Instituto de Computación

Facultad de Ingeniería

Universidad de la República

Diego Bonilla
Orientador: Franco Simini

Montevideo, 1 de diciembre de 2014

Resumen

El trasplante de órganos ha evolucionado científicamente, ha vencido las barreras inmunológicas y ha entendido las compatibilidades entre organismos. El éxito de los trasplantes junto a esperanzas de vida crecientes ha hecho aumentar la demanda de órganos. Es el momento de apuntar a mejorar la eficiencia de la asignación de órganos para contribuir con el objetivo de intentar salvar más vidas.

En este trabajo se propuso y se construyó un sistema de software que permite, a partir de un órgano donado, determinar el receptor más adecuado a quien asignarlo: encontrar el mejor receptor. La definición de mejor receptor, se hace en tiempo de ejecución por parte de usuarios administradores del sistema y puede cambiarse según las necesidades de la realidad del trasplante de órganos.

En primer lugar, se definió el criterio Donamatch para el matching de órgano y receptor con el fin de descubrir todas las funciones y variables presentes en la asignación y que determina cómo calcular el matching. Luego se buscó un compromiso entre la abstracción del criterio y la buena experiencia de usuario que necesitaba brindar el sistema principalmente para permitir utilizar eficientemente el tiempo. Del que se desprendió la definición de los conceptos *Entidades*, *Atributos*, *Unidades de Medida, Funciones, Metafunciones, Operadores, Asociación de Atributos, Políticas de ranking y Matchings*.

El sistema presenta una interfaz de usuario de estilo *Single Page Application* donde *Knockout* y *Bootstrap* dan soporte a la experiencia de usuario. La arquitectura propuesta en capas combina *Model View Controller*, *Model View ViewModel* y *Unit of Work*.

Donamatch se publicó en el proveedor de *Cloud Computing Azure* para validar la usabilidad mediante encuestas sobre el uso de la aplicación de las que participaron 51 usuarios.

Palabras claves: donamatch, matching, trasplante de órganos, criterio DM.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

INTRODUCCIÓN

Este trabajo propone, diseña y construye un sistema de software de matching que busca, por su motivación inicial inmersa en el contexto del trasplante de órganos contribuir con la asignación eficiente de órganos.

El presente informe contiene los aspectos más relevantes relacionados al estudio, investigación, desarrollo del software y resultados obtenidos.

En este capítulo se encuentran las motivaciones del proyecto, los objetivos principales planteados, así como una descripción de la organización del documento y algunas consideraciones.

1.1 Motivaciones

Han pasado unos sesenta años desde el primer trasplante exitoso de órganos y hoy día el trasplante es un hecho. La ciencia ha evolucionado suficientemente como para que se trasplanten órganos todos los días, en casi todo el mundo, exitosamente. Con mayor o menor tiempo de supervivencia del órgano trasplantado pero, en general, permitiendo mejorar la calidad de vida de las personas.

En cuanto a edades hay un parecido con la Ingeniería del Software, ambas ciencias son relativamente jóvenes pero han alcanzado cierta madurez como para plantearse nuevos retos.

El momento indicado

Ya se vencieron las barreras inmunológicas, ya se entendieron las compatibilidades entre organismos y ya se sutura eficientemente. Es el momento de dar un paso más. Mejorar la eficiencia de la asignación de órganos contribuirá con el intento de salvar más vidas.

El correcto órgano para el receptor correcto

Con el éxito de los trasplantes de órganos y un número creciente de personas con enfermedad terminales de órganos, la demanda de trasplantes ha crecido constantemente. La gente vive más. Desde el primer trasplante de riñón, solo ha crecido la esperanza de vida de las personas en el mundo; ha pasado de ser de cincuenta años a setenta en el 2012, como se ve en la Figura 1.1. En Uruguay, la esperanza de vida en el año 2012 fue de 77 años. Si las

personas viven más, por un lado hay menos órganos para donar y por otro se necesitan más órganos pues las personas que antes no demandaban órganos (no llegaban a una vejez avanzada), ahora si lo hacen.

Este aumento ha dado lugar a una mayor necesidad de utilizar órganos de la mayor cantidad posible de donantes, esto es: aumentar la cantidad de órganos disponibles.

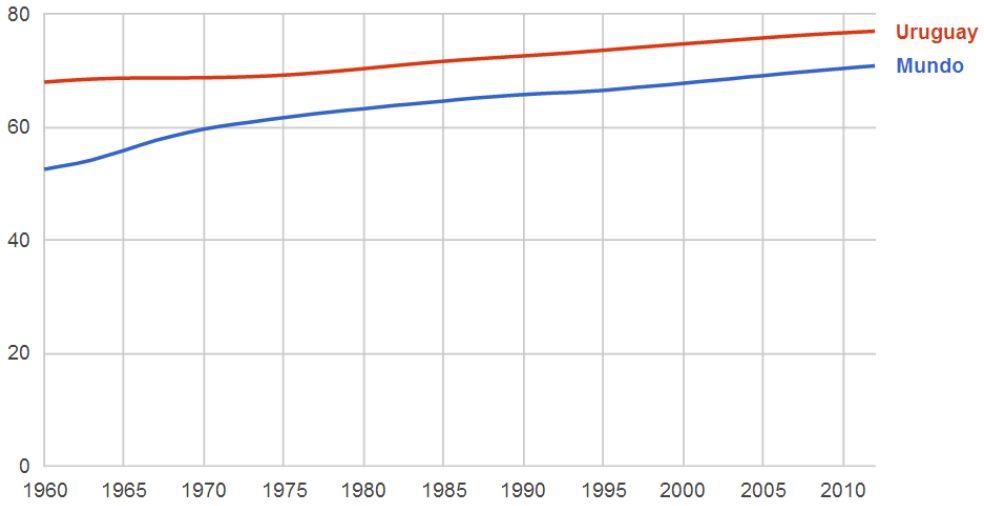


Figura 1.1 - *Esperanza de vida en años Uruguay vs Mundo Desde 1960 a 2010. Datos de Banco Mundial [1].*

Como los criterios de selección se han vuelto menos estrictos para dar abasto a la demanda, los resultados del trasplante son más fuertemente influenciados por factores del receptor y del donante por lo que encontrar el órgano adecuado para el destinatario correcto es más importante que nunca. En el *Ninth Annual American Society of Transplant Surgeons* [2], se trató la problemática relacionada a esta realidad. ¿Se debe asignar según características de riesgo del receptor?, ¿usar órganos con riesgo de trasmisión de enfermedades?, ¿usar órganos de donantes tras muerte cardíaca?

En el año 2010, la Organización Mundial de la Salud (OMS), estableció, en uno de los *Principios rectores de la OMS sobre trasplante de células, tejidos y órganos humanos*, que "la asignación de órganos, células y tejidos deberá regirse por criterios clínicos y normas éticas, y no atendiendo a consideraciones económicas o de otra índole. Las reglas de asignación, definidas por comités debidamente constituidos, deberán ser equitativas, justificadas externamente y transparentes" [3].

Proveer un sistema que permita encontrar el mejor receptor para un órgano determinado teniendo en cuenta los aspectos sociales correspondientes y que pueda adaptarse y cambiar la definición de mejor receptor según las

necesidades del estado del arte del trasplante parece ser un buen aporte para el trasplante de órganos.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar un sistema de software que permita asignar eficientemente órganos donados a los mejores receptores, según criterios definidos previamente.

Alineados a este objetivo, se plantean otros.

- Permitir que los criterios puedan cambiar dinámicamente ya sea como respuesta al advenimiento de mejores formas de realizar la asignación o como ajustes al contexto en el que se llevan a cabo los trasplantes.
- Usar tecnologías que permitan proveer una buena experiencia de usuario promoviendo su uso ágil y sencillo.
- Trasparentar el proceso del matching para contribuir con los aspectos éticos relacionados al trasplante de órganos.
- Generar información que enriquezca la Historia Clínica Electrónica (HCE) del paciente receptor.

1.3 Organización del documento

En el Capítulo 2, se presenta un resumen de la historia de los trasplantes llegando hasta la situación actual y una breve exposición sobre algunos aspectos éticos y legislativos de los trasplantes.

El Capítulo 3 contiene los requerimientos funcionales y no funcionales del sistema.

El Capítulo 4 presenta la solución planteada incluyendo el álgebra del trasplante.

El Capítulo 5 aborda la construcción del software como una implementación de la solución propuesta.

En el Capítulo 6 se presentan los resultados obtenidos, incluyendo la información obtenida en encuestas a usuarios.

En el Capítulo 7 se describe la Gestión del Proyecto.

Y en el Capítulo 8 se muestran las conclusiones del trabajo realizado, las contribuciones y el trabajo a futuro.

1.4 Consideraciones

A lo largo del documento se usará la palabra *matching*, principalmente por su popularidad en ciencias de la computación, para referirse a la coincidencia, correspondencia, concordancia y compatibilidad, que en el contexto de este trabajo son sinónimos. Hacer el matching de un órgano con un receptor, es encontrar el receptor que por sus características es el indicado para recibir el órgano. Esto no se trata de menospreciar el idioma español, sino de facilitar el entendimiento del lector.

Se entenderá por asignación al proceso mediante el cual, en base a criterios médicos, culturales y legales se selecciona a los pacientes receptores de órganos y tejidos.

Se entenderá por centro de trasplantes de órganos, o centro de trasplantes a aquella organización o institución que se encargue de la asignación de órganos de una comunidad, ciudad, país o región.

El nombre Donamatch surge de la unión de las palabras dona en homenaje a las personas que ceden su sangre, órganos y tejidos, y *match* del inglés como símbolo de la búsqueda del órgano adecuado para el receptor adecuado.

Todo el código se escribió en inglés (aunque los comentarios en español), favoreciendo el futuro de Donamatch, ya que actualmente es el idioma más promovido en programación. Además del beneficio de poder escribir nombres nemotécnicos sin faltas de ortografía por no contar con tildes, cedillas, etc.

Capítulo 2

EL TRASPLANTE DE ÓRGANOS

2.1 Mirar al pasado

El trasplante de órganos no tuvo un origen único o absoluto. No surge instantáneamente de la nada cual si fuera la teoría de la generación espontánea de Aristóteles. Sin embargo, hay momentos claves, hitos importantes en la historia, y quizá muchos olvidados en el camino por plumas de cobardes, que en una especie de sinergia hicieron que hoy día un ser humano pueda verse beneficiado al punto de poder vivir gracias a un órgano proveniente de otro. Para intentar entender el presente del trasplante de órganos es necesario mirar al pasado.

Entre mitos y relatos

La literatura mítica de muchas culturas ilustra el trasplante como un símbolo de renovación y súper-poder. Para enaltecer los poderes de dioses pero también como cura de enfermedades de los simples mortales.

El relato más creíble, de los bien lejanos, es el que cuenta que el indio Sushruta [4], en el siglo I o II a.C. habría practicado trasplantes de tejidos en seres humanos: con la piel de uno, le recompuso la nariz a otro. Luego hay unos cuantos que más bien parecen mitos por la realidad científica que se cree existía en la época, destaquemos algunos. Entre los siglos 2 a.C. y 2 d.C. habrían ocurrido en China, trasplantes de órganos incluyendo el corazón [5]. En el siglo I a.C. los conocidos como santos patrones de la medicina Cosme y Damián, habrían reemplazado la pierna de un gladiador etíope muerto en la arena por la gangrenada de un sacristán [6].

Años intrascendentes

Pasó muchísimo tiempo hasta que tuviera su inicio lo que se conoce a menudo como la era moderna del trasplante recién a comienzos del s. XX.

En el s. XVI Gasparo Tagliacozzi que reconstruía narices y orejas a sus pacientes a partir de la piel de sus brazos, reconoció que si la piel utilizada provenía de un individuo diferente a quien se le injertaba, en general el procedimiento fallaba; lo que observaba, en realidad, era la respuesta inmune del cuerpo receptor.

Allá por el s. XVIII, el escocés John Hunter [7 - 8], reemplazó con éxito un premolar de un hombre [9]. Hunter, conocido como el padre de la cirugía moderna [10], creía que todas las sustancias vivientes podían unirse a otra al ponerse en contacto firme con ésta. Siguiendo esta línea de pensamiento, en el s. XIX, Brown-Sequard sugiere que los miembros amputados se vuelvan a coser al cuerpo [11]. Curiosamente estos hombres están relacionados al descubrimiento de la Testosterona, Hunter realizó experimentos de trasplante testicular en 1767 mientras estudiaba técnicas de trasplante de tejidos y casi un siglo después, en 1889, Brown-Sequard anunció que su auto-inyección de extractos testiculares resultó en capacidades físicas y mentales rejuvenecidas [12], hecho que abrió camino para el descubrimiento.

Los primeros intentos

Uno de los primeros intentos exitosos de trasplantar una parte del cuerpo de un humano a otro del que se tiene registro es el que se da en diciembre de 1905. El austríaco y amante del violín Eduard Zim trasplantó la córnea de un niño de once años que no sobrevivió a un accidente que le dejó trozos de metal en los ojos a un trabajador en República Checa. Y si bien algunas complicaciones afectaron un ojo, el otro le permitió volver al trabajo. La intención era usar los ojos pero la enucleación, o extracción del ojo que deja los músculos y contenidos orbitales del ojo intactos, no funcionó y por ende utilizó las córneas [13]. Le siguieron 50 años no de los más exitosos, hubo muchos intentos fallidos, incluso de trasplantar órganos de animales en humanos.

Hasta entonces mucha sangre había sido derramada con intentos clandestinos y científicamente poco justificables. Que más tenían que ver con magia que con ciencia. El siglo XX es de los más fructíferos para el trasplante. Algo cambió.

2.2 La era moderna del trasplante

En el siglo XX, el enfoque fue diferente. Aquello que había empezado como algo principalmente experimental hasta el siglo XIX, de a poco empezó a tomar forma de ciencia. Se empezó a estudiar los aspectos teóricos biológicos del trasplante y empezó a entenderse la incidencia del sistema inmunológico, la última gran barrera clínica. Con una brillante excepción.

Alexis Carrel, quien desarrollaba trabajos experimentales quirúrgicos con cadáveres y perros, quizá motivado por la forma en que muere el presidente francés Sadi Carnot cuando visita Lyon (los cirujanos no pudieron suturar la vena porta que le había sido afectada por un cuchillazo), empezó a estudiar la anastomosis vascular y técnicas de *suturación*. Diez años después recibió el

premio nobel por el desarrollo de técnicas de *suturación* que hasta hoy día se practican en los trasplantas con pequeñas modificaciones [14].

En 1900, Landsteiner y Miller fueron los primeros en reconocer que los seres humanos podrían ser agrupados de acuerdo a la presencia de aglutininas en sus sueros fisiológicos. Además sentaron las bases para las pruebas de histocompatibilidad [15]. Los primeros intentos de transfusión de sangre entre animales y seres humanos antecedieron este hecho en unos 150 años; los resultados fueron tan devastadores que la práctica se vio interrumpida hasta entonces.

El rechazo

Hasta el siglo XIX el trasplante se limitó a los procedimientos de autoinjerto, al comienzo de este siglo, los xenoinjertos fueron objeto de experimentación aunque sin éxito. En las últimas décadas se ha puesto sobre el tapete nuevamente los xenotrasplantes como medida para aumentar el número de donantes, tema que se tratará más adelante. El problema del rechazo comenzó a hacerse realidad clínica en el siglo XX, teniendo al aloinjerto como modelo para investigación. Ante el Congreso Médico Alemán, Erick Lexer, en 1911, reportó que en general los aloinjertos no duraban más de 3 semanas [11]. Luego, en 1924, Holman reconoció que el injerto de piel era rechazado más rápido en la segunda aplicación.

En un intento de buscar la inmunidad a los tumores, George Snell, en la década del 30, descubrió el locus dominante de histocompatibilidad en el ratón, llamado luego H-2 [16]. Un paso importante, ya que este locus es análogo a los antígenos leucocitarios humanos (HLA), sistema utilizado para el *matching* de tejidos. A quien se le atribuye haber ideado el concepto *self vs nonself*, es a Peter Gorer, después de darse cuenta de que los antígenos en las células de los tejidos están determinadas genéticamente y son capaces de provocar la destrucción del injerto extranjero. Fue quien además en 1937 identificó el primer antígeno de histocompatibilidad en los seres humanos [17-18].

El sistema inmunológico

Los primeros estudios de los mecanismos inmunes se desarrollaron por la necesidad de entender las enfermedades que desolaron a la humanidad. Sin embargo el trasplante de órganos y tejidos dio un impulso importante a los esfuerzos científicos que revolucionaron el campo de la inmunología y la vida. No obstante la biología del rechazo del injerto sigue siendo solo parcialmente entendida.

A los tres años de que Carrel ganara el Nobel por sus técnicas que permiten unir elementos anatómicos, nace en Río de Janeiro Peter Medawar, quien en 1960 también ganaría el premio Nobel por sus trabajos sobre cómo el sistema

inmunológico acepta o rechaza trasplantes de tejidos y el descubrimiento de la tolerancia inmunológica adquirida [19]. Medawar, investigó injertos de piel en conejos diferenciando respuestas inmunológicas entre homoinjertos y autoinjertos [20]. Fue el primero en razonar respecto de la paradoja de que el feto no desencadene las respuestas inmunitarias de la madre, tolerancia que solo se da durante la gestación, ya que luego si se intenta trasplantar un tejido del bebe a la madre, es inmediatamente rechazado. Entendió que si se logra imitar el comportamiento de la etapa de gestación el órgano extraño no será rechazado. Medawar, junto al cirujano plástico Gibson observó que si un injerto de piel se colocaba desde un animal a otro, tenía una supervivencia de unos 7 días, si después se aplicaba otro injerto de piel, era rechazado en la mitad del tiempo; confirmando así las conclusiones de Holman. También observó que los injertos de piel de un miembro de la familia, se toleran mejor que los de un donante no familiar y que los injertos de piel de un gemelo se toleran mejor que los de otros miembros de la familia.

La primera evidencia de los grupos sanguíneos en humanos la brindó Dausset en 1952; reconoció que los individuos que habían recibido múltiples transfusiones de sangre, contenían en sus sueros *leucoaglutininas*, mientras que aquellos que habían recibido pocas o ninguna tenían *isoaglutininas* [21]. Esta distinción abrió el camino para el descubrimiento de los HLA en humanos ya que permitió saber que los *aloanticuerpos*, así como *autoanticuerpos* de hecho existen. Ya en el 64, Teraski et al [22], desarrollaron métodos de ensayo de presencia de anticuerpos citotóxicos introduciendo el test de *microlinfocitotoxicidad* [23].

Riñón

En 1936, Yu Yu Voronoy realizó el primer trasplante de riñón humano, utilizando un órgano de un donante fallecido. El receptor murió poco tiempo después, como consecuencia del rechazo.

Luego de un accidente en 1952, una mujer fallece y un riñón suyo le es trasplantado a su hijo que sobrevivió sin rechazar el órgano durante veintidós días. Dos años después el mismo equipo médico compuesto por Murray, Merril, Harrison y Hume, realizan el primer trasplante de riñón exitoso, de Ronald a su hermano gemelo (monocigóticos) Richard Herrick, que vivió ocho años después de la operación [24]. Es un momento importante: se supo que el trasplante funcionaba… Aunque con hermanos gemelos. Además se confirma que hay una base genética para la compatibilidad donante-receptor. El trasplante de órganos era un hecho, pero entre personas que comparten el mismo ADN. El próximo desafío era saber cómo hacer para que el cuerpo del receptor acepte el órgano del donante sin que éstos tengan parentesco.

Inmunosupresión

John Loutit en la década del cincuenta, experimentó con irradiación corporal total (TBI) en roedores sometidos a injertos de piel [25], lo que se consideró una innovación en la inmunosupresión inducida. En 1958 el TBI se aplicó a humanos, aunque sin éxito, con complicaciones letales por Murray en Boston [26] y Hamburger en Paris [27]. Schwarz y Damechek revolucionaron la historia de los trasplantes con su estudio sobre la droga 6 mercaptopurina [28-29], que se utiliza para el tratamiento de la leucemia, demostrando que evita la formación de anticuerpos en conejos inoculados con proteína extraña; llamaron a esta observación tolerancia inmunológica inducida por fármacos [30]. A partir de este antimetabolito, la empresa Borroughz-Wellcome desarrolló la azatioprina [31], el antiproliferativo que ha sido pieza fundamental del régimen inmunosupresor en los receptores de trasplante durante casi cuarenta años. En los sesenta, los trabajos de Calne en trasplante de riñón animal, como Merril en humanos fueron seguidos después con Starzl quien combinó exitosamente la azatioprina con corticosteroide como régimen inmunosupresor para trasplante. En 1960 se utilizó el fármaco por primera vez en humanos con resultados mixtos. A pesar de que podría evitar el rechazo del nuevo órgano, los médicos tuvieron dificultades para distinguir entre dosis útiles, perjudiciales e incluso mortales [18]. Desde 1962, todos los trasplantes de órganos y tejidos entre personas no relacionadas se han realizado con presencia de inmunosupresión farmacológica.

Murray, quien había participado de la operación a Herrick en 1954, realizó el primer trasplante exitoso de riñón de un cadáver a un humano vivo en 1962 [29]. Utilizó azatioprina y esteroides para que el cuerpo del receptor acepte el órgano extraño. El paciente sobrevivió por más de un año sin tener que realizarse diálisis, fue la persona que más había sobrevivido en estas condiciones hasta el momento. El hecho hizo que los científicos investigaran más profundamente la posibilidad de utilizar órganos de cadáveres. Claramente, la oferta de órganos utilizables inmediatamente aumentó ya que no se necesitan donantes vivos, aunque con esto un debate moral importante asomaba. Además la invención de los fármacos inmunosupresores permitió eliminar el requisito de que el receptor y el donante estén estrechamente relacionados.

En 1983 se inventó la ciclosporina, un eficaz inmunosupresor usado hasta hoy día [32].

El trasplante de órganos, limitado a realizarse en no más que una decena de centros en todo el mundo, pasaba a realizarse en más lugares a medida que se avanzaba en la disciplina. El riñón era el órgano más trasplantado, pero poco a poco se fueron teniendo mejores resultados con el corazón, el hígado, y con el tiempo los pulmones y el páncreas.

El desarrollo del tratamiento de diálisis durante los cuarenta y cincuenta, tuvo un importante impacto en el trasplante renal. Con la hemodiálisis se amplió considerablemente la cantidad de pacientes que podrían beneficiarse de un trasplante de riñón.

Hígado

El órgano noble por excelencia fue trasplantado por primera vez de un humano a otro el 1 de marzo de 1963 en la Universidad de Colorado por Although Starzl, el niño receptor padeciente de atresia biliar tuvo solo cinco horas de superviviencia [33], su segundo intento fue dos meses más tarde entre adultos, veintidós días de supervivencia y con éxito absoluto en 1967 luego de que Cannon lo intentara en el 56 en California, Los Ángeles y se experimentara con animales en la década del 50; en 1955 Welch realizó el primer trasplante hepático heterotópico en un perro.

Corazón

En Ciudad del Cabo, el tres de diciembre de 1967, Christiaan Barnard implantó por primera vez un corazón a un ser humano [34]. La donante, Dénise Darvall tenía politraumatismos tras un accidente y presentaba mínima actividad cerebral, por lo que el médico solicitó la donación del corazón al padre de la víctima, y éste decide dar vida a quien sería el receptor del órgano, un hombre con miocardiopatía isquémica en estado terminal. Al cesar la actividad cardíaca de la joven y se comprobara la ausencia de respiración se implantó el corazón en el hombre que a los diez días caminaba por la habitación. No obstante muere a los dieciocho días de realizado el trasplante. Poco después Shumway realizó un trasplante de corazón exitoso en Stanford [35]. Otros intentos fueron tan desastrosos que el trasplante cardíaco humano fue abandonado hasta mediados de los setenta.

Antes, en 1964 James D. Hardy trasplantó el corazón de un chimpancé en un paciente de 68 años [36 - 37], el paciente sobrevivió solo una hora ya que el corazón del donante era demasiado pequeño como para mantener el volumen de sangre del receptor. Los primeros intentos habían sido en 1905 con Carrel y Guthrie que trasplantaron un corazón en el cuello de un perro. En la década del sesenta, Lower y Shumway tuvieron éxito con el trasplante ortotópico entre caninos en Stanford.

Pulmón

James D. Hardy, en Jackson (Mississippi) efectuó el primer trasplante de pulmón de la historia en junio de 1963 [37] y salvó la vida del receptor del pulmón John Russel. De 58 años con cáncer de pulmón e insuficiencia respiratoria y renal que estaba condenado a muerte, por la naturaleza obviamente, pero además por la justicia estadounidense por asesinato. Se le

propuso ser el primer hombre trasplantado pulmonarmente de la historia a cambio de la libertad por contribución a la causa de la humanidad y aceptó. Se le trasplantó entonces el pulmón izquierdo y duró dieciséis días con buenas funciones pero el enfermo falleció como consecuencia de la agudización de su insuficiencia renal.

Le siguieron otros intentos pero sin mejor suerte. Dentoy Cooley en 1968, arriesgó haciendo el primer trasplante en bloque de corazón y ambos pulmones pero sin éxito. En este mismo año el cirujano belga Fritz Derom logró que un paciente sobreviva diez meses luego de trasplantarle el pulmón. Nuevamente malos resultados hacen que en los setenta casi que desaparezca la técnica, que vuelve finalmente para triunfar en el ochenta gracias a la aparición de la ciclospirina. El Dr. Cooper, en Canadá sentó las bases del trasplante unipulmonar, al inicio de la década, y bipulmonar en 1986 [38].

Páncreas

En 1966 Richard Lillehei junto a William Kelly realizaron el primer trasplante exitoso de páncreas/riñón y duodeno en la Universidad de Minnesota en Minneapolis a una mujer de veintiocho años. Tras el trasplante los niveles de azúcar en sangre disminuyeron inmediatamente pero finalmente murió tres meses después de una embolia pulmonar [39].

2.3 Actualidad

El Baltimore's Johns Hopkins Hospital, en 2005, es pionero de probablemente el primer *domino chain* método para matching de donantes y receptores. Se trasplantaron tres riñones de donantes vivos en receptores con los que no tenían parentesco (pero si compatibilidad), es más: desconocidos hasta entonces. Uno de los donantes era incluso altruista. Otro era la esposa de un receptor y el último era la hija de otro receptor [40]. Una forma original de buscar el mejor receptor para un órgano y que continúa dándose exitosamente.

2.4 Ética

Al trasplante de órganos lo rodean muchos aspectos y dilemas éticos. Empezando casi que por su definición: la justificación del trasplante. Hay corrientes de opinión que no justifican el trasplante porque lo consideran una mutilación del cuerpo humano, sea este vivo o difunto [41]. Sobre la comercialización de órganos, si bien es una actividad considerada ilegal en casi todo el mundo, hay quienes están a favor de la misma argumentando que 1) vender o comprar partes del cuerpo forma parte de la privacidad de cada uno, 2) el comercio libre incrementaría enormemente los trasplantes [41].

Uno de los aspectos más importantes para nuestro estudio es el que tiene que ver con el criterio de asignación de órganos a los receptores. El método de asignación podría considerar únicamente aspectos científicos: a partir de un órgano disponible obtener el mejor receptor. ¿Pero qué es ser el "mejor" receptor? ¿Aquel en el que el órgano durará más tiempo? ¿El qué es más biológicamente compatible? Sobre la lista de espera, ¿debe ser únicamente *First In First Out*? Si lo es, aquellos individuos que necesitan órganos con mayor urgencia que otros posiblemente morirán en la espera mientras que otros, que pueden sustituir sus funciones vitales, reciben el órgano únicamente por haberlo necesitado desde antes. Si le agregamos un criterio de prioridad que tenga que ver con la urgencia de la necesidad del órgano esperado, y con un criterio únicamente científico para asignar el órgano, posiblemente en la cola haya posposición indefinida y los de baja prioridad nunca tendrán posibilidad de recibir un órgano. Por otro lado el método de asignación podría tener en cuenta más bien aspectos sociales como por ejemplo la edad del individuo: asignar un órgano al individuo en espera más joven (pues en general tendrá más vida por delante), pero, si no es compatible el organismo lo rechazará. Parece razonable hacer un balance entre los criterios científicos o médicos y los aspectos sociales. Pero sobre todo el método deberá buscar la máxima equidad en la asignación de órganos, ser público, claro y trasparente.

"Un buen sistema de asignación de los órganos donados a la sociedad debe procurar el equilibrio entre equidad, utilidad médica y eficiencia. Debe garantizar transparencia y objetividad en la selección del receptor, lograr la utilización óptima del órgano donado, maximizar la sobrevida del injerto y del paciente y brindar condiciones de acceso al trasplante para pacientes con desventajas biológicas o médicas. A pesar de lo anteriormente señalado, algunos sistemas de asignación han estado o están basados exclusivamente en criterios clínicos y de territorialidad" [42].

Otro dilema surge del hecho de que la mayoría de los órganos que se utilizan son de donante muerto. ¿Cómo se define la muerte? El avance en el trasplante de órganos dio lugar a una nueva definición.

2.5 Legislación

En un contexto de controversia, con una demanda de órganos creciente y una oferta estancada, el *Ad Hoc Committee of the Harvard Medical School*, declaró una nueva forma de definir la muerte en Estados Unidos (e involuntariamente en el mundo) que permitía aprovechar más órganos que hasta entonces; le llamó muerte encefálica [43]. Anteriormente una persona se decía muerta (muerte biológica) cuando su corazón dejaba de latir y no podía respirar voluntariamente. El comité definió cuatro nuevos estándares para determinar la muerte: 1) *unreceptivity* y *unresponsitivity*, 2) ausencia de

movimientos o respiración, 3) ausencia de reflejos, y 4) electroencefalograma plano [44].

El tema empezó a abordarse en todo el mundo. Permitió además la creación de herramientas para asistencia respiratoria mecánica que permiten mantener oxigenado los órganos aún ante la muerte del individuo para poder reutilizar los órganos en estado pre-muerte.

En Uruguay la ley 14.005 [45] de 1971 y vigente hasta hoy día[1], no deja del todo claro la forma de definir la muerte: "cambios patológicos irreversibles, incompatibles con la vida", por lo que deja el diagnóstico de muerte a su comprobación por parte de los médicos, aunque al menos condiciona que éstos no pertenezcan a los equipos de trasplantes.

La muerte encefálica se ha abordado públicamente en nuestro país desde 1980 a través de declaraciones de Neurocirugía, Neurología y de Medicina Intensiva [46]. El Instituto Nacional de Donación y Trasplantes elaboró y actualiza un consenso [46] de muerte encefálica que sirve de guía al médico para determinarla.

La flamante Ley N° 18.968 de Donación y trasplante de células, órganos y tejidos, en vigencia desde setiembre de 2013, que plantea modificaciones a la Ley N° 14.005 decreta en su art. 1° que: "Toda persona mayor de edad que, en pleno uso de sus facultades, no haya expresado su oposición a ser donante por alguna de las formas previstas en el artículo 2º de la presente ley, se presumirá que ha consentido a la ablación de sus órganos, tejidos y células en caso de muerte, con fines terapéuticos o científicos.". Por lo tanto todos los uruguayos son donantes a menos que expresen lo contrario.

2.6 Uruguay exitoso

El Banco Nacional de Órganos y Tejidos (BNOT) nace el 17 de noviembre de 1978, como un proyecto conjunto del Poder Ejecutivo y La Universidad de la República de ser la organización nacional que instrumenta y gestiona los cometidos de la ley de donación y trasplantes de órganos y tejidos de 1971 [47]. El ocho de agosto de 2005 pasó a llamarse Instituto Nacional de Donación y Trasplante de Células, Tejidos y Órganos (INDT) [48].

Algunos de los logros hasta el 2003 son [47]:

- 1978 - Homoinjertos de piel.
- 1980 - Homoinjertos de amnios.

[1] Salvo algunos artículos que fueron derogados por la Ley 18.968

- 1981 - Trasplante renal cadavérico.
- 1985 - Trasplante de médula ósea.
- 1996 - Homoinjertos valvulares.
- 1996 - Trasplante cardíaco.
- 1998 - Trasplante hepático.
- 2002 - Trasplante reno-pancreático.
- 2002 - Homoinjertos de arterias crioconservadas.
- 2003 – Trasplante Alogénico de médula ósea con donante no emparentado.

En el año 2013, se realizaron en el INDT cinco trasplantes de corazón, veinticinco de hígado, 111 de riñón y cuatro de riñón-páncreas; 145 en total. Tal como se ve en la Figura 2.1. En 2012 se trasplantaron 132 órganos, mientras que en 2011, 162.

Población (millón de hab.)	3,2				3,3										3,28			
	2005		2006		2007		2008		2009		2010		2011		2012		2013	
	Nº	pmp	Nº	pmp	Nº	pmp	Nº	pmp	Nº	pmp	Nº	pmp	Nº	pmp	Nº	pmp	Nº	pmp
CORAZON																		
Procuración	28	8,75	24	7,50	11	3,33	22	6,66	14	4,24	12	3,63	10	3,03	7	2,13	7	2,13
Trasplante (Tx)	7	2,19	7	2,19	3	0,91	11	3,33	7	2,12	7	2,12	8	2,42	7	2,13	5	1,52
CORAZON-BIPULMON																		
Procuración					1	0,30												
Trasplante (Tx)					1	0,30												
PULMON																		
Procuración					2	0,60	4	1,21			2	0,60	2	0,60	6	1,82		
Trasplante (Tx)					2	0,60	4	1,21			2	0,60	2	0,60	8	2,44		
HIGADO																		
Procuración	1	0,31	11	3,44	5	1,51	3	0,9	10	3,03	13	3,93	23	6,96	18	5,48	25	7,62
Trasplante (Tx)	1	0,31	10	3,13	5	1,51	3	0,9	8	2,42	12	3,63	22	6,66	17	5,18	25	7,62
RIÑON																		
Procuración	116	36,25	146	45,63	106	32,12	119	36,06	123	37,27	92	27,87	128	38,78	94	28,65	107	32,62
Trasplante (Tx)	119	37,19	142	44,38	102	31	121	36,66	115	34,84	89	26,96	124	37,57	93	28,35	111	33,84
cadavérico	116	36,25	134	41,88	96	29,09	114	34,54	108	32,72	86	26,06	120	36,36	90	27,44	100	30,48
intervivo	3	0,94	8	2,50	6	1,82	7	2,12	7	2,12	3	0,90	4	1,21	3	0,91	11	3,35
RIÑON - CORAZON																		
Procuración			1	0,31														
Trasplante (Tx)			1	0,31														
RIÑON – PANCREAS																		
Procuración	3	0,94	8	2,50	3	0,91	6	1,81	6	1,81	4	1,21	5	1,51	5	1,52	5	1,52
Trasplante(Tx)	2	0,63	6	1,88	2	0,60	6	1,81	5	1,51	4	1,21	5	1,51	5	1,52	4	1,21
HIGADO – RIÑON																		
Procuración													1	0.3	1	0,3		
Trasplante (Tx)													1	0.3	1	0,3		
INTESTINO																		
Procuración																		
Trasplante (Tx)															1	0.3		

Figura 2.1 - *Procuración y trasplante de órganos. Período 2005-2013 [49]*

El trasplante de órganos y tejidos de Uruguay es exitoso principalmente por dos factores. El buen trabajo que desde su creación viene llevando el BNOT/INDT, reconocido internacionalmente, y que en la actualidad tiene una tasa de efectividad en los trasplantes de más del 90% [50]. Y porque Uruguay lidera la cantidad de donantes per cápita de Latinoamérica. En 2012, tal como muestra la Figura 2.2, en Uruguay la media fue de 28,2 por millón de población [51], mientras que en Latinoamérica fue de 6,5 por millón. La Ley N° 18.968 es una consecuencia de nuestra sociedad donante. Por supuesto que

para que la población sea donante es fundamental la confianza en el sistema de trasplante. Hecho que hace fracasar los sistemas de otros países.

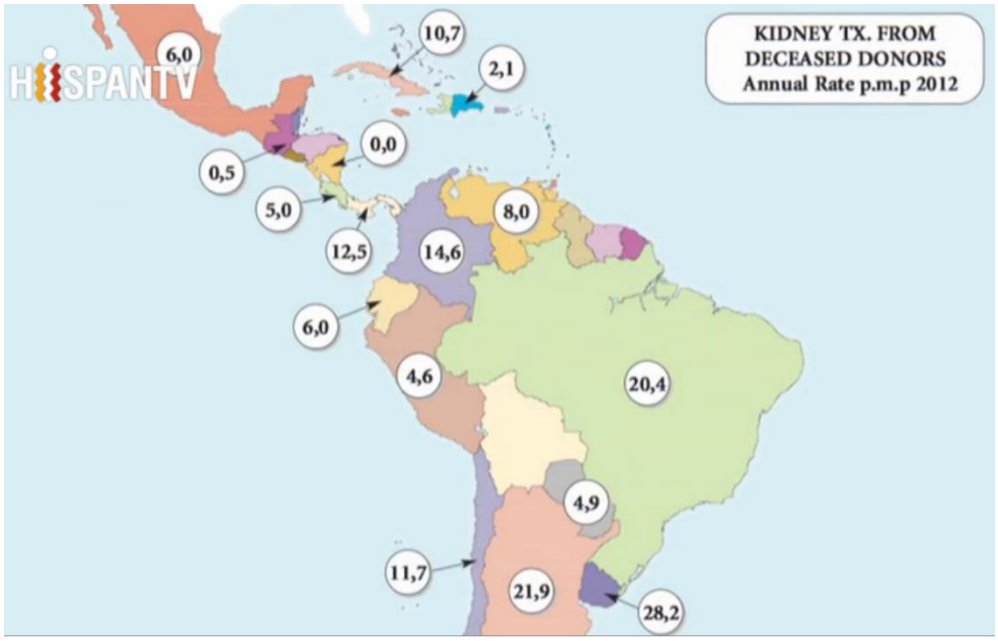


Figura 2.2 - *Donantes por millón de población*
Datos de La gran historia [51]

Pero hay otros motivos que colaboran. Nuestra geografía contribuye a los trasplantes. Órganos como el corazón y pulmones sobreviven afuera del cuerpo entre 4 y 6 horas, por lo que en Brasil es imposible transportar un corazón de Porto Alegre a Fortaleza en tal lapso de tiempo. Sin embargo en Uruguay es posible ir de un extremo a otro en alrededor de una hora. El nuevo sistema de Transporte Aeromédico de Emergencia de la Fuerza Aérea Uruguaya aprovechará este aspecto [52]. La idiosincrasia uruguaya hizo que tengamos un único centro nacional de gestión de los órganos. En otros países los diferentes centros compiten por los órganos y los aspectos políticos pueden tener lugar. Hay centros de trasplante que sólo trasplantan casos “sencillos” para tener buenas estadísticas y atraer socios. En nuestro país no existe cualquier tipo de competencia.

Capítulo 3

REQUISITOS

En el presente capítulo se busca definir y principalmente especificar los requisitos del software a construir.

3.1 Visión General

Se espera que Donamatch sea un sistema de información para el apareamiento de donantes y receptores: dado un donante de órganos (informado a raíz de un accidente), se ingresan sus características y el sistema busca en bases de datos los mejores receptores en las listas de espera.

Se busca realizar un sistema para mejorar la asignación de órganos y sus derivados. Dar a los médicos y al personal que trabaja en los hospitales la posibilidad de conocer en tiempo real los posibles apareamientos de forma rápida y certera.

3.2 Actores

Se identifican dos actores en el sistema que serán sus usuarios:

- Administrador
  Los usuarios administradores serán los encargados de la configuración del matching. Son las personas que en la actualidad se encargan de la definición de la asignación de órganos de los centros de trasplante.

- Colaborador
  Los usuarios colaboradores son los encargados de ejecutar el matching. Son aquellas personas que pertenecen a personal de salud y que hacen parte del sistema de emergencia de trasplante de órganos de un centro.

Donamatch no cambia los actores de la asignación de órganos de la realidad. Los mismos actores encargados de definir los modelos de asignación, serán quienes definirán estos modelos en el sistema. Y los actores que en la actualidad asignan órganos a receptores según esos modelos, serán los responsables de hacerlo en Donamatch.

3.3 Especificación de requisitos

A continuación se especifican tanto los requisitos funcionales como los no funcionales. Están priorizados por importancia en el sistema.

3.3.1 Definición cambiante de criterios de matching

Los criterios de asignación deberán poder cambiar a lo largo de la vida de Donamatch. Podrán definirse y luego modificarse en cualquier momento. El sistema deberá proveer una interfaz para este cometido, la cual no podrá ser accedida por cualquier usuario. Únicamente usuarios con autorización podrán hacerlo, en general será un comité del centro de trasplantes el responsable de definir los criterios de matching.

3.3.2 Agilidad para encontrar el mejor receptor

Dado que el tiempo que dura un órgano fuera del cuerpo del donante es relativamente pequeño, y se tienen que desencadenar una serie de acciones para poder llevar a cabo el trasplante, el tiempo que lleve encontrar el mejor receptor para el órgano disponible debe ser el menor posible.

El sistema deberá ser construido de forma tal que realizar el matching tenga una latencia promedio igual al 10% del tiempo que se mantiene sano un órgano fuera del donante, en el peor caso.

Tabla 3.1 – *Duración de órganos fuera del donante. Datos de Organdonor [51]*

Órgano	**Duración (horas)**
Corazón	4 – 6
Pulmón	4 – 6
Hígado	12 – 15
Riñón	36 – 48

Según la Tabla 3.1 el tiempo que debe emplear realizar el matching, debe ser menor a veinticuatro minutos.

Este tiempo implica seleccionar el matching a realizar, ingresar los datos requeridos para el órgano correspondiente, ejecutar el procesamiento lógico de los datos contra los candidatos a receptores por parte del sistema, mostrar resultados ordenados, obtener datos del receptor elegido y generar la información que se adjuntará a la Historia Clínica Electrónica (HCE).

3.3.3 Seguridad

El sistema deberá ser seguro. No deberá poder accederse por un usuario que no esté previamente registrado. Se debe poder identificar, en cualquier momento, a todos y cada uno de los usuarios del sistema.

La configuración de matching no podrá hacerse por cualquier usuario, sino que por uno que esté autorizado para ello.

3.3.4 Registrar acciones del sistema

Cada acción que se ejecute en el sistema deberá quedar registrada, identificando quién la realizó y en qué momento. Tanto para la definición del matching como para la ejecución.

En la asignación de un órgano debe quedar registrado el resultado obtenido por todos los receptores que participaron en la competencia por el órgano con el fin de realizar auditorías sobre el proceso.

Este requisito está alineado al Principio Rector n° 9 definido por la OMS: "La organización y ejecución de las actividades de donación y trasplante, así como sus resultados clínicos, deben ser trasparentes y abiertos a inspección, pero garantizando siempre la protección del anonimato personal y la privacidad de los donantes y receptores" [3].

3.3.5 Generar información para Historia Clínica Electrónica

Luego de realizado el matching, deberá generarse un archivo en formato *Clinical Document Architecture* (CDA) de los estándares de *Health Level-7* (HL7) con información de la asignación realizada.

Esta información se adjuntará a la HCE del receptor y además se usará como registro del matching realizado.

3.3.6 Internacionalización

Donamatch deberá ser un sistema multilenguaje para facilitar la colaboración fuera de fronteras. Se deberá soportar tres idiomas: español, inglés y portugués.

Además, los principales textos del sistema deberán poder corregirse dinámicamente para mejorar su uso. O para el caso en el que quien define el término no está seguro de su correcta traducción.

3.3.7 Disponibilidad

El sistema deberá estar disponible para utilizarse desde cualquier punto del planeta y todos los días del año.

El disparador del uso del sistema será un órgano disponible en el centro de trasplantes (o en el contexto particular de dónde se use: ciudad, país, etc.), que puede surgir en cualquier momento, es impredecible. Por lo tanto el sistema debe estar disponible para usarse siempre, pues sino será inútil.

3.4 Requisitos excluidos

Originalmente, se tenía como objetivo que el sistema también buscara donantes de sangre: dado un receptor de sangre en un hospital, se ingresa sus características y el sistema busca en bases de datos de bancos de sangre los posibles donantes compatibles, ordenándolos. Trabajo vinculado con el proyecto *Hemológica* que se ocupa de la logística de la donación de sangre. Tal como se muestra en la Figura 7.2, este requisito fue recortado debido al tamaño del alcance total respecto del tiempo disponible para el proyecto de grado.

En los próximos capítulos, se afrontarán las soluciones propuestas para el sistema a construir, determinado principalmente por los requerimientos funcionales y no funcionales vistos. Además se detallarán aspectos técnicos y de implementación, resultados obtenidos, conclusiones y líneas de trabajo futuro.

Capítulo 4

SOLUCIÓN PLANTEADA

En el presente capítulo se presenta y detalla la solución propuesta. Empezando por describir su aspecto genérico y el criterio Donamatch (DM) de matching, luego la definición de conceptos, la arquitectura y por el último el diseño de bajo nivel.

4.1 Álgebra del trasplante

Para cumplir con el requerimiento 3.3.1, se ha diseñado un sistema totalmente configurable (aunque existen algunos aspectos secundarios que no se pueden configurar). El principal reto de Donamatch fue su aspecto genérico y también su principal cualidad. Su capacidad de cambiar los criterios de asignación de órganos.

Aunque a lo largo del documento se evidencian algunas motivaciones, vale la pena destacar otras. Un software a medida, por un lado le serviría probablemente a un único centro de trasplantes del mundo. Pues aunque algunos tienen criterios más semejantes entre sí que otros, todos terminan por diferir en el método de asignación con el que deciden a qué receptor le corresponde el órgano donado. Ya sea por aspectos culturales, sociales, políticos o científicos. Difieren. Por ende para que otro centro de trasplantes pudiera utilizarlo, el software debería indefectiblemente adaptarse a la realidad de éste (modificando el código fuente). Por otro lado, si el centro resuelve cambiar la forma con la que hace el matching, por ejemplo porque hay evidencia científica que demuestra que existen nuevas técnicas para la compatibilidad donador-receptor (nunca más vigente que ahora con el estudio del genoma humano), el software debería rehacerse para considerar estos nuevos aspectos.

Como corolario de esto, Donamatch puede transformarse, con adaptaciones, en software de matching de casi cualquier cosa. Por ejemplo: de desempleado con empleador, de inmuebles y compradores, de parejas, etc. Mediante la adecuada configuración del sistema, que impacta en el Criterio Donamatch que se introducirá en la siguiente sección. Las adaptaciones tendrán que ver con las pantallas que deberán adecuarse a cada caso.

Entre las inspiraciones filosóficas del software se encuentra la fuerte influencia del Sistema Informático Perinatal (SIP) [54], que ha sido estudiado profundamente en este trabajo no solo por ser un caso de éxito en

Latinoamérica y en este contexto tener el objetivo común de traspasar fronteras, sino por ser un sistema altamente configurable, aunque en un contexto algo diferente, ya muchos años y tecnologías atrás.

Criterio Donamatch

Dadas las condiciones genéricas del sistema, es necesario abstraerse para lograr definir un criterio general que pueda ser usado para la asociación de cualquier órgano-receptor.

En nuestra realidad tenemos dos grandes entidades: el órgano donado y el receptor. Entonces, los criterios de asignación serán siempre aspectos relacionados:

- al órgano donado
- al receptor
- a ambos

El órgano donado vive una transición: es extraído del donante para cumplir funciones vitales en el receptor. Por lo tanto, carecería de sentido tener aspectos que determinen el matching que solo tengan que ver con tal órgano.

Aspectos referentes a ambos, por definición, no podrían faltar pues son los que los relacionan. Y los aspectos relacionados únicamente con el receptor son necesarios para aplicar, por ejemplo aspectos sociales.

Como corolario del razonamiento anterior se definió el criterio Donamatch con el que se hace el matching. El DM para el receptor r es:

$$DM_r = \prod_{i=1}^{l} \alpha_i \times \prod_{j=1}^{m} \beta_j \times \sum_{k=1}^{n} (\gamma_k \times \delta_k) \tag{4.1}$$

Donde:

$$\begin{gathered} \alpha_i / \alpha_i \in \mathbb{R}, \alpha_i \in [0,1] \\ \beta_j / \beta_j \in \mathbb{N}, \beta_j \in \{0,1\} \\ \gamma_k / \gamma_k \in \mathbb{R}, \gamma_k \in [0,1] \\ \delta_k / \delta_k \in \mathbb{N}, \delta_k \in \{0,1\} \\ i, j, k, l, m, n \in \mathbb{N} \end{gathered} \tag{4.2}$$

Criterios científicos - γ_k

Son los criterios que relacionan al órgano donado y al receptor. En general serán criterios científicos, de ahí el nombre. Determinan la compatibilidad

entre el órgano y el receptor: por ejemplo Pruebas Cruzadas por *Microlinfocitotoxicidad* para el caso de trasplante de riñón.

γ_k determina el peso que tiene el k-ésimo criterio científico en el DM.

$k \in \mathbb{N}$, es la cantidad de criterios científicos existentes para el matching órgano-receptor.

Factores de compatibilidad - δ_k

Determinan si un criterio científico k estará presente en el DM de un caso particular. Tienen que ver con el resultado obtenido para el criterio científico en cuestión. Para el ejemplo de Pruebas Cruzadas, si el resultado es negativo, δ_k tomará el valor 1, en cambio si el test da positivo, δ_k tomará el valor 0.

En el primer caso, γ_k no interferirá en el valor de DM.

Excluyentes - β_j

Son los criterios que determinan si el receptor está apto para recibir el órgano donado. También son criterios científicos pero a diferencia de los anteriores, si no se cumplen hacen que el DM de cero. Eliminando así al receptor de la posibilidad de recibir el órgano.

Un ejemplo es la compatibilidad de sangre. Si un receptor no tiene compatibilidad sanguínea con el donante, no podrá recibir el órgano pues será agudamente rechazado.

$j \in \mathbb{N}$, es la cantidad de criterios excluyentes.

Políticas de receptor - α_i

Las políticas del receptor son los aspectos que están relacionados únicamente directamente al receptor. Podrán tener que ver con la lista de espera, la cultura o la sociedad. Pero en todos los casos promoverán o degradarán al receptor. Son aspectos de índole política.

Por ejemplo la edad del receptor es una política de receptor.

$i \in \mathbb{N}$, es la cantidad de políticas de receptor que existe para el matching.

Asignación

El órgano donado se asignará al receptor r que verifique:

$$\max\{\, CG_r \in \mathbb{R} \mid CG_r = \prod_{i=1}^{l} \alpha_i \times \prod_{j=1}^{m} \beta_j \times \sum_{k=1}^{n} (\gamma_k \times \delta_k) \,\} \qquad (4.3)$$

Con $\alpha_i, \beta_j, \gamma_k, \delta_k, i, j, k, l, m, n$ como en (4.1)

4.2 Compromiso abstracción-usabilidad

La definición de asignación de órganos genérica de (4.3) permite ser flexible y adaptarse a la realidad del trasplante de órganos brindando al usuario la posibilidad de modificar el criterio de asignación mediante la determinación de $\alpha_i, \beta_j, \gamma_k$ y δ_k según (4.2) e i, j, k, l, m y n.

Esto conllevaría a la necesidad de entender conceptualmente el significado de cada variable por parte del usuario para que pueda mapearlo a la realidad. Lo que sesgaría fuertemente el perfil del usuario final de Donamatch. En cambio facilitaría el diseño y la implementación del sistema, ya que la configuración del matching se reduciría a almacenar los valores ingresados por el usuario, y la búsqueda del mejor receptor a aplicar (4.1) para obtener (4.3).

Diseñar un sistema fuertemente adaptativo pero que requiere un muy alto nivel de complejidad para poder configurarse parece ser contradictorio. Además de aumentar el riesgo de que el matching quede mal configurado, lo que desencadenaría en la posible asignación ineficiente del órgano; objetivo contrario al del sistema. Por otra parte no se cumpliría el requerimiento 3.3.2 ya que sería sumamente complejo hacer el matching.

Se buscará disminuir al máximo la complejidad de la configuración del sistema. Ahora bien, cuanto más lejos esté la solución de la abstracción de (4.3) más compleja se tornará la construcción del sistema, aspecto que vale la pena asumir en beneficio de la usabilidad. Sin embargo hay un límite al que, si bien se tenderá, no será posible pasarse. Pues si se concreciona mucho, se perderá el aspecto genérico del sistema.

Se propone una solución que surge del compromiso de necesidades: entre la abstracción necesaria para fortalecer la flexibilidad del sistema y la usabilidad necesaria para evitar el fracaso de su uso.

Se definen conceptos que serán el núcleo del sistema: entidades, atributos, funciones, metafunciones, operadores, unidades de medida, políticas, asociación de atributos y matchings. Estos conceptos son una especie de punto intermedio entre la abstracción de (4.3) y el trasplante de órganos.

4.3 Conceptos

Entidades

Las entidades son cosas que el sistema emparejará. Una entidad es un ente con determinadas características, mediante las cuales se hará corresponder con otra entidad.

Es uno de los conceptos más importantes. Los órganos y los receptores de órganos deberán configurarse como entidades del sistema.

Atributos

Atributos o atributos de entidad, son las cualidades de las entidades. Y cualquier otro aspecto concerniente al matching que tenga relación directa con la entidad.

Los atributos pertenecen a la entidad, por lo tanto existen si existe la entidad. Pueden ser mensurables.

Unidades de medida

Son unidades de medida que en el caso de atributos de entidad mensurables, indican en qué se mide.

Funciones

Son funciones matemáticas o lógicas de los atributos de entidades provistas por el sistema para aplicar el apareamiento.

Metafunciones

Las metafunciones son funciones sobre atributos de entidades que se definen a partir de funciones. Se combinan funciones para crear otras mediante la aplicación de operadores.

Operadores

Son operadores lógicos y matemáticos provistos por el sistema que se usan en la definición de las metafunciones.

Asociación de Atributos

La asociación de atributos es una asociación entre atributos de entidades definida mediante una metafunción o una función. Puede ser un requisito para el emparejamiento de entidades o puede tener una ponderación asociada para ese emparejamiento.

La asociación es vista como un resultado, cuantificable.

Políticas de ranking

Políticas o políticas de ranking, son las políticas del receptor que, una vez definidas permiten clasificar a los receptores, luego de aplicar aspectos científicos.

Matchings

Un matching o matching entre entidades es un matching entre dos entidades que el sistema será capaz de hacer.

Se define mediante dos entidades, un conjunto de asociaciones de atributos y un conjunto de políticas.

4.4 Arquitectura

Es cierto que el estilo y estructura particular de una aplicación en general dependen fuertemente de los requerimientos no funcionales. Sin embargo en el caso de Donamatch, se agrega que el estilo arquitectónico se vio fuertemente influenciado por su aspecto funcional. Es muy importante la seguridad, disponibilidad y agilidad para encontrar el receptor pero tan o más importante es la usabilidad del sistema, es fundamental que el matching sea práctico, amigable y funcional (es cierto también que estos puntos van de la mano con el último requisito no funcional nombrado).

Para cumplir con los requisitos 3.3.2 y 3.3.7, se diseñó un sistema web, ya que permite acceder fácilmente desde cualquier lugar a la asignación de órganos y dependiendo de la infraestructura podrá tener hasta una disponibilidad 24/7. E incluso, podrá funcionar sin conexión a internet. El fácil acceso podrá contribuir con consumir el menor tiempo posible para determinar el receptor.

No por tener una fisonomía autosuficiente (en el sentido de cambiar la forma de hacer el matching sin necesidad de reescribirse), deberá dejarse de lado la escalabilidad y mantenibilidad del sistema. Es sabido que tarde o temprano los sistemas necesitan cambiar y se encuentran defectos, o a veces solamente fallas (otra semejanza: aún no somos capaces de escribir programas con cero defectos así como el trasplante de órganos no es capaz de asignar órganos con perfección). Por lo tanto es importante diseñar una arquitectura que facilite el mantenimiento del sistema así como la incorporación de nuevas funcionalidades y corrección de defectos.

La arquitectura general que presenta el sistema, está basada en Modelo-Vista-Controlador (MVC por su nombre en inglés *Model View Controller*), Modelo-Vista-Modelo de Vista (MVVM por su nombre en inglés Model View ViewModel), Unidad de Trabajo (UOW por su nombre en inglés *Unit Of Work*) y Aplicación de una sola página (SPA por su nombre en inglés *Single Page Application*).

A continuación se presentarán los principales aspectos de cada una.

Modelo Vista Controlador

La principal ventaja de MVC es que separa la interfaz gráfica y sus eventos y comunicaciones de la lógica de negocio proponiendo tres grandes componentes: el Modelo, la Vista y el Controlador [55]. El Modelo representa la información del sistema, es la "realidad". Gestiona los accesos a la información. El Controlador es el intermediario entre la Vista y el Modelo, controla los eventos del usuario e invoca al Modelo solicitando acciones sobre la información. La Vista presenta el Modelo de una forma adecuada al usuario [56]. La Figura 4.1 muestra la arquitectura MVC aplicada al sistema.

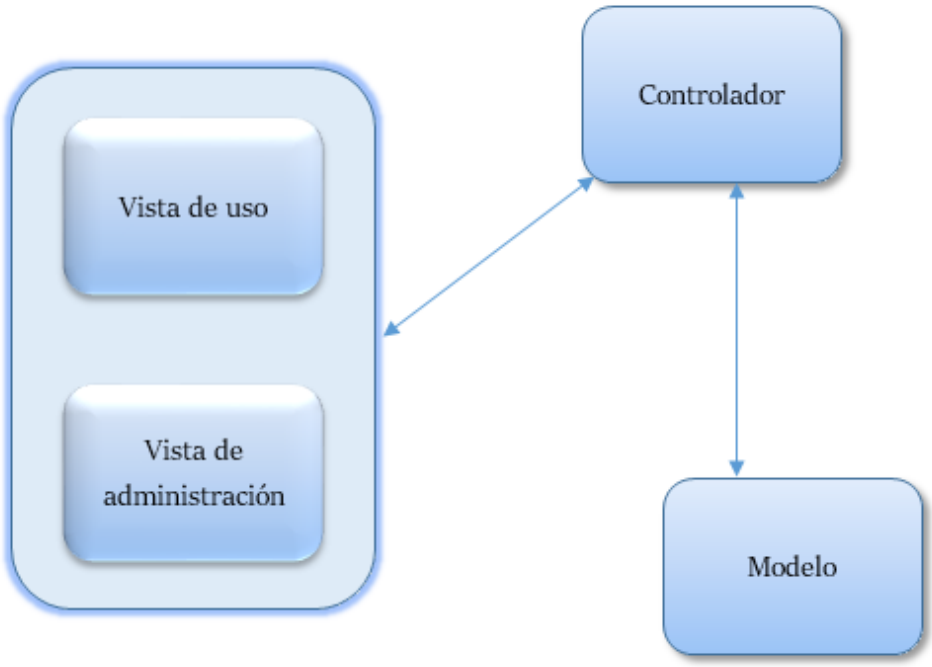


Figura 4.1 - *MVC aplicado en Donamatch.*
Las líneas representan las asociaciones directas entre componentes.

Los componentes de la arquitectura MVC del sistema se pueden ver en la Figura 4.1 y son las interfaces gráficas, los controladores y el modelo. MVC se puede ver como un corte transversal de las capas de presentación y lógica.

Modelo Vista Modelo de Vista

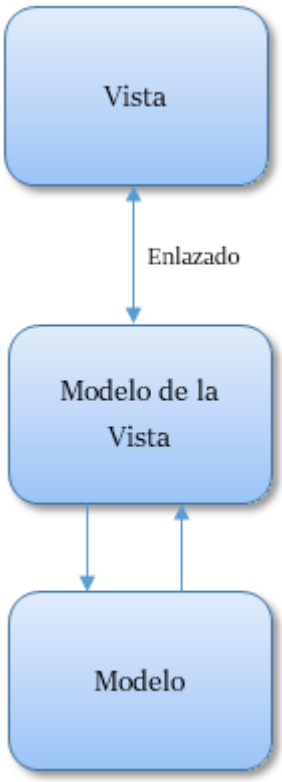


Figura 4.2 - *MVVM en Donamatch*

El MVVM, es un patrón de arquitectura en gran parte basado en MVC, de hecho es una implementación de MVC. Dirigido a plataformas de desarrollo de interfaz de usuario que soportan la programación orientada a eventos.

Tal como lo muestra la Figura 4.2, los principales conceptos asociados son: Modelo y Vista tal como en MVC. Modelo de la Vista (VM por su nombre en inglés *ViewModel*), que es una abstracción de la vista y a su vez mediación entre el Modelo y la Vista; puede verse como un controlador del MVC en el sentido de que convierte los cambios de información del modelo en información de la vista y pasa comandos desde la vista al modelo. El VM es el estado conceptual de los datos en oposición al estado real de los datos en el Modelo. Y por último, el Enlazador (traducción libre de *binder*) quien básicamente se encarga de sincronizar el Modelo de la Vista con la Vista.

El MVVM, facilita una clara separación entre la interfaz gráfica y la lógica.

Aplicación de una Sola Página

SPA son los sitios web que se navegan en una sola página con el principal objetivo de brindar una experiencia de usuario más fluida. Dando la idea de una aplicación de escritorio [57]. En una SPA, toda la información necesaria se obtiene en una simple carga. Luego todo lo requerido por la vista se carga a demanda en general como respuesta a acciones del usuario.

Además, permite guiar al usuario en el paso a paso de la aplicación. Usaremos SPA únicamente para la interfaz de uso común de matching que es la más crítica en cuanto a amigabilidad y usabilidad.

SPA permite aprovechar el aspecto de cliente fino de MVC haciendo que el software de presentación brinde una buena experiencia de usuario, aspecto que se complementará con la ejecución asincrónica de las acciones de las vistas.

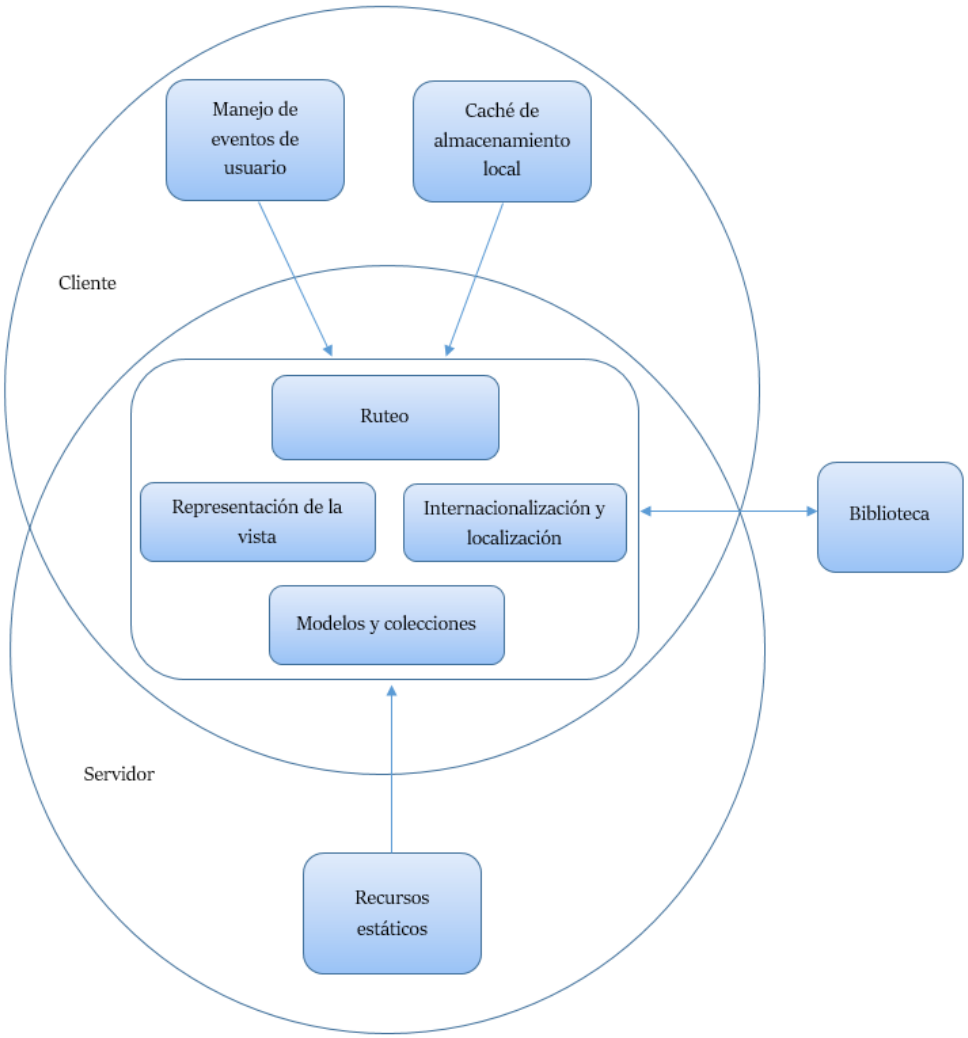


Figura 4.3 - *SPA en Donamatch.*
Con una arquitectura similar a la caracterizada como The Hard Way [58].

Unidad de Trabajo

Cambiar la base de datos con cada cambio que se realiza en el modelo puede dar lugar a un montón de invocaciones muy pequeñas que terminan impactando en el rendimiento del sistema, enlenteciéndolo. Además requiere tener una transacción abierta para toda la interacción lo que es poco práctico si tenemos una transacción de negocios que abarca varias peticiones. Una unidad de trabajo mantiene un registro de todo lo que se hace durante una transacción de negocio que puede afectar a la base de datos. Cuando haya terminado, se fija todo lo que ha cambiado desde la última transacción como resultado de su trabajo y lo impacta en la base de datos [59].

La arquitectura del sistema es en capas combinando MVC, UOW y SPA. La Figura 4.4 muestra la arquitectura parcial del sistema.

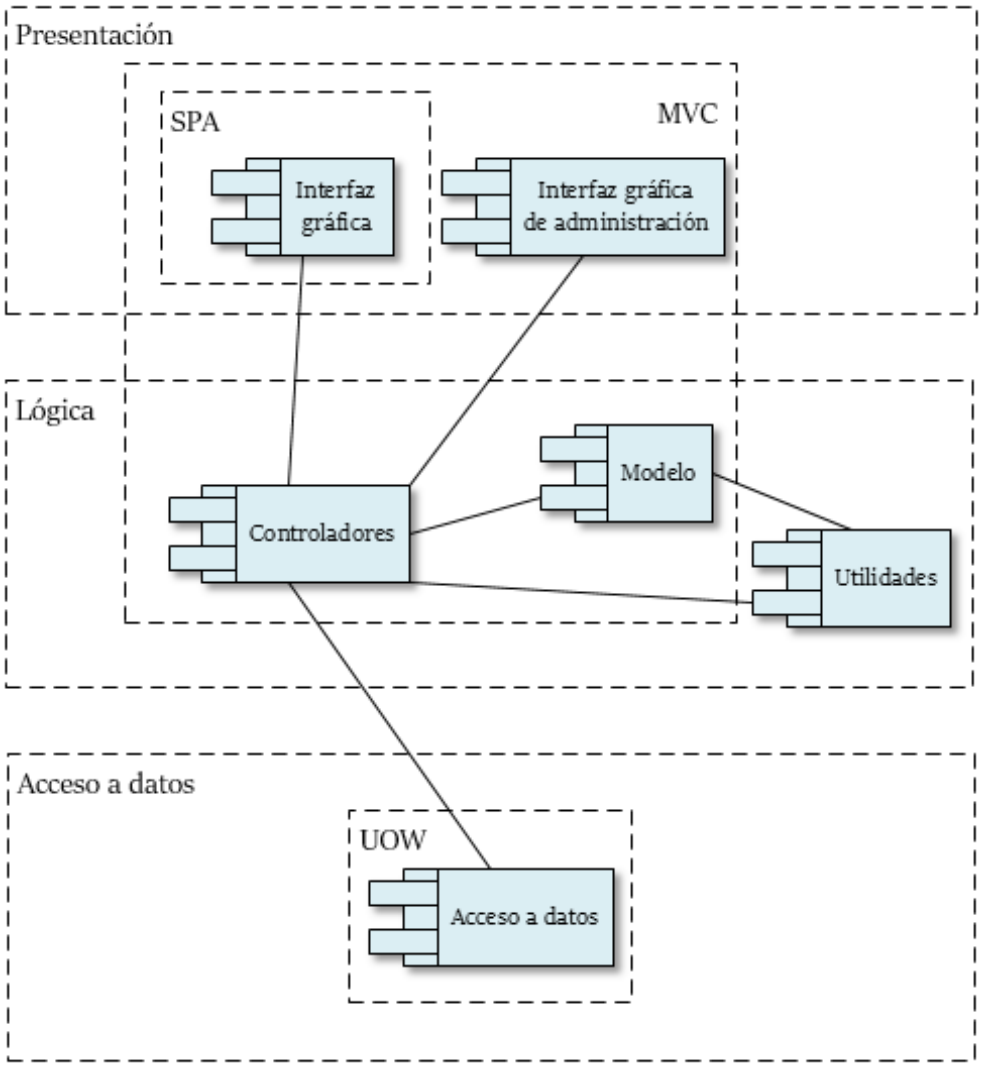


Figura 4.4 - *Arquitectura parcial de Donamatch.*

El sistema presenta tres grandes capas: Presentación, Lógica y Acceso a datos. En la Presentación se destacan la Interfaz Gráfica (a secas para diferenciar de la interfaz de administración) y la Interfaz Gráfica de Administración. En la capa de Lógica se encuentran los Controladores, el Modelo y Utilidades. La capa de Acceso a Datos está compuesta por un único componente.

Interfaz Gráfica

Es uno de los puntos de entrada. Es la interfaz entre el usuario y la aplicación. Permite interactuar con el sistema mediante acciones del usuario incluyendo ejecutar el matching de órganos. Las acciones del usuario disparan invocaciones a los Controladores para completar las tareas correspondientes. Su arquitectura interna es SPA tal como se ve en la Figura 4.3.

Interfaz Gráfica de Administración

Es la interfaz que permite configurar el sistema.

Permite crear, editar, y eliminar entidades, atributos, metafunciones, políticas, asociación de atributos, matchings, referencias científicas y textos dinámicos (ver 5.4 Internacionalización).

Modelo

Representa el universo del sistema, la realidad. Para algunas entidades, contiene metadatos que agregan información importante.

Controladores

Los controladores son los encargados de la lógica de las vistas. Resuelven las peticiones del usuario y exponen las funcionalidades que brindan las vistas al usuario.

Si los eventos disparados por el usuario tienen que ver con lógica de aplicación, delega la tarea al componente Utilidades.

Utilidades

Tiene la responsabilidad de ejecutar la lógica de negocio.

Acceso a Datos

Todas las funcionalidades referentes al almacenamiento de los datos están encapsuladas en Acceso a Datos.

Está diseñada con el patrón UOW y Repositorio.

Arquitectura General

El estilo arquitectónico de Donamatch, presentado en Figura 4.5, es Cliente-Servidor en dos niveles (un servidor o varios servidores idénticos) de Cliente Fino MVC donde el procesamiento de la información y la gestión de los datos está a cargo del lado servidor, de tres capas de modelo estricto (una

capa solo utiliza servicios de la inmediata inferior) y está compuesta por las capas ya comentadas: Presentación, Lógica y Gestión de Datos.

Si en el futuro, ya sea por el uso masivo del sistema o algún otro aspecto aún no tenido en cuenta, se necesita mejorar la escalabilidad, se podrá pasar a una arquitectura de más niveles, con pequeñas modificaciones.

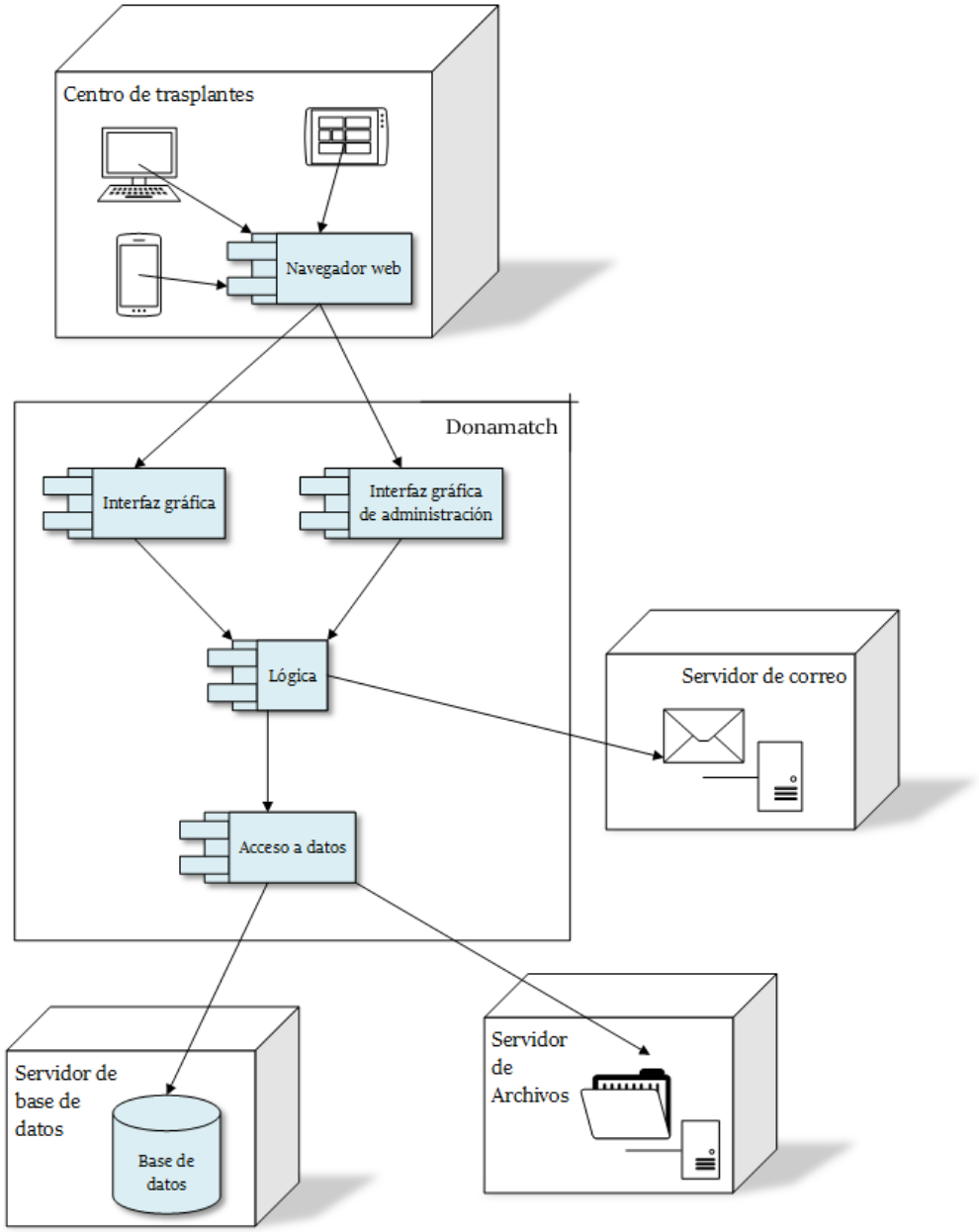


Figura 4.5 - *Arquitectura de Donamatch*

El sistema se accede a través del punto de entrada, desde un navegador web, ya sea por medio de una estación de trabajo, una tableta o un teléfono inteligente.

La capa de acceso a datos o gestión de datos, gestiona datos tanto en una base de datos como en un sistema de archivos. Se almacenan archivos generados como resultado de la asociación de órganos-receptores para cumplir con el requerimiento 3.3.5. Mientras que la Lógica es la encargada de envió de correos electrónicos comunicándose con un servidor de correos específico.

La arquitectura se ha diseñado buscando cumplir con los requisitos y además buscando tener propiedades como comprensión, escalabilidad, mantenibilidad, reutilización y portabilidad. La modularidad de la arquitectura está pensada principalmente para cumplir con el requisito 3.3.3. A continuación se abordará el tema seguridad con mayor profundidad.

Seguridad

Antes de avanzar con los aspectos de seguridad, aclararemos a qué nos referimos con el término.

Seguridad, en español, significa cualidad de estar libre y exento de todo peligro, daño o riesgo[2]. En el inglés tiene la misma definición la palabra *safety*[3]. Sin embargo cuando hablemos de seguridad también nos referiremos al estado de estar protegidos y no expuestos al peligro, libre de amenazas; en el inglés la palabra *security*[4] se aproxima a esta definición y es hartamente usada en ciencias de la computación para referirse por ejemplo a la protección de datos contra accesos no autorizados. A veces suelen usarse como sinónimos. *Safety* es usada en general al hablar de integridad de datos.

Dado que la información relevante del sistema se maneja en los Controladores, debido a la independencia de responsabilidades de MVC, y en Utilidades (y que las funcionalidades de éste se consumen desde Controladores), el enfoque de la seguridad está puesto en los Controladores, sin descuidar la interfaz gráfica ya que es el punto de entrada al sistema.

Si protegemos la información sensible, aumentaremos la seguridad.

Las grandes acciones que se tomarán, son, a saber:

- Autenticación de usuarios.

[2] Según la Real Academia Española.

[3] [4] Según el Diccionario de Oxford.

Se creará un mecanismo de registro de usuarios, e inicio de sesión. Un usuario podrá registrarse si posee una clave que deberá solicitar previamente al sistema. El sistema genera la clave y la envía por correo al administrador principal.

Este paso no automático permite que el administrador principal valide la identificación del actor que quiere registrarse. Luego debe hacerle llegar la clave al actor, ya sea por correo o personalmente. Una vez con la clave, el usuario podrá registrarse e iniciar sesión.

- Autorización de usuarios.
  - Definición de roles administrador y colaborador.

    Administrador es el rol que tiene todos los privilegios. Puede leer y escribir datos sensibles. Colaborador es el rol que solo puede leer datos sensibles.

- Autorización a nivel de controladores.

  Cualquier operación relacionada a información sensible a nivel de controladores no podrá realizarse si el usuario que la solicita no está autorizado.

- Matching como proceso de varios pasos.

  El principal uso del sistema es la asignación de un órgano a un receptor. Este proceso se guiará fuertemente y no se permitirá vuelta atrás en pasos críticos, así como se pedirá confirmación para ciertas acciones.

- Prevención de inyección.

  Mantener datos no confiables separados de los comandos y consultas [60].

- Prevención de pérdida de autenticación y gestión de sesiones.

  Poseer un único conjunto de controles de autenticación y gestión de sesiones fuerte [60]. No exponer datos importantes en direcciones URL ni soportar re-escritura, establecer correctos tiempos de expiración. Cifrar contraseñas.

- Prevenir Secuencia de Comandos Cruzados.

  Mantener los datos no confiables separados del contenido activo del navegador [60]. Utilizar API de JavaScript segura. Codificar y validar adecuadamente las entradas de datos ingresadas por el usuario.

- Prevenir Referencia directa insegura a objetos.

Usar referencias indirectas por usuario o sesión para objetos sensibles. Cuando se haga una referencia directa, comprobar el acceso para determinar si el usuario está autorizado a acceder al objeto solicitado.

- No exponer datos sensibles.
- Prevención de Falsificación de peticiones en sitio Cruzados (CSRF)

  Incluir un *token* no predecible en cada solicitud HTTP. Consta de los siguientes pasos.

  Generar un *token* anti-CSRF desde la vista invocando el método *AntiForgeryToken()*, si la solicitud HTTP ya contiene un *token*, se obtiene de ella, si no contiene o si su obtención falla, se genera aleatoriamente uno nuevo. Se genera además un campo anti-CSRF, usando el *token* y la identidad del usuario logueado.

  Luego se debe invocar desde el controlador el método *ValidateAntiForgeryToken*, que compara el *token* de la sesión con el enviado desde la vista, deben ser idénticos por como se han generado, además si el usuario está autenticado se compara el nombre de usuario con el almacenado en el token y deben ser iguales.

  Si la validación no tiene éxito se debe abortar la transacción pues puede haber existido un intento de falsificación de peticiones.

4.5 Análisis y Diseño

Para diseñar el sistema se tuvieron en cuenta los principios de Modularidad, Abstracción y Dividir y Conquistar. Y en general en la construcción del sistema debido a su naturaleza.

Por otra parte se buscó que el diseño siga buenas prácticas que aseguren la calidad del software como los patrones GRASP [61] y siempre con el objetivo de tener bajo acoplamiento y alta cohesión.

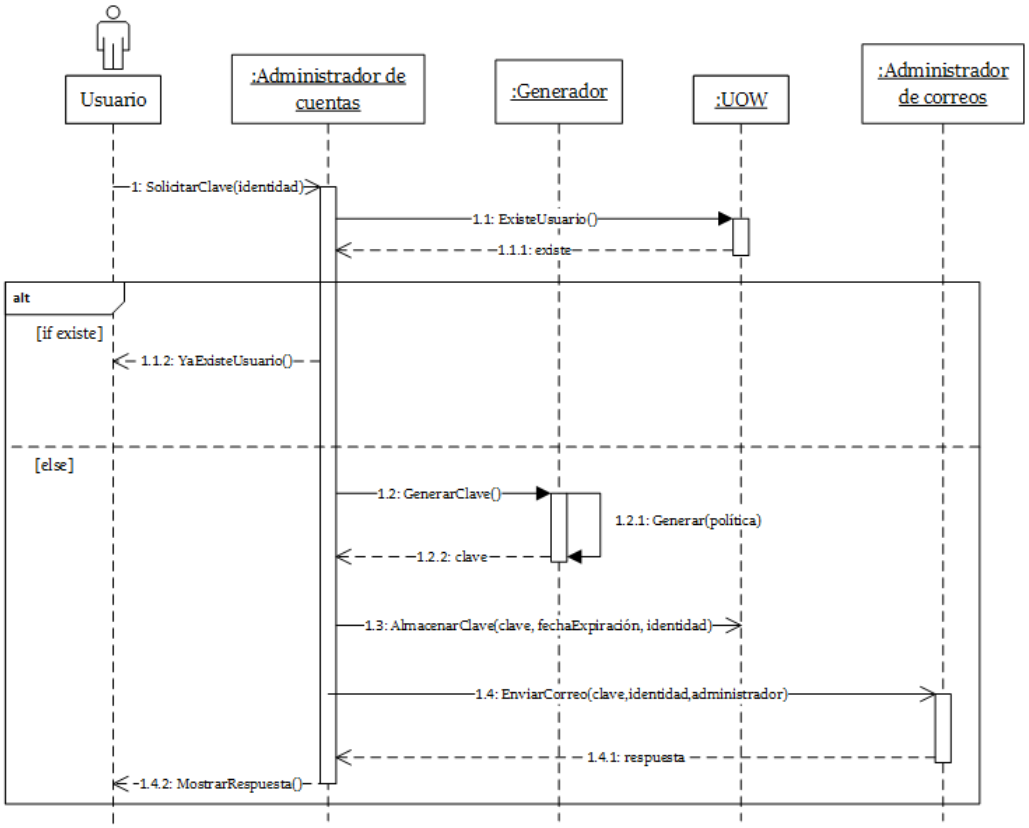


Figura 4.6 - *Diagrama de secuencia Solicitud de Clave*

El diseño, orientado a objetos, no fue completo. En el sentido de que la etapa de implementación no se trató únicamente de agregar detalles. Se diseñaron solamente las funcionalidades más importantes o complejas. El resto se diseñó informalmente o simplemente se pasó a la implementación.

La Figura 4.6 presenta un diagrama de secuencia para la solicitud de clave que permite al usuario registrarse.

El usuario mediante la vista de la interfaz gráfica (no plasmada en el diagrama), solicita una clave ingresando su nombre e identificación. El Administrador de cuentas delega la tarea de chequear que no exista el usuario a una instancia de UOW quien se encargará de validarlo en la base de datos. Al recibir la respuesta, en caso de existir el usuario, el Administrador de cuentas devuelve a la vista un mensaje para que ésta lo muestre. En caso contrario, invoca al generador de claves solicitando una generación, que se llevará a cabo con determinadas políticas configurables. Con la clave, el Administrador de cuentas solicita a la instancia de UOW que almacene la clave, una fecha de expiración configurable y el identificador del usuario. Luego el Administrador de cuentas invoca a la función que envía el correo electrónico del Administrador de correos, pasándole la clave, el identificador del usuario, y la dirección de correo, configurable, del administrador principal

del sistema. Luego, el Administrador de cuentas muestra al usuario, mediante la vista, una respuesta adecuada, indicando éxito o falla en la operación.

En la Figura 4.7, se muestra la primera parte del diagrama de secuencia de Matching. La funcionalidad más importante del sistema. El estado previo del sistema es:

- el usuario está autenticado y autorizado,
- el usuario eligió entre los matchings disponibles en el sistema, el que desea ejecutar, y el sistema muestra los datos necesarios para proceder con la operación.

El Usuario solicita la ejecución del matching enviando los datos correspondientes al matching en cuestión a la vista. La Vista verifica que los datos sean válidos. En el caso en que los datos no lo sean, las siguientes acciones se repiten hasta que los datos sean válidos:

1) se muestra un mensaje al usuario solicitando que corrija los datos.
2) el usuario vuelve a ingresarlos.
3) se valida los datos.

Luego, la Vista solicita la confirmación de la información enviada, debido a la importancia de esta acción ya que no existe marcha atrás en la búsqueda del mejor receptor para el órgano disponible. Aún no se ha asignado el órgano pero es importante que no exista error humano en el ingreso de la información pues hará que el proceso de matching no sea exitoso. El Usuario confirma los datos y si no son válidos se repite el proceso de validación de los datos como antes. A continuación, la Vista refina la información y le envía al Controlador de Matching las entidades y los atributos de entidad origen para que éste se encargue de la operación.

Si se llegó hasta aquí es porque existe un matching configurado para tales entidades y los datos son válidos pues se validaron en la Vista. Sin embargo, el Controlador valida que exista el matching entre las entidades y que los atributos sean válidos. Si no existe el matching o los atributos son inválidos, el Controlador le avisa a la Vista y ésta al Usuario. Y, por seguridad, se termina la operación debido a que ha ocurrido algún error.

Si se valida, el controlador delega la responsabilidad a Matching, quien se encargará de aplicar las funciones y asociaciones de atributos definidas en el sistema tal como se muestra en la Figura 4.8.

Para lograr el objetivo, la instancia de Matching solicita a la instancia de UOW, por un lado, las asociaciones para el matching definido por la entidad A y la entidad B, quien las obtiene a partir de los datos en la base de datos.

Por otro lado, le solicita los candidatos a ser emparejados con la Entidad A, de tipo Entidad B. UOW lo resuelve consultando en la base de datos.

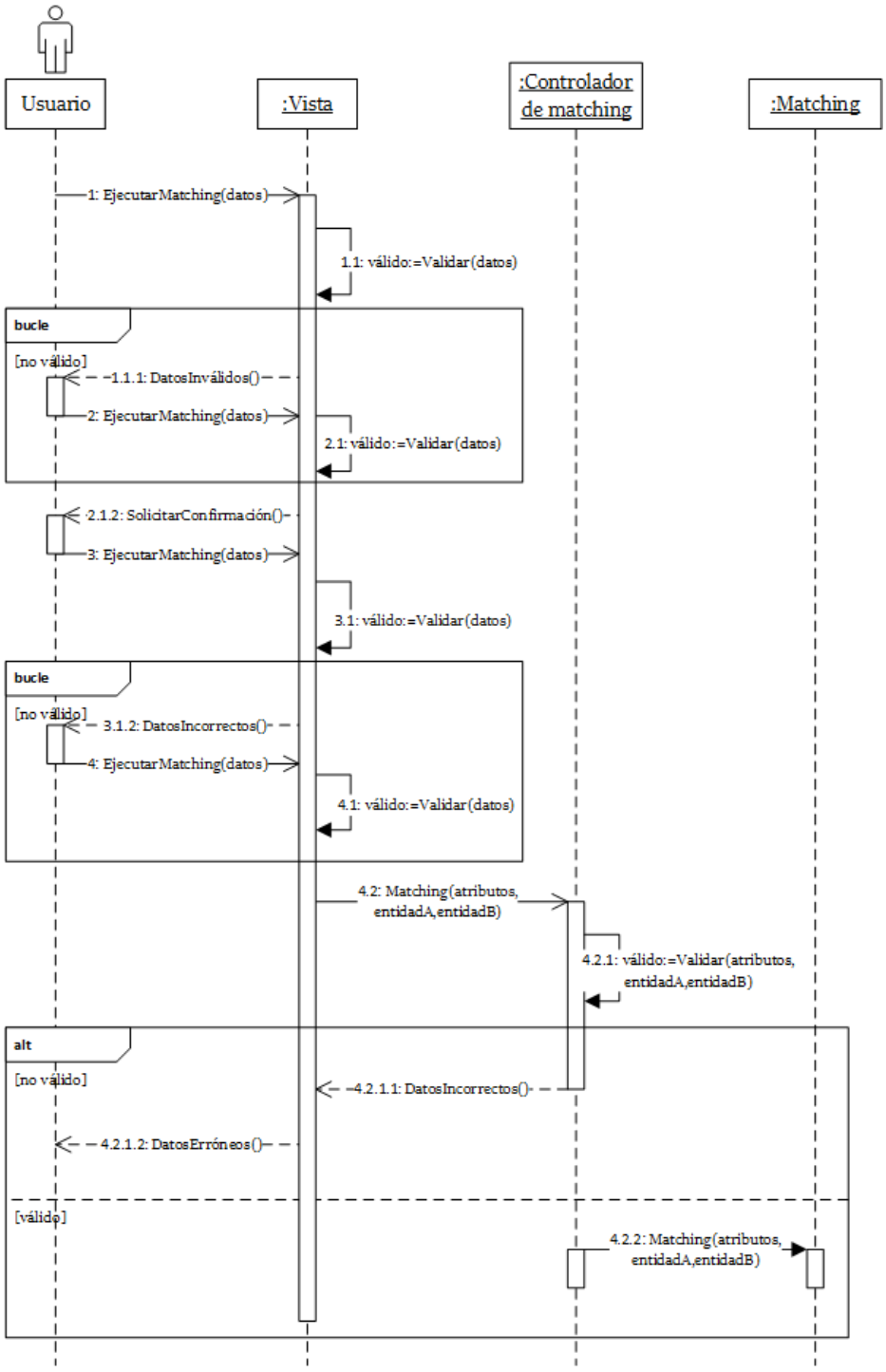


Figura 4.7 - *Diagrama de secuencia Matching parte 1*

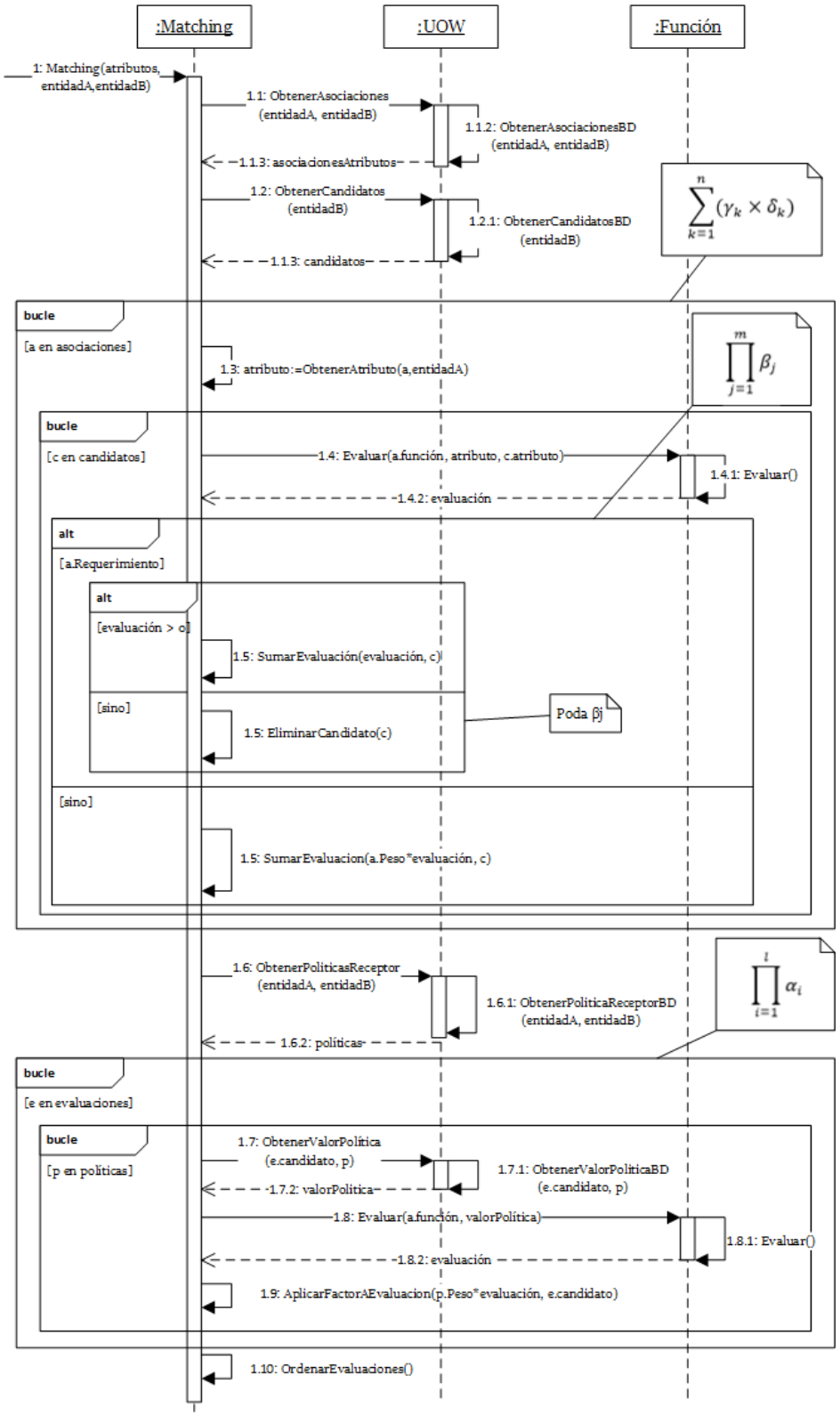


Figura 4.8 - *Diagrama de secuencia Matching parte 2*

Hasta el momento la instancia de Matching, dispone de las asociaciones de atributos configuradas para esas entidades, así como de los candidatos a participar de la competencia por aparearse con la Entidad A. Por lo tanto está en condiciones de calcular la sumatoria de (4.1). Para esto, ejecuta un bucle entre todos los candidatos para cada una de las asociaciones. En el cual, primero solicita a la instancia de manejo de funciones: Función que evalúe la función de la asociación de atributos actual según el valor del atributo correspondiente ingresado por el Usuario y el valor del atributo del candidato actual. Función evalúa la función a partir de estos datos y le devuelve el resultado a Matching. Quien distingue entre una evaluación positiva y no positiva. Por lo tanto aplicará la productoria del factor β_j de (4.1).

Si la evaluación es positiva, se suma la evaluación al candidato, en caso contrario se aplica una poda eliminando al candidato de la competencia. Pues significa que β_j es igual a cero. No tiene sentido seguir adelante en el cálculo del DM para el candidato ya que ya sabemos que dará cero.

Si la asociación de atributos no es un requerimiento, entonces la instancia Matching multiplica la evaluación por el peso de la asociación de atributos en el DM.

Luego de la secuencia 1.5, Matching ha calculado:

$$\prod_{j=1}^{m} \beta_j \times \sum_{k=1}^{n} (\gamma_k \times \delta_k) \tag{4.4}$$

Para calcular el resto de (4.1), solicita a UOW las políticas de receptor definidas para el matching. Después para cada una de las evaluaciones que ha obtenido hasta ahora, le multiplica el valor obtenido de aplicar la función definida en la política de todas las políticas. Divide y conquista: primero le solicita a UOW el valor de la política actual para el candidato de la evaluación actual. Luego de tener este valor, le solicita a Función que evalúe la función de la política según el valor. Luego, multiplica esta evaluación por el peso de la política actual y por último, al resultado, lo multiplica por (4.4).

Después simplemente ordena los CG_r obtenidos para pasárselos a la Vista en orden de mayor a menor según el puesto obtenido en la competencia.

La Figura 4.9 muestra la devolución en cascada hasta mostrarle los receptores al Usuario. En caso que en el primer lugar se encuentre más de un receptor, el Usuario deberá desempatar eligiendo a uno de los receptores en primer lugar. Si esto sucede o bien los criterios de asignación no son demasiado específicos, o bien el sistema están mal configurado; aunque pueden existir excepciones.

Luego de desempatar o bien dar por finalizado el matching, la Vista muestra al Usuario un resumen del matching realizado.

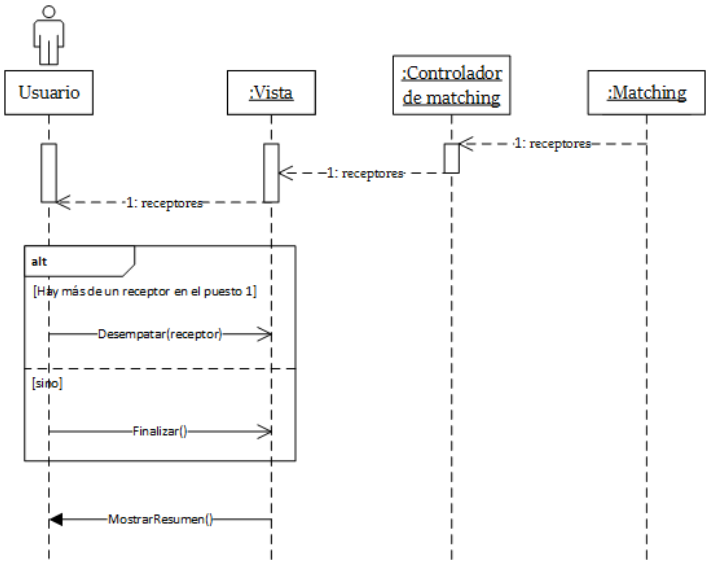


Figura 4.9 - *Diagrama de secuencia Matching parte 3*

Capítulo 5

IMPLEMENTACIÓN

En este capítulo se presenta detalladamente y con un enfoque de arriba hacia abajo la implementación de los componentes más relevantes del sistema.

Aplicación de MVVM

Una de las principales características de Donamatch debe ser la usabilidad. De nada sirve la aplicación y los órganos disponibles, si el matching es inusable, o poco amigable, o lento, por nombrar algunas.

Teniendo en cuenta que la experiencia de usuario debía ser buena, desde la parte gráfica se utilizaron tecnologías que lo facilitaban, casi que sin importar el costo que traían asociado.

Se implementó una interfaz que presenta una fluida interacción con el usuario. Como biblioteca, se usó Knockout [62-63] (ver Figura 4.3), que permitió clara y fácilmente asociar elementos del Modelo de Objetos del Documento (traducción libre de *Document Object Model*, DOM) HTML, con los datos del modelo; configurar relaciones entre datos del modelo para poder combinarlos y por sobre todas los beneficios reflejar automáticamente los cambios de estado del modelo en la interfaz de usuario, dándole una característica de tiempo real, mostrando un excelente tiempo de interacción.

Abro paréntesis: Donamatch es un sistema (el software, el hardware, la información y el mecanismo de emergencia que se encarga de desde mantener sano el órgano hasta trasplantarlo) de tiempo real, si la salida (el trasplante) del sistema no se da en un determinado período de tiempo, el sistema fracasa, su funcionamiento no es correcto. De ahí la importancia de que el tiempo de interacción y respuesta del sistema sea muy bueno.

¿Por qué *Knockout* en vez de otras bibliotecas como *Angular* [64] o *Backbone* [65]? Una discusión de este estilo puede motivar un trabajo por sí solo. No es el objetivo del presente, sin embargo se expondrán algunos conceptos.

Para la elección se consideraron únicamente las bibliotecas más populares y con casos de éxito que las avalen. *Backbone*, si bien beneficia la experiencia de usuario no sigue ni el patrón MVC ni el MVVM planteado en la arquitectura, así que fue fácilmente descartada.

Para viabilizar el uso de *Knockout* o *Angular*, se hizo una prueba de concepto que buscaba estimar la curva de aprendizaje de cada biblioteca, con el objetivo de usar eficientemente el tiempo del proyecto.

La funcionalidad a desarrollar era una vista con dos cuadros de textos y un botón; al ingresar nombre y apellido, se debía desplegar instantáneamente un texto mostrando el resultado de concatenar el nombre y el apellido y la cantidad de vocales y consonantes. Para las pruebas se usó la herramienta *JsFiddle* [66], como en gran parte del desarrollo ya que permite probar pequeñas funcionalidades, detectar defectos (por permitir aislar fragmentos de código), mitigar riesgos, etc.

La cantidad aprendida con *Knockout* creció más rápidamente que con *Angular* para la prueba de concepto tal como lo muestran la Figura 5.1 y la Figura 5.2. Esto determinó que *Knockout* sea la biblioteca seleccionada. Y permitió vislumbrar la evolución del aprendizaje que se tendría usando *Knockout*.

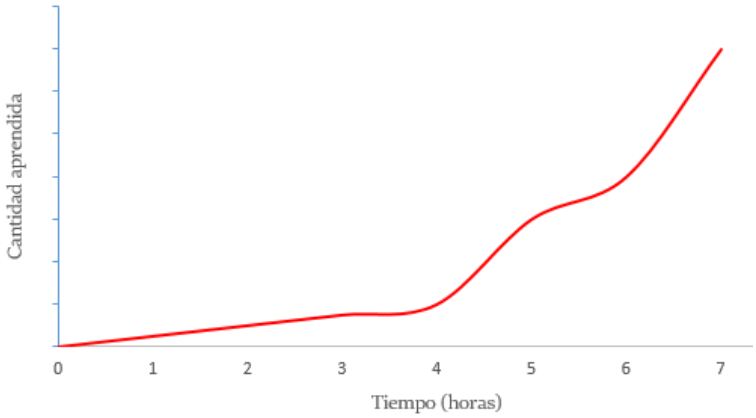


Figura 5.1 - *Curva de aprendizaje en prueba de concepto con Angular.*

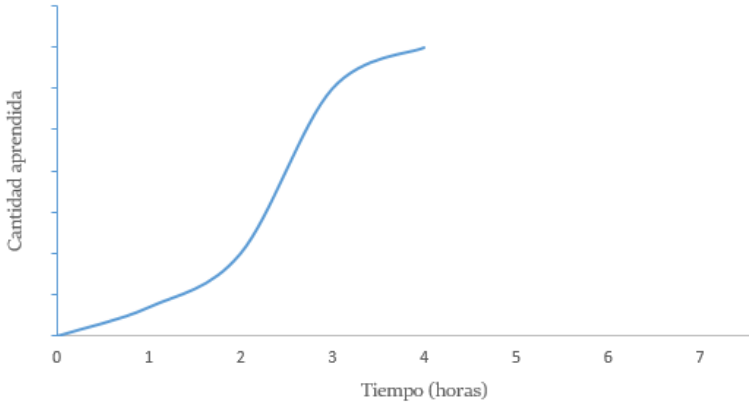


Figura 5.2 - *Curva de aprendizaje en prueba de concepto con Knockout.*

No obstante, la curva de aprendizaje real fue diferente, tal como se ve en la Figura 5.3. El aprendizaje refiere a todo lo aplicado en Donamatch, que incluye casi todos (90% aproximadamente) los conceptos de *Knockout*: *observables*, *bindings*, *control flows*, etc.

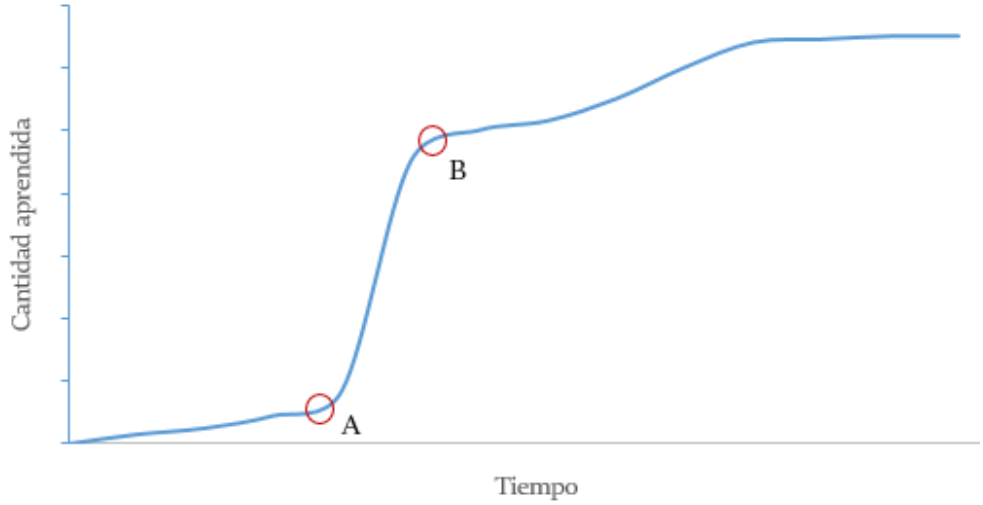


Figura 5.3 - *Curva de aprendizaje de Knockout en Donamatch.*

El estudio de ambas bibliotecas permitió concluir que *Angular* parece ser más poderoso que *Knockout*, pero al usar fuertemente por ejemplo inyecciones de dependencia, se tiene la sensación que las cosas suceden por arte de magia. Cuesta mucho, al menos al principio, entender cómo funciona *Angular*. En cambio con *Knockout* los conceptos básicos son más fáciles de entender, pero cuando se necesita construir estructuras más complejas la falta de documentación hace que cueste mucho encontrar soluciones y discernir sobre cuál es la mejor, en la Figura 5.3 se ve reflejado en la evolución desde B hasta la primera tangente horizontal. Al principio cuesta bastante entender cómo funciona *Knockout* y sus conceptos básicos (hasta A), pero una vez aprendidos se avanza muy rápido como lo muestra el lapso entre A y B.

En la Tabla 5.1 se describen brevemente los Modelos de Vista que se implementaron en Donamatch. En las siguientes secciones algunos se explicarán con mayor detalle.

Pero los modelos de vista no hacen el trabajo solos, hay otros elementos que contribuyen a la separación de la presentación y la lógica, como es el caso del modelo de datos que se implementó: *AppDataModel*, que es quien principalmente se encarga de hacer las invocaciones a los controladores (operaciones de acceso a los datos), conoce los ruteos que llevan a los métodos de los controladores y contiene algunas operaciones auxiliares para ello.

Tabla 5.1 – *Modelos de Vista de Donamatch*

ViewModel	**Descripción**
AppViewModel	Es el modelo de vista general de la aplicación, administra a los demás modelos de vista, encapsula algunas funcionalidades generales y administra el flujo entre secciones de pantallas.
HomeViewModel	Es el más importante ya que se encarga de la presentación de los matchings disponibles, de la selección y de la ejecución del matching. Se encarga además, de la presentación de los resultados obtenidos. Hace todas las validaciones referentes al matching, etc.
LoginViewModel	Es el que administra el inicio de sesión.
ManageViewModel	Implementa las funcionalidades de administrar la cuenta de usuario.
RegisterViewModel	Es el que gestiona el registro de usuarios.
RequestKeyViewModel	Es el que gestiona la solicitud de clave.
UserInfoViewModel	Se encarga de administrar los datos del usuario.

En Donamatch los modelos de vista tienen, en general, la estructura que se muestra en la Tabla 5.2. Los datos son los que se manejan en el modelo pero que se presentarán en la vista o se usarán en la presentación de otros datos, o bien se usarán en invocaciones a métodos de los controladores que desencadenarán como resultado, mostrar información en la vista. A menudo son datos "puros" pero a veces también calculados a partir de otros.

Tabla 5.2 – *Tipos de datos de Knockout en Donamatch*

Sección	**Tipo en *Knockout* o *JavaScript***
Datos	*observable, observable arrays, extenders*
Estado privado	*computed observables*
Estado de IU	*observable, observable arrays, computed observables*
Operaciones privadas	*function*
Operaciones de IU	*function*

El estado privado es el estado del modelo de vista. Mientras que el estado de Interfaz de Usuario es el estado de la vista. La vista o está cargando alguna funcionalidad para dejarla disponible al usuario, o está esperando por alguna solicitud del usuario, o ejecutando alguna acción del sistema o mostrando mensajes de éxito o de error. El flujo entre los estados por los que pasa la IU se puede observar en el diagrama de la Figura 5.4.

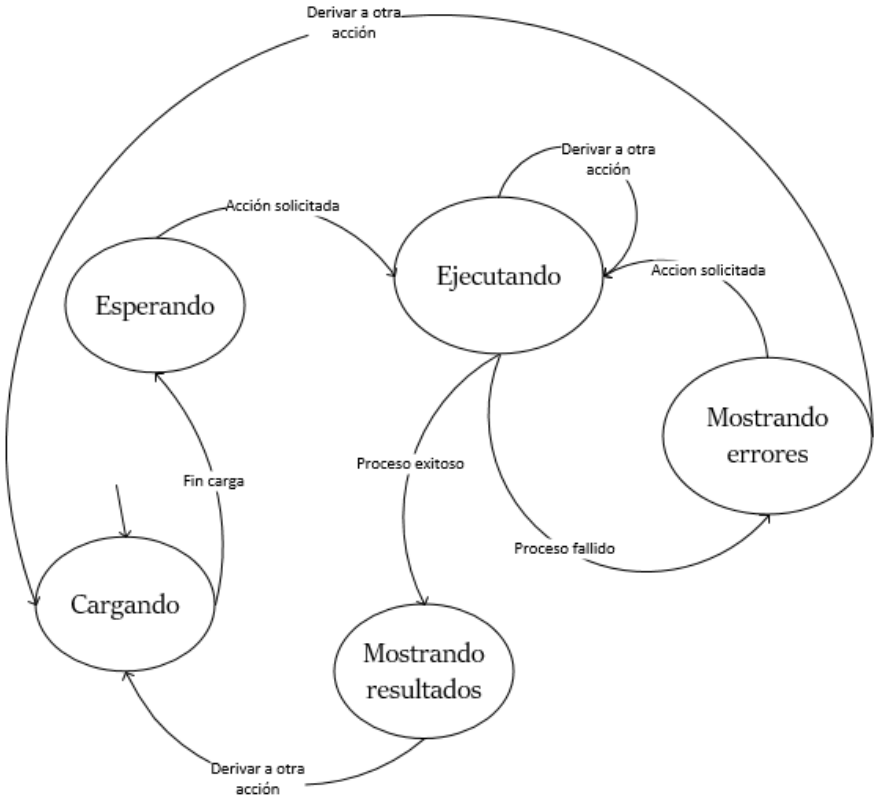


Figura 5.4 - *Diagrama de estados de la Interfaz de Usuario de Donamatch*

Las operaciones de Interfaz de Usuario son las operaciones que ofrece el sistema. Muchas veces en estas operaciones se ejecutan invocaciones a los controladores. Cuando se estén ejecutando estas operaciones, la IU estará en el estado Ejecutando (Figura 5.4).

Las operaciones privadas, son aquellos procedimientos internos al modelo de la vista que contribuyen a la gestión de los datos, estados y operaciones de IU.

5.1 Seguridad

Los aspectos de seguridad implementados son los descriptos en la sección 4.4.

Registro de usuarios

Para implementar el registro de usuarios se creó el *RegisterViewModel*, que tiene como datos:

- *key*
- *userName*
- *password*
- *confirmPassword*
- *identity*

Key es la clave Donamatch que se solicita mediante una vista que presenta los datos *email*, *name* e *identity* de *RequestKeyViewModel*. Luego de tener los datos y la confirmación del usuario, el MV invoca a la operación *RequestKey* de *AccountController* quien se encarga de generar la clave y enviar el correo electrónico con el nombre, la dirección de correo electrónico e identidad del solicitante, al administrador del sistema (la dirección del correo electrónico de administrador es configurable, ver Sección 5.6 Configuración) para que pueda verificar la identidad y enviarle la clave. Implementa el diagrama de secuencia de la Figura 4.6.

Cuando el usuario ya posee la clave, puede registrarse ingresándola junto a su nombre de usuario, contraseña e identidad. Se invoca la operación *register* de *RegisterViewModel*, el MV compara las contraseñas y en caso de ser iguales y que cumplan las políticas de contraseñas del sistema, invoca asincrónicamente a *Register* de *AccountController* quien 1) valida el modelo, 2) obtiene la última clave solicitada por el usuario que no haya expirado ni haya sido usada y la compara con la provista por el usuario, 3) crea la cuenta encriptando la clave, y 4) devuelve al MV un resultado acorde a según haya sido exitoso o no. Al recibir la respuesta inicia la sesión del usuario mediante *AppDataModel* o muestra un mensaje de error apropiado.

El manejo de claves, tokens y roles se basó en *AspNet.Identity*.

Sesión de usuario

De la sesión del usuario se encarga *LoginViewModel*. Presenta en la vista de usuario, dos campos para ingresar el nombre de usuario y la contraseña y opcionalmente una opción de ser recordado. Únicamente luego de que ambos campos se hayan completado, invoca a *AppDataModel* quien delega la tarea de chequear que la combinación contraseña y nombre de usuario sean correctos a *AccountController*. Si *AppDataModel* le devuelve un *accessToken* (significa que la contraseña y usuarios son correctos), *LoginViewModel* invoca a *navigateToLoggedIn* de *AppDataModel* donde 1) se asigna al modelo de datos el *accessToken* para usar en validaciones futuras, 2) se crea una instancia de *UserInfoViewModel* a partir del modelo de datos y el nombre del usuario, 3) se cambia el estado de la vista para navegar al inicio, y al mismo tiempo 4)

se invoca a *UserIsAdministrator* de *HomeController* quien verifica si el usuario está entre los administradores del sistema. Cuando responde, únicamente si es un administrador se mostrará en la vista de inicio la opción de administrar el sistema. Los roles del sistema son los descriptos en 4.4. En la Figura 5.5 se muestra el modelo de datos de identificación en Donamatch.

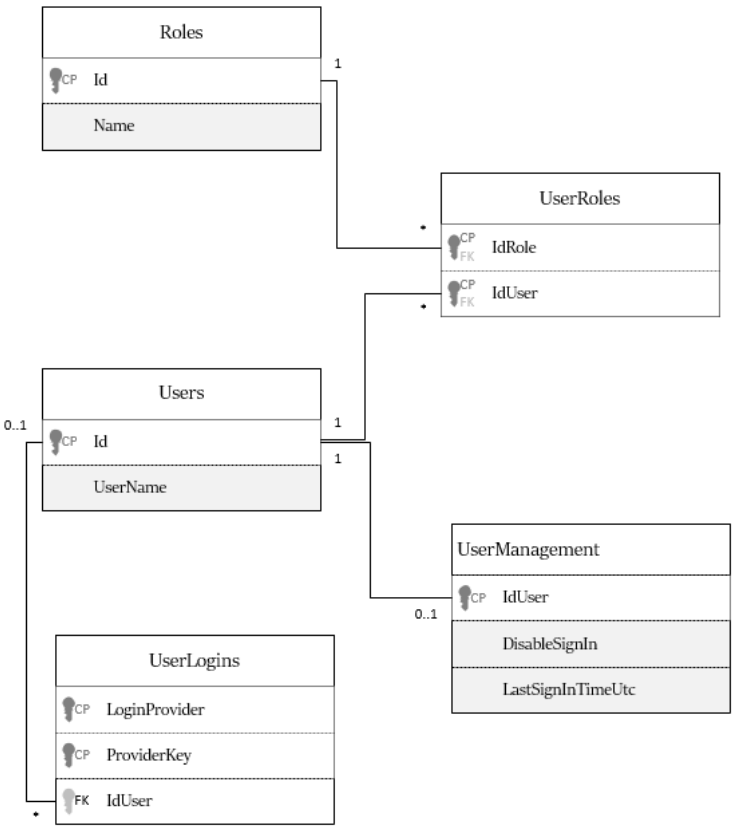


Figura 5.5 - *Modelo de datos de Identificación en Donamatch*

Si el usuario elige ser recordado por el sistema, no necesitará iniciar sesión nuevamente. Incluso cerrando el navegador. Pero si cierra sesión, el sistema "olvidará" la identidad del usuario.

Esta funcionalidad puede ser una vulnerabilidad del sistema, si su uso es irresponsable. No se quitó, pues también es un beneficio no tener que iniciar sesión cada vez, pues permite mayor agilidad en el uso del sistema. Es ideal para equipos privados. En equipos públicos siempre se debe cerrar sesión. Como solución de compromiso, se cierra la sesión después de determinado período de tiempo.

Si se desea agregar un inicio de sesión por medio de un proveedor externo, se puede incorporar fácilmente al sistema. Por ejemplo si el centro de trasplantes posee un sistema con manejo de sesión podrá incorporarse a Donamatch.

Ingreso de datos

Para el ingreso de datos de cualquier tipo se procede de la siguiente forma:

1. Se valida en el modelo de vista
2. Se completa la validación en el Controlador (a veces se repite la misma validación que en la Vista)
3. Si se detecta cualquier combinación de datos no prevista en la ejecución de algún método de algún Controlador, se devuelve un Error 400 *BadRequest.*

Para evitar un posible ataque aprovechando la confianza del sitio en el usuario, en cada vista (que corresponda), se genera una *AntiForgeryToken* y una *Cookie* (cada vez que se cargue la vista se generará un par diferente), se comprueba luego si el par fue generado en el mismo servidor, con la anotación *ValidateAntiForgeryToken* en los controladores.

Controladores

Los controladores, a nivel de clase, poseen una anotación que autoriza a usuarios únicamente con sesión activa en el sistema y que estén tengan determinado rol, acceder a los métodos públicos. Notar que siempre que se desencadene un proceso desde la interfaz de usuario, que concluya con el acceso a un método de un controlador, el usuario deberá tener sesión activa y tener el rol debido pues el sistema está diseñado para ello. La única excepción es un intento de ataque, por tal motivo existe este control.

5.2 Administración

La sección de administración de Donamatch es visualmente drásticamente diferente al resto del sitio. No es ni SPA ni MVVM. Podríamos decir que es un sitio MVC a parte.

La etapa de desarrollo para la administración, consistió básicamente en implementar los conceptos (que llamaremos elementos) definidos en 4.3. Y permitir ejecutar altas, bajas, consultas y modificaciones sobre los conceptos.

Se implementaron ocho controladores que se encargan de la lógica de la administración:

- *AdministratorController*
- *EntitiesController*
- *AttributesController*
- *FunctionsController*
- *ComplexFunctionsController*
- *MatchingController*

- *PoliciesController*
- *LanguageController*

La Tabla 5.3, muestra las vistas que se implementaron en la administración para cada controlador.

Tabla 5.3 – *Vistas de la sección Administración*

Vistas	**Controlador**
Index	*Administrator*
Create, Delete, Details, Edit, Index, Language	*Entities*
Create, Delete, Details, Edit, Index, Language	*Attributes*
Create, Delete, Details, Edit, Index	*Functions*
Create, Delete, Details, Edit, Index, Language	*ComplexFunctions*
Create, CreatePolicy, Define, Delete, DeleteDefinition, DeletePolicy, Details, Edit, EditDefinition, EditPolicy, Index, Language, LanguagePolicy, UploadReference	*Matching*
Create, Delete, Edit, Index, Language	*Policies*
EditAttribute, EditComplexFunction, EditEntitie, EditMatching, Index	*Language*

La vista, *_LayoutAdministrator*, es el contenedor de las demás y quien le da el aspecto general. Se implementaron cuatro grandes secciones: Entidades, Funciones, Matchings y Traducciones. Que agrupa en familias a los elementos del sistema. Y la filosofía de diseño en cada sección es la misma: se muestra una lista de los elementos existentes, y las funcionalidades disponibles para tal elemento: crear, editar, consultar y eliminar.

La implementación de las operaciones también es análoga para los elementos. El alta de conceptos, en general sigue la estructura de los Algoritmos 1 y 2.

Algoritmo 1. Iniciar vista crear.

```
1: Obtener datos para mostrar
2: Crear vista
3: fin
```

En el primer algoritmo, se prepara la vista. El segundo recibe los datos de la vista para crear el elemento.

Algoritmo 2. Crear elemento. Recibe un elemento desde la vista.

```
1: si Modelo es válido
2:    Crear traducciones por defecto
3:    Guardar elemento en el sistema
4:    Guardar registro del evento
5:    Direccionar a otra vista
6: sino
7:    Direccionar a la vista del elemento
8: fin
```

El modelo de datos de los conceptos son los que se presentan en la Figura 5.6.

La tabla ENTITIES, posee los siguientes atributos:

- Id: identificador único de la entidad.
- Name: nombre de la entidad en el idioma actual.
- Description: descripción de la entidad en el idioma actual.

Por idioma actual, se entiende al idioma de Donamatch al momento de realizar la acción (alta o modificación).

La tabla ATTRIBUTES, que representa a los atributos de las entidades, es una entidad débil que posee los siguientes atributos:

- Id: identificador único del atributo dentro de una entidad.
- IdEntity: identificador de la entidad a la que pertenece el atributo.
- Name: nombre del atributo de la entidad.
- Description: descripción del atributo de la entidad.
- IdUnit: identificador de la unidad de medida del atributo, para el caso en que sea mensurable.
- Measurable: indicador de si el atributo es mensurable o no.

La tabla UNITS, representa a las unidades de medidas manejadas. Presenta los atributos:

- Id: identificador de la unidad.
- Name: Nombre que se mostrará al usuario.

Symbol: Símbolo que representa a la unidad.

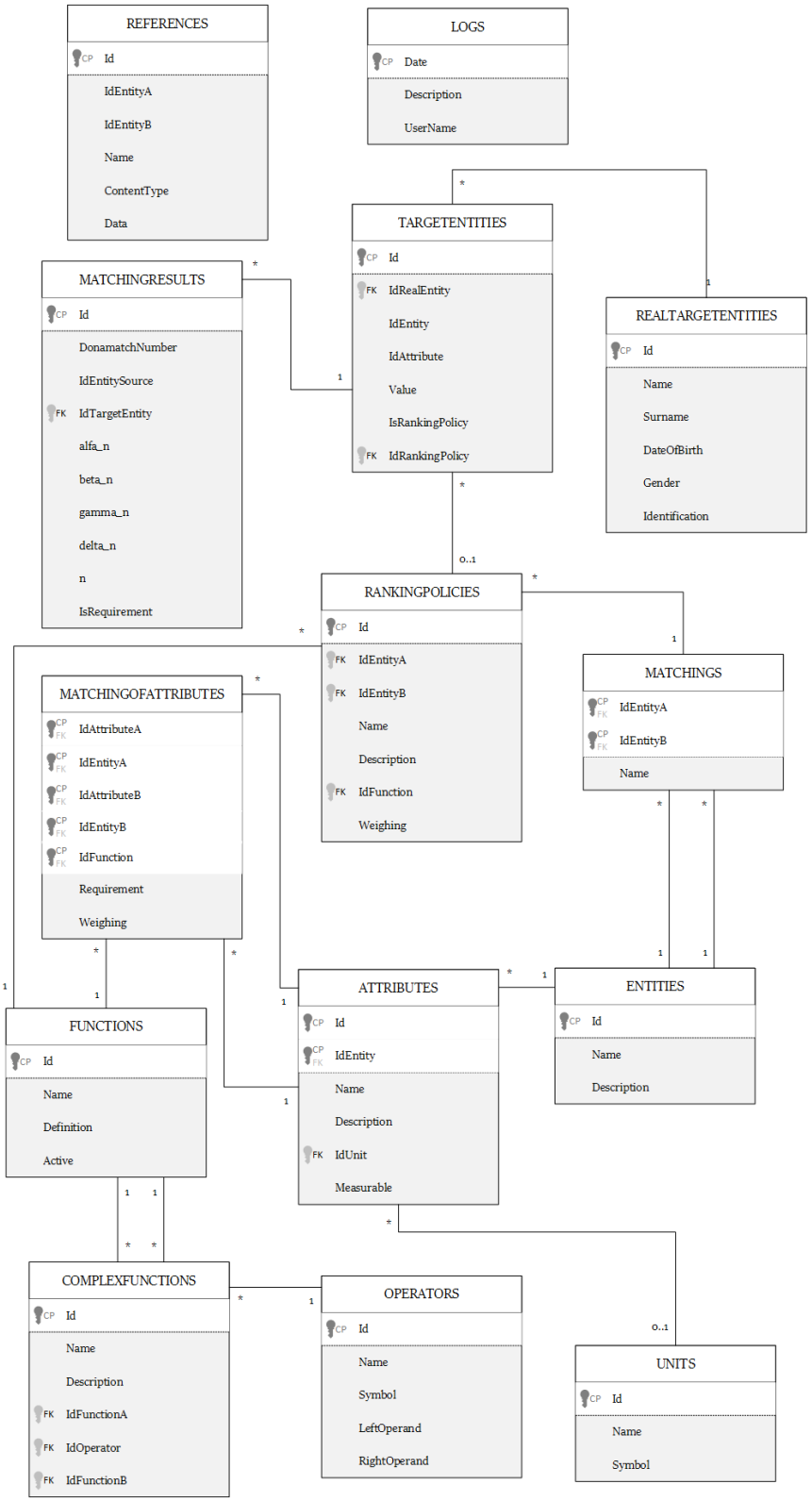


Figura 5.6 - *Modelo de datos de Donamatch*

La tabla FUNCTIONS, que contiene a las funciones, posee los atributos:

- Id: identificador de la función.
- Name: nombre de la función.
- Definition: definición matemática de la función.
- Active: indica si está activa en el sistema.

La tabla COMPLEXFUNCTIONS, persiste a las metafunciones que se crean en la administración y posee los siguientes atributos:

- Id: identificador de la metafunción.
- Name: nombre de la metafunción.
- Description: descripción de la metafunción.
- IdFunctionA: identificador de la función del dominio de la metafunción.
- IdOperator: identificador del operador lógico que será la aplicación de la metafunción.
- IdFunctionB: identificador de la función del codominio de la metafunción.

La tabla OPERATORS, que posee a los operadores lógicos que se usan para crear metafunciones como en la administración, tiene los siguientes atributos:

- Id: identificador del operador.
- Name: Nombre del operador.
- Symbol: Símbolo que representa al operador.
- LeftOperand: Indicador de que el operador admite un operando a la izquierda.
- RightOperand: Indicador de que el operador admite un operando a la derecha.

Notar que si LeftOperand y RightOperand son verdaderos admite operandos a izquierda y derecha.

La tabla MATCHINGOFATTRIBUTES, persiste a las asociaciones de atributos y contiene las siguientes propiedades:

- IdAttributeA: identificador del atributo de la entidad A de la asociación.
- IdEntityA: identificador de la entidad A de la asociación.
- IdAttributeB: identificador del atributo de la entidad B de la asociación.
- IdEntityB: identificador de la entidad B de la asociación.
- IdFunction: identificador de la función que asocia a los atributos.
- Requirement: indica si la asociación es un requisito para el matching entra las entidades A y B.

- Weighing: en el caso de que la asociación no sea un requisito para el matching, indica el peso que tiene en la asociación.

La tabla RANKINGPOLICIES, persiste a las políticas de receptor y contiene los siguientes atributos:

- Id: identificador de la política.
- IdEntityA: identificador de la entidad A del matching en el que aplica la política.
- IdEntityB: identificador de la entidad B del matching en el que aplica la política.
- Name: nombre de la política.
- Description: descripción de la política.
- IdFunction: identificador de la función que se aplica a la entidad A y B.
- Weighing: peso de la política.

La tabla MATCHING, almacena los matching existentes en el sistema. Tiene los atributos:

- IdEntityA: identificador de la entidad A del matching.
- IdEntityB: identificador de la entidad B del matching.
- Name: nombre del matching.

La tabla TARGETENTITIES, representa a las entidades destino del matching, contiene los valores tanto para políticas como para las demás asociaciones y posee los siguientes atributos:

- Id: identificador entidad destino.
- IdRealEntity: identificador de la entidad destino real (el receptor).
- IdEntity: identificador de la entidad que representa.
- IdAttribute: identificador del atributo.
- Value: valor del atributo.
- IsRankingPolicy: indica si es el valor corresponde a una política de receptor.
- IdRankingPolicy: en el caso de que sea un valor para una política de receptor, tiene el identificador a la política.

La tabla REALTARGETENTITIES, representa a las entidades destino reales, los candidatos a receptores y posee los siguientes atributos:

- Id: identificador interno de la persona.
- Name: nombre de la persona.
- Surname: apellido de la persona.
- DateOfBirth: fecha de nacimiento de la persona.
- Gender: Género de la persona.

- Identification: identificador de la persona (cédula de identidad en el caso de Uruguay).

Las tablas TARGETENTITIES y REALTARGETENTITIES, modelan los candidatos a receptores en lista de espera y los valores de sus atributos.

La tabla MATCHINGRESULTS, persiste los resultados parciales de cada matching, tanto del receptor elegido como de los candidatos a receptor. Tiene los siguientes atributos:

- Id: identificador interno.
- DonamatchNumber: identificador del matching.
- IdEntitySource: identificador de la entidad origen en el matching.
- IdTargetEntity: identificador de la entidad destino del matching.
- alfa_n: valor de α_n obtenido en el matching.
- beta_n: valor de β_n obtenido en el matching.
- gamma_n: valor de γ_n obtenido en el matching.
- delta_n: valor de δ_n obtenido en el matching.
- n: valor del índice de la sumatoria o productoria del DM.
- IsRequirement: indica si el matching de atributos es un requerimiento para el matching de las entidades origen y destino.

La tabla REFERENCES, persiste las referencias del matching, que lo justifican. Contiene:

- Id: identificador interno.
- IdEntityA: identificador de la entidad A.
- IdEntityB: identificador de la entidad B.
- Name: nombre de la referencia.
- ContentType: tipo del archivo.
- Data: el archivo en bits.

IdEntityA y IdEntityB, son los identificadores del matching.

La tabla LOGS, persiste todas las acciones del sistema. Registra:

- Date: fecha y hora del registro.
- Description: descripción de la acción del sistema.
- UserName: nombre del usuario que realizó la acción.

5.3 Matching

El corazón de Donamatch es el matching. El principal caso de uso de la aplicación es la asociación de órganos con receptores.

La Figura 5.7 muestra los estados por los que pasa el matching en Donamatch.

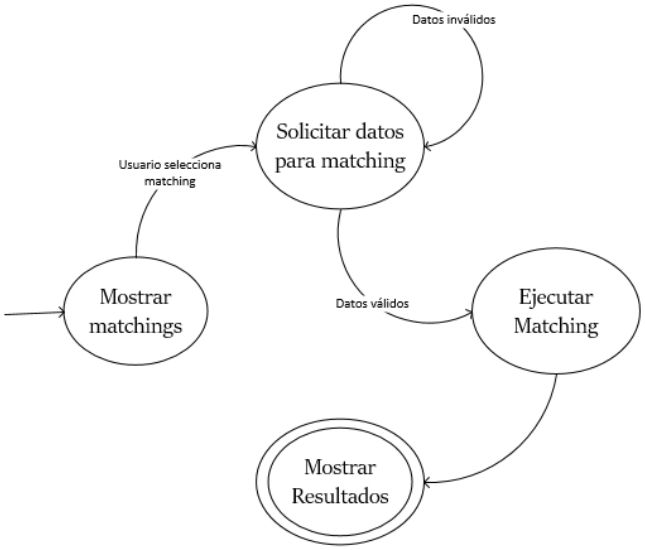


Figura 5.7 - *Estados de Donamatch*

Cuando el usuario ya ha iniciado sesión, el sistema muestra los matchings disponibles, esto es, los que se hayan configurado. Cuando el usuario selecciona uno, el sistema muestra los datos necesarios para proceder. Si los datos son válidos se pasa a ejecutar el matching con los datos ingresados y los datos de los pacientes en lista de espera. El proceso finaliza cuando se muestran los resultados obtenidos. Sin importar si se hayan encontrado o no receptores el proceso finaliza, mostrando la información pertinente. El mismo usuario puede iniciar nuevamente el proceso.

El modelo de vista que administra el proceso de matching es *HomeViewModel*, que presenta la estructura que se muestra en la Tabla 5.4. Las colecciones de tipo *observableArray* tienen elementos que son *observables*.

En el estado *Mostrar matchings*, el sistema presenta para cada elemento de la colección matchings su nombre en la vista *_SectionSelectMatching*. La transición a *Solicitar datos para matching*, se da mediante la función *selectMatching* quien, mediante el uso de *indexesForAttributes* se encarga de dibujar en la vista *_SectionMatching*, los atributos para la entidad origen del matching elegido por el usuario (la función recibe como parámetro el observable *matching*). Para cada atributo, se muestra el nombre y un campo de texto para que el usuario ingrese el valor del órgano donado.

Las transiciones que salen del estado *Solicitar datos para matching*, se disparan con la función *sendEntity* que con *selectedDonorAttributes* y

matching, tiene como objetivo enviar la información a *HomeController* para que haga efectivo el matching.

Tabla 5.4 – *Detalle del modelo de vista HomeViewModel*

Nombre	Tipo	Sección
matchings	*observableArray*	Datos
attributes	*observableArray*	Datos
indexesForAttributes	*observableArray*	Datos
recipients	*observableArray*	Datos
matchingSuccess	*observable*	Estado IU
matched	*observable*	Estado IU
dataEntered	*observable*	Estado
dataIsInvalid	*observable*	Estado IU
matchOnlyOne	*observable*	Estado IU
matchResult	*observable*	Datos
selectedRecipient	*observable*	Datos
selectedDonorAttributes	*observableArray*	Datos
tableResumeElements	*observableArray*	Datos
tableResumeOk	*observable*	Estado IU
sendEntitie	*function*	Operación IU
validateEntity	*function*	Operación
selectMatching	*function*	Operación IU
changePage	*function*	Operación IU
selectRecipient	*function*	Operación IU
generateCDA	*function*	Operación IU

El Algoritmo 3 muestra un pseudocódigo de la función *sendEntity*. Lo primero que hace es chequear la validez de los datos ingresados por el usuario de Donamatch, si son inválidos, avisa a *_SectionMatching* para que lo refleje al usuario (el bucle de la Figura 5.7). Mientras el usuario siga invocando a la función con datos inválidos, el matching se mantendrá en el mismo estado. Esto es lo que se observa además en el diagrama de secuencias de la Figura 4.7 en los pasos 1 y 2.

Si los datos son válidos, luego de avisar a la vista envía los atributos y los identificadores de las entidades que definen el matching a *HomeController* (pasa al estado *Ejecutar Matching)*.

Algoritmo 3. Ejecutar Matching.

```
1: si los datos de los atributos son válidos
2:    avisar a la vista que los datos son válidos
3:    si es la segunda vez que se ingresan datos válidos
4:       avisar a la vista que es la segunda vez
5:       para cada atributo
6:          si el atributo es de entidad origen
7:             agregar atributo a atributos de entidad origen
8:       enviar asincrónicamente atributos e identificadores
                              de entidad origen y destino
9:       si se encuentra un solo receptor
10:         avisar a la vista que hay un solo receptor
11:      para cada receptor
12:         agregar receptor a receptores según la posición
13:      si hay al menos un receptor
14:         avisar a la vista que el matching ha sido exitoso
15:      avisar a la vista que puede mostrar los resultados
16:   sino
17:      avisar a la vista que es la primera vez
18: sino
19:    avisar a la vista que los datos son inválidos
20: fin
```

Cuando recibe la respuesta (línea nueve del pseudocódigo), prepara los receptores para mostrarlos en orden en la vista o avisa de los casos de borde para que se muestren adecuadamente. En este momento pasa al estado *Mostrar Resultados* de la máquina de estados de la Figura 5.7.

SendEntity es la implementación de *EjecutarMatching* del diseño reflejado en el diagrama de secuencia de la Figura 4.7 y 4.9. En el diagrama pueden observarse todas las interacciones entre el usuario y *sendEntity*, y *sendEntity* y *HomeController*.

Cuando *sendEntitie* agrega a todo *recipient* en *recipients* y avisa a *_SectionMatching*, la vista despliega la lista ordenada de los receptores mostrando:

- Result. Resultado obtenido en el matching.
- Position. Posición en el ranking.

- Name. Nombre de la persona.
- Surname. Apellido de la persona.
- DateOfBirth. Fecha de nacimiento de la persona.
- Gender. Género de la persona.
- Identity. Identidad de la persona.

Y solo permite "elegir" a los que hayan obtenido el mayor puntaje, asociando a cada uno de éstos la función *selectRecipient* quien se encarga de dibujar la vista *_SectionResume*.

La función *SendEntity* de *HomeController* únicamente valida los datos que recibe y delega la responsabilidad de ejecutar el matching a la función *Match* de la clase *DonaMatching* de *Utilities*. Con el resultado, *SendEntity* únicamente crea, a partir de ellos, objetos *Json* que devuelve a *sendEntity* de *HomeViewModel*. Por lo tanto *Match* es la implementación de la función diseñada en la Figura 4.8 de nombre *Matching.*

Match, interactúa con los repositorios *MatchingOfAttributesRepository*, *TargetEntitiesRepository*, *MatchingResultsRepository*, *RankingPoliciesRepository y RealTargetEntitiesRepository* disponibles mediante la interfaz *IUnitOfWork*. Los repositorios, en el diagrama de la Figura 4.8 se corresponden con la instancia de UOW. Así como la instancia de Función de las secuencias, se corresponde con *Function* en la implementación. *EvaluateFunction* es el método que consume *Match* para evaluar a las funciones incidentes en el matching.

La vista *_SectionResume*, es la que presenta el resumen del matching. El resumen final no se muestra hasta que se genere un archivo de tipo *Clinical Document Architecture*. Esto se hace mediante la invocación a *generateCDA* del modelo de vista, quien llama a *Index* de *CDAController*. El controlador es quien genera el CDA. Obtiene el receptor a partir del *id* del receptor que recibe *Index* como parámetro, luego el *donamatchNumber* a partir del receptor, y por último con todas las entradas de la tabla MATCHINGRESULTS de la Figura 5.6 registradas en el matching, la plantilla de diseño y los estilos XSL, crea el CDA.

Almacena en el directorio configurado el CDA, y además le avisa a *generateCDA* que está listo; *generateCDA* por su parte, muestra el documento en una nueva pestaña mientras crea los elementos de *tableResumeElements*, cuando ya generó la información le avisa a *_SectionResume* mediante el cambio de estado de *tableResumeOk*. Entonces *_SectionResume* muestra los datos del donante y los datos de todos los receptores candidatos. Además se habilita el botón que permite iniciar un nuevo matching en la vista *_SectionFinish*.

5.4 Internacionalización

Para cumplir con el requerimiento 3.3.6, el sistema soporta los lenguajes español, portugués e inglés.

El módulo de internacionalización se divide en dos grandes partes:

- Estática.
  Todos los textos que se muestren en las interfaces gráficas, propios del sistema, son estáticos. No se pueden modificar en tiempo de ejecución.

- Dinámica.
  Los textos de conceptos creados en el sistema, podrán modificarse dinámicamente (además de crearse). Generalmente los textos representan nombres, descripciones, etc. Se deben definir en el idioma del sistema en el momento del alta del texto, así como en los demás. Los textos dinámicos son los que se definen en la sección de Administración.

El idioma del sistema, se toma primero de la elección del usuario. Si aún no ha elegido idioma, se toma del idioma del navegador web o las *cookies*. En todas las pantallas del sistema, se puede cambiar la cultura del sistema.

Para los textos estáticos, se crearon 17 espacios de nombres, cada uno para un contexto diferente. Cada espacio de nombres tiene tres archivos (uno por cada idioma), y en cada archivo existe un conjunto de pares {nombre, valor}. Según el idioma del sistema es el espacio de nombres actual y para ese espacio se muestra el valor del nombre del texto que se desea desplegar en pantalla. Por ejemplo *Entities.resx* es el archivo que contiene valores en español para el contexto de Entidades, *Entities.pt.resx* el que los contiene en portugués y *Entities.en.resx* en inglés.

Entonces si se desea agregar un idioma en Donamatch, es muy sencillo: basta con crear un archivo por cada espacio de nombres. Así como si se desea modificar los valores de los textos, se deben cambiar los valores en los archivos de recursos.

La sección de administración presenta una subsección que permite modificar los textos dinámicos creados en Donamatch. Si bien no se refleja en el modelo de datos de la Figura 5.6 (para simplificar el diagrama), en todas las tablas donde existe algún atributo que es visible en el sistema (se mostrará en alguna vista), existen otros dos atributos que son las traducciones del texto en los demás idiomas. Por defecto, al momento del alta, si no se ingresan las traducciones el sistema registrará los textos igual al idioma original (sin traducir).

El punto frágil del módulo de internacionalización es que para los textos dinámicos, si se quiere agregar otro idioma deberá modificarse el esquema físico de base de datos, agregando un atributo por tabla que corresponda.

Así como es importante internacionalizar el sistema, lo es no entorpecerlo. Por ello es que los pasos de internacionalización no son obligatorios. Si el sistema se usa en un único país, no será necesario traducir.

5.5 Registro del matching

Todas las acciones realizadas en Donamatch, se registran. Todas las altas, bajas, modificaciones y ejecuciones de matching. Por varias razones:

1. Permitir generar informes del uso del sistema.
2. Posibilidad de hacer auditorías de los matchings realizados.
3. Trazabilidad de los conceptos de Donamatch.

El módulo de registro opera sobre la tabla *Logs* y la tabla *MatchingResults*, que se pueden observar en la Figura 5.6. Cada vez que sea crea, edita o elimina un objeto de Donamatch, se crea un registro en la tabla *Logs*. En el Algoritmo 2, esto se corresponde con el paso *Guardar registro del evento*.

En cada matching ejecutado en el sistema, se crea un registro para cada instancia de la operación y para todos los candidatos a receptores en *MatchingResults*.

5.6 Configuración

Además de la configuración del matching que puede hacerse a nivel de usuario mediante la sección de administración, existen otros valores configurables. Los principales son:

- los datos de conexión a la base de datos, mediante la clave *donamatchEntities*.
- Dirección de correo electrónico a donde se enviarán las solicitudes de claves para registro en el sistema. Clave: *AdministratorMail*.
- Dirección de correo electrónico desde el que se enviarán los correos del sistema. Clave: *DonamatchMail.*
- Contraseña del correo de clave *DonamatchMail*. Clave: *DonamatchPassword*.
- Ruta del archivo donde se encuentra la plantilla de diseño del CDA para Donamatch. Clave: *PathCDATemplate.*
- Ruta del archivo donde se almacenará el XML auxiliar para la generación del CDA. Clave: *PathCDAAuxiliar*.

- Directorio donde se almacenarán los CDA generados por Donamatch. Clave: *PathCDAResult*.
- Ruta del archivo XSL de estilos de CDA. Clave: *PathXSLCDA*.

Los archivos .config tienen definidas las claves y valores por defecto, que pueden modificarse.

5.7 Diseño gráfico

Si se hubiera dedicado el esfuerzo necesario para crear un diseño de buena calidad, se hubiera consumido mucho tiempo. Si no se hubiera dado importancia al diseño probablemente la calidad visual del proyecto sería mala impactando en la usabilidad. Las premisas lo evidencian, no se tomaron esos caminos.

Se optó por basar el diseño en *Bootstrap* [67-68] y *Bootswatch* (que está basado en *Bootstrap*) [69-70]. Ya que presentan HTML, CSS y JS limpios y que permiten tener un buen diseño rápida y fácilmente, escalándolo o cambiándolo. Además como una de las principales cualidades, sus estructuras benefician un diseño *responsive* lo que, al usarlo, nos permite brindar un sistema que se adapte a casi cualquier dispositivo. Agrandamos las fronteras: Web, Teléfonos inteligentes, Tabletas, etc. La Figura 5.8 muestra la portada de Donamatch.

Figura 5.8 – *Portada de interfaz de usuario de Donamatch*

La primera vista que se carga en Donamatch (cada vez que se ejecuta una vista) es *_ViewStart.cshtml* que únicamente indica que el *layout* del sistema es *_Layout.cshtml*. Éste incluye la carga de estilos y scripts y ubica dónde se cargará la vista que lo use como plantilla. En el Apéndice 1 se describe con mayor detalle.

Las vistas, a nivel de sistema de archivos, están organizadas en directorios. Si son vistas que son administradas por controladores, el directorio lleva el nombre del controlador que la administra.

La vista inicial del sistema es *Index* de *Home* (controlador: *HomeController*), que determina la estructura del diseño de la SPA. Incluye a todas las secciones: *_SectionWelcome*, *_SectionSelectMatching*, *_SectionMatching*, *_SectionResume*, *_SectionFinish*, *_SectionHelp* y *_SectionFooter*, que se muestran o se ocultan según los *observables* de estado de IU. Así como también sucede con *_Login*, *_Register*, *_RequestKey* y *_Manage*. La sección de administración se accede también desde la pantalla *Index* de *Home*, para los usuarios que corresponda según se detalla en *Sesión de Usuarios*.

En la Tabla 5.3, se muestran las vistas de la sección de Administración. Las demás vistas son las que se presentan en la Tabla 5.5.

Tabla 5.5 – *Vistas de Donamatch*

Vistas	**Directorio**	**Descripción**
Index	*Home*	Vista principal
_Login	*Home*	Inicio de sesión
_Register	*Home*	Registro de usuarios
_RequestKey	*Home*	Solicitud de clave
_Manage	*Home*	Cambio de contraseña
_ SectionFooter	*SinglePage*	Pie de SPA
_SectionMatching	*SinglePage*	Sección de Matching
_SectionResume	*SinglePage*	Resumen del matching
_SectionSelectMatching	*SinglePage*	Elección de matching
_SectionWelcome	*SinglePage*	Vista de bienvenida
_ViewStart	-	Punto de entrada
_Layout	*Shared*	*Layout* de interfaz de uso
_LayoutAdministrator	*Shared*	*Layout* de Administración
_ResourcesJS	*Shared*	Recursos de internacionalización
Error	*Shared*	Vista de error genérica

La carga de los archivos CSS y JS, se hace mediante *bundles* que agrupan los archivos según su uso. Así, cada vista carga únicamente los archivos que necesita, para que la carga de las vistas sea más eficiente.

Al implantar el sistema, deberán generarse los archivos *.min* que permiten mejorar el tiempo de carga debido a que minimizan el tamaño de los archivos. Si bien son ilegibles para desarrollo, cumplen su función en producción.

El ruteo de direcciones URL que admite el sistema y se usa internamente tiene la forma: "{cultura}/{controlador}/{acción}/{parámetros}" y está definido *RouteConfig*.

Si bien uno de los objetivos iniciales fue obtener un diseño gráfico responsivo, con el avance en la construcción del sistema no se le dio la prioridad necesaria para poder soportarlo, por ende el resultado es un diseño poco responsivo, lo que determina que en teléfonos inteligentes no se vea de forma óptima.

Interacción Persona - Donamatch

Los principales objetivos del diseño gráfico fueron:

- Facilitar la velocidad de ejecución del matching.
- Mejorar la curva de aprendizaje del usuario.
- Disminuir la tasa de errores del usuario.
- Lograr la satisfacción subjetiva en el usuario.
- Aumentar la retención del uso de la interfaz a través del tiempo.

La pantalla de inicio de sesión, de registro de usuario, de solicitud de clave, de cambio de contraseña y de bienvenida, es similar a las de un sistema clásico; de colores blanco y azul. Sin embargo, las pantallas del caso de uso más común en el sistema: el matching, al principio fueron diseñadas muy diferente a las de los sistemas clásicos de salud y en general de la mayoría de los sistemas, buscando cumplir el objetivo del quinto punto. El color predominante era el negro y se resaltaban los blancos para los textos. Luego de algunas instancias de validaciones con usuarios finales, se concluyó que el diseño gráfico por sus colores, podría poner en riesgo la aceptación del sistema, por lo que se rediseñó totalmente. El nuevo diseño se corresponde con el Prototipo 4 según la distribución del tiempo mostrada en la Figura 7.2. En la Figuras 5.8 y 5.9 puede observarse el resultado del nuevo diseño.

Los mensajes de éxito, alerta o error, son mostrados en colores verde, amarillo y rojo respectivamente. Esto busca guiar al usuario, resaltar lo importante para contribuir con el primer y segundo objetivo.

También para cumplir con los dos primeros objetivos, pero principalmente para el tercer objetivo es que se implementó una fuerte guía en el flujo de interacción entre el usuario y el sistema. El matching es un paso a paso de acciones, con alternativas claras y mensajes simples. Se buscó que el uso sea intuitivo.

El hecho de que la arquitectura sea SPA también responde a contribuir con los objetivos planteados. Además de poder moverse en la SPA mediante las acciones de los botones, en la barra superior se muestran seis botones que corresponden cada uno a una sección de la SPA. En cualquier momento, el botón de color blanco indica la sección actual del sistema. Y busca actuar en el subconsciente de la persona que usa el sistema indicando el porcentaje de cumplimiento que lleva, de los pasos requeridos para ejecutar el matching.

Además, en la sexta sección de la SPA, se incluye un video que busca guiar al usuario para la ejecución del matching buscando el segundo y el tercer objetivo.

Figura 5.9 – *Vista de interfaz gráfica de Donamatch*

Capítulo 6

RESULTADOS OBTENIDOS

Con Donamatch construido, y el enfoque orientado a la experiencia de usuario, es importante medir, por ejemplo, qué tan fácil de usar es el sistema, qué tan eficiente administra el tiempo, qué tan atractivo resulta a los usuarios. En este capítulo se presenta un intento de medir algunos aspectos de usabilidad y de performance aunque no se dan en el mejor escenario. Los resultados no son concluyentes o representativos debido a la metodología usada.

6.1 Cloud Computing

Para que el sistema pueda ser usado por la mayor cantidad de usuarios posible, debe estar disponible en Internet. Otra opción es instalar el sistema en algún ambiente accesible para los usuarios, por ejemplo en la red interna de centros de salud que quieran cooperar con el proyecto. Pero el costo de la infraestructura necesaria y el esfuerzo que requiere la instalación hacen que no sea una buena opción, de hecho que ni siquiera sea una opción.

Al publicarlo en algún *Cloud*, los costos de hardware y el servicio solo dependerán del uso. Además el tiempo que lleva dejarlo disponible para el mundo entero, es casi nulo.

Entre las opciones de qué *Compute Cloud* usar se evaluó (por ser líderes del mercado):

- *Google Cloud Platform [71]*
- *Amazon Web Services [72]*
- *Microsoft Azure [73]*

Los tres proveen servicios gratuitos durante un período determinado de prueba. Sus condiciones de servicio son muy similares y todos permiten alojar sitios web en .Net, por lo que a priori, sin ahondar en el análisis de qué beneficios brinda una ante otra, se podría elegir cualquiera. La opción seleccionada fue *Azure*, pero no al azar. Dado que la tecnología usada en el sistema fue .Net, con *Azure* no se corre el riesgo de tener que hacer alguna adaptación extra.

La implantación en la nube consistió en dos etapas: migrar la base de datos y publicar el sitio (incluyendo apuntar a la base de datos en *Azure*). Ambas tareas llevaron tres horas. Los principales obstáculos tuvieron que ver con imágenes no incluidas en la solución, y archivos *.svg* y *.ttf*.

La dirección brindada por *Azure* fue donamatch.azurewebsites.net. Dado que no es mnemotécnica, se optó por comprar un dominio durante un año: **donamatch.com**, para que sea memorizable y fácil de distribuir.

6.2 Encuesta

Para medir la experiencia de usuario de Donamatch, se configuró el matching con tres asociaciones de atributos y se elaboró una encuesta que: 1) consta de doce preguntas, lo que no la hace extensa; 2) está escrita en español y portugués para ampliar el universo de encuestados; 3) está disponible en Internet para que sea fácilmente accesible.

Se eligió el sitio Online Encuesta [74] por tener una buena interfaz gráfica, permitir crear encuestas rápidamente y sin costo para fines educativos, contar con un buen análisis en tiempo real de las respuestas y permitir exportar los datos en diferentes formatos.

A continuación se muestran las preguntas elaboradas para la encuesta:

1. Por favor complete la siguiente información: Nombre, Edad y Ocupación.
2. ¿Es usted médico o funcionario de la salud? Opciones:
    a. Sí
    b. No
3. ¿Es usted experto en el uso de tecnologías? Opciones:
    a. Sí
    b. No
4. ¿Ha logrado encontrar el mejor receptor para el órgano donado? Opciones:
    a. Sí
    b. No
5. ¿Ha cometido errores al usar el sistema? Opciones:
    a. Sí
    b. No
6. ¿Cómo clasificaría los pasos que hay que seguir para realizar el matching? Opciones:
    a. Sencillos
    b. Intermedio
    c. Difíciles
7. ¿Cuál cree que es el nivel de dificultad de aprendizaje de Donamatch? Opciones:
    a. muy fácil
    b. fácil
    c. intermedio
    d. difícil

    e. muy difícil
8. ¿Aproximadamente cuánto tiempo le llevó encontrar el receptor (desde que ingresó al sistema)? Opciones:
    a. 5 minutos
    b. 10 minutos
    c. 15 minutos
    d. 20 minutos
    e. 25 minutos
    f. 30 minutos
    g. Más de 30 minutos
9. ¿Cómo cree usted que es el sistema? Opciones:
    a. muy lento
    b. lento
    c. normal
    d. rápido
    e. muy rápido
10. ¿Cuáles de estas cualidades cree usted que posee Donamatch? Opciones (se puede elegir más de una opción):
    a. Comprensible
    b. Atractivo
    c. Inteligente
    d. Novedoso
    e. Ninguna
11. ¿Cree usted que recordará la interfaz (pantallas, colores, textos, etc.) de Donamatch a través del tiempo? Opciones:
    a. Sí
    b. No
12. Si desea, indique sugerencias o comentarios. Es la única pregunta no obligatoria.

La pregunta 1, busca identificar la edad y ocupación de los participantes. La pregunta 2 permitirá determinar qué cantidad de médicos resolvieron la encuesta. La pregunta 3, intenta determinar el perfil del usuario, si es un usuario experto probablemente no tenga dificultades con obtener el mejor receptor. La pregunta 4 cuestiona si el usuario entendió el objetivo del sistema. La pregunta 5 intenta determinar si el usuario se encontró con dificultades. La pregunta 6 apunta a la facilidad de uso. La pregunta 7, por su parte, a la curva de aprendizaje que tiene Donamatch. La pregunta 8 refiere al requerimiento no funcional 3.3.2. La pregunta 9 consulta sobre la percepción del usuario. La pregunta 10 intenta determinar si el sistema cumple con los aspectos de usabilidad de Jakob Nielsen [75-77]. La pregunta 11 está enfocada a la interacción persona computadora. Y la pregunta 12 permite obtener comentarios sobre el sistema.

Para recoger esta información sobre experiencia de usuario, se publicó información vía correo electrónico y redes sociales.

Para que una persona sin conocer el contexto de Donamatch, pueda transformarse en usuario, se creó un video [78] que muestra cómo usar Donamatch. Además muestra cómo solicitar la clave y registrarse.

6.3 Análisis de respuestas

El tamaño de la muestra es de 51 participantes. La encuesta estuvo disponible entre el veinticinco de setiembre de 2014 y el veinte de octubre de 2014.

La Tabla 6.1 muestra las franjas etarias de los encuestados. El promedio de edad es alrededor de treinta.

Tabla 6.1 – *Franjas etarias de los encuestados*

Intervalo	**Cantidad de participantes**
[20 – 29]	38
[30 – 39]	5
[40 – 49]	4
[50 – 59]	2
[60 – 69]	2

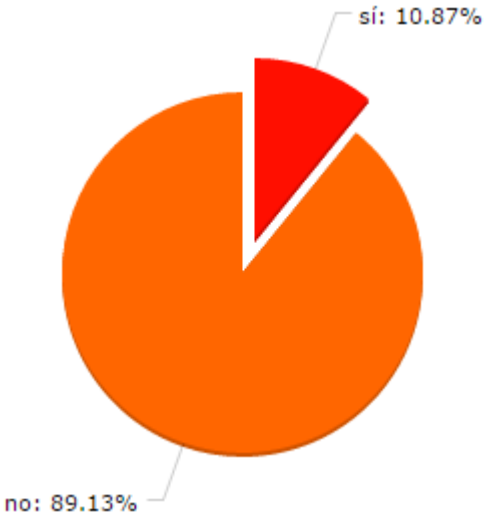


Figura 6.1 – *Respuestas en porcentaje a la pregunta: ¿Es usted médico o funcionario de la salud?*

Como puede verse en el gráfico de la Figura 6.1, únicamente el 10,87% de los encuestados es médico o personal de salud. Se esperaba superar ampliamente este número.

Poco más de la mitad de los usuarios es experto en el uso de tecnologías, precisamente el 56,52%, tal como se ve en la Figura 6.2.

Figura 6.2 – *Respuestas en porcentaje a la pregunta: ¿Es usted experto en el uso de tecnologías?*

De todos los participantes, el 95,65% dice haber encontrado el mejor receptor. Esto puede observarse en la Figura 6.3.

Figura 6.3 – *Respuestas en porcentaje a la pregunta: ¿Ha logrado encontrar el mejor receptor para el órgano donado?*

El 89,13% dice no haber cometido errores en el uso del sistema tal como se puede ver en la Figura 6.4. Para el caso de respuesta negativa, no podemos conocer a qué se refería el usuario con error (la pregunta no es feliz, debería haber sido acompañada por una aclaración). La intención de la pregunta era saber por ejemplo si el usuario olvida un campo obligatorio o se equivoca de paso.

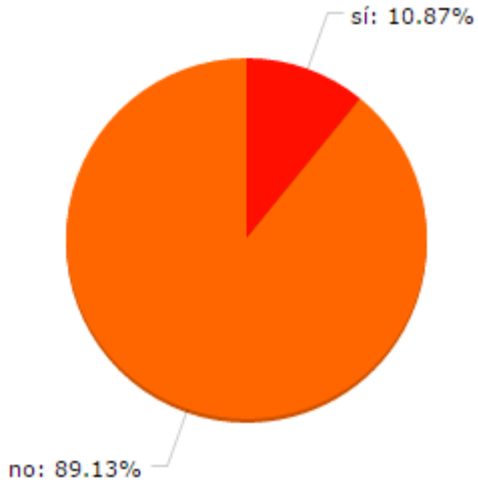


Figura 6.4 – *Respuestas en porcentaje a la pregunta: ¿Ha cometido errores al usar el sistema?*

La Figura 6.5, muestra que ninguno de los participantes cree que los pasos que tuvo que seguir para hacer efectivo el matching son difíciles, casi un 24% cree que tienen dificultad intermedia, y la mayoría: 76% cree que los pasos son sencillos.

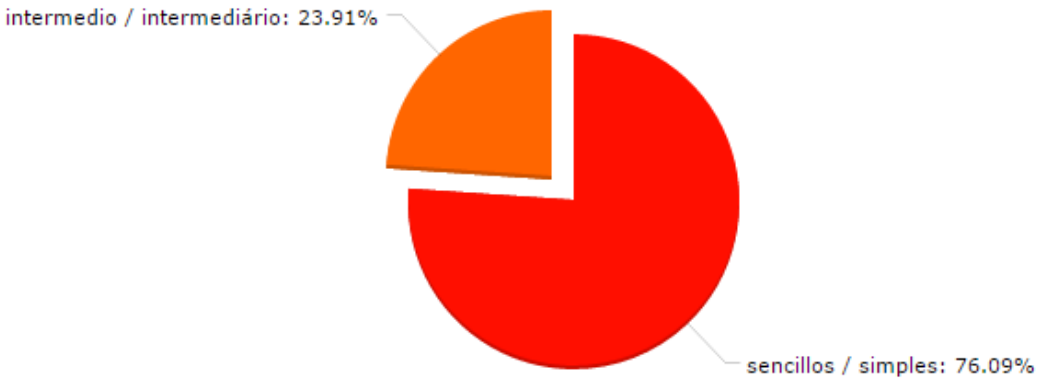


Figura 6.5 – *Respuestas en porcentaje a la pregunta: ¿Cómo clasificaría los pasos que hay que seguir para realizar el matching?*

Respecto a la curva de aprendizaje, la mayoría cree que el nivel de dificultad de aprendizaje es fácil o muy fácil tal como se muestra en la Figura 6.6. Ninguno cree que es difícil o muy difícil.

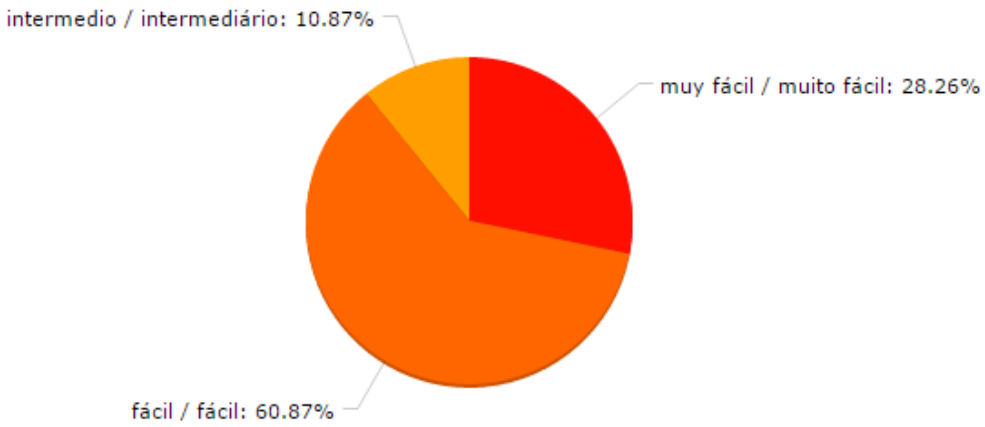


Figura 6.6 – *Respuestas en porcentaje a la pregunta: ¿Cuál cree que es el nivel de dificultad de aprendizaje de Donamatch?*

En cuanto al tiempo que lleva realizar el matching, al 84,78% de los participantes le llevó no más que cinco minutos ejecutar el matching. En todos los casos se cumple con el Requerimiento 3.3.2. La Figura 6.7 permite verlo.

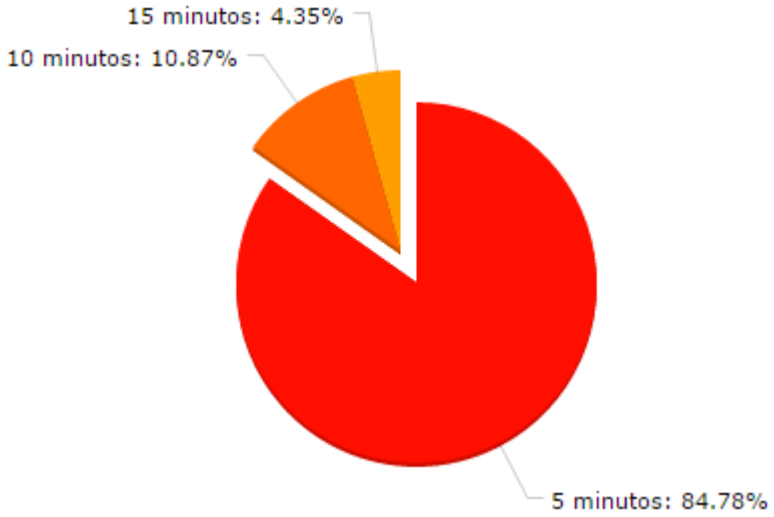


Figura 6.7 – *Respuestas en porcentaje a la pregunta: ¿Aproximadamente cuánto tiempo le llevó encontrar el receptor (desde que ingresó al sistema)?*

La mayoría de los encuestados cree que el sistema es muy rápido o rápido. El 41,3% cree que es normal tal como lo muestra la Figura 6.8. Nadie seleccionó la opción lento o muy lento.

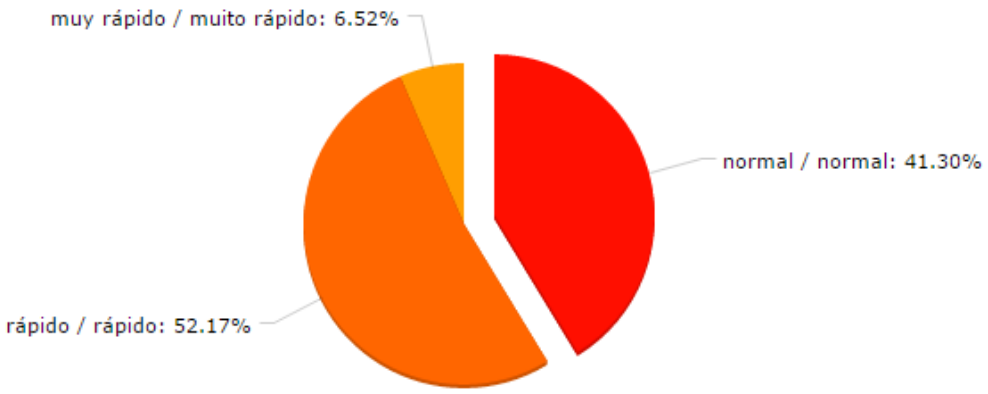


Figura 6.8 – *Respuestas en porcentaje a la pregunta: ¿Cómo cree usted que es el sistema?*

En lo que refiere a las cualidades de Donamatch, 34 personas lo consideran novedoso, 23 lo consideran inteligente, 17 lo consideran comprensible y 15 de los participantes lo consideran atractivo tal como se muestra en la Figura 6.9.

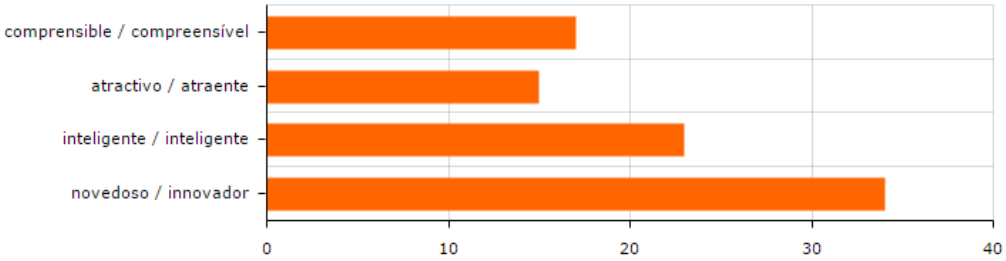


Figura 6.9 – *Respuestas en cantidades a la pregunta: ¿Cuáles de estas cualidades cree usted que posee Donamatch?*

De todos los participantes, la gran mayoría: el 86,96% cree que recordará la interfaz de Donamatch a través del tiempo. Esto puede verse en la Figura 6.10.

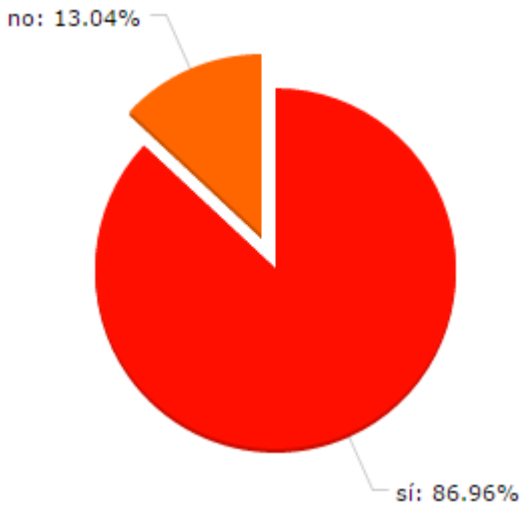


Figura 6.10 – *Respuestas en porcentaje a la pregunta: ¿Cree usted que recordará la interfaz (pantallas, colores, textos, etc.) de Donamatch a través del tiempo?*

La pregunta 12 que permitía hacer comentarios u observaciones sobre Donamatch, fue contestada por quince participantes. Uno de los comentarios más reiterado fue: “el proceso de registro es lento”, “creí que había fallado”, “no me llegaba el correo con la clave”, etc. En la información publicada no quedó claro (incluido el video) que el paso de obtener la clave no es automático, tal como se explica en la sección Seguridad del Capítulo 4. Esto tuvo un impacto negativo importante en algunos usuarios que, luego de solicitar la clave, al no recibir automáticamente un correo electrónico, creían que el sistema fallaba o demoraba en responder. Relacionado a esto, una sugerencia interesante fue: sustituir la clave enviada por un link de confirmación, es más cómodo para el usuario y desde el punto de vista de la seguridad da igual.

Algunos comentarios permitieron corregir incidencias, como por ejemplo problemas de visualización en algunas resoluciones y mensajes poco claros.

Capítulo 7

GESTIÓN DE PROYECTO

Cuando a tempranos días de marzo de 2013 empieza a nacer Donamatch, difícil era imaginarse el rumbo que seguiría, de software de matching a medida a sistema de matching adaptativo bajo el criterio DM. La gestión del proyecto fue fundamental para dar soporte a esta transformación. En el presente capítulo se presenta lo que refiere a tal gestión.

7.1 Metodología de desarrollo y plan de trabajo

Principalmente movido por la ausencia de precisión de requerimientos iniciales se utilizó una metodología que permitiera la evolución del proyecto, pero de facto, no ideada.

El plan de trabajo inicial estaba compuesto de:

- Estudio del estado del arte en sistemas de información y aplicaciones sobre bancos de sangre.
- Estudio de normas legales y estándares.
- Análisis de requisitos.
- Diseño del sistema.
- Implementación del prototipo de Donamatch.
- Pruebas del prototipo y ajustes.
- Puesta en operación con entrenamiento de usuarios.
- Conformidad de los usuarios con el prototipo de Donamatch.
- Publicación de lo obtenido en revista arbitrada.

Todos los puntos se cumplieron excepto el último, que quedará para trabajo futuro, sin embargo el plan de trabajo fue mutando y creciendo exponencialmente tal como puede observarse en el presente documento. Además, el estudio de normas legales y estándares fue únicamente el necesario para entender los tipos de criterios de asignación relacionados a ellos.

En la Tabla 7.1 se encuentra el cronograma con los hitos planteados inicialmente.

El proceso de construcción usado es el representado en la Figura 7.1 que puede definirse iterativo e incremental basado en prototipos *pseudoevolutivos*.

Tabla 7.1 – *Cronograma inicial Donamatch*

Fecha	**Descripción**
Marzo 2013	Estudio preliminar
Abril 2013	Estudio del arte
Mayo a octubre 2013	Diseño e implementación de DONAMATCH
Noviembre 2013	Pruebas, ajustes, publicación y defensa del proyecto

La evolución del software se dio iterativamente entre las etapas de Requerimientos, Análisis, Diseño, Implementación, Verificación, Validación e Instalación.

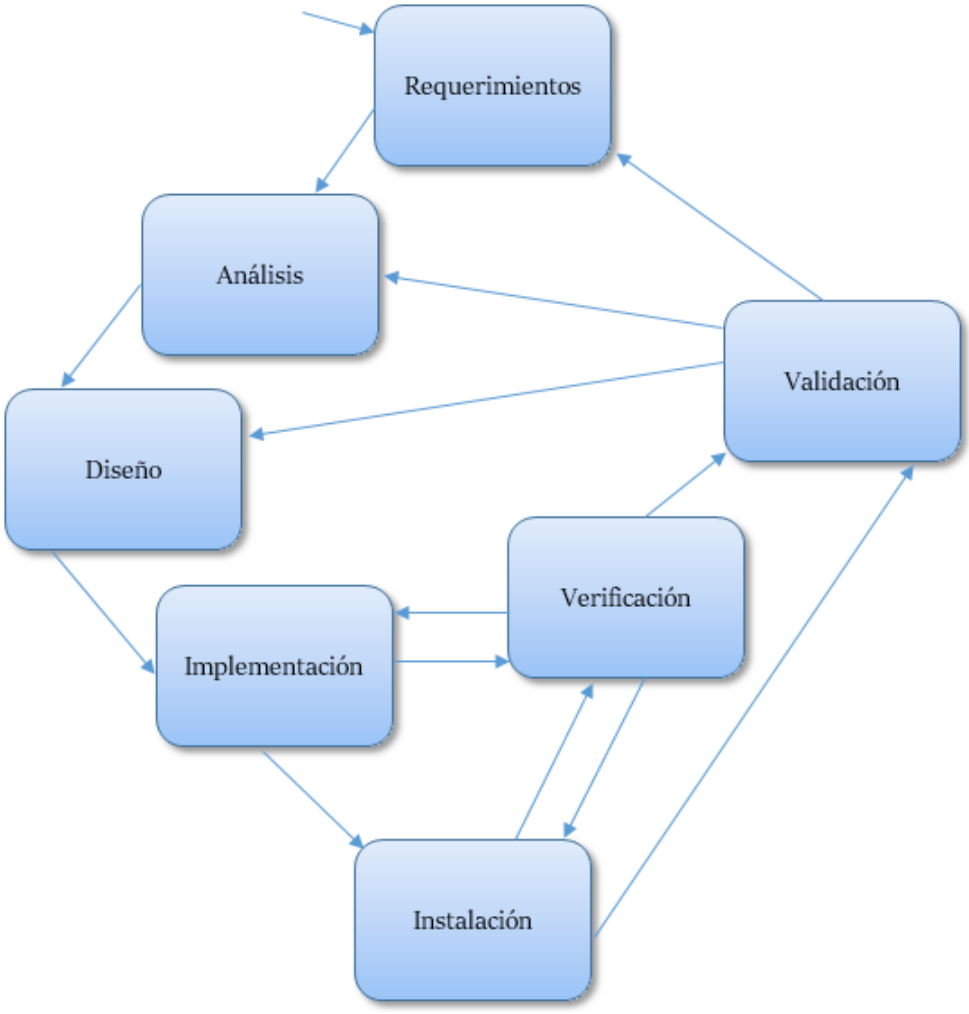


Figura 7.1 – *Proceso de desarrollo de Donamatch*

En las dos primeras iteraciones las transiciones se dieron de Requerimientos a Análisis, de Análisis a Diseño, de Diseño a Implementación,

de Implementación a Verificación (y viceversa) y de Verificación a Validación. Luego, de Validación a Requerimientos nuevamente, debido al cambio en los requisitos. Esto hizo que el prototipo obtenido en la etapa de Validación, sea en parte desechado (requerimientos recortados), en parte reutilizado (requerimientos existentes y nuevos), de allí el nombre asignado al proceso.

Luego del tercer prototipo, la transición se da de Validación a Análisis, o de Validación a Diseño, hasta que en el quinto prototipo se pasa de la etapa Validación a Instalación y partir de allí hasta el producto final, el ciclo está dado por Diseño – Implementación – Verificación – Instalación – Validación.

Sobre el final del proceso, principalmente por la facilidad de instalar en *Azure* y el grupo de verificadores (voluntarios) más grande, el proceso se transformó en uno de pasaje Producción Continua: de Implementación se pasaba directo a Instalación y luego a Verificación.

7.2 Tiempo dedicado

Los principales hitos del proyecto fueron las liberaciones de los prototipos (ocho en total, de los cuales cuatro fueron directo sobre producción) tal como lo muestra la Figura 7.2.

<table>
<tr><th colspan="2">Fecha</th><th>Iteración</th><th>Hitos</th><th>Hitos secundarios</th></tr>
<tr><td rowspan="6">2013</td><td>Mayo</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="3">Estado del arte</td></tr>
<tr><td>Junio</td></tr>
<tr><td>Julio</td><td rowspan="4">Iteración 1</td><td rowspan="5">Prototipo 1</td></tr>
<tr><td>Agosto</td><td></td></tr>
<tr><td>Setiembre</td><td></td></tr>
<tr><td>Octubre</td><td>Matching genérico</td></tr>
<tr><td rowspan="6">2014</td><td>Noviembre</td><td rowspan="4">Iteración 2</td><td>Quitar módulo de sangre</td></tr>
<tr><td>Diciembre</td><td rowspan="4">Prototipo 2</td><td rowspan="4"></td></tr>
<tr><td>Enero</td></tr>
<tr><td>Febrero</td></tr>
<tr><td>Marzo</td><td rowspan="3">Iteración 3</td><td rowspan="3">Acercamiento sistema INDT</td></tr>
<tr><td>Abril</td><td rowspan="3">Prototipo 3</td></tr>
<tr><td>Mayo</td></tr>
<tr><td></td><td>Junio</td><td></td><td></td><td rowspan="2"></td></tr>
<tr><td></td><td>Julio</td><td>Iteración 4</td><td>Prototipo 3</td></tr>
<tr><td></td><td>Agosto</td><td>Iteración 5</td><td>Prototipo 4</td><td></td></tr>
<tr><td></td><td>Setiembre</td><td>Producción continua</td><td>Prototipo 5, 6 y 7</td><td>Publicación en Azure</td></tr>
<tr><td></td><td>Octubre</td><td>Producción continua</td><td>Prototipo 8</td><td></td></tr>
</table>

Figura 7.2 – *Distribución aproximada del tiempo por actividad*

El principal hito secundario del proyecto fue la decisión de pasar de matching a medida a matching genérico en octubre de 2013, lo que

desencadenó la definición del Criterio DM. Como consecuencia, la iteración más larga (cinco meses) fue la segunda, que consistió en volver a rehacer algunas etapas como por ejemplo especificación de requerimientos, tal como lo muestra la transición entre Validación y Requerimientos de la Figura 7.1. En esta iteración se hizo el análisis, diseño, e implementación únicamente de la Sección de Administración cuya implementación se detalla en la Sección 5.2 ya que constituía el mayor reto, es el alma de Donamatch.

En la tercera iteración se analizó, diseñó e implementó la parte más visible del proyecto: la sección usada por el actor que ejecuta el matching. De la cuarta iteración en delante se agregaron funcionalidades secundarias.

El tiempo dedicado para el presente trabajo se presenta en la Tabla 7.2. La actividad que llevó más dedicación horaria fue Investigación con el 23%. La sigue Diseño con el 21% e Implementación con el 20,3%. Por lo tanto se le dedicó casi el mismo tiempo al diseño que a la implementación, esto se debe a que el diseño fue parcial, ya que lo referente a la interfaz gráfica no fue diseñado por no ser orientado a objetos.

Tabla 7.2 – *Tiempo dedicado desde mayo de 2013 a octubre de 2014*

Actividad	**Horas**	**Porcentaje**
Investigación	431	23%
Gestión de la configuración	33	1,8%
Análisis	242	12,9%
Diseño	395	21%
Implementación	381	20,3%
Verificación	175	9,3%
Instalación	11	0,6%
Documentación	162	8,7%
Otros	45	2,4%
Total	**1875**	

La etapa de instalación representa el 0,6% del trabajo realizado e incluye la instalación inicial, las liberaciones de los prototipos, la configuración del dominio adquirido, etc.

La cuarta actividad que llevó mayor dedicación fue Análisis con un 12,9%, mientras que la verificación requirió el 9,3% de la dedicación total.

En la actividad Otros, se incluye por ejemplo la creación del ambiente de desarrollo, el uso de herramientas como *Mendeley* [79] que permitió una sencilla administración de referencias bibliográficas y *Trello* [80] que se usó para organizar las tareas e hitos del proyecto.

Capítulo 8

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo, se presentan las conclusiones del trabajo y el trabajo futuro.

8.1 Conclusiones generales

El trabajo realizado permitió cumplir con el principal objetivo del proyecto. Se construyó un sistema de software que permite la asignación eficiente de órganos a receptores. Aunque lo de asignación eficiente es discutible, ante la ausencia de pruebas empíricas que lo muestren. Es cierto también, que algunos aspectos no necesitan pruebas. Es indiscutible la transparencia en el matching contribuyendo con los aspectos éticos del trasplante, el beneficio de poder redefinir el sistema cuando necesario permitiendo cambiar los criterios de asignación ante el descubrimiento de mejores métodos, el beneficio de la posibilidad de usar el mismo sistema en diferentes países ya sea para compartir el método de asignación, datos, e incluso órganos.

Se definió un criterio general para el matching: el Criterio DM y un conjunto de conceptos que permitió diseñar el sistema genéricamente.

Se construyó un software usando nuevas tecnologías y patrones de arquitectura que si bien al principio dificultaron el avance, principalmente por su complejidad, permitieron luego desde el punto de vista teórico, tener una buena experiencia de usuario. Se aprovechó la existencia de excelentes diseños HTML libres usándolos como base del diseño gráfico del sistema.

La experiencia de usuario parece ser buena según los resultados obtenidos a partir de las respuestas dadas por 51 participantes a las preguntas planteadas en la encuesta Donamatch, tal como se detalla en el Capítulo 6.

La Gestión de la Configuración del proyecto fue muy buena. Pues los fuentes, documentos y datos se administraron de forma tal que están plenamente identificados, controlados y organizados. Si bien la gestión de configuración en un grupo unipersonal puede parecer redundante, permitió que se conozca la trazabilidad de todos los elementos de configuración. Nunca se perdió ningún cambio, siempre se identificaron perfectamente las versiones, etc.

8.2 Trabajo Futuro

Donamatch, busca ser el puntapié inicial de trabajos futuros con objetivos alineados. A continuación se describen posibles trabajos a futuro relacionados al proyecto.

Aprendizaje automático

La rama de la inteligencia artificial que le daría mucho beneficio al proyecto es aprendizaje automático. Si se extendiera el sistema permitiendo almacenar información referente a los receptores de los órganos asignados, y dicha información fuera usada para aprender respecto al tiempo de vida pos trasplante, Donamatch podría adaptarse a la realidad mejorando la eficiencia de la asignación de órganos de una comunidad, país o continente. O al menos permitiría brindar información al respecto para que las decisiones sean tomadas por juntas médicas.

Definir criterios reales de asignación

Es necesario comparar el sistema. A lo largo del proyecto, hubo intentos, fallidos, de obtener acceso al sistema usado por el INDT con el fin de configurar en Donamatch los criterios usados por tal sistema. Y luego introducir datos históricos de donantes y candidatos a receptores del INDT a Donamatch y comparar los receptores obtenidos contra los receptores reales. Para poder mostrar (o mostrar que no es posible refutar), que cualquier sistema de matching de órganos es un caso particular de Donamatch (una configuración).

Se propone como trabajo a futuro, configurar a Donamatch con criterios reales de asignación ya sea del INDT o cualquier centro de trasplantes del mundo que permita comparar el sistema con uno exitoso usado en la realidad.

Uso de Donamatch

Se plantea someter el sistema a más usuarios de centros de trasplantes o con perfil similar, como médicos, personal de un hospital, etc. Con tres principales objetivos:

- Entrenarlos, y tomar tiempos de ejecución del matching.
- Medir la curva de aprendizaje de los usuarios.
- Analizar nivel de aceptación de los usuarios.

Con los tiempos, se puede concluir acerca de la velocidad con la que se puede hacer el matching, el nivel medio de aceptación que puede tener su uso así como ver qué tan rápido aprende a usar el sistema un usuario promedio, de diferentes edades y países. Las pruebas realizadas en este trabajo, cuyos

resultados se presentan en la sección 6.3 Resultados, permitieron tener una primera impresión del uso de Donamatch, es importante aumentar la muestra.

Se debería usar el sistema en dos modalidades: 1) Administración del sistema. 2) Uso del sistema.

Si bien el punto 2) tiene mayor importancia por el hecho de que la configuración se hace en muy pocas ocasiones, si se configura mal el sistema, no importa qué tan buena pueda ser la ejecución del matching, la asignación será ineficiente.

Testing

Si bien se incluyeron etapas de test unitarios, de interfaces y de sistema, se encontraron y corrigieron defectos, un sistema que ayuda a decidir qué órgano se asignará a qué receptor, debe ser testeado mucho más fuertemente.

Una debilidad del testeo del sistema fue que la misma persona que lo desarrolló, lo testeó. Es sabido que no es una buena práctica por los sesgos subconscientes que pueden existir. Si bien hubo testeo funcional por parte de otras personas, no fue formal. Además en general las pruebas fueron de caja negra.

Se plantea como línea de trabajo futuro, ejecutar pruebas de caja blanca e inspección de código.

Mejorar configuración

Si bien el sistema es muy configurable, se propone como trabajo a futuro, dejarlo más configurable. Por ejemplo los textos estáticos no pueden modificarse en tiempo de ejecución, y si bien son textos que no cambian en el sistema, facilitaría la internacionalización o culturización al permitir cambiarlos a nivel de usuario.

Por otra parte existen algunas constantes a nivel de código que podrían agregarse a los archivos de configuración para que puedan modificarse sin la necesidad de volver a compilar el sistema.

Seguridad

En el presente trabajo, solo se consideraron los aspectos más básicos de seguridad de la aplicación. Dada la importancia de la función de Donamatch, se plantea mejorar aspectos de seguridad. Podrá tenerse como guía otros aspectos considerados en OWASP [60].

Como posible línea de trabajo, podría intentarse atacar el sistema para descubrir vulnerabilidades y corregirlas.

Menor abstracción

El DM definido y los conceptos permiten tener un buen balance entre abstracción y usabilidad. No hay pruebas que indiquen que la administración de Donamatch sea difícil de usar. No obstante, gap mediante, parece ser que configurar el sistema es una tarea compleja. El mapeo de los criterios científicos, sociales y políticos al sistema no es trivial. Por tal motivo es que se plantea como trabajo a futuro intentarlo llevar a más alto nivel. Principalmente si se demuestra que al usuario le dificulta la configuración.

Integración con Historia Clínica Electrónica

El CDA que se genera en Donamatch, luego debe adjuntarse manualmente a la HCE del receptor. Se plantea como una mejora al sistema, su automatización.

Integración con Listas de espera

Se propone como trabajo a futuro la implementación de un módulo de Donamatch, o una nueva aplicación que integre los datos de diferentes fuentes de datos a la base de datos de Donamatch. O bien definir una interfaz que permita la integración con otros sistemas, por ejemplo mediante servicios *Rest*.

REFERENCIAS

[1] Google, Public data. Datos de Banco Mundial.

[2] Sung RS, Abt PL, Desai DM, Garvey CA, Segev DL, Kaufman DB. The right organ for the right recipient: the Ninth Annual American Society of Transplant Surgeons' State-of-the-Art Winter Symposium. Clin Transplant 2011.

[3] Asamblea Mundial de la Salud, en su resolución WHA63.22. Principios rectores de la OMS sobre trasplante de células, tejidos y órganos humanos. 2010.

[4] M. Tewari, H. S. Shukla. Sushruta: The Father of Indian Surgery. Indian J Surg, 2005

[5] Veith I, Huang Ti Nei Ching Su Wen. The Yellow Emperor's Classic of Internal Medicine. Baltimore: Williams & Wilkins Co; 1949.

[6] Mackinney Loren. Medical Illustrations in Medieval Manuscripts. Berkley: University of California Press; 1965.

[7] Stephen Paget, James Paget. John Hunter: Man of science and surgeon (1728-1793). London: T. Fisher Unwin, 1897.

[8] Gray C. The remarkable surgical collection of John Hunter., Canadian Medical Association Journal [Can Med Assoc J]. 1983.

[9] Susan L. Smith. Historical Perspective of Transplantation. 2002. http://www.medscape.com/viewarticle/436532.

[10] Wendy Moore. The Knife Man: The Extraordinary Life and Times of John Hunter, Father of Modern Surgery. London: Transworld Publishers, 2005.

[11] Southard J H, Belzer FO. History of Transplantation. In: Flye MW, ed. Principles of OrganTransplantation. Philadelphia, Pa: WB Saunders Co; 1989.

[12] Freeman Erica R., Bloom David A., McGuire Edward J., A Brief History of Testosterone, The Journal of Urology. 2001.

[13] Armitage, W. J., Tullo, A. B., & Larkin, D. F. P. The first successful full-thickness corneal transplant: A commentary on eduard zirm's landmark paper of 1906. *British Journal of Ophthalmology*. 2006.

[14] Alexis Carrel – Biographical. http://www.nobelprize.org/nobel_prizes/medicine/laureates/1912/carrel-bio.html.

[15] Landsteiner, K. Cell antigens and individual specificity. J Immunol 15. 1928.

[16] Medawar, Peter and Lehner, T.. Major histocompatibility system: the Gorer Symposium. Blackwell Scientific Publications. 1983.

[17] Cruse, J.M.; R.E. Lewis. Illustrated Dictionary of Immunology. 1994.

[18] Scott H. Podolsky, Alfred I. Tauber. The Generation of Diversity: Clonal selection theory and the rise of molecular immunology. 2000.

[19] Peter Medawar – Biographical. http://www.nobelprize.org/nobel_prizes/medicine/laureates/1960/medawar-bio.html.

[20] P. B. Medawar. The behaviour and fate of skin autografts and skin homografts in rabbits A report to the War Wounds Committee of the Medical Research Council. J Anat. 1944.

[21] Dausset J. Iso-leuco-anticorps. Acta Haematol. 1958.

[22] Terasaki PI, Marchioro TL, Starzl TE. Serotyping of human lymphocyte antigens. Preliminary trials on long-term kidney homograft survivors. Nat Acad Sci Monograph. 1965.

[23] Terasaki PI, Marchioro TL, Starzl TE. Histocompatibility Testing. Washington, DC: National Academy of Sciences. 1965.

[24] History of Organ and Cell Transplantation. Nadey S. Hakim, Vassilios Papalois. 2003.

[25] Moore FD, Birtch AG, Dagher F, et al. Immunosuppression and vascular insufficiency in liver transplantation. Ann N Y Acad Sci. 1964.

[26] Murray JE, Merrill JP, Dammin GJ, et al. Study of transplantation immunity after total body irradiation: clinical and experimental investigation. Surgery. 1960.

[27] Hamburger J, Vaysse J, Crosnier J, et al. Transplantation of a kidney between nonmonozygotic twins after irradiation of the receiver. Good function at the fourth month. Presse Med. 1959.

[28] Schwartz R, Dameshek W. The effects of 6-mercaptopurine on homograft reactions. J Clin Invest. 1960.

[29] Hakim, N., & Papalois, V. History of Organ and Cell Transplantation. Imperial College Press. 2003

[30] Calne, Roy. Essay History of transplantation. Lancet. 2006.

[31] Elion, G. "The purine path to chemotherapy". Science 244. 1989.

[32] Organ Transplant History. www.donatelifeny.org/transplant/organ_history.html.

[33] Starzl TE, Marchioro TL, Waddell WR. The reversal of rejection in human renal homografts with subsequent development of homograft tolerance. Surgery, gynecology & obstetrics. 1963.

[34] Barnard CN. What we have learned about heart transplants. J Thorac Cardiovasc Surg. 1968.

[35] Stinson EB, Griepp RB, Clark DA, et al. Cardiac transplantation in man. VIII. Survival and function. J Thorac Cardiovasc Surg. 1970.

[36] Hardy JD, Chavez CM, Kurrus FD, et al. Heart transplantation in man. Developmental studies and report of a case. JAMA. 1964.

[37] James D. Hardy, 84, Dies; Paved Way for Transplants. The New York Times. http://www.nytimes.com/2003/02/21/us/james-d-hardy-84-dies-paved-way-for-transplants.html

[38] Historia del trasplante. Sociedad Española de Neumología y Cirugía Torácica SEPAR. http://separ2013trasplantepulmonar.com/historia-del-trasplante/

[39] Kelly WD, Lillehei RC, Merkel FK, Idezuki Y, Goetz FC. Allotransplantation of the pancreas and duodenum along with the kidney in diabetic nephropathy. Surgery 61. 1967.

[40] Altruistic donor makes possible the first “domino” three-way kidney transplant operation. Johns Hopkins Medicine. http://www.hopkinsmedicine.org/Press_releases/2005/05_19_05.html/

[41] França-Tarragó, Omar. Aspectos éticos del trasplante de órganos.

[42] Bengochea, Milka. Asignación en trasplante de órganos. 2014.

[43] Sims Sterling, Fiester Autumn. A Brief History of Organ Transplantation. Penn Bioethics Journal.

[44] Stevens, M. L. (2003).

[45] Ley N° 14.005 Publicada en el Diario Oficial el 20.8.71, Trasplante de órganos y tejidos. http://www.indt.edu.uy/documentos/documentacion_legal/ley_14005.pdf.

[46] Muerte encefálica, Instituto Nacional de Donación y Trasplante, Septiembre 2012 - 4a Edición.

[47] Historia. INDT. http://www.indt.edu.uy/?S=historia

[48] Decreto N° Ref. 001-1170/2005. http://www.indt.edu.uy/documentos/documentacion_legal/decreto_8-8-2005.pdf

[49] Procuración y trasplante de órganos y tejidos. INDT. Período 2005-2013. http://www.indt.edu.uy/documentos/estadisticas/estadisticas2013.pdf

[50] Uruguay busca exportar orgulloso a la región su exitoso modelo de trasplantes. Entrevista a Inés Alvarez. Efe http://mexico.servidornoticias.com/24_salud/2495000_uruguay-busca-exportar-orgulloso-a-la-region-su-exitoso-modelo-de-trasplantes.html.

[51] La gran historia - Donación de órganos en Uruguay. INDT. http://www.indt.edu.uy/?S=abrir&item=147.

[52] Fuerza Aérea Uruguaya presentó el nuevo Sistema de Transporte Aeromédico de Emergencia. INDT. http://www.indt.edu.uy/?S=abrir&item=148.

[53] Datos de U.S. Government Information on Organ and Tissue Donation and Transplantation.

[54] Simini F. Perinatal information system (SIP): a clinical database in Latin America and the Caribbean. Lancet. 1999.

[55] Burbeck Steve. Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). 1992.

[56] Reenskaug, Trygve and O. Coplien, James. The DCI Architecture: A New Vision of Object-Oriented Programming. 2009.

[57] User interface design practices in simple single page web applications. First International Conference on the Applications of Digital Information and Web Technologies, ICADIWT. 2008.

[58] Brehm, Spike. Building Single-Page Apps. Tech Talks. 2012. http://nerds.airbnb.com/slides-and-video-from-spike-brehms-tech-talk/

[59] Fowler Martin, Rice Dave, Foemmel Matthew, Hieatt Edward, Mee Robert, and Stafford Randy. Patterns of Enterprise Application Architecture. 2002.

[60] Open Web Application Security Project (OWASP) Top Ten. The Ten Most Critical Web Application Security Risks.

[61] Gamma Erich, Helm Richard, Johnson Ralph, Vlissides John. Design Patterns, Elements of Reusable Object-Oriented Software.

[62] Knockout. http://knockoutjs.com/

[63] Knockout MIT License. http://opensource.org/licenses/mit-license.php.

[64] Angular. https://angularjs.org/

[65] Backbone. http://backbonejs.org/

[66] Jsfiddle. http://jsfiddle.net/

[67] Bootstrap. http://getbootstrap.com/

[68] Bootstrap MIT License. https://github.com/twbs/bootstrap/blob/master/LICENSE

[69] Bootswatch. http://bootswatch.com/

[70] Bootswatch MIT License. https://github.com/thomaspark/bootswatch/blob/gh-pages/LICENSE

[71] Google Cloud Platform. https://cloud.google.com/

[72] Amazon Web Services. http://aws.amazon.com/

[73] Microsoft Azure. https://azure.microsoft.com

[74] Online Encuesta. https://www.onlineencuesta.com/

[75] Nielsen Jakob, Usability 101: Introduction to Usability. 2002.

[76] Nielsen Jakob. Top 10 Mistakes in Web Design. 2011.

[77] Nielsen Jakob. How Users Read on the Web. 1997.

[78] Video de ejemplo de uso de Donamatch. https://www.youtube.com/watch?v=3gLeNkZE_cY

[79] Mendeley, http://www.mendeley.com/

[80] Trello, https://trello.com/

APÉNDICE A – HERRAMIENTAS Y LENGUAJES

En este apéndice se detallarán únicamente las herramientas utilizadas que no son nombradas en el resto del documento.

El sistema se desarrolló en C# de .NET. Se usó como entorno de desarrollo Visual Studio 2013 (se empezó con Visual Studio 2012 y luego se migró a 2013). Se siguió el esquema de solución y proyectos. La Figura A.1 muestra la estructura de la solución de Donamatch que contiene seis proyectos.

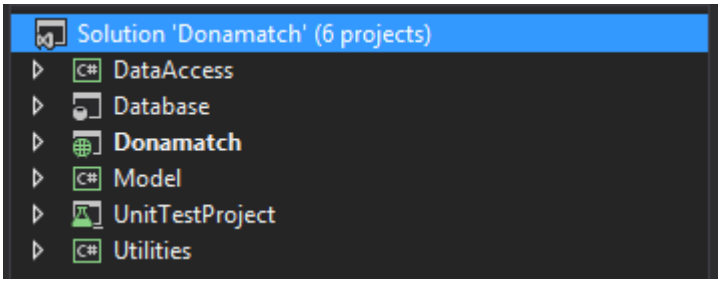


Figura A.1 - *Solución Donamatch*

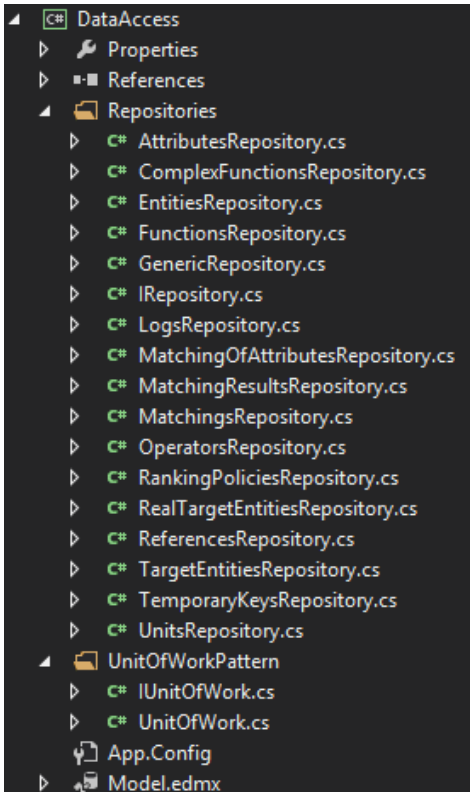


Figura *A.2 - Proyecto DataAccess*

El proyecto *DataAccess* representa la capa de Acceso a Datos. Las clases de este proyecto están organizadas en dos carpetas: *Repositories* y *UnitOfWorkPattern*. La primera contiene los repositorios y la segunda la implementación del patrón UOW. Tal como lo muestra la Figura A.2.

En total son quince clases repositorios, todos heredan de *GenericRepository*, esta clase base implementa la interfaz *IRepository*.

Model.edmx, es el modelo de datos mapeado de las tablas del esquema de base de datos. Permite actualizar el modelo a partir de la base de datos, y actualizar la base de datos desde el modelo. Como base de datos se usó Microsoft SQL Server 2012.

El proyecto *Database* contiene los scripts que permiten crear la base de

datos así como scripts que permiten agregar datos (los provistos por Donamatch) a la base de datos.

UnitTestProject es un proyecto que contiene testeos unitarios hechos sobre las clases de la solución.

El proyecto *Utilities*, contiene las clases *DonaMatching* que se encarga de la lógica del matching, *Function* que se encarga de implementar las funciones provistas por el sistema, *Generator* que tiene la capacidad de generar claves solicitadas para el registro de usuarios, *Mailing* que envía correos electrónicos, *Constants* que contiene a las constantes del código y *Enums* que contiene los enumerados de Donamatch.

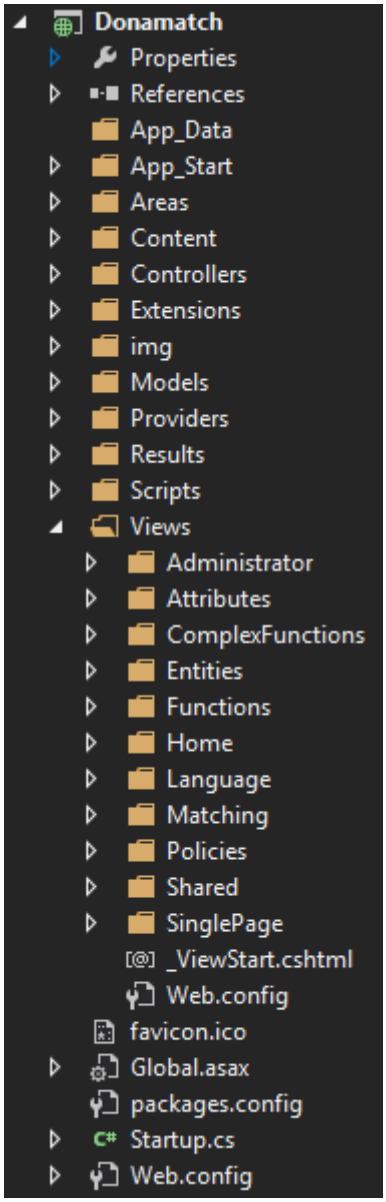


Figura A.3 - *Proyecto Donamatch*

El proyecto *Model* representa el modelo de MVC y además contiene metadatos de él y los archivos de recursos para los textos estáticos de la internacionalización.

El proyecto Donamatch, que se observa en la Figura A.3, contiene carpetas que agrupan a los archivos conceptualmente:

- *App_Start* contiene a las clases de inicio del sitio.
- *Areas* contiene clases para la página de ayuda (no está en uso).
- *Content* posee todos los archivos relacionados a los estilos (CSS y fuentes).
- *Controllers* contiene a todos los controladores.
- *Extensions*, contiene a la clase *Extensions* que extiende a Json.
- *img* contiene a las imágenes.
- Models, contiene los modelos que se usan para las vistas.
- *Providers*, proveedor de autenticación.
- *Results*, para gestión de acciones.
- *Scripts* contiene todos los archivos de JavaScript.
- *Views*, contiene a todas las vistas.

El archivo de configuración principal del sistema es *Web.config*.

APÉNDICE B – LAYOUT

La Figura B.1 presenta la implementación del *layout* de Donamatch.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>@ViewBag.Title</title>
        <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta name="format-detection" content="telephone=no">
        @Styles.Render("~/Content/csscyborg")
        @Scripts.Render("~/bundles/modernizr")
    </head>
    <body>
        @RenderBody()

        @Scripts.Render("~/bundles/jquery")
        @Scripts.Render("~/bundles/knockout")
        @Scripts.Render("~/bundles/app")
        @Scripts.Render("~/bundles/bootstrap")
        @Scripts.Render("~/bundles/app/control")
        <script src="~/Scripts/knockout.mapping-latest.js"></script>
        @RenderSection("Scripts", required: false)
    </body>
</html>
```

Figura B.1 - *Layout.cshtml*

APÉNDICE C – REGISTROS

Tal como se describe en la sección 5.2, todas las acciones realizadas por los administradores del sistema se registran en la tabla LOGS. La figura C.1 muestra una vista de esta tabla para la configuración de la validación de experiencia de usuarios.

Por otra parte, cada acción ejecutada por los usuarios se registra en la tabla MATCHINGRESULTS, la Figura C.2 muestra una vista de los datos de la tabla recogidos durante la validación de experiencia de usuarios.

Tabla C.1 – *Vista de datos de la tabla Logs*

Date	Description	UserName
2014-08-17 23:03:03.767	Se ha creado una entidad en la tabla Entities de Id = 1.	diego
2014-08-17 23:04:11.163	Se ha modificado culturización en la tabla Entities, Id = 1.	diego
2014-08-17 23:07:19.930	Se ha creado una entidad en la tabla Entities de Id = 2.	diego
2014-08-17 23:09:02.573	Se ha modificado culturización en la tabla Entities, Id = 2.	diego
2014-08-17 23:30:05.793	Se ha creado una entidad en la tabla Attributes de Id = 1.	diego
2014-08-17 23:30:40.247	Se ha modificado culturización en la tabla Attributes, Id = 1.	diego
2014-08-17 23:30:57.973	Se ha creado una entidad en la tabla Attributes de Id = 2.	diego
2014-08-17 23:31:24.137	Se ha modificado culturización en la tabla Attributes, Id = 2.	diego
2014-08-17 23:32:10.437	Se ha creado una entidad en la tabla Attributes de Id = 3.	diego
2014-08-17 23:32:37.423	Se ha modificado culturización en la tabla Attributes, Id = 3.	diego
2014-08-17 23:33:17.100	Se ha creado una entidad en la tabla Attributes de Id = 4.	diego
2014-08-17 23:33:36.523	Se ha modificado culturización en la tabla Attributes, Id = 4.	diego
2014-08-17 23:33:51.793	Se ha creado una entidad en la tabla Attributes de Id = 5.	diego

Tabla C.2 – *Vista de datos de la tabla MatchingResults*

C1	C2	C3	C4	C5	C6	C7	C8	C9
1509461675	1	8	NULL	1	1	1	2	1
1509461675	1	9	NULL	0	NULL	NULL	2	1
1509461675	1	25	NULL	1	1	1	2	1
1509461675	1	28	NULL	0	NULL	NULL	2	1
1509461675	1	31	NULL	0	NULL	NULL	2	1
1509461675	1	34	NULL	0	NULL	NULL	2	1
1509461675	1	553	NULL	1	1	1	2	1
1509461675	1	620	NULL	0	NULL	NULL	2	1
1509461675	1	621	NULL	0	NULL	NULL	2	1
1509461675	1	622	NULL	0	NULL	NULL	2	1
1509461675	1	623	NULL	1	1	1	2	1
1509461675	1	629	NULL	1	1	1	2	1
1509461675	1	630	NULL	0	NULL	NULL	2	1
1509461675	1	631	NULL	0	NULL	NULL	2	1
1509461675	1	9	NULL	0	NULL	NULL	2	1
1509461675	1	25	NULL	1	1	1	2	1
1509461675	1	28	NULL	0	NULL	NULL	2	1

Referencias de columnas. C1: DonamatchNumber, C2: IdEntitieSource, C3: IdTargetEntity, C4: alfa_n, C5: beta_n, C6: gamma_n, C7: delta_n, C8: n, C9: IsRequirement.

APÉNDICE D – MANUAL TÉCNICO

En este apéndice se describirán aspectos técnicos, de diseño o implementación de bajo nivel que no se hayan abordado antes en el informe. Sin embargo el código escrito es auto-descriptivo, por la nemotecnia de los métodos, *datatypes*, clases, interfaces, etc. Además incluye comentarios cuando necesarios.

D.1 Transacciones con bases de datos

Las transacciones sobre las bases de datos se dan mediante los repositorios expuestos por la interfaz IUnitOfWork, tal como puede verse en la Figura A.1.

```
public interface IUnitOfWork : IDisposable
{
    4 references
    TemporaryKeysRepository TemporaryKeysRepository { get; }
    19 references
    EntitiesRepository EntitiesRepository { get; }
    26 references
    AttributesRepository AttributesRepository { get; }
    14 references
    FunctionsRepository FunctionsRepository { get; }
    14 references
    MatchingsRepository MatchingsRepository { get; }
    14 references
    MatchingOfAttributesRepository MatchingOfAttributesRepository { get; }
    5 references
    UnitsRepository UnitsRepository { get; }
    5 references
    OperatorsRepository OperatorsRepository { get; }
    17 references
    ComplexFunctionsRepository ComplexFunctionsRepository { get; }
    32 references
    LogsRepository LogsRepository { get; }
    5 references
    TargetEntitiesRepository TargetEntitiesRepository { get; }
    7 references
    ReferencesRepository ReferencesRepository { get; }
    3 references
    RealTargetEntitiesRepository RealTargetEntitiesRepository { get; }
    19 references
    RankingPoliciesRepository RankingPoliciesRepository { get; }
    5 references
    MatchingResultsRepository MatchingResultsRepository { get; }
    46 references
    bool Save();
}
```

Figura D.1 – *Interfaz IUnitOfWork*

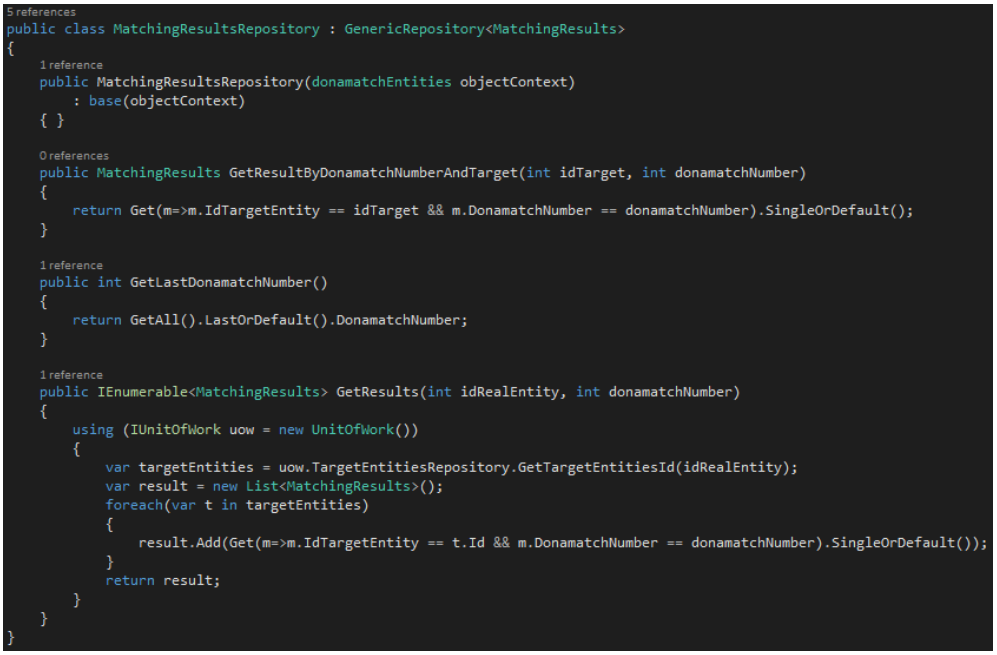
La Interfaz, a su vez hereda de IDisposable para el manejo de liberación de recursos, mediante el método Dispose().

Cada repositorio mostrado en la Figura D.1 (por ejemplo MatchingResultsRepository), hereda de la clase GenericRepository<T> donde T es una clase genérica. GenericRepository por su parte, implementa a la interfaz IRepository<T>. La interfaz IRepository expone los servicios que se muestran la Figura D.2.

```
1 reference
public interface IRepository<T> where T : class
{
    37 references
    IEnumerable<T> GetAll();
    34 references
    IEnumerable<T> Get(Expression<Func<T, bool>> filter = null,
        Func<IQueryable<T>, IOrderedQueryable<T>> orderBy = null,
        string includeProperties = "");

    30 references
    T GetById(object id);
    1 reference
    T SingleOrDefault(Expression<Func<T, bool>> where);
    1 reference
    T FirstOrDefault(Expression<Func<T, bool>> where);

    43 references
    void Add(T entity);
    19 references
    void Update(T entity);
    11 references
    void Delete(T entity);
    1 reference
    void Delete(object id);
}
```

Figura D.2 – *Interfaz IRepository*

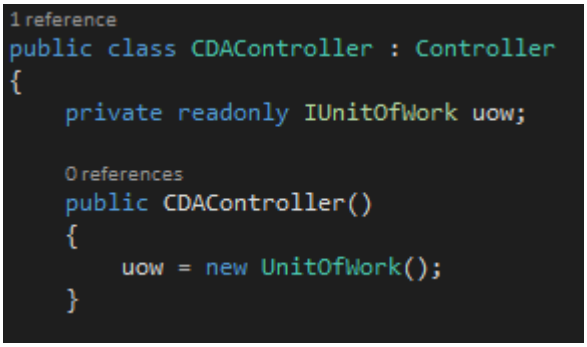
Los métodos expuestos brindan un conjunto de funcionalidades básicas usadas para cualquier objeto *persistible* a través de los repositorios (por herencia y polimorfismo). Además, si quieren definirse métodos personalizados, puede hacerse en cada repositorio particular. La Figura D.3 muestra un ejemplo: la clase MatchingResultsRepository.



```
5 references
public class MatchingResultsRepository : GenericRepository<MatchingResults>
{
    1 reference
    public MatchingResultsRepository(donamatchEntities objectContext)
        : base(objectContext)
    { }

    0 references
    public MatchingResults GetResultByDonamatchNumberAndTarget(int idTarget, int donamatchNumber)
    {
        return Get(m=>m.IdTargetEntity == idTarget && m.DonamatchNumber == donamatchNumber).SingleOrDefault();
    }

    1 reference
    public int GetLastDonamatchNumber()
    {
        return GetAll().LastOrDefault().DonamatchNumber;
    }

    1 reference
    public IEnumerable<MatchingResults> GetResults(int idRealEntity, int donamatchNumber)
    {
        using (IUnitOfWork uow = new UnitOfWork())
        {
            var targetEntities = uow.TargetEntitiesRepository.GetTargetEntitiesId(idRealEntity);
            var result = new List<MatchingResults>();
            foreach(var t in targetEntities)
            {
                result.Add(Get(m=>m.IdTargetEntity == t.Id && m.DonamatchNumber == donamatchNumber).SingleOrDefault());
            }
            return result;
        }
    }
}
```

Figura D.3 – *Clase MatchingResultsRepository*

La forma de invocar los métodos de acceso a datos se da en la lógica, desde los Controladores. En el constructor de cada controlador, se crea una instancia de UOW con la que después se realizarán las transacciones.



```
1 reference
public class CDAController : Controller
{
    private readonly IUnitOfWork uow;

    0 references
    public CDAController()
    {
        uow = new UnitOfWork();
    }
```

Figura D.4 – *Ejemplo de creación de instancia UOW*

La conexión a la base de datos es manejada por contextos. La clase que tiene la responsabilidad de administrarlo es donamatchEntities que extiende de la clase DbContext. La información de conexión se maneja en el archivo app.config.

El contexto se genera mediante los objetos DbSet<T> donde T es la clase para la que se quiere generar el contexto. El mapeo entre base de datos y estos objetos se genera en Model.edmx. En la Figura D.4 puede observarse una vista parcial de la clase donamatchEntities. Notar que los métodos no tienen referencias, pues se crean al crean en forma *lazy loading*, a demanda, cuando son usados por el patrón UOW.

```
20 references
public partial class donamatchEntities : DbContext
{
    1 reference
    public donamatchEntities()
        : base("name=donamatchEntities")...

    0 references
    protected override void OnModelCreating(DbModelBuilder modelBuilder)...

    0 references
    public virtual DbSet<TemporaryKeys> TemporaryKeys { get; set; }
    0 references
    public virtual DbSet<UserLogins> UserLogins { get; set; }
    0 references
    public virtual DbSet<UserManagement> UserManagement { get; set; }
    0 references
    public virtual DbSet<Users> Users { get; set; }
    0 references
    public virtual DbSet<UserSecrets> UserSecrets { get; set; }
    0 references
    public virtual DbSet<Operators> Operators { get; set; }
    0 references
    public virtual DbSet<Logs> Logs { get; set; }
    0 references
    public virtual DbSet<Attributes> Attributes { get; set; }
    0 references
    public virtual DbSet<Units> Units { get; set; }
    1 reference
    public virtual DbSet<ComplexFunctions> ComplexFunctions { get; set; }
    0 references
    public virtual DbSet<Functions> Functions { get; set; }
    0 references
    public virtual DbSet<MatchingOfAttributes> MatchingOfAttributes { get; set; }
    0 references
    public virtual DbSet<Matchings> Matchings { get; set; }
    0 references
    public virtual DbSet<Entities> Entities { get; set; }
```

Figura D.5 – *Clase donamatchEntities*

D.2 Controladores

Todos los controladores tienen la misma estructura. Heredan de la clase Controller y proveen métodos GET y métodos POST. La mayoría tiene autorización a nivel de controladores, por roles, tal como se especifica en el Capítulo 4, Sección 4.4 Arquitectura, Seguridad.

En la Figura D.5 se muestra una vista parcial del controlador MatchingController. En ella puede verse la *annotation* ValidateAntiForgeryToken, para el pedido POST del método Define, cuya función se explica en el Capítulo 5, Sección 5.5 Seguridad.

En la Figura D.6 también puede verse que la autorización para métodos de este controlador se brindará solo para Administradores.

```
[Authorize(Roles = Constants.Roles.Administrator)]
1 reference
public class MatchingController : Controller
{
    private readonly IUnitOfWork uow;
    0 references
    public MatchingController()...

    // GET: /Matching
    0 references
    public ActionResult Index()...

    // GET: /Matching/DownloadReference
    0 references
    public ActionResult DownloadReference(int? id)...

    // GET: /Matching/Define
    0 references
    public ActionResult Define()...

    // POST: /Matching/Define
    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public ActionResult Define([Bind(Include = "Id,IdEntitieA,IdEntitieB,Name_es")] Matchings matching)...
```

Figura D.6 – *Vista parcial del controlador MatchingController*

D.3 *ViewModels*

La comunicación entre los modelos de vista y los controladores se da mediante invocaciones *ajax* y el protocolo HTTP.

```
if (!self.tableResumeOk()) {
    ko.utils.arrayForEach(self.selectedDonorAttributes(), function (d) {
        self.tableResumeElements.push(new self.resumeElement(d.idAttribute, documer
    });
    $.ajax({
        url: self.datamodel.selectRecipient,
        type: 'POST',
        contentType: 'application/json; charset=utf-8',
        data:
            JSON.stringify({
                id: self.selectedRecipient().id()
            }),
        success: function (data) {
            for (var j = 0; j < self.tableResumeElements().length; j++) {
                var i = 0;
                while (i < data.length) {
                    if (self.tableResumeElements()[j].id == data[i].Key) {
                        self.tableResumeElements()[j].valueTarget = data[i].Value;
                        i = data.length;
                    }
                    i++;
                }
            }
            self.tableResumeOk(true);
        },
        fail: function () {
            self.showMessageError();
        }
    });
}
```

Figura D.7 – *Ejemplo de uso de métodos HTTP*

Se define la URL, el tipo de método HTTP, el tipo de contenido, los datos que puedan pasarse, y funciones a realizar tanto para el éxito de la respuesta como para el fracaso. En la Figura D.7 se puede ver un ejemplo de invocación a un método del controlador mediante un HTTP POST.

El formato usado para los datos transferidos desde los *viewmodel* hacia los controladores y viceversa es JSON, tal como se ve en el método *JSON.stringify()* y en el *contentType* de la invocación (*application/json*), de la Figura D.7.

La ruta que indica dónde reside el método se resuelve mediante el módulo *app.datamodel*, por tal motivo es que en el ejemplo se obtiene de la forma: *self.datamodel.selectRecipient*. En la Figura D.8 se muestran algunas rutas a modo de ejemplo.

```
function AppDataModel() {
    var self = this,
        // Routes
        addExternalLoginUrl = "/api/Account/AddExternalLogin",
        changePasswordUrl = "/api/Account/changePassword",
        loginUrl = "/Token",
        logoutUrl = "/api/Account/Logout",
        registerUrl = "/api/Account/Register",
        registerExternalUrl = "/api/Account/RegisterExternal",
        removeLoginUrl = "/api/Account/RemoveLogin",
        setPasswordUrl = "/api/Account/setPassword",
        siteUrl = "/",
        userInfoUrl = "/api/Account/UserInfo",
        requestkeyUrl = "api/Account/RequestKey";
```

Figura D.8 – *Ejemplo de rutas para métodos HTTP*

D.4 Vistas

Así como los controladores y los modelos de vista, las vistas de Donamatch siguen la misma estructura. La lógica de la vista se maneja con los *viewmodels* de *knockout* y el HTML es basado en *bootstrap*.

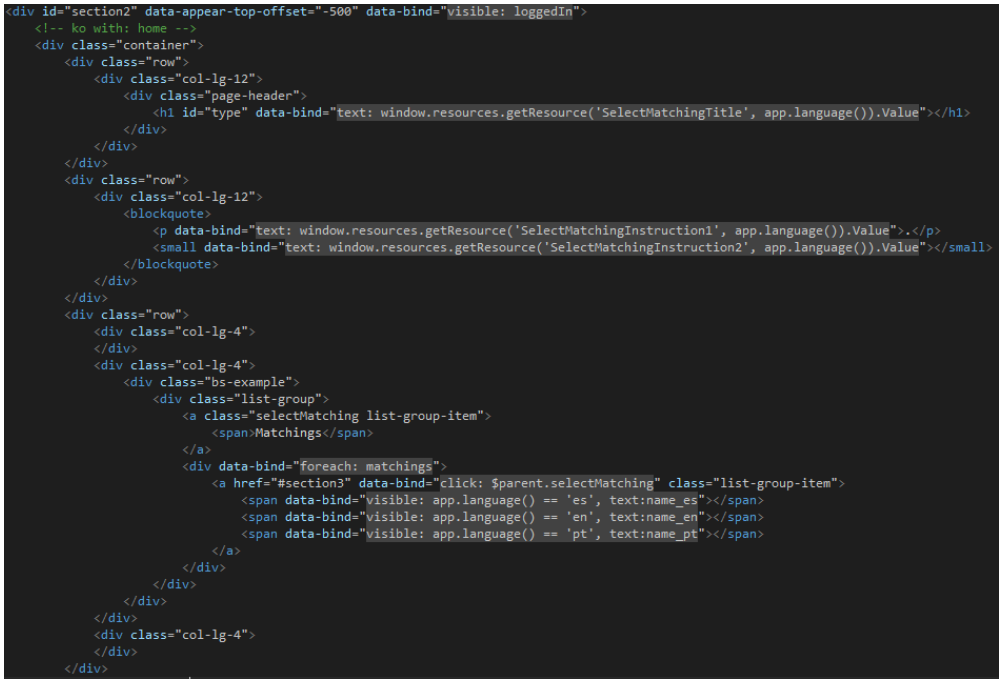
En general, una vista tiene asociado un *viewmodel*, y el contexto de la vista está dado por él, para definir el contexto se usa el atributo *with*.

La Figura D.9 presenta la vista *_SectionSelectMatching.cshtml*, donde puede verse que el *viewmodel* que le corresponde es home del archivo home.viewmodel.js, por la línea:

```
<!-- ko with: home -->
```

El atributo *visible* determina la condición que debe cumplirse para dibujar el HTML contenido. En este caso se invocará a la función *loggedIn* que es un

computable de *knockout* y en este caso particular determina si el usuario ha iniciado sesión.



```
<div id="section2" data-appear-top-offset="-500" data-bind="visible: loggedIn">
    <!-- ko with: home -->
    <div class="container">
        <div class="row">
            <div class="col-lg-12">
                <div class="page-header">
                    <h1 id="type" data-bind="text: window.resources.getResource('SelectMatchingTitle', app.language()).Value"></h1>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="col-lg-12">
                <blockquote>
                    <p data-bind="text: window.resources.getResource('SelectMatchingInstruction1', app.language()).Value">.</p>
                    <small data-bind="text: window.resources.getResource('SelectMatchingInstruction2', app.language()).Value"></small>
                </blockquote>
            </div>
        </div>
        <div class="row">
            <div class="col-lg-4">
            </div>
            <div class="col-lg-4">
                <div class="bs-example">
                    <div class="list-group">
                        <a class="selectMatching list-group-item">
                            <span>Matchings</span>
                        </a>
                        <div data-bind="foreach: matchings">
                            <a href="#section3" data-bind="click: $parent.selectMatching" class="list-group-item">
                                <span data-bind="visible: app.language() == 'es', text:name_es"></span>
                                <span data-bind="visible: app.language() == 'en', text:name_en"></span>
                                <span data-bind="visible: app.language() == 'pt', text:name_pt"></span>
                            </a>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-lg-4">
            </div>
        </div>
```

Figura D.9 – *Parte de la vista* _SectionSelectMatching.cshtml

Tanto para mostrar la internacionalización estática como la dinámica se usa *knockout*. En el caso de la estática, la encargada de obtener el texto según el idioma actual es la función *getResource* quien recibe el nombre del recurso, que hace de clave y el idioma actual. Esto puede verse en el ejemplo en el *data-bind="text:window.resources.getResource('SelectMatchingTitle', app.language()).Value"*.

Para los textos dinámicos, en general se usan condiciones con el texto. El idioma actual es el *observable app.language()*, que se usa para mostrar el texto correspondiente según el idioma del usuario.

En la Figura D.9 puede verse que se muestra un elemento HTML *span* en el idioma actual:

```
<span data-bind="visible:app.language()=='es', text:name_es"></span>
<span data-bind="visible:app.language()=='en', text:name_en"></span>
<span data-bind="visible:app.language()=='pt', text:name_pt"></span>
```

Otro tipo de *data-bind* usado es por ejemplo el *click*. Se le asocia al clic de un elemento una función, en el ejemplo se invoca la función *selectMatching.*