {...}

    public function comparaListaPalabras( $palabras1 , $palabras2 )
    {...}

    public function getNombreReglas()
    {...}

    public function getReglaByID( $regla_id)
    {...}

    public function newRegla( $nombre_regla )
    {...}

    public function setNombre ( $regla_id , $nombre_new_regla )
    {...}
    public function delRegla ( $regla_id )
    {...}

    public function isFechaAnterior ( $fecha1 , $fecha2    )
    {...}

    public function getReglas()
    {...}

    public function getRelations( $persona_id , $relation )
    {...}

    protected function nombreRegla( $nombre )
    {...}

    protected function isNombreValid( $nombre )
    {...}

    protected function setValidNombreRegla ( $nombre )
    {...}

    public function isParent( $persona1 , $persona2 )
    {...}
```

```
    public function isHijo ( $personal , $persona2 )
    {...}

    public function isPareja( $personal, $persona2 )
    {...}

    protected function stripAccents( $String )
    {...}

    public function loadFile( $regla_name, $regla_file )
    {...}

    protected function updateRegla( $regla_name, $regla_file )
    {...}

    public function computeMean( $valores , $peso = null)
    {...}

    public function computeMediana ( $valores )
    {...}

    public function computeProducto ( $valores )
    {...}

    public function getProbabilitats ( $points )
    {...}

    protected function CallRecursiveAverage($m0, $vector)
    {...}

    protected  function is_unique ( $numbers )
    {...}

    protected function checkPoints( $m0, $points )
    {...}
}
```

Código del modelo que gestiona las reglas

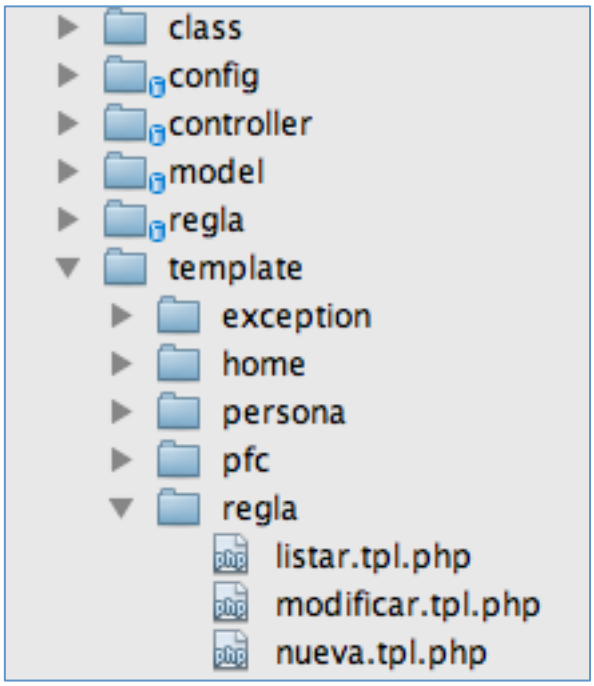
Vistas

Las vistas se guardan en la carpeta de template. Como pasa con los modelos y controladores, las vistas se guardan en ficheros usando la siguiente nomenclatura:

<nombre_template>.tpl.php

Para saber que vista tiene un controlador, los templates se agrupan de la misma manera que los controladores.

De esta manera la vista que es responsable de modificar una regla tiene como nombre modificar.tpl.php.



Directorio donde se guarda la vista responsable de la modificación de una regla

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>Title of the document</title>
    <link rel="stylesheet" type="text/css" href="<?php echo CSS_PATH . '/global.css'; ?>" />
</head>
<body>
    <?php echo $this->nombre ?>

<form method="post" action="" >
    <input type="text" value="<?php echo $this->nombre?>" name ="nombre_regla" />
    <input type="submit" name="Modificar"/>
</form>

<p> <?php echo $this->message ?> </p> <br/>

<a href="<?php echo HTTP. DOMAIN . '/'; ?>"> Menu </a><br/>
<a href="<?php echo HTTP. DOMAIN . '/listar_regla';?> "> Listar Reglas</a>
</body>
</html>
```

Código de la vista responsable de la modificación de una regla

Reglas

Para implementar una regla se utiliza PHP y SQL. El primero para poder comunicarse con el servidor y la otra para poder gestionar la base de datos y poder hacer consultas. Un fichero tiene que contener una regla y el nombre del fichero sigue la siguiente nomenclatura:

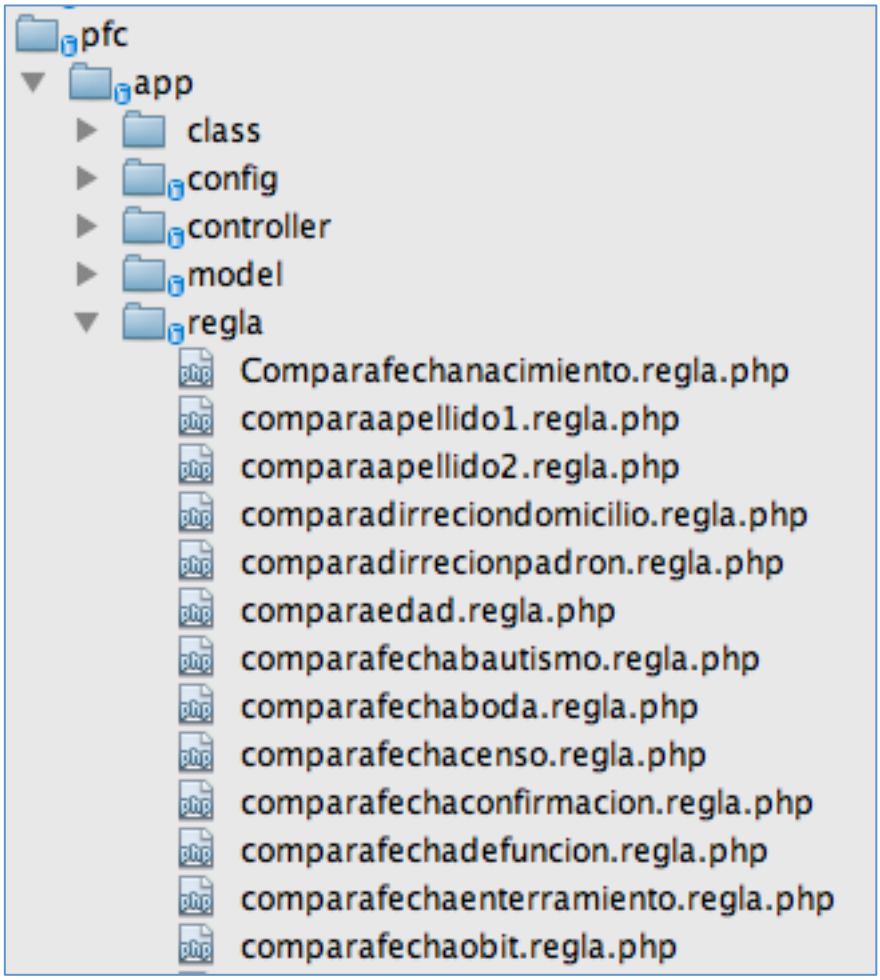
<nombre_regla>.regla.php

El fichero se ha de guardar en la carpeta regla y la clase que lo implementa debe llamarse de la siguiente manera:

<nombre_regla>Regla

De esta manera la herramienta puede identificar fácilmente las reglas e incluirlas cuando se necesite.

En este caso para la regla ***comparafechanacimiento*** el fichero tiene como nombre ***comparafechanacimiento***.regla.php.



Directorio donde se guarda la regla comparafechanacimiento

La clase que lo implementa se llama ***MismosexoRegla.***

```php
<?php


class ComparaFechaNacimientoRegla extends ReglaModel{

    public function run( $persona1_id, $persona2_id )
    {

        $distancia = 0;
        $fecha1  = $this->getFecha( $persona1_id , 'fecha_nacimiento' );
        $fecha2  = $this->getFecha( $persona2_id , 'fecha_nacimiento' );

        if ( $fecha1 && $fecha2 )
        {
            $distancia  = $this->ComparaFecha($fecha1 ,$fecha2);
        }

        return $distancia;

    }

}

?>
```

Clase que implementa la regla comparafechanacimiento

Inclusión de una regla

Todas las reglas que se implementen heredan de la clase ReglaModel.

Para incluir los ficheros que guardan una regla, se utiliza una función llamada autoload.

```php
function __autoload( $class_name )
{

    if ( 'Twig_Autoloader' === $class_name )
    {
        $filename = CORE__CLASSES_DIR . '/lib/twig/Autoloader.php';
    }

    else
    {
        preg_match('/^([A-Z][a-z0-9]+)([A-Z][a-z0-9]+)?(Controller|Model|Interface|Regla)/'
                , $class_name, $matches );

        $objet_type = ( isset( $matches[3] ) ) ? $matches[3] : '';
        switch ( $objet_type )
        {
            case 'Controller':
                $filename = CORE__CONTROLLER_PATH . '/' .
                    strtolower( $matches[ 2 ] )  . '/' .
                    strtolower( $matches[ 1 ] ) . '.ctrl.php';
                break;

            case 'Model':
                $filename = CORE__MODEL_PATH . '/' .
                    strtolower( $matches[ 1 ] ) . '.model.php';
                break;

            case 'Interface':
                $filename = CORE__CLASSES_DIR . '/' .
                    strtolower( $matches[ 1 ] ) . '.iface.php';
                break;
            case 'Regla':
                $filename = CORE__REGLA_PATH . '/' .
                    strtolower( $matches [ 1 ] ) . '.regla.php';
                break;

            default:
                $filename = CORE__CLASSES_DIR . '/' .
                    strtolower( $class_name ) . '.php';

        }
    }
    require_once $filename;
}
```

Codigo de Autoload

Identifica una regla a partir de las condiciones que se han descrito anteriormente para implementar una regla. Las incluye en el fichero si es necesario. Esta función se ejecuta automáticamente si una clase no esta incluida dentro de un fichero que la necesita.

Uso de una regla

Utilizando el mecanismo descrito anteriormente, para utilizar una regla solo se necesita el nombre de la clase que implementa la regla, crear una instancia de ella y ejecutar la regla.

El siguiente código es una parte de la implementación del algoritmo para el calculo de probabilidades. La variable ***$regla*** tiene el nombre de todas las reglas seleccionadas.
Se llama la función ***getData*** para ejecutar una regla. Los argumentos que necesita una regla están guardados en la variable ***$arguments***.

```
while ( $probabilidad >= 0 && $count < $numero_regla  )
{
    $probabilidad = $this->getData( $reglas[ $count ] . 'Regla', 'run', $arguments);
    $parejas[$num]['distancia'][ $reglas[ $count] ] = round( $probabilidad, 2) ;
    if ( null != $probabilidad )
    {
        $probabilidades_parciales[] = $probabilidad;
    }
    $count++;
}
```

Trozo de código donde se ejecuta una regla

```
public function getData( $model, $method, $arguments )
{
    $cache_obj = CacheGetData::getInstance();

    return $cache_obj->get( $model, $method, $arguments );
}
```

```
public function get( $model, $method, $arguments )
{
    $get_data_key = $model . $method;

    $get_data_key = $get_data_key . serialize( $arguments );

    if ( !array_key_exists( $get_data_key , $this->cache_data ) )
    {
        $object = new $model();
        $result = call_user_func_array( array( $object, $method ), $arguments );
        $this->set( $model, $method, $arguments, $result );
    }

    return $this->cache_data[ $get_data_key ];
}
```

Función encargada de crear una instancia de una regla y ejecutarla

Pruebas

Esta etapa consiste en comprobar que el sistema funcione correctamente. Tras cada operación el sistema debe quedar consistente y ausente de errores.
La etapa de pruebas tiene un papel más importante de lo que se le acostumbra a dar. No solamente abarca los errores y fallos de programación.

También es una etapa en la que deben comprobarse ciertos aspectos más subjetivos. La dificultad de uso en los diseños de pantalla puede hacer de un buen proyecto un fracaso absoluto. Por ello, es necesario comprobar que el sistema es usable e intuitivo para los usuarios a los que va dirigido.

Para realizar un análisis exhaustivo es recomendable diseñar los juegos de pruebas de antemano. Los juegos de pruebas se suelen dividir en tres tipos: las pruebas funcionales, las pruebas de usabilidad y las pruebas de rendimiento. A continuación se explicará las pruebas que se han hecho y los resultados obtenidos.

Pruebas funcionales

Las pruebas funcionales son las que pretenden comprobar el correcto funcionamiento del sistema. Para ello hay que comprobar todas las funcionalidades que hay en el sistema.

Si un caso de uso funciona correctamente debe dejar el sistema estable. Un caso de uso esta formado por varias entidades que se relacionan entre sí. Por esta razón es recomendable empezar por asegurar el funcionamiento de estas entidades más pequeñas antes de comprobar el caso de uso completo. Estas comprobaciones se hacen mediantes pruebas unitarias.

Las pruebas unitarias han servido para verificar el correcto funcionamiento de una determinada función, independientemente del resto. Su importancia radica en que estas pruebas se suelen realizar de manera automatizada por las herramientas de desarrollo.

Una vez comprobado el funcionamiento de los métodos implementados, se deberán realizar pruebas de integración. Estas pruebas consisten en comprobar la validez de procesos que implican relaciones entre entidades. Estas pruebas son las que se utilizarán para la comprobación de todas las funcionalidades del sistema.

La mayoría de las pruebas funcionales se han llevado a cabo durante la implementación. A medida que se añaden nuevas funcionalidades, se prueban de forma exhaustiva, comprobando el correcto funcionamiento de cada una de las opciones posibles que puede realizar el usuario. A continuación se detallan algunas de las pruebas realizadas en el sistema.

Inserción de una nueva regla

Se ha insertado una nueva regla en el sistema desde la pantalla de insertar. El sistema ha guardado el nombre de la regla en la tabla Reglas.

Modificación de una regla

Desde la pantalla de modificar regla se ha modificado la implementación de una regla existente. El sistema ha modificado el nombre que identifica la regla en la tabla Reglas.

Eliminación de una regla

Se ha eliminado del sistema una regla existente desde la pantalla de eliminar regla. El sistema ha eliminado la entrada de la tabla Reglas que identifica a la regla que se ha eliminado.

Calculo de probabilidades

En la pantalla de calculo de probabilidades se ha calculado la probabilidad de que dos personas guardadas en la base de datos sean la misma. Después de insertar los datos necesarios, el sistema ha aplicado las reglas y mostrado los resultados obtenidos.

Pruebas de usabilidad

Las pruebas de usabilidad tienen como objetivo demostrar si un sistema puede ser usado de forma correcta por un usuario. Normalmente, el creador del sistema realizar también las pruebas de usabilidad, pero no se puede considerar una prueba fiable, ya que conoce el uso del sistema antes de probarlo. Por esta razón estas pruebas deben realizarse sobre usuarios que no conozcan la herramienta.

El propósito de las pruebas realizadas para este proyecto es saber si la navegación y distribución de los elementos de las pantallas son sencillos de usar. Se ha pedido a un usuario con conocimientos básicos de informática y se le ha pedido realizar las siguientes tareas:

- Insertar una regla
- Modificar una regla
- Eliminar una regla
- Ejecutar el algoritmo de calculo
- Ver los resultados de los cálculos
- Ver los detalles de los cálculos

Para analizar los resultados se puede comprobar si han usado el camino más corto contando el número de clics o el tiempo total dedicado a conseguir cada una de las tareas.

Los usuarios han sido capaces de encontrar todas las tareas de manera más o menos rápida. Se ha detectado que a la hora de ver los detalles de un calculo, no se le da al usuario toda la información necesaria para que pueda interpretarlo de forma correcta. Estas pruebas han permitido detectar el problema y resolverlo de forma adecuada.

Pruebas de rendimiento

Las pruebas de rendimiento son aquellas que pretenden demostrar que el sistema es estable y fiable con una carga alta de trabajo.

En el sistema, como solamente cuenta con un usuario, la carga de trabajo reside en la utilización de reglas. Para realizar las pruebas se ha ejecutado el algoritmo de cálculo en diferentes escenarios:

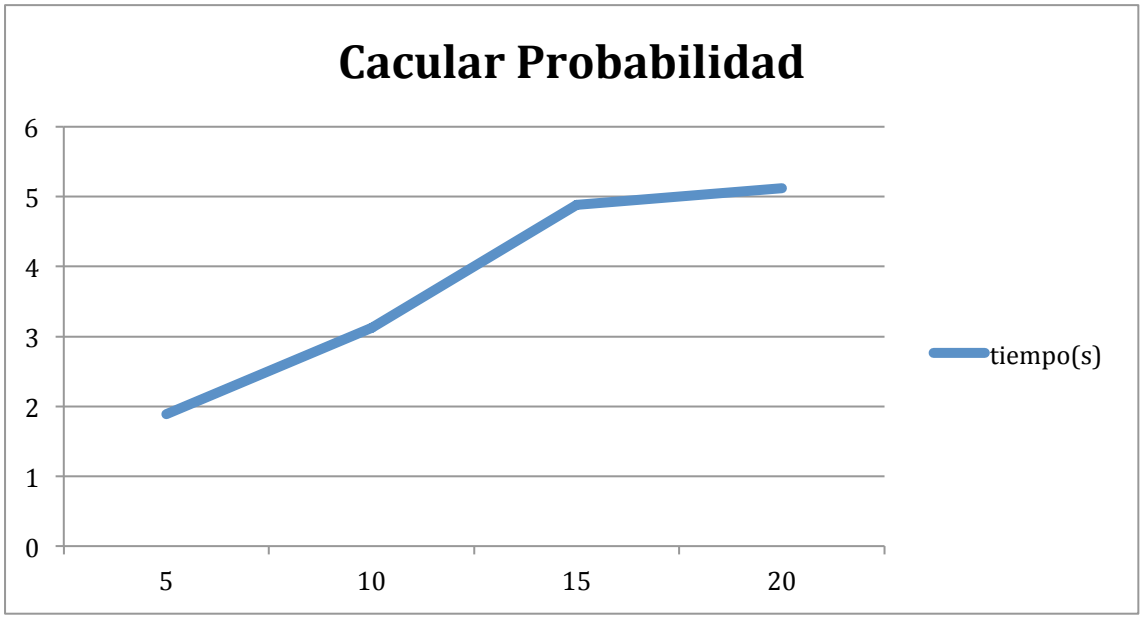
- Un escenario con carga de trabajo nula (5 reglas)
- Un escenario con carga de trabajo media (10 reglas)
- Un escenario con carga de trabajo media (15 reglas)
- Un escenario con carga de trabajo alta (20 reglas)

Con estos escenarios se ha medido el tiempo de las diferentes tareas realizando cada una de ellas 10 veces para una medición más fiable.

Tras realizar varias veces las pruebas en para un número determinado de reglas y calcular la media aritmética se han obtenido los siguientes resultados:

Numero de Reglas	**5**	**10**	**15**	**20**
Tiempo	1,88	3,12	4,87	5,11

Resultados de la Prueba de Rendimiento



Gráfica de la prueba de rendimiento

Se puede observar que a medida que las reglas aumenta también se incrementa el tiempo en que tarda el calculo de las probabilidades. Se aplica más reglas a cada una de las parejas para calcular que probabilidad tienen de que sea la misma persona.

Otro aspecto interesante es que el tiempo también depende de la complejidad de las reglas. La diferencia de tiempo entre aplicar 15 y 20 reglas no es tan grande como debería. La razón de esto es por la complejidad de las reglas. Si las reglas fueran iguales de compleja se vería un incremento lineal.

Conclusiones y trabajo futuro

Este apartado concluye la memoria de este proyecto. Se realiza una valoración personal, comentando los conocimientos adquiridos, problemas encontrados y otros detalles. También se comenta el futuro del proyecto, valorando opciones que se han quedado fuera del proyecto y las posibilidades de ampliación.

Conclusiones y valoración personal

Durante este proyecto he podido ver gran parte del ciclo de vida de un proyecto, desde su gestación hasta su finalización, exceptuando la parte de mantenimiento. Gracias a este proyecto he podido poner en practica los conocimientos adquiridos durante la carrera y también mejorado algunas de ellas. Este proyecto ha sido una forma de comprobar mi nivel de autonomía y constancia. Aunque ha sido un aspecto que ha ido mejorando durante el proyecto, en mi opinión, debo desarrollar más.

Planificar y gestionar el proyecto me ha ayudado a tener más experiencia. He podido mejorar aspectos como la previsión de tiempo e identificar tareas, a parte de las que hay en la mayoría de los proyectos. Si tuviera ocasión de hacer el mismo proyecto o uno similar, tendría en cuenta aspectos que no antes no había tenido en cuenta.

En la parte de especificación y diseño he podido aplicar todos los conocimientos adquiridos en distintas asignaturas durante la carrera relacionados con esta etapa. Esta parte ha sido la parte más complicada. La complejidad reside en que es un proyecto con pocos antecedentes y no habían muchos proyectos en las que me podía basar. Se ha tenido que pensar mucho en la representación de los datos, y en la forma de hacer los cálculos. Se ha discutido mucho el modelo que tiene que utilizar la herramienta y se ha tenido que mirar las desventajas y ventajas que tenían cada uno.

La implementación ha sido una tarea más sencilla. Probablemente el calculo de probabilidades ha sido la más compleja, ya que he tenido que encontrar una forma de que la reglas se puedan incluir automáticamente, sin tener que ir modificando el código a

medida que se creaban nuevas reglas. Esta etapa me ha servido para mejorar mis conocimientos del lenguaje SQL y PHP.

Antes de hacer el proyecto mis conocimientos de PHP eran básicas, pero al leer documentación relacionada con PHP, he descubierto librerías o funciones estándares que desconocía que han sido necesarios para la implementación de la aplicación.

La etapa de testeo me ha permitido aprender como enfocar y preparar las pruebas. He aprendido los diferentes tipos de pruebas necesarias para comprobar el funcionamiento y usabilidad del sistema y a ponerlos en practica.

Trabajo Futuro

Durante el proyecto han ido surgiendo ideas que se podían realizar, pero por falta de tiempo o porque no era objetivo del proyecto no se han hecho. Estas tareas se puede realizar en trabajos futuros.

Nuevas funcionalidades

Se pueden añadir las funcionalidades para que el usuario pueda insertar los datos de las personas. En esta herramienta se trabaja con una base de datos que tiene la estructura necesaria con datos que se han insertado desde el gestor de base de datos que se ha utilizado.

Mejorar el cálculo de Probabilidades

En la herramienta se pueden hacer mejoras en el calculo de probabilidades. Actualmente la herramienta cuenta con un algoritmo que no tiene en cuenta la fiabilidad que tiene un atributo dependiendo de la fuente documental de donde proviene.

No es lo mismo la fecha de nacimiento en una partida de bautismo que en una partida de casamiento. Una mejora para el calculo de probabilidad es tener en cuenta estos documentos.

Convertirlo en Herramienta Online

Aprovechando que esta en PHP se podría convertir en una herramienta online, donde todos puedan exportar o importar bases de datos con las que trabajan, siempre que tengan la misma estructura que la usada en la herramienta. De esta manera la tarea de compartir reglas es más fácil.

Habría que tener en cuenta que el sistema actual esta pensada para una persona y que habría que tener en cuenta aspectos de seguridad que no se verifican.

Bibliografía y referencias

Bibliografía

Lledó Plana, Pau, (2010).
Integració d'eines pel buidatge de fonts documentals d'interés genealògic.

Generelo Perulero ,Pedro Lucas, (2010).
Modelización de fuentes documentales civiles de interés genealógico.

Altés Martí, Jordi, (2009).
Modelització de fonts documentals d'interès genealògic.

Sklar, D (2005).
Introducción a PHP 5. Anaya Multimedia

DuBois, Paul; Hinz, Stefan; Pedersen, Carsten. (2005).
MySQL 5.0 Certification Study Guide)

Referencias

Archivos y documentos webs

Documentación online de MySQL v 5.5: Contiene toda la documentación de la sintaxis de consultas SQL y funciones que tiene. Además se puede encontrar ejemplos de cómo utilizar cada una de ellas.
http://dev.mysql.com/doc/refman//5.5/en/insert.html

Documentación online de PHP. Contiene toda la documentación de sus clases y funciones con descripciones exhaustiva y comentarios y ejemplos de usuarios.
http://www.php.net/manual/en/

Sitio web del World Wide Web Consortium (W3C). Se encuentran tutoriales de diferentes lenguajes de la web (HTML, CSS Ajax) aplicando los estándares que recomienda la W3C, que tiene como objetivo mejorar la accesibilidad de los usuarios.
http://www.w3schools.com/css/

The Gedcom standart Release 5.5. Es un anexo donde se encuentra un glosario de todos los tags que se utiliza dentro de un archivo GEDCOM.
http://homepages.rootsweb.ancestry.com/~pmcbride/gedcom/55gcappa.htm

Fuentes genealógicas online

Sitio web de Family Search. Es una web donde se puede encontrar diferentes fuentes documentales de interés genealógico.

https://familysearch.org/

Página web de la Societat Catalana de Geneologia Heràldica, Vexil·llogia i Nobiliària. En ella se puede encontrar diferentes documentos de interés genealógico.
http://www.scgenealogia.org/

ANEXOS

Instalación y Ejecución

Instalación

Para poder instalar la aplicación, se tienen que cumplir los siguientes requisitos:

- Servidor Apache
- PHP v5.2.4 o superior
- MySQL v5.0

Para instalar y configurar el servidor web en iOs v10.7.3 se puede instalar el paquete MAMP o MAMP Pro para Mac.

MAMP (iOS)

MAMP es una aplicación que permite montar un servidor local. De forma rápida y sencilla se dispone de Apache, MySQL y PHP.

Para este proyecto se ha utilizado la versión Pro, aunque también se puede utilizar la versión normal.

La instalación consiste en seguir los pasos hasta terminar la instalación. Para la instalación se requiere bajarse el ejecutable en la página http://www.mamp.info/en/index.html

Instalación en Linux (Ubuntu)

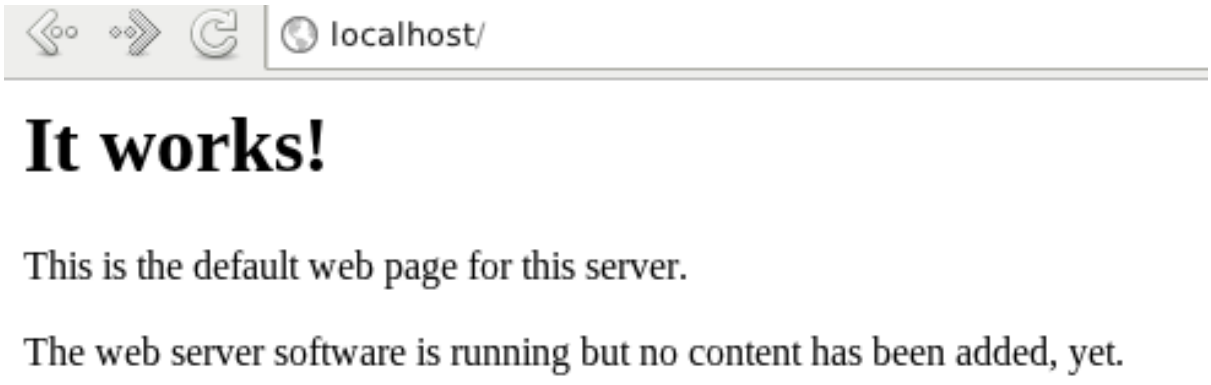
Instalar Apache2

Para instalar Apache2 a través del repositorio de Ubuntu, se ejecutará el siguiente:

apt-get install apache2

Si el sistema pide conformación hay que aceptar. Después descargará los paquetes necesarios para instalarlos y crear una configuración por defecto.

Si la instalación ha tenido éxito, accediendo desde el navegador a la dirección local http://localhost se obtiene una imagen similar a la siguiente:



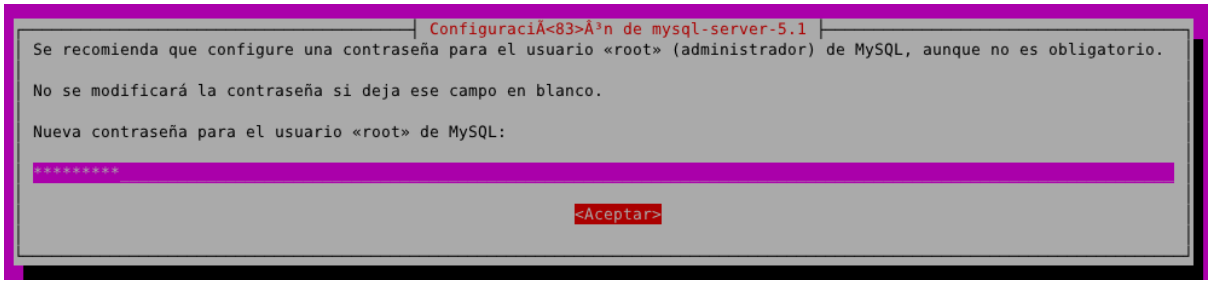
Página de inicio de Apache Server

MySQL

Para instalar el servidor de base de datos hay que ejecutar el siguiente comando en el terminal :

apt-get install mysql-server

Durante la instalación y configuración el sistema solicitará una contraseña para el usuario administrador (root) de la base de datos y continuará con la configuración.

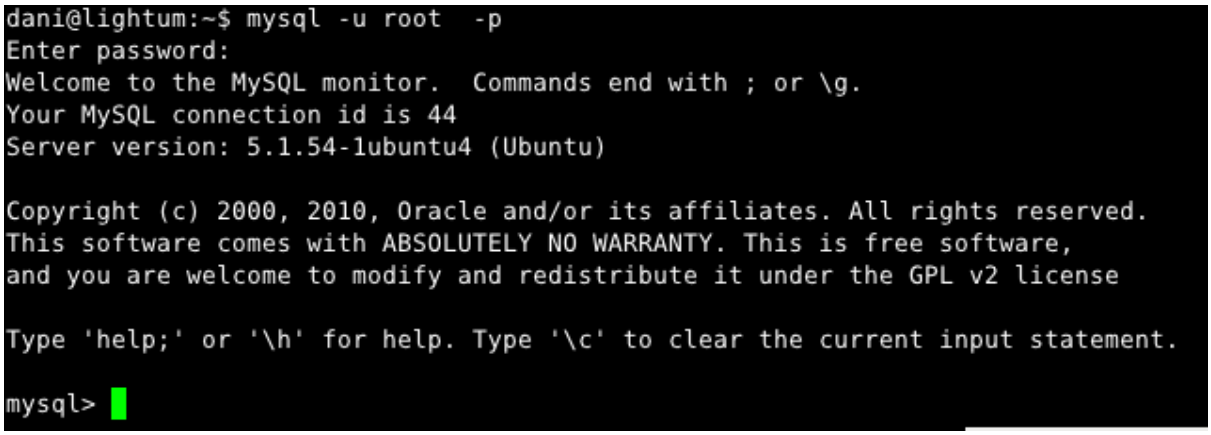


Configuración de MySQL

Para comprobar al correcta instalación y configuración, se ejecuta el siguiente comando:

Mysql –u root –p

El sistema nos pedirá la contraseña. Una vez introducida la contraseña propuesta anteriormente deberíamos ver algo similar a la siguiente imagen.



Cliente MySQL

Para salir solo hay que escribir **quit;** .

PHP

Para instalar el lenguaje PHP hay que ejecutar el siguiente comando:

apt-get installa php5

La instalación detectará los servidores webs instalados y realizar las acciones necesarias para que PHP funcione correctamente en cada uno de ellos. Será necesario reiniciar el servidor web apache para que cargue correctamente las nuevas configuraciones.

Para comprobar su correcta instalación se puede crear un fichero PHP y situarlo en el directorio web.

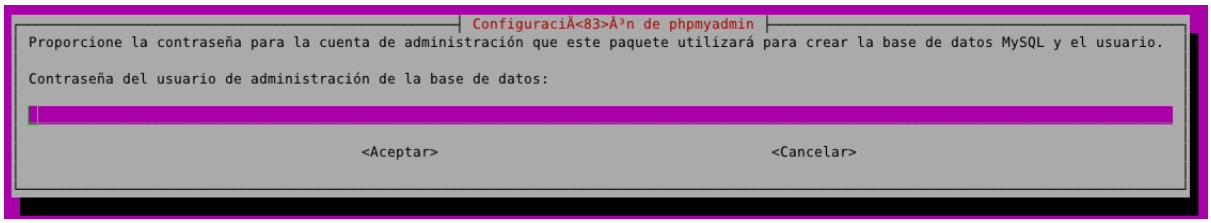
PhpMyAdmin

Esta herramienta nos permite gestionar la base de datos de MySQL. Esta herramienta consiste en una web PHP que realiza peticiones a la base de datos. A parte de facilitar la gestión de base de datos, nos ayudará a comprobar que PHP se ha instalado correctamente. Para su instalación ejecutaremos el siguiente comando:

apt-get installa phpnyadmin

Durante la instalación, se nos solicitará que indiquemos en qué servidor web deseamos configurar phpMyAdmin. También se nos preguntará si queremos configurar phpMyAdmin con db-config-common (contestaremos que sí).

Para crear la configuración se necesitará acceder a la base de datos MySQL por lo que el sistema nos pedirá la contraseña del administrador.



Configuración phpMyAdmin

Para comprobar el correcto funcionamiento tanto de PHP como de phpMyAdmin, accederemos a la dirección http://localhost/phpmyadmin desde el navegador web.

Configurar el servidor

MAMP y Linux

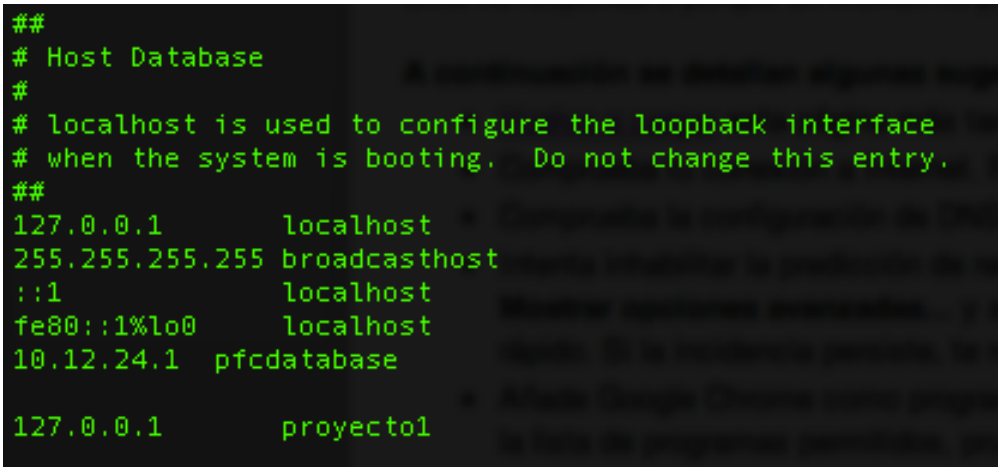
Para poder utilizar la herramienta es necesario configurar el servidor.
Primero necesitaremos modificar el nombre del servidor para que responda a **http://pfcdatabase** . Para esto tendremos que hacer 2 cosas :

- Insertar una nueva entrada en el archivo de host del ordenador
- Configurar apache para que acepte Servidores virtuales y configurar el servidor virtual que estamos definiendo.

Primero tenemos que modificar el archivo /etc/hosts. En ella introducimos una nueva entrada:

127.0.0.1 pfcdatabase

En el fichero quedaría de la siguiente manera :



contenido del fichero hosts

Modificamos el archivo de configuración de apache. En MAMP se encuentra en el siguiente directorio: /Applications/MAMP/conf/apache/httpd.conf .

En el caso de Linux se encuentra en el fichero donde se ha instalado apache.
Introducimos esta línea al final de todo

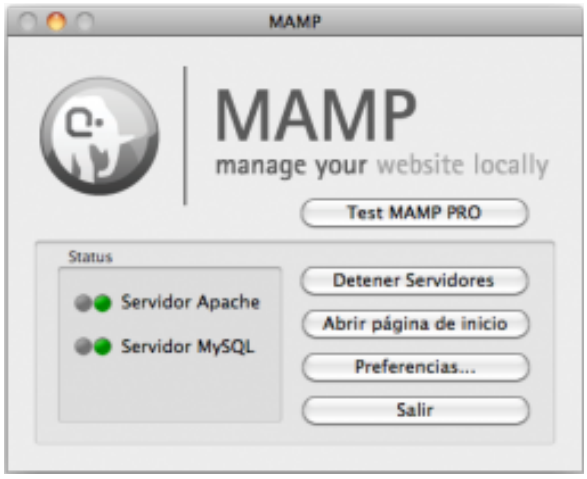
```
1. NameVirtualHost *
2. <VirtualHost *>
3. DocumentRoot "PATH__________/htdocs"
4. ServerName localhost
5. </VirtualHost>
6. <VirtualHost *>
7. DocumentRoot "PATH_______/pfc/httpdocs"
8. ServerName pfcdatabase
9. </VirtualHost>
```

Las cinco primeras líneas del código activa los servidores virtuales en apache y hace de la carpeta htdocs como el directorio por defecto de http://localhost. En la linera siete tenemos que poner la dirección de la carpeta hasta httpdocs. PATH____ es la dirección hasta llegar a la aplicación, que en nuestro caso es pfc/httpdocs

Una vez cambiado y guardado estos dos archivos, si todo ha ido correctamente al introducir la dirección http://pfcdatabase se cargará correctamente la aplicación.

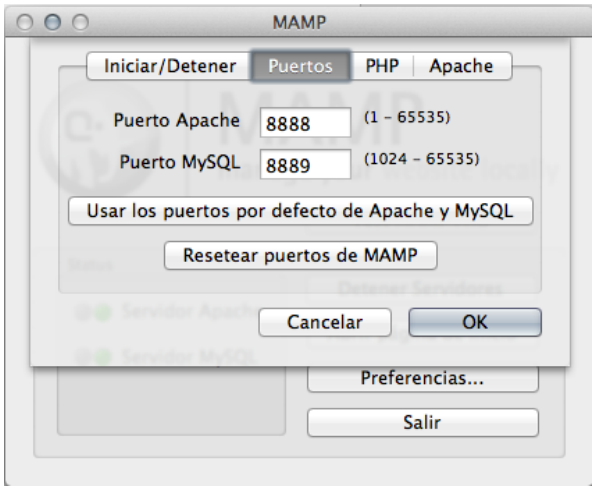
Si utilizamos MAMP, hay que modificar también el número de puerto con que escucha.
Por defecto, Apache escucha en el puerto 80, y MySQL en el 3306.
En los servicios que instala MAMP, estos puertos están cambiados. Apache está en el puerto 8888 y MySQL en el 8889. Para modificarlo tenemos que hacer lo siguiente.

Abrimos el panel de control de MAMP y seleccionamos preferencias.



Panel de Control de MAMP

Vamos a la pestaña puertos, y pinchamos sobre “Usar los puertos por defecto de Apache y MySQL“.

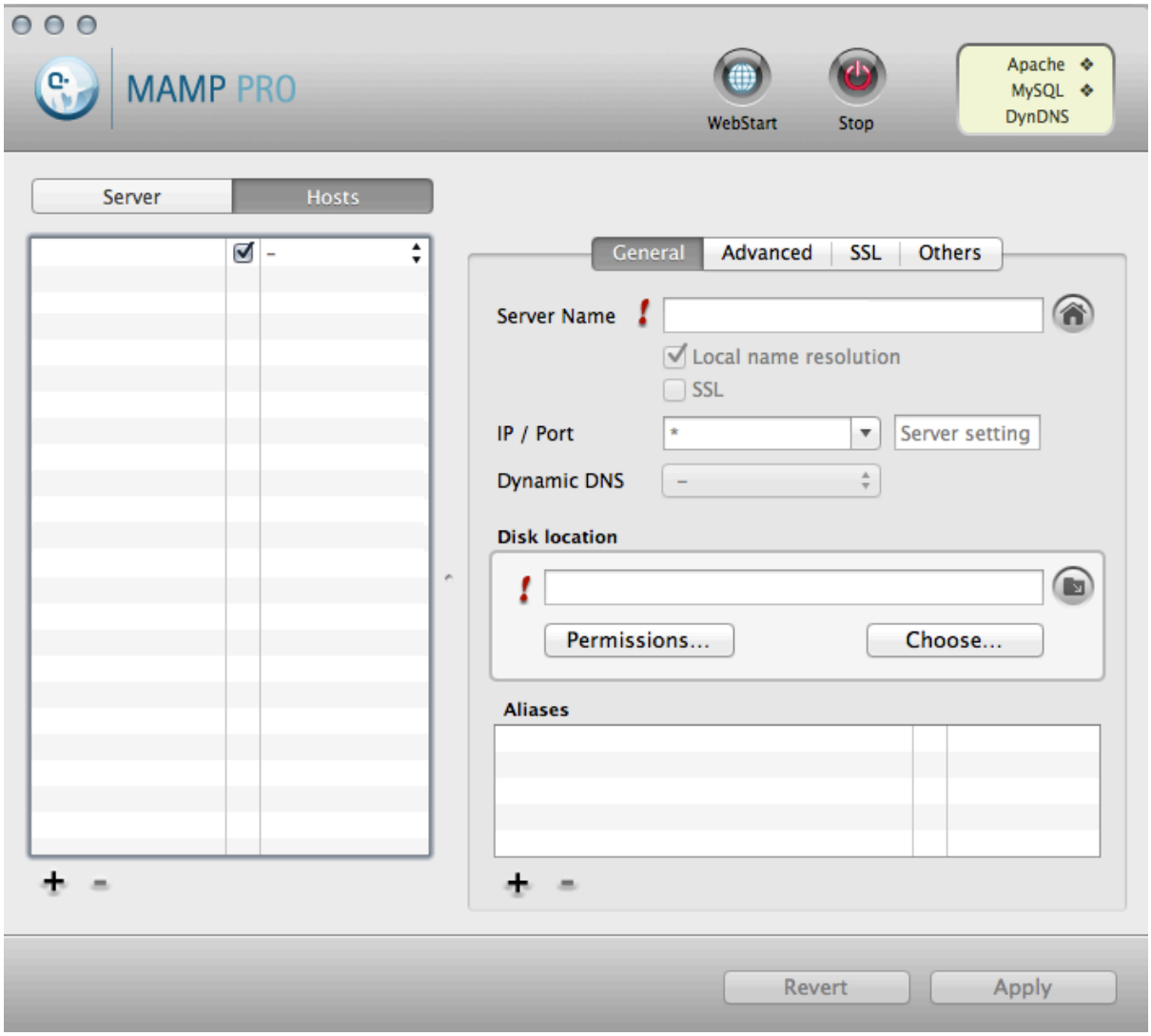


Ventana de Preferencias de MAMP

MAMP Pro

Con la versión de MAMP Pro, la tarea de crear servidores virtuales es más fácil.

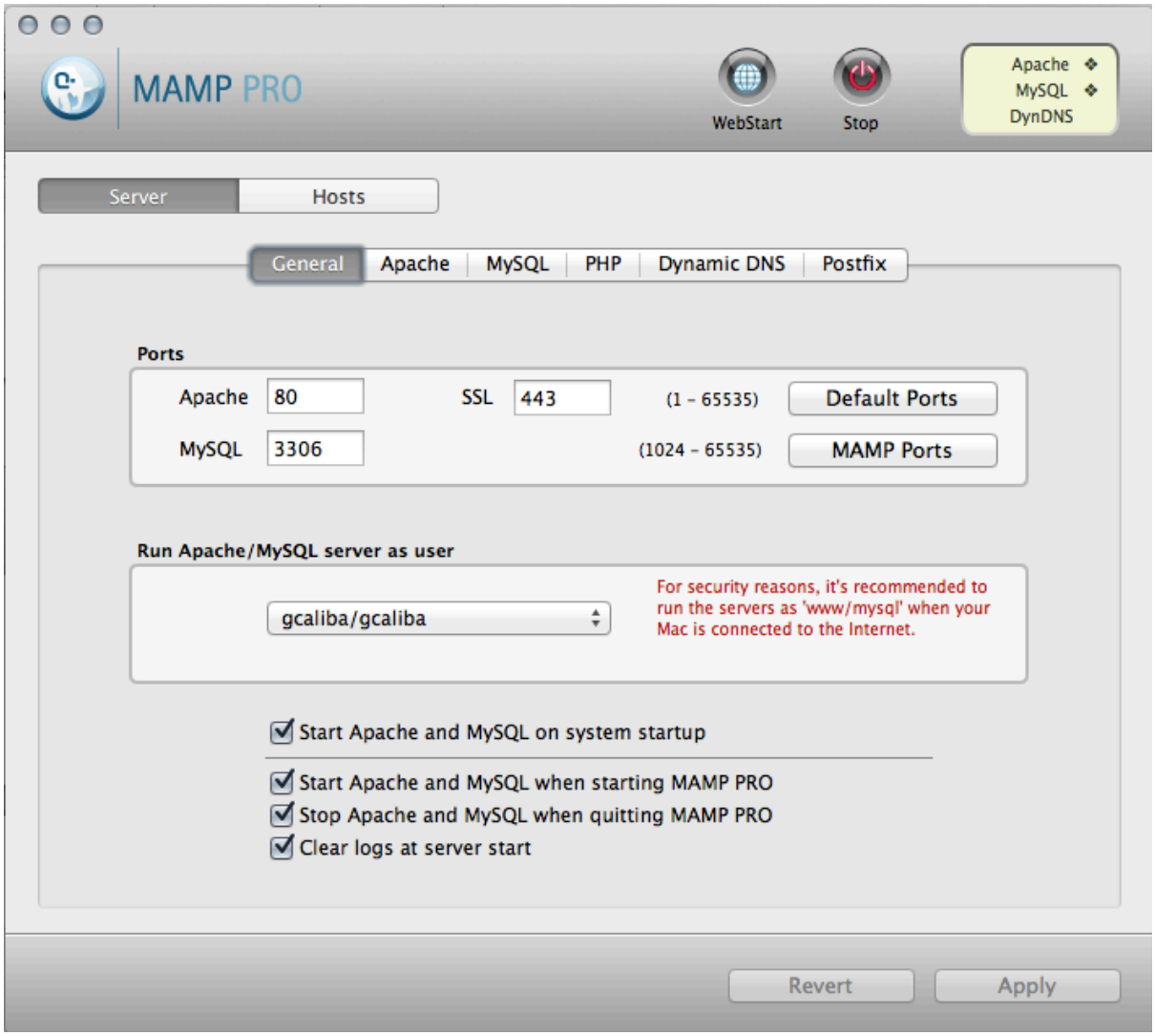
Primero tenemos que seleccionar la pestaña host del panel de control y clicamos el botón +



Panel de control de MAMPro

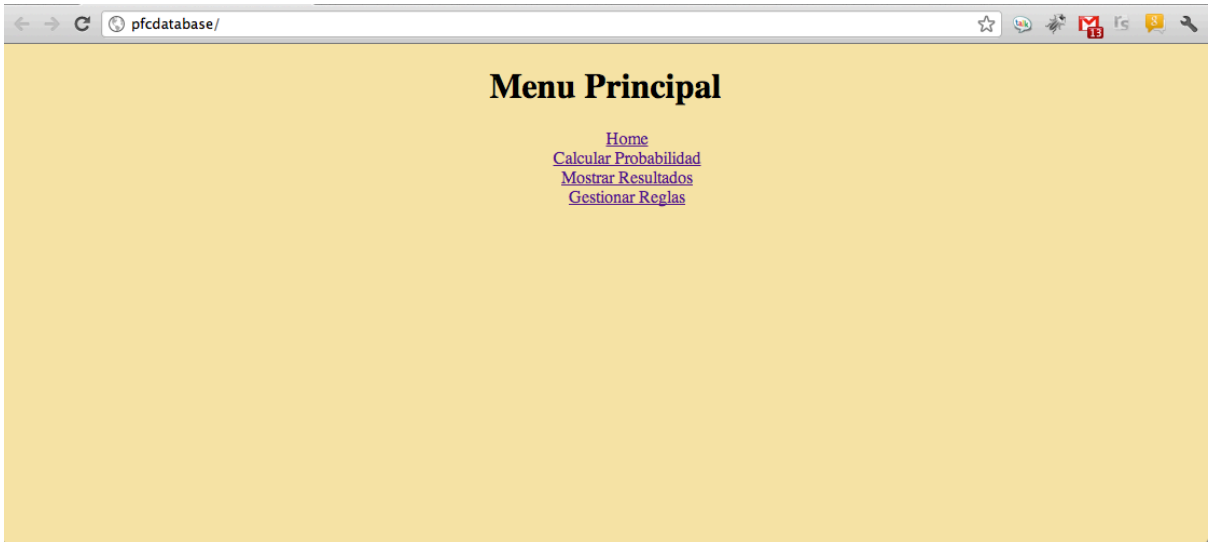
En los campos de *Server Name* indicamos el nombre del servidor, en nuestro caso pfcdatabase y en *Disk location* la carpeta donde se encuentra la aplicación. En nuestro caso la dirección tiene que llegar hasta App. Por ejemplo si la carpeta esta en /Users pondríamos /Users/pfc/httpdocs siendo pfc la carpeta que contiene toda la aplicación.

Como en MAMP tenemos que modificar los puertos que se utilizan por defecto. Para esto vamos en la pestaña Server , General e indicamos que se utilice los puertos por defecto a pulsando el botón Default Port



Utilización

Abrimos nuestro navegador e introducimos la siguiente dirección:
http://pfcdatabase/
Accederemos a la siguiente ventana:



Pantalla Principal

Gestionar Reglas

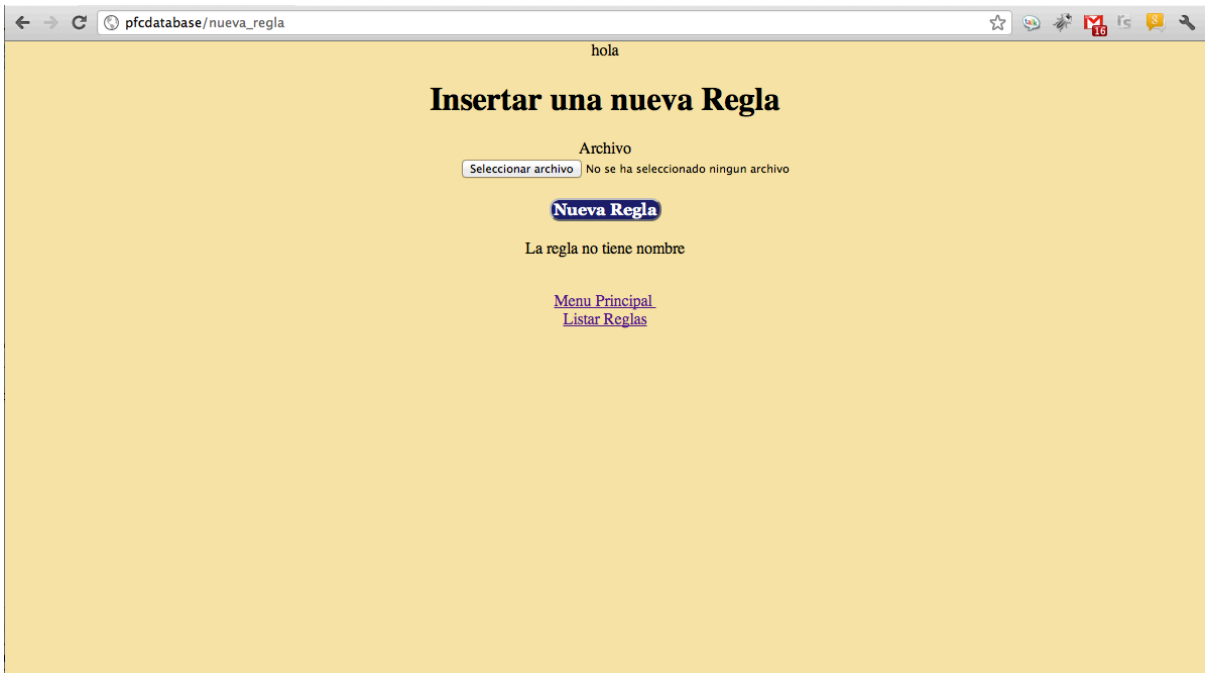
Cuando queramos añadir, modificar o eliminar una regla, desde el menú principal clicamos, "Gestionar Reglas". Nos saldrá una ventana similar a la siguiente:

Ventana de gestion de reglas

En el caso de que no hubiera ninguna regla, nos aparecería una lista vacía.

Añadir Regla

Para añadir una regla clicamos desde la ventana de gestión de regla la opción “Nueva Regla”. Una vez clicado, nos aparecerá una ventana con la siguiente, donde tendremos que indicar en que directorio esta la regla que queremos añadir.



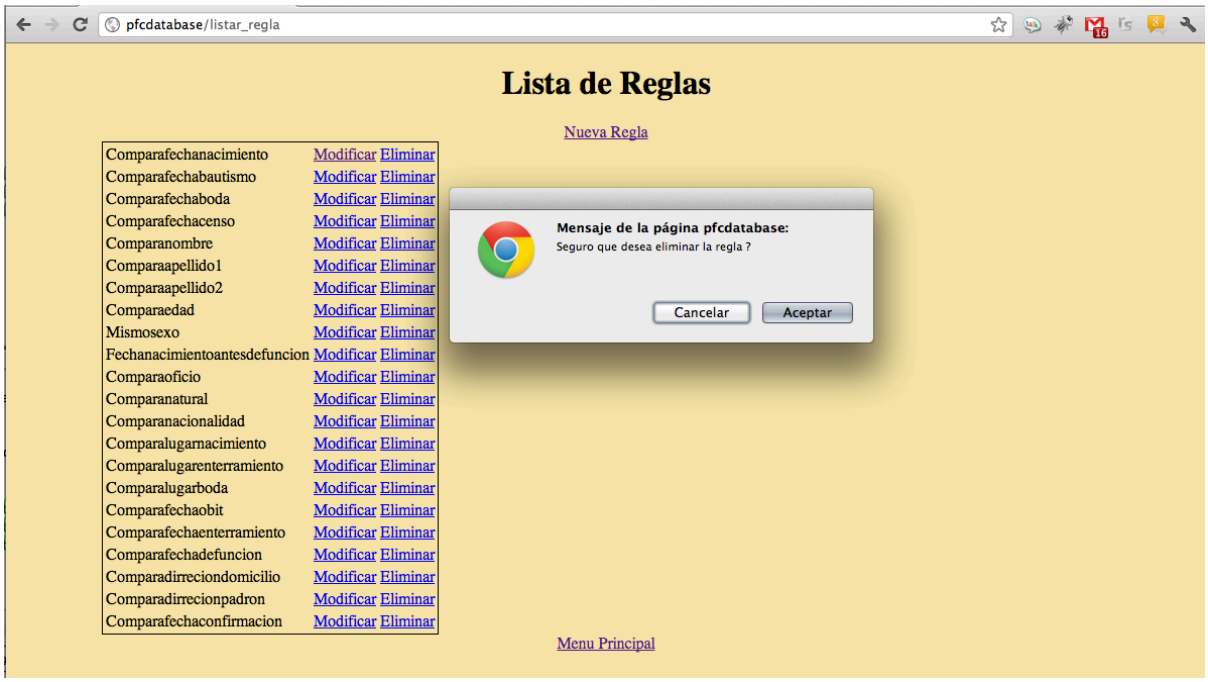
Ventana de insertar reglas

Modificar una Regla

La operación es similar al de añadir una regla.
Seleccionamos de la regla que queremos modificar, el link “Modificar” y nos aparecerá una ventana similar a la siguiente:

Ventana de modificar regla

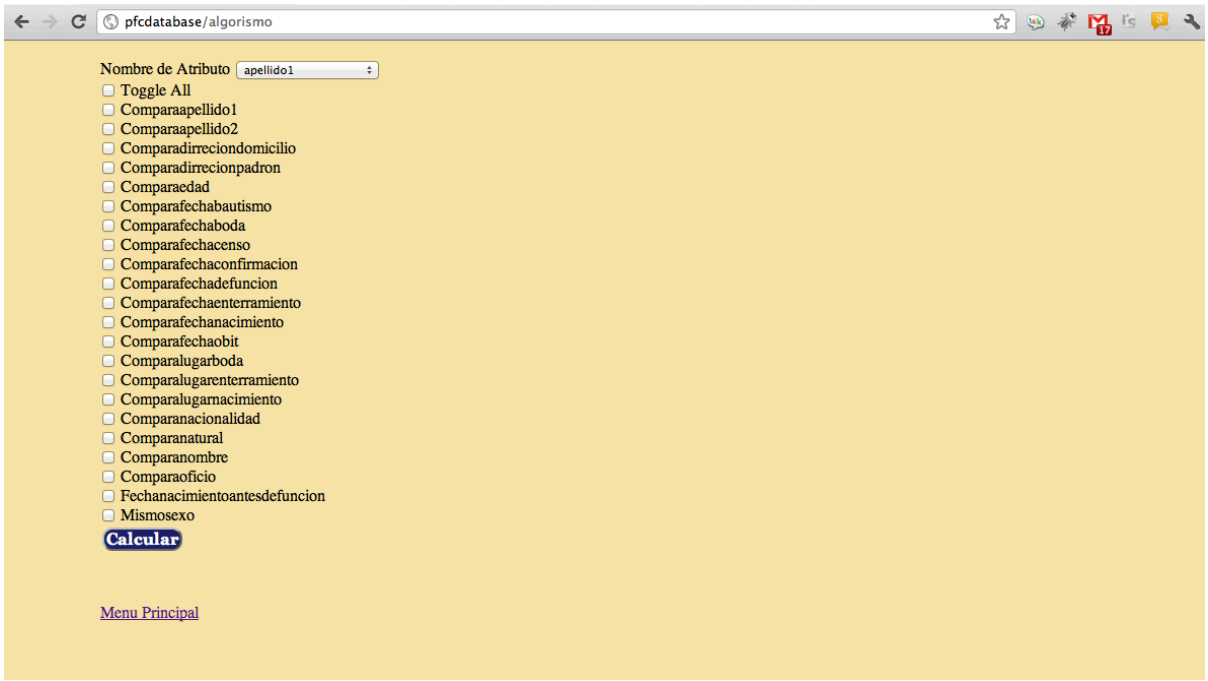
Tenemos que indicar que archivo substituirá a la regla actual.
Para finalizar clicamos en “Modificar”.



Ventana de eliminar regla

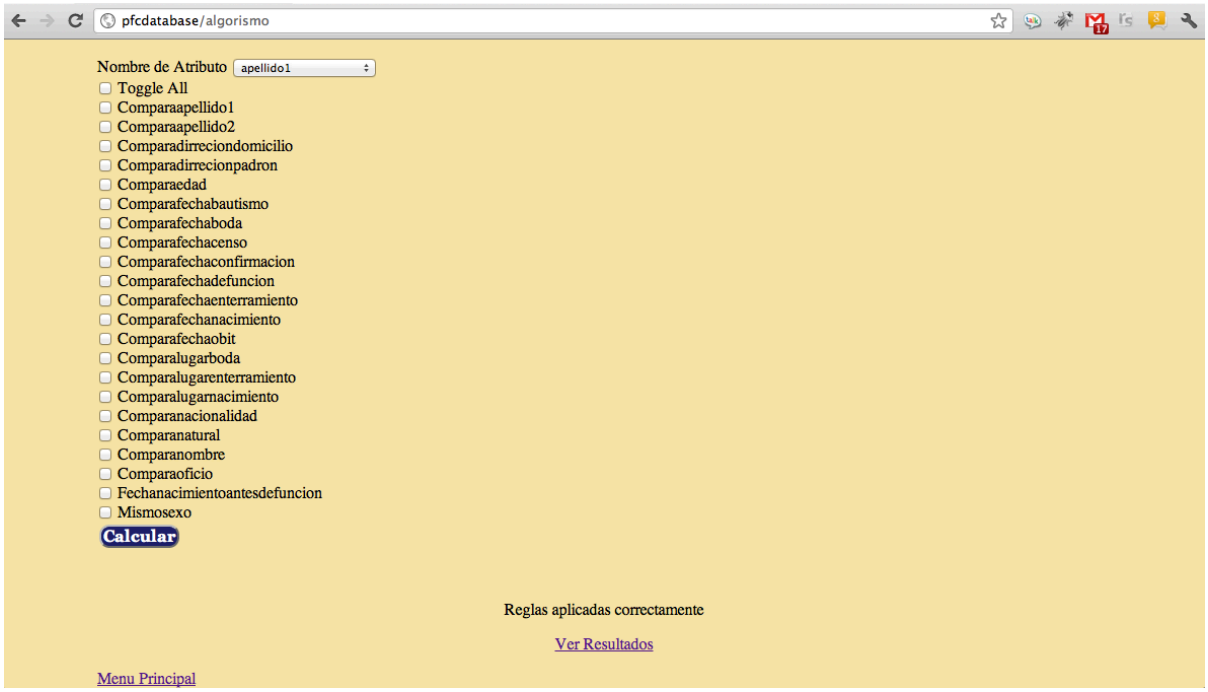
Calcular Probabilidad

A la hora de calcular la probabilidad de que dos personas sean la misma, clicamos en el link “Calcular Probabilidad” del menú principal. Una vez clicado, nos aparecerá una ventana como la siguiente:



Ventana de calculo de probabilidades

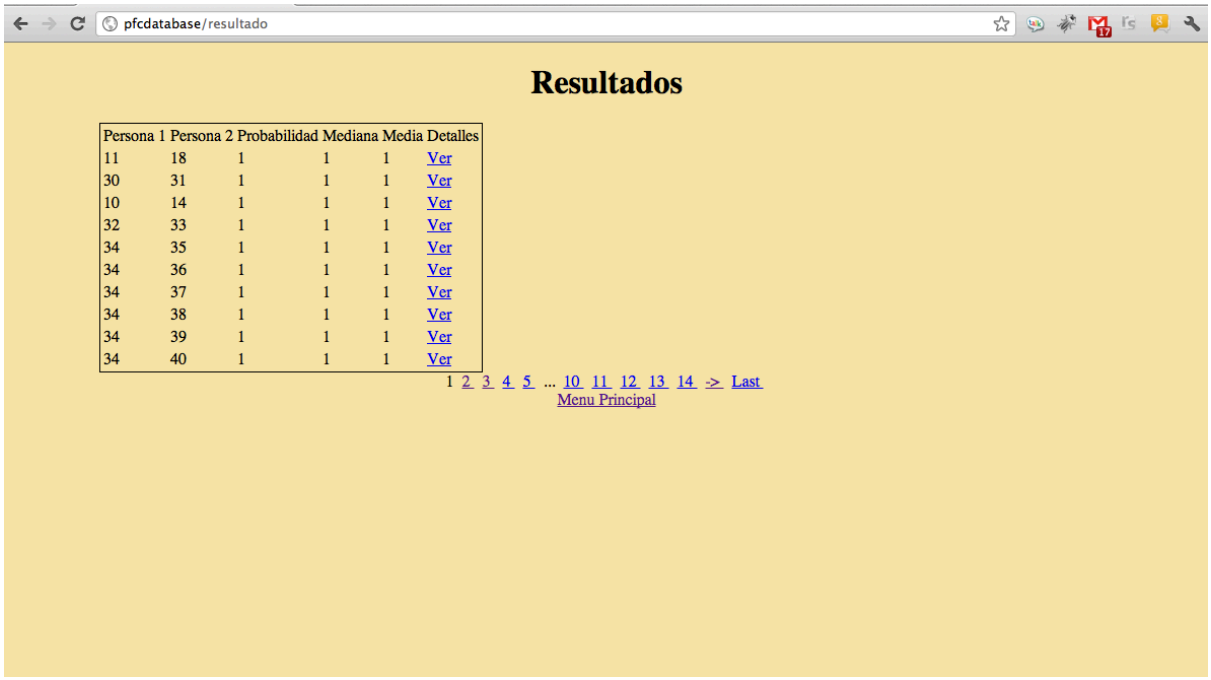
En ella tenemos que especificar que reglas se quiere utilizar y que atributo utilizará la herramienta para crear parejas entre las personas que hay registradas en la base de datos.
Clicamos en Calcular para ejecutar el proceso.
Cuando haya finalizado, nos aparecerá debajo un link indicando que el proceso ha terminado.



Ventana de calculo de probilidades (después de ejecutar)

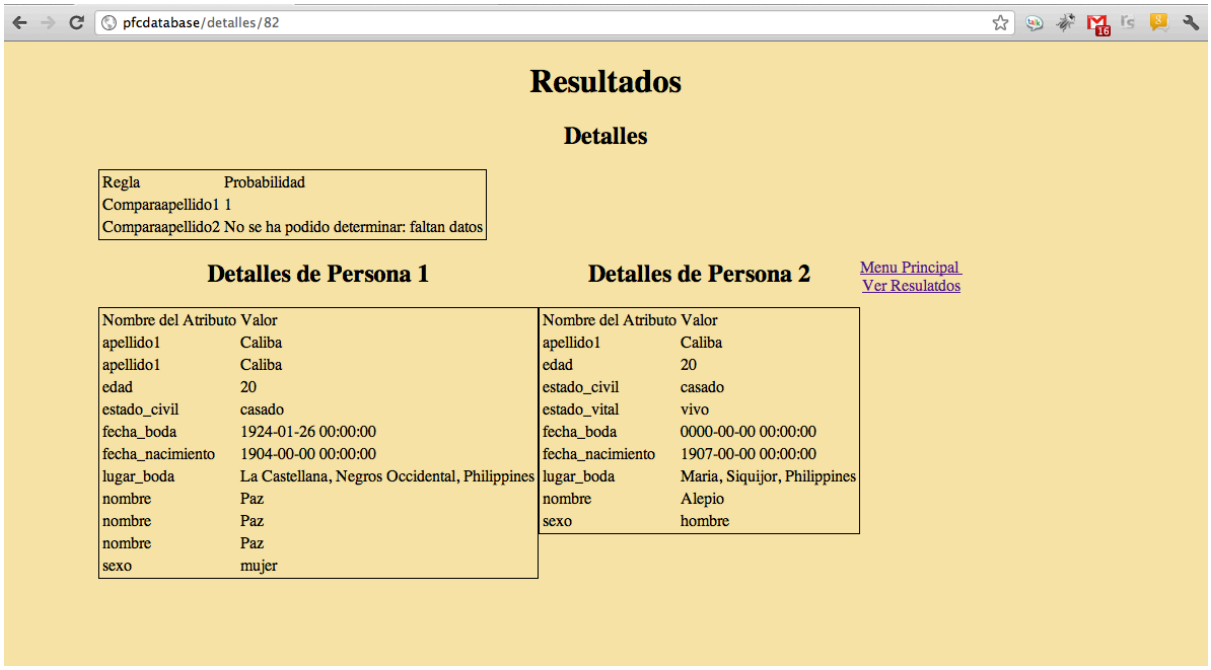
Ver Resultados

Una vez hecho calculado las probabilidades, se puede ver los resultados. Para acceder a los resultados tenemos dos opciones. Podemos clicar tanto el link “Ver Resultados” del menú principal como el que sale en la ventana de calcular probabilidad una vez finalizado los cálculos. En los dos casos nos aparecerá una ventana similar a la siguiente imagen:



Ventana de resultado

Cada fila representa una pareja con sus respectivos identificadores y resultados (Probabilidad, Mediana y Media). Para poder ver la información en que se han basado estos cálculos podemos clicar el link “Ver” de una pareja. Una vez seleccionado el link nos aparecerá una ventana como la siguiente:

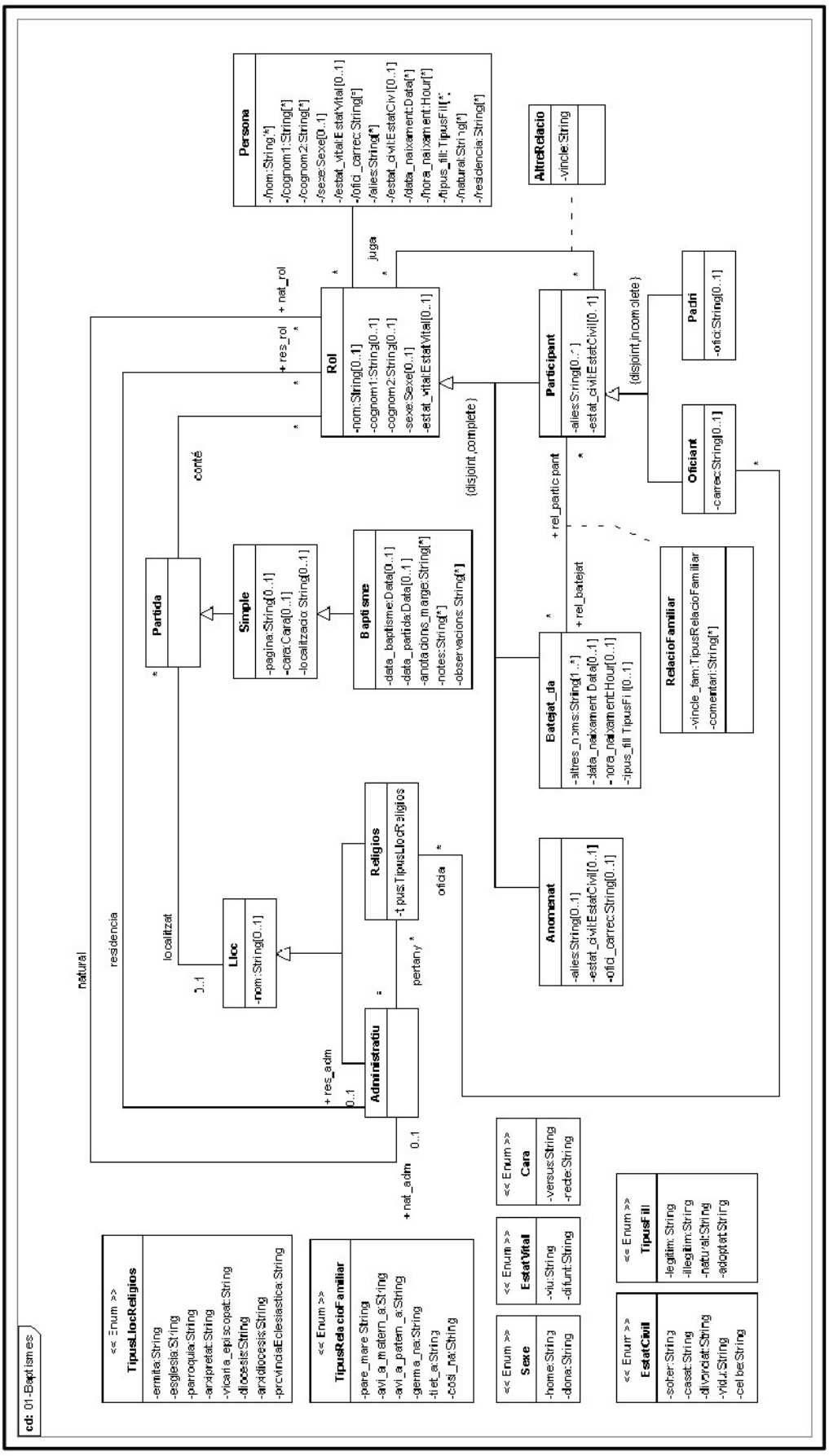


Ventana de Detalles

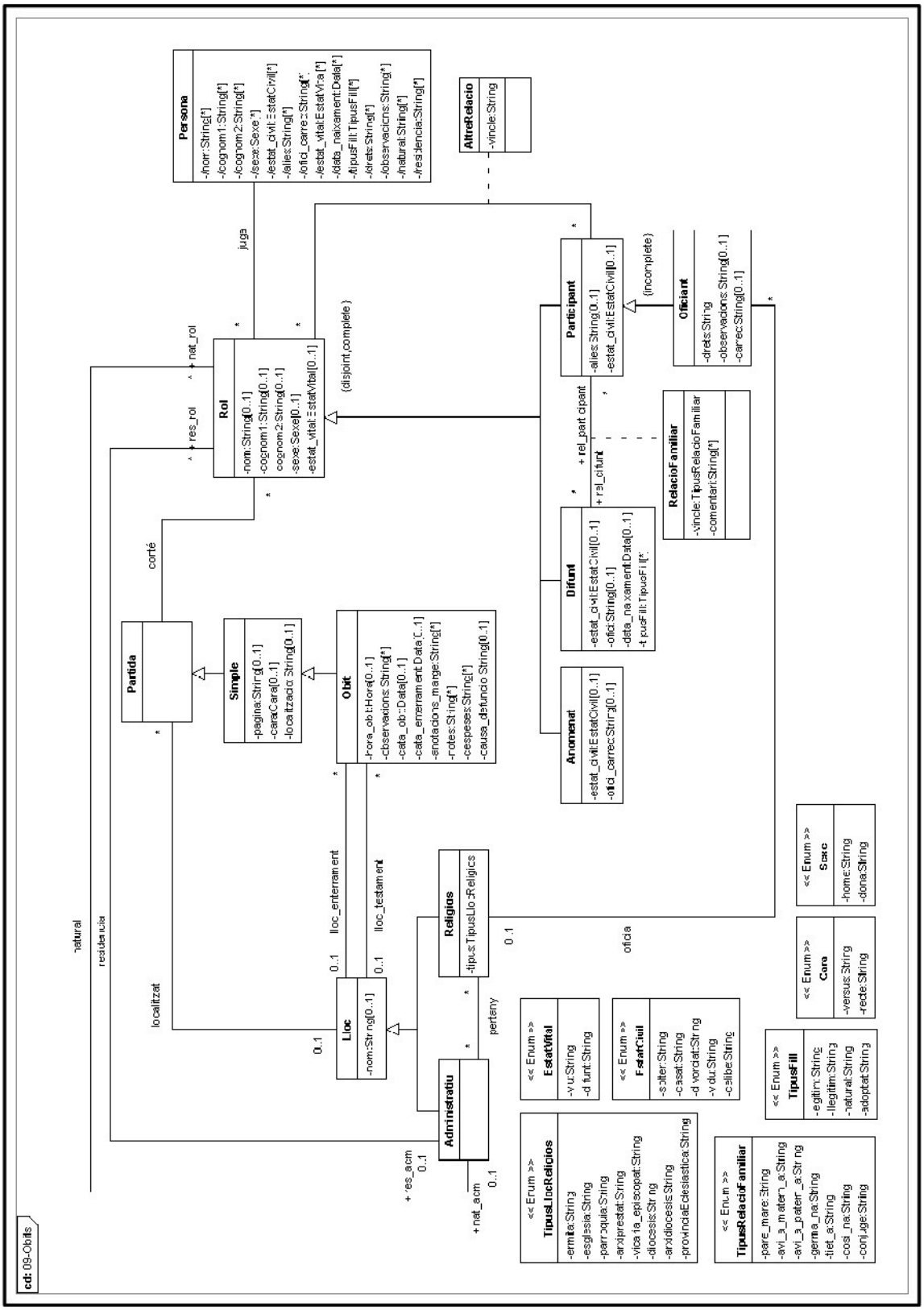
Diagramas Utilizados

A continuación se mostraran algunos de los modelos con que me he basado para obtener una lista de atributos y también para elaborar el diagrama que se ha utilizado finalmente para representar a una persona.

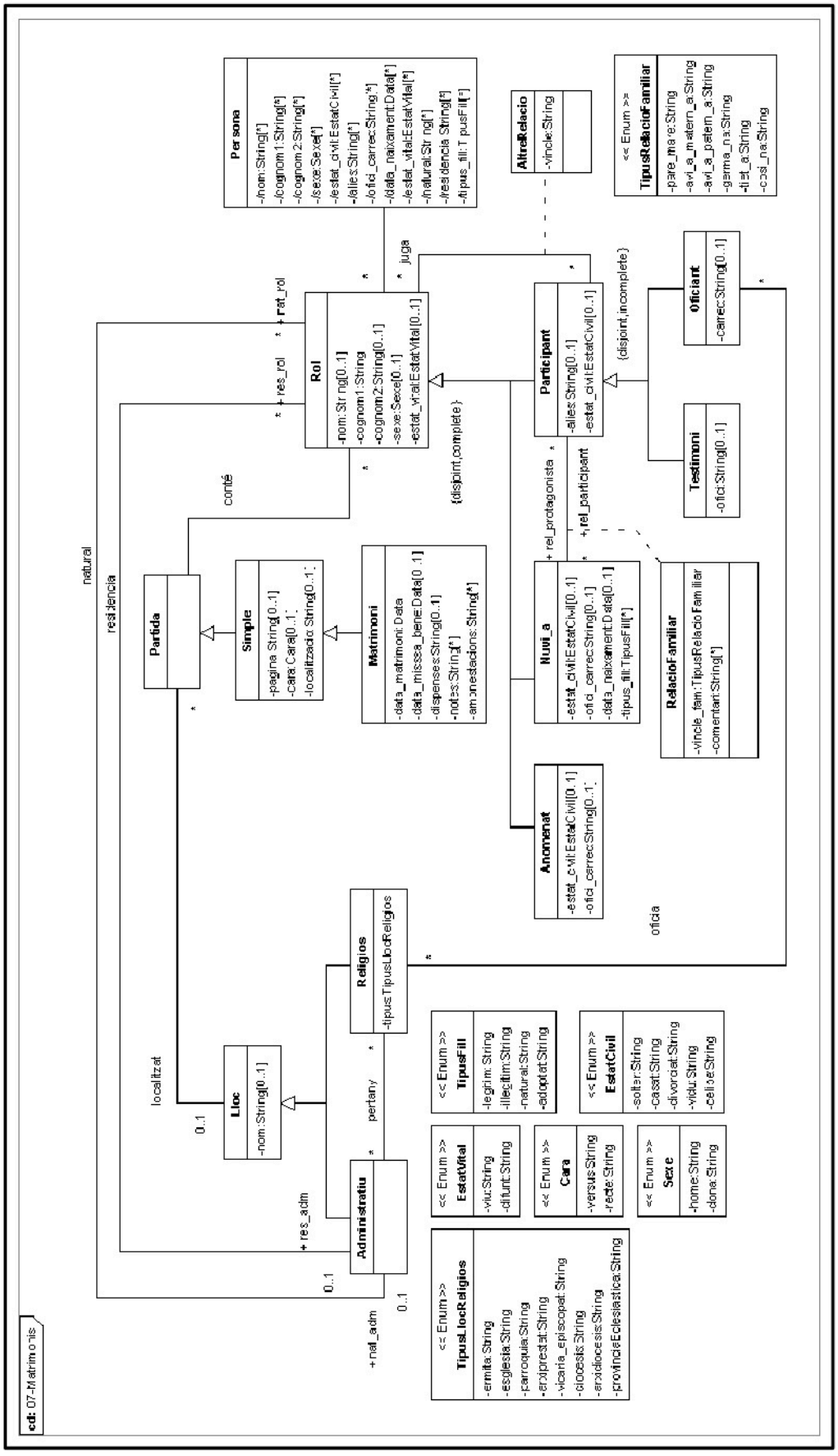
Los diagramas que se presentan a continuación pertenecen a los diagramas del proyecto de fin de carrera *Modelització de fonts documentals d'interés genealògic*.



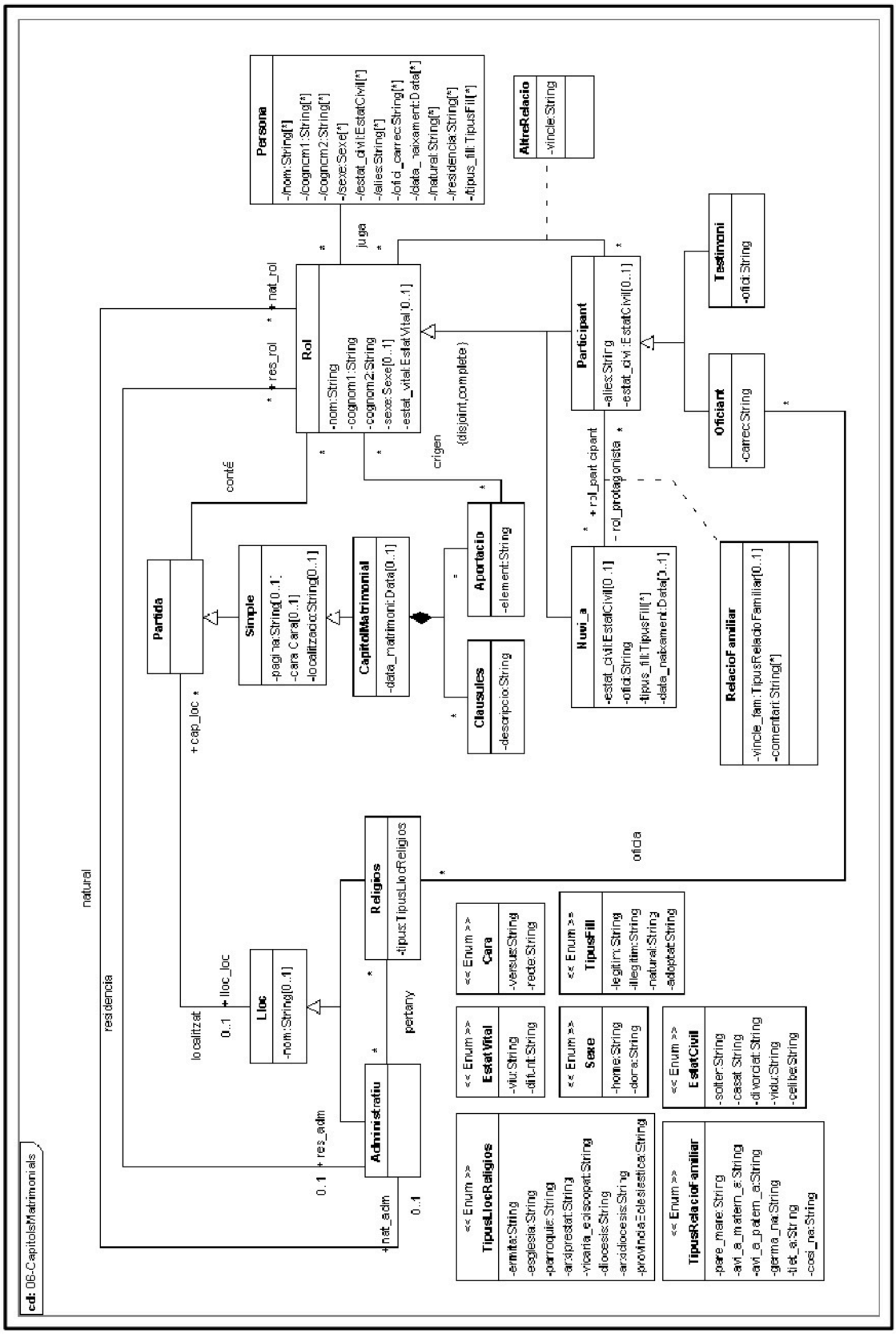
Modelo Conceptual de Partida de Bautismo



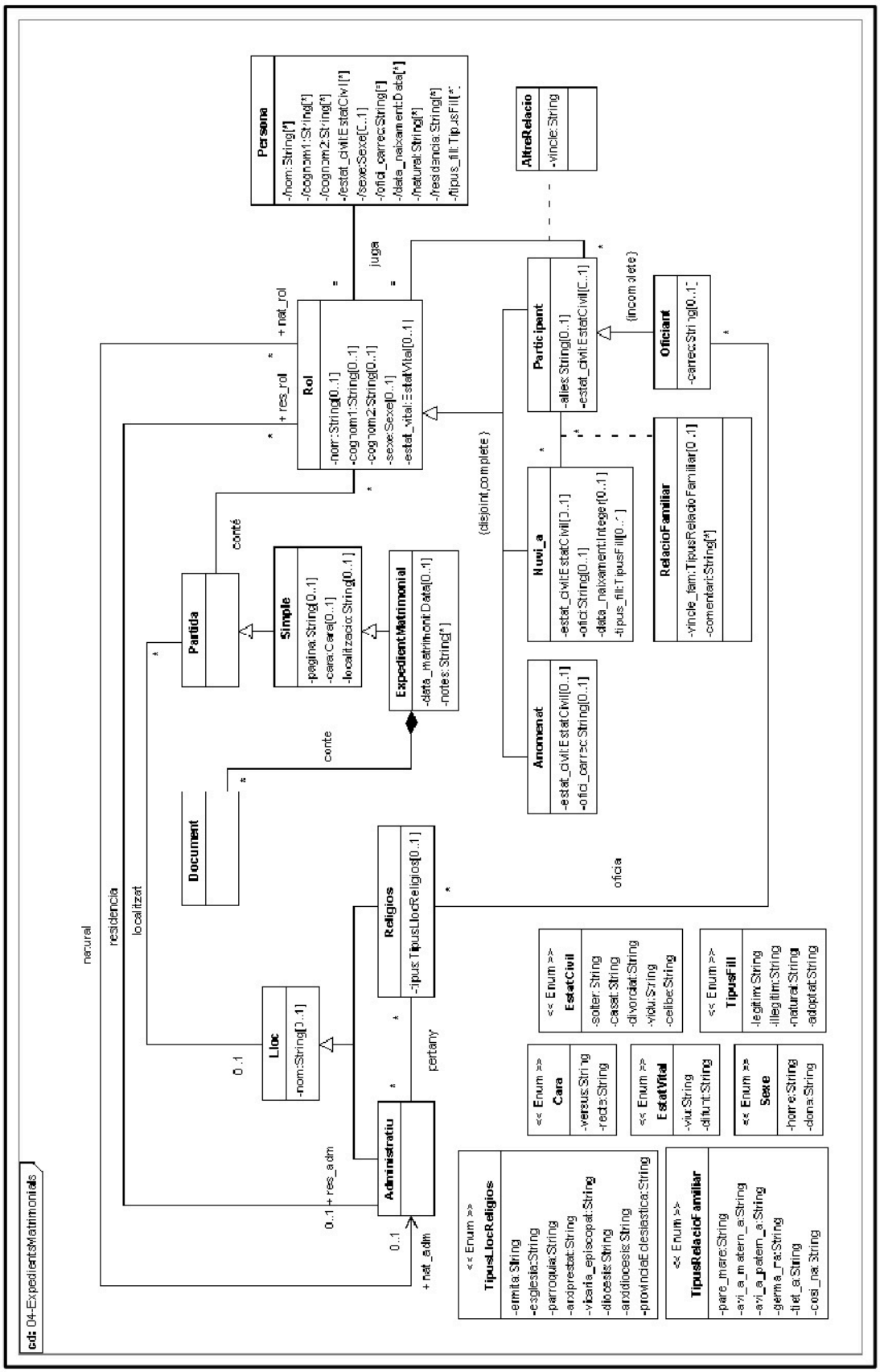
Modelo Conceptual de Obituario



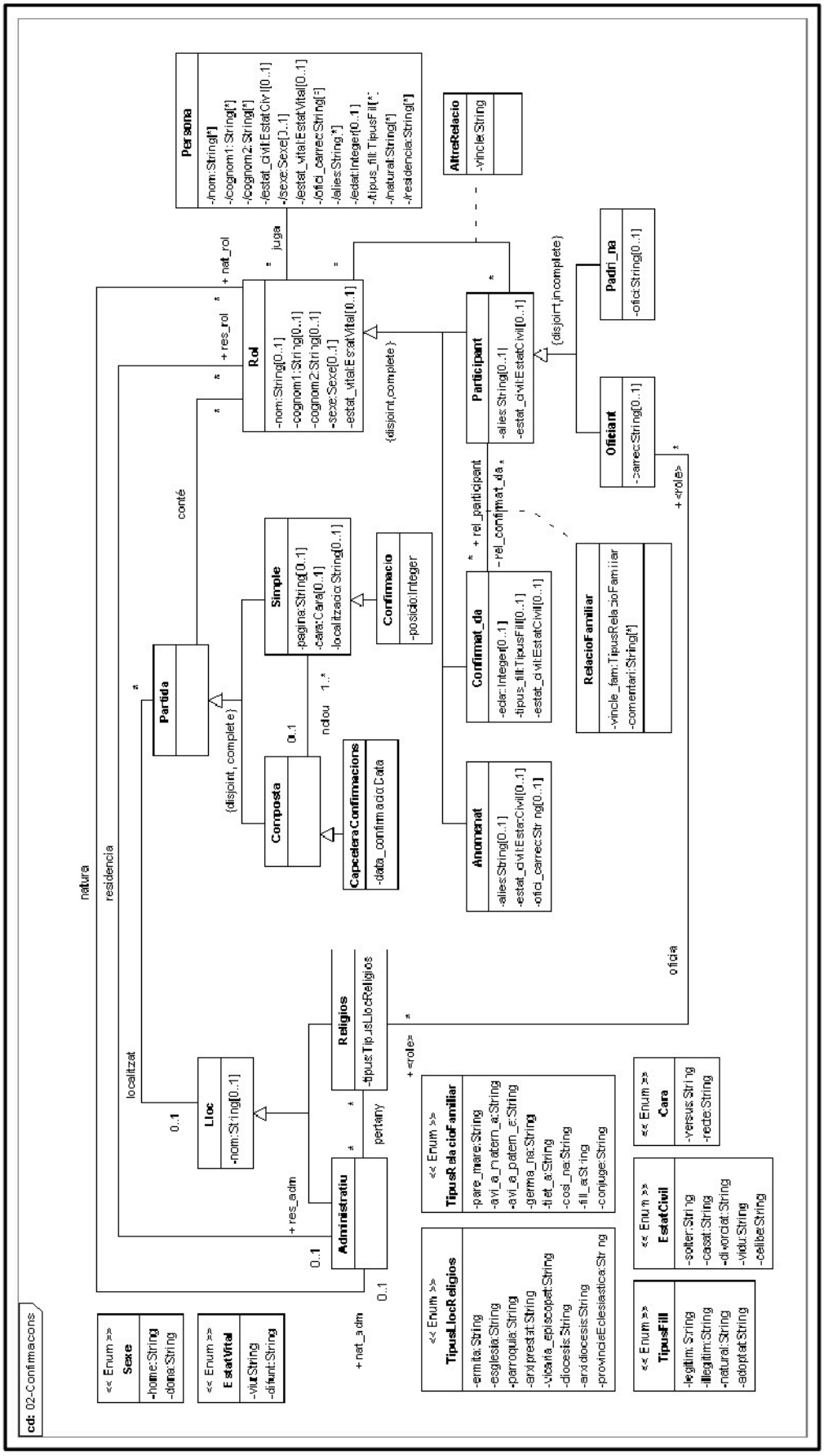
Modelo Conceptual de Partida Matrimonial



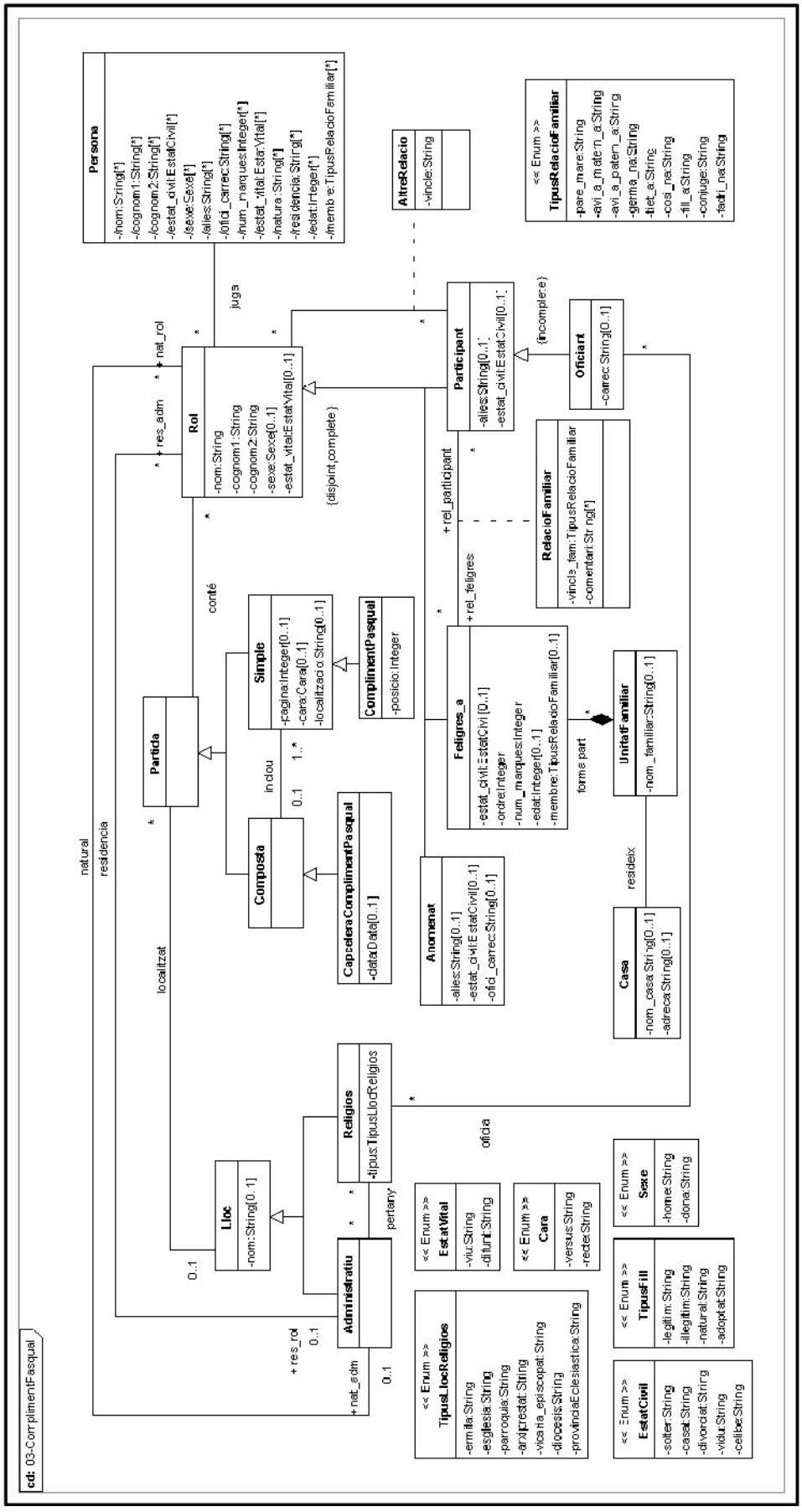
Modelo Conceptual de Capituloción Matrimonial



Modelo Conceptual de Expediente Matrimonial



Modelo Conceptual de Partida de Confirmación



Modelo Conceptual de Cumplimiento Pascual

Inserción de datos

A la hora de insertar datos de personas, para facilitar la tarea, se programó unos scripts para poder insertar de forma eficiente los datos de personas. La única tarea que se necesitaba hacer es la de insertar nuevos identificadores de personas, de documentos y de partidas en el gestor de base de datos (Sequel o MyphpAdmin). A continuación se mostrará el código utilizado para esta tarea.

```
     */
    public function newDato( $nombre ,$valor ,$id_persona ,$id_partida ,$tipo )
    {

        $this->connectDb( 'pfc' );
        $this->db->autocommit(false);
        $nombre = $nombre;
        $sql =<<<SQL
INSERT INTO Dato(nombre, dato_id, tipo_dato, partida_id, persona_id)
    VALUES ('$nombre', NULL ,'$tipo', $id_partida, $id_persona);
SQL;
        $this->db->query( $sql );
        switch ( $tipo )
        {
            case 'string':
                $sql = <<< SQL
INSERT INTO  String (dato_id , valor)
VALUES (last_insert_id(), '$valor' );
SQL;
                break;
            case 'integer':
                $sql = <<< SQL
INSERT INTO  Numero (dato_id , valor)
VALUES (last_insert_id(), $valor );
SQL;
                break;
            case 'date':
                $fecha = $valor['year'] . '-' . $valor['month'] . '-' . $valor['day'];
                $sql = <<< SQL
INSERT INTO  Date (dato_id , valor)
VALUES (last_insert_id(), '$fecha' );
SQL;
                break;
            case 'boolean':
                $sql = <<< SQL
INSERT INTO  Boolean (dato_id , valor)
VALUES (last_insert_id(), $valor );
SQL;
                break;
        }
        $res = $this->db->query( $sql );
        if ($res)
        {
            $this->db->commit();
        }
        return $res;
    }
```

Modelo para Insertar datos

```php
<?php
    class NewPersonaController extends Controller
    {
        public function run()
        {

            $nombres_dato = $this->getData('DatoModel','getNombresAtributo',array());
            $personas = $this->getData('PersonaModel','getPersonasId' ,array());
            $partidas = $this->getData('DocumentoModel','getPartidasId',array());
            $errors = array();
            $lista = array();
            //$post = FilterPost::getInstance();
            if ($_POST)
            {

                foreach ($nombres_dato as $f)
                {
                    $f['nombre'] = utf8_encode($f['nombre']);
                    $lista[$f['nombre']] = $f;
                }
                foreach ($_POST as $k => $v)
                {

                    if (!array_key_exists($k, $lista) || empty($_POST[$k]))
                    {
                        continue;
                    }
                    $tipo = $lista[$k]['tipo'];
                    switch ($tipo)
                    {
                    case 'string':
                        if(!is_string($v))
                        {
                            $errors[$k] = true;
                        }
                        break;
                    case 'integer':
                        if(!is_numeric($v))
                        {
                            $errors[$k] = true;

                        }
                        break;
                    case 'date':
                        // Falta refinar este
                        $date = array();
                        $date['day'] = '00';
                        $date['month'] = '00';
                        $date['year'] = '00';
                        $fecha = date_create_from_format('d-m-Y', $v);
                        if($fecha )
                        {
                            $date['day'] = $fecha->format('d');
                            $date['month'] = $fecha->format('m');
                            $date['year'] = $fecha->format('Y');
                        }

                        else if ($fecha= date_create_from_format('Y', $v))
                        {
                            $date['year'] = $fecha->format('Y');
                        }
                        else if ($fecha = date_create_from_format ('m-Y', $v))
                        {
                            $date['month'] = $fecha->format('m');
                            $date['year'] = $fecha->format('Y');
                        }
                        else
                        {
                            $error[$k] = true;
                        }
                        $v = $date;
                        break;
                    case 'boolean':
                        if ( $v )
                        {
                            $v = true;
                        }
                    }

                    if( empty($errors) && $_POST['partida_id'] && $_POST['persona_id']  )
                    {
                        $id_partida = $_POST['partida_id'];
                        $persona_id = $_POST['persona_id'];
                        $arguments = array($k ,$v , $persona_id ,$id_partida ,$tipo);
                        $insert = $this->getData('DatoModel','newDato' ,$arguments);
                        //newDato( $k ,$v , $id_persona ,$id_partida ,$tipo);
                    }
                }
            }
            $this->template->assign('personas',$personas);
            $this->template->assign('nombres_dato',$nombres_dato);
            $this->template->assign('errors',$errors);
            $this->template->assign('partidas',$partidas);
            $this->template->setTemplate('persona/new');
```

Controlador para insertar datos

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>Title of the document</title>
    <link rel="stylesheet" type="text/css" href="<?php echo CSS_PATH . '/global.css'; ?>" />
</head>
<body>
    <?php

$enum = array(
    'sexo' => array('hombre', 'mujer', 'desconocido'),
    'estado_civil' => array('soltero', 'casado', 'viudo', 'divorciado', 'desconocido'),
    'estado_vital' => array('muerto', 'vivo', 'desconocido'),
    'tipo_hijo' =>array('legitimo','ilegitimo','adoptado','natural', 'desconocido'),
    'tipo_doc' => array('dni','pasaporte', 'tarjetaResidencia','cedulaPersonal', 'desconocido'),
    'tipo_boda' => array('civil','canonico', 'desconocido'),
    'bool' => array('si', 'no', 'desconocido'),
    'relacion' => array('hijo' ,'hermanos', 'pareja', 'desconocido')
    );

?>

<form method="post" action="">
<label> Partida </label>
    <select name="partida_id">
        <option value="" selected="selected"> Selecione un id de Partida </option>
<?php foreach( $this->partidas  as $partida)
{
    echo '<option value="'.$partida['partida_id'].'">'.$partida['partida_id'].' </option>';
}
?>
    </select><br />
<label> idPersona</label>
<select name="persona_id">
    <option value="" selected="selected"> Selecione un id de Persona </option>
<?php
foreach ($this->personas as $id_persona)
        echo '<option value="'.$id_persona['persona_id'].'">'.$id_persona['persona_id'].' </option>';
?>
    </select><br />

    <?php foreach($this->nombres_dato as $f)
  {
      $f['nombre'] = utf8_encode($f['nombre']);
      if( key_exists($f['nombre'],$this->errors ) )
      {
      echo 'ERROR ';
      }
      echo '<label>' . $f['nombre'] . '</label>';
      if (($f['enum']))
      {
          echo '<select name="'.$f['nombre'].'">';
          foreach($enum[$f['enum']] as $t)
          {
              echo '<option value="';
              if('desconocido' === $t)
                  echo '" selected="selected" ';
              else
              {
                  echo $t.'"';
              }

              echo '>'.$t.'</option>';
          }
          echo "</select><br />";
      }
      else
      {
          if($_POST)
          {
              $value = $_POST[$f['nombre']];
          }
          else
          {
              $value = '';
          }
          echo '<input type="text" name="' . $f['nombre'] . '" value="'.$value.'" /><br />';
      }
  }
  ?>
  <input type="submit" name ="Enviar"/>
  </form>
  </body>
  </html>
```

Formulario para insertar datos