



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE DEPARTAMENTO DE CIENCIAS DE
LA COMPUTACIÓN

DISEÑO Y DESARROLLO DE UNA HERRAMIENTA DE REPRESENTACIÓN Y
VISUALIZACIÓN DE REDES SOCIALES CON CAPACIDADES DISTRIBUIDAS

MEMORIA PARA OPTAR AL TÍTULO DE TÍTULO DE INGENIERO CIVIL EN
COMPUTACIÓN

FELIPE ANIBAL RICARDO ESPINOZA CASTILLO

PROFESOR GUÍA:
CLAUDIO GUTIÉRREZ GALLARDO

MIEMBROS DE LA COMISIÓN:
GONZALO NAVARRO BADINO
LUIS MATEU BRULE

SANTIAGO DE CHILE
JULIO 2013

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE:
INGENIERO CIVIL EN COMPUTACIÓN
POR: FELIPE ANÍBAL RICARDO ESPINOZA CASTILLO
PROFESOR GUÍA: CLAUDIO GUTIÉRREZ GALLARDO
FECHA: 12/07/2013

DISEÑO Y DESARROLLO DE UNA HERRAMIENTA DE REPRESENTACIÓN Y VISUALIZACIÓN DE REDES SOCIALES CON CAPACIDADES DISTRIBUIDAS

Hoy existen diversas bases de datos centralizadas de relaciones de *quién conoce a quién*, como LittleSis y Poderopedia, que contienen información de posibles conflictos de intereses entre las personas poderosas de un país. Ellas son de mucha utilidad como medios de entrega de información a las personas. Ellas presentan un modelo de red social que puede ser replicado en otras áreas del conocimiento.

Los investigadores y trabajadores sociales que se enfrentan con este tipo de estructuras sociales requieren aplicaciones que les faciliten su creación y manejo. Con el objetivo de apoyar el trabajo de estas personas, que no necesariamente tienen conocimientos avanzados de computación, en este trabajo de memoria se abordó la creación de una herramienta que le permitiera a sus usuarios exponer información de las estructuras sociales en forma de redes, manejarlas (crearlas, modificarlas, borrarlas, etc.) en los contextos que los ellos estimen conveniente.

Este trabajo partió de la experiencia de una memoria similar realizada anteriormente por el ex alumno del departamento Manuel Bahamonde. En este trabajo se mejoró su propuesta inicial, aplicando su experiencia y usando el modelo mejorado propuesto por Mauro San Martín en su tesis de doctorado.

A partir de lo anterior, se creó una aplicación web que le permite al usuario realizar las tareas señaladas, con una interfaz amigable y usando las últimas tecnologías en lo que a desarrollo web se refiere. Esta aplicación además permite la interacción entre los usuarios de la aplicación ya que les provee herramientas para combinar la información que producen entre ellos.

En ese sentido, se logró el objetivo propuesto ya que la aplicación aprovecha la flexibilidad del modelo de Mauro San Martín para modelar redes sociales de escala pequeña con una estructura que permite una adición flexible de información. Adicionalmente, se logró un primer acercamiento a la integración con tecnologías de la web semántica, debido a que la información producida puede ser importada y exportada en formato RDF.

Finalmente se discutieron los desafíos encontrados durante el trabajo, además de indicar los principales focos de mejora y avance de este proyecto.

A María Cristina, Andrés, Jorge y Simón.

Agradecimientos

Índice general

Introducción	1
1. Descripción del Proyecto	4
1.1. Objetivos	4
1.1.1. Objetivo General	4
1.1.2. Objetivos Específicos	4
1.2. Resultados Esperados	5
1.3. Alcances	5
2. Marco Conceptual	6
2.1. Conceptos de Redes Sociales	6
2.1.1. Actor	6
2.1.2. Vínculos Relacionales	7
2.1.3. Relaciones	7
2.1.4. Red Social	7
2.2. Conceptos de Desarrollo	8
2.2.1. Desarrollo Web	8
2.2.1.1. Desarrollo Front-End	9
2.2.1.2. Desarrollo Back-End	10
2.2.2. Modelo MVC	10
2.2.3. Frameworks de Desarrollo	11
2.2.3.1. Frameworks Server Side	11
2.2.3.2. Frameworks Client Side	11
2.2.4. HTML5	11
2.2.4.1. SVG, gráficos vectoriales en la web	12
2.3. Herramientas Elegidas	12
2.3.1. Ruby on Rails	12
2.3.2. EmberJS	13
2.3.3. D3.js	13
3. Especificación del Problema	14
3.1. Antecedentes y soluciones existentes	14
3.1.1. LittleSis y Poderopedia	14
3.1.2. Memoria Manuel Bahamonde	17
3.1.3. Otras Alternativas	19
3.2. Relevancia del Problema	19

3.3.	Usuarios Objetivo	20
3.4.	Requisitos de la Aplicación	20
3.4.1.	Requisitos Funcionales	20
3.4.2.	Requisitos de Calidad	20
4.	Descripción de la Solución	21
4.1.	SNM, Social Network Model	21
4.1.1.	Elementos de Redes Sociales	21
4.1.2.	Definición Matemática	22
4.1.3.	Representación Gráfica	23
4.1.4.	Representación Como Triples	24
4.2.	Arquitectura de la Aplicación	25
4.2.1.	Arquitectura de Software	25
4.2.2.	Entidades	28
4.2.3.	Diseño de la Base de Datos	29
5.	Funcionamiento de la Solución	31
5.1.	Registro e Ingreso de Usuarios	32
5.2.	Creación de Redes Sociales	32
5.3.	Edición de Redes Sociales	34
5.3.1.	Área de Edición de Redes Sociales	34
5.3.1.1.	Modos de Edición	35
5.3.2.	Creación y Edición de Familias	36
5.3.3.	Creación y Edición de Actores	37
5.3.4.	Creación y Edición de Relaciones	39
5.3.5.	Creación y Edición de Atributos en Nodos	40
5.3.6.	Creación y Edición de Roles	41
5.4.	Exportación en RDF	42
5.5.	Importación en RDF	45
5.6.	Unión de Redes Sociales	45
5.7.	Ejemplos de Uso	48
6.	Conclusión	49
6.1.	Evaluación de la Solución	49
6.2.	Dificultades Técnicas	50
6.2.1.	Modelamiento de Redes Sociales	50
6.2.2.	Interfaz de Alta Interacción en Desarrollo Web	50
6.3.	Trabajo Futuro	51
6.4.	Distribución del Software y Documentación	52

Índice de tablas

4.1. Representación en triples de la red social de la figura 4.2	25
--	----

Índice de figuras

2.1. Componentes Presentes en el Desarrollo Web	9
2.2. Modelo MVC	10
3.1. Interfaz de LittleSis, Rupert Murdoch	15
3.2. Interfaz de Poderopedia, Michelle Bachelet	16
3.3. Interfaz de Grafo de Poderopedia, Michelle Bachelet	16
3.4. Interfaz SocialNetworks Manager	18
4.1. Elementos Gráficos SNM	24
4.2. Ejemplo Red Social de Amistad en SNM	24
4.3. Arquitectura General Aplicación-Usuarios	26
4.4. Arquitectura de la Aplicación	27
4.5. Esquema Base de Datos de La Aplicación	29
5.1. Interfaz de Inicio de La Aplicación	31
5.2. Formulario de Creación de Usuarios	32
5.3. Formulario de Ingreso	32
5.4. Pantalla Principal de Redes Sociales	33
5.5. Formulario Nueva Red Social	33
5.6. Lista de Redes Sociales	34
5.7. Área de Edición de Redes	35
5.8. Modos de Edición	35
5.9. Área de Familias	36
5.10. Formulario de Edición de Familias	37
5.11. Detalle Familia Creada	37
5.12. Formulario de Creación de Actor	38
5.13. Ejemplo Actor con 2 Familias	39
5.14. Diálogo de Eliminación de Actor	39
5.15. Formulario de Creación de Relación	40
5.16. Añadiendo Atributos a un Actor	41
5.17. Formulario de Edición de Rol	42
5.18. Botón de Exportación de la Red Social	42
5.19. Micro Ejemplo de Red Social	43
5.20. Importación de una Red Social	45
5.21. Selección de Redes a Unir	46
5.22. Selección de Equivalencias entre Familias	46
5.23. Operación de Unión de Nodos	48

Introducción

Existe una red social llamada **LittleSis**[12] que consiste en una base de datos de relaciones de *quién conoce a quién* entre gente política, económica y socialmente poderosa en el mundo de las organizaciones en Estados Unidos cuyo fin es entregar el poder de la información a la sociedad civil.

El origen del nombre de *LittleSis* se relaciona con el personaje *Big Brother* de la novela de George Orwell llamada *Nineteen Eighty-Four*. Este personaje consiste en un dictador que maneja toda la información sobre la población. *Big Brother* posteriormente fue un concepto con el cual se describió un ente poderoso.

De esta forma, al grupo compuesto por las personas poderosas y políticos de un país se les atribuye esta imagen de *Big Brother*, por su poder e influencia en un país. Esta es la motivación para bautizar con el nombre *LittleSis* (*Little Sister*), en oposición a *Big Brother*, a una aplicación cuyo fin es entregar poder de la información a toda la población en lugar de esta minoría poderosa.

La idea detrás de *LittleSis* tiene la capacidad de ser replicada en muchos países con el mismo fin, esto es, el de entregar poder a la ciudadanía e informarle sobre los posibles conflictos de interés que las autoridades locales y/o nacionales puedan tener al lidiar con el ejercicio de su poder. En el caso particular de Chile, existe una iniciativa llamada **Poderopedia**[16] que busca ser una réplica de *LittleSis* para Chile.

En el caso de **Poderopedia**, uno de los líderes de ese proyecto es el ex alumno del DCC, Álvaro Graves, quienes tienen un **modelo RDF publicado en un repositorio en Github**[24]. Poderopedia, al igual que LittleSis, es un repositorio centralizado de información sobre la gente de poder político y económico de Chile, sus relaciones entre sí y su relación con organizaciones.

Con toda sus potencialidades, tanto *LittleSis* como *Poderopedia*, poseen un fin muy particular. En este sentido, sólo tienen la capacidad de cubrir la información sobre las personas muy importantes del mundo político, social y económico poderoso. Pero si una persona quisiera recolectar información sobre grupos de personas (por ejemplo la red social de la comunidad investigadora en computación dentro de latinoamérica) para formar una red social, estas

plataformas son de poca ayuda. Tampoco estas herramientas permiten la creación de redes sociales privadas, que cumplan los objetivos que el usuario desee, pero que al mismo tiempo le permiten interactuar con los datos de otras redes sociales pertenecientes a otras personas o comunidades.

El objetivo de esta memoria es llenar ese vacío, diseñando y desarrollando una herramienta para representar redes sociales, y que tenga capacidades distribuidas, donde los usuarios que poseen sus redes sociales públicas o privadas, puedan unir las y compartirlas a voluntad.

Esta herramienta posee una gran gama de aplicaciones posibles, tanto para estudios de tipo sociológico, histórico, biológico, político, etc. Un caso de ejemplo reciente es el caso de las redes de autoridades e intereses posibles dentro del ambiente educacional chileno, lo que puede dar una visión más informada sobre el entramado de intereses detrás de los problemas denunciados por el conflicto estudiantil que se vive desde el año 2011.

Para lograr desarrollar esta herramienta se deben sortear una serie de desafíos técnicos. Aunque pareciera una aplicación bastante intuitiva, hasta el momento no hemos encontrado ninguna aplicación con estas características.

Entre sus mayores desafíos está el de tener una aplicación de fácil instalación y uso amigable para los usuarios. Por otro lado, con lo que respecta al modelamiento de datos, debe basarse en algún estándar flexible de representación de redes sociales que permita la interoperabilidad. Esta memoria partirá de la base y la experiencia del recientemente titulado doctor en computación Mauro San Martín, cuya tesis de doctorado consistió en diseñar un modelo para el manejo de redes sociales, de esta forma, se aprovechará el conocimiento de su experiencia de investigación, aplicándolo a un trabajo práctico y útil para la investigación y estudio de varias disciplinas.

También habrá que desarrollar el aspecto de la visualización de datos, la integración con otras herramientas enfocadas al análisis de redes sociales, el modelamiento y uso de la información de redes sociales.

Dado que la estructura de redes sociales está fuertemente ligada a los grafos, y se necesita un estándar aceptado (para el que existan herramientas de desarrollo disponibles), se ha optado por representarlo en el modelo **RDF**. Este estándar es además una representación que permitirá todo tipo de usos incorporándose a tecnologías de la **Web Semántica**. Finalmente, además de los anteriores, hay desafíos de negocios que se vinculan con lo técnico, como hacer algunos casos de prueba de usabilidad para entregar una herramienta de valor a usuarios y contactar diversas personas que puedan estar interesadas en la herramienta.

Este informe comprende la experiencia obtenida al realizar esta memoria, las motivacio-

nes, las soluciones propuestas y los resultados obtenidos durante el periodo de la misma. En el capítulo 1 se presentan los objetivos y alcances para este trabajo. Este informe está organizado de la siguiente manera: en el capítulo 2 se abordan los conceptos relacionados a redes sociales y de desarrollo que son útiles para entender el resto del informe. En el capítulo 3 se aborda en mayor medida el contexto en el cual se desarrolla el problema y se extraen los puntos principales a resolver. Luego en el capítulo 4 se documenta la arquitectura conceptual y concreta de la aplicación desarrollada para solucionar lo planteado en todo sus detalles, en el capítulo 5 se expone como esta solución funciona y finalmente se discuten las conclusiones del trabajo y sus alternativas a futuro.

Capítulo 1

Descripción del Proyecto

1.1. Objetivos

A continuación se presentan los objetivos para este trabajo de memoria.

1.1.1. Objetivo General

El objetivo de este trabajo consiste en crear una herramienta por la cual personas que estudian diversos tipos de redes sociales como: sociólogos, periodistas, biólogos, etc; puedan representar, administrar y visualizar redes sociales de mediana escala, contando además con la capacidad de combinar redes sociales con otros usuarios de la herramienta para obtener redes sociales con información más completa ofreciendo esto como un servicio centralizado con la capacidad de ser distribuido.

1.1.2. Objetivos Específicos

De lo escrito anteriormente en el objetivo general, se desprenden los siguientes objetivos intermedios:

1. Implementar un modelo de representación de redes sociales en RDF.
2. Hacer una interfaz amigable de ingreso de datos: actores y relaciones.
3. Desarrollar un sistema centralizado que administre la asignación de identificadores únicos a los actores que los usuarios agregan a sus redes sociales.
4. Unir redes sociales, combinando la información y relaciones de actores en común.
5. Visualización
 - (a) De la estructura general de las redes sociales creadas.
 - (b) Específica de elementos de interés en esas redes sociales.

6. Representar algunas redes con el modelo implementado anteriormente, que varíen en tamaño y en complejidad.

1.2. Resultados Esperados

Para llevar a cabo los objetivos expuestos en esta memoria, los resultados que se esperan consisten en: un sistema que permita el modelamiento de redes sociales en donde debe existir una aplicación que permita de manera cómoda crear estas redes sociales, complementar su información para posteriormente unir las redes de un usuario con otro en caso de que este lo quiera. Además de lo anterior, los datos generados por el sistema deben estar en un formato amigable para computadores, que faciliten la posterior interoperabilidad con otras aplicaciones y fuentes de conocimiento.

1.3. Alcances

Es importante mencionar, que con el objetivo que el trabajo a realizar cumpla con las limitaciones de tiempo correspondientes a una memoria de ingeniería, el problema de la creación y manipulación de redes sociales por personas se acota en los siguientes aspectos:

- El tamaño de redes sociales, se considerará pequeño con un número de alrededor 100 actores por red social, debido a que las redes serán principalmente creadas de forma manual, por tanto debe manejar un número que sea posible de alcanzar por los usuarios de la aplicación, de esta forma, ahorrando problemáticas asociadas con la escalabilidad de la aplicación para redes sociales más grandes.
- Dependiendo de las aplicaciones para las cuales se use el modelamiento de redes sociales, puede requerirse ver los cambios temporales que sufren estas, para analizar la estructura de una red social y su evolución en el tiempo. Dicho problema no será abordado en esta memoria.

Capítulo 2

Marco Conceptual

En este capítulo se describen los conceptos necesarios para que el lector se pueda familiarizar con los conceptos usados en el resto del informe, describiendo lo que se entiende por los conceptos usados en términos de redes sociales y conceptos asociados al desarrollo de la solución.

En caso de así preferirlo, el lector puede saltar las secciones que estime convenientes dentro de este capítulo, en caso de dominar los conceptos y usar este capítulo como referencia en caso de necesitarlo.

2.1. Conceptos de Redes Sociales

A continuación se expondrán los conceptos fundamentales que se usarán en la memoria en relación a redes sociales. Estos conceptos fueron tomados de la tesis de doctorado del alumno del DCC, Mauro San Martín[29], que representan lo necesario para el entendimiento de este trabajo. Su reuso e inclusión en el informe se hace sólo para que el lector no necesite revisar la memoria de Mauro para poder ver estos conceptos.

Para construir la definición formal de redes sociales, es necesario definir algunos conceptos claves previamente, siguiendo las definiciones clásicas de este dominio[30].

2.1.1. Actor

Un actor es una entidad social, la cual está bajo estudio junto con sus interacciones sociales. En estricto rigor, los actores pueden ser definidos como individuos, corporaciones o unidades sociales colectivas. Ejemplos de actores son gente en un grupo, departamentos en una empresa o agencias de servicio público en una ciudad. El uso del término actor no significa que estas entidades necesariamente tienen la habilidad de actuar. Más allá, la mayoría de las

aplicaciones en redes sociales se enfocan en colecciones de actores que son de un mismo tipo (por ejemplo, gente en un grupo de trabajo). A veces, sin embargo, la investigación necesita mirar a actores de diversos niveles o tipos conceptuales, o desde diversos conjuntos. Los datos pueden incluir atributos no relacionales asociados a los diferentes actores.

2.1.2. Vínculos Relacionales

Los actores están conectados hacia otros por vínculos sociales. El rango y tipo de estos puede ser muy amplio. La característica principal de un vínculo es que establece una conexión entre un par de actores. Algunos de los ejemplos más comunes de vínculos empleados en el análisis de redes sociales son:

- La evaluación de una persona por otra, ej: amistad declarada, gusto o respeto.
- Transferencia de recursos materiales, ej: transacciones de negocios, prestar o pedir prestado.
- Asociación o afiliación, ej: atender conjuntamente a un evento social, o pertenecer al mismo club social.

2.1.3. Relaciones

El conjunto de vínculos entre un tipo específico de miembros de un grupo se llama una relación. Por ejemplo, el conjunto de las amistades entre pares de niños en un salón de clases, o el conjunto de uniones diplomáticas entre pares de naciones en el mundo, son vínculos que definen relaciones. Para cualquier grupo de actores, podemos encontrar diversas relaciones; por ejemplo, además de las relaciones diplomáticas entre países, podemos encontrar la existencia de comercio en un determinado año. Las relaciones (o vínculos específicas) pueden tener atributos que las describen. Por ejemplo en el caso del comercio, su cantidad de transacciones total puede haber sido almacenado.

Con lo expuesto anteriormente, finalmente podemos definir una red social.

2.1.4. Red Social

Una red social consiste en uno o muchos conjuntos finitos de actores, junto con las relaciones definidas entre ellos. La presencia de información relacional es una característica crítica de las redes sociales. Una red social es un caso particular de red, de esta manera su estructura puede ser formalizada como un grafo.

En adición al uso de conceptos relacionales (Wasserman y Faust[30]) se deben tener en cuenta las siguientes consideraciones:

- Actores y sus acciones son vistos como interdependientes, más que independientes, unidades autónomas.

- Vínculos relacionales (conexiones) entre actores son canales de transferencia o "flujo" de recursos (material o no material).
- Modelos de red enfocados en individuos muestran el ambiente estructural de la red, además de proveer la capacidad de definir limitaciones en nivel individual.
- Los modelos de red conceptualizan estructura (social, económica, política, entre otras) como patrones generales de relaciones entre actores.

2.2. Conceptos de Desarrollo

A continuación se explican los términos referentes al desarrollo de software del trabajo realizado en esta memoria.

2.2.1. Desarrollo Web

El desarrollo web se denomina al desarrollo de software cuyo fin es la creación de aplicaciones que se ejecuten en computadores que puedan ser accedidos vía la *world-wide web*. Este estilo de desarrollo conlleva a que una aplicación que pueda estar ejecutándose en un servidor pueda entregar resultados a clientes provenientes de cualquier parte del mundo, usando una gran diversidad de dispositivos, plataformas de software, etc. Esas aplicaciones son frecuentemente accesadas por personas vía un navegador web.

Esta propiedad de multiplataforma de las aplicaciones desarrolladas para la web, junto con propiedades de accesibilidad desde diversos medios a las aplicaciones, ha hecho que el desarrollo web sea un enfoque de desarrollo ampliamente usado en la industria en los últimos 10 años.

En términos generales el desarrollo web posee una arquitectura clásica de cliente/servidor, en donde una aplicación corriendo en un servidor proporciona resultados y ejecuta instrucciones proveniente de clientes, que pueden ir desde un navegador web de p.c. o dispositivo móvil, o servir de API, *Application Programming Interface*, para otras aplicaciones. A continuación se adjunta un diagrama que muestra los principales componentes cuando se habla de desarrollo web.

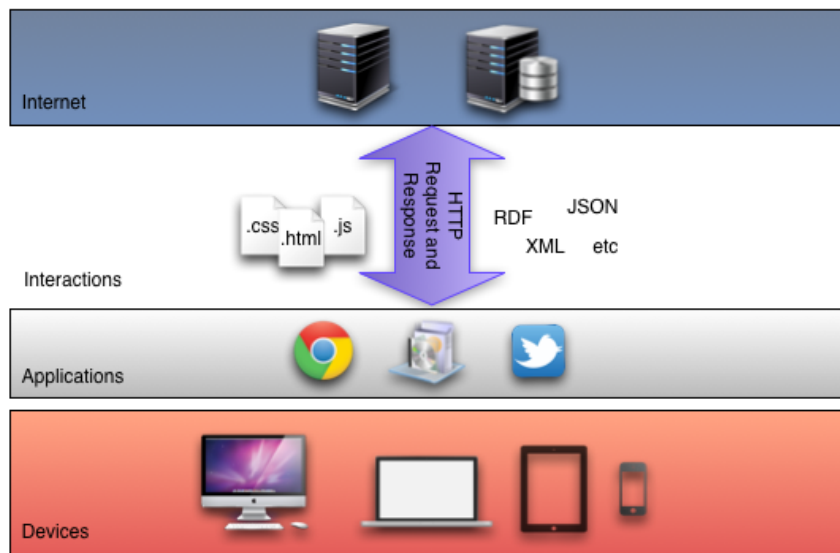


Figura 2.1: *Componentes Presentes en el Desarrollo Web*. Descritos de manera general se tienen las siguientes capas: internet (con servidores de aplicación y servidores de base de datos), transmisión (via http en formato de documentos u otro tipo de mensajes), aplicaciones (web browser, aplicaciones desktop, aplicaciones mobile) y dispositivos (desktop pcs, notebooks, tablets y teléfonos).

A modo de ejemplo de componentes en el desarrollo web de una aplicación que posee un servidor de ejecución de la aplicación, con un servidor de base de datos, que es accesada vía teléfono, tablet o computador por medio de un navegador web o de una interfaz API que use otra aplicación, por la cual se acceden sus datos en formato de documentos HTML (junto con sus estilos en CSS y manejo de eventos en JavaScript) o documentos en formato JSON, RDF, XML entre otros.

Dentro del desarrollo web, también se encuentran subcategorías que son expuestas a continuación.

2.2.1.1. Desarrollo Front-End

Es el desarrollo orientado a todos los componentes con los cuales interactúa un usuario de la aplicación, enfocado en la interfaz y sus componentes gráficos, además de la experiencia usuario, la usabilidad de la aplicación, etc. Es el desarrollo del cliente que el usuario usa para acceder al core de la aplicación.

En términos de tecnologías, se asocia el desarrollo front-end al uso de tecnologías como: HTML, CSS, Javascript, entre otras.

2.2.1.2. Desarrollo Back-End

Es el desarrollo de la aplicación que almacena y posee la lógica común a todos los clientes de la aplicación, en donde el código de este tipo de desarrollo se ejecuta en el servidor. Además generalmente en el desarrollo back-end se incluye todo lo relacionado a persistencia de datos generados a través del uso de la aplicación.

2.2.2. Modelo MVC

El modelo MVC (Modelo, Vista, Controlador)[18], consiste en un modelo para separar la lógica de una aplicación agrupándola en clases u otras unidades modulares, de acuerdo con la responsabilidad que estos módulos cumplan dentro de un sistema. A modo de ejemplo, a fin de ilustrar este concepto, se puede dar un caso de una aplicación que registre compras en un sistema de tienda online, un ejemplo de los módulos asociados a compras en MVC puede ser el siguiente:

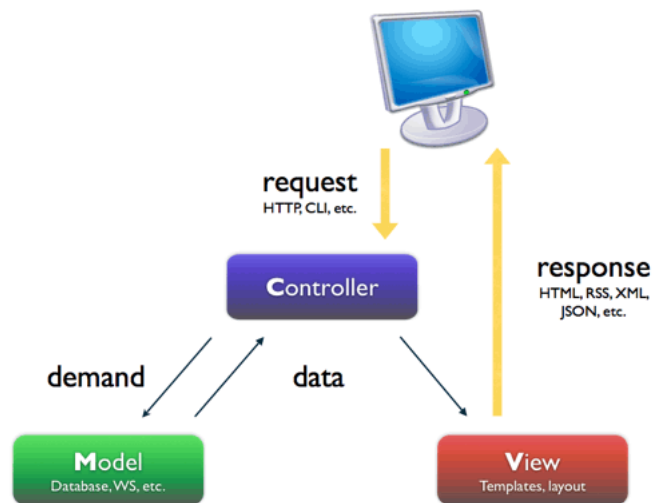


Figura 2.2: Modelo MVC

- **Modelo:** el modelo corresponde a una clase **Compra** que contiene toda la lógica de negocio asociada a las compras, que además se asocia directamente a cómo se almacena una compra en la base de datos.
- **Vista:** un ejemplo de vista para una compra, puede ser una interfaz **HTML** en la cual el cliente efectúe operaciones sobre la compra, por ejemplo agregar productos, cabe destacar que la vista no ejecuta las acciones, sólo se encarga de recibirlas y enviarlas al último componente del modelo MVC, el controlador.
- **Controlador:** de acuerdo a lo recién expresado, el controlador es el encargado de coordinar uno o más modelos para ejecutar las acciones capturadas en la vista. En el caso de la compra, el controlador es quien efectivamente procesaría el pago de la misma.

2.2.3. Frameworks de Desarrollo

Un framework de desarrollo define un marco en el cual desarrollar una aplicación. Provee una gran cantidad de funcionalidad común lista de manera de no desarrollar un proyecto desde cero, pero además de funcionalidad, provee de una estructura lógica con la cual se escribe el código, que está influida por muchas otras personas que han usado el framework en aplicaciones reales.

2.2.3.1. Frameworks Server Side

Dentro de los frameworks de desarrollo web existe una categoría llamada *Sever Side*, la cual consiste en que el código de la aplicación corre desde un servidor en internet, lo que permite que si la computación es común para muchos clientes, los resultados de esa computación pueden ser usados múltiples veces.

Ejemplos actuales de esta categoría de frameworks son: **Ruby on Rails**[19], **Sinatra**[20] que utilizan el lenguaje Ruby; **Django**[8], **Pylons**[17] para Python; **Spring**[21] para Java y **CakePHP**[4] para PHP entre otros.

2.2.3.2. Frameworks Client Side

En el último tiempo, surgió una nueva categoría de frameworks de desarrollo web llamada *Client Side*, en la cual el código de la aplicación se ejecuta en el computador del usuario de la aplicación, ahorrando recursos necesarios en un servidor, además de ahorrar el tiempo de latencia entre el cómputo de una respuesta y su transmisión al equipo del usuario.

Comúnmente estos frameworks son escritos para ser usados con el lenguaje **Javascript**, pues posee la propiedad que todos los navegadores web implementan un motor de **Javascript** y por lo tanto el usuario no necesita instalar nada más. Ejemplos de frameworks client side son: **AngularJS**[2], **EmberJS**[28], **Meteor**[13] y alternativamente **BackboneJS**[3] que es una librería más que un framework.

2.2.4. HTML5

Html5[6] es el nuevo estándar para *HTML* (Hyper-Text Markup Language), definido por la WC3[25] y Web Hypertext Application Technology Group (WHATWG), es un trabajo en progreso, sin embargo desde hace tiempo los principales navegadores soportan muchas de los nuevos elementos de HTML y sus APIs.

Algunas de las nuevas características más interesantes en HTML5 son: el tag `<canvas>`

para dibujo en 2D, los tags `<video>` y `<audio>` para reproducción multimedia, soporte para almacenamiento local en el browser, algunos elementos específicos para el contenido como `<article>`, `<footer>`, `<header>`, `<nav>` y `<section>`; nuevos controles para formularios como para fecha, hora, email, url y búsqueda. Además de esto, soporte para SVG dentro de los sitios, característica especialmente importante para esta memoria.

2.2.4.1. SVG, gráficos vectoriales en la web

SVG, en inglés se refiere a gráficos vectoriales escalables, los cuales pueden ser usados directamente en documentos HTML5, de manera de crear gráficos complejos en un formato de tipo XML. La diferencia de este tipo de gráficos con respecto a imágenes por ejemplo, es que los gráficos SVG no pierden calidad si son redimensionados o se ven sus detalles por medio de una funcionalidad de lupa. Estos elementos son altamente animables y corresponde a una recomendación por parte de la W3C[25].

2.3. Herramientas Elegidas

A continuación se presentan las herramientas elegidas para el desarrollo de la aplicación y los fundamentos de estas elecciones:

2.3.1. Ruby on Rails

Para el desarrollo del backend de la aplicación se eligió el framework de desarrollo web Ruby on Rails[19], comúnmente conocido simplemente como Rails, en el cual se desarrolla sobre el lenguaje Ruby. Es un framework maduro con 10 años de existencia, el cual es una de las plataformas más populares para este tipo de desarrollo actualmente en Silicon Valley.

Dentro de las características destacables de rails se pueden destacar: sus principios de privilegiar las convenciones sobre las configuraciones, es decir, el framework entrega mucha funcionalidad hecha mientras se cumplan sus convenciones, las cuales de ser necesario se pueden sobre escribir; aplicaciones que pueden ser ejecutadas en diversos ambientes independientemente configurados como producción, testing o desarrollo.

Además de aspectos técnicos, dentro de las motivaciones al elegir rails, se encuentra la presencia de una gran comunidad de desarrolladores que crean muchas librerías, o en terminología ruby, 'gemas' las cuales hacen el desarrollo mucho más rápido reusando estas librerías que resuelven múltiples problemas frecuentes. Junto con lo anterior el alumno memorista posee vasta experiencia laboral en esta tecnología.

2.3.2. EmberJS

EmberJS es un framework de desarrollo web javascript en el lado del cliente, fue creado por Yehuda Katz y Tom Dale cuando ellos hacían la segunda versión del framework web Sproutcore.

EmberJS se destaca debido a que se pueden escribir aplicaciones completas que son ejecutadas en el lado del cliente, con clases especializadas como *modelos*, *vistas*, *controladores*, *templates* en una versión un poco distinta del modelo MVC. Una característica principal de emberjs es la actualización automática de templates cuando la información el sistema cambia; la presencia de observadores y la capacidad de adaptar emberjs a diversos modelos de persistencia como vía API JSON o el almacenamiento local integrado en los navegadores web de última generación.

Para el desarrollo de la aplicación de esta memoria, debido a que se trata de una aplicación que permite editar grafos interactivamente y es vía web, es necesaria mucha integración entre el javascript de las vistas y los datos de los nodos de las redes sociales, razón por la cual fue un mejor enfoque desarrollar la aplicación con emberJS, que resultó en pruebas iniciales mucho mejor que la otra alternativa evaluada, BackboneJS[3].

Uno de los aspectos útiles de EmberJS es que uno de sus creadores, Yehuda Katz, es integrante de los equipos principales de desarrollo de Ruby on Rails y la librería de javascript jQuery. De esta forma, EmberJS está pensado para tener una muy buena integración con Rails y jQuery, potenciando su atractivo como herramienta de desarrollo en el lado del cliente.

2.3.3. D3.js

Dentro de las herramientas principales se incluye D3.js[7], que es una librería javascript para el despliegue y manipulación de datos como información visual vía SVG en la web. Además de lo anterior D3.js es una de las opciones más utilizadas en lo que a gráficos SVG en la web se refiere.

El funcionamiento básico de D3 es el siguiente: se parte por asociar un arreglo de elementos que contienen los datos con los que trabajamos y son asociados con elementos SVG, como círculos por ejemplo y para el caso del arreglo de datos, D3 tiene 3 estados: el de entrada, es decir, cuando un elemento nuevo es agregado al arreglo, D3 se encarga de crear un nuevo elemento SVG para asociarlo a este nuevo dato; el estado de actualización es el cual que a todos los datos disponibles actualiza su posición u otras propiedades y finalmente el estado de salida es cuando un elemento del arreglo es eliminado y por consiguiente D3.js se encarga de remover su elemento SVG respectivo.

Capítulo 3

Especificación del Problema

3.1. Antecedentes y soluciones existentes

A continuación se presentan algunas de las soluciones existentes que apuntan a resolver problemas similares al de esta memoria.

3.1.1. LittleSis y Poderopedia

LittleSis, pequeña hermana en oposición al gran hermano (Big Brother), es una plataforma estadounidense que contiene una base de datos libre de relaciones de *quien conoce a quien* en el mundo de las grandes empresas y políticas de ese país. Todo esto con el fin de entregar información a la gente sobre qué posibles conflictos de intereses pueden tener los políticos que toman las decisiones en el país.

LittleSis
[Login](#)
[Sign Up](#)
[Explore](#)
[Help](#)
[About](#)
[Blog](#)

Rupert Murdoch

Chairman and CEO of News Corp

[add connection](#)
[edit](#)
[flag](#)
[Tweet](#)
[Recommend](#)

Edited by **Bot** 13 days ago

Australia-born tabloid titan welcomed Dow Jones and its crown jewel, the Wall Street Journal, into News Corp. empire in December after \$5.6 billion all-cash takeover. Plans to challenge the New York... [more »](#)

[Relationships](#)
[Interlocks](#)
[Giving](#)
[Political](#)

Business Positions

News Corp Global media company run by Rupert Murdoch; owner of Fox News, WSJ,
 • [Director](#) [+2]

The Directv Group Inc TV entertainment broadcasting, satellite networks and...
 • [Director \(past\)](#) [+1]

Other Positions & Memberships

The Partnership for New York City Nonprofit membership organization comprised...
 • [Co-Chairman \(past\)](#) [+1]

Cornell Medical College Cornell University's medical school, in NYC
 • [overseer](#)

Source Links

Articles documenting info on this page
 1-10 of 176 :: [see all](#)

[FEC contribution search](#)
[News Corp Form 4, Aug 19 '08](#)
[Forbes.com](#)
[News Corp board](#)
[Harlem Village Academies board](#)
[FEC Filing 21020062907](#)

Figura 3.1: *Interfaz de LittleSis, Rupert Murdoch*. Un ejemplo de como littlesis almacena la información de las personas y el tipo de personas presentes en littleSis.

La idea detrás de LittleSis de entregar el poder de la información a la gente es muy buena desde su punto de vista de análisis de estructuras sociales económicas y políticas. Sin embargo, no cubre el problema que trata resolver esta memoria, debido a que si bien las personas pueden complementar esta información, esto se hace de una manera centralizada, además de que no se puede agregar información asociada a otro tipos de redes sociales, como por ejemplo: la red de panaderos de un país, ya que no está dentro del objetivo y alcance de LittleSis, sin mencionar que esta idea está enfocada en Estados Unidos.

Poderopedia es el clon chileno de LittleSis, una plataforma centralizada por la cual se expone información sobre las redes de poder e influencias en el ambiente económico, social y político chileno, en donde nuevamente el objetivo es enfocarse en las personas de los más altos cargos empresariales y políticos en Chile.

PODEROPEDIA BETA

AZ PERSONAS EMPRESAS ORGANIZACIONES Sobre Poderopedia Registrarse **INGRESAR**

POPULAR AHORA: Marcel Claude Carlos Heller Solari Andrés Allamand Grupo Luksic

Buscar en todo Poderopedia

Michelle Bachelet
20-09-1951

Pediatra y política chilena. Fue presidenta de la República de Chile entre 2006 y 2010 y ex directora ejecutiva de ONU Mujeres hasta el 15 de marzo de 2013, día en que anunció su renuncia por "razones personales". Luego de tres años de vivir en Estados Unidos alejada de la contigencia política chilena arribó al país el 27 de marzo de 2013, ese mismo día, en un acto ciudadano en la comuna de El Bosque anunció formalmente su candidatura presidencial. El 4 de abril de 2013, el mismo día en que la Cámara de Diputados aprobó la acusación constitucional en contra del ministro Harald Beyer, Michelle Bachelet Jeria anunció su segundo comando presidencial con miras a las primarias de la Concertación: "Pacto Nueva Mayoría" el próximo 30 de junio.

[sugerir otro perfil] [sugerir conexiones] [reportar error] [reportar contenido inadecuado] [requiere actualización] [republicar] [ver historial]

Actualizado 11-06-2013

CONEXIONES PERFIL MAPA DE RELACIONES DOCUMENTOS FUENTES RIO DE NOTICIAS

Datos Básicos Mostrar

Familia (6 Items) Mostrar

Cónyuge o Pareja (3 Items) Mostrar

Amigos (12 Items) Mostrar

28 PERSONAS RELACIONADAS

0 EMPRESA RELACIONADAS

19 ORGANIZACIÓN RELACIONADAS

Figura 3.2: *Interfaz de Poderopedia, Michelle Bachelet*. Un ejemplo de como poderopedia almacena la información de las personas y el tipo de personas presentes en esta plataforma.

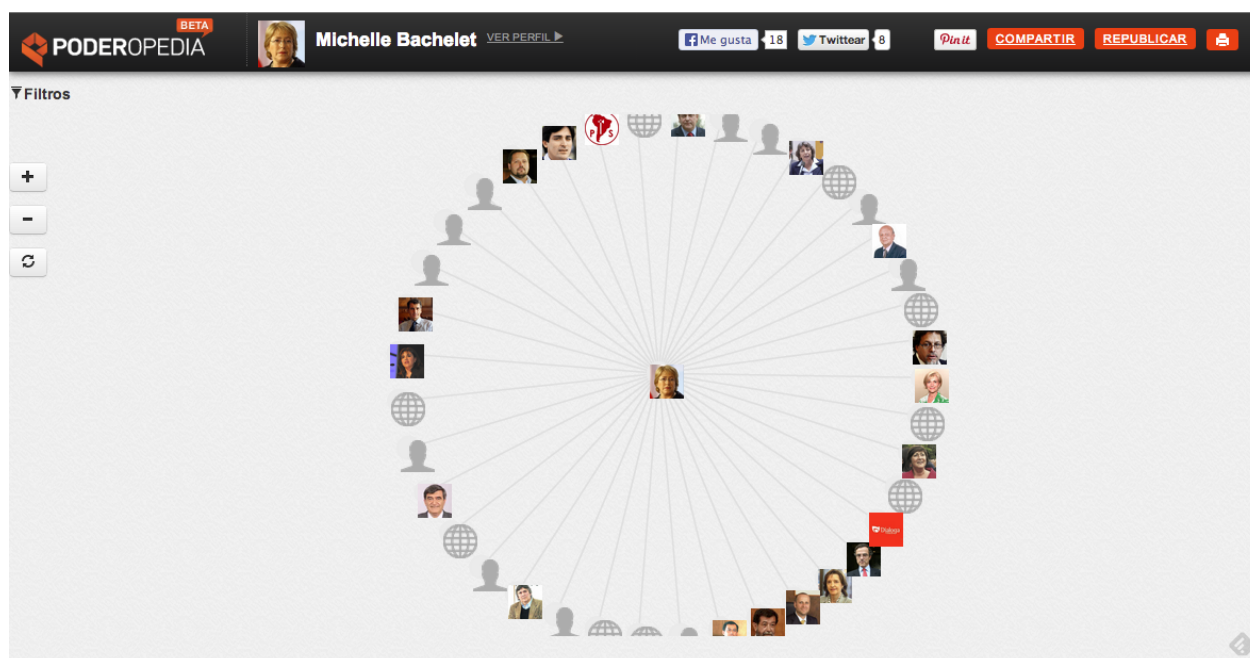


Figura 3.3: *Interfaz de Grafo de Poderopedia, Michelle Bachelet*. Poderopedia tiene una opción de visualización de grafo al rededor de una persona.

De la misma forma de LittleSis, las desventajas de Poderopedia en relación al problema

que se desea resolver en esta memoria, van por el lado de que su foco es exclusivamente la gente con poder político y económico del país, que además es centralizada, que su visualización de grafo no entrega información real de las estructuras sociales y no es fácilmente manipulable.

Ambos LittleSis y Poderopedia son excelentes herramientas, pero su objetivo difiere de lo que se desea resolver en esta memoria, a pesar de que la idea básica es la misma, exponer información sobre estructuras sociales.

3.1.2. Memoria Manuel Bahamonde

Un intento de resolver este problema fue desarrollado por el alumno del DCC Manuel Bahamonde, quien hizo su trabajo de memoria titulado: *Aplicación para la creación y administración de redes sociales* [27] el año 2009, memoria también guiada por el profesor Claudio Gutiérrez. Entonces, se gestó la idea de este trabajo de memoria, como una forma de mejorar lo desarrollado anteriormente por Manuel, aprovechando los avances tecnológicos y experiencias con el modelamiento de redes sociales y la web semántica.

La aplicación desarrollada en Java por Manuel, permite al usuario descargar un ejecutable y usarlo tanto en Windows, Linux o Mac, la cual permite crear redes sociales en forma de grafo, creando actores y sus relaciones respectivas, asociándolos a familias y pudiendo exportar la información de las redes sociales en formato para la utilización del software de análisis de redes sociales, pajek[15].

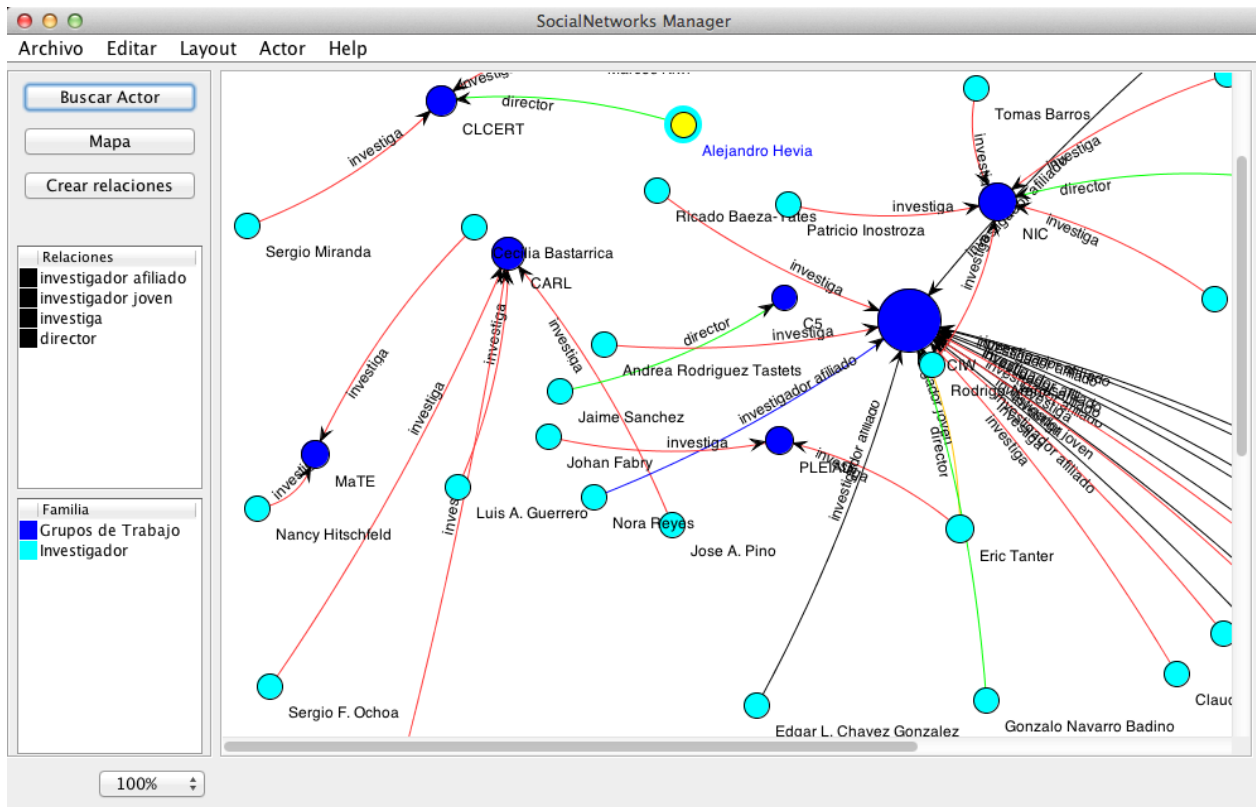


Figura 3.4: *Interfaz SocialNetworks Manager*. La aplicación creada por el alumno Manuel Bahamonde con el objetivo de modelar redes sociales como aproximación inicial al problema de esta memoria.

El trabajo de Manuel ayudó como experiencia para acercarse más a una solución para el modelamiento de redes sociales y con el conocimiento actual del tema, es necesario hacer modificaciones a lo que hizo Manuel, resolviendo algunos problemas extras, como:

1. El formato de almacenamiento de las redes sociales, no permite una interoperabilidad con otras aplicaciones fuera de pajek[15], en especial con las tecnologías de la web semántica.
2. Las relaciones entre actores sólo pueden ser 2 actores a la vez, no hay posibilidad de crear n -relaciones, o relaciones entre n actores.
3. Esta aplicación no permite complementar la información de una red social uniéndola con otra.
4. No se puede agregar atributos a los actores y relaciones.
5. Necesita que el usuario de la aplicación tenga instalado Java en su equipo.
6. No se puede acceder a la información del usuario desde cualquier computador.
7. La interfaz podría ser mejor en términos de usabilidad

3.1.3. Otras Alternativas

Dentro de las otras opciones disponibles al tema de esta memoria, se tiene algunos softwares de creación y visualización de grafos, como por ejemplo yED[26] u OnmniGraffle[14]. Este tipo de soluciones sólo genera los grafos de forma visual, sin tener los datos estructurados de la red para poder ser analizados posteriormente por otra aplicación, además de que no ayudan con la creación efectiva de redes sociales, pues no poseen un esquema concreto de redes sociales por ser herramientas de visualización multipropósito.

Otras herramientas existentes de la rama de análisis de redes sociales son por ejemplo Gruff[10] que es una herramienta de visualización de la base de datos AllegroGraph[1], la cual no posee las características que permiten crear redes sociales en forma estructurada, en base a un modelo estándar que se sepa que funciona, como el modelo SNM de Mauro San Martín[29]. Por otro lado, se encuentra Graph[9], que es una herramienta de visualización de grafos genéricos, que en este caso comparte los problemas de Gruff, para el manejo de redes sociales, un caso específico de grafo.

Por lo tanto dentro de las alternativas disponibles investigadas, entre otras, las características distintivas claves de este trabajo de memoria es la capacidad de unir redes sociales, el uso de un modelo estándar para redes sociales y la generación de datos RDF con la herramienta.

3.2. Relevancia del Problema

El problema es relevante de resolver, debido a que se está entregando una herramienta que le permitiría a personas que no tengan conocimientos avanzados de computación, almacenar redes sociales en forma gráfica interactivamente.

Además estas redes sociales anteriormente creadas, se pueden complementar uniéndolas con otras redes sociales creadas por otros usuarios de la aplicación, aprovechando el conocimiento de más personas sobre estructuras sociales determinadas.

Otro aspecto clave de resolver este problema con este tipo de aplicación corresponde a que los datos generados por sus usuarios son compatibles con los formatos y tecnologías pertenecientes a la web semántica, por lo cual esta información puede ser relacionada con otras fuentes de conocimiento posteriormente.

3.3. Usuarios Objetivo

La herramienta que se desarrollará para esta memoria está enfocada principalmente en personas en cuya profesión requiera el estudio de diversos tipos de redes sociales, las cuales pueden encontrarse en campos como: sociología, biología, historia, negocios etc. Es decir personas con un conocimiento básico de redes sociales y el estudio de estas, aplicadas a algún campo en particular.

3.4. Requisitos de la Aplicación

Una vez especificado el contexto del problema a resolver, se exponen los requisitos funcionales y de calidad de la aplicación que busca resolver las necesidades planteadas, escritas como requisitos que puedan ser usados posteriormente como criterios de aceptación de la aplicación del trabajo de memoria, un sistema de creación y manejo de redes sociales:

3.4.1. Requisitos Funcionales

1. Creación de usuario para identificación dentro del sistema.
2. Creación, edición y eliminación de redes sociales y sus componentes.
3. Dentro de las redes sociales, crear, editar y eliminar actores y sus atributos.
4. Dentro de las redes sociales, crear, editar y eliminar relaciones n -árias (entre n actores) y sus atributos.
5. Exportar los datos de una red social en formato RDF.
6. Importar/Cargar datos de una red social en formato RDF.
7. Unir redes sociales.

3.4.2. Requisitos de Calidad

1. Una aplicación fácil de usar.
2. Mantener la transparencia con el manejo de información, que las personas sean dueñas de sus datos.
3. Una aplicación de fácil instalación.

Capítulo 4

Descripción de la Solución

A continuación se describe la solución propuesta con motivo de este trabajo de memoria, partiendo de conceptos más teóricos para luego abordar algunos temas de decisiones de implementación y mostrar el resultado final de la aplicación que se desarrolló.

4.1. SNM, Social Network Model

Uno de los puntos principales de esta memoria, es poner en práctica el modelo realizado por el alumno de doctorado en el DCC, Mauro San Martín[29], quien investigó y propuso un modelo llamado *Social Network Model* o SNM. Por lo tanto, la generación de redes sociales en esta aplicación tendrán en cuenta este modelo, trabajo que fue guiado por el mismo profesor guía de esta memoria, aprovechando la experiencia e investigación previa sobre el tema. Entonces, se procederá a definir y explicar brevemente en qué consiste este modelo.

4.1.1. Elementos de Redes Sociales

El modelo de Mauro San Martín, tiene los elementos pertenecientes a las redes sociales, que fueron tratados en los conceptos de redes sociales en la sección 2.1, a los cuales se les agregan algunas restricciones:

- Los *Actores*: estos tienen un identificador único y un conjunto de atributos, además pueden participar en cualquier número de relaciones.
- Las *Relaciones* también tiene un identificador único, un conjunto de atributos y un número de actores participantes. El número de participantes puede ser uno o más, y puede cambiar sin afectar el resto de las propiedades de la relación.
- Los *Atributos* tienen un significado asociado y un valor literal. Un atributo es identificado por el identificador del objeto al cual está añadido (acto o relación), por su significado y su valor literal. La clase de un objeto (actor o relación) es un tipo de atributo especial llamado *familia*.

- *Actores, Relaciones, Atributos* y sus conexiones forman una red social. Compartiendo y reusando metadata al nivel, por ejemplo, proveniencia de conjuntos de datos.

4.1.2. Definición Matemática

Dado lo anterior, el modelo *SNM* describe una red social como un grafo de la siguiente forma:

Definición 4.1 (*Red Social Generalizada*) Una red social generalizada es definida como un multigrafo dirigido tripartito con etiquetas junto con una familia de funciones etiquetadoras f y un conjunto de familia de etiquetas L_f :

$$G = (N, E, L_N, L_E, L_f, \iota, \nu, \varepsilon, f)$$

Donde:

- El conjunto de nodos $N = A \cup T \cup C$ es una unión disjunta del conjunto de actores A , con el conjunto de relaciones T y el conjunto de atributos C .
- Existe una colección finita de familias (subconjuntos) de actores $\mathcal{A} = \{A_1, A_2, \dots, A_k\}$ de manera tal que cada $A_i \subseteq A$ y $\cup_{1 \leq i \leq k} A_i = A$.
- Existe una colección finita de familias (subconjuntos) de relaciones $\mathcal{T} = \{T_1, T_2, \dots, T_j\}$ donde cada $T_i \subseteq T$ y $\cup_{1 \leq i \leq k} T_i = T$.
- El conjunto de arcos $E = E_{AT} \cup E_{AC} \cup E_{TC}$ es la unión disjunta del conjunto de arcos entre actores y relaciones E_{AT} , con el conjunto de arcos entre actores y atributos E_{AC} , el conjunto de arcos entre atributos y relaciones E_{TC} .
- El conjunto de etiquetas de nodo $L_N = L_A \cup L_T \cup L_C$ es la unión disjunta de los conjuntos de etiquetas de actores L_A , de etiquetas de relaciones L_T y el de etiquetas de atributos L_C .
- El conjunto de etiquetas de arcos $L_E = L_{AT} \cup L_{AC} \cup L_{TC}$ es la unión disjunta de los conjuntos de etiquetas de arcos entre actores y relaciones L_{AT} (roles de participación), de etiquetas de arcos entre actores y atributos L_{AC} (significado de atributos de actores) y el de etiquetas de arcos entre relaciones y atributos L_{TC} (significado de atributos de relaciones).
- El conjunto de etiquetas de familias $L_f = L_{f_A} \cup L_{f_T}$ es la unión disjunta entre el conjunto de etiquetas de familias de actores L_{f_A} y el conjunto de etiquetas de familias de relaciones L_{f_T} .
- $\iota = \{\iota_{AT}, \iota_{AC}, \iota_{TC}\}$ es el conjunto de funciones de incidencia tales que $\iota_{AT} : E_{AT} \rightarrow A \times T$ es una función de incidencia que asocia cada arco de participación a un actor y su relación; $\iota_{AC} : E_{AC} \rightarrow A \times C$ es una función de incidencia que asocia un arco de significado a un actor y a un atributo; $\iota_{TC} : E_{TC} \rightarrow T \times C$ es una función de incidencia que asocia cada arco de significado a una relación y un atributo.
- $\nu = \{\nu_A, \nu_T, \nu_C\}$ es un conjunto de funciones etiquetadoras de nodos tales que $\nu_A : A \rightarrow L_A$ es una función biyectiva desde actores a las etiquetas de actores; $\nu_T : T \rightarrow L_T$ es una función biyectiva desde relaciones a las etiquetas de las relaciones; $\nu_C : C \rightarrow L_C$ es una función biyectiva desde atributos a las etiquetas de atributos.

- $\varepsilon = \{\varepsilon_{AT}, \varepsilon_{AC}, \varepsilon_{TC}\}$ es el conjunto de funciones etiquetadoras de arcos tales que $\varepsilon_{AT} : E_{AT} \rightarrow L_{AT}$ es una función desde arcos de participación a sus etiquetas; $\varepsilon_{AC} : E_{AC} \rightarrow L_{AC}$ y $\varepsilon_{TC} : E_{TC} \rightarrow L_{TC}$ son funciones desde arcos de significado a sus etiquetas.
- $f = \{f_A, f_T\}$ es el conjunto de funciones etiquetadoras de familias tal que $f_A : \mathcal{A} \rightarrow L_{f_A}$ es una función de familias de actores a las etiquetas de familias de actores y $f_T = \mathcal{T} \rightarrow L_{f_T}$ es una función de familias de relaciones a las etiquetas de familias de relaciones.
- La siguiente condición se mantiene para todos los arcos entre el mismo par de actores y relaciones, Para todo e_1 y e_2 de manera tal que $\iota(e_1) = \iota(e_2) = (u, v)$ con $u \in A$ y $v \in T$, $e_1, e_2 \in E \Leftrightarrow \varepsilon(e_1) \neq \varepsilon(e_2)$.
- Cada función de etiquetado en ν, ε y f , excepto ν_C , deben ser invertibles.
- Para una relación $r \in T$ entre dos actores $a_1, a_2 \in A$, tal que existe $e_1, e_2 \in E$, y $\iota(e_1) = (a_1, r)$, $\iota(e_2) = (a_2, r)$ con etiquetas $\varepsilon(e_1) = p_1$, $\varepsilon(e_2) = p_2$. La dirección de r puede ser especificada por el par ordenado de las etiquetas de participación, eso es que una dirección (p_1, p_2) indica que r comienza en a_1 y termina en a_2 , la dirección opuesta es representada por (p_2, p_1) .

De lo anterior se extrae información relevante sobre las características de las redes sociales y sus elementos:

- Un **Nodo** puede ser un *Actor*, *Relación* o *Atributo*.
- Las relaciones pueden ser de uno a múltiples actores.
- Los actores juegan un **Rol** en la relación.
- Los *Actores* y *Relaciones* pertenecen a **Familias de Actores** y **Familias de Relaciones** respectivamente.
- Una **Familia** de relación o actor, define un conjunto de actores/relaciones en común.

4.1.3. Representación Gráfica

En su tesis de doctorado[29], Mauro presenta un lenguaje gráfico para representar redes sociales con su modelo, el cual se adjunta a continuación por motivos de completitud del trabajo y para expresar cómo este modelo influye posteriormente el diseño de la interfaz del usuario.

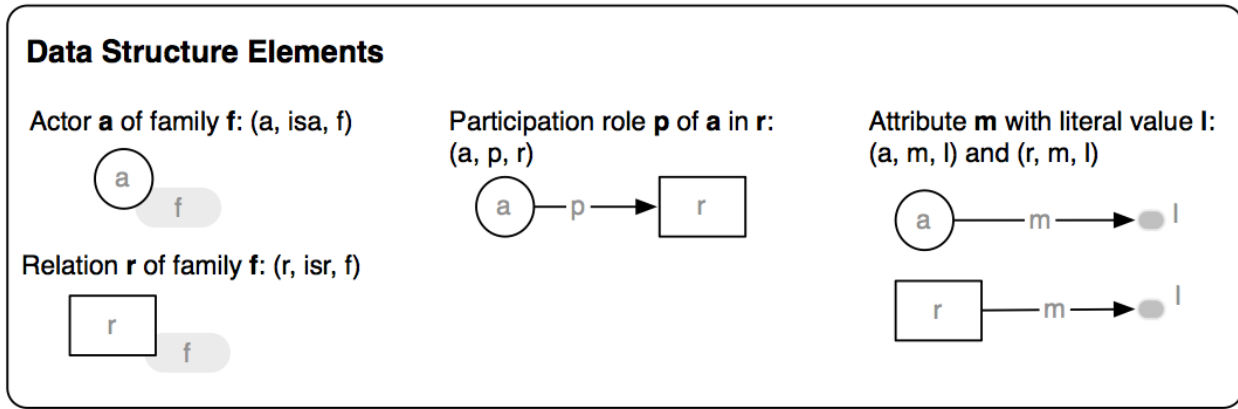


Figura 4.1: *Elementos Gráficos SNM*. Desde izquierda a derecha, los 4 bloques de construcción gráfica de una red social: un actor (arriba) y su etiqueta de familia, una relación y su etiqueta de familia, un rol de participación de un actor en una relación y atributos sobre actores y relaciones. Además por cada bloque se muestra la equivalencia en forma de triple.

Dados estos elementos gráficos podemos modelar una red de ejemplo como:

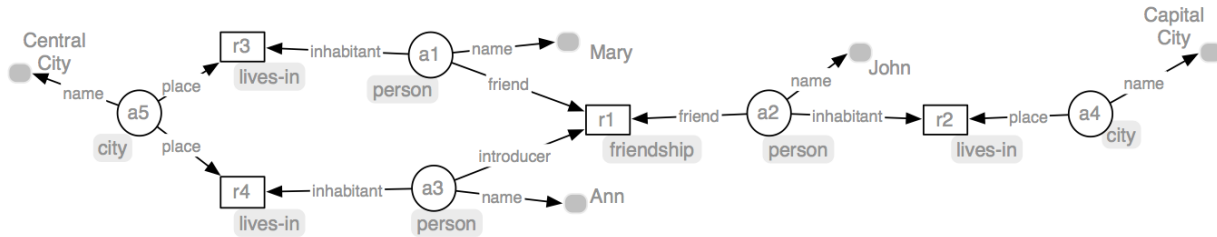


Figura 4.2: *Ejemplo Red Social de Amistad en SNM*. Una red social representando relaciones de amistad (nodos cuadrados) entre Mary y John, quienes fueron presentados por Ann (los actores se muestran como nodos circulares y sus atributos como puntos grises). Las ciudades de residencia también son representadas como actores.

4.1.4. Representación Como Triples

El modelo SNM, al ser una representación de grafo, es posible representarlo como triples y vice-versa (la demostración se encuentra en la memoria de Mauro[29]). Esto es particularmente útil para usar este modelo dentro de un contexto de bases de datos relacionales, razón por la cual veremos una de estas transformaciones a continuación.

Es posible traducir una red social dada, representada como tres conjuntos de triples (N, R, M) , en donde N es el conjunto de triples de *Tipos*, R el de *Roles* y M de *Atributos*; en una red social generalizada G con el siguiente algoritmo que produce un nodo o arco por cada triple:

- Para cada triple $(\nu_A(a), \text{isa}, f_A(A_i))$ en N , se añade un nodo actor a en A_i .
- Para cada triple $(\nu_T(t), \text{isr}, f_T(T_i))$ en N , se añade un nodo de relación t a T_i .
- Para cada triple $(\nu_A(u), \varepsilon_{AT}(e), \nu_T(v))$ en R , se añade un arco de participación e a E_{AT} con nodos finales $u \in A$ y $v \in T$.
- Para cada triple $(\nu_A(u), \varepsilon_{AC}(e), \nu_C(v))$ en M , se añade un nodo de atributo v a C , y su arco de significado e a E_{AC} con nodos finales $u \in A$ y $v \in C$.
- Para cada triple $(\nu_T(u), \varepsilon_{TC}(e), \nu_C(v))$ en M , se añade un nodo de atributo v a C , y su arco de significado e a E_{TC} con nodos finales $u \in T$ y $v \in C$.

Por ejemplo se muestra la siguiente tabla que representa este conjunto de triples, que es similar a lo que se tendría en una base de datos relacional para el caso de la red de amistades de la figura 4.2.

N: Typing			R: Roles			M: Attributes		
a1	isa	'person'	a1	friend	r1	a1	name	'Mary'
a2	isa	'person'	a2	friend	r1	a2	name	'John'
a3	isa	'person'	a3	introducer	r1	a3	name	'Ann'
a4	isa	'city'	a2	inhabitant	r2	a4	name	'Capital City'
a5	isa	'city'	a4	place	r2	a5	name	'Central City'
r1	isr	'friendship'	a1	inhabitant	r3			
r2	isr	'lives-in'	a5	place	r3			
r3	isr	'lives-in'	a3	inhabitant	r4			
r4	isr	'lives-in'	a5	place	r4			

Tabla 4.1: Representación en triples de la red social de la figura 4.2

4.2. Arquitectura de la Aplicación

4.2.1. Arquitectura de Software

La solución considera una aplicación para el manejo de grafos, que consta con un back-end que provee servicios para el almacenamiento y persistencia de las redes sociales junto con un front-end el cual despliega la lógica del editor mismo que los usuarios usan en el día a día.

Este esquema permite la creación de diversos sistemas distintos que se alimenten de la data generada por la aplicación como programas completamente independientes. Por otro lado, debido a que el front-end de la aplicación usa un framework client side como EmberJS, el código de este front-end se obtiene inicialmente desde el back-end, para luego correr en el dispositivo del cliente, donde existirá comunicación entre los componentes de la aplicación, vía peticiones HTTP de tipo JSON para la sincronización y almacenamiento de los datos, mencionando además que estas llamadas son todas de forma autenticada, por lo que el trabajo de diversos usuarios no colisiona entre sí. Para mostrar de mejor manera lo recién descrito, se agrega una figura de la arquitectura general de la aplicación y como sus usuarios interactúan con ella (ver figura 4.3).

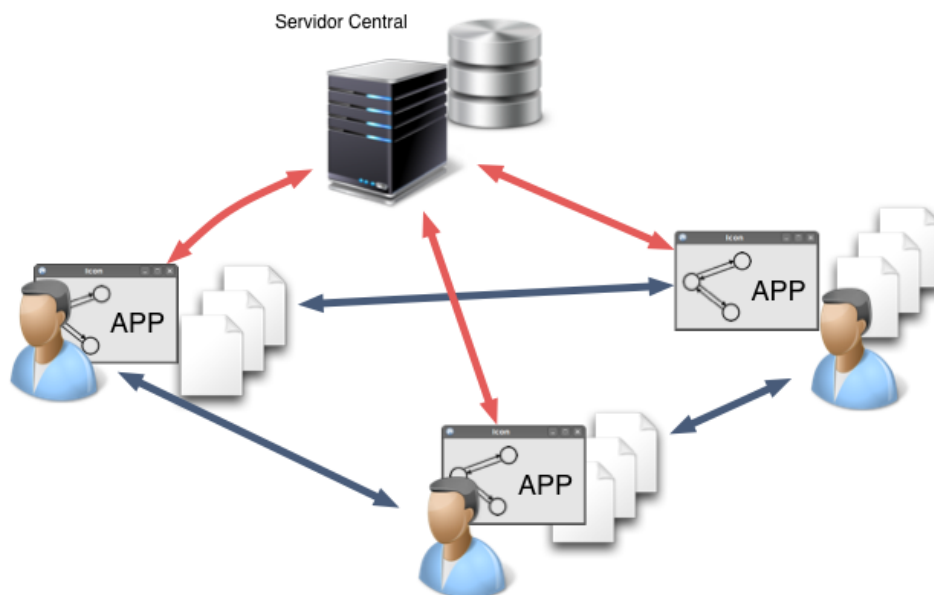


Figura 4.3: *Arquitectura General Aplicación-Usuarios*. Se puede observar que la aplicación del lado del cliente se comunica con un servidor central que almacena los datos, pero además diversos usuarios de la aplicación pueden comunicarse independientemente entre ellos por medio de archivos RDF/N3.

Los usuarios pueden interactuar entre sí intercambiando información generada entre ellos por medio de los archivos de exportación RDF/N3, esta forma de interacción de los usuarios de la aplicación es una aproximación inicial a este problema en particular, debido a las restricciones de tiempo del trabajo de memoria, junto con la gran cantidad de otros problemas y detalles que solucionar primero.

Este esquema general, debido a la calidad de proyecto de código abierto, puede ser replicado por terceros, que puedan instalar su propio servidor de la aplicación de manera de tener un ambiente de redes sociales privados, como puede ser el caso de una universidad en particular.

Según lo que se comentó anteriormente, esta arquitectura posee la potencialidad de integrar diversas otras aplicaciones que usen este tipo de datos de redes sociales, sin embargo, como la lectura en sí de los datos sólo se hace en el lado del cliente, en el servidor no se entrega ningún dato que no pertenezca a su propio cliente, de esta forma, no hay conflictos con la privacidad de la información, donde la aplicación entrega la funcionalidad al usuario y este se hace cargo del uso que hace de esta.

En una vista más en detalle de la aplicación, para explicar de mejor manera como funciona el software internamente, a continuación se muestra un diagrama de cómo este está estructurado por sus componentes.

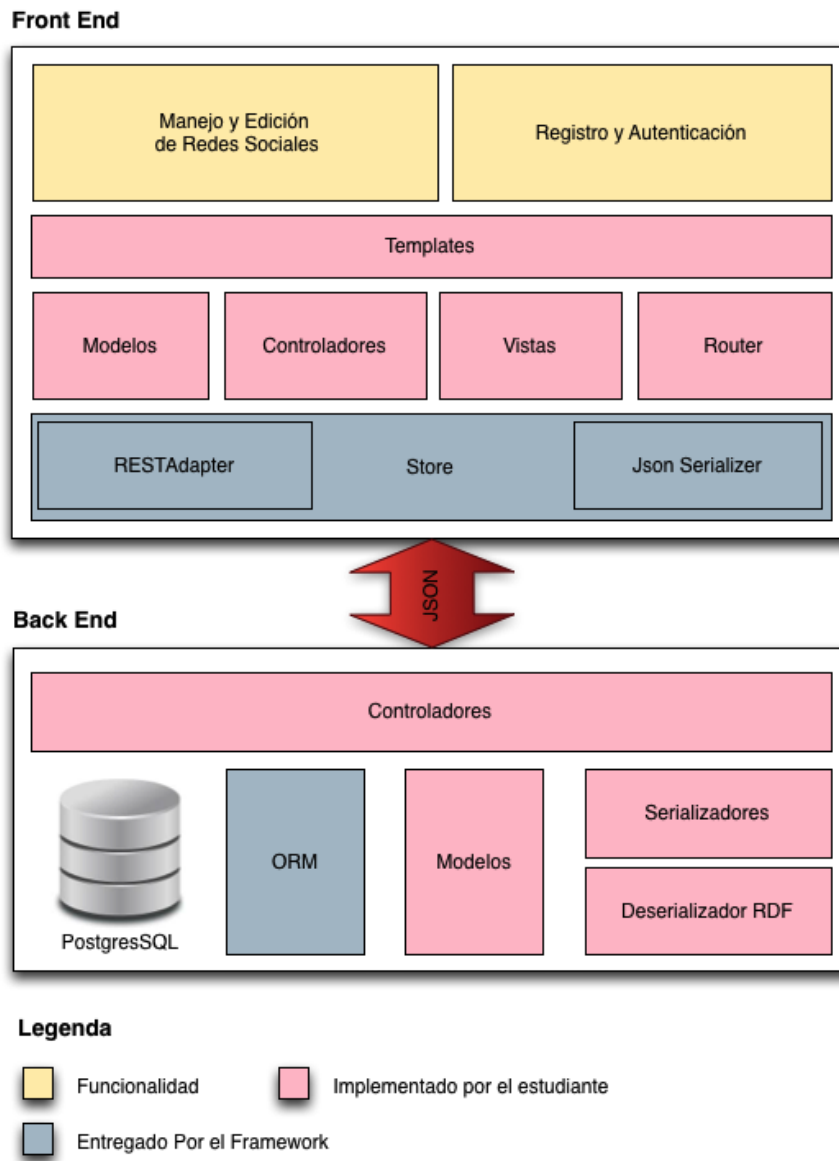


Figura 4.4: *Arquitectura de la Aplicación*. En una vista de componentes, donde algunos de estos son entregados por las herramientas utilizadas.

Desde punto de vista del back-end, se definen los *modelos* correspondientes a la aplicación (Node, Role, etc) los cuales se definen como clases, luego el **ORM** es el componente encargado de convertir ciertos métodos del modelo en código SQL que se envía a **PostgreSQL**. De esta forma, al trabajar con los modelos no es necesario escribir nada de SQL, haciendo la programación mucho más fácil. La interfaz que posee el back-end la conforman un conjunto de *controladores*, quienes son los encargados de responder las peticiones HTTP que llegan al servidor, en donde interactúan con los modelos y como se trata de en su mayoría de API JSON, es necesario definir componentes que representan un modelo de la base de datos en formato JSON, esto lo hacen los *serializadores*. Por su parte, hay un *deserializador RDF* que hace el proceso inverso y crea modelos a partir del contenido de un archivo RDF, lo que se usa en la importación de redes sociales.

Para el front-end, en donde se definen las interfaces gráficas con las que el usuario trabaja, además de definir la lógica correspondiente a la aplicación misma, esta se abstrae casi completamente del back-end siendo su único punto de conexión las API JSON en donde intercambian mensajes hacia ambos lados, donde en el caso del front-end esto lo maneja un componente de EmberJS llamado *Store*.

Se observa en la figura 4.4 que existen modelos, en el front-end que son los mismos presentes en el back-end, sin embargo difieren ligeramente en la lógica de operaciones con respecto a la edición y alternativamente el back-end puede cambiar completamente y mientras la API se respeta, los modelos del front-end no son afectados de ninguna forma.

4.2.2. Entidades

Para comprender de mejor manera la aplicación y los diferentes procesos, es necesario especificar qué entidades existen y qué rol cumplen dentro de la aplicación. Estas están con sus nombres en inglés, idioma que se usó para escribir el código de la aplicación.

- **User:** representa a un usuario del sistema, este cuenta con la información necesaria para que un usuario de la aplicación pueda autenticarse dentro del sistema y de esa forma entregarle los permisos necesarios para realizar operaciones sobre sus redes sociales.
- **Social Network:** es la unidad principal dentro de la aplicación, es la entidad que agrupa toda la información que se desea plasmar dentro de la aplicación y representa al grafo completo que forma la red social.
- **Node:** es una unidad atómica dentro de una red social que tiene un *tipo* que puede ser **Actor** o **Relación**, dado que ambos son equivalentes dentro del modelo de Mauro [29]. Un nodo tiene un atributo por defecto y opcional llamado *name* y además puede estar asociados a familias. En general se refiere a *Actor* o *Relación* cuando se necesita hacer una distinción entre ellos o simplemente *Nodo* cuando se habla de ambos.
- **Family:** una familia representa un conjunto para clasificar en tipos a nodos, las familias por su parte cuentan con un tipo que indica a que tipo de nodo pueden ser asociadas (Actor o Relación), junto a esto tienen algunas opciones de presentación, como el color con que se mostrarán los nodos pertenecientes a una familia.
- **Node Attribute:** son los atributos de un nodo, representados como un par key-value, cuyo nombre no fue *Attribute* para evitar conflictos de nombres con los frameworks de desarrollo.
- **Role:** representa la participación de un *Actor* dentro de una *Relación*. Posee un nombre que es opcional para identificar un rol específico dentro de una relación, por ejemplo: la persona que presenta a otras 2 en una relación de amistad.

4.2.3. Diseño de la Base de Datos

A partir de las entidades definidas anteriormente, a continuación se presenta la representación en la base de datos, en este caso PostgreSQL, en donde las relaciones siguen la convención de Ruby on Rails, es decir, `model_name_id`, los nombres de tabla, son iguales al nombre del modelo en plural en minúscula, ej: la tabla del modelo *Node* se llama *nodes*, además Rails crea campos de tiempo de creación y actualización que los maneja internamente como ayuda al desarrollo.

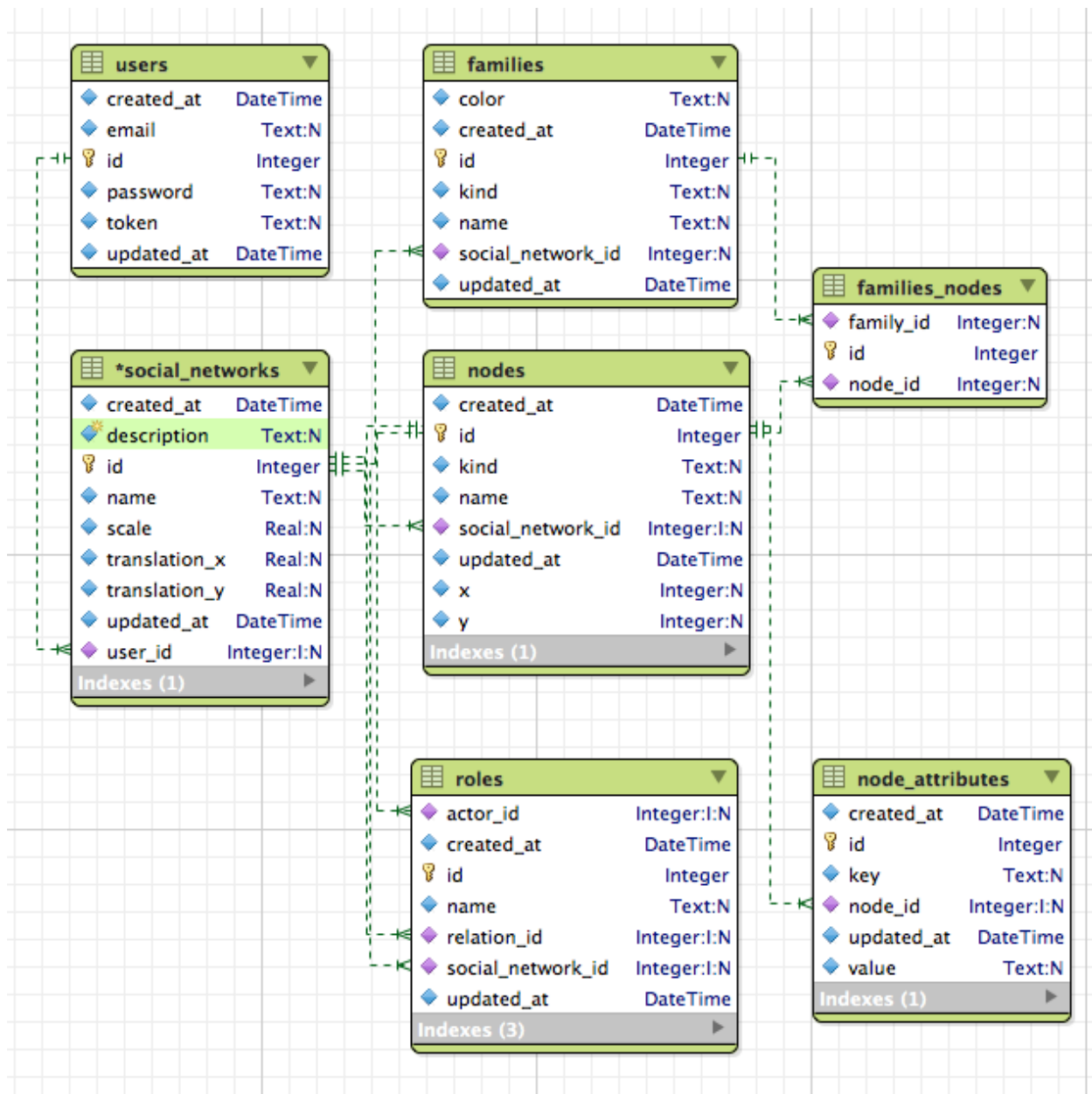


Figura 4.5: *Esquema Base de Datos de La Aplicación*. Una visualización del esquema de la base de datos con sus relaciones correspondientes.

Esta representación se basa en la descrita por el modelo de Mauro en la sección 4.1.4, con las extensiones correspondientes al dominio de este problema, que se listan a continuación:

1. El modelo de Node, contiene campos de posición x e y para almacenar su posición en el canvas.
2. El modelo de Social Network, no existente en el modelo de Mauro, se crea con el fin de agrupar la entidad de una red social, donde tiene detalles de la misma, además de algunos campos para recordar preferencias de visualización, como por ejemplo la escala y la translación del canvas.
3. Las familias poseen un campo de color para visualizar los nodos que pertenecen a ella.
4. La tabla *families_nodes* es la forma de Ruby on Rails de crear relaciones de N a M.
5. La tabla de roles posee también una referencia a su red social, debido a de esta forma se puede obtener de una manera simple, toda la información de una red social en una sola query, sin problemas de duplicidad por las referencias al rol por parte de actores y relaciones.
6. La tabla *node_attributes* posee un nombre que evita conflictos de nombre con las herramientas usadas en el desarrollo.

En relación a los campos visuales dentro de los modelos, es perfectamente viable tener un modelo separado para manejar toda la información que se refiere a estilos, pero debido a que esta es muy poca, se optó por tenerla en sus mismos modelos por simplicidad.

Capítulo 5

Funcionamiento de la Solución

En este capítulo se mostrarán las características principales de la aplicación desarrollada y a modo de referencia para usuarios, mostrando pantallazos de la interfaz, discutiendo las decisiones de diseño de interfaces y experiencia de usuario.

Cuando el usuario se conecta por primera vez a la aplicación encontrará la siguiente página de inicio donde este se registrará o autenticará, junto con un video de un demo visual sobre cómo funciona la aplicación.

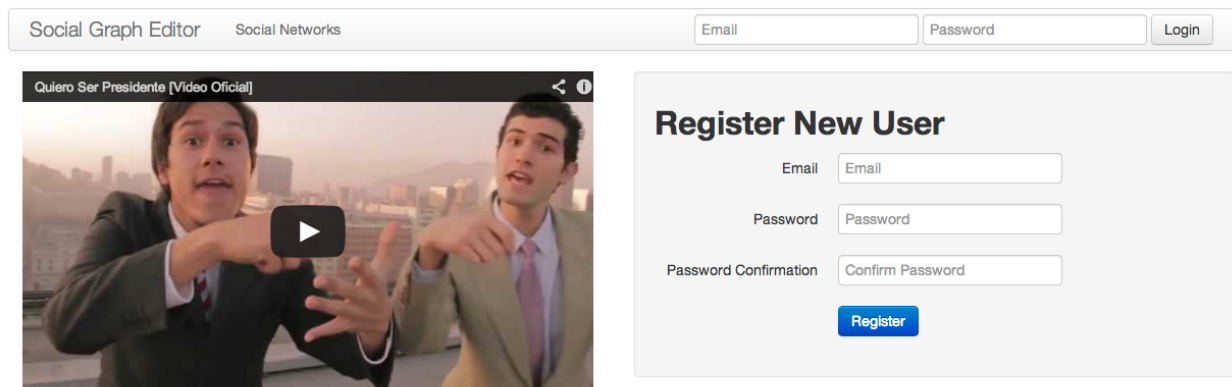
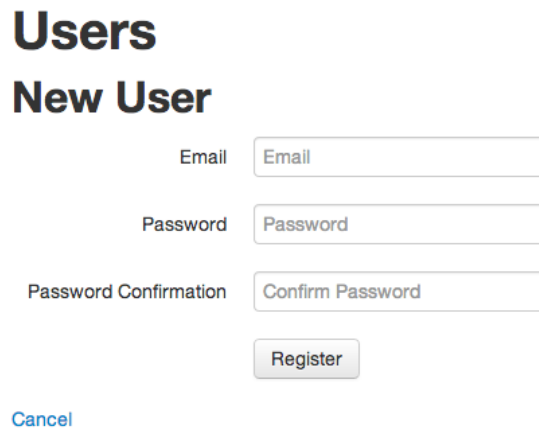


Figura 5.1: *Interfaz de Inicio de La Aplicación*. Esta interfaz está pensada para que el usuario se conecte o registre al sitio, junto con entregar un demo en forma de video de como esta aplicación funciona.

A continuación se procede a mostrar los principales procesos que se llevan a cabo con la aplicación con sus respectivas interfaces de usuario vistas en detalle.

5.1. Registro e Ingreso de Usuarios


Con el fin de mantener la privacidad de las redes sociales que sus usuarios crean, se necesita un sistema de autenticación de usuarios, el cual en este caso requiere una combinación de email y password. Al ingresar a la aplicación por primera vez, el usuario crea su cuenta, es autenticado y redirigido a la página principal donde se muestran sus redes sociales.



The screenshot shows a web form titled "Users" with a subtitle "New User". It contains three input fields: "Email", "Password", and "Password Confirmation" (labeled "Confirm Password"). Below the fields is a "Register" button and a "Cancel" link.

Figura 5.2: Formulario de Creación de Usuarios

Cuando el usuario accede de nuevo a la aplicación, esta vez en lugar de registrar un usuario debe autenticarse de modo de acceder a su información vía este pequeño formulario.



The screenshot shows a web form titled "Users" with a subtitle "Login". It contains two input fields: "Email" and "Password", followed by a "Login" button.

Figura 5.3: *Formulario de Ingreso*. Con una combinación de email y password el usuario de autentifica en el sistema.

5.2. Creación de Redes Sociales

Dado que el usuario registró una cuenta en la aplicación, esta habilitado para crear una red social, inicialmente en la pantalla principal de redes sociales, presiona el botón *Add Social Network* con el cual aparece el formulario de creación que es para darle un nombre y una descripción a la red social como detalles de esta.



Figura 5.4: *Pantalla Principal de Redes Sociales*. Es la primera página luego de que el usuario se autentifica, la cual le permite crear redes sociales para luego complementar su información.

The screenshot shows the 'New Social Network' form. The header is identical to the previous screenshot. On the left, the list of existing networks is also present. The main area is titled 'New Social Network' and contains a 'Name' input field. Below the input field are two buttons: 'Cancel' and 'Create'. The 'Create' button is highlighted in blue.

Figura 5.5: Formulario Nueva Red Social

Esta información ingresada de la red social sirve como detalles de la misma y puede ser editada en cualquier momento, sin embargo, lo más relevante de la creación de la red, es la adición de contenido a la estructura de esta, lo que se cubrirá en la siguiente sección.

5.3. Edición de Redes Sociales

Una vez que el usuario tiene su cuenta y creó una red social, está listo para ir agregando los datos relevantes a sus redes sociales, haciendo click sobre el link con el nombre de la red social recién creada.

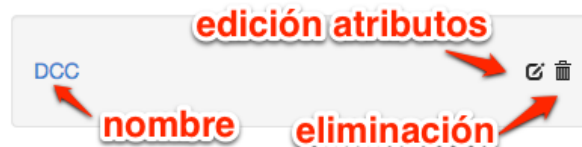


Figura 5.6: *Lista de Redes Sociales*. En este lugar aparecen las redes sociales creadas, donde el nombre llega a la edición del contenido de esta, además de contar en el costado con botones para editar sus detalles o eliminar la red.

5.3.1. Área de Edición de Redes Sociales

El área de edición de redes sociales consiste básicamente de 4 elementos: el canvas donde se crea el grafo de la red social (1), una barra de herramientas para la edición (2), un área donde se definen las familias de actores y relaciones (3) y un formulario con los detalles de las entidades seleccionadas (4).

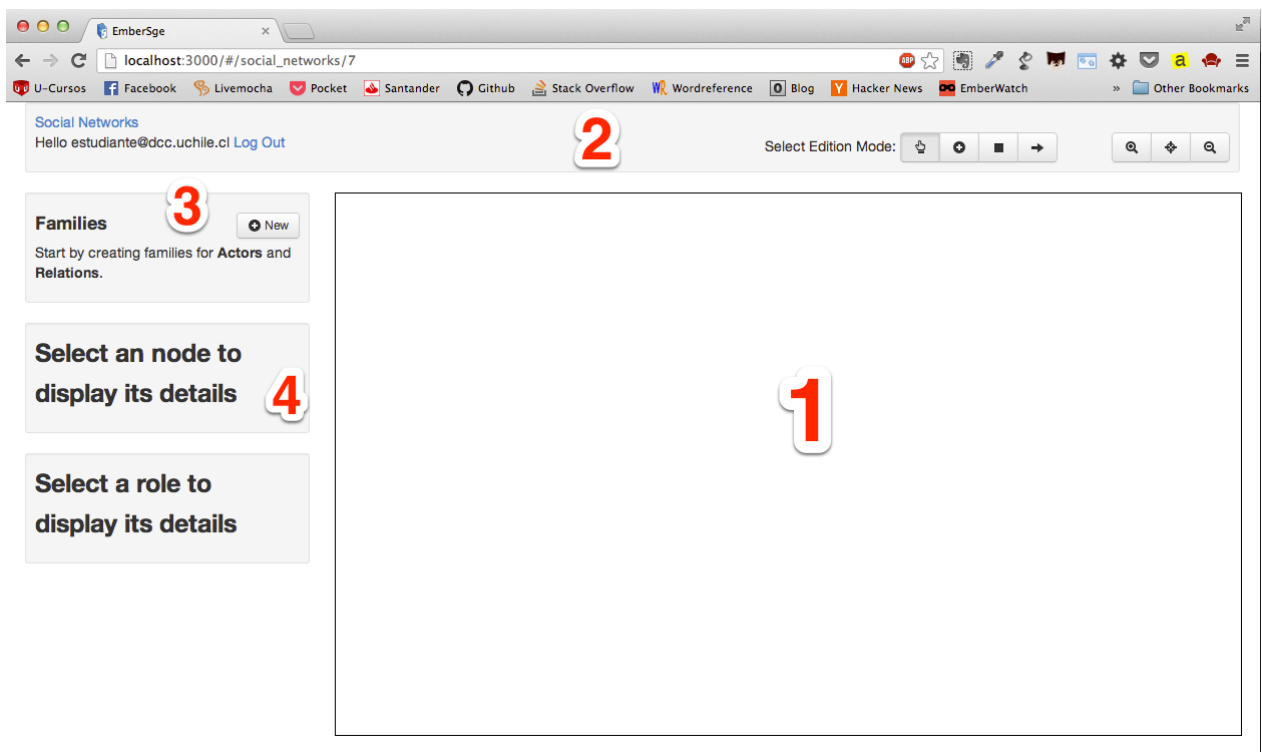


Figura 5.7: *Área de Edición de Redes*. Esta es el área principal donde el usuario completa y manipula la información de las redes sociales creadas, con números para la explicación de las diversas secciones de esta interfaz.

Esta interfaz está pensada para tener la menor cantidad de elementos posibles, haciendo énfasis en las herramientas primordiales necesarias para la edición del grafo.

5.3.1.1. Modos de Edición

Para la zona de edición se cuenta con 4 modos de edición, de acuerdo a los cuales puedo realizar acciones diversas cuando hago click dentro de la zona del canvas.

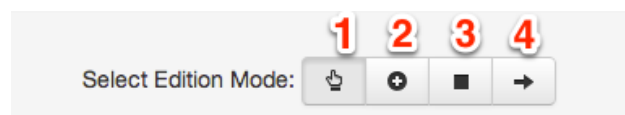


Figura 5.8: *Modos de Edición*. Botones para seleccionar el modo de edición, con números para explicar los diversos modos.

Los modos existentes son:

1. **Herramienta de Movimiento:** esta herramienta me permite cambiar la posición de los nodos dentro del canvas a voluntad, además de al hacer click en estos seleccionarlos

para la edición de sus detalles.

2. **Modo Actor:** en este modo, al hacer click en algún punto del canvas creará un nuevo actor dentro de esta en la posición donde se especificó haciendo click. El enfoque será puesto automáticamente en el formulario de detalles del nuevo actor para rellenar rápidamente sus campos necesarios y confirmar la creación del nuevo actor.
3. **Modo Relación:** en este modo, al hacer click en algún punto del canvas creará una nueva relación situada en esas coordenadas. El enfoque será puesto automáticamente en el formulario de edición de la relación, pero a diferencia de los actores, las relaciones son automáticamente guardadas al momento de ser creadas y cambian al *modo Rol*.
4. **Modo Rol:** en este modo, al momento de hacer click en un actor (manteniendo el botón presionado), puedo arrastrar una flecha y situarla sobre una relación, donde al soltar el botón del mouse, creará automáticamente un nuevo rol del actor seleccionado en la relación seleccionada, para después editar sus detalles en el formulario de roles correspondiente. Además, puedo repetir esta acción cuantas veces sea necesario.
5. **Modo Unión de Nodos:** en este modo, al arrastrar un nodo (actor o relación) sobre otro del mismo tipo provocará la aparición de un cuadro de confirmación de la unión entre nodos, que de ser confirmada hará que se unan todas las propiedades de estos en uno solo, en caso contrario el nodo arrastrado retornará a su posición original.

5.3.2. Creación y Edición de Familias

Para poder agrupar los nodos (actores y relaciones) dentro de familias, hay un área reservada para la creación propia de estas familias dentro de la red, en donde a continuación se explican las principales operaciones con familias dentro de una red social.

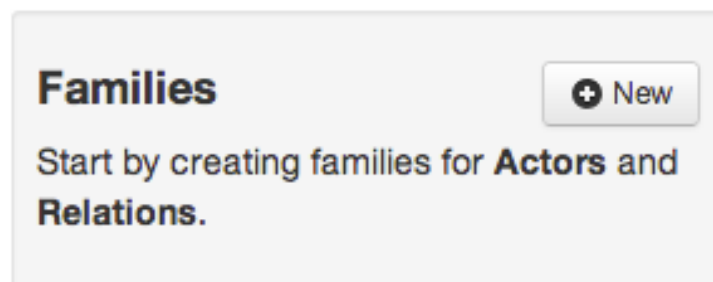


Figura 5.9: *Área de Familias*. Detalle del área donde se crean las familias en una red social.

Para crear una familia se presiona el botón *New* en la sección de familias, con lo que aparece un formulario con el siguiente:

Figura 5.10: *Formulario de Edición de Familias*. Formulario por el cual se define el nombre, color y tipo de familia.

Acá se rellena el nombre y se selecciona el color para mostrar los nodos de esta familia y el tipo de nodos a los cuales se les asignará esta familia.

Una vez creada, se muestra la familia en el listado con un ícono de *A* para el caso de familias de actores y de *R* en el caso de familias de relaciones. Donde puedo editar sus detalles o eliminarlas con los botones que se encuentran a su lado.

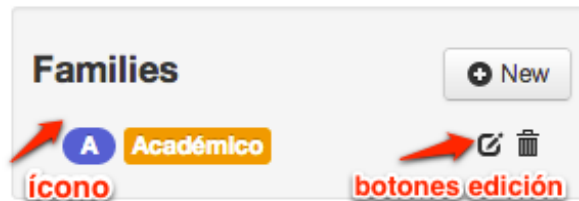


Figura 5.11: *Detalle Familia Creada*. Un ejemplo de una familia creada, con su ícono que indica si corresponde a familia de Actores o Relaciones, además de sus botones de edición y eliminación.

Es importante destacar, que se puede presionar cualquier tipo de familia, en donde al hacer esto, se cambiará al *modo Actor* o *modo Relación* según corresponda y a continuación cuando creo un actor o relación, por defecto pertenecerá a la familia seleccionada.

5.3.3. Creación y Edición de Actores

Para crear actores, se debe definir el modo de edición a *modo Actor* 5.3.1.1, y hacer click en el canvas donde aparecerá un actor en dicho punto y se enfocará automáticamente el

formulario de creación del actor.

Details for Actor #

Name

Families

after saved you can add attributes to this node

Figura 5.12: Formulario de Creación de Actor

En este formulario se puede rellenar el nombre (opcionalmente), seleccionar las familias a las que pertenece el actor para finalmente confirmar la creación. Luego de esto, la información visual del actor es actualizada, mostrando al actor del color de la(s) familia(s) a la cual pertenece, además de un borde indicando de que es dicho actor el que está seleccionado en este momento.

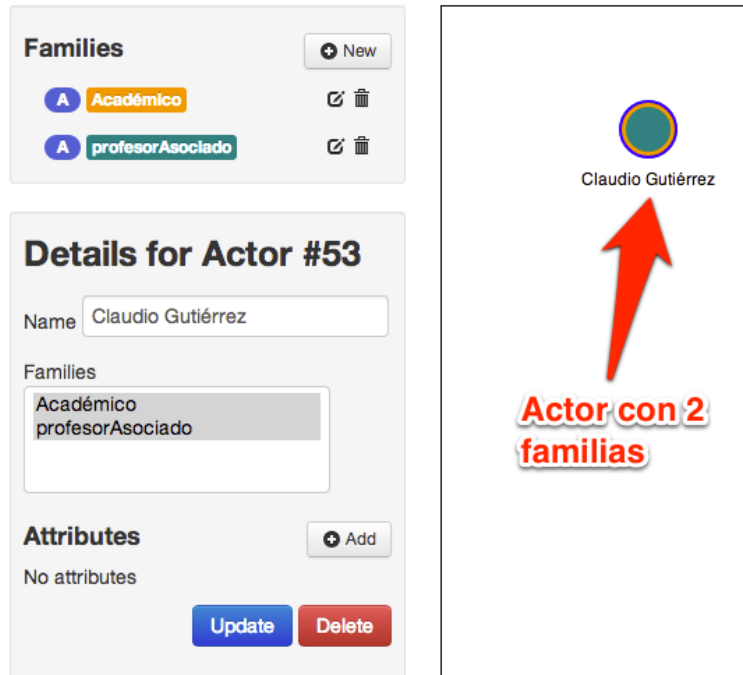


Figura 5.13: *Ejemplo Actor con 2 Familias*. Un actor con 2 familias tendrá un círculo con 2 colores de sus familias correspondientes.

El actor puede ser editado en cualquier momento vía el formulario de actor, luego se presiona el botón de actualizar para persistir los cambios, o puede ser eliminado con el botón de borrar, después de confirmar en el cuadro de diálogo que aparece.

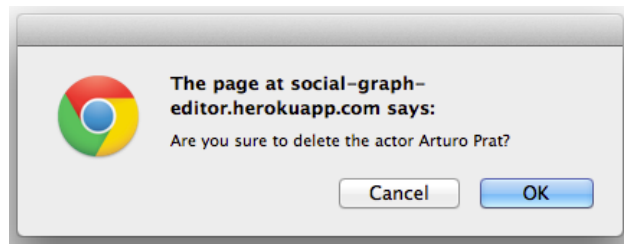


Figura 5.14: Diálogo de Eliminación de Actor

5.3.4. Creación y Edición de Relaciones

Para crear una relación, se debe seleccionar el *modo Relación* 5.3.1.1, posteriormente hacer click dentro del canvas en donde aparecerá la nueva relación y se mostrará el formulario de edición de la relación. A diferencia de los actores, las relaciones son creadas automáticamente, lo cual cambiará al modo de edición de roles, para agregar los roles correspondientes a la relación sin perder el contexto.

Details for Relation #

Name

Families

-

after saved you can add attributes to this node

Figura 5.15: Formulario de Creación de Relación.

Las relaciones pueden o no tener un nombre, además de pertenecer a una familia, lo que generalmente denota el tipo de relación con la cual se está trabajando, ej: estudiaEn, dueñoDe, etc.

La relación puede ser editada en cualquier momento vía el formulario y presionando el botón de actualizar, o eliminada con el botón de borrar, luego de confirmar el cuadro de dialogo que aparece.

5.3.5. Creación y Edición de Atributos en Nodos

Una vez teniendo actores y relaciones creados dentro de la red social, es posible agregarles todos los atributos que se estimen conveniente por medio del formulario de edición de actores o relaciones, para esto, en la subsección de atributos en dicho formulario se puede agregar uno presionando el botón *Add*, en donde puedo ingresar un atributo como un par key-value, por ejemplo puedo agregar el atributo *Edad* (key) con el valor *24* a un actor.

Details for Actor #53

Name

Families

Attributes + Add

Update Delete

Figura 5.16: *Añadiendo Atributos a un Actor*. Se presiona el botón Add y luego se puede agregar el nuevo atributo como un par Key Value.

Para editar atributos, se pueden editar directamente en sus campos y luego presionar el botón *Update* para que los cambios sean persistentes, o borrar un atributo presionando el ícono junto a la definición del mismo.

5.3.6. Creación y Edición de Roles

Los roles representan la participación de un actor en una relación, dicha participación o rol, puede tener un nombre o no. Un ejemplo del último caso: en una relación de amistad entre dos personas, puede haber un tercer actor que fue quien los presentó, pero nuevamente, el nombre de un rol es opcional. Para crear un rol, se debe seleccionar el modo de edición de roles 5.3.1.1, luego con el mouse, pincho un actor y arrastro el mouse hacia una relación, al soltar el mouse el rol va a ser creado inmediatamente y el foco va a ser puesto dentro del formulario de edición del rol.

Details for Role #39

Name

Figura 5.17: *Formulario de Edición de Rol*. En caso de ser necesario, se puede agregar un nombre al rol o eliminarlo desde aquí.

En este formulario puedo actualizar el nombre del rol o de ser necesario eliminar el rol. Es importante mencionar que los roles sólo serán creados desde un *Actor* hacia una *Relación*, cualquier otra combinación no resultará en la creación de un rol.

5.4. Exportación en RDF

Como primer paso en la integración de la aplicación con la web semántica, esta permite una exportación de las redes sociales modeladas en formato RDF/N3. Con este fin, se usa la estructura de triples descrita en el modelo de Mauro en la subsección 4.1.4. Entonces, para exportar la información de la red social en este formato puede hacerse vía el botón *Export in RDF/N3* en el menú de edición de redes sociales.

Export in RDF/N3 ▾

- Structure
- Structure + Visual Data
- Vocabulary

Figura 5.18: *Botón de Exportación de la Red Social*. Con este botón se puede exportar en formato RDF/N3: la estructura de la red social, la estructura más la información visual de la red o el vocabulario de esta red.

Por ejemplo, podemos tener el caso de una red social como la de la figura 5.19:

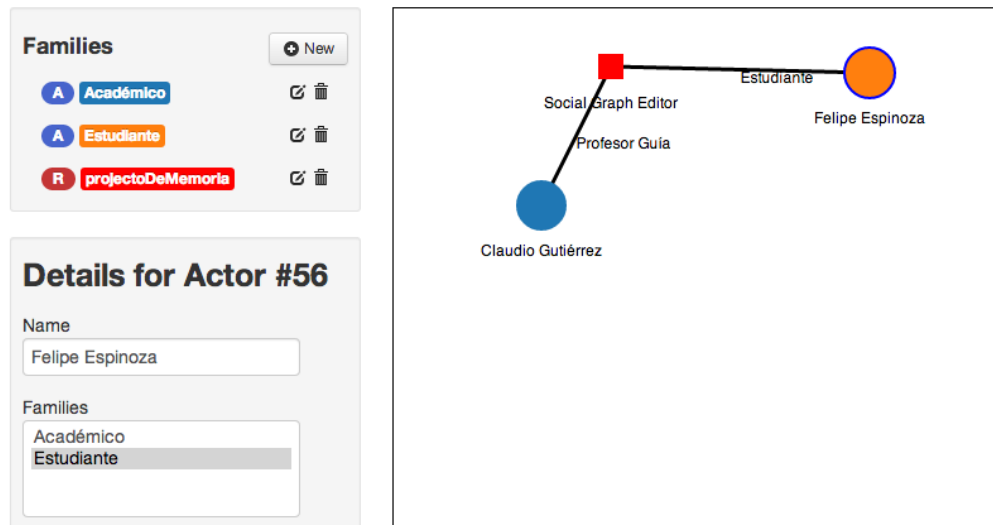


Figura 5.19: *Micro Ejemplo de Red Social*. Una red social que muestra la relación de un estudiante con su profesor guía.

Exportando la estructura únicamente de esta red social, que contiene 2 actores y una relación, se obtendría el siguiente resultado

```

1  @base <http://sn.dcc.uchile.cl/2013/> .
2  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3  @prefix sn: <http://sn.dcc.uchile.cl/social_networks/8/vocabulary#> .
4
5  </social_networks/8> a sn:socialNetwork;
6    sn:name "Memorias" .
7
8  </social_networks/8/nodes/55> a sn:actor;
9    sn:belongsToFamily </social_networks/8/families/19>;
10   sn:kind "Actor";
11   sn:name "Claudio Gutierrez";
12   sn:participatesAs </social_networks/8/roles/40> .
13
14  </social_networks/8/nodes/56> a sn:actor;
15   sn:attributeEdad "24";
16   sn:belongsToFamily </social_networks/8/families/20>,
17     </social_networks/8/families/23>;
18   sn:kind "Actor";
19   sn:name "Felipe Espinoza";
20   sn:participatesAs </social_networks/8/roles/41> .
21
22  </social_networks/8/families/19> a sn:family;
23   sn:kind "Actor";
24   sn:name "Academico" .
25
26  </social_networks/8/families/20> a sn:family;
27   sn:kind "Actor";
28   sn:name "Estudiante" .
29
30  </social_networks/8/families/21> a sn:family;
31   sn:kind "Relation";
32   sn:name "proyectoDeMemoria" .
33
34  </social_networks/8/families/23> a sn:family;
35   sn:kind "Actor";
36   sn:name "Memorista" .
37

```

```

38 </social_networks/8/roles/40> a sn:role;
39   sn:inRelation </social_networks/8/nodes/57>;
40   sn:name "Profesor Guia" .
41
42 </social_networks/8/roles/41> a sn:role;
43   sn:inRelation </social_networks/8/nodes/57>;
44   sn:name "Estudiante" .
45
46 </social_networks/8/nodes/57> a sn:relation;
47   sn:belongsToFamily </social_networks/8/families/21>;
48   sn:kind "Relation";
49   sn:name "Social Graph Editor" .

```

Listing 5.1: Exportación RDF red social figura 5.19

Esta representación RDF/N3 está validada [23]. Es importante mencionar que para dicha representación RDF se define una ontología propia de la aplicación, que cuenta con las definiciones comunes de la aplicación, como por ejemplo Actores, Relaciones y Roles, pero además cuenta con las definiciones de elementos propios de la red como Atributos definidos en esta. El vocabulario correspondiente de la red sería el siguiente:.

```

1  @prefix   sn: <http://sn.dcc.uchile.cl/social_networks/8/vocabulary#> .
2  @prefix   rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3  @prefix   rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
4
5  # Classes
6  sn:socialNetwork a rdfs:Class .
7  sn:family a rdfs:Class .
8  sn:node a rdfs:Class .
9  sn:role a rdfs:Class .
10 sn:actor a rdfs:Class ;
11   rdfs:subClassOf sn:node .
12 sn:relation a rdfs:Class ;
13   rdfs:subClassOf sn:node .
14
15 # Properties
16 sn:name a rdf:Property ;
17   rdfs:range rdfs:Literal .
18
19 sn:belongsToFamily a rdf:Property;
20   rdfs:domain sn:node;
21   rdfs:range sn:family .
22
23 sn:participatesAs a rdf:Property ;
24   rdfs:domain sn:actor ;
25   rdfs:range sn:role .
26
27 sn:inRelation a rdf:Property ;
28   rdfs:domain sn:role ;
29   rdfs:range sn:relation .
30
31 sn:kind a rdfs:Property .
32
33 sn:positionX a rdfs:Property ;
34   rdfs:domain sn:node ;
35   rdfs:range rdfs:Literal .
36 sn:positionY a rdfs:Property ;
37   rdfs:domain sn:node ;
38   rdfs:range rdfs:Literal .
39
40 sn:color a rdfs:Property ;
41   rdfs:domain sn:family ;
42   rdfs:range rdfs:Literal .
43
44 sn:attributeEdad a rdfs:Property ;
45   rdfs:domain sn:node ;

```

Listing 5.2: Vocabulario red social figura 5.19

Además en caso de ser necesario, como se menciona antes, se puede exportar agregando la información gráfica de la red social, en donde se incluyen las posiciones de los nodos en el canvas, los colores de las familias definidas, etc.

5.5. Importación en RDF

Junto con la exportación en formato RDF que usa la aplicación, esta tiene la funcionalidad de importar a partir de un archivo RDF de la red social como el definido en el ejemplo de código 5.4, para crear una red social a partir de lo importado.

Import Social Network From File

RDF/N3 File

Upload a valid RDF/N3 social network file previously exported with this application.

[Cancel](#)

Figura 5.20: *Importación de una Red Social*. Para archivos RDF/N3 exportados previamente con la aplicación.

Cabe destacar que el contenido N3 de la red social puede no definir las posiciones de los nodos, en caso de un archivo que fue exportado sólo con la estructura de la red, no la información visual y por lo tanto la aplicación asigna las posiciones en base al algoritmo de layout de grafos de Fruchterman-Reingold [30], con lo cual crea los nodos, las familias, roles e interacciones entre ellos según lo especifica el archivo usado.

5.6. Unión de Redes Sociales

Una de las funcionalidades claves de esta aplicación, que es una mejora con respecto a la experiencia de Manuel Bahamonde [27], es la unión de redes sociales, esta consiste en que los datos generados de forma manual con la aplicación pueden ser complementados con datos generados por otros usuarios de la misma.

Para unir redes sociales, se utiliza el botón de creación de redes sociales y se elige la opción de unir dos redes sociales, en donde aparece el formulario de unión de redes sociales 5.21, en donde inicialmente seleccionamos un archivo RDF/N3 para importar la red, luego seleccionamos a que red se va a unir esta red social importada como indica el formulario de la figura.

Join Two Social Networks

Step 1. Select Social Networks

Select one of your Social Networks

Original Social Network

Import an external Social Network to Join

External RDF/N3 File

Upload a valid RDF/N3 social network file previously exported with this application.

[Next](#)

[Cancel](#)

Figura 5.21: *Selección de Redes a Unir*. Se selecciona una red social que el usuario posee y se importa una red social externa con la que se va a unir.

Luego de presionar *Next* en el formulario de unión, lo siguiente es definir las equivalencias dentro de las familias de ambas redes sociales 5.22, en donde a cada familia de la red original, selecciono una, más de una o ninguna equivalencia con una familia de la red social importada. Una vez estas equivalencias son definidas, se procesan en el back-end para añadir todos los datos de la red social importada en la seleccionada.

Join Two Social Networks

Step 2. Define Family Equivalences

Define which families of the imported social network are equivalent to the original social network families in order to produce a better union of the social networks

Imported Family	Original Family Equivalence
A alumnoDCC	<input type="text" value="(none)"/>
A profesorDCC	<input type="text" value="(none)"/>
A profesorGuía	<input type="text" value="(none)"/>
R amigoDe	<input type="text" value="(none)"/>
R memoria	<input type="text" value="(none)"/>

[Finish Join!](#)

[Cancel](#)

Figura 5.22: *Selección de Equivalencias entre Familias*. En esta pantalla se seleccionan que familias de la red importada para unir, son equivalentes a las familias de la red social original, en múltiples equivalencias a la misma familia original son permitidas.

Con respecto a las relaciones, cabe mencionar que si en una red social está la relación A con familia F_1 y en la otra red social se encuentra la relación B con familia F_2 , al momento de determinar que $F_1 = F_2$, la relación B es eliminada y todos sus roles pasan a la relación A debido a que ambas relaciones se consideran equivalentes

Luego de completar las operaciones anteriores, las redes sociales estarán unidas dentro de la original, donde los nodos fueron reordenados. Lo siguiente es definir las equivalencias entre nodos, para esto se selecciona el modo de unión en el menú de edición 5.3.1.1, con esto, solo basta arrastrar un nodo sobre otro y confirmar el dialogo que aparece al soltar el nodo para unirlos (figura 5.23). Al unir un par nodos se combinan las familias, los atributos y roles de estos, comportamiento el cual se decidió de esta forma debido a que un nodo puede no tener un nombre y su contexto (familias, atributos y nodos) expresan mucho mejor la entidad que representa el nodo y de esta forma visual es más sencillo encontrar la equivalencia entre estos.

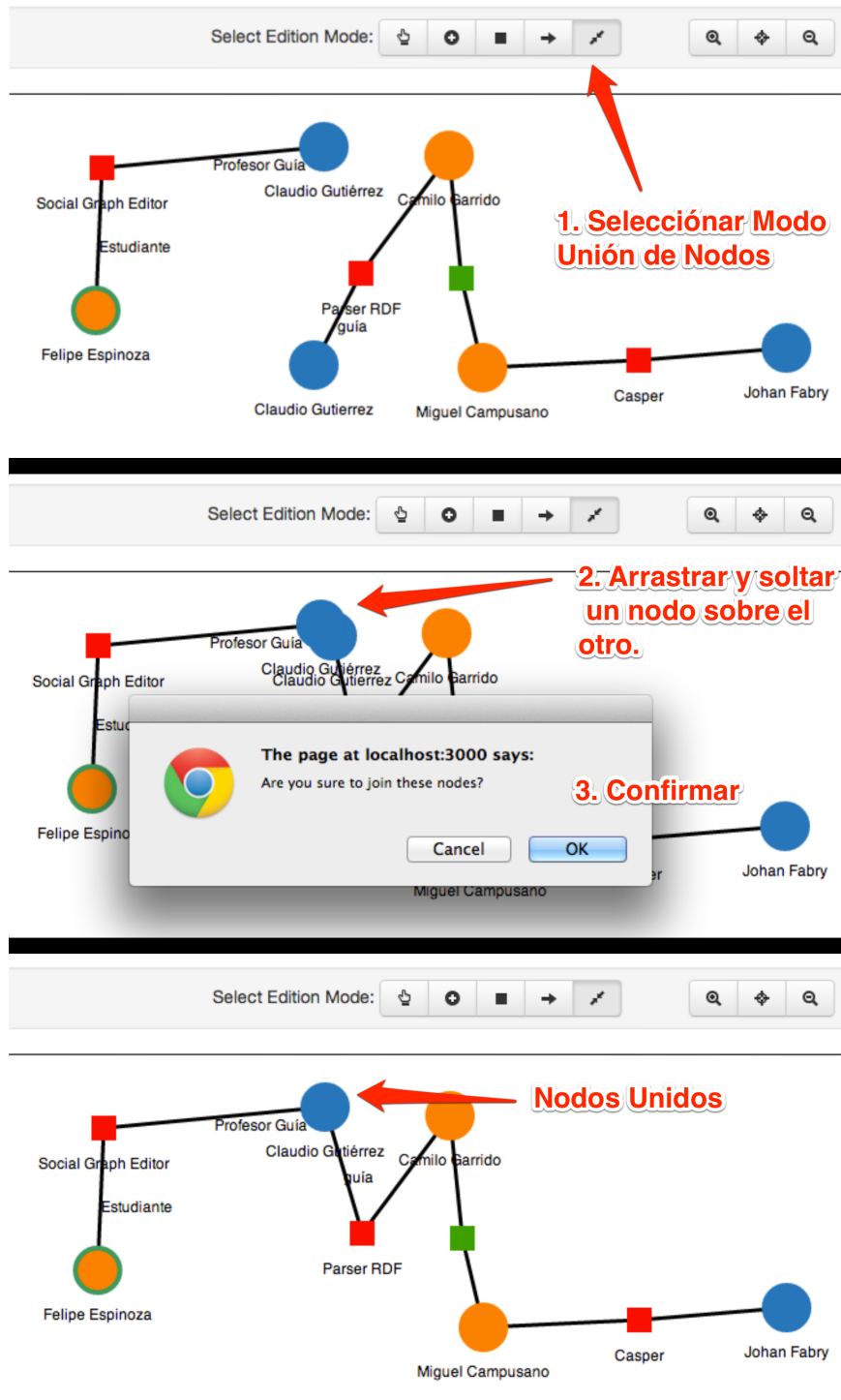


Figura 5.23: *Operación de Unión de Nodos*. Explicación de los 3 pasos para unir dos nodos entre sí (actores o relaciones): seleccionar el modo de unión de nodos, tomar un nodo y arrastrarlo al que se desea unir y confirmar la operación. Con estos pasos se unen los nodos.

5.7. Ejemplos de Uso

Capítulo 6

Conclusión

6.1. Evaluación de la Solución

En este trabajo de memoria, aprovechando la experiencia de la memoria de Manuel Bahamonde [27], se ha realizado una aplicación que permite a usuarios básicos la creación y manejo de redes sociales multi propósito y de esta forma ser una herramienta para asistir en estas tareas a investigadores de diversos campos que incluyan el estudio de redes sociales.

Esta aplicación se compone de un servicio de almacenamiento de redes sociales que presenta una API en formato JSON y RDF para redes sociales, que puede ser usado con otro tipo de consumidores de esta API.

Por otro lado una aplicación web que corre en el lado del cliente la cual cuenta con una interfaz amigable de ingreso y manipulación de datos, en donde se cumplen los objetivos de crear relaciones de aridad múltiple, agregar atributos complementarios a los nodos.

Además de lo anterior, los usuarios son capaces de importar o exportar en formato RDF/N3 las redes sociales que crean con la herramienta e incluso unir la información que ellos generan, funcionalidad que abre un nuevo nivel de interacción entre usuarios con respecto a la creación de diversos tipos de redes sociales.

Se lograron visualizaciones limpias de los datos, que pueden ser manipuladas en formatos estándares y que permiten una visión sin pérdida con distintos niveles de acercamiento o alejamiento.

Se representaron algunas redes sociales con la aplicación a modo de ejemplificar su utilidad, se documentó toda la experiencia en este informe que se entrega junto con el software,

que posee código abierto, con las posibilidades de seguir explotando y expandiendo esta herramienta a futuro.

6.2. Dificultades Técnicas

Dentro del transcurso del desarrollo de esta memoria, se encontraron algunas dificultades a nivel de implementación, razón por la cual estas serán discutidas brevemente en esta sección.

6.2.1. Modelamiento de Redes Sociales

En este sentido, dentro del prototipado que requirió el abordaje del desarrollo, inicialmente se pasó por alto el modelo de Mauro San Martín [29], creando un modelo básico de redes sociales a medida que las necesidades se iban presentando en términos de desarrollo. Esto hizo que el mismo fuera más dificultoso, lo cual forzó a estudiar de mejor manera el modelo de Mauro, que mejoró tanto la estructuración de la información, además del esfuerzo requerido para desarrollar la aplicación, ahorrando dificultades por ejemplo: de tener que manejar código para actores y relaciones separadamente en vez de considerarlos un ente común llamado nodo, entre otras cosas.

A continuación se presenta un listado con los puntos más relevantes con las dificultades en el modelamiento y como el modelo de Mauro ayuda con esto.

1. Considerar Actores y Relaciones como Nodos, ayuda a factorizar mucha implementación debido a que su comportamiento es casi el mismo.
2. La manera de como se expresan los Roles en el modelo de Mauro permite sin mayor esfuerzo implementar una relación de M a N dentro de un mismo modelo (Nodo), sin mayores inconvenientes e incluso provee la convención de que un Rol posee un actor y una relación, más fácilmente implementable que especificar que sólo tuviera 2 nodos.
3. El modelamiento de atributos está pensado en tener diversa clase de atributos en cantidades variables para un nodo, es decir, se puede simular la funcionalidad de una base de datos sin esquema como MongoDB con una base de datos que usa esquema como MySQL.

6.2.2. Interfaz de Alta Interacción en Desarrollo Web

Técnicamente, si se deseaba hacer una aplicación con una alta interactividad en términos de edición de grafos y que a su vez, esta tuviera las propiedades que entrega el hecho de que sea una aplicación web, el lenguaje de programación único para completar esta tarea es JavaScript, por lo tanto, en el camino, se tuvo que adoptar un enfoque distinto al desarrollo web tradicional (*server side*) y optar por el desarrollo *client side*, debido a que en este se pueden acceder limpiamente todos los atributos provenientes de la base de datos, tratarlos

como objetos y asignarle lógica de modelos, implementar lógicas de vista mucho más complejas que lo que se puede lograr con JavaScript plano, junto con tener mejor integración con SVG, que era necesario también para este proyecto.

Al igual que en el ítem anterior, se presenta un listado con los puntos más importantes a considerar sobre las dificultades en la elección de herramientas en el desarrollo web:

1. Una aplicación que tenga mucha interacción de interfaz, con mucha lógica en estas interacciones es mucho más fácil implementarla en un framework de desarrollo del lado del cliente como EmberJS, de acuerdo a 4 demos iniciales que usaron BackboneJS, EmberJS, Ruby on Rails con enfoques distintos para comunicar los datos del modelo al Javascript.
2. Un problema de EmberJS es que para el tiempo de desarrollo, otoño 2013, ember-data, el proyecto que comunica el *back-end* de la aplicación con su *front-end* en EmberJS, es bastante inestable, sin embargo, posee una comunidad activa que actualiza las versiones de este proyecto rápidamente.
3. Para representar los elementos manipulables en el Canvas de Redes Sociales, la mejor opción fue usar D3.js para generar los gráficos vectoriales, que son tags dentro del documento HTML, por lo cual, es más simple asignarles eventos para poder interactuar con esos elementos. Además D3.js puede obtener y usar los datos de la misma forma que EmberJS los provee, por lo tanto la integración es más fácil.

6.3. Trabajo Futuro

A partir de lo realizado, los siguientes pasos a futuro pueden enriquecer este proyecto:

1. **Agregar Endpoint SPARQL:** uno de los aspectos técnicos a futuro, sería la instalación de un endpoint virtuoso, u otro para aprovechar de mejor manera la utilización del formato RDF, de esta forma, reemplazar el almacenamiento relacional de los datos por uno de grafos y lograr una mayor integración con la web semántica.
2. **Mejorar Escalabilidad Aplicación:** si el proyecto llega a un nivel popularidad grande, se puede mejorar la escalabilidad de manera tal de que la aplicación soporte redes sociales de mayor tamaño, sin perder performance de la misma.
3. **Agregar Funcionalidad de Clases a Familias:** esto se refiere en que se puede agregar la capacidad de agregar atributos a familias de actores y relaciones, de manera tal, que al crear un actor o relación de determinada familia con atributos, automáticamente, el nodo creado tenga estos atributos de acuerdo a la plantilla que presenta la familia.
4. **Habilitar Modo Offline Aplicación:** es posible, debido a la utilización de frameworks client side, hacer que la aplicación de edición de grafos me permita un modo offline, reemplazando el almacenamiento centralizado por uno local en el navegador, sincronizando los datos si es pertinente posteriormente.
5. **Temporalidad en Redes Sociales:** se puede extender la aplicación de manera tal de que se pueda agregar temporalidad a las redes sociales, que sirva para analizar los cambios en las estructuras sociales con el paso del tiempo, aspecto que puede ser

prometedor para la utilidad de esta herramienta en el estudio de ciertas disciplinas relacionadas con redes sociales.

6. **Aspectos de Privacidad:** la aplicación se puede mejorar en términos de privacidad de los datos, explicitando que los datos agregados a la aplicación son de propiedad exclusiva de sus usuarios, que no habrá ningún tipo de mal uso de la información, además de agregar opciones para diferenciar redes privadas y públicas con sus restricciones de permisos pertinentes.
7. **Aspectos Legales:** se debe resolver además temas legales con respecto a la aplicación y la publicación de datos privados de personas, en redes sociales de tipo público, que pudieran legalmente afectar de una u otra forma a terceros por la exposición de estos datos.
8. **Mejorar Interacción entre Usuarios de la Aplicación:** actualmente la forma en que 2 usuarios puede compartir redes sociales entre ellos es por medio de archivos RDF/N3 exportados desde la aplicación, sin embargo, esta experiencia puede ser mejorada haciendo una aplicación que muestre una lista de redes públicas listas para importar desde la cuenta del usuario.

6.4. Distribución del Software y Documentación

Todo el software, ejemplos y documentación desarrollados en este proyecto de memoria, se pueden encontrar y descargar libremente desde el sitio:

`http://fespinoza.github.io/social-graph-editor/`

En dicho sitio, existen instrucciones precisas de como crear un ambiente de desarrollo para la aplicación, el cual está automatizado por la herramienta de gestión de máquinas virtuales Vagrant [22], es decir, que con unos pocos comandos se puede instalar una máquina virtual que usa recetas Chef [5] para instalar todas las aplicaciones necesarias para definir el ambiente de desarrollo las cuales están incluidas en este proyecto.

Además por el lado de la puesta en producción del proyecto en un servidor externo, la alternativa más fácil y rápida es Heroku, cuyos pasos de instalación se encuentran en la guía de su sitio [11]. En caso de querer hacer un deploy en un servidor propio, se recomienda Ubuntu 12.04 con postgresql y hacer un deployment con capistrano como una aplicación rails estándar.

Bibliografía

- [1] Allegrograph, motor de base de datos para grafos. <http://www.franz.com/agraph/allegrograph/>, June 2013.
- [2] Angularjs home page. <http://angularjs.org/>, June 2013.
- [3] Backbonejs home page. <http://backbonejs.org/>, June 2013.
- [4] Cakephp home page. <http://cakephp.org/>, June 2013.
- [5] Chef, herramienta de automatización de instalación de programas. <http://www.opscode.com/chef/>, July 2013.
- [6] Curso html5, w3schools. http://w3schools.com/html/html5_intro.asp, June 2013.
- [7] D3.js, data driven documents. <http://d3js.org/>, June 2013.
- [8] Django home page. <https://www.djangoproject.com/>, June 2013.
- [9] Graph herramienta que construye, visualiza y modifica grafos. <http://www.cosbi.eu/index.php/research/prototypes/graph>, June 2013.
- [10] Gruff un browser basado en la base de datos allegrograph. <http://www.franz.com/agraph/gruff/>, June 2013.
- [11] Guía de inicio en la utilización de heroku para el deployment de aplicaciones en ruby on rails. <https://devcenter.heroku.com/articles/quickstart>, July 2013.
- [12] Littlesis, una base de datos libre de relaciones de 'quien conoce a quien' de los negocios y el gobierno estadounidense. <http://littlesis.org/>, June 2013.
- [13] Meteorjs home page. <http://meteor.com/>, June 2013.
- [14] Omnigraffle, aplicación de diagramación para mac. <http://www.omnigroup.com/products/omnigraffle/>, June 2013.
- [15] Pajek, programa para análisis de grandes redes. <http://pajek.imfm.si/doku.php?id=pajek>, June 2013.
- [16] Poderopedia, quién es quién en la política y los negocios en chile. <http://poderopedia.>

org/, June 2013.

- [17] Pylons home page. <http://www.pylonsproject.org/>, June 2013.
- [18] Rails guides, the mvc architecture. http://guides.rubyonrails.org/getting_started.html#the-mvc-architecture, June 2013.
- [19] Ruby on rails, home page. <http://rubyonrails.org/>, June 2013.
- [20] Sinatra home page. <http://www.sinatrarb.com/>, June 2013.
- [21] Spring home page. <http://www.springsource.org/>, June 2013.
- [22] Vagrant, herramienta que gestiona la creación y uso de máquinas virtuales. <http://www.vagrantup.com/>, July 2013.
- [23] Validador de rdf/xml y rdf/n3. <http://www.rdfabout.com/demo/validator/>, July 2013.
- [24] Vocabulario en formato rdf de poderopedia. <https://github.com/poderopedia/PoderVocabulary>, June 2013.
- [25] W3c, home page. <http://www.w3.org/>, June 2013.
- [26] yed, graph editor. http://www.yworks.com/en/products_yed_about.html, June 2013.
- [27] Manuel Bahamonde. Aplicación para la creación y administración de redes sociales, 2009. Memoria para optar al título de ingeniero civil en computación.
- [28] Yehuda Katz. Emberjs home page. <http://emberjs.com/>, June 2013.
- [29] Mauro San Martín. A model for social networks data management, 2012. Tesis para optar al grado de doctor en ciencias mención computación.
- [30] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, first edition, structural analysis in the social sciences edition, 1994.