UNIVERSIDAD DE LA REPUBLICA

SAREM: Sistema de Alertas y Registros Médicos

Definición, especificación, proyecto e implementación de un sistema original para aumentar la eficiencia del agendado de horas de centros de salud y el empoderamiento de personas atendidas en el primer nivel de atención.

Documentación del trabajo realizado por

Valentina Da Silva

Leonardo Clavijo

como parte de los requerimientos para aprobar la asignatura
Proyecto de Grado
de la Carrera de Ingeniería en Computación.

Tutores: Prof. Saadia Zawadski, Ing. Lucía Grundel y Prof. Ing. Franco Simini
Núcleo de Ingeniería Biomédica de las Facultades de Medicina e Ingeniería

Trabajo realizado en colaboración con la Escuela Universitaria de Tecnología Médica (EUTM)
con la participación de Victoria Aldaz, María Fernanda Rodríguez y Silvana Arrambide bajo la guía de la Prof. Saadia Zawadski directora de la Licenciatura en Registros Médicos, en la modalidad de
PIRIM - Proyecto Interdisciplinario de Registros Informáticos Médicos

Montevideo, URUGUAY abril 2015 – abril 2016

INDICE

Resumen

SAREM, Sistema de Alertas y Registros Médicos, es un proyecto realizado en colaboración con la carrera de Registros Médicos (RRMM) de la Escuela Universitaria de Tecnología Médica (EUTM) de la Facultad de la Medicina. El trabajo interdisciplinario de estudiantes de RRMM con estudiantes de ingeniería genera un intercambio beneficioso para ambas partes, acercando a los estudiantes de registros al mundo de la informática, al plantear un problema de su ámbito de trabajo y que pueda ser resuelto por estudiantes de Ingeniería. El problema planteado fue la dificultad de los pacientes y proveedores de salud para la gestión de citas médicas.

SAREM surgió con el objetivo de optimizar la interacción de los pacientes con sus centros de salud, y mejorar la gestión de los recursos médicos.

Desde el año 2008, el sistema de salud uruguayo ha evolucionado con la implementación del Sistema Nacional Integrado de Salud (SNIS). El sistema permite el acceso a los trabajadores y jubilados a servicios de salud a través del Seguro Nacional de Salud (SNS). Además, desde el año 2010, también se incluye a su cónyuge e hijos.

Esto ha aumentado el número de personas que asisten a los proveedores de atención médica. Lo que trae como consecuencia una variedad de problemas, como por ejemplo, encontrar lugares libres para citas médicas, para especialidades altamente demandadas. Otro problema importante es que muchos pacientes faltan a las citas médicas sin avisar, dejando lugares libres que no son aprovechados por pacientes que lo necesitan. Lo que genera pérdidas económicas para los centros de salud y una pobre calidad de atención para sus pacientes.

Con el fin de mejorar esta situación, la agencia de gubernamental AGESIC, diseño un plan para la informatización del Sistema de Salud Uruguayo, denominado Salud.uy.[1] Este plan incluyó la adopción de la Historia Clínica Electrónica Nacional (HCEN), que ha llevado a aumentar la adopción de sistemas de información de proveedores de atención médica.[2] Sin embargo, el estado actual de informatización se encuentra en una fase intermedia. La mayoría de los centros de salud basan su principal gestión en papel y almacenan poca información en medios electrónicos. [3]

El resultado logrado es un sistema que mejora las problemáticas planteadas, proveyendo una interfaz amigable para la interacción de sus usuarios. Se puede interactuar con SAREM desde su aplicación Web y su aplicación para dispositivos móviles Android. El sistema esta implementado con tecnologías de código abierto como Node.js y servicios de la nube (Amazon Web Services), lo que permite bajos costos, alta fiabilidad y facilidad de instalación en cualquier plataforma existente. SAREM dio lugar a un producto sólido, con una alta aceptación y buena crítica después de ser

mostrado en Ingeniería deMuestra 2015. Aunque queda trabajo por hacer, creemos que el sistema es aplicable y adaptable a diversos centros de salud en el país y la región.

1. Introducción

En la actualidad en el Uruguay existen sistemas de agenda electrónica en algunas mutualistas del país como ejemplo podemos citar al CASMU y Hospital Británico. Pero no existe un acceso universal de este tipo de sistemas a nivel nacional. Muchos usuarios de diferentes centros de salud no cuentan con la facilidad de poder solicitar una consulta médica desde sus hogares así como tampoco pueden ver los diferentes horarios posibles de la misma. También desde el punto de vista de los centros de salud, muchos usuarios faltan a las consultas sin avisar, lo que resulta en horas desperdiciadas del profesional médico, que traen como consecuencia una mala administración de los recursos. Por otra parte, el contacto entre el paciente y el médico en la actualidad resulta bastante distante. El paciente solicita una consulta, asiste a ella, en la misma el médico se informa de su situación de salud, le receta un tratamiento acompañado de medicación si así corresponde, pero luego depende enteramente del paciente seguir en contacto con el médico para que este pueda ver su evolución. Muchas veces también, dada la demanda y la poca oferta que existe en algunos centros de salud, pueden pasar meses para que el paciente pueda agendar una nueva consulta con su médico. En esta era tecnológica en la cual vivimos, donde la tecnología cada vez nos ayuda más a resolver problemas de nuestra vida cotidiana, se puede ver claramente que las problemáticas planteadas anteriormente tienen solución mediante el uso de un sistema informático. El cual puede permitir una correcta utilización de los recursos médicos, un contacto más cercano entre el profesional de salud y el paciente, y una mejor optimización del tiempo tanto para los funcionarios de los centros de salud como para sus usuarios.

Formar parte de este proyecto nos interesó desde un principio ya que nos permitió interactuar con profesionales de otras áreas y definir un sistema en conjunto con estudiantes avanzadas de RRMM. Las estudiantes involucradas nos plantearon diversas problemáticas que existían en los centros de salud en lo que refiere al agendado de consultas médicas. Nosotros vimos la oportunidad de resolver dichas dificultades mediante un sistema informático al cual denominamos SAREM.

2. Estado del Arte

Antes de diseñar, buscamos lo que ya fue realizado, publicado o en uso para responder a la problemática planteada. Por un lado buscamos a nivel mundial cuales eran las mejores soluciones de software que existían en la actualidad para la administración de consultas médicas y por otro lado investigamos a nivel nacional centros de salud que tuvieran este tipo de software y sus características principales.

2.1 Aplicaciones en uso a nivel mundial

NueMD

NueMD [4][5] está diseñado para oficinas pequeñas a medianas que necesitan una solución de software de gestión de la práctica médica avanzada a un precio asequible. NueMD es fácil de usar y ofrece una gama completa de características y servicios personalizables que mejoran las prácticas médicas administrativas y las necesidades clínicas. Además, el NueMD EHR puede añadirse al sistema de gestión para las prácticas que desean un sistema totalmente integrado. El presupuesto inicial con el que se debe contar para utilizar este tipo de software para acceder a los paquetes estándares por proveedor es de 149 dólares mensuales. [6]

Gestión de la práctica médica:

NueMD desarrolló el primer software de gestión de la práctica basado en la nube, lo que elimina la necesidad de costosos servidores y permiten que los médicos y el personal puedan acceder a la información práctica y a los registros incluso cuando se encuentran fuera de la oficina. Se ofrecen planes mensuales o anuales flexibles que se adaptan a diferentes presupuestos. Además, los formadores NueMD se encuentran entre los mejores en la industria. NueMD es utilizado por más de 20.000 médicos, gerentes de oficina y cobros en todo Estados Unidos.

Registros electrónicos de salud:

Un sistema de atención clínica completa, NueMD EHR es fácil de usar, asequible y totalmente integrado con el software de gestión de la práctica. Las prácticas son capaces de trazar, códificar y compartir información con otros proveedores y sistemas de laboratorio de manera rápida y fácil. NueMD EHR es también basado en la nube, por lo que todos los registros se pueden acceder por los médicos y el personal designado desde un dispositivo móvil.

Características y beneficios incluyen:

- Navegación fácil y rápida a través de un diseño simple y ordenado.

- Los informes personalizados se ejecutan contra una variedad de filtros.
- Fácil integración móvil.
- A demanda uso significativo de documentación para agilizar la recopilación de datos.

Servicios médicos de facturación:

NueMD software de facturación médica combina softwares potentes de la compañía tales como EHR y el software PM con la experiencia de un equipo experimentado de los emisores de facturas médicas y los codificadores para proporcionar a los clientes un programa de facturación asequible que aumenta el reembolso y simplifica el proceso de facturación.

Benchmark Systems

Benchmark Systems [7] tiene soluciones de software de práctica médica basadas en la nube y en servidor. Estas soluciones cubren la programación de citas, los registros electrónicos de salud (EHR), facturación y cobro, y otras características de prácticas de administración. Además de los ordenadores de escritorio, el software se puede utilizar en ordenadores portátiles, iPads y iPhones. Benchmark Clinical es el módulo de EHR que contiene una base de conocimientos de la especialidad pre-cargado que busca coincidir con las necesidades de la práctica del cliente en cuestión. Todos los formularios son totalmente personalizables.
Los flujos de trabajo pueden ser modificados para los médicos individuales, incluyendo alertas de mantenimiento de la salud, protocolos y otras preferencias. También hay un portal para pacientes, donde los mismos pueden entrar y acceder a toda su información de la salud, desde las historias clínicas a las citas y estado de cuenta.
Más de un usuario ha señalado que una de las mejores cosas de Benchmark es el soporte técnico y de usuario que ofrecen. Los usuarios dicen que el personal de apoyo les resulta útil y de gran ayuda y siempre están dispuestos a hacer un esfuerzo adicional para hacer que las necesidades del cliente se cumplan y los problemas se resuelvan.
El presupuesto inicial con el que se debe contar para utilizar este tipo de software para acceder a los paquetes estándares por proveedor es de 160 dólares mensuales. [8]

2.2 Aplicaciones en uso a nivel nacional

CASMU

En febrero de 2015 el CASMU inició la implementación de un nuevo sistema de Historia Clínica Electrónica (HCE), diseñado para optimizar la atención y los servicios que la institución brinda a sus afiliados.

Flexible a las necesidades y formas de trabajo de cada especialidad y nivel de atención, el sistema permite centralizar el registro de afecciones y antecedentes de los pacientes, reduciendo tiempos de espera, mejorando procesos y brindando información en tiempo real facilitando la realización de diagnósticos y la determinación de tratamientos.

Además, la nueva HCE permite agendar consultas e indicar y visualizar exámenes de laboratorio o imagenología por vía electrónica, así como digitalizar las prescripciones farmacológicas, erradicando paulatinamente el uso de las recetas en papel.

La plataforma, desarrollada por la empresa chilena Saydex e instalada en conjunto con la firma local Medifix SRL, está en proceso de quedar totalmente operativa a comienzos de 2016. Durante los 12 meses que ha llevado su realización, se ha realizado una importante inversión en hardware y se ha capacitado al personal médico y no médico.[9]

Hospital Británico

Desde finales del año 2010 el Hospital Británico puso a disposición de sus afiliados un novedoso sistema de reservas online que les brinda la posibilidad de agendar día y hora para citas médicas, cancelar las previamente pactadas y realizar consultas de su estado de cuenta mensual. También desde ese tiempo utiliza para sus afiliados la historia clínica electrónica (HCE), incluyendo la posibilidad de realizar la impresión de estudios y recetas en los consultorios, lo que brinda al médico la posibilidad de consultar en tiempo real resultados y resúmenes clínicos, optimizando así la consulta médica con sus pacientes. [10][11]

Anda

ANDA, centro asistencial modelo a nivel de Latinoamérica, a comienzos del año 2012 eligió GeneXus Consulting para la implantación de la Historia Clínica Digital. El objetivo: poner la tecnología al servicio de los usuarios para mejorar la calidad médica.

El equipo de ANDA (médico e informático) y los profesionales de GeneXus Consulting lograron construir una solución que sitúa a la historia clínica del paciente como núcleo central y conecta tanto la información médica con todos los servicios y costos asociados a ese paciente. Se implementó K2B Health, sistema que permite que todos los servicios de atención que se brindan a los afiliados puedan registrarse electrónicamente, sin necesidad de utilizar papel. Entre las ventajas de K2B Health destacan la posibilidad de no solo digitalizar la historia clínica, sino también informatizar otros sistemas satélites como Farmacia, Enfermería y Coordinación de Consultas. *"El usuario de ANDA llega a cualquiera de las clínicas y se autogestiona en*

las pantallas touch <u>únicamente</u> con el número de cédula sin necesidad de esperas, sin presentar órdenes o cualquier otro tipo de trámite", explicó Julio López Navarro, vicepresidente de ANDA. Los usuarios acceden al instante a la hora, especialista que lo va atender, consultorio y, una vez que se acepta la cita, su nombre aparece en la lista de pacientes del médico en cuestión. Además, si el usuario es atendido por varios médicos un mismo día e incluso en sucursales diferentes, la historia clínica se actualiza en tiempo real, registrando cada evento y cada acción con máxima seguridad y confidencialidad.

Paralelamente, la Historia Clínica Digital permite ingresar los distintos exámenes médicos del paciente, incluir imagenología, conectarse con la cuenta corriente del paciente en la farmacia, calcular el stock de medicamentos, programar alertas, etc. *"Por ejemplo, si hay medicamentos que se contraponen la historia le avisa al médico que está recetando algo que puede ser perjudicial para el paciente. También se puede programar la Historia para que notifique cuando se vencen las vacunas, cuando es fecha de realizarse el PAP o una mamografía, etc. Se puede configurar cualquier alerta que advierta tanto a médicos como a pacientes"*, expresó Daniel Bulla, director del Servicio Médico de ANDA.

La solución diseñada por GeneXus Consulting cumple con los principios de veracidad, confidencialidad y accesibilidad para cada usuario, cumpliendo de esta forma las pautas establecidas por el Ministerio de Salud Pública. La Historia Clínica Digital de K2B Health además de abarcar todas las áreas de servicio, es una herramienta fundamental para la gestión y definición de políticas médicas, pues permite obtener estadísticas en tiempo real del estado de grupos poblacionales, y realizar informes epidemiológicos y análisis puntuales. [12]

SAE – Sistema de Agenda Electrónica

La Intendencia de Montevideo es uno de los organismos precursores del software público en Uruguay. El sistema de agenda electrónica fue desarrollado durante 2008 con el objetivo de organizar la atención al público brindando un servicio individual y personalizado, siendo liberado como Software Público a comienzos de 2010.

Tiempo después, el BPS, que tenía necesidades similares respecto a la atención al público, adopta la agenda como solución efectuando distintas modificaciones para adaptarla a las normas de infraestructura y seguridad del organismo e incorporando nuevas funcionalidades. De esta forma logra rápidamente satisfacer las necesidades iniciales implantando el sistema en un tiempo considerablemente menor que un desarrollo a medida. Luego comienza una etapa de trabajo colaborativo, planificando el mantenimiento correctivo y extensivo de la aplicación y empleando fuerza de desarrollo de ambas partes.

Actualmente el software ha sido implantado con éxito en ambos organismos. En la IM se ha configurado una agenda para el servicio de Contralor de Conductores para la reserva de horas para la renovación de licencia de conducir, en Contralor y Registro de Vehículos para reserva de hora de trámites, entre otros; registrándose actualmente más de 900 reservas diarias. En BPS se desplegaron más de 20 agendas, recibiendo más de 1000 reservas diarias a través de distintos canales.

Funcionalidades

Entre sus principales funcionalidades se encuentra la configuración de recursos (locales de atención, un grupo de abogados, la atención al público), el ingreso de información de reservas por parte del ciudadano y su validación por parte del Organismo.

Actualmente el software ha sido implantado con éxito en la IM. Se ha configurado una agenda para el servicio de Contralor de Conductores para la reserva de horas para la renovación de licencia de conducir; y otra en Contralor y Registro de Vehículos para reserva de hora de trámites. Asimismo, se encuentra en uso en el BPS, en donde brinda soporte a la reserva de horas de distintas dependencias.[13]

Como podemos ver tanto los sistemas descritos a nivel mundial como nacional ofrecen diversas funcionalidades, entre ellas la administración de citas médicas, historia clínica electrónica, servicios de facturación, y configuración de recursos. SAREM solo provee servicios que permitan el agendado y cancelación de consultas online, manejo de listas de esperas, manejo del parte diario y notificaciones. Tratamos de hacer un sistema que maximice la experiencia de usuario, que resulte intuitivo y práctico tanto para funcionarios de los centros de salud como para sus pacientes.

3. Especificación

SAREM fue especificado en un proceso iterativo que incluyo a futuros usuarios directos del sistema como son los estudiantes de RRMM, actualmente trabajando sobre la tesis final de sus carreras. De esta manera se buscó obtener un producto muy adaptado a la realidad profesional cuyo desempeño está destinado a mejorar la utilización de los recursos médicos, un contacto más cercano entre el profesional de salud y el paciente, y una mejor optimización del tiempo tanto para los funcionarios de los centros de salud como para sus usuarios. El análisis del problema fue abordado en forma interdisciplinaria, con las dificultades de lenguaje, diferencia de experiencias.

SAREM consta de tres tipos de usuarios: Paciente, Profesional y Administrador. Cada uno de estos usuarios posee diferentes roles y por lo tanto realizará diferentes acciones dentro del sistema.

3.1 Requerimientos Funcionales

El sistema debe permitirle a un usuario de tipo Paciente:

- Loguearse en el sistema.
- Modificar sus datos personales.
- Ver horarios de atención del médico solicitado, dada la especialidad elegida, en el lugar físico de atención elegido con una apertura de agenda de tres meses.
- Agendar turnos en consultas.
- Cancelar turnos en consultas.
- Ingresar a listas de espera de consultas.
- Borrarse de listas de espera.
- Elegir un médico de referencia en caso de no tener uno.
- Recibir notificaciones asignadas por un usuario de tipo Profesional y de índole general (Ejemplo campañas de salud masivas).
- Recibir notificaciones relacionadas a las consultas en que este suscripto.
- Poder cambiar manualmente el idioma de SAREM.

El sistema debe permitirle a un usuario de tipo Profesional:

- Loguerase en el sistema.
- Modificar sus datos personales.
- Completar parte diario.
- Ver lista de pacientes que han solicitado referenciarse con él/ella.
- Aceptar referencia como médico de cabecera.

- Denegar referencia.
- Asignar notificaciones a usuarios de tipo Paciente.
- Ver las notificaciones creadas en el sistema.
- Poder cambiar manualmente el idioma de SAREM.

El sistema debe permitirle a un usuario de tipo Administrador:

- Loguerase en el sistema.
- Modificar sus datos personales.
- Crear consultas.
- Eliminar consultas.
- Ver consultas creadas en el sistema.
- Buscar consultas.
- Crear locales.
- Eliminar locales.
- Ver listado de locales creados en SAREM.
- Crear especialidades.
- Eliminar especialidades.
- Ver listado de especialidades creadas en SAREM.
- Crear usuarios.
- Ver listado de usuarios creados en SAREM.
- Asignar roles a usuarios.
- Crear notificaciones.
- Enviar notificaciones.
- Eliminar notificaciones.
- Editar notificaciones.
- Ver listado de notificaciones creadas en SAREM.
- Ver registro de notificaciones enviadas.
- Poder cambiar manualmente el idioma de SAREM.

Muchas de las tareas especificadas para usuarios de tipo Profesional serán ejecutadas habitualmente por los propios Licenciados en RRMM.

3.2 Requerimientos No Funcionales

Además para esta aplicación hemos definido los siguientes requerimientos no funcionales:

- **Interoperabilidad:**

El sistema debe estar en capacidad de interactuar con SALUD.UY[1] a través de la herramienta de middleware seleccionada para el sistema. La interoperabilidad debe estar regida por los estándares definidos en la guía de estándares internacionales de informática en salud, proporcionada por SALUD.UY.

- **Desempeñó:**

Garantizar la confiabilidad, la seguridad y el desempeño del sistema informático a los diferentes usuarios a nivel nacional. En este sentido la información almacenada podrá ser consultada y actualizada permanente y simultáneamente, sin que se afecte el tiempo de respuesta.
SAREM debe estar en capacidad de dar respuesta al acceso de todos los usuarios y a los procesos batch con tiempo de respuesta aceptable y uniforme, en la medida de las posibilidades tecnológicas, en períodos de alta, media y baja demanda de uso del sistema. Para ello se fija una tasa de respuesta de 5000 milisegundos como máximo, independientemente del número de usuarios.

- **Disponibilidad:**

Estar disponible 99.99% o muy cercano a esta disponibilidad durante las 24 horas del día, todos los días del año. Esto se puede lograr utilizando diversas instancias con tecnologías Cloud Computing (Azure, AmazonCE) o la correcta configuración de servidores locales utilizando técnicas de Alta Disponibilidad y Recuperación de Desastres tanto a nivel de bases datos como servidores Web.

- **Expansibilidad:**

SAREM debe ser construido sobre la base de un desarrollo evolutivo e incremental, de manera tal que nuevas funcionalidades y requerimientos relacionados puedan ser incorporados afectando el código existente de la menor manera posible; para ello deben incorporarse aspectos de reutilización de componentes.
El sistema debe estar en capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, modificar o eliminar funcionalidades después de su construcción y puesta en marcha inicial.

- **Escalabilidad:**

La arquitectura del sistema debe soportar principalmente la escalabilidad horizontal. El sistema puede crecer potencialmente en cuanto a usuarios, para ello la arquitectura debe soportar el agregado de nodos, nuevas instancias de bases de datos. A diferencia de la escalabilidad vertical, la horizontal se puede lograr contratando servicios en Cloud Computing, como ser Amazon AWS o Azure. La arquitectura del sistema será diseñada teniendo en cuenta un potencial despliegue en la nube.

- **Facilidad de uso e ingreso de la información:**

El sistema debe ser de fácil uso tanto como para usuarios de tipo Paciente y Profesional. Debe ser de fácil entrenamiento y adaptación para los usuarios de tipo Profesional.

El sistema no debe permitir el cierre de una operación hasta que todos sus procesos, subprocesos y tareas relacionados, hayan sido terminados y cerrados satisfactoriamente, garantizando así la consistencia de la información ingresada por un usuario.

El ingreso de información al sistema debe diseñarse con transacciones que permitan el ingreso de los datos de forma parcial; es decir, que el tamaño de las páginas de registro (o formularios) de información sean adecuadas de acuerdo con la estabilidad de la red.

El sistema debe presentar mensajes de error que permitan al usuario identificar el tipo de error y comunicarse con el administrador del sistema.

- **Facilidad para las pruebas:**

El sistema debe contar con facilidades para la identificación de la localización de los errores durante la etapa de pruebas y de operación posterior.

- **Mantenibilidad:**

El sistema debe estar en capacidad de permitir en el futuro su fácil mantenimiento con respecto a los posibles errores que se puedan presentar durante la operación del sistema.

- **Seguridad:**

La seguridad del sistema debe estar regida por la guía jurídica de protección de datos personales dada por SALUD.UY. [1]

El acceso al Sistema debe estar restringido por el uso de claves encriptadas asignadas a cada uno de los usuarios. Sólo podrán ingresar al Sistema las personas que estén registradas, estos usuarios serán clasificados en varios tipos de usuarios (o roles) con acceso a las opciones de trabajo definidas para cada rol.

El control de acceso implementado debe permitir asignar los perfiles para cada uno de los roles identificados.

Respecto a la confidencialidad, el sistema debe estar en capacidad de rechazar accesos o modificaciones indebidas (no autorizados) a la información y proveer los servicios requeridos por los usuarios legítimos del sistema.

El sistema deberá contar con mecanismos que permitan el registro de actividades con identificación de los usuarios que los realizaron.

El sistema debe contar con pistas de auditoría de las actividades que se realizan sobre el sistema con niveles razonables para su reconstrucción e identificación de los hechos.

- **Validación de la información:**

El sistema debe validar automáticamente la información contenida en los formularios de ingreso. En el proceso de validación de la información, se deben tener en cuenta

aspectos tales como obligatoriedad de campos, longitud de caracteres permitida por campo, manejo de tipos de datos, etc.

- **Eficiencia:**

SAREM debe tener un tiempo de respuesta por transacción promedio y máximo de 2500 y 5000 milisegundos.
Debe soportar unas 200 de consultas por segundo.
El número de usuarios conectados en paralelo que el sistema debe permitir es 500.

El Número de transacciones en paralelo que el sistema debe poder alojar es 50.

4. Casos de Uso

A continuación se citan los casos de uso relativos a SAREM. Las descripciones se realizan en alto nivel, especificando en detalle los casos que así lo requieran.

4.1 Crear Usuario

Pre-Condición: Tener acceso a internet y un dispositivo por el cual poder conectarse.

Usuario: Administrador.

Descripción: El administrador deberá proveer los siguientes datos para crear un usuario en el sistema:

- Apellidos y Nombres
- Número de Documento (sin puntos ni guiones, con número verificador si es cédula)
- Teléfono; fijo y/o móvil
- E-mail
- Fecha de Nacimiento.
- Sexo
- Dirección

Al finalizar el caso de uso, se le creará al usuario una contraseña y un código de verificación que tendrá que recordar para poder operar en el sistema. Esta información se le enviará al e-mail usuario.

4.2 Login de Usuario

Pre-Condición: Ser usuario registrado en el sistema.

Usuario: Paciente, Profesional, Administrador.

Descripción: Deberá proveer sus credenciales (código de usuario y contraseña). El sistema autenticará a dicho usuario.

4.3 Modificar Usuario

Pre-Condición: El usuario debe estar autenticado en el sistema.

Usuario: Paciente, Profesional, Administrador

Descripción: Se le brinda la opción al usuario de modificar sus datos ingresados en el caso de uso Crear Usuario. Para ello el sistema despliega en pantalla los campos Nombres, Apellidos, Fecha de Nacimiento, Sexo, Correo, Contacto y Dirección. Una vez el usuario finaliza la edición de los datos, procede a presionar el botón “Guardar”, se persisten los datos y el sistema despliega una notificación alertando sobre la correcta actualización o por lo contrario si existe un error en el ingreso alerta al usuario. Ya sea por campos vacíos o datos incoherentes referentes a los campos en cuestión.

4.4 Listar Usuarios

Pre-Condición: Logueado al Sistema.

Usuario: Administrador.

Descripción: El administrador del sistema podrá acceder a la lista de usuarios registrados en SAREM que corresponden al centro médico donde trabaja. Tendrá acceso a los datos personales de estos usuarios registrados.

4.5 Crear Consulta

Pre-Condición: El usuario debe estar autenticado en el sistema.

Usuario: Administrador.

Descripción: El siguiente caso de uso está destinado a la fijación de horarios disponibles para las consultas que posteriormente serán requeridas por los usuarios.

Para ello el Sistema despliega en pantalla:

- Fecha: Fecha de realización de la consulta.
- Hora inicio: Hora y minutos de inicio de la consulta.
- Minutos turno: Minutos de duración de un turno.
- Pacientes en consulta: Cantidad de pacientes en consulta.
- Pacientes lista de espera: Cantidad de pacientes en lista de espera.
- Especialidad: Especialidad del médico de la consulta.
- Local: Lugar físico donde se realizará la consulta.
- Médico: Médico asignado a la consulta.
- Consulta reiterada: Si se habilita este radio se visualizan en pantalla dos campos *Repetir cada (días)* y *Hasta*. En el campo *Repetir cada (días)* se ingresa un número entero que indica la frecuencia con la que se debe volver a crear la consulta y el campo *Hasta* es la fecha que culmina la frecuencia de creación de consultas.
- Alertas asignación consulta: Si se habilita este radio se visualizan en pantalla dos campos *Avisos por día* y *Días confirmación*. En el campo *Avisos por día* se ingresa un número entero que indica la cantidad de notificaciones que le deben llegar a la persona por día. En *Días confirmación* también se ingresa un número entero que

indica la cantidad de días antes de que se realice la consulta en que deben empezar a enviarse las notificaciones.

Una vez ingresados los campos se procederá a continuar con la operación presionando el botón verde con la imagen del disquete. En caso de ya existir una agenda exactamente igual a la ingresada se alertara al usuario con un mensaje y la misma no se creará. De lo contrario se le informará al usuario que la creación fue exitosa.

4.7 Eliminar Consulta

Pre-Condición: El usuario debe estar autenticado en el sistema, debe existir una consulta para la fecha seleccionada.

Usuario: Administrador.

Descripción: El usuario podrá buscar la consulta deseada filtrando por Fecha, Especialidad y Medico. Una vez identificada la consulta podrá eliminarla del sistema.

Nota: Una vez eliminada una consulta, se deberá notificar a todos los pacientes agendados a la consulta sobre el cambio (Ver Caso de Uso “Notificar Paciente”).

4.8 Agendar Turno en Consulta

Pre-Condición: Logueado al Sistema. La fecha en la que se realiza este caso de uso es anterior a la fecha de finalización de la consulta.

Usuario: Paciente.

Descripción: El paciente selecciona la opción de agendar una consulta médica.

Una vez realizada esta acción podrá utilizar filtros de búsqueda sobre cualquier información relacionada a la consulta ya sea localidad, especialidad médica, nombre o número de documento del médico, fecha de inicio de la consulta, etc.

Una vez encontrada la consulta deseada podrá seleccionar el turno dentro de la misma en el que desee asistir, dentro de los turnos disponibles.

En caso de que la consulta que desee tenga su cupo completo, podrá anotarse en lista de espera si existe lugar para ingresar a la lista, y tendrá prioridad si se encuentra referenciado con el médico asignado a la consulta.

4.9 Cancelar Turno en Consulta

Pre-Condición: Logueado al Sistema. La fecha en la que se realiza este caso de uso es anterior a la fecha de finalización de la consulta.

Usuario: Paciente

Descripción: El usuario selecciona la consulta en la que tiene un turno asignado, y posteriormente elige la opción de cancelar la consulta.

4.10 Completar Parte Diario

Pre-Condición: Logueado al Sistema.

Usuario: Profesional.

Descripción: El profesional estará a cargo de ingresar el diagnóstico de un paciente, así como también confirmar su asistencia. Para ello podrá visualizar un calendario donde encontrara los pacientes que tiene asignados.

4.11 Iniciar Trámite de Médico de Referencia

Pre-Condición: Logueado al Sistema.

Usuario: Paciente.

Descripción: Se brinda la funcionalidad al paciente de poder seleccionar su médico cabecera. Para ello el usuario no debe poseer un médico referente (en caso de tenerlo se le brindará la opción de eliminar la referencia). El paciente podrá buscar un médico con el que desee referenciarse a través de los siguientes filtros: Especialidad, nombre del médico, y número de documento del profesional.

4.12 Finalizar Trámite de Médico de Referencia

Pre-Condición: Logueado al Sistema.

Usuario: Profesional.

Descripción: El usuario de tipo Profesional podrá realizar de manera ágil la finalización del trámite de médico de referencia. Podrá visualizar mediante una grilla todos los pacientes que han solicitado referenciarse con él/ella.

El usuario de tipo profesional será el encargado/a de confirmar o rechazar la solicitud realizada por el/los paciente/s.

4.13 Listar Pedidos de Referencia Pendientes

Pre-Condición: Logueado al Sistema.

Usuario: Profesional.

Descripción: Permite al profesional listar todos las solicitudes de médico de referencia que se encuentran pendientes de su aprobación. Para ello se despliega en pantalla una lista que contiene los datos personales de los pacientes que han solicitado referenciarse y aún esperan confirmación.

4.14 Ver Lista de Solicitudes de Referencia Confirmadas

Pre-Condición: Logueado al Sistema.

Usuario: Profesional.

Descripción: El profesional podrá visualizar el listado completo de las personas que solicitaron referenciarse al mismo y a las cuales ha confirmado la referencia. Para esto será de utilidad visualizar Documento, Tipo de Documento, Nombre y Apellido de los pacientes referenciados a dicho profesional.

4.15 Crear Notificación

Pre-Condición: Usuario debe estar autenticado al sistema.

Usuario: Administrador.

Descripción: El administrador del sistema podrá crear eventos de notificación generales a todos los pacientes de un centro de salud, o más específicos de acuerdo al sexo o franja etaria de los mismos.

4.16 Listar Notificaciones Paciente

Pre-Condición: Logueado al Sistema.

Usuario: Paciente.

Descripción: El paciente podrá ver todos los tipos de notificaciones que se le asignaron.

4.17 Subscripción de Notificaciones a Paciente

Pre-Condición: Logueado al Sistema.

Usuario: Profesional.

Descripción: El profesional referente de un paciente tendrá la oportunidad de asignar a sus pacientes notificaciones del sistema, dadas las notificaciones posibles para dicho paciente. Notar que estas notificaciones pueden estar destinadas a pacientes específicos (rango de edad, sexo), por lo cual se desplegarán únicamente las posibles para cada paciente referenciado.

Queda a criterio del profesional seleccionar las notificaciones específicas dado el paciente en cuestión. El profesional selecciona un paciente de la lista, y asigna las notificaciones que considere necesarios al caso.

4.18 Listar Notificaciones

Pre-Condición: Usuario debe estar autenticado al sistema.

Usuario: Administrador, Profesional.

Descripción: El sistema despliega en pantalla todas las notificaciones creadas, mostrando nombre de la notificación y mensaje de la notificación.

4.19 Envío de Notificaciones

Pre-Condición: Los usuarios a los que se le van a enviar las notificaciones tienen que estar registrados en sistema.

Usuario: Sistema.

Descripción: Se enviaran notificaciones a los usuarios de tipo paciente que se encuentren suscriptos a las mismas, en el intervalo de tiempo especificado. Las notificaciones se enviaran por E-mail y mediante la aplicación Android.

4.20 Notificaciones Relacionadas a la Consulta

Pre-Condición: Una alteración en la agenda causada por liberación de cupos en la consulta u otras causas.

Usuario: Sistema.

Descripción: Una alteración en la agenda causada por liberación de cupos en la consulta en la cual se encontraban pacientes en lista de espera, suspensión de consulta, etc. Se le notificará al Paciente vía el medio coordinado previamente (Notificación Android, Email).

Para ello el sistema detectará los cambios en la agenda, notificando en tiempo real. En la notificación para el paciente existirán algunos campos dependiendo del caso en cuestión.

Vía notificación Android/Email:

- Motivo suspensión agenda.
- Enlace a cancelar consulta o agendar una nueva.
- Ingreso al cupo de la consulta, con el turno correspondiente.

4.21 Asignación de Roles a Usuarios

Pre-Condición: El administrador debe estar autenticado en el sistema.

Usuario: Administrador.

Descripción: El administrador podrá visualizar mediante un listado todos los usuarios creados en el sistema. Tendrá la opción de asignarle a cada uno roles de Administrador, Profesional y Paciente.

4.22 Ingreso a Lista de Espera

Pre-Condición: Existe un paciente logueado en el sistema y una consulta con cupo lleno pero con cupo libre en su lista de espera.

Usuario: Sistema.

Descripción: Un usuario de tipo paciente solicita ingresar en la lista de espera de una determinada consulta. Una vez ingresado el paciente, el sistema reordena la lista de espera ordenando primero por los pacientes que están referenciados con el médico asignado y luego por fecha de ingreso a la lista de espera.

4.23 Egreso de Lista de Espera

Pre-Condición: Se deben cumplir las condiciones 1 y 2.

1) Se ha liberado o no un turno en una consulta.
2) Existe al menos un paciente en su lista de espera.

Usuario: Sistema.

Descripción:

Existen dos posibilidades en las cuales un usuario de tipo Paciente puede egresar de la lista de espera:

1) Se libera un turno en la consulta:

Se libera un cupo en una consulta que posee pacientes en su lista de espera. Al paciente que se encuentre en primer lugar en la lista se le enviara una notificación informándole que se ha liberado un cupo con determinado turno y se le preguntará si desea ocupar este lugar. Si el paciente no responde confirmando la notificación, esta se le enviara dos veces más dentro de un intervalo de tiempo especificado al momento de creación de la notificación. Si luego de enviadas estas tres notificaciones el paciente no responde en un lapso de tiempo especificado al crear el evento de notificación, este

perderá el turno y se lo eliminara de la lista. A continuación se buscara al paciente siguiente en la lista de espera repitiéndose el mismo procedimiento.

Si el paciente confirma que no desea ocupar el turno que se encuentra libre, se lo eliminara de la lista y se procederá a buscar al paciente siguiente en la lista de espera repitiéndose el mismo procedimiento de notificaciones.

Si el paciente confirma que desea ocupar el turno libre, este saldrá de la lista de espera y ocupara el turno correspondiente en la consulta. Una vez realizada esta acción, el sistema reordena la lista de espera ordenando primero por los pacientes que están referenciados con el médico asignado a la consulta y luego por fecha de ingreso a la lista de espera.

2) El usuario desea por voluntad propia salir de la lista de espera de la consulta:

El usuario selecciona la opción de abandonar la lista de espera de la consulta elegida, y el sistema lo elimina de la lista. Una vez realizada esta acción, el sistema reordena la lista de espera ordenando primero por los pacientes que están referenciados con el médico asignado a la consulta y luego por fecha de ingreso a la lista de espera.

4.24 Selección de Idioma

Pre-Condición: El usuario se encuentra logueado al sistema.

Usuario: Paciente, Profesional, Administrador.

Descripción: El usuario podrá elegir el idioma de la aplicación si desea que este en inglés o español. Por defecto el sistema se mostrara en español.

Desde el sitio web o la aplicación móvil el usuario podrá elegir el idioma manualmente, y esta elección se guardará localmente en él para futuras visitas. El sistema almacena en el perfil del usuario el último idioma utilizado, para saber en qué idioma enviar notificaciones por email y otros medios donde los textos son generados por el servidor y sin interacción del usuario.

5. Arquitectura

La arquitectura del sistema se encuentra distribuida en capas de forma no estricta. El modelo arquitectónico se basa fuertemente en la propuesta generada por el proyecto StrongLoop [14], el cual presenta como componente principal la API.
En la **Figura A1** se observa una división en módulos del Framework completo, el componente modelos (Models) hace referencia a la Capa de Negocios estructurada por modelo, en este caso cada modelo desprende una lógica de negocio particular, que en conjunción resultan en el módulo completo de la Capa de Negocios.

La capa de Acceso a Datos se elabora de forma directa utilizando el componente datasource-juggler, el cual brinda una capa de abstracción que permite el manejo invisible entre distintas fuentes de datos a utilizar. En este punto se puede trabajar con Datasources distintos para cada modelo, a tal punto de formar una colección de distintas bases específicas para cada uno.
En SAREM, se utiliza el datasource loopback-connector-postgresql y loopback-connector-mongodb, ya que para este caso se utiliza ambas bases de datos (PostgreSQL y MongoDB) para distintos modelos de la aplicación.

Esta arquitectura modular, permite cambiar la fuente de datos de forma invisible, únicamente manipulando un archivo de configuración donde define para cada modelo el conector a utilizar. Por lo tanto, se puede trabajar con una amplia gama de bases de datos específicas. En SAREM se utiliza una base de datos MongoDB para alojar sin problemas documentos relativos a consultas, los cuales se transcriben fácilmente al formato establecido por HL7 versión XML. Esto considerando que los documentos son almacenados en JSON binario (BSON) pudiendo ser exportados a JSON y transformados directamente.
Por lo tanto, la modularidad de la aplicación garantiza un nivel de abstracción alto, permitiendo agregar más modelos a medida que el sistema avanza.

La integración con sistemas externos se logra mediante conectores provistos por el Framework, como ser el mail. Se desarrollaron módulos especiales orientados a la integración con Google Cloud Messaging [15] y el uso de amqplib [16] para la integración con la Cola de mensajes, la cual permite el envío de mensajes de forma asincrónica sin realizar una estrategia de *Pooling* a la base en búsqueda de notificaciones a enviar.

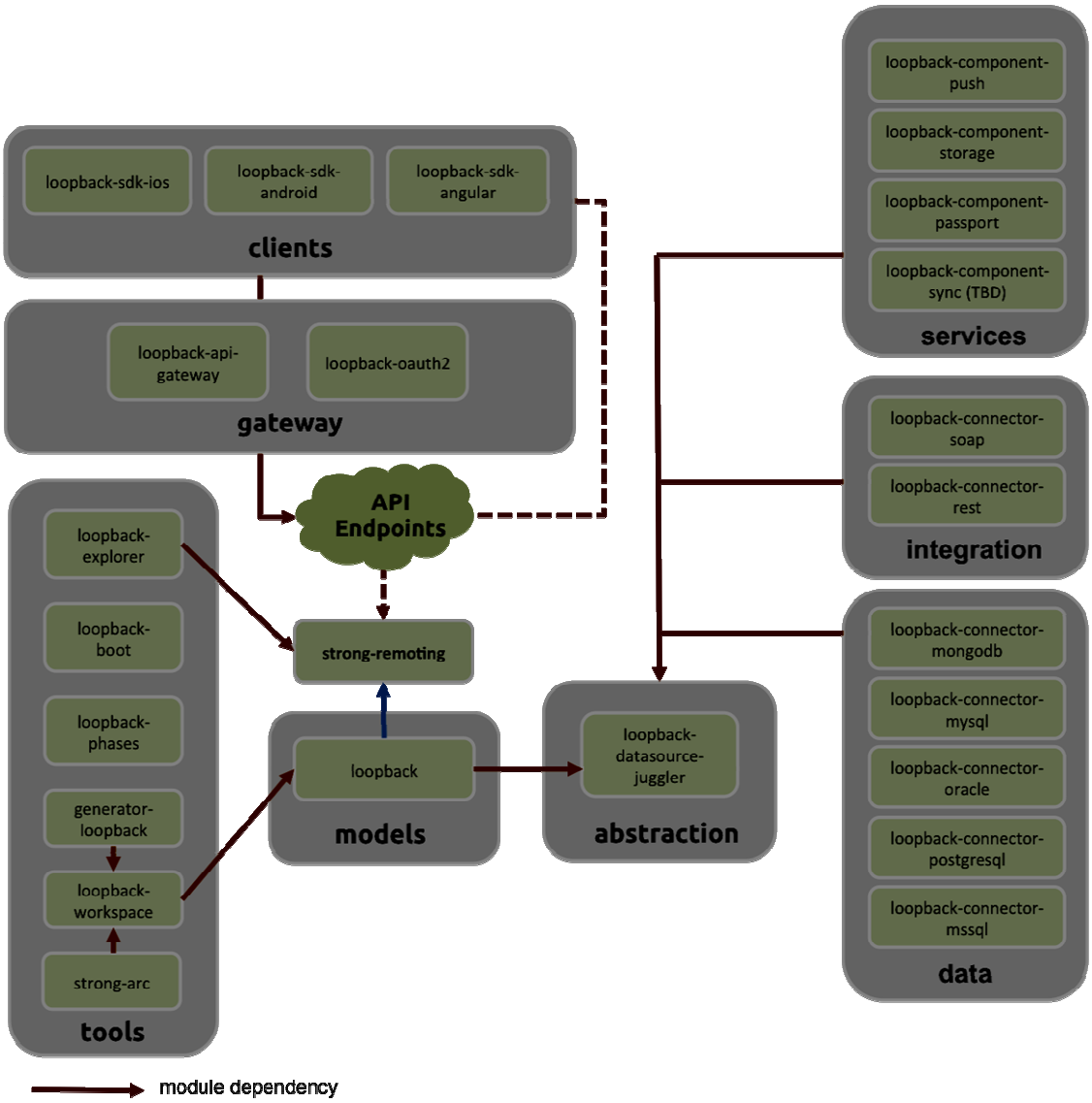


Figura A1 – División de módulos Framework StrongLoop.

5.1 Componentes Arquitectónicos

Si bien la estructura base del proyecto utiliza el concepto arquitectónico anteriormente mencionado, se realiza una descripción del núcleo principal del sistema, en el concepto anterior es un desglosamiento del módulo Modelos.

En la **Figura A2** se observa cómo se diagramaron los componentes arquitectónicos más importantes de la aplicación.

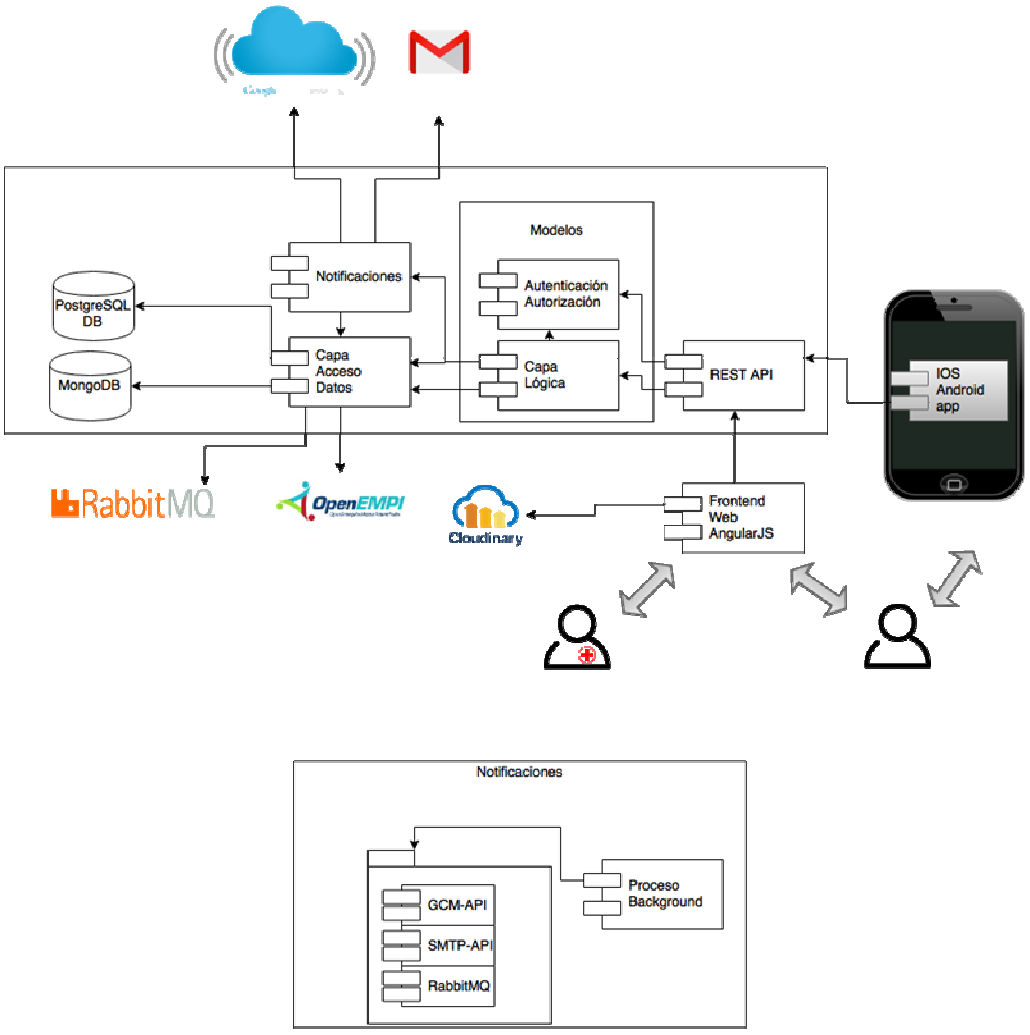


Figura A2 – Componentes arquitectónicos más importantes de SAREM.

Se presenta en la **Figura A3** un enfoque Top-Down de los componentes del sistema, desde la capa de presentación hasta los servicios de datos (Data Services).

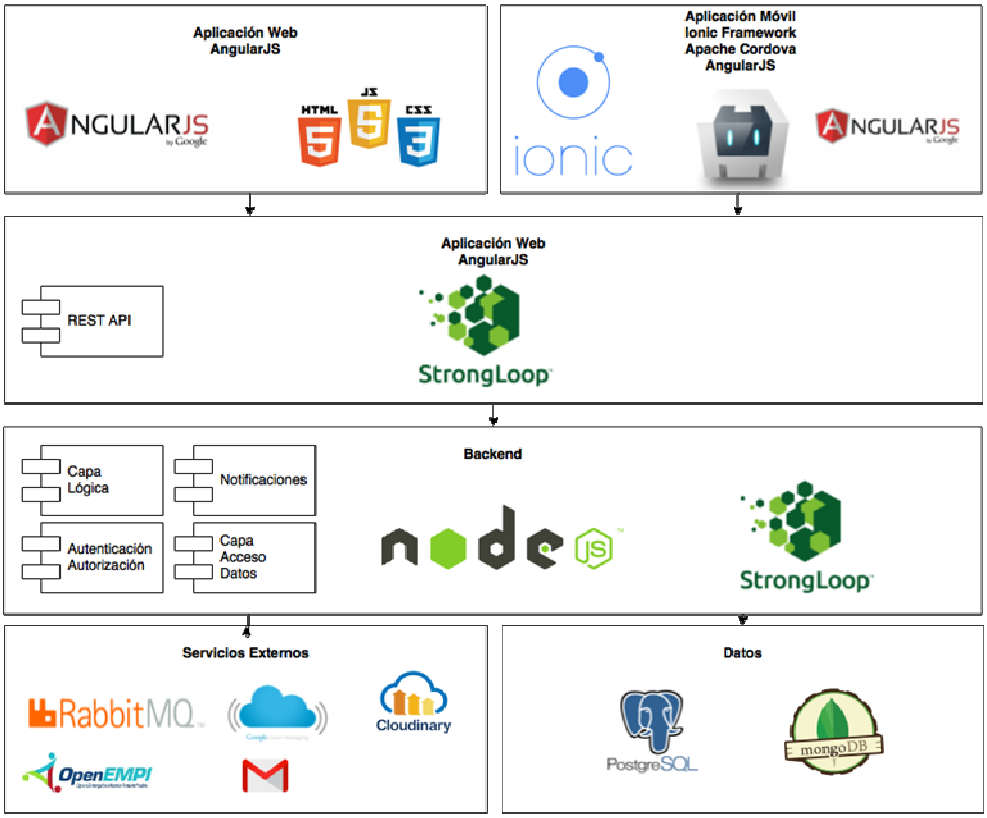


Figura A3 – Enfoque Top-Down de los componentes de SAREM.

A continuación se detalla cada componente del sistema:

5.1.1 REST API

Componente Middleware brindado por el Framework StrongLoop [14]. Se encarga de exponer las principales funcionalidades del sistema ofreciendo un AMB con operaciones básicas sobre los modelos del sistema. Esto es insertar, listar, eliminar y actualizar, además de los métodos personalizables que pueden ser expuestos con una PATH específico según sea la demanda de la lógica de negocios en cuestión.

5.1.2 Capa Lógica

Se agrupa en una serie de componentes directamente mapeados a los modelos presentes. Cada entidad en cuestión presenta un modelo particular. Por lo tanto el modelo Consulta tendrá un componente particular, se describe todos los componentes expuestos.

- **Account:** Componente orientado a la creación de cuentas de usuario, manipula permisos y roles de las cuentas de usuario del sistema.
- **Consulta**: Componente diseñado específicamente para las operaciones relacionadas al modelo Consulta, esto implica a nivel de aplicación la creación, listado y eliminación de las mismas.
- **Consulta Agenda**: Componente especializado en Consultas Agendadas, básicamente son los turnos dentro de una Consulta.
- **Consulta Espera:** Componente dedicado a Consultas en Espera, se embebe la lógica que relaciona todos los procedimientos inherentes a pacientes involucrados en consultas que se encuentran en la cola de espera.
- **Consulta Cancelar:** Las consultas canceladas tienen su componente específico, su existencia se remite a todas las operaciones que involucran la acción cancelar consulta. Por lo tanto este componente se encarga crear un evento una vez un paciente cancela una consulta, notificando a el primer paciente de la lista de espera.
- **Especialidad**: Componente orientado a la creación, listado y eliminación de especialidades del sistema.
- **Image**: Componente orientado a la generación de URL para la publicación de imágenes utilizando el servicio Cloudinary [17].
- **Local:** Componente orientado a la creación, listado y eliminación de locales.
- **Notificación:** Componente orientado a la creación de notificaciones, se encarga de publicar mensajes en la cola una vez las notificaciones son creadas. Notar que

pueden ser notificaciones en tiempo real o en diferido, indistintamente del caso, las mismas son publicadas en una cola, la cual se encarga de distribuir la carga entre distintos procesos los cuales procesan las notificaciones, es decir, envían efectivamente las mismas a los destinatarios.

- **Openempi:** Se trata de un modelo que abstrae las operaciones relacionadas al servicio externo OpenEMPI [18], las cuales implican consultas sobre pacientes, creación y actualización de los mismos.
- **Paciente Tipo Notificación:** Este módulo se encarga de activar evento de notificación ante la asignación de un evento a un Paciente realizada por un Médico.
- **País:** Módulo encargado de la creación y listado de países.
- **Persona:** Componente dedicado a la manipulación de Pacientes y Profesionales del Sistema, principalmente orientado en el tratamiento de Pacientes, en el cual se obtienen las consultas asignadas, en espera y canceladas de los mismos.
- **Referencia:** Componente destinado a la manipulación de referencias, las mismas son la relación Paciente Médico. Actualmente el componente no ofrece operaciones personalizadas, pero en futuras versiones de la aplicación, se puede crear funciones especiales que involucren específicamente el modelo.
- **Tipo Notificación**: Componente dedicado a notificaciones estáticas, brinda métodos específicos para la manipulación de las mismas.
- **Translation**: Módulo orientado a manejo de traducciones del sistema, aquí se almacenan los documentos de traducciones relativas al sistema.

5.1.3 Autenticación y Autorización

Módulo mixto que permite una capa de acceso al sistema. Generalmente conocida como ACL (Access Control Layer).

Este módulo se independiza de la lógica de negocio a tal modo que el manejo de autenticación y autorización es invisible en el proceso.

Para cada modelo se define las políticas de acceso a los mismos, para usuarios autenticados, no autenticados, dueños de objeto se pueden establecer políticas de acceso especiales. La creación de roles también reestructura el acceso a los métodos, brindando una capa en la cual reutiliza todos los métodos expuestos según el rol que el usuario posea. Nuevamente el manejo es invisible, por lo cual no debe controlarse en el método qué usuarios tienen permisos sobre el mismo.

5.1.4 Notificaciones

Componente especial, el cual reutiliza la capa de lógica de negocios del sistema.

Está constituido en esencia por un proceso consumidor de mensajes, por lo tanto se trata de un proceso en background encargado del envío real de notificaciones a través de los distintos servicios (Google Cloud Messaging, Google Mail). En el mismo se embebe la lógica de procesamiento según el tipo de notificaciones a enviar.

Según el tipo de notificación, el proceso se encarga de la distribución. Esto se planteó de esta forma por un simple motivo, escalabilidad horizontal en procesos consumidores.

Esta estrategia permite distribuir N procesos por instancia, esto quiere decir que el proceso puede ejecutarse en una infraestructura distribuida sin afectar de forma alguna el comportamiento de la API. Si bien utiliza componentes de backend para la reutilización de código, son independientes.

Por lo tanto dada una cola o un cluster de las mismas, se puede fácilmente contar con procesos que se encargan de la distribución de los mensajes. Esto permite entre otros beneficios, desacoplar la lógica del procesamiento de mensajes, por lo tanto el Backend se remite únicamente a la lógica primordial del sistema, dedicando la lógica de distribución de mensajes a estos procesos.

Este componente utiliza bibliotecas NodeJS amqplib mantenidas por la comunidad, y node-gcm [19] para el envío de Push Notifications a los dispositivos móviles.

Reutiliza los subcomponentes de la lógica de negocios de forma independiente de la API.

5.1.5 Acceso a Datos

Se utiliza para esta finalidad loopback-connector-postgresql, a modo tal que las operaciones de manipulación de datos están totalmente abstraídas según el conector para cada manejador de base de datos o servicio externo. Las operaciones sobre los modelos se traducen utilizando los conectores que se encuentras especificados de antemano por una capa de abstracción (interfaz) que define el conjunto de operaciones a definir para que se cumpla de forma correcta el uso del conector.

5.1.6 Aplicación Web

Se trata de una aplicación cliente diseñado utilizando AngularJS [20], entre otros beneficios permite el procesamiento distribuido. Se delega al cliente la renderización de datos a través de los datos obtenidos de la API REST. Esto implica que no es necesario contar con un servidor dedicado al renderizado de los datos como suele ocurrir con aplicaciones que utilizan JSF [21] o ASP.NET [22].

El beneficio arquitectónico que brinda AngularJS [20] permite escalar rápidamente sin contar con capacidad computacional, de hecho la misma se distribuye en el cliente, con lo cual únicamente es necesario contar con un servidor de contenidos. En el desarrollo del proyecto se utiliza servicios de AWS [23] para distribuir el contenido de forma distribuida, replicando en servidores de contenido la aplicación en sí misma.

La división de cliente permite obtener servicios dedicados, utilizando una Arquitectura orientada a Microservicios. Por lo cual cada componente cumple una función específica sin dependencia directa de desarrollo entre servicios.

Al tratarse de contenido estático, la aplicación AngularJS[20] permite ser descargada totalmente a un ordenador, y su renderizado se realiza mediante un navegador Web que soporte Javascript en su totalidad. Por lo cual la aplicación se ve limitada a navegadores modernos.

5.1.7 Aplicación Móvil

Desarrollada utilizando Ionic Framework [24], de uso multiplataforma (iOS, Android, Blackberry, Windows Phone), es un stack poderoso que integra Apache Cordova[25] y AngularJS [20]. Básicamente permite el desarrollo de aplicaciones móviles agrupando una capa de abstracción que permite la funcionalidad en un gran número de sistemas operativos móviles, por lo cual, se genera código para una aplicación y se despliega para todos los sistemas.

Nuevamente arquitectónicamente es una solución elegante, debido a que utiliza un stack HTML, CSS y Javascript, por lo cual el equipo de desarrollo que implementa el Frontend Web puede perfectamente realizar o reutilizar los componentes en la versión móvil.

Apache Cordova [25] provee plugins para todas las plataformas, de modo tal que simplifica el desarrollo en caso contrario de desarrollar una aplicación por plataforma, donde el costo de mano de obra suele ser alto ya que implica potencialmente varios equipos de desarrollos.

Al orientarse en dispositivos móviles cumple un conjunto de casos de uso de la

aplicación web, aunque fácilmente se podría proveer todos y con capa de persistencia local, el cual es el principal beneficio del desarrollo de este tipo de aplicaciones.

Se muestra en la **Figura A4** la arquitectura de Apache Cordova [25]. Lo que se debe apreciar es la modularización que brinda el proyecto, a tal punto que se escriben pequeños servicios de forma nativa para cada plataforma y se reutilizan utilizando una API, de esta forma se abstrae de la plataforma permitiendo al desarrollador generar vistas para todas las plataformas en paralelo.

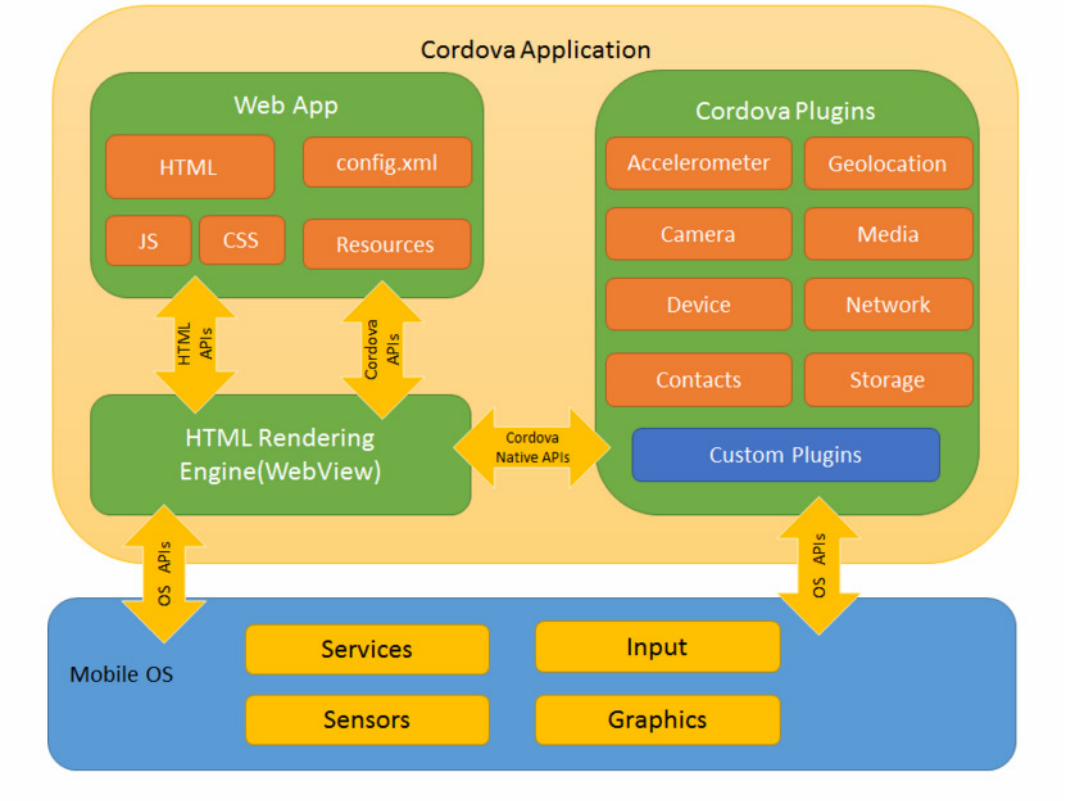


Figura A4 – Arquitectura Apache Cordova.

En la implementación únicamente se debe remitir al desarrollo del componente Web App de la **Figura A4**.

5.2 Modelo Base de Datos

El modelo de la base de datos se genera de forma automática debido a la utilización de Loopback [26] como ORM, por lo cual el diagrama de clases coincide de igual forma que el modelo presentado en la **Figura A5**.

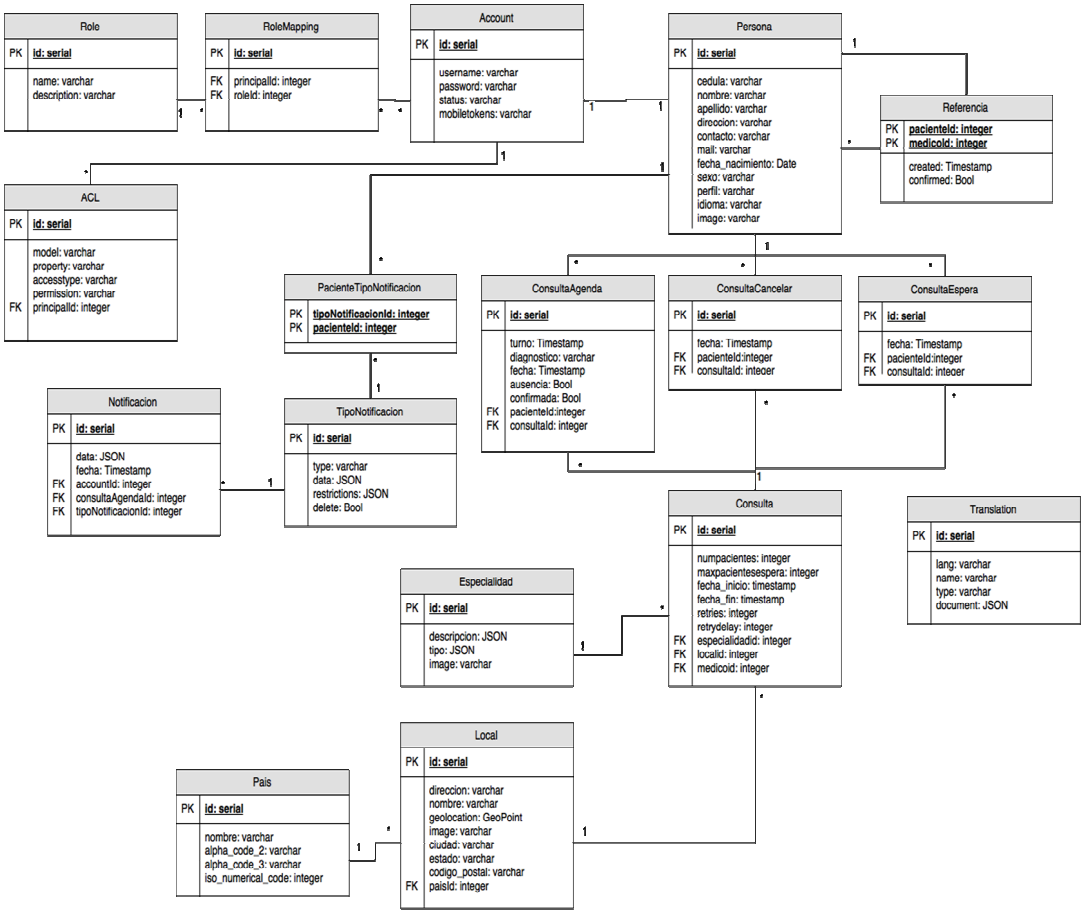


Figura A5 – Modelo de Datos.

5.3 Diagrama de Despliegue

En el despliegue de SAREM se utilizó como infraestructura Amazon Web Services [23]. En la **Figura A6** se muestran los principales servicios utilizados, su funcionalidad y como pueden ser sustituidos si se utiliza una infraestructura local.

El hito principal del despliegue es reflejar que SAREM posee el requerimiento no funcional de escalabilidad horizontal. Con los servicios adecuados, la misma puede escalar en función de la demanda actual o de forma manual. Dependiendo fundamentalmente de la infraestructura que se posee, estos procedimientos pueden ser complejos de resolver, sin embargo al utilizar servicios provistos por AWS, la complejidad se solventa de forma inmediata.

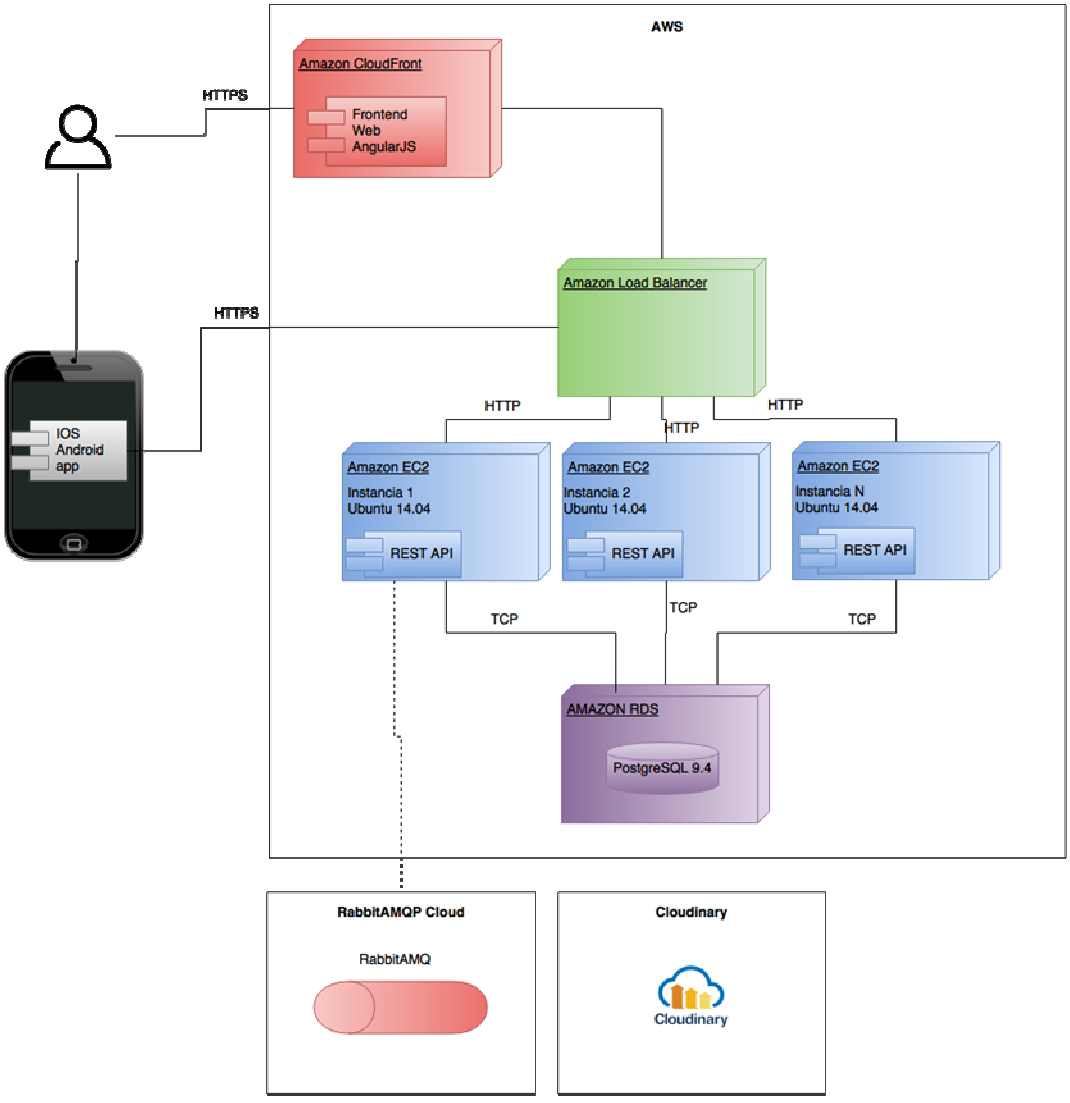


Figura A6 – Diagrama de Despliegue.

- **Amazon RDS (Relational Data Service):** Servicio orientado a proveer bases de datos relacionales. Para el desarrollo se utilizó PostgreSQL como manejador de base de datos. El servicio permite escalar verticalmente en la infraestructura brindada (CPU, RAM, Storage). También brinda servicios de alta disponibilidad con replicación de solo lectura, con esto se logra escalar horizontal en lectura. Ofrece también gestión para respaldos automáticos, disponibilidad por zonas geográficas, entre otros beneficios.

- **Amazon CloudFront:** Se trata de un CDN (Content Delivery Network), en conjunto con S3 para el almacenamiento se puede lograr una técnica performante para el cliente Web, ya que el contenido es estático, la mejor opción es dedicar exclusivamente las instancias EC2 para el despliegue del backend y la API REST. Delegando de esta forma a CloudFront para el almacenamiento y distribución del cliente Web. Como el contenido es estático, utiliza un CDN es beneficioso, puesto que se replica geográficamente en servidores de Amazon, brindando una tasa de transferencia óptima sin penalizar la performance del Backend.

- **Amazon EC2:** Se trata de instancias de máquinas virtuales, se trata de un servicio de infraestrucutra (IAAS). Es responsable de la instanciación del backend y la API REST, por lo cual el poder computacional es dedicado para dicha funcionalidad. Como se tratan de máquinas virtuales, se puede instanciar a demanda utilizando Amazon Beanstalk. Para el despliegue de SAREM, se opta por instanciaciones de forma manual.

- **Amazon Load Balancing:** Con un balanceador de carga se puede escalar horizontalmente rápidamente según la demanda actual. Se encarga de distribuir las peticiones a la API hacia las instancias EC2 anteriormente descriptas. Ofrece alta disponilidad y a su vez se configura certificados de seguridad para conexiones seguras y cifradas, quitando la responsabilidad de configurar los certificados en cada instancia EC2.

- **RabbitAMQ Cloud:** Servicio externo a Amazon que brinda instancias de RabbitMQ, dedicado a servicio de mensajería utilizando una estrategia asincrónica para la entrega de mensajes Push y emails. Se encarga del almacenamiento y distribución de mensajes a los procesos encargados de distribución de mensajes (background process). De esta forma se elimina la necesidad de realizar poling a una base de datos, únicamente manteniendo una conexión TCP de larga duración con el servicio y manejando el problema de las notificaciones de forma escalable y eficiente.

- **Cloudinary:** Un componente sustituible pero para el caso de SAREM es utilizado para el almacenamiento gratuito de imágenes de perfiles de usuario, locales y especialidades. Se trata de un CDN orientado a la manipulación de imágenes. Por lo

cual, el cliente en sí mismo se encarga de realizar el envío y descarga de imágenes utilizando el ancho de banda de un servicio externo.
Como alternativa se puede utilizar Amazon S3 para el almacenamiento, o servidores locales, se trata de un componente sustituible.

6. Comportamiento del Sistema

A continuación se especifica la lógica de negocio de SAREM para los casos de uso más importantes.

6.1 Cómo se agendan los usuarios

Un usuario puede agendar su consulta utilizando filtros de especialidad, local en que desea atenderse, médico y fecha de inicio. O simplemente puede buscar todas las consultas que existen en el sistema que no hayan finalizado al momento de realizar la acción. Al usuario se le desplegaran todas las consultas que satisfagan los filtros seleccionados y que tengan turnos disponibles o cupos libres en su lista de espera.

Una vez que encuentre la consulta que desea, si esta posee turnos libres, procederá a seleccionar el turno disponible que le quede mejor y seleccionará la opción de agendarse.

Por más información ver caso de uso **4.8 Agendar Turno en Consulta**.

6.2 Cómo ingresan los usuarios a la lista de espera de una consulta

Esta situación se da cuando el usuario desea agendarse a una consulta que no tiene turnos disponibles, pero si cupos libres en su lista de espera. El sistema le da la opción de ingresar a la lista de espera de la consulta. Una vez ingresado el paciente, el sistema reordena la lista de espera ordenando primero por los pacientes que están referenciados con el médico asignado y luego por fecha de ingreso a la lista de espera.

Por más información ver caso de uso **4.22 Ingreso a Lista de Espera**.

6.3 Cómo egresan los usuarios de la lista de espera de una consulta

Existen dos posibilidades en las cuales un usuario de tipo Paciente puede egresar de la lista de espera:

1) Se libera un turno en la consulta:

Se libera un cupo en una consulta que posee pacientes en su lista de espera. Al paciente que se encuentre en primer lugar en la lista se le enviara una notificación informándole que se ha liberado un cupo con determinado turno y se le preguntará si desea ocupar este lugar. Si el paciente no responde confirmando la notificación, esta se le enviara dos veces más dentro de un intervalo de tiempo especificado al momento de

creación de la notificación. Si luego de enviadas estas tres notificaciones el paciente no responde en un lapso de tiempo especificado al crear el evento de notificación, este perderá el turno y se lo eliminara de la lista. A continuación se buscara al paciente siguiente en la lista de espera repitiéndose el mismo procedimiento.

Si el paciente confirma que no desea ocupar el turno que se encuentra libre, se lo eliminara de la lista y se procederá a buscar al paciente siguiente en la lista de espera repitiéndose el mismo procedimiento de notificaciones.

Si el paciente confirma que desea ocupar el turno libre, este saldrá de la lista de espera y ocupara el turno correspondiente en la consulta. Una vez realizada esta acción, el sistema reordena la lista de espera ordenando primero por los pacientes que están referenciados con el médico asignado a la consulta y luego por fecha de ingreso a la lista de espera.

2) El usuario desea por voluntad propia salir de la lista de espera de la consulta:

El usuario selecciona la opción de abandonar la lista de espera de la consulta elegida, y el sistema lo elimina de la lista. Una vez realizada esta acción, el sistema reordena la lista de espera ordenando primero por los pacientes que están referenciados con el médico asignado a la consulta y luego por fecha de ingreso a la lista de espera.

Por más información ver caso de uso **4.23 Egreso de Lista de Espera**.

6.4 Recordatorios y Confirmaciones

El envío de recordatorios de consultas y rutinas, se realiza un determinado tiempo X antes de la fecha y hora del evento, donde X es un valor configurado en el sistema. Además, para el caso de notificaciones a dispositivos móviles, se cuenta con la opción de que al recibir la notificación, el dispositivo avise al sistema la correcta recepción de esta. Por último, tanto desde la web y dispositivos móviles, se tiene la opción de que el usuario confirme su asistencia al recibir el recordatorio (para el caso de recordatorio a través de email, el usuario deberá ingresar a SAREM y confirmar la asistencia). Entonces, con la indicación de notificación recibida, y la confirmación de asistencia, se puede tener un seguimiento más fino del usuario para determinar si este olvidó o no el evento. Inicialmente estos datos se utilizan únicamente para realizar un seguimiento manual de los usuarios, pero queda como trabajo a futuro, realizar optimizaciones adicionales a partir de estos datos mediante procesos automáticos.

Por más información ver casos de uso **4.19 Envío de Notificaciones** y **4.20 Notificaciones Relacionadas a la Consulta**.

6.5 Turnos cancelados

El usuario podrá cancelar su turno asignado en una consulta, siempre y cuando la consulta no haya finalizado.

Por más información ver caso de uso **4.9 Cancelar Turno en Consulta**.

6.6 Selección de idioma

El usuario podrá elegir el idioma de la aplicación si desea que este en inglés o español. Por defecto el sistema se mostrara en español.

Desde el sitio web o la aplicación móvil el usuario podrá elegir el idioma manualmente, y esta elección se guardará localmente en él para futuras visitas. El sistema almacena en el perfil del usuario el último idioma utilizado, para saber en qué idioma enviar notificaciones por email y otros medios donde los textos son generados por el servidor y sin interacción del usuario.

Por más información ver caso de uso **4.24 Selección de Idioma**.

6.7 Asignación de notificaciones por parte de un Profesional a un Paciente

El profesional referente de un paciente tendrá la oportunidad de asignar a sus pacientes eventos de notificación del sistema, dados los eventos posibles para dicho paciente. Notar que estos eventos pueden estar destinados a pacientes específicos (rango de edad, sexo), por lo cual se desplegarán únicamente los posibles para cada paciente referenciado.

Queda a criterio del profesional seleccionar los eventos específicos dado el paciente en cuestión. El profesional selecciona un paciente de la lista, y asigna los eventos que considere necesarios al caso.

Por más información ver caso de uso **4.17 Subscripción de Notificaciones a Paciente**.

7. Implementación

7.1 Elección de entorno de desarrollo

En principio estuvimos en la disyuntiva de si utilizar Java o .NET para realizar este proyecto, pero finalmente optamos por .NET, ya que permite crear aplicaciones robustas, seguras y de alta calidad, además de que el tiempo de desarrollo comparado con frameworks similares utilizando Java es menor. Sumado a esto teníamos más experiencia realizando aplicaciones web en .NET.[27][28]

Empezamos este proyecto utilizando tecnologías que el stack de Microsoft nos brindaba. Por ejemplo utilizamos ASP .NET 4.5 MVC [22] para realizar el frontend de la aplicación, junto con bibliotecas de JavaScript como JQuery [29] y Twitter Bootstrap [30]. Para el diseño de la aplicación utilizamos Twitter Bootstrap también.

En lo que refiere al backend de la aplicación utilizamos el lenguaje de programación C#, y elegimos utilizar ADO .NET Entity Framework [31] como ORM [32] ya que es un framework que permite a los desarrolladores trabajar con datos relacionales de manera más sencilla ya que automatiza las actividades de manipulación de la base de datos. Por otra parte al elegir utilizar el stack de tecnologías Microsoft nos vimos obligados a utilizar Microsoft SQL Server[33] como sistema de manejo de base de datos. Lo que trae como desventaja licencias costosas, además de que posee una compatibilidad limitada. Microsoft SQL Server sólo está diseñado para ejecutarse en servidores basados en Windows. Por varias razones, incluyendo los costos de licencias y los problemas de seguridad, los desarrolladores pueden optar por alojar sus sitios web en los equipos basados en Unix. Por lo tanto no podrán utilizar SQL Server. Además de esto, pueden surgir problemas de compatibilidad con respecto a aplicaciones que se ejecuten en otras plataformas. Si bien nuestra idea era que SAREM se ejecutara en la nube también queríamos darles a los usuarios la posibilidad de que pudieran ejecutar la aplicación utilizando sus servidores locales si así lo deseaban.

Continuamos nuestro desarrollo utilizando el stack de tecnologías Microsoft, pero sabiendo que era una desventaja utilizar SQL Server como manejador de base de datos. Por eso analizando la situación en la cual nos encontrábamos decidimos migrar el backend y utilizar tecnologías open source para mitigar el problema de compatibilidad entre el manejador de base de datos y los servidores Unix.

SAREM necesitaba un framework de desarrollo seguro, escalable, respaldado por una comunidad grande de desarrolladores que pudiera ser utilizado en escenarios de producción de gran escala. Java era una opción obvia, pero como mencionamos anteriormente tiene sus desventajas. Pricipalmanente relacionadas con su velocidad de

desarrollo y la carga de los gastos generales de legado, hacen de esta una opción cuestionable cuando se trata de desarrollos web modernos y dinámicos.

Necesitábamos un framework que fuera open source y nos permitiera trabajar de manera productiva y con una curva de aprendizaje baja. Investigando las diversas posibilidades existentes decidimos implementar SAREM con Node.js [34]

A continuación presentamos un estudio comparativo entre ASP.NET 4.5 y Node.js, que justifica la elección de Node.js como framework de desarrollo.

7.1.1 Comparación entre ASP.NET 4.5 y Node.js

7.1.1.1 Modelos de Procesamiento

La principal diferencia entre los frameworks ASP.NET 4.5 y Node.js es su modelo de procesamiento. Node.js utiliza un estilo de procesamiento asíncrono mientras que ASP.NET le ofrece al desarrollador la posibilidad de elección.

Node.js fue construido en base a un modelo asíncrono desde su creación mientras que la palabra clave *async* apareció por primera vez en ASP.NET MVC 4. Las **Figura I1** y la **Figura I2** demuestran las diferencias entre modelos sincrónicos y asincrónicos para una aplicación web.

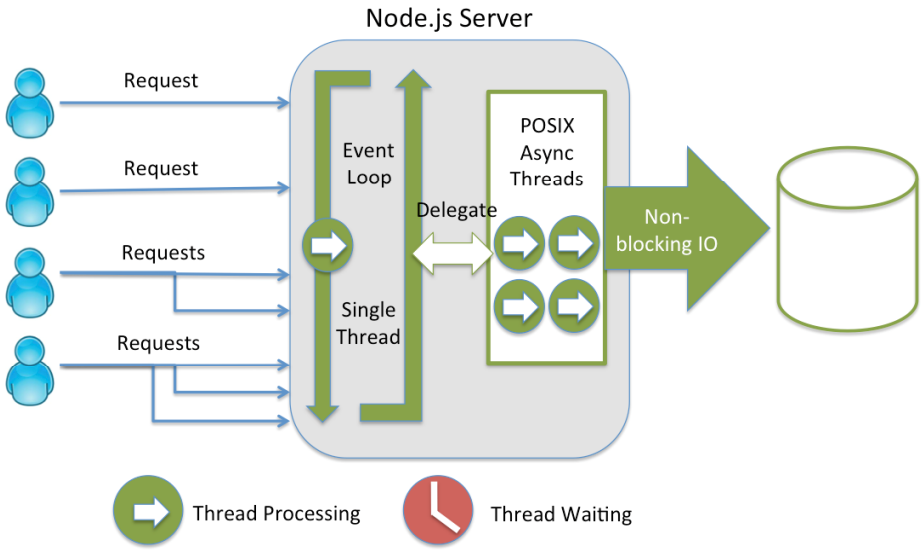


Figura I1 – Node.js Server

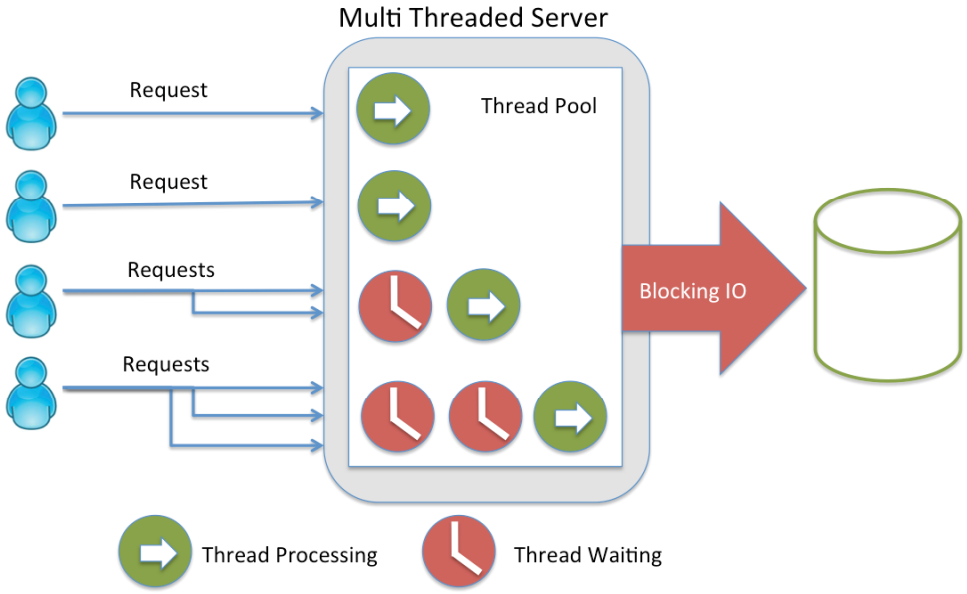


Figura I2 – Multi Threaded Server

Como muestra la **Figura I1**, Node.js utiliza un solo hilo (thread) para el manejo de solicitudes y muchos hilos (en realidad, no son hilos realmente) para proporcionar entrada y salida no bloqueante (a la base de datos o a otro servidor REST). Menos hilos significan menor utilización de memoria y un uso más económico de CPU.

El diagrama de la **Figura I2** muestra un servidor multi hilo que puede ser encontrado, por ejemplo en Java. En este modelo el servidor genera un nuevo hilo para el manejo de cada solicitud que se duerme en el bloqueo de las operaciones de entrada/salida que consumen recursos de CPU y memoria.

ASP.NET no utiliza un único hilo, sino que el uso está restringido por el número de solicitudes concurrentes en el pool de hilos y encola las solicitudes en este. Los hilos pueden ser terminados en operaciones asíncronas como en Node.js. De todas formas, el modelo de procesamiento de ASP.NET es más propenso al cambio de contexto que implica costes adicionales de CPU. Más que eso, ASP.NET y .NET no fueron diseñados teniendo la programación asíncrona en cuenta; algunas bibliotecas todavía no tienen soporte para esta u ofrecen un “falso soporte”.

[35][36][37]

7.1.1.2 Lenguajes de Programación

ASP.NET utiliza C# como su lenguaje principal. Node.js utiliza JavaScript y todos los lenguajes que pueden ser compilados a este como CoffeeScript [38], Microsoft TypeScript [39] y ES6 [40].

Sin ninguna duda C# es un lenguaje más poderoso que JavaScript. C# es un lenguaje tipado, que brinda comprobación de errores en tiempo de compilación. En JavaScript se puede obtener algo similar utilizando TypeScript [39].

C# tiene el modelo clásico de herencia, las clases propuestas por EcmaScript6[40] ofrecen una nueva sintaxis para la herencia basada por prototipo propia de JavaScript, que tiene sus ventajas y peculiaridades sobre la herencia clásica.

Sin embargo, algunos desarrolladores temen el sistema de patentes de .NET.[41][42]. Por otro lado la comunidad no siente lo mismo con respecto a las patentes de software de JavaScript porque es open source.

JavaScript es un lenguaje más ubicuo, aunque para dominar Node.js también se requiere que el desarrollador este familiarizado con el estilo de programación asíncrona.

7.1.1.3 Soporte de Programación Asíncrona

Ambos lenguajes, C# y JavaScript soportan las programación asíncrona.

C# utiliza el poder de sus palabras clave *async* y *await*.

Como pionero Node.js ha propuesto varios enfoques para la programación asíncrona.

- Utilizar callbacks:
  De esta manera el código se convierte difícil de leer, y el flujo de control se torna complicado de entender. Pude aparecer el problema denominado “callback hell”[43]. Este fue el primer enfoque que surgió.

- Utilizar la palabra clave *yield*, co-rutinas, generadores y Promesas permiten que el código sea fácil de leer y evitan el problema de tener varios callbacks anidados. Este enfoque está demostrado por el framework Koa.js[44]

- Las palabras clave *await* y *async* son parte del estándar EcmaScipt7 que aún se encuentra en fase experimental. Actualmente puede ser utilizado con el uso del compilador de JavaScript Babel [45].

En resumen, el modelo asíncrono de C# es más claro. Node.js es ligeramente diferente sin embargo:

- Obliga al desarrollador a escribir el código de forma asíncrona, por lo tanto la mayoría de las bibliotecas soportan el modelo asíncrono.
- Existe un menor número de cambios de contexto debido a su modelo de procesamiento.

7.1.1.4 Curva de Aprendizaje

Al ser JavaScript un lenguaje dinámicamente tipado es más fácil de aprender, pero en Node.js para desarrollar código de alta calidad, finalmente el desarrollador tendrá que saber utilizar Promesas, generadores y co-rutinas. Nuevos tipos de sintaxis especificados en EcmaScript6 y EcmaScript7 agregan más formas de expresar lo mismo haciendo que el código no desarrollado recientemente resulte menos legible para los desarrolladores que recién se encuentren aprendiendo.

Tabla I1 – Curva de Aprendizaje JS/Node.js vs C#/ASP.NET 4.5

Características relacionadas	JS/Node.js	C#/ASP.NET 4.5
DevTools	Cualquier editor, terminal	Visual Studio, GUI menos flexible
Deploying server	Fácil	A través de GUI, menos flexible
Async programming	Difícil	Fácil

7.1.1.5 Performance

- ASP.NET MVC 4/Web API vs Node.js

Desde luego, hay tareas donde ASP.NET supera a Node.js y hay tareas en las que ASP.NET pierde.

Node.js fue desarrollado como una plataforma de un solo hilo, donde el proceso de Node.js tiene un solo hilo que se encarga de todas las peticiones web entrantes cuya manipulación está programada en función del orden en que llegan las respuestas.

La naturaleza de utilizar un único hilo es una decisión de diseño deliberada por los creadores de Node.js para evitar el uso de semáforos y/o mecanismos de exclusión mutua que en última instancia introducen complejidad al código y errores difíciles de depurar. Sin embargo mientras se mantiene la simplicidad en el código, el hecho de utilizar un único hilo no aprovecha el paralelismo disponible de los sistemas actuales multi core y multi socket. Por ejemplo, cuando una aplicación Node.js corre en un procesador con 16 cores sin tomar ningún tipo de consideración especial para las arquitecturas multi core, solo un único core será utilizado mientras que los otros 15 permanecerán *idle*.

Para que Node.js pueda tomar ventaja de los sistemas multi core, se tienen dos opciones principales. La primera opción es dejar que ocurra la asignación de recursos a nivel de sistema en el que las peticiones entrantes se distribuyen en múltiples procesos Node.js de un solo hilo cada uno ejecutándose en una máquina virtual asignada a un único core de un procesador multi core. Todas las instancias viven detrás de un proxy que sirve para balancear las peticiones entrantes a los procesos Node disponibles. Este es un buen enfoque y funciona muy bien, siempre y cuando los costes de virtualización se consideren aceptables.

Sin embargo, la comunidad de Node.js ha presentado una solución mucho mejor. Este enfoque utiliza un módulo que habilita realizar clustering en Node.js que ejecuta muchos procesos de un solo hilo dedicados bajo un proceso maestro Node.js sin la necesidad de elaborar una infraestructura de máquina virtual. Esto se traduce en mejoras significativas de performance sin los costos asociados a la virtualización.[46]

Para obtener un buen rendimiento en tareas de Entrada/Salida costosas ASP.NET debe utilizar las palabras clave *async/await*.

Las palabras clave *async/await* se utilizan para soportar la asíncronisidad en C# lo que permite no tener que utilizar hilos directamente en el código, lo que mejora la legibilidad del mismo.

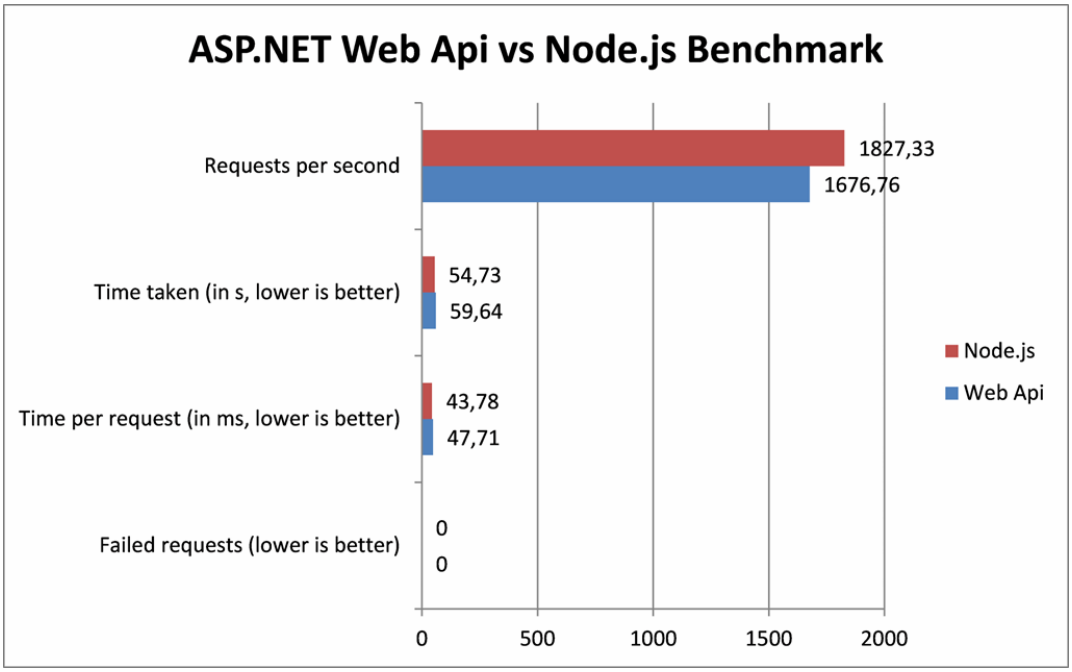


Figura I3 – ASP.NET Web API y Node.js para leer de forma asíncrona el cuerpo de una petición POST.

Los datos de la **Figura I3** fueron obtenidos fruto de realizar test de performance a dos servidores sencillos realizados en Node.js y en ASP .NET Web API, utilizando Apache ab tool[47]. Los servidores aceptan una petición POST y luego responden con los datos obtenidos del cuerpo de la petición.

- **Implementación Node.js**

```
var express = require('express')
    , app = express.createServer();

app.use(express.bodyParser());

app.post('/', function(req, res){
    res.send(req.body);
});

app.listen(8080);
```

- **Implementación ASP .NET Web API**

```
public class ValuesAsyncController : ApiController
{
    public async Task<string> Post()
    {
        return await
this.ControllerContext.Request.Content.ReadAsStringAsync();
    }
}
```

Se utilizó Apache ab tool[47] para probar la performance de las plataformas. Las configuraciones utilizadas fueron las siguientes:

- Número total de peticiones: 100000
- Concurrencia: 80

El JSON (estará guardado en test.dat) con el que se realizara las pruebas es el siguiente:

```
{
     "firstName": "John",
     "lastName" : "Smith",
     "age"       : 25,
     "address"   :
     {
         "streetAddress": "21 2nd Street",
         "city"           : "New York",
         "state"          : "NY",
         "postalCode"    : "10021"
     },
     "phoneNumber":
     [
         {
           "type"   : "home",
           "number": "212 555-1234"
         },
         {
           "type"   : "fax",
           "number": "646 555-4567"
         }
     ]
 }
```

El comando utilizado para correr el test de performance es el siguiente:

```
ab –n 100000 –c 80 –p .test.dat –T
'application/json; charset=utf-8' http://localhost/
```

El test de performance fue ejecutado tres veces y el mejor resultado para cada plataforma fue seleccionado. La diferencia de performance entre los resultados de los tests fue mínima.

- **Entorno de pruebas**:

Las pruebas fueron ejecutadas sobre un servidor Windows 2008 R2, alojado en una instancia c1.medium Amazon EC2.

- Especificaciones de la instancia:
  - Memoria 1.7 GB
  - 5 EC2 Compute Units (2 virtual cores)
- Versiones:
  - Node.js: 0.6.19
  - ASP.NET Web API 4.5

La explicación de porqué Node.js asíncrono es más rápido que ASP.NET asíncrono puede ser debido a que utiliza un número menor de cambios de contexto.

En los casos donde Node.js no se encuentre en un cluster o no tome ventaja de la asíncronisidad puede llegar a perder ante ASP .NET.

7.1.1.6 Comparación General

Tabla I2 – Comparación General JS/Node.js vs C#/ASP.NET 4.5 [48][34][49][50]

Características	JS/Node.js	C#/ASP.NET 4.5
Lenguajes	JS pierde	C# gana
Paradigmas	Menor soporte	Mayor soporte
Performance E/S	Gana	Pierde
Programación asíncrona	Requerida, sintaxis complicada	Soportada
Portabilidad	Multiplataforma	Windows
Dependencia	Gana, soporte multi-plataforma	Pierde, dependencia en Windows y Visual Studio
Fiabilidad	No es robusto	Gana
Ecosistema	Moderno y amplio	Robusto, Nuget puede fallar
Curva de aprendizaje	Fácil para empezar, aun no maduro	Maduro, pero abstracto
Simplicidad	Simple a excepción del manejo de Promesas y particularidades del lenguaje	Server deployment, NuGet
Costo	Gana, es libre	Freemium
Flexibilidad	Gana	MVC, Web API son difíciles de adaptar
Escalabilidad	Mejor para operaciones de Entrada/Salida	Mejor para operaciones de cargar de CPU

7.1.1.7 Justificación de Elección de Node.js

Decidimos migrar el stack de tecnologías de Microsoft para utilizar tecnologías open source debido a que al utilizar este tipo de tecnologías, nos veíamos ligados a utilizar SQL Server como manejador de base de datos que solo se puede ejecutar en servidores Windows.

Además al participar en el evento Ingeniería deMuestra 2015 recibimos críticas por utilizar tecnologías privativas.

Nosotros estábamos familiarizados con JavaScript, cuando investigamos sobre la existencia de Node.js pensamos que sería una muy buena idea ya que podríamos trabajar con un solo lenguaje de programación del lado del cliente y del servidor, lo que aumentaría nuestra productividad ya que nos permitiría reutilizar bibliotecas.

En la Tabla I2 se puede ver que JS/Node.js presenta varias ventajas sobre C#/ASP.NET 4.5.

7.1.2 Elección de framework para el front-end de la aplicación

Una vez tomada la elección de Node.js, también debíamos elegir un framework open source para migrar el frontend y que se integrara fácilmente con el backend escrito en Node.js. En la actualidad uno de los frameworks más importantes para el desarrollo frontend de aplicaciones web es Angular.js[20]. AngularJS, o simplemente Angular, es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página (SPA). Una de las principales razones por la que elegimos utilizar este framework es porque está construido y mantenido por ingenieros de Google [51]. Debido a esto podemos estar seguros de que estamos tratando con código eficiente y confiable. Además de que provee mecanismos sencillos para comunicarnos con el servidor desarrollado en Node.js. Con tan solo una línea de código JavaScript, se puede hablar de forma rápida con el servidor y obtener los datos necesarios para interactuar con las páginas web de SAREM.

7.1.3 Elección de framework para el desarrollo de aplicación móvil

Para realizar la parte mobile de la aplicación decidimos utilizar Ionic [24][52]. Ionic es un SDK open source completo para el desarrollo de aplicaciones híbridas. Construido

encima de AngularJS [20] y Apache Cordova [25], Ionic ofrece herramientas y servicios para el desarrollo de aplicaciones móviles híbridas utilizando tecnologías web como CSS, HTML5, y Sass [53].

Sus principales ventajas son:

- Un único código fuente para todas las plataformas soportadas (mayoritariamente Android y iOS).
- El desarrollo principal se realiza en HTML, CSS y JS.
- Permite utilizar AngularJS, framework que esta embedido en Ionic.

Su principal desventaja es:

- Menor performance en comparación con desarrollos nativos. Esto no significa un problema a menos que se quieran realizar juegos o aplicaciones de alta carga, aquellas que utilizan una gran cantidad de recursos.

Ya que nosotros no teníamos experiencia realizando aplicaciones móviles nativas, y si en desarrollo web, encontramos Ionic como una excelente opción, además de que podíamos reutilizar gran parte del código realizado para la aplicación web de SAREM en Angular.js[20]. Decidimos utilizar Ionic para construir una aplicación Android para agendar consultas y recibir notificaciones.

7.2 Elección de framework para implementar REST API

Decidimos comunicar el frontend y el backend de la aplicación utilizando una REST API[54].

Dentro de Node.js existen diversos frameworks que se pueden utilizar para crear una REST API, entre ellos uno de los más conocidos es Express.js [55]. En un principio pensamos utilizar este framework ya que es muy popular dentro de la comunidad de desarrolladores, además de que la curva de aprendizaje es pequeña y tiene la gran ventaja de que es muy personalizable. Pero luego nos dimos cuenta que la flexibilidad que brinda también trae como consecuencia varias desventajas [56]:

- Todos los endpoints se deben crear manualmente, lo que provoca que se tenga que repetir código.
- Se necesitan probar todos los endpoints, para corroborar su funcionamiento.

- La refactorización se vuelve tediosa, porque todo necesita ser actualizado en todos lados.

- No sigue un “estándar”, se debe descubrir el enfoque a seguir.

Por lo tanto decidimos buscar otro framework, investigando encontramos a LoopBack [26][57]. El cual es un framework open source desarrollado por la compañía Strongloop [14], una de las mayores contribuyentes en el desarrollo de Express.js y Node.js. LoopBack está construido encima de Express.js. Está optimizado para la construcción de APIs para móviles, web, y otros dispositivos. Se puede conectar a múltiples fuentes de datos y permite escribir la lógica de negocio en Node.js. Utilizando este framework nos libramos de las desventajas de Express.js y optimizamos el tiempo de desarrollo.

7.3 Elección de manejador de bases de datos

Por otra parte, decidimos elegir un sistema de gestión de bases de datos relacional para almacenar los datos de SAREM, el sistema elegido fue PostgreSQL [58]. Decidimos utilizar un sistema de gestión de bases de datos relacional, por las siguientes razones:

- El modelo de datos de SAREM se puede mapear a datos tabulados. Sigue una estructura.
- Al utilizar un gestor de bases de datos relacional se cumple la propiedad ACID [59].

La elección de PostgreSQL fue basada principalmente debido a que tenemos experiencia laboral en este sistema de gestión de bases de datos, conocemos sus ventajas y desventajas. Es open source y multiplataforma, lo que nos libra de las desventajas que traía consigo el SQL Server, motivo por el cual decidimos cambiar la implementación del backend a Node.js. Además posee otras ventajas interesantes [60]:

- Fiabilidad y estabilidad:

A diferencia de muchas bases de datos propietarias, es extremadamente común que las compañías reporten que PostgreSQL jamás ha colisionado en los años que ellos llevan trabajando con él, inclusive realizando operaciones de alta actividad.

- Diseñado para entornos de alto volumen de datos:

PostgreSQL utiliza una estrategia de almacenamiento de datos de varias filas denominado MVCC, lo que lo hace extremadamente receptivo en entornos de altos volumenes. El proveedor líder de bases de datos propietarias, Oracle, utiliza está tecnología por las mismas razones.

7.4 Elección de ORM

Una vez realizada la elección de PostgreSQL como manejador de bases de datos, tuvimos que elegir un ORM en JavaScript. Ya que decidimos utilizar el framework LoopBack [26] para generar la API REST de SAREM optamos por elegir el ORM que viene con este framework ya que nos permite trabajar de manera sencilla e integrada. Una de las grandes ventajas que tiene utilizar este ORM es que permite cambiar de manejador de base de datos sin complicaciones, incluso se puede utilizar una base de datos NoSql como MongoDB [61].

7.5 Elección de implementación de envío de notificaciones

Para realizar el envió de notificaciones a los usuarios del sistema SAREM, se utilizó el módulo Nodemailer [62] para el envió de emails. Este módulo permite el envió de emails de manera simple para aplicaciones desarrolladas en Node.js y posee una comunidad grande de desarrolladores lo que nos garantiza tener soporte y es un indicador de la aceptación que el modulo tiene.

Para hacer el envió de Notificaciones Push (Notificaciones que permiten que la aplicación notifique al usuario de nuevos mensajes o eventos incluso cuando el usuario no se encuentra activamente usando la aplicación) para dispositivos Android, se necesita utilizar el servicio GCM (Google Cloud Messaging) [15]. Este servicio es un servicio gratuito que permite enviar mensajes entre los servidores y las aplicaciones cliente. Para utilizar este servicio del lado del servidor utilizamos el módulo node-gcm [19]. Este módulo provee la implementación de GCM para Node.js. Elegimos utilizar este módulo porque es muy popular en la comunidad de desarrolladores (esto nos garantiza soporte y actúa como indicador de aceptación del módulo) y además es bastante simple de utilizar.

Para implementar el envió de Notificaciones Push, se establecieron estrategias de creación de colas con envío diferido, para ello se utilizó RabbitMQ [63] como manejador de mensajes, que entre otros beneficios, permite la creación de mensajes en diferido utilizando TTL especificado en el encabezado del mensaje, enrutamiento de mensajes entre diversas instancias de cola, eliminación de colas dado un TTL por cola, basado en el uso de la misma.

7.6 Elección de plataforma en la nube

En lo que refiere al despliegue de la aplicación SAREM en la nube, decidimos utilizar Amazon Web Services [64]. Amazon Web Services (AWS) es una plataforma de servicios en la nube segura, que ofrece poder de computo, almacenamiento de bases de datos, entrega de contenidos y otras funcionalidades para ayudar a las empresas a escalar y crecer. Elegimos utilizara AWS por las siguientes razones [65]:

- Seguridad:

SAS 70 Type II, PCI DSS Level 1, HIPAA, FISMA Moderate e ISO 27001 son algunas de las certificaciones bien reconocidas que tiene AWS. Estamos hablando de una plataforma de tecnología segura y duradera. Para garantizar la seguridad e integridad de los datos, los centros de datos y servicios de Amazon tienen varias capas de seguridad físicas y operativas.
AWS lleva a cabo frecuentes auditorías para garantizar la seguridad de su infraestructura.
Ha implementado las mejores prácticas en seguridad y también proporciona documentación sobre cómo implementar funciones de seguridad. Asegura la disponibilidad, integridad y confidencialidad de los datos y proporciona privacidad y seguridad de "extremo a extremo".

- Rentabilidad:

Cada cliente paga solo por el poder de cómputo, almacenamiento y otros recursos que utilice. No existen contratos a largo plazo o pagos por adelantado. La flexibilidad de AWS nos permite recortar o ampliar el uso de recursos según sea necesario.

- Flexibilidad:

AWS permite seleccionar el sistema operativo, lenguaje de programación, plataforma de aplicaciones web, bases de datos y otros servicios que se necesiten. Con AWS, se obtiene un entorno virtual que permite cargar el software y los servicios que la aplicación requiera. Esto facilita el proceso de migración de aplicaciones existentes conservando opciones para la construcción de nuevas soluciones.

- Escalable y de alto rendimiento:

El uso de herramientas de AWS, Auto Scaling y Elastic Load Balancing, permite que las aplicaciones puedan escalar hacia arriba o hacia abajo dependiendo de la demanda. Con el respaldo de la infraestructura masiva de Amazon, se tiene acceso a recursos de cómputo y almacenamiento cuando se necesite.

- Experiencia:

Con AWS, se puede aprovechar de una solución escalable, fiable, segura, con una infraestructura a nivel mundial que continua innovando y mejorando. Amazon tiene más de 15 años de experiencia en la entrega de infraestructura global a gran escala.

7.7 Elección de implementación de traducciones a diferentes idiomas

Para implementar las traducciones en inglés y español de SAREM, y permitir en un futuro el soporte de nuevos idiomas de manera sencilla, decidimos crear una API con Node.js [34] capaz de servir JSONs en más de un idioma.

Para almacenar los JSONs decidimos utilizar la base de datos documental MongoDB [61].

Para almacenar los datos en la base consideramos las siguientes posibilidades:

- **Separar los idiomas en colecciones diferentes:**

Crear una colección para cada idioma y consultar la colección apropiada de acuerdo con lo que acepta el cliente. En nuestro caso tendremos las colecciones sarem_es y sarem_en.

El inconveniente obvio de este enfoque es que muchos de los datos están siendo innecesariamente duplicados.

- **Incrustar los datos multilingües en el mismo documento:**

Para cada campo que necesite la traducción en un documento, vamos a insertar los datos multilingües en el interior del campo. Por ejemplo:

```
{

"sarem_appointment": {

    "en": "Appointment",

    "es": "Cita"

  }

}
```

Esto resuelve la duplicación de los campos del primer enfoque, pero añade un poco de complejidad al objeto que estamos guardando.

Preferimos utilizar este segundo enfoque porque consideramos que es mucho más escalable. Imaginemos que tenemos que soportar 25 idiomas diferentes. El mantenimiento de todos los datos duplicados en el primer enfoque se vuelve una pesadilla. En el segundo enfoque agregamos un único nivel de jerarquía para el objeto, lo que no representa mayores complicaciones.

Finalmente, después de decidir cómo guardar los datos en la base MongoDB [61], tuvimos que buscar un enfoque para servir los datos al cliente. Decidimos optar por construir una API que se encargue de enviar los datos tal cual son para el cliente y dejar que este se preocupe por escoger el lenguaje y analizarlo para mostrarlo al usuario. Para esto utilizamos el modulo angular-translate [66] de Angular.js [20].

7.8 Elección de sistema de control de versionado

Alojamos SAREM en la plataforma de desarrollo colaborativo GitHub[67]. GitHub utiliza el sistema de control de versiones Git [68]. Git es similar a otros sistemas de control de versionado como subversión o CVS, pero es distribuido. Lo que significa que si se clona un proyecto Git, se tiene la historia completa del proyecto.

Se puede hacer commit, branch y tag todo en una maquina local sin tener que interactuar con el servidor. Si se estuviera trabajando con subversion o CVS todas las interacciones ocurrían interactuando con el servidor. Esto significa que se puede usar Git sin tener la necesidad de tener conexión a internet, y se gana en performance ya que todas las operaciones (a excepción de push y fetch) son locales y por lo tanto no hay latencia de red implicada.

Elegimos alojar SAREM en GitHub por diversas razones la principal es que utiliza Git que posee una naturaleza distribuida, que lo hace mucho más performante en comparación con otros sistemas de versionado además de que permite trabajar de manera offline. Git también provee una mejor forma de auditar ramificaciones y fusionar eventos. La otra razón por la que tomamos esta es elección es porque hoy en día, casi todos los proyectos de código abierto utilizan GitHub para gestionar su proyecto. GitHub es libre si el proyecto es de código abierto e incluye un wiki y seguimiento de incidencias que hacen que sea fácil incluir más documentación en profundidad y obtener retroalimentación del código hecho.

7.9 Explicación de porqué no se tomó como base el proyecto de la IMM y BPS para la implementación de SAREM

Nos enteramos de la existencia del proyecto SAE[69] a finales de febrero al participar en la defensa del proyecto de grado SAMI donde uno de los miembros del tribunal les pregunto a los integrantes del grupo sobre porque no habían utilizado este proyecto como base y a lo que ellos respondieron que no conocían la existencia del proyecto.

A esa altura la implementación de SAREM estaba bastante avanzada utilizando tecnologías open source como Node.js y Angular.js. Debido a esto decidimos agregarlo al estado del arte del informe pero ya no podíamos tomarlo como base.

Igual nos tomamos el tiempo de investigar sobre el proyecto y tuvimos la idea de comparar SAREM con SAE[69] en la etapa de pruebas, pero al visitar el repositorio donde esta subido el proyecto nos encontramos con falta de documentación sobre como instalar el sistema y hacerlo funcionar lo que nos impidió realizar la comparación.

8. Análisis de Privacidad y Seguridad de Datos

8.1 Privacidad / Seguridad

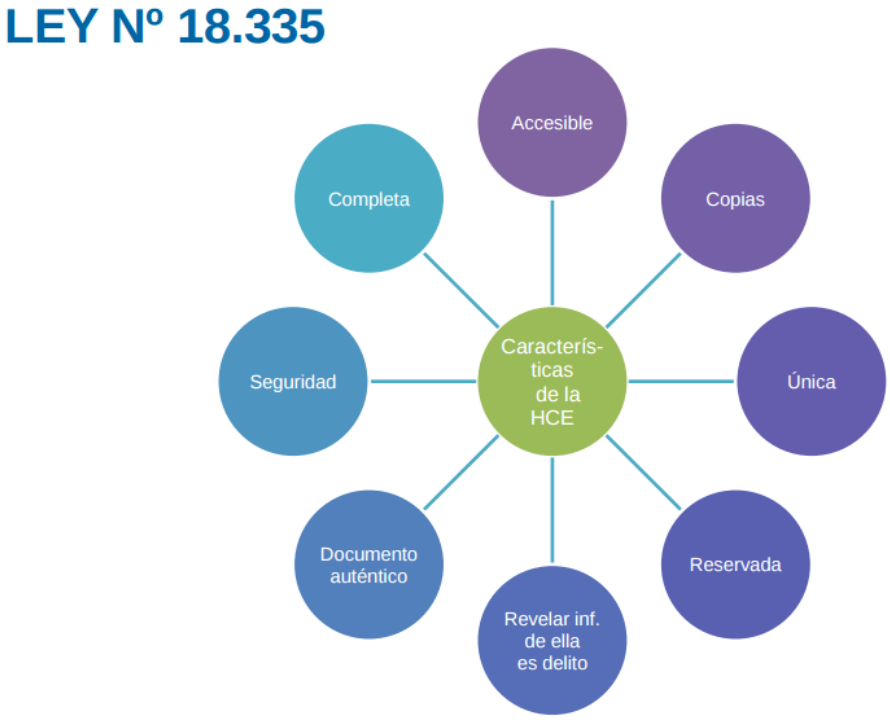
8.1.1 Privacidad de Datos en Uruguay

Es necesario conocer las normativas jurídicas para establecer claramente los límites en cuanto a la publicación de información de los pacientes. Para ello se recurre a la Ley Nº 18.335 [70] que define los derechos y obligaciones de los pacientes y usuarios en los Servicios de Salud.
Primeramente debe destacarse el Capítulo V, Artículo 18 "Todo paciente tiene derecho a conocer todo lo relativo a su enfermedad. Esto comprende el derecho a:", en particular es de suma importancia tener en cuenta el inciso D, el cual menciona lo siguiente:

D) *"Que se lleve una historia clínica completa, escrita o electrónica, donde figure la evolución de su estado de salud desde el nacimiento hasta la muerte.*
La historia clínica constituye un conjunto de documentos, no sujetos a alteración ni destrucción, salvo lo establecido en la normativa vigente.
El paciente tiene derecho a revisar su historia clínica y a obtener una copia de la misma a sus expensas, y en caso de indigencia le será proporcionada al paciente en forma gratuita.
En caso de que una persona cambie de institución o de sistema de cobertura asistencial, la nueva institución o sistema deberá recabar de la o del de origen la historia clínica completa del usuario. El costo de dicha gestión será de cargo de la institución solicitante y la misma deberá contar previamente con autorización expresa del usuario.
La historia clínica es de propiedad del paciente, será reservada y sólo podrán acceder a la misma los responsables de la atención médica y el personal administrativo vinculado con éstos, el paciente o en su caso la familia y el Ministerio de Salud Pública cuando lo considere pertinente.
El revelar su contenido, sin que fuere necesario para el tratamiento o mediare orden judicial o conforme con lo dispuesto por el artículo 19 de la presente ley, hará pasible del delito previsto en el artículo 302 del Código Penal."

Algunos puntos a tener en cuenta con según el Articulo 18, los pacientes de SAREM pueden revisar su historial clínico y obtener una copia de la misma, esto permite a través de mecanismos seguros el intercambios de datos a través de la autenticación y autorización del paciente y el servicio brindado por el centro de Salud correspondiente que implementa SAREM como plataforma de agendado de consultas clínicas.
En lo referente a privacidad, lo critico a tener en cuenta para garantizar dicho punto, es restringir el acceso a datos patronímicos de los pacientes, así como las consultas agendadas y su respectivo parte diario.

Al recabar un mayor volumen de información se establecen se recurre a la imagen provista en una presentación realizada por AGESIC [71], mencionando los conceptos clave de la Ley anteriormente mencionada.



Es importante destacar el Artículo 20 mencionando lo siguiente:
Artículo 20.- Es de responsabilidad de los servicios de salud dotar de seguridad a las historias clínicas electrónicas y determinar las formas y procedimientos de administración y custodia de las claves de acceso y demás técnicas que se usen.
El Poder Ejecutivo deberá determinar criterios uniformes mínimos obligatorios de las historias clínicas para todos los servicios de salud.

Del segundo párrafo no existe mayor descripción posible, no se establecen los criterios mínimos obligatorios para las historias clínicas y no acota específicamente si se refiere a mecanismos de seguridad y privacidad de las mismas o propiedades mínimas que debe brindar el documento.

Ley Nro 18.331

Otra ley jurídica que debe ser tenida en cuenta en el marco legal de SAREM es la ley N 18.331 [72], llamada PROTECCIÓN DE DATOS PERSONALES Y ACCIÓN DE "HABEAS DATA", en la misma se establece el concepto de Dato persona, Dato Sensible, Responsable de la base de datos o del tratamiento, entre otros conceptos de importancia que involucran a SAREM de forma directa o indirecta.
Primeramente se debe establecer si el dominio de la ley aplica a SAREM en su totalidad o parcialmente, para ello se debe tener en cuenta el artículo 3 de la presente Ley.

Artículo 3 - Ámbito objetivo.- *"El régimen de la presente ley será de aplicación a los datos personales registrados en cualquier soporte que los haga susceptibles de tratamiento, y a toda modalidad de uso posterior de estos datos por los ámbitos público o privado. No será de aplicación a las siguientes bases de datos:*
A) A las mantenidas por personas físicas en el ejercicio de actividades exclusivamente personales o domésticas.
B) Las que tengan por objeto la seguridad pública, la defensa, la seguridad del Estado y sus actividades en materia penal, investigación y represión del delito.
C) A las bases de datos creadas y reguladas por leyes especiales."

SAREM no aplica a ninguno de los dominios mencionados, por lo tanto la ley N 18.331 aplica en su totalidad al proyecto.

El Artículo 5 refiere a la actuación de los responsables de la base de datos.

Artículo 5 - Valor y fuerza.- *"La actuación de los responsables de las bases de datos, tanto públicos como privados, y, en general, de todos quienes actúen en relación a datos personales de terceros, deberá ajustarse a los siguientes principios generales:*
A) Legalidad.
B) Veracidad.
C) Finalidad.
D) Previo consentimiento informado.
E) Seguridad de los datos.
F) Reserva.
G) Responsabilidad.

Dichos principios generales servirán también de criterio interpretativo para resolver las cuestiones que puedan suscitarse en la aplicación de las disposiciones pertinentes."

Los puntos A hasta G incluso se encuentran detallados en los artículos 6 hasta 12 incluso.

Los principios de Legalidad, Veracidad, Finalidad se aplican sin problema legal alguno en el desarrollo del proyecto, el sistema posee como finalidad primaria facilitar el acceso al agendado de citas médicas de forma segura y confidencial. Por lo cual la veracidad de los datos queda relegada exclusivamente al dictamen del profesional cuando refiere a un diagnóstico primario dado una cita médica realizada.
Inclusive en el Artículo 8 se define el principio de la finalidad, refiriendo a datos que deben ser eliminados "cuando hayan dejado de ser necesarios o pertinentes a los fines para los cuales hubieren sido recolectados", en caso particular del proyecto, el almacenamiento de datos relativos a historial de consultas debe considerarse como la posibilidad de realizar análisis estadísticos para optimizar los recursos humanos o inmobiliarios dedicado al agendado de citas, en este caso se trata la problemática en la sección trabajos futuros.

En cuanto al consentimiento informado, se trata de una falencia del sistema, se debe crear un formulario explicitando la normativa relativa al almacenamiento de datos con el consentimiento explícito del usuario.

Correspondiente a la Seguridad de Datos, la ley especifica en el artículo citado a continuación, las características que debe presentar el sistema.

Artículo 10 - Principio de seguridad de los datos.- *"El responsable o usuario de la base de datos debe adoptar las medidas que resultaren necesarias para garantizar la seguridad y confidencialidad de los datos personales. Dichas medidas tendrán por objeto evitar su adulteración, pérdida, consulta o tratamiento no autorizado, así como detectar desviaciones de información, intencionales o no, ya sea que los riesgos provengan de la acción humana o del medio técnico utilizado.*
Los datos deberán ser almacenados de modo que permitan el ejercicio del derecho de acceso de su titular. Queda prohibido registrar datos personales en bases de datos que no reúnan condiciones técnicas de integridad y seguridad"

Por ende es necesario para el despliegue de la aplicación que la base de datos se encuentre en dominio exclusivo del organismo que utiliza SAREM, es responsabilidad exclusiva del centro de Salud brindar una infraestructura acorde a las mencionadas en el presente documento o que se adecuen fielmente a las normativas legales actuales.
Se debe tener en cuenta en el segundo párrafo a que refiere exactamente con "condiciones técnicas de integridad y seguridad". Al no existir un detalle técnico exhaustivo sobre tal punto, se garantiza que el manejador de base de datos utilizado, cumple con los criterios de integridad [73] y seguridad[74] en sus distintos niveles.

En cuanto al artículo 11, es responsabilidad en su totalidad del centro de salud, en lo que refiere al acceso de información a las personas responsables del mantenimiento de la base de datos. Esto implica un contrato profesional con obligaciones de mantener en secreto los datos del cual se encarga de administrar.

8.1.2 Criterios de Seguridad

Los criterios de seguridad y privacidad que deben ser considerados en el transcurso del proyecto, involucra una serie de consideraciones previas. Para ello se recurre a una serie de hipótesis relativas a la temática "Seguridad en Centros de Salud" [75]:

- Integridad, la información transmitida, recibida, procesada y persistida deben reflejar una representación fidedigna de la realidad. Redunda en la idea de que debe ser manipulada de forma correcta.
- Confidencialidad debe garantizarse así como la información de salud de los pacientes en el sistema. Asimismo debe cumplir que el acceso a la información debe ser limitado en su totalidad a los organismos autorizados.

- Autenticación, se debe brindar mecanismos de autenticación no solamente a los usuarios del sistema, sino a las entidades que consumen información del mismo, por ejemplo aplicaciones que consumen los datos de un repositorio central.
- No repudio, este punto involucra la no negación de la información, para ello es necesario establecer el origen de la fuente, a su vez el destinatario debe asegurar que el envío fue efectuado.
- Control de acceso, a nivel de infraestructura únicamente debe proveerse acceso a personas autorizadas a la infraestructura del sistema. Este punto suele ser importante puesto que los servicios de datos deben tener acceso restringido, tanto a nivel de dominios como accesos de usuarios.
- Anonimidad, en caso que el servidor se encuentre desplegado en "la nube" es necesario garantizar que el sistema no brinde mecanismos de aprendizaje sobre la identidad del usuario. Se debe garantizar el anonimato del paciente en sentido estricto.
- Imposibilidad de vinculación a través de autorización por los datos obtenidos a partir de distintas fuentes. Este punto refiere que no se puede acceder a datos privados a partir de la obtención de otros.
- Auditar, toda la información debe brindarse de forma segura, así como también se debe establecer mecanismos de monitorización sobre las mismas. Por ejemplo a nivel de bases de datos se debe tener en cuenta modificaciones pertinentes, autorías en las tablas.

De las hipótesis recientemente mencionadas, se desglosa cuáles son las fortalezas y debilidades del proyecto implementado. En el caso de las falencias en las hipótesis de seguridad y privacidad, se plantea el marco teórico y práctico necesario para llevarlo a cabo. En el caso de lo implementado, se realiza una descripción de los mecanismos utilizados que garantizan la propiedad.

Integridad

Los datos son persistidos utilizando manejador de bases de datos relacional. La integración de distintas operaciones que pueden resultar ante una falla, logrando operaciones incompletas con datos parcialmente incompletos. Para ello las operaciones deben ser manipuladas de forma atómica a nivel transaccional en la base de datos [76].
Como el sistema presenta una arquitectura de almacenamiento centralizada, no existen problemas de integridad con la utilización de transacciones, permitiendo la atomicidad de un conjunto de operaciones. Para ello se configura en el manejador PostgreSQL[58] un nivel de aislamiento transaccional "READ-COMMITTED", de esta forma se garantiza que no existirán lecturas oscuras, conocidas como "DIRTY-READ", cuando el nivel de concurrencia aumenta en el sistema.

Confidencialidad

El sistema brinda acceso confidencial de la información estableciendo una política estricta de roles sobre los usuarios del sistema. Con la jerarquía establecida se logra manipular la información de forma segura y destinada a los actores del flujo que son dueños o propietarios de la información. De esta forma se asegura que a través de peticiones a la API, un atacante no logra obtener información confidencial de cada paciente o propiedades del sistema en sí mismo.
Se solventa el problema logrando una capa de seguridad que permite la Autorización y Autenticación de usuarios al sistema ya que con el Framework seleccionado se puede establecer políticas de acceso autoritativas.
Esto no es suficiente, los puntos de acceso hacia la API Rest deben ser monitorizados, se puede establecer en este nivel una nueva capa de seguridad permitiendo un control centralizado, administrando las credenciales de acceso para cada aplicación involucrada en el proceso, utilizando distintos mecanismos de autorización y autenticación sin lograr una dependencia exclusiva a nivel de aplicación. En trabajos a futuros se describe una alternativa utilizando un servicio para tal finalidad, Keycloak [77] es un servicio orientado al manejo de credenciales, podría ser agregado como una nueva capa al sistema para lograr una independencia importante a lo que refiere la problemática de Autenticacion y Autorizacion a nivel de microservicios.
Con el Framework seleccionado cuidadosamente, se establece para cada rol del sistema un conjunto de permisos, de esta forma ante solicitudes con distintos verbos HTTP (POST, GET, PUT, HEAD, DELETE) y la validación de la autorización del usuario con un Token de acceso que define los permisos del mismo, es posible lograr un nivel de granularidad en cuanto a la restricción de los puntos de acceso sensibles.
Lo anteriormente mencionado refiere a la confidencialidad ante atacantes externos al sistema, pero también debe considerarse mecanismos que garanticen la confidencialidad de los datos a aquellas entidades que poseen acceso a los distintos servidores que pueden contener los Servicios de Datos.
Para este punto es necesario que a nivel de sistema operativo se asignen los permisos adecuados a los responsables de la gestión de los recursos o aplicación. Para ello se debe establecer permisos a nivel de servicios de datos, configurando el manejador de bases de datos con autenticación por usuario (típicamente delegando un usuario por servicio o administrador).
Algunas medidas que deben tomarse en cuenta en cuanto a la implantación de los servicios de datos.

- Base de datos con acceso local únicamente, si es posible incluso restringir la base ante cualquier tipo de conexión TCP/IP y utilizar a cambio Unix/Socket, con este cambio se logra tener un acceso totalmente restringido a la base de datos y una mejora de rendimiento sustancial (33%) de la aplicación según comparte Bruce Momjiam [78] cofundador de PostgreSQL [58].
- Políticas de acceso con roles diferenciados, para ello como Administrador de Bases de Datos se debe crear los usuarios a nivel de base, una vez definidos los mismos se debe autorizar configurando correctamente pg_hba.conf[79] de PostgreSQL correctamente el acceso de cada usuario a su respectiva base. El nivel de granularidad es importante en este punto, se debe crear un usuario para cada

servicio, y si es necesario crear permisos de lectura y escritura sobre las tablas en cuestión.
- No compartir en distintos dominios el acceso a la base de datos, esto se logra en conjunto con el primer punto, no realizando binding del proceso postmaster de PostgreSQL.

Con los puntos mencionados se garantiza que incluso con el acceso al servidor principal de los servicios de datos, no será posible realizar consultas sobre los datos confidenciales de los pacientes.
Ante el caso de secuestro de información, esto es la apropiación indebida del recurso informático, se debe tener en cuenta encriptación del sistema de archivos, garantizando esta forma la ilegibilidad de los datos ante atacantes externos con acceso directo al sistema.

Autenticación y Autorización

Este punto tiene dependencia con el anterior, el acceso a datos confidenciales debe ser limitado, para ello se deben establecer de antemano la serie de dominios y las políticas de seguridad de cada uno de ellos.
Los dominios identificados presentan una serie de restricciones, por lo cual debió realizarse un análisis primario de la situación y ajustar las políticas de acceso a cada uno de ellos.
En el diagrama de la **Figura AA1** se identifican claramente los dominios en cuestión, identificados por dominio público y dominio interno (intranet).

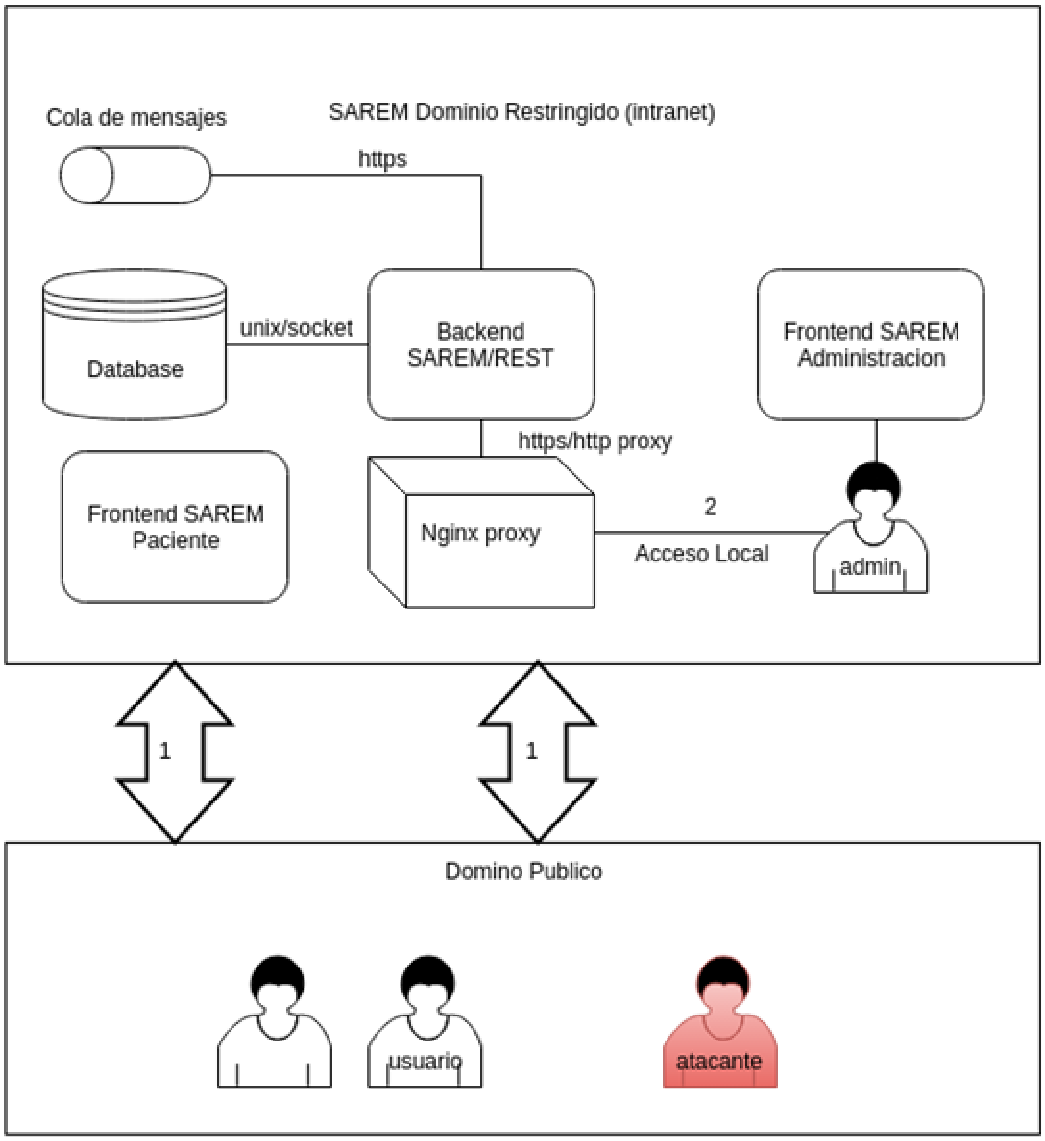


Figura AA1 – Dominios Restringidos

En el diagrama de la **Figura AA1** se establece la interacción según roles de los usuarios, únicamente se logra acceso a SAREM desde el dominio público a los usuarios que poseen el rol de paciente, en otras palabras, se restringe cualquier acción privilegiada (administración de recursos internos como lo son los locales, consultas, médicos, personas, etc).

Para cumplir con el objetivo primario se utiliza un servicio proxy (Nginx), básicamente consiste en la configuración de dos perfiles de acceso con interfaces claramente diferenciadas. Para la interfaz pública (cuyos puntos de acceso se reflejan con el número uno en el diagrama) se limita únicamente a los puntos de acceso de lectura y

escritura necesarios por parte del paciente, delimitando de esta forma el acceso a instrucciones privilegiadas de administración.

En la **Tabla AA1** se listan las rutas habilitadas en el proxy con sus respectivos verbos para el dominio público.

Tabla AA1 – Rutas Habilitadas en el Proxy (Nginx)

Verbo	Ruta
POST	/accounts/login
POST	/accounts/reset
POST	/accounts/logout
POST	/accounts/password
GET	/accounts/roles
POST	/personas/:id/espera
POST	/personas/:id/consultas
POST	/personas/:id/canceladas
GET	/personas/:id/consultas/*
GET	/personas/:id/history
GET/POST/DELETE	/personas/:id/referencia
GET	/personas/medicos
GET	/consultas/search
GET	/notificaciones
GET	/locales
GET	/especialidades
GET	/translations

Las implicancias de las políticas de acceso mencionadas, se ven reflejadas en el archivo nginx.public.config adjunto en la sección **14.7 Nginx Security Configuration** del anexo.

En cuanto a las políticas para el dominio intranet, los usuarios con roles de administrador no presentan restricciones sobre las rutas debido a que el rol administrador en sí mismo permite realizar lectura sobre todas las rutas del servicio.

En cuanto al protocolo de seguridad en los llamados HTTP, se ofrece HTTPS, creando un canal encriptado SSL/TLS para el intercambio punto a punto de forma segura. Este punto se aplica en el despliegue del cliente Web así como en la API REST, ambos contenidos deben estar cifrados, el primero particularmente en el formulario de inicio sesión, en cambio en el segundo, todos los llamados deben ser encriptados ya que potencialmente brindan datos sensibles del usuario.

Auditoría

Debe garantizarse en un sistema de salud que ante cualquier cambio en los registros los mismos sean registrados. Esto quiere decir que ante operaciones DML sobre la base de datos, el tiempo de acceso, el usuario, y la dirección de donde efectivamente se realiza el cambio debe ser registrado.
Para el cumplimiento de este punto, se implementaron triggers [80] a nivel de base de datos para las tablas consideradas más importantes en el sistema. Tanto las operaciones sobre las consultas, como agendado de las mismas, datos pertinentes a los pacientes del sistema, referencias entre pacientes y médicos, se agregan triggers de auditoría, se utiliza una tabla en común para auditoría de las tablas. De esta forma se logra un control estricto sobre las operaciones aplicadas ante posibles manipulaciones en los datos, ampliando el espectro de seguridad incluso ante accesos de forma directa a la base.

La tabla de auditoría posee la siguiente estructura las siguientes columnas:

- Schema_name: Refiere al esquema en el cual se ejecuta la operación DML.
- Table_name: La tabla en la cual se ejecuta el trigger.
- User_name: El nombre de usuario que realiza la operación.
- Action_tstamp: El timestamp que se realiza la operación.
- Action: Inserción (I), Actualización (U), Eliminación (D).
- Original_data: Valores del registro antiguo, también conocido por el valor de la variable OLD.
- New_data: Nuevos valores que adopta el registro (de utilidad cuando se trata de una operación update).
- Query: Puede ser útil conocer la sentencia realizada, si la eliminación se realiza masivamente (delete from table where 1=1), se podrá detectar casos mal intencionados de forma directa.

Recuperación de Datos

Se debe tener en cuenta mecanismos de Recuperación ante Desastres, derivadas de eventuales errores del personal de administración que implique la pérdida total o parcial de la información.
Primeramente se debe tener en cuenta el intervalo en el cual debe efectuarse los respaldos, de la base de datos y en qué condiciones deben ser almacenadas, los motivos que desencadenan las condiciones, refiere estrictamente a mecanismos de encriptación sobre los respaldos. Ya que los mismos pueden ser restaurados en bases de datos de terceros si se posee acceso de forma ilícita al sistema. Para garantizar que los respaldos no implique infringir algunos de los puntos mencionados en las Leyes 18.331 [72] y 18.335 [70].
El organismo encargado de la base de datos, es el responsable de definir el intervalo de tiempo en el cual el salvaguardo será efectuado, se sugiere independientemente del caso que las políticas de retención de los mismos oscile entre 3 o más meses con la finalidad mantener distintos “snapshots” a lo largo del tiempo y detectar ante posibles anomalías, el origen del problema.

Para efectuar el respaldo plano se puede utilizar la herramienta pg_dump [81], determinando la base de datos, el esquema, así como el usuario responsable con su debida clave de acceso.
A continuación se describe un ejemplo de cómo se procedería al respaldo de una base de datos de forma local.

```
pg_dump -d $DATABASE_NAME -U $DATABASE_ADMIN -W > database.sql
```

Se asume en este caso el esquema, host y puerto heredados de la configuración habitual de PostgreSQL, public, localhost, y 5432 respectivamente.

Si bien este paso garantiza obtener una copia (snapshot) actual de la base de datos, no se respaldan los archivos transaccionales conocidos en la jerga de postgreSQL como archive
_logs, lo cual permite secuencialmente aplicando los logs, reconstruir la base de datos de un estado a cualquier instante delta que sea deseado. Entonces de esa forma se logra reconstruir la base de datos desde un snapshot a cualquier intervalo de tiempo necesario, por ejemplo el segundo previo a una sentencia “truncate table consultaagenda cascade”, lo cual implicaría la pérdida de todas las consultas agendadas en el sistema.

Además, la realización de respaldos de forma plana no garantiza de forma alguna el principio de privacidad, ante el acceso no autorizado de terceros se puede descifrar sin problema alguno y replicar los datos de forma inmediata. Por ello es recomendable que el responsable de efectuar los mecanismos de recuperación, encripte los mismos con un sistema de encriptación.

Para este caso se recomienda el uso de OpenSSL [82] provisto en distribuciones GNU/Linux, tomando como entrada el respaldo previamente generado, se procede a la encriptación del archivo con una clave previamente solicitada por la herramienta.

```
cat database.sql | openssl enc -aes-256-cbc > database_enc.dat
```

Con el paso mencionado se crea un archivo llamado daabase_enc.dat, cifrando los datos utilizando un sistema de cifrado simétrico AES [83] inicializando el algoritmo con un vector inicial.
Para el proceso de desencriptación se debe conocer la clave provisionada (vector) en el paso anterior.

```
openssl enc -aes-256-cbc -d -in database_enc.dat
```

De esta forma se logra un sistema de respaldos seguro, utilizando un sistema de cifrado simétrico AES-256. También puede utilizarse otros sistemas de encriptación, en este punto se utiliza AES [83] a modo de demostrar que es posible realizar respaldos de forma segura.
Si se observan ambas líneas expuestas anteriormente, se puede combinar la salida del respaldo de forma tal de no obtener resultados intermedios legibles. Para ello se puede redirigir la salida de pg_dump [81] directamente a OpenSSL [82] para la encriptación adecuada.

```
pg_dump -d $DATABASE_NAME -U $DATABASE_ADMIN -W | openssl enc -aes-256-cbc
> database_enc.dat
```

8.1.3 Conclusiones

Se han expuesto mecanismos de seguridad en SAREM en los componentes cruciales del sistema, además se provee estrategias relativas al almacenamiento de datos, la gestión de los mismos, lo cual implica un uso correcto de los usuarios que mantienen acceso a la base de datos. Proveer estrategias de respaldo y encriptación de datos, así como el uso de las herramientas para la finalidad, reflejan el conocimiento conciso sobre las herramientas utilizadas en el desarrollo de SAREM, tanto el manejador de bases de datos PostgreSQL, como el proxy (Nginx) para restringir los llamados desde el dominio público, utilizar estrategias de caché para optimizar el rendimiento del sistema, así como distribuir la carga utilizando distintos recursos computacionales.
La normativa legal en Uruguay sostiene una serie de condiciones a ser cumplidas en cuanto a la retención de la información, sin embargo no establece en ningún inciso si la solución debe ser implementada con productos específicos. Dada la escasez de información en ciertos puntos (seguridad ante todo), se considera que los mecanismos de seguridad provistos en el sistema, proveen las características primarias que debe brindar un sistema de salud.
Cabe mencionar, que si bien en un marco teórico es posible realizar todo lo mencionado, pueden existir errores propios de la aplicación, por lo cual un paso previo

a la puesta en producción, se deben realizar pruebas exclusivamente dedicadas al área de seguridad, potencialmente en coordinación con organismos dedicados a dicha problemática.
Otro punto a destacar es el planteo de una arquitectura local, para el despliegue de la aplicación sin dependencia exclusiva de servicios de datos externos (a excepción de Google Cloud Messaging para el envío de Notificaciones Push para los dispositivos móviles).
Con el planteo mencionado, es posible establecer todos los criterios de seguridad investigados, si bien existen falencias ya mencionadas relativas al consentimiento por parte del usuario, el problema es potencialmente menor y puede ser solventado de forma simple.
La separación en distintos dominios, permite un análisis exhaustivo de la fuente de datos expuesta a través de la API REST, así como también la autenticación y autorización de roles administrativos únicamente pueden ser realizados internamente al dominio intranet, ofrece una capa extra de seguridad, ya que las manipulaciones críticas de los datos deben ser realizadas exclusivamente por agentes que pertenecen al dominio.

8.2 Arquitectura

El enfoque arquitectónico presentado, realiza énfasis en la utilización de recursos en "la nube". Si bien se trata de un enfoque simple, fácil despliegue, delegando la responsabilidad de operaciones a los servicios de un proveedor Cloud, urge la necesidad de contar con una infraestructura propia, de esta forma se logra mitigar problemáticas de alojamiento de datos privados en infraestructura físicamente no accesible.
Para ello se adopta una postura arquitectónica similar a la planteada en cuanto a componentes, con la modificación de los servicios que se encarga del despliegue y seguridad del sistema en su totalidad.

Por lo tanto, de los componentes mencionados anteriormente utilizaremos un estrategia de despliegue para encapsular los servicios en contenedores, ofreciendo un nivel de acceso a los datos sensibles personalizada.
De los componentes anteriormente mencionados, se debe ofrecer estrategias de seguridad para la persistencia de los datos. Con la estrategia de contenedores, es posible lograr aislamiento entre procesos, disponer de sistema de archivos encriptados a nivel de sistema operativo, de forma tal de lograr una jerarquía de seguridad personalizable.

Para ello se recurre a la estrategia de virtualización con contenedores, ofrece entre otros beneficios un bajo costo de instanciación en comparación con la estrategia de Hipervisores tradicional. Para este caso se optó por la utilización de Docker [84] como plataforma proveedora de distintas instancias en contenedores.

Docker [84] presenta una serie de características que benefician notoriamente el despliegue de SAREM en cualquier institución que utilicen servidores GNU-Linux en su infraestructura.

- Brinda aislamiento de los servicio, esto quiere decir que cada contenedor provee sus variables de entorno de forma aislada de los demás servicios que se desea instanciar.
- Una imagen docker [84] presenta un conjunto de capas, con un conjunto minimal de bibliotecas para el funcionamiento de la aplicación.
- Del concepto anterior, si de alguna forma es necesario una biblioteca extra en la imagen para extender una aplicación existente como imagen, únicamente se especifica la nueva capa a construir, esto puede implicar únicamente realizar mención por ejemplo en un sistema GNU-Debian “apt-get install nginx”, se crea una nueva capa con el servicio Nginx [85].
- Reutilización de capas, es posible generar un sistema de versionado similar a git [68], logrando de esta forma caching de las imágenes. De esta forma, cuando se obtiene una nueva versión de un servicio que incluye una nueva biblioteca, únicamente se descargara la nueva capa. En comparación con una imagen de máquina virtual tradicional, se debe obtener la imagen en su totalidad.
- Uso de dos piezas fundamentales del Kernel Linux, namespaces y cgroups. Para el primero se puede establecer un manejo de las interfaces de red por proceso, el concepto en sí mismo de namespaces. Por lo cual se puede manipular de esta forma un espacio virtual a nivel de red para cada proceso, generando de esta forma aislamiento incluso a nivel de topología de la red.
  Para el segundo caso, se utiliza cgroups para la administración de recursos de hardware, por lo cual para cada proceso se puede destinar una porción acorde a las necesidades de los servicios, por ejemplo establecer el consumo máximo de RAM, así como requerimiento computacionales.
- El uso de namespaces brinda un nivel de configuración superior a los hipervisores tradicionales, por ejemplo replicar un servicio en el cual el proceso es “bindeado” en el puerto 80, se puede lograr N instancias de dichos servicios con la misma configuración. Y manipular mediante “EXPOSE” las redirecciones hacia el host principal, por ejemplo: instancia 1 se mapea al puerto 8080, instancia 2 es mapeado al puerto 8081, así sucesivamente y en las interfaces requeridas.
- Arquitectura distribuida, si bien docker [84] es instanciado en un host, es posible generar un modo cluster de Docker utilizando Swarm [86], ofreciendo contenedores de forma distribuida en diversos servidores, con una capa de abstracción en modo cluster.
- Del punto anterior se pueden establecer servidores dedicados a servicios de datos, en un host ofrecer un cluster de bases de datos relacionales o NOSQL por ejemplo y en otro servidor un conjunto de servicios (contenedores) NodeJS [34].

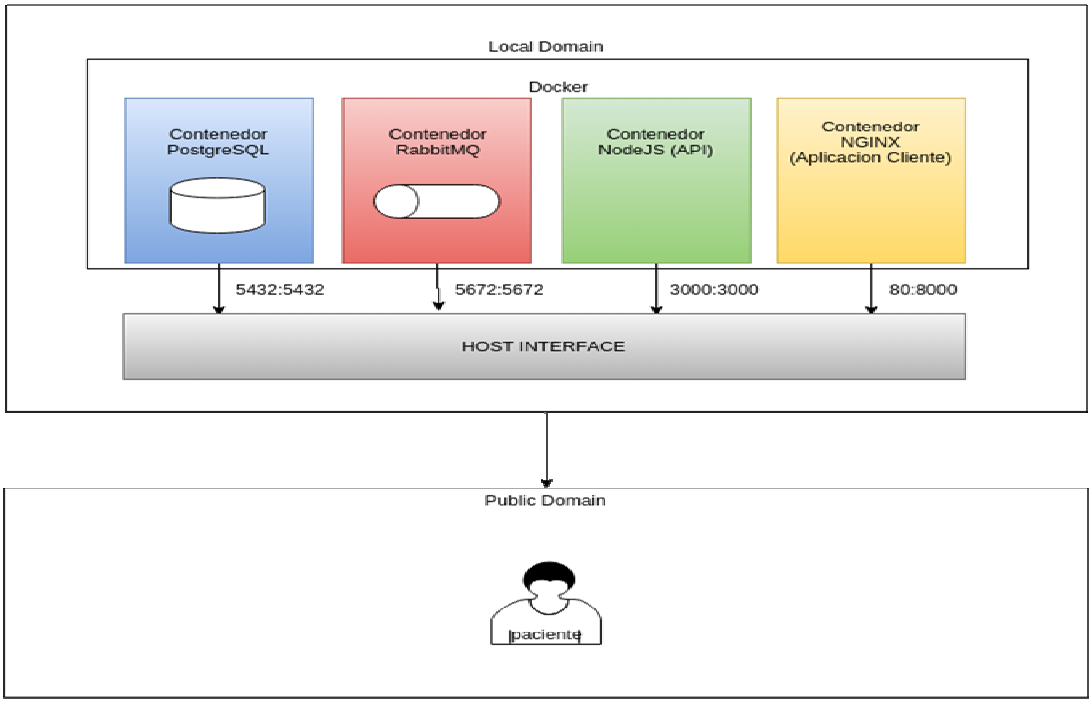


Figura AA2 – Docker orquestador principal de servicios brindados por SAREM

Se brinda como solución alternativa al despliegue Cloud utilizando AWS [64]. En el cual Docker [87] ofrece como orquestador principal de los servicios brindados en SAREM.
Los componentes se desglosan en Servicio de datos y Componentes de Aplicación:

- Servicios de Datos, típicamente la base de datos relacional PostgreSQL[58] y RabbitMQ [63] garantizando de esta forma la exclusividad de los datos al dominio local.
- Componentes de Aplicación, en su totalidad se desglosan en la aplicación cliente Angular [20] del cual se encarga el contenedor Nginx [85] de servir el contenido estático y la aplicación Backend, en un contenedor NodeJS [34] con despliegue automático utilizando Git [68] como referencia para la descarga del código y posterior instanciación del servicio.

Gracias al encapsulamiento de los servicios, se logra exponer todos los servicios a nivel local, con esto se evita errores de exposición de servicios críticos como lo son la base de datos o la cola de mensajes.
La ventaja arquitectónica ofrece escalabilidad horizontal en el despliegue de servicios. También un control estricto de los servicios utilizando cgroups y namespaces correctamente configurados. A tal punto se pueden instanciar servicios de forma totalmente arbitraria, reutilizando componentes como por ejemplo la base de datos. Por ejemplo, a nivel de manejador, se puede instanciar un único servicio PostgreSQL [58], y utilizar un particionamiento por base de datos o por esquema. Esto implicaría la instanciación de la API REST con distintas variables de entorno, lo cual es simple de realizar utilizando una estrategia de contenedores.

Con el correcta configuración de subdominios, se puede lograr instanciar N servicios SAREM para distintos proveedores de salud de la rama pública, utilizando la misma infraestructura con niveles de configuración sencillas.

Imágenes Docker

Se utiliza una estrategia de reutilización de servicios previamente configurados. Para ello se recurre al repositorio DockerHub [88] el cual aloja distintas imágenes de los servicios anteriormente mencionados.

Tabla AA2 – Imágenes Docker

Imagen	Tamaño	Capas	Enlace
Nginx	183 MB	8	https://hub.docker.com/_/nginx/
NodeJS	647 MB	10	https://hub.docker.com/_/node/
PostgreSQL	266 MB	22	https://hub.docker.com/_/postgres/
RabbitMQ	301 MB	24	https://hub.docker.com/_/rabbitmq/

Como atributos notables de la arquitectura Docker [87], es posible reutilizar las imágenes provistas de manera oficial, esto implica la creación de una nueva capa y el correcto uso de Dockerfile[89], por ejemplo para la correcta configuración de Nginx[85] se debió modificar correctamente el archivo /etc/nginx/sites-available/default para los mecanismos de seguridad detallados en la sección Seguridad y Privacidad. La única forma de realizar este cambio es crear una imagen docker propia con el agregado de los archivos pertinentes.
El mismo criterio debe ser tenido en cuenta cuando se instancia la base de datos, agregando nuevas capas para el aprovisionamiento del esquema principal del sistema.
Este tipo de arquitectura demuestra el poder de modularización en servicios, la extensibilidad de los mismos, agregando propiedades o restricciones para cada sistema en particular a nivel de instancia.
Otro beneficio que se desprende inmediatamente es la capacidad de despliegue de forma masiva, sin la necesidad de instalar servicios en el sistema principal, manteniendo el sistema de archivos lo más limpio posible y niveles de configuración dedicadas por contenedores. Esta implicancia restringe el despliegue de la aplicación en dos niveles, el Sistema operativo GNU-Linux y la herramienta Docker [87], notar que todos los servicios descritos son instanciados virtualmente utilizando namespaces y cgroups del Kernel como estrategia fundamental.

9. Gestión

9.1 Descripción General

SAREM surgió del trabajo en conjunto de estudiantes avanzados de la carrera RRMM e Ingeniería en Computación. Una descripción más detallada de cómo fue la gestión del proyecto se puede encontrar en el anexo sección **13.6 Bitácora del Proyecto**.

Lo importante que hay que destacar es que el proyecto inicio con tres integrantes de Ingeniería en Computación. En setiembre de 2015 uno de los integrantes decidió abandonar el grupo, por lo que hubo que realizar un esfuerzo mayor para poder finalizar el proyecto y en consecuencia la dedicación total sobrepaso la carga horaria estipulada para un proyecto de estas características. La pérdida de un integrante así como el feedback obtenido al presentar el proyecto en Ingeniería deMuestra 2015, provocaron cambios en el alcance del proyecto lo que género que su culminación se extendiera.

9.2 Distribución de Horas y Esfuerzo

Se muestra en la Figura G1 una gráfica que corresponde a las horas dedicas al proyecto.

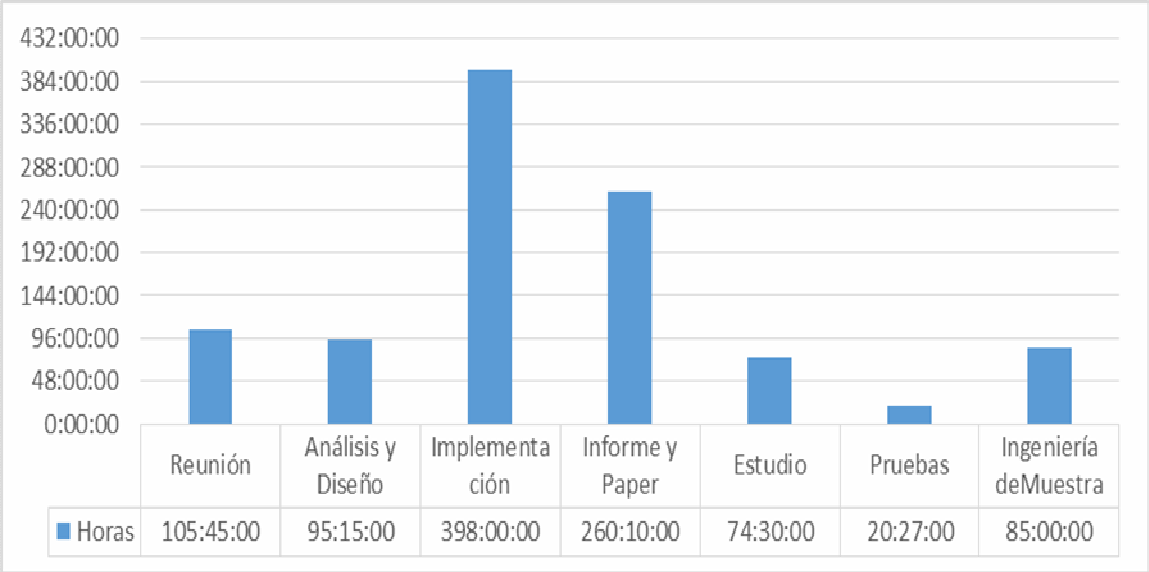


Figura G1 – Distribución de horas.

La Implementación incluye las horas dedicadas a la primera tecnología utilizada que luego se decidió abandonar. La implementación de SAREM con tecnologías open source insumió unas 245 horas.

Se muestra en la Figura G2 una gráfica comparativa que muestra las horas dedicadas al proyecto implementado con tecnologías Microsoft versus las horas dedicadas al proyecto con tecnologías open source (Node.js + Angular.js).

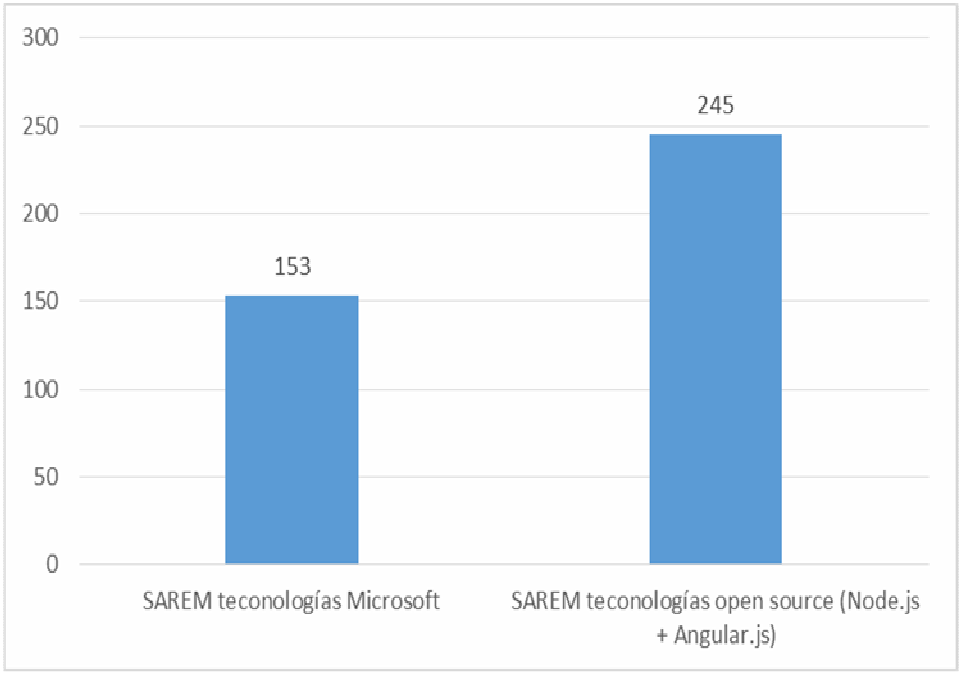


Figura G2 – Doble implementación de SAREM.

Comparación de la cantidad de horas-persona insumidas dadas las diferentes tecnologías utilizadas.

Como se puede ver el proyecto con tecnologías open source consumió más horas de implementación. Esto es debido también a que la implementación de SAREM con tecnologías Microsoft comenzó a finales de julio de 2015 y finalizo a finales de noviembre de 2015. Ese periodo de tiempo coincidió con el semestre electivo de la facultad donde se cursaron materias por partes de los integrantes del equipo que quitaron tiempo para el proyecto, además de que en la nueva implementación de SAREM se implementó la aplicación móvil y se tomaron en cuenta las mejoras plantadas por el tutor del proyecto Franco Simini. Estos hechos impactaron en el aumento de horas del segundo proyecto.

En la Figuras G3 y G4 se muestra el esfuerzo en horas por mes, y la cantidad de "commits", o subidas de código al repositorio, lo cual puede utilizarse como medida de esfuerzo de implementación durante cada mes para el proyecto SAREM realizado con tecnologías de Microsoft.

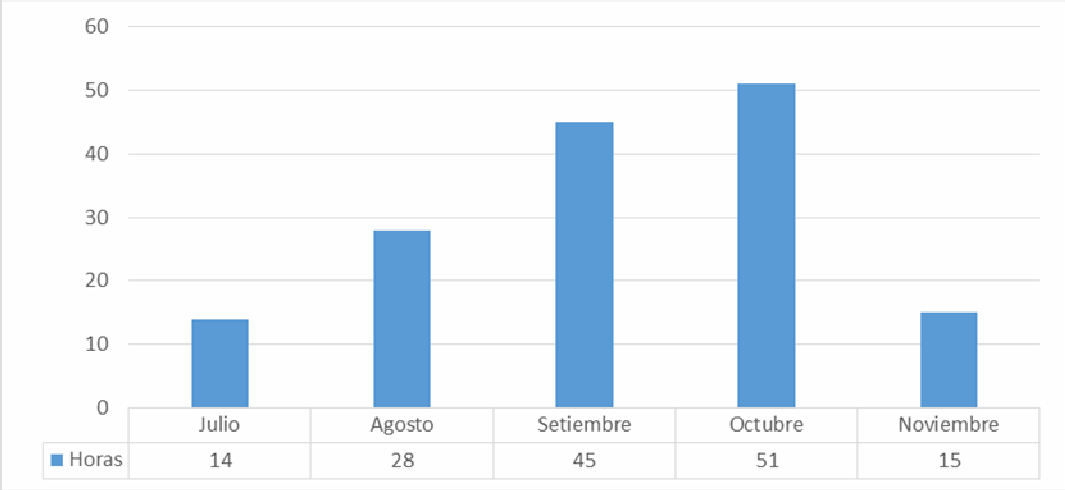


Figura G3 – Horas dedicadas al proyecto SAREM (tecnologías Microsoft) por mes en fase de implementación.

Se observa en la gráfica de la Figura 4 que setiembre y octubre son los meses que han consumido más horas. Esto es debido al esfuerzo que se realizó para llegar a Ingeniería de Muestra 2015, evento que se llevó a cabo a finales de octubre de ese año. La baja de horas de implementación en noviembre está relacionada con el hecho de que decidimos migrar el proyecto para utilizar tecnologías open source.

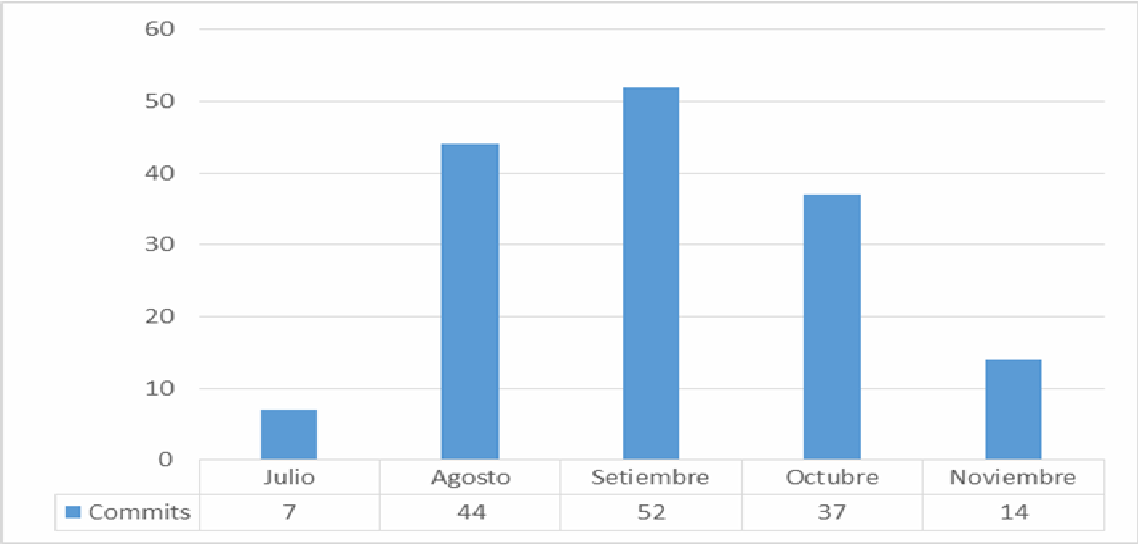


Figura G4 – Commits por mes proyecto SAREM (tecnologías Microsoft) en fase de implementación.

En la Figuras G5 y G6 se muestra el esfuerzo en horas por mes, y la cantidad de "commits", o subidas de código al repositorio, lo cual puede utilizarse como medida de esfuerzo de implementación durante cada mes para el proyecto SAREM realizado con tecnologías open source.

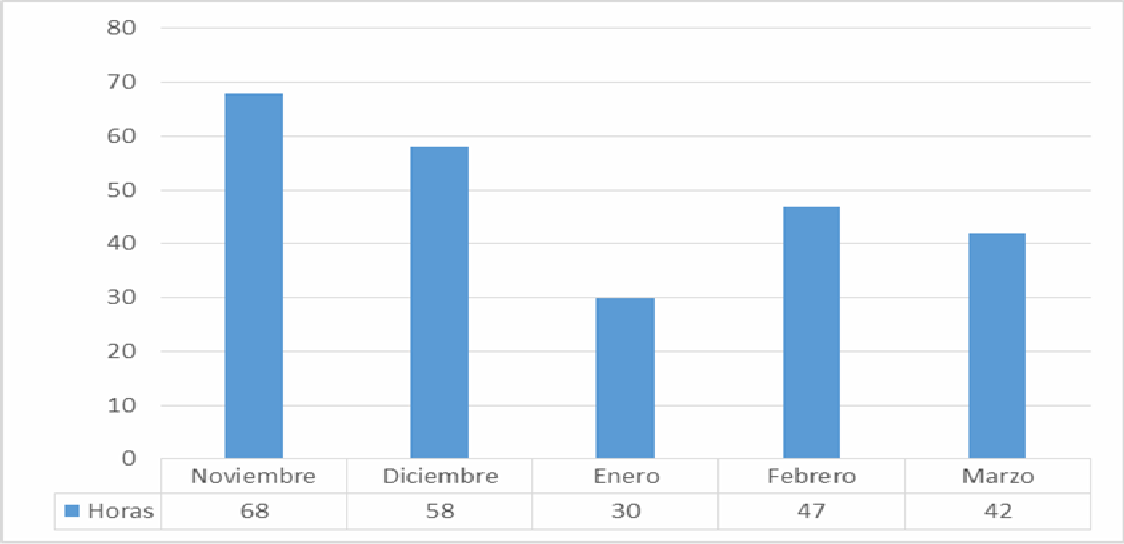


Figura G5 – Horas dedicadas al proyecto SAREM (tecnologías open source) por mes en fase de implementación.

Se puede ver en la gráfica de la figura 6 que los meses que han llevado más horas de implementación son noviembre y diciembre, y que en enero el registro de horas ha bajado. La baja de horas en enero se puede justificar por días de descanso que nos hemos tomado en las vacaciones. En febrero el número de horas vuelve a aumentar, bajando nuevamente en marzo debido a que la fase de implementación culminó el 12 de marzo de 2016.

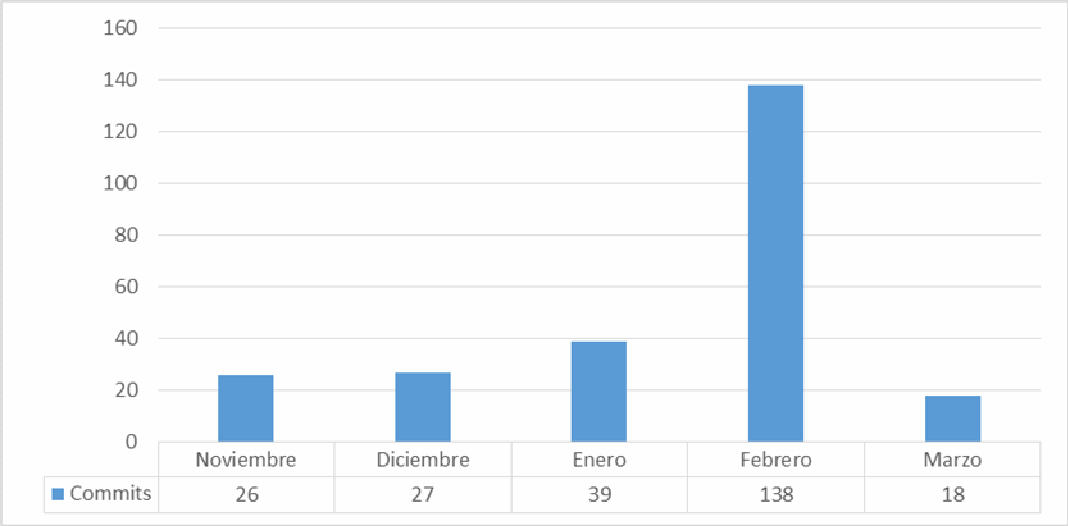
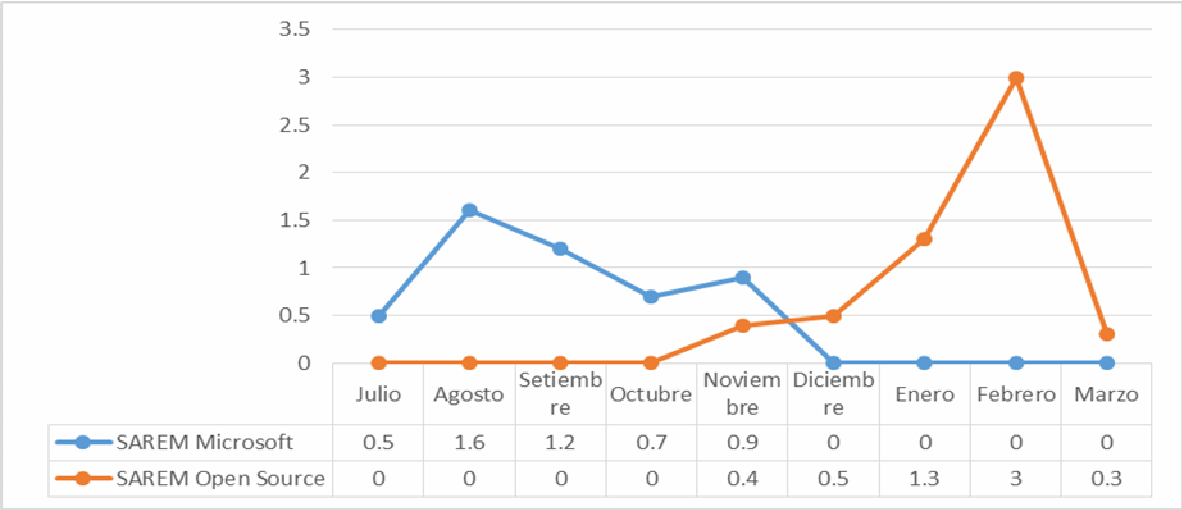


Figura G6 – Commits por mes proyecto SAREM (tecnologías open source) en fase de implementación.



	Julio	Agosto	Setiembre	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo
SAREM Microsoft	0.5	1.6	1.2	0.7	0.9	0	0	0	0
SAREM Open Source	0	0	0	0	0.4	0.5	1.3	3	0.3

Figura G7 – Commits/Hora por mes proyecto SAREM Microsoft vs SAREM Open Source en fase de implementación.

Se puede ver en la gráfica de la Figura G7 que el número de Commits/Hora del proyecto SAREM Microsoft comienza con una curva más empinada que la del proyecto SAREM Open Source. Esto se puede explicar debido a que la curva de aprendizaje de las tecnologías Microsoft es mayor ya que se utilizan diferentes lenguajes en el frontend y el backend del proyecto, mientras que en el proyecto SAREM Open Source se utiliza un único lenguaje de programación JavaScript tanto para el backend como para el frontend, que fomenta la productividad.

En la gráfica correspondiente al proyecto SAREM Open Source se puede ver que el número de Commits/Hora aumenta al transcurrir los meses llegando a un pico en febrero y luego disminuye en marzo. El mes de febrero ha sido el mes que llevo más esfuerzo de programación ya que teníamos la meta de finalizar el proyecto en marzo de 2016. Esta grafica presenta más productividad en comparación con la de la gráfica SAREM Microsoft. Esto puede ser explicado por las tecnologías utilizadas pero también es debido a que dispusimos de más tiempo disponible para el proyecto, al no coincidir con el semestre electivo de la facultad, se realizó un esfuerzo mayor de programación.

En la gráfica correspondiente al proyecto SAREM Microsoft se observa que el número de Commits/Hora al pasar los meses crece entre julio y agosto y vuelve a decrecer en los meses de setiembre y octubre, aumentando nuevamente entre octubre y noviembre y finalmente decayendo entre noviembre y diciembre fruto de su finalización. Estas irregularidades se pueden explicar debido a que el proyecto SAREM Microsoft coincidió con el semestre electivo de la facultad, en la cual se cursaron materias por parte de los integrantes del grupo y esto saco tiempo para el proyecto.

Entre octubre y noviembre existe un crecimiento debido al esfuerzo que se realizó para participar en Ingeniería deMuestra 2015.

9.3 Líneas de Código

La Tabla G1 muestra la cantidad de líneas de código (excluyendo líneas en blanco, librerías externas y frameworks) implementadas, obtenidas con una herramienta libre llamada GitStats [90] para el proyecto SAREM con tecnologías de Microsoft.

Tabla G1 - Cantidad de líneas de código proyecto SAREM (Microsoft).

Lenguaje	Archivos	Líneas	Líneas/Archivo
C#	138	16542	119
CSHTML	100	9136	91
HTML	14	986	70
CSS	10	200	20
JavaScript	15	630	42
XML	1	13	13
Total	278	27507	

A partir de la cantidad de horas (aproximadamente 153 horas) y las 27507 líneas de código implementadas para el proyecto SAREM con tecnologías .NET, se puede calcular **una productividad de 178 líneas de código por hora.**

La Tabla G2 muestra la cantidad de líneas de código (excluyendo líneas en blanco, librerías externas y frameworks) implementadas, obtenidas con una herramienta libre llamada GitStats [90] para el proyecto SAREM con tecnologías open source.

Tabla G2 - Cantidad de líneas de código proyecto SAREM (Open Source).

Lenguaje	Archivos	Líneas	Líneas/Archivo
HTML	89	5414	60
CSS	6	400	68
JavaScript	180	175225	973
XML	1	43	43
SQL	4	277	69
Total	280	181359	

A partir de la cantidad de horas (aproximadamente 245 horas) y las 181359 líneas de código implementadas para el proyecto SAREM con tecnologías open source, se puede calcular **una productividad de 740 líneas de código por hora.**

La diferencia de productividad entre los dos proyectos puede explicarse por el uso de diferentes tecnologías. En el primer proyecto se utilizaron tecnologías de Microsoft, el frontend fue implementado con ASP.NET MVC[22] y bibliotecas de JavaScript como jQuery [29]. El backend fue implementado con un lenguaje diferente C#.

La diferencia de lenguajes utilizados entre el frontend y el backend es un factor que disminuye la productividad. Dado que un programador puede estar familiarizado únicamente con tecnologías del frontend y no saber el lenguaje de programación que se utiliza en el backend, o viceversa. Esto lleva a que no se pueda desempañar con el mismo nivel de productividad en las diferentes capas.

En el proyecto SAREM implementado con tecnologías open source, se utilizó un mismo lenguaje tanto para el backend como para el frontend de la aplicación. Node.js [34] y Angular.js[20] están escritos en JavaScript. Esto es un factor que incrementa la productividad si ya se tiene conocimiento del lenguaje.

Otro factor importante que aumento la productividad del segundo proyecto fue utilizar el framework Angular.js [20] para implementar el frontend de la aplicación. Angular.js provee muchas más características que no se consiguen con el uso de una bibliotecas como jQuery [29] . Entre ellas podemos destacar:

- Patrón MVW (Similar al patrón MVC)

- Unión de dos vías de datos (Two way data binding)
- Plantillas
- Validación de formularios
- Comunicación con el servidor
- Reutilización de componentes

Además cabe destacar también que las horas dedicadas a SAREM (Open Source) fueron mayores, dado que dispusimos de más tiempo de dedicación al no coincidir el tiempo de implementación con materias del semestre de facultad.

10. Pruebas

10.1 Pruebas Unitarias

Se realizaron pruebas unitarias para cada método personalizado en la API, dado que el framework autogenera los puntos de acceso por modelo generando un ABM.

Las pruebas unitarias se realizaron sobre los siguientes componentes de integración.

- OpenEMPI, se realizaron pruebas con aserciones corroborando la integración en el caso de registro.
- GCM (Google Cloud Messaging), este módulo se creó un conjunto de pruebas unitarias para el envío de Push Notifications a la plataforma móvil.
- Cola de mensajes, dado que el sistema se interrelaciona con una cola de mensajes para el envío asincrónico de mensajes, se debió crear pruebas de publicación y subscripción, de modo tal de garantizar el correcto funcionamiento de los procesos en background para el envío de notificaciones a plataforma móvil y SMTP.

10.2 Pruebas de Integración

En conjunto con las pruebas unitarias orientadas a servicios externos, se generaron casos de generación de datos de modo tal de probar la integración total del sistema.

Para ello se generaron scripts que permiten la generación de datos de los casos de uso más habituales, a tal punto que se obtiene un buen conjunto de pruebas para detectar errores en los componentes UI.

A continuación se listan los casos de usos más utilizados en el flujo normal de SAREM.

- Listar Especialidades.
- Listar Médicos.
- Listar Consultas dada una Especialidad.
- Listar Consultas dada una Especialidad y un Médico.
- Listar Notificaciones.

Se crearon datos para cada Modelo del dominio mencionado anteriormente, el número final de registros se detalla en la **Tabla P1**.

Tabla P1 – Número final de registros

Tabla	**Cantidad de Filas**
Especialidades	24
Persona (Pacientes y Profesionales)	1706
Consultas	1006
Notificaciones	1024000
Locales	1004

Las especialidades se generaron de forma manual consultando en diferentes sitios sobre las especializaciones en el área de Medicina. [91]

En cuanto a las personas (Médicos y Pacientes) se generaron de forma aleatoria utilizando una biblioteca Node.js llamada chance [92] para la generación de atributos aleatorios.

Las consultas se crearon con fecha de inicialización en el año 2016, de modo tal de mantener el dominio de forma balanceada para evaluar las consultas en la base de datos de forma completa. Es decir, considerando consultas activas, filtradas por especialidad y Médico dependiendo el caso de prueba a ejecutar.

Observar que los datos generados no consideran tendencias en cuanto a los horarios de las consultas, pudiendo de esta forma generar fechas de inicialización que no coinciden con el horario habitual de la zona.

Las notificaciones se generaron de una forma matemáticamente atractiva. Para el primer paso se generaron aleatoriamente 1000 notificaciones asociadas a personas del sistema con datos aleatorios. Una vez generadas las notificaciones se procederá a la ejecución de 10 veces la sentencia SQL:

```
insert into notificacion(data,tiponotificacionid,accountid,fecha) (select data,tiponotificacionid,accountid, NOW() from notificacion)
```

Se trata de una exponencial N 2^N por lo cual realizando la ejecución 10 veces se obtendrán 1024000 registros, lo suficiente como para realizar una prueba de stress adecuada a las características del sistema.

En las pruebas a realizar se definirán los criterios utilizados sobre las notificaciones, puesto que realizar invocaciones sobre un volumen de datos tan importante sin filtros interesantes no aporta al objetivo primordial de las pruebas. Incluso se optimiza el número de ejecuciones de comandos, delegando al manejador de base de datos la creación de registros “aleatorios” en pasos sencillos.

10.3 Pruebas de Carga

10.3.1 Infraestructura

Para las pruebas de simulación se realizó la instancia provista por Amazon de forma gratuita, a continuación se mencionará el Hardware utilizado en dicha instancia así como los aspectos más relevantes de Software.

- **Hardware:**
    - 1 vCPU Xeon frecuencia máxima 3.3Ghz
    - 1 GB RAM
    - 15GB SSD Almacenamiento

Por más información acerca de hardware utilizado se puede indagar en el sitio web de Amazon la instancia virtual utilizada [93], la misma se trata de una instancia T2 modelo micro (t2.micro) de propósito general.

Lo importante a destacar es que los resultados de las pruebas realzan el potencial del lenguaje de programación seleccionado, logrando alta performance con una instancia pequeña únicamente dedicada a la instanciación de la API.

- **Software:**

En cuanto a las especificaciones del Sistema Operativo Linux, se trabajó con Ubuntu Linux 10.04, a continuación se adjunta la salida de la descripción completa de la versión. Para ello se ejecuta el siguiente comando en un intérprete de comandos.

```
ubuntu@sarem:~$ cat /etc/os-release
NAME="Ubuntu"
VERSION="14.04.3 LTS, Trusty Tahr"
PRETTY_NAME="Ubuntu 14.04.3 LTS"
VERSION_ID="14.04"
```

En cuanto al framework node utilizado nuevamente se adjunta la versión brindada por el intérprete de comandos.

```
ubuntu@sarem:~$ node -v
v0.10.25
```

La base de datos como se describió anteriormente, se utiliza una instancia de AMAZON RDS con manejador PostgreSQL, para ello se logra una conexión al mismo mediante

Psql [94], una vez establecida la conexión, se indaga cual es la versión exacta del manejador.

```
postgres=> select version();
                                        version
-----------------------------------------------------------------------------------------------------------
 PostgreSQL 9.4.5 on x86_64-unknown-linux-gnu, compiled by gcc (GCC) 4.8.2 20140120 (Red Hat 4.8.2-16), 64-bit
```

El servidor web se utilizó NGINX [85] nuevamente se depliega la versión utilizando intérprete de comados.

```
ubuntu@sarem:~$ /usr/sbin/nginx -v
nginx version: nginx/1.4.6 (Ubuntu)
```

Se utilizó un Manejador de Procesos (Process Manager) para instanciar el backend en modo cluster si es necesario. Para ello se recurrió al Manejador de Procesos PM2 [95].

```
ubuntu@sarem:~$ pm2 --version
1.0.2
```

En resumen, contamos con la siguiente infraestructura para las pruebas de carga:

- Sistema Operativo Ubuntu 14.04.3 LTS.
- Manejador de base de datos PostgreSQL 9.4.5 (En pruebas de desarrollo se recurrió a la versión 9.5.1 con éxito).
- NodeJS 0.10.25
- NGINX 1.4.6
- PM2 v1.0.2

En cuanto a memoria disponible o utilizada por los principales procesos en involucrados en cuestión, se obtiene un remanente de 145000 KB disponibles, esto es suficiente para realizar las pruebas de forma óptima.

```
ubuntu@sarem:~$ free
             total       used       free     shared    buffers     cached
Mem:       1016324     870868     145456      16256      99808     338272
```

Espacio en disco no es relevante para las pruebas, dado que el manejador de base de datos PostgreSQL se encuentra en otro servidor distribuido utilizando el servicio AMAZON RDS descrito anteriormente.

10.3.2 Descripción de Pruebas

Las pruebas realizadas consideran los casos de usos más importantes a tener en cuenta. Esto implica desde el punto de vista de usuario, las interfaces más utilizadas de la API. Por lo cual, los casos de uso de Agendar Consulta, Listar Notificaciones, Listar Médicos Referencia, Listar Consultas Agendadas y Listar especialidades, fueron consideradas prioridad por la esencia del Proyecto en sí mismo.

Un punto importante a destacar es el entorno de pruebas utilizado. Es de notorio conocimiento que existen diversos obstáculos cuando que impiden tener un análisis minucioso de la performace del sistema. A continuación listamos algunos de los problemas encontrados para realizar las pruebas y como se decidió proceder para eliminar los mismos y obtener datos para establecer comparaciones realistas.

Dentro de los obstáculos encontrados para realizar, se destacan los siguientes:

- Latencia de Red: El factor seguramente más importante a tener en cuenta al realizar las pruebas, el servidor seleccionado se ubica en Oregon US, por lo cual de antemano se podría obtener datos divergentes cuando comparamos tiempos de respuesta entre un servidor que se ubica geográficamente más cerca del usuario que lo invoca. Para ello realizamos una medición primaria de la latencia de la red para tener en cuenta el tiempo de respuesta real de la API, independientemente de la geolocalización del servidor.
- Hardware de Procesamiento: Seguramente con Hardware con una capacidad computacional superior en orden de núcleos, se reduciría el tiempo de procesamiento de consultas de forma drástica. Únicamente con la idea de paralelizar procesos y habilitar una mayor cantidad de Process Workers en el balanceador de carga, los tiempos de respuestas concurrentes se verían

mejorados en forma lineal a la cantidad de unidades de procesamiento.

- Arquitectura distribuida: Es un punto de debate, para las pruebas realizadas se establecieron únicamente una instancia con base de datos distribuida, así como el Cliente Web y el servidor Backend (API). Para evaluar la performace sobre una instancia, una arquitectura monolítica posee un rendimiento superior, en cambio, si a la solución le agregamos la capacidad de escalar horizontalmente, el potencial se ve reflejado cuando ejecutamos dos o más instancias para evaluar la performace.

Para ello se puede utilizar plataformas como AMAZON Beanstalk [96] para escalar rápidamente el Backend que es posiblemente la instancia que más se utilice a lo largo del tiempo de vida de la aplicación. Queda como trabajo a futuro realizar pruebas de la aplicación utilizando una infraestructura como la mencionada. Para este caso se establecieron pruebas únicamente para una instancia de hardware, dejando la Arquitectura orientada a Microservicios para escalar rápidamente y satisfacer las necesidades de carga del sistema.

En cuanto a las pruebas realizadas, como se describió anteriormente se utilizaron los casos de uso más relevantes de la aplicación.

Para las mismas se definió el siguiente conjunto de pruebas con el objetivo primordial de medir la performance del sistema en cuanto a tiempos de respuesta promedio, mínimo y máximo con dos variables establecidas, la cantidad de solicitudes a realizar al Backend y la concurrencia entre las mismas.

En cuanto a la cantidad de solicitudes realizadas, básicamente definimos un número con el cual nos permita obtener datos estadísticos de tal forma que se pueda realizar un promedio y una media del tiempo de respuesta, tiempo de espera y de conexión.

A lo que refiere a concurrencia, demostraremos como la misma puede influir en el sistema utilizando un número variable de conexiones simultáneas de esta forma se logrará probar si el sistema tiene la capacidad de respuesta ante eventos concurrentes en un intervalo de tiempo acotado. Se estudiará el impacto del mismo en el tiempo de respuesta promedio para una única instancia, así como también la cantidad de respuestas generadas.

En resumen, para las pruebas tomaremos dos variables

- Cantidad de solicitudes a la API.
- Cantidad de procesos concurrentes destinados a ejecutar la cantidad de solicitudes definidas previamente.

Para la primera variable se considera un número razonable 1000 ya que genera un dominio de resultados lo suficientemente amplio para generar promedio, media, cota inferior y superior.

En cuanto la cantidad de procesos concurrentes, se opta en realizar 10, 100 y 500 procesos concurrentes, de forma tal de evaluar los tiempos de respuestas ejecutando 1000 solicitudes al servidor entre los números de procesos concurrentes mencionados.

Por tanto tendremos los siguientes casos de prueba:

- Cantidad de solicitudes 1000.
- Cantidad de procesos concurrentes 10, 100, 500.
- Métodos a ejecutar: Listar Consulta por Especialidad, Listar Consultas por Especialidad y Médico, Listar Especialidades, Listar Notificaciones Usuario, Listar Usuarios Profesionales.
- En total se obtendrán 15 casos de prueba, teniendo en cuenta diversos contextos de ejecución que serán descritos a continuación.

Para la ejecución se utilizará la herramienta JMeter brindada por el equipo de Apache [97], para ello se brinda tres contextos de ejecución en creados con el mismo Jmeter.

```
ubuntu@sarem:~/test$  ls -ltr | grep jmx
-rw-r--r-- 1 leonardo leonardo  89664 Apr  3 00:49 stress-remote.jmx
-rw-r--r-- 1 leonardo leonardo 107808 Apr  3 15:12 stress-local.jmx
-rw-r--r-- 1 leonardo leonardo  90800 Apr  3 13:55 stress-local-nginx.jmx
```

- El primer contexto stress-remote, realiza el conjunto de prueba utilizando la API de forma remota, geolocalizada en Oregon US.
- El segundo contexto aplica a un test de stress local, sin optimizaciones.
- El tercer test aplica las optimizaciones y la aplicación de estrategia cache.

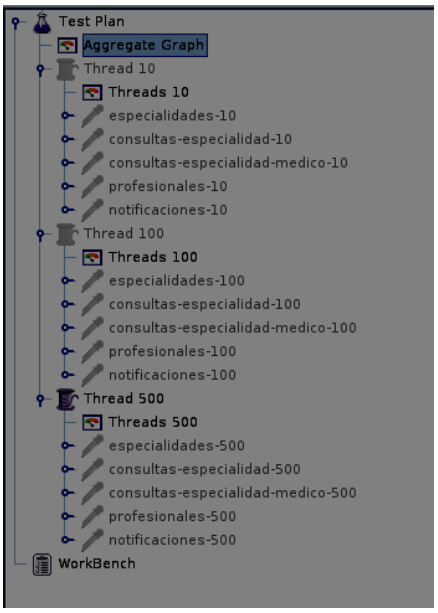


Figura P1 – Captura de cómo se define el conjunto de pruebas de stress para lograr el entorno de configuración.

.

Tabla P2 - Conjunto de pruebas y puntos de acceso mencionados.

Prueba	**Punto de Acceso (Endpoint)**
Listar Especialidades	/especialidades
Listar Consultas Especialidad (Especialidad # 2)	/api/consultas?filter=%7B%22where%22%3A%7B%22especialidadId%22%3A%202%7D%7D
Listar consultas por especialidad y profesional. (Especialidad #3, Profesional #155)	/api/consultas?filter=%7B%22where%22%3A%7B%22especialidadId%22%3A%203%2C%20%22medicoId%22%3A%20155%7D%7D
Listar profesionales	/api/personas?filter=%7B%22where%22%3A%7B%22perfil%22%3A%22medico%22%7D%7D
Listar Notificaciones (Usuario # 1)	/notificaciones?filter=%7B%22where%22%3A%7B%22accountId%22%3A%201%7D%7D

10.3.3 Contextos de Ejecución

- Local

Con la ejecución de las pruebas en el contexto local, lo que se concluirá el tiempo de procesamiento promedio, cantidad de respuestas por segundo, mínimos, máximos y media.

Para ello se utilizará una instancia local, de esta forma se elimina un factor importante como es el Round Trip Time [98], con esta variable anulada se puede concluir cuáles serán los tiempos de procesamiento de la API con un juego de datos predefinido en la sección anterior.

Se analizarán los resultados listados a continuación y las mejoras realizadas o a realizar en eventuales trabajos a futuro.

Para el contexto localhost se obtuvieron los siguientes resultados:

Tabla P3 – Contexto Local: 1000 Peticiones, Nivel de concurrencia 10.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-10	1000	25	23	30	22	34	0.00%	39.1	67.2
consultas-especialidad-10	1000	640	627	670	169	694	0.00%	11.6	788.5
consultas-especialidad-medico-10	1000	39	36	44	35	49	0.00%	37.3	30.4
profesionales-10	1000	108	107	113	32	114	0.00%	28.4	421.1
notificaciones-10	1000	237	238	244	65	272	0.00%	20.8	512.5
TOTAL	5000	210	107	627	22	694	0.00%	12	263.8

1000 Peticiones, Nivel de concurrencia 10

Tabla P4 – Contexto Local: 1000 Peticiones, Nivel de concurrencia 100.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-100	1000	148	150	187	30	194	0.00%	47.7	82
consultas-especialidad-100	1000	5742	5971	6027	99	6273	0.00%	15.2	1039.5
consultas-especialidad-medico-10	1000	255	258	304	11	326	0.00%	46.5	38
profesionales-100	1000	598	617	666	50	675	0.00%	44.8	664.6
notificaciones-100	1000	1202	1206	1299	57	1485	0.00%	40.2	991.8
TOTAL	5000	1589	617	5971	11	6273	0.00%	8.6	190.5

1000 Peticiones, Nivel de concurrencia 100

Tabla P5 – Contexto Local: 1000 Peticiones, Nivel de concurrencia 500.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-500	1000	270	228	445	23	489	0.00%	133.5	229.3
consultas-especialidad-500	1000	25364	29910	32509	99	33205	0.00%	14.7	999.2
consultas-especialidad-medico-50	1000	909	699	1530	9	1615	0.00%	134.3	109.7
profesionales-500	1000	3848	3684	5031	308	5791	0.00%	73.8	1095.7
notificaciones-500	1000	6716	7629	9247	40	9841	0.00%	46.6	1149.1
TOTAL	5000	7422	3572	29910	9	33205	0.00%	23.1	509.1
1000 Peticiones, Nivel de concurrencia 500									

Los tiempos de ejecución con concurrencia son prácticamente inaceptables, un tiempo de respuesta promedio mayor a un segundo compromete la performance del sistema de manera drástica. Por ello se adoptaron medidas para reducir los tiempos de respuestas. Considerando que se cuenta con una única instancia con capacidad computacional baja, mejorar los resultados en la misma, permitiría en planes posteriores del proyecto considerar más instancias y escalar horizontalmente. Solventando de esa forma el problema de la concurrencia.

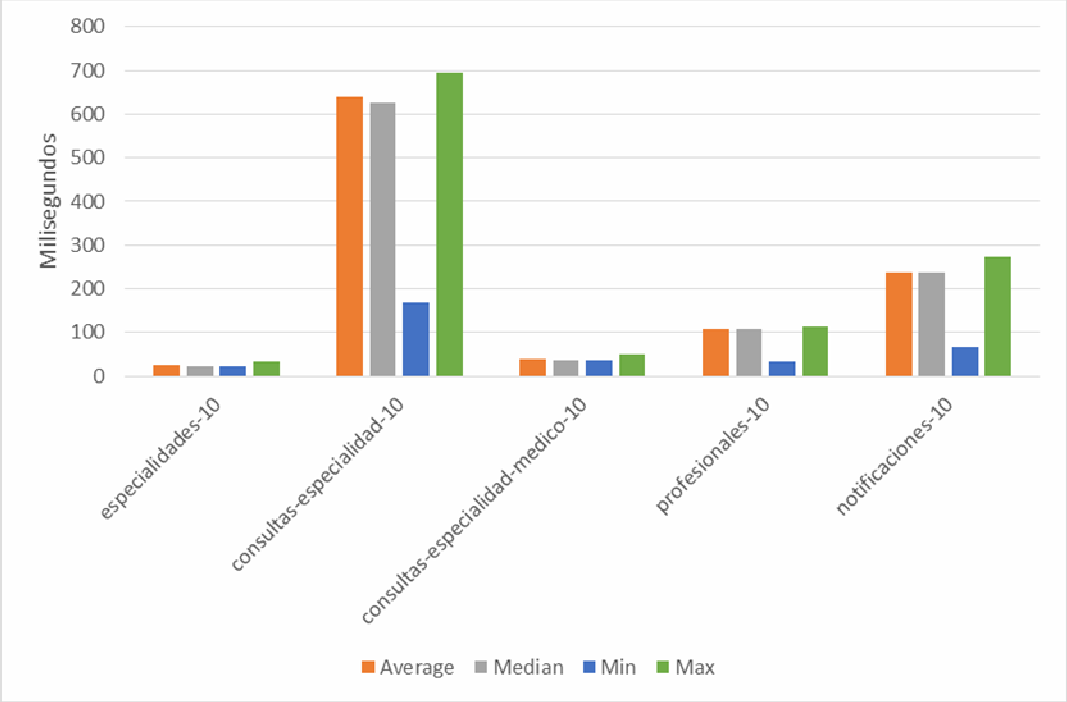


Figura P2 – Contexto Local: 1000 Peticiones, Nivel de concurrencia 10.

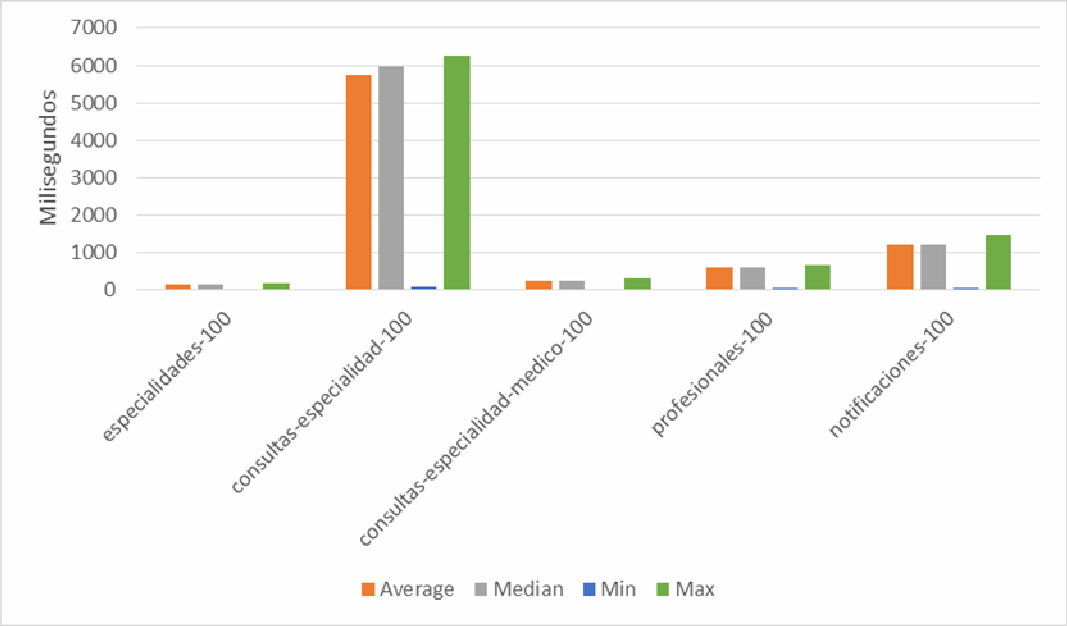


Figura P3 – Contexto Local: 1000 Peticiones, Nivel de concurrencia 100.

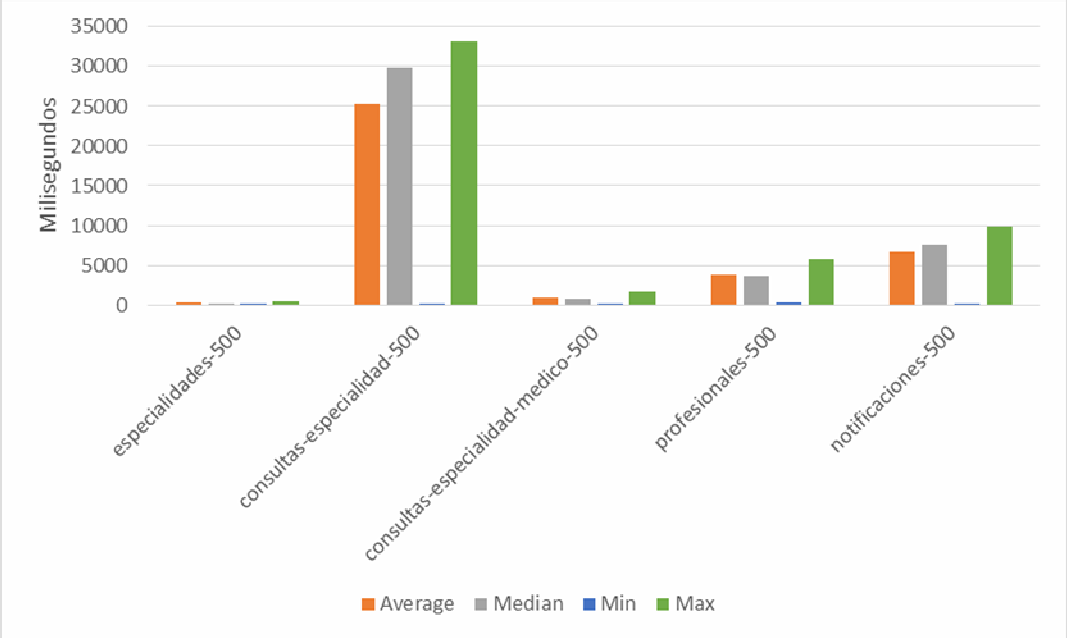


Figura P4 – Contexto Local: 1000 Peticiones, Nivel de concurrencia 500.

Se pueden extraer una serie de conclusiones en función de los resultados obtenidos. Primeramente se debe tener en cuenta la complejidad de la consulta en cada caso, las consultas de prueba #2 y #3 relacionan una mayor cantidad de tablas que las anteriores, por lo cual es necesario realizar un trabajo de performance en la base de datos. Para ello deberán agregar índices en las columnas localid, medicoid, especialidadid, de forma tal de reducir la complejidad en los JOINs de las consultas, cambiando el plan de acceso y por ende mejoras en los tiempos de ejecución.

A continuación se presentan una serie de pasos a mejorar dadas las circunstancias:

- Agregar índices en la base por Especialidad, Médico y Local en la tabla de Consultas, de esta forma se tratará de normalizar los tiempos de respuesta de la prueba #2 y por ende #3.
- Agregar índice Hash en la tabla Personas, ya que la misma es compartida entre pacientes y profesionales, en este caso el agregado de un índice por el discriminador resultaría en una mejora de sustancial en performance.
- Optimizar el tiempo de respuesta agregando una capa extra y almacenando las respuestas estáticas por un intervalo de tiempo definido (5 segundos). Esta técnica se conoce como caching, logra resultados importantes ya que reduce la cantidad de consultas en la base de datos, impide procesamiento por parte del Backend, reduciendo de esta forma el tiempo de procesamiento en servidor.
- El caso de prueba número 2 será reducido debido a que se retornan 257 elementos que cumplen con el predicado de la consulta. En en caso de la aplicación, la misma presenta paginación, por lo cual los resultados están acotados a 15 elementos. La prueba será acotada a 50, 100, 150 registros, para establecer una comparación entre el tiempo de respuesta en función de la cantidad de elementos. En ese caso se estudia la relación existente en los tiempos de respuesta en función del contenido de la respuesta.

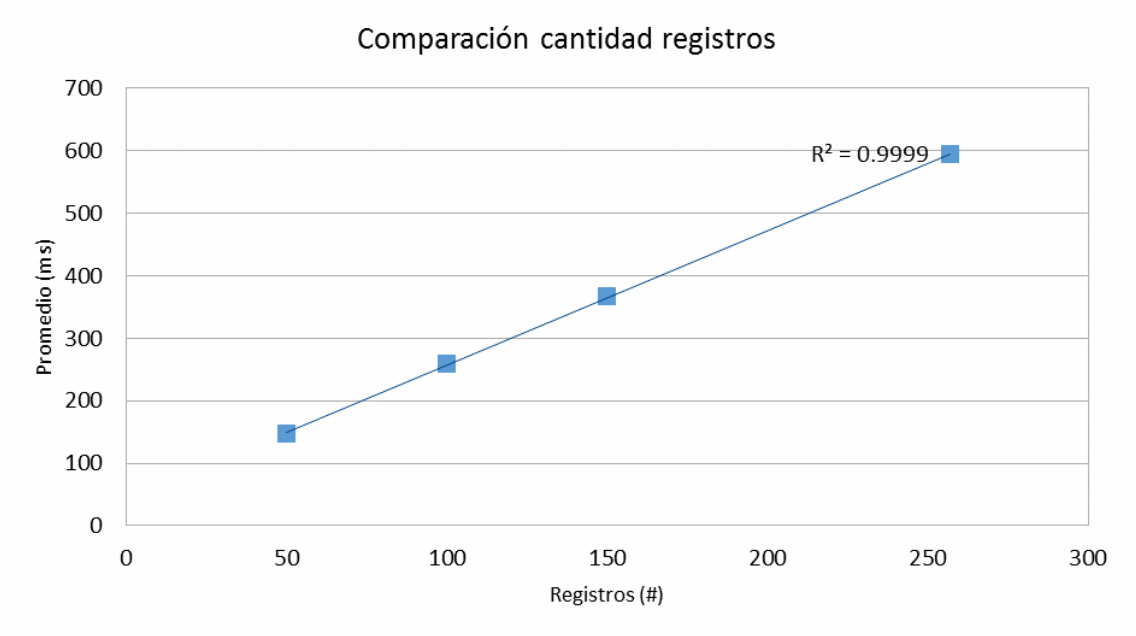


Figura P5 – Comparación cantidad de registros.

Se puede concluir que existe una relación lineal con coeficiente 1, por lo cual se puede predecir el comportamiento en función de la cantidad de registros obtenidos.

Este caso es habitual ya que el parsing JSON tiene un tiempo promedio, además se debe considerar que la respuesta no sólo obtiene los datos de una tabla, sino que se incluyen los datos de local, médico y especialidad, por lo cual el tiempo de ejecución se considera normal.

- Local-NGINX

Con este contexto se demostrará el potencial de la estrategia de Cache[99][100] en el servidor web (NGINX ofrece técnicas de caching, load balancer y servidor de contenido estático).

La desventaja que se obtiene con esta estrategia es la consistencia real de los datos servidos, es decir, si se agrega por ejemplo una consulta, el contenido real será renderizado luego de N segundos. Esta técnica es usualmente utilizada cuando la variación de los datos es esporádica, por ejemplo los Profesionales, consultas, locales, especialidades. Básicamente todos los modelos reflejados en el dominio de la aplicación.

De lo mencionado anteriormente podemos concluir que agregar una capa intermedia que permita el caching de forma independiente a la implementación de la aplicación,

es la forma más eficiente incluso desde el punto de vista arquitectónico de la aplicación.

Tabla P6 – Contexto Local-NGINX: 1000 Peticiones, Nivel de concurrencia 10.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-10	1000	1	2	2	0	29	0.00%	49.3	87.1
consultas-especialidad-10	1000	3	2	5	2	74	0.00%	47.2	3223.2
consultas-especialidad-medico-10	1000	2	2	3	1	64	0.00%	49.2	42.6
profesionales-10	1000	1	1	3	0	19	0.00%	48.9	728.9
notificaciones-10	1000	1	1	2	1	107	0.00%	46.8	1158.2
TOTAL	5000	2	2	2	0	107	0.00%	22.3	492.8
1000 Peticiones, Nivel de concurrencia 10									

Tabla P7 – Contexto Local-NGINX: 1000 Peticiones, Nivel de concurrencia 100.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-100	1000	8	7	9	6	62	0.00%	52.6	92.9
consultas-especialidad-100	1000	11	9	18	4	19	0.00%	56.1	3829.5
consultas-especialidad-medico-100	1000	9	7	9	2	22	0.00%	56.2	48.6
profesionales-100	1000	8	7	14	6	75	0.00%	54	804.7
notificaciones-100	1000	8	6	8	4	18	0.00%	57.1	1412.5
TOTAL	5000	9	7	15	2	75	0.00%	24.4	540.1
1000 Peticiones, Nivel de concurrencia 100									

Tabla P8 – Contexto Local-NGINX: 1000 Peticiones, Nivel de concurrencia 500.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-500	1000	75	66	124	3	329	0.00%	187.9	331.9
consultas-especialidad-500	1000	59	37	117	8	152	0.00%	153.2	10452
consultas-especialidad-medico-500	1000	58	46	113	6	123	0.00%	163.8	141.7
profesionales-500	1000	58	48	122	6	122	0.00%	146.9	2188.2
notificaciones-500	1000	71	39	123	2	289	0.00%	202.3	5003.9
TOTAL	5000	64	37	123	2	329	0.00%	31.2	690.1
1000 Peticiones, Nivel de concurrencia 500									

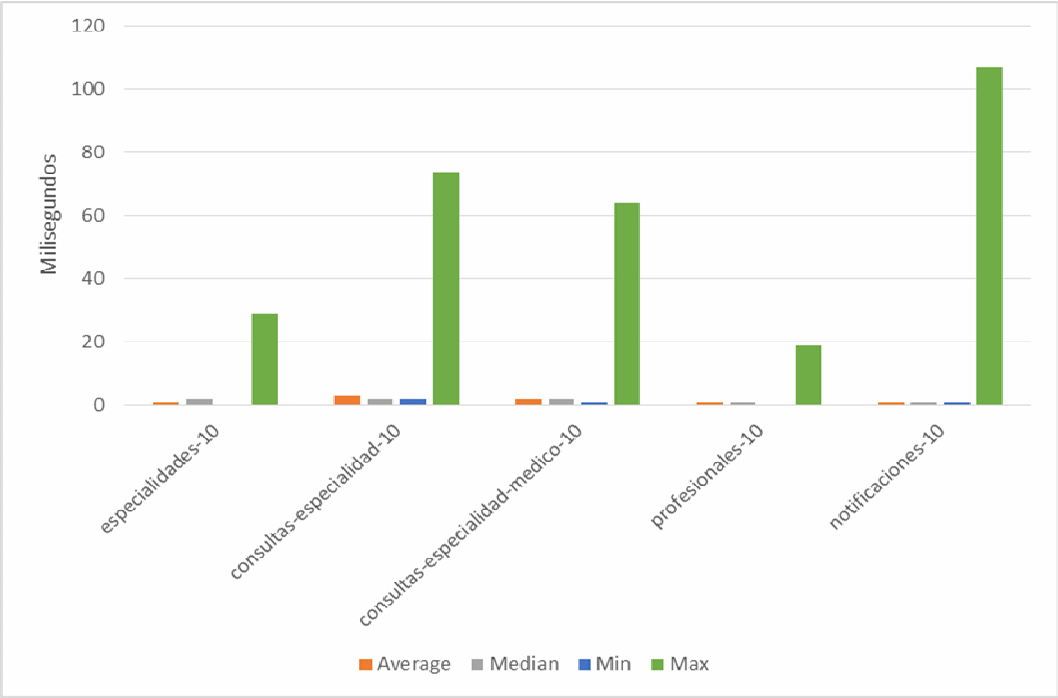


Figura P6 – Contexto Local-NGINX: 1000 Peticiones, Nivel de concurrencia 10.

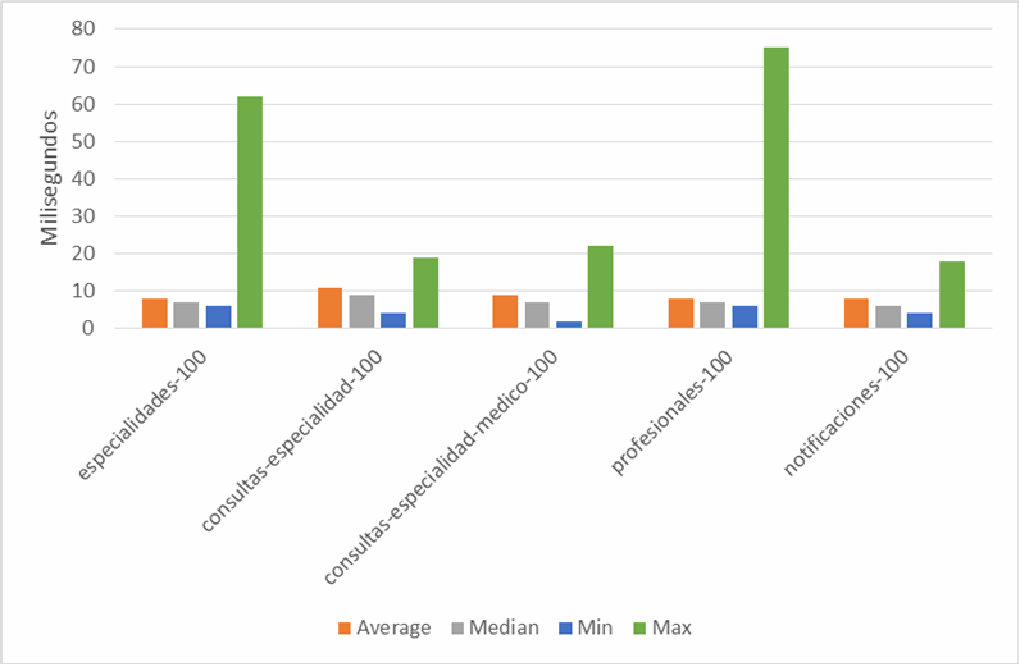


Figura P7 – Contexto Local-NGINX: 1000 Peticiones, Nivel de concurrencia 100.

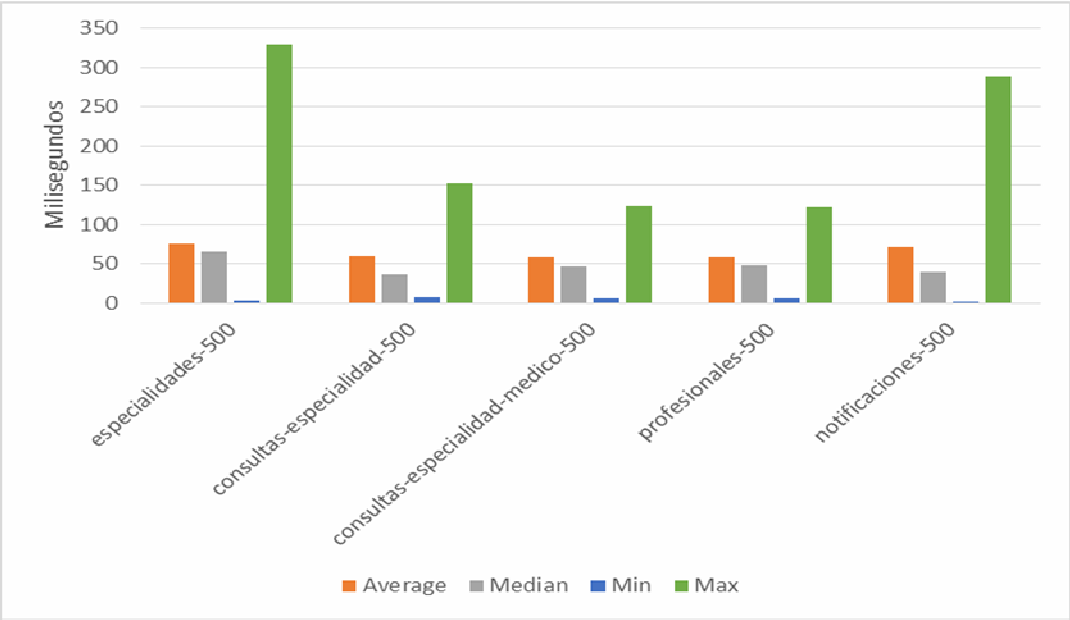


Figura P8 – Contexto Local-NGINX: 1000 Peticiones, Nivel de concurrencia 500.

Se observa que la mejora es sustancial utilizando la técnica cache. La reducción de consultas a la base así como el parsing de los resultados en el servidor, permiten obtener respuestas prácticamente sin procesamiento. Esto se debe gracias al balanceador de carga, que al presentar la característica de permitir caching, se puede utilizar sin problema alguno y sin agregar overhead a la solución.

Punto a destacar, es que el aumento de nivel de concurrencia, mejora la cantidad de respuestas en las pruebas, pero aumenta el tiempo promedio de las mismas. Por tanto se debe tener en cuenta este factor para determinar un balance entre nivel de concurrencia y tiempo de respuesta promedio.

- Remoto

Dado los casos contextos anteriores y gracias a los resultados obtenidos se pudieron estudiar y ajustar parámetros importantes para la mejora de la performance.

La inclusión de cache así como evaluar el tiempo de respuesta en función a la cantidad de datos a parsear en JSON, resultó en la mejor solución posible hasta el momento. Cabe destacar que se pueden realizar más ajustes a nivel de esquema de la base para mejorar aún más los resultados.

El único contexto faltante es el remoto, con todas las variables adversas relacionadas a latencia de red, conexiones simultáneas, entre otros problemas específicos que surgen de un contexto distribuido.

Para ello se adjunta los datos obtenidos y se evalúa cuáles son los resultados de forma global y como mejorarlos.

Tabla P9 – Contexto Remoto: 1000 Peticiones, Nivel de concurrencia 10.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-10	1000	248	244	257	230	745	0	36.87179676	65.13748571
consultas-especialidad-10	1000	315	287	398	253	1114	0	30.08061605	2052.063378
consultas-especialidad-medico-10	1000	245	243	253	231	580	0	37.64209892	32.56839246
profesionales-10	1000	253	248	260	230	769	0	36.72959671	547.248286
notificaciones-10	1000	259	249	263	233	995	0	36.03603604	891.152942
TOTAL	5000	264	249	291	230	1114	0	19.34078856	427.3520393
1000 Peticiones, Nivel de concurrencia 10									

Tabla P10 – Contexto Remoto: 1000 Peticiones, Nivel de concurrencia 100.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-100	1000	271	246	292	230	669	0	257.6655501	455.2135001
consultas-especialidad-100	1000	1500	1239	2341	472	9381	0	49.44131316	3372.829072
consultas-especialidad-medico-100	1000	271	246	302	229	536	0	256.5418163	221.9723055
profesionales-100	1000	423	370	561	248	1191	0	178.0626781	2653.032004
notificaciones-100	1000	643	556	971	245	3767	0	99.02951079	2448.951351
TOTAL	5000	622	394	1249	229	9381	0	43.74950782	966.6865955
1000 Peticiones, Nivel de concurrencia 100									

Tabla P11 – Contexto Remoto: 1000 Peticiones, Nivel de concurrencia 500.

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
Especialidades-500	1000	482	451	646	228	1761	0	322.8931224	570.4925508
Consultas-especialidad-500	1000	7433	5570	15407	926	29005	0	31.98260146	2181.818934
Consultas-especialidad-medico-500	1000	624	509	945	231	3515	0	191.424196	165.6275424
Profesionales-500	1000	1830	1573	2732	567	4558	0	133.8329764	1994.038028
Notificaciones-500	1000	2775	1963	5362	474	8985	0	95.38344143	2358.792826
TOTAL	5000	2629	1454	6471	228	29005	0	47.23977967	1043.810384
1000 Peticiones, Nivel de concurrencia 500									

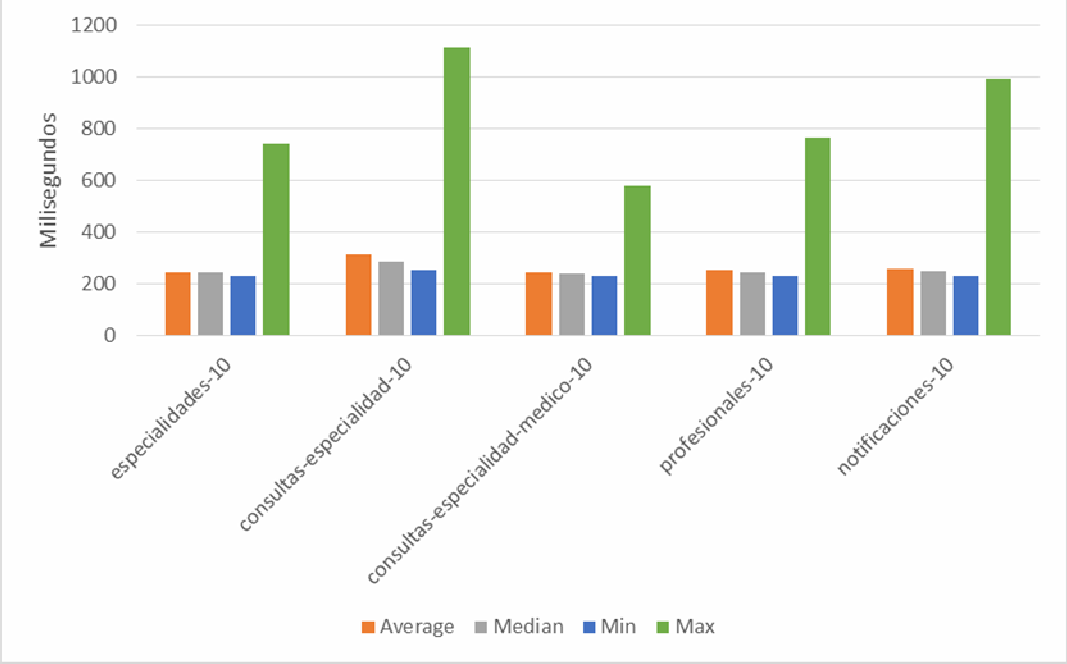


Figura P9 – Contexto Remoto: 1000 Peticiones, Nivel de concurrencia 10.

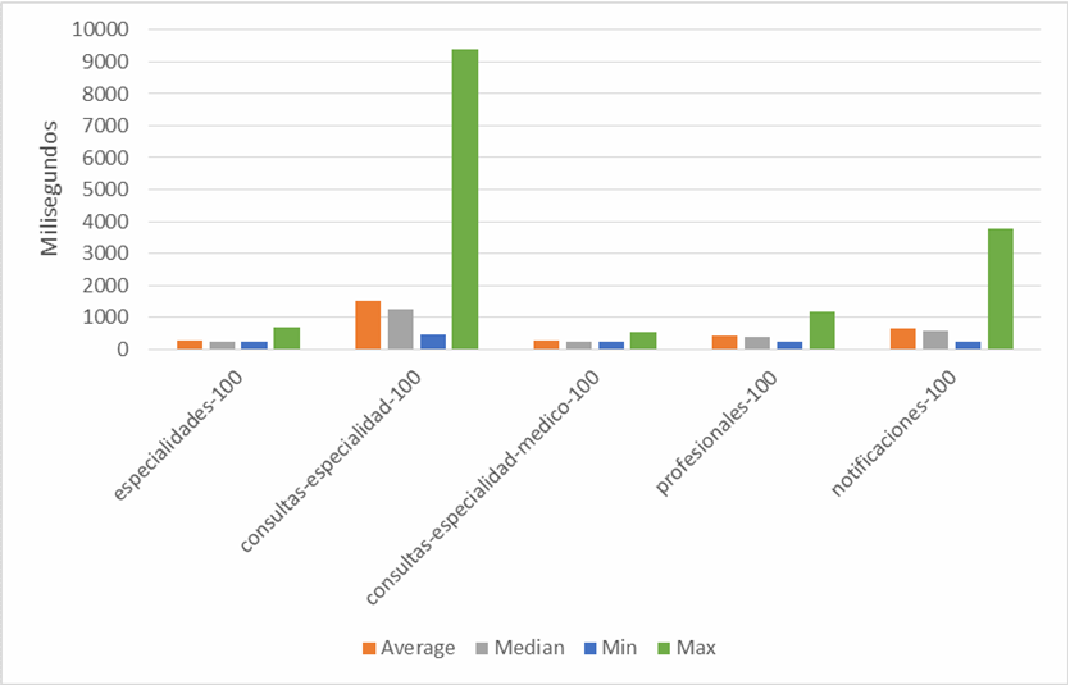


Figura P10 – Contexto Remoto: 1000 Peticiones, Nivel de concurrencia 100.

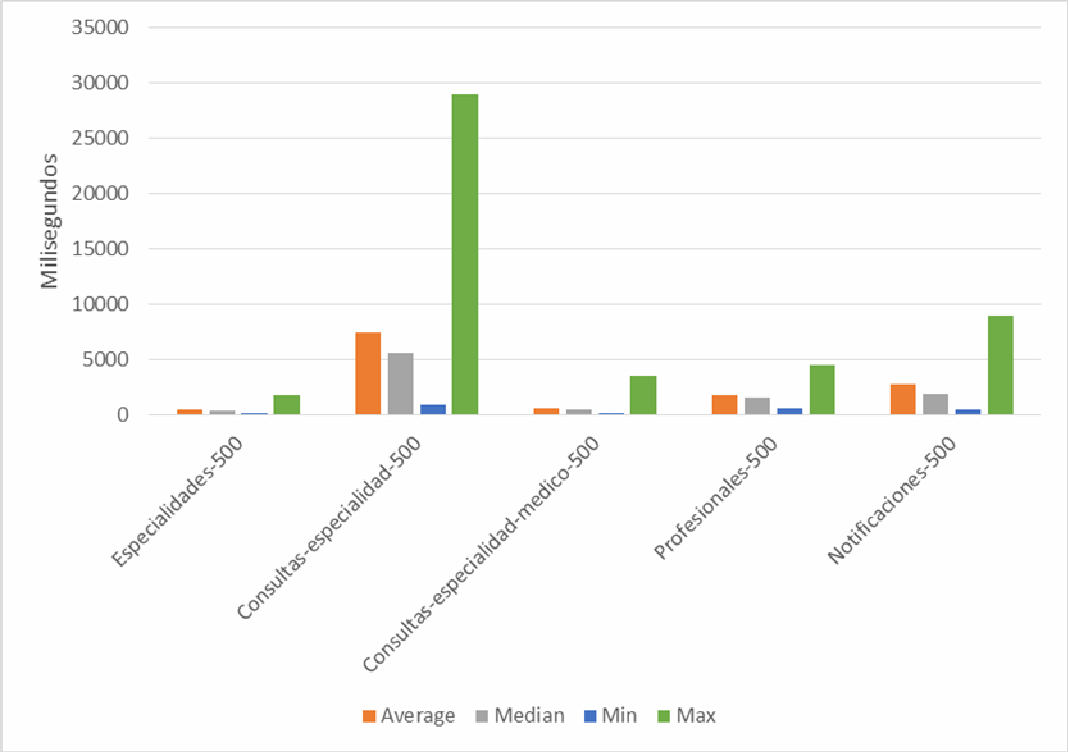


Figura P11 – Contexto Remoto: 1000 Peticiones, Nivel de concurrencia 500.

Nuevamente, el problema principal radica en la prueba #2 donde el tiempo promedio cuando se establecen 500 conexiones concurrentes, obteniendo un promedio de 7433 ms para cada respuesta.

Esto se debe a que para esta prueba se ejecutó la consulta obteniendo los 257 resultados, obteniendo documento extenso (69522 bytes). Como se demostró en el apartado anterior, esto se reduce linealmente si se altera el límite de registros para las consultas.

Los resultados con cache remotamente son superiores a los obtenidos localmente sin cache. Esto refleja que la optimización se debe básicamente a la inclusión de esta estrategia.

Se adjunta una tabla comparativa **Tabla P12**, con los tiempos de respuesta promedio, media, mínimo y máximo con el objetivo de establecer una comparación meramente ilustrativa de la performance que brinda cache a nivel global, incluso mejorando los resultados sobre localhost para la mayor parte de los casos en tiempo promedio que de las variables mencionadas, es la más importante para este tipo de sistema.

Tabla P12 – Tiempos de respuesta promedio, media, mínimo y máximo obtenidos al utilizar cache a nivel global.

Label	Average	Median	Min	Max	Throughput
especialidades-10	223	221	208	711	-2.23
consultas-especialidad-10	-325	-340	84	420	18.48
consultas-especialidad-medico-1	206	207	196	531	0.34
profesionales-10	145	141	198	655	8.33
notificaciones-10	22	11	168	723	15.24
TOTAL	54	142	208	420	7.34
1000 Peticiones, Nivel de concurrencia 10					

Label	Average	Median	Min	Max	Throughput
especialidades-100	123	96	200	475	209.97
consultas-especialidad-100	-4242	-4732	373	3108	34.24
consultas-especialidad-medico-1	16	-12	218	210	210.04
profesionales-100	-175	-247	198	516	133.26
notificaciones-100	-559	-650	188	2282	58.83
TOTAL	-967	-223	218	3108	35.15
1000 Peticiones, Nivel de concurrencia 100					

Label	Average	Median	Min	Max	Throughput
especialidades-500	212	223	205	1272	189.39
consultas-especialidad-500	-17931	-24340	827	-4200	17.28
consultas-especialidad-medico-5	-285	-190	222	1900	57.12
profesionales-500	-2018	-2111	259	-1233	60.03
notificaciones-500	-3941	-5666	434	-856	48.78
TOTAL	-4793	-2118	219	-4200	24.14
1000 Peticiones, Nivel de concurrencia 500					

Los resultados en valor absoluto señalados en rojo, para Promedio, Media, Mínimo y Máximo son la cantidad de milisegundo que se logró mejorar. Para la cantidad de solicitudes, se presenta también en rojo la diferencia en cantidad de resultados por segundo, en el cual el contexto remoto se comporta mejor.

Observar que de ninguna manera se logra mejorar los tiempos mínimos, esto se debe a que en el contexto local, no existe tiempo de establecimiento de conexión, ni factores adversos de red, por lo cual este resultado sería esperado, en cambio el tiempo máximo si puede mejorarse en el contexto remoto.

En la **Tabla P13** se muestra el porcentaje de mejora remoto ante local, donde en los valores en rojo el contexto remoto obtuvo mejor desempeño.

Tabla P13 – Remoto vs Local

Label	Average	Median	Min	Max	Throughput
especialidades-10	-892.00%	-960.87%	-945.45%	-2091.18%	-5.70%
consultas-especialidad-10	50.78%	54.23%	-49.70%	-60.52%	159.32%
consultas-especialidad-medico-10	-528.21%	-575.00%	-560.00%	-1083.67%	0.92%
profesionales-10	-134.26%	-131.78%	-618.75%	-574.56%	29.33%
notificaciones-10	-9.28%	-4.62%	-258.46%	-265.81%	73.25%
TOTAL	-25.71%	-132.71%	-945.45%	-60.52%	61.17%

1000 Peticiones, Nivel de concurrencia 10

Label	Average	Median	Min	Max	Throughput
especialidades-100	-83.11%	-64.00%	-666.67%	-244.85%	440.18%
consultas-especialidad-100	73.88%	79.25%	-376.77%	-49.55%	225.27%
consultas-especialidad-medico-10	-6.27%	4.65%	-1981.82%	-64.42%	451.70%
profesionales-100	29.26%	40.03%	-396.00%	-76.44%	297.46%
notificaciones-100	46.51%	53.90%	-329.82%	-153.67%	146.34%
TOTAL	60.86%	36.14%	-1981.82%	-49.55%	408.72%

1000 Peticiones, Nivel de concurrencia 100

Label	Average	Median	Min	Max	Throughput
especialidades-500	-78.52%	-97.81%	-891.30%	-260.12%	141.87%
consultas-especialidad-500	70.69%	81.38%	-835.35%	12.65%	117.57%
consultas-especialidad-medico-50	31.35%	27.18%	-2466.67%	-117.65%	42.53%
profesionales-500	52.44%	57.30%	-84.09%	21.29%	81.35%
notificaciones-500	58.68%	74.27%	-1085.00%	8.70%	104.69%
TOTAL	64.58%	59.29%	-2433.33%	12.65%	104.50%

1000 Peticiones, Nivel de concurrencia 500

Concurrentemente la mejora promedio fue de 60% y 64% con procesos concurrentes 100 y 500 respectivamente. Apreciar que los cambios generaron un impacto importante, a tal punto que la comparación puede realizarse en lo que aparentemente sería una comparación injusta debida a la mayor cantidad de variables que posee el contexto remoto ante el local.

10.4 Comparación de Sistemas: SAREM vs SAMI

Para la comparación entre sistemas, se buscaron en la órbita pública posibles aplicaciones de comparación, sin embargo los organismos que poseen sistemas de agenda requieren autenticación y por ende ser usuario del organismo.

No fue posible lograr una comparación con sistemas de organismos públicos, sin embargo se logró un mejor resultado gracias a la colaboración de los integrantes del proyecto SAMI, que amablemente cedieron un token de acceso para poder realizar las pruebas contra la API. De esta forma se lograron extraer resultados y en función de los mismos comparar el rendimiento de ambas API's.

Primeramente existió un problema en cuanto a la comparación, los servidores se encontraban geográficamente en posiciones distantes, lo que imposibilita realizar una comparación en el mismo contexto.

Sin embargo se conoce la infraestructura brindada para ambos proyectos, tratándose de instancias EC2 de Amazon Web Services (Free tier).

En la **Figura P12** se muestra la ejecución de traceroute para evaluar si es factible realizar las pruebas con servidores geográficamente distantes. Para ello se ejecuta traceroute con la opción -U ya que los paquetes ICMP son bloqueados por el firewall de Amazon.

Figura P12 – Ejecución de Traceroute para evaluar si es factible realizar las pruebas con servidores geográficamente distantes.

```
leonardo@kraken:~$ traceroute -U samiuy.tk

traceroute to samiuy.tk (54.207.102.151), 30 hops max, 60 byte packets

 1  192.168.1.1 (192.168.1.1)  1.707 ms  1.689 ms  1.681 ms

 2  cor2bras1.antel.net.uy (200.40.161.195)  7.257 ms  7.259 ms  8.404 ms

 3  ibb2cor4-0-2-0-0.antel.net.uy (200.40.161.14)  6.440 ms  8.374 ms ibb2cor3-0-2-0-
0.antel.net.uy (200.40.161.12)  8.378 ms

 4  cbb2tiu1-be200.antel.net.uy (200.40.162.1)  9.922 ms cbb2tia1-be200.antel.net.uy
(200.40.78.5)  9.901 ms cbb2une1-be150.antel.net.uy (200.40.162.5)  10.397 ms

 5  ibe2agu1-be150.antel.net.uy (200.40.78.2)  13.363 ms ibe2uni1-be150.antel.net.uy
(200.40.162.6)  9.889 ms ibe2agu1-be150.antel.net.uy (200.40.78.2)  12.598 ms

 6  ibr2bar2-0-0-0-2.antel.net.uy (200.40.16.154)  23.646 ms  18.871 ms  21.620 ms

 7  176.52.252.58 (176.52.252.58)  21.586 ms xe-1-0-21-0-
```

grtbueba3.net.telefonicaglobalsolutions.com (176.52.252.134) 29.366 ms 84.16.11.194 (84.16.11.194) 23.601 ms

8 213.140.49.22 (213.140.49.22) 46.444 ms 46.445 ms 44.563 ms

9 94.142.97.44 (94.142.97.44) 57.597 ms 5.53.3.82 (5.53.3.82) 50.470 ms 5.53.3.78 (5.53.3.78) 49.678 ms

10 5.53.0.194 (5.53.0.194) 50.453 ms 47.579 ms 5.53.0.198 (5.53.0.198) 49.061 ms

11 177.72.240.141 (177.72.240.141) 48.066 ms 50.479 ms 177.72.240.135 (177.72.240.135) 53.990 ms

12 177.72.240.253 (177.72.240.253) 53.973 ms 177.72.240.153 (177.72.240.153) 50.463 ms 177.72.240.253 (177.72.240.253) 46.609 ms

13 ec2-54-207-102-151.sa-east-1.compute.amazonaws.com (54.207.102.151) 49.100 ms **43.160 ms 44.571 ms**

leonardo@kraken:~$ traceroute -U ec2-54-191-45-10.us-west-2.compute.amazonaws.com

traceroute to ec2-54-191-45-10.us-west-2.compute.amazonaws.com (54.191.45.10), 30 hops max, 60 byte packets

1 192.168.1.1 (192.168.1.1) 1.149 ms 1.913 ms 1.913 ms

2 cor2bras1.antel.net.uy (200.40.161.195) 5.525 ms 6.342 ms 7.046 ms

3 ibb2cor4-0-2-0-0.antel.net.uy (200.40.161.14) 7.050 ms 8.049 ms ibb2cor3-0-2-0-0.antel.net.uy (200.40.161.12) 7.474 ms

4 cbb2agu1-be150.antel.net.uy (200.40.78.1) 9.577 ms cbb2une1-be150.antel.net.uy (200.40.162.5) 10.586 ms 10.591 ms

5 ibe2uni1-be150.antel.net.uy (200.40.162.6) 10.047 ms 10.053 ms ibe2agu1-be200.antel.net.uy (200.40.78.6) 8.750 ms

6 ibr2nap4-0-1-4-0.antel.net.uy (200.40.16.38) 147.883 ms ibr2nap4-0-2-1-0.antel.net.uy (200.40.16.178) 151.717 ms 153.270 ms

7 xe-0-2-0-35.r04.miamfl02.us.bb.gin.ntt.net (128.241.1.225) 160.663 ms 164.708 ms *

8 ae-2-52.ear3.Seattle1.Level3.net (4.69.203.169) 242.856 ms 243.383 ms 243.349 ms

9 4.16.168.34 (4.16.168.34) 240.990 ms atl-bb1-link.telia.net (62.115.117.54) 174.596 ms 4.16.168.34 (4.16.168.34) 241.479 ms

10 * chi-b21-link.telia.net (62.115.116.125) 192.486 ms *

11 * * ae-8.r23.snjsca04.us.bb.gin.ntt.net (129.250.4.154) 238.698 ms

12 * * *

13 * 205.251.232.209 (205.251.232.209) 258.159 ms *

14 205.251.232.61 (205.251.232.61) 248.762 ms 205.251.232.221 (205.251.232.221) 253.888 ms ae-3.amazon.sttlwa01.us.bb.gin.ntt.net (198.104.202.182) 272.949 ms

15 * 205.251.232.94 (205.251.232.94) 234.652 ms *

16 205.251.232.147 (205.251.232.147) 238.626 ms * *

17 54.239.48.181 (54.239.48.181) 238.928 ms * *

18 * * *

19 * * 205.251.232.165 (205.251.232.165) 266.778 ms

20 ec2-54-191-45-10.us-west-2.compute.amazonaws.com (54.191.45.10) **236.393 ms 234.512 ms ***

Aparentemente no es correcto trabajar con servidores geográficamente muy distantes, SAMI se ubica en San Pablo (Brasil) mientras que SAREM en Oregon (EEUU), como la prueba se realizó de Uruguay es evidente que los tiempos de respuestas logrados desde San Pablo son menores.

Observando el caso se decidió crear una instancia en San Pablo para equidistar distancias y realizar una prueba con justicia.

Los casos de prueba se tomaron en función de los llamados más comunes en el sistema por parte de los usuarios.

- Listar Especialidades.
- Listar consultas por Especialidad.
- Listar consultas por Profesional.
- Listar Profesionales.

10.4.1 Resultados

Se detallan los resultados obtenidos para cada sistema y finalmente se procede a la comparación de los tiempos de respuesta de ambos sistemas a través de la invocación de ambas APIs. Se tendrá en cuenta cómo impacta el nivel de concurrencia en el sistema así como algunas medidas específicas (promedio, media, mínimo, máximo) en función de los resultados.

Para la prueba se estableció un conjunto de ejecuciones para llevarla a cabo, para ello se utilizó nuevamente Jmeter[97] con los endpoints para cada caso ajustando 1000 peticiones http con niveles de concurrencia de 10, 100 y 500 procesos.

10.4.1.1 Resultados SAMI

Tabla P14 – SAMI

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-10	1000	54	52	64	45	123	0	146.19883	292.100694
consultas-especialidad-10	1000	196	181	270	100	661	0	40.3991435	1663.24686
consultas-especialidad-medico-10	1000	60	60	69	46	130	0	131.96094	587.38224
profesionales-10	1000	156	142	216	99	700	0	51.7303813	1639.20151
TOTAL	4000	116	103	208	45	700	0	46.9268762	930.406331

1000 Peticiones, Nivel de concurrencia 10

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-100	1000	103	105	137	49	226	0.01	478.697942	956.114057
consultas-especialidad-100	1000	1212	1113	2084	152	4554	0	69.0178756	2841.48791
consultas-especialidad-medico-100	1000	116	101	181	3	721	0.025	410.172272	1801.04241
profesionales-100	1000	843	725	1443	174	4089	0	90.3016074	2861.42354
TOTAL	4000	568	247	1446	3	4554	0.00875	61.3930073	1216.28736

1000 Peticiones, Nivel de concurrencia 100

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-500	1000	283	246	504	0	828	0.023	524.934383	1047.58038
consultas-especialidad-500	1000	7537	6850	13809	0	24531	0.04	38.9468765	1542.03494
consultas-especialidad-medico-500	1000	585	483	1142	57	1906	0	329.05561	1465.24543
profesionales-500	1000	5566	4530	10557	0	41471	0.012	23.8504102	747.155815
TOTAL	4000	3493	1101	9956	0	41471	0.01875	36.5243435	706.458961

1000 Peticiones, Nivel de concurrencia 500

Los resultados se expresan en milisegundos excepto las columnas Error, Throughput y KB/s. Los valores señalados en rojo reflejan errores ocurridos en tiempo de ejecución.

El detalle que se puede observar en la prueba es el porcentaje de peticiones erróneas, que en sí mismo es muy bajo teniendo en cuenta que únicamente ocurren cuando el nivel de concurrencia es alto.

Los casos más complejos aparentemente son consultas por especialidad y consultas por especialidad médico, en cual en todas las pruebas refleja un mayor tiempo de respuesta que los demás.

10.4.1.2 Resultados SAREM

Tabla P15 – SAREM

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-10	1000	59	58	74	44	592	0	134.300295	245.3871
consultas-especialidad-10	1000	343	307	532	120	1306	0	24.230676	1646.57768
profesionales-10	1000	79	60	114	48	660	0	92.0979923	1372.20729
consultas-especialidad-medico-10	1000	53	51	62	43	132	0	139.140114	120.384597
TOTAL	4000	134	61	345	43	1306	0	24.5096537	524.175979

1000 Peticiones, Nivel de concurrencia 10

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-100	1000	69	62	107	44	269	0	558.971492	1021.41614
consultas-especialidad-100	1000	674	555	1216	116	3171	0	108.790252	2903.45205
consultas-especialidad-medico-10	1000	86	85	109	45	219	0	533.049041	461.217038
profesionales-100	1000	366	255	753	59	1894	0	192.938453	2874.678
TOTAL	4000	299	134	774	44	3171	0	90.0353389	996.703445

1000 Peticiones, Nivel de concurrencia 100

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
especialidades-500	1000	154	140	232	49	1125	0	470.588235	860.040901
consultas-especialidad-500	1000	3006	2634	6166	112	11925	0	80.749354	2155.08828
consultas-especialidad-medico-50	1000	247	184	390	46	1135	0	437.062937	378.174255
profesionales-500	1000	1671	1361	3473	96	5951	0	141.183114	2103.55768
TOTAL	4000	1270	349	3760	46	11925	0	70.7901956	783.664488

1000 Peticiones, Nivel de concurrencia 500

Se refleja con los resultados obtenidos remotamente (Brasil) y se puede concluir en primera instancia que las pruebas más costosas en todos los casos son consultas por especialidad y profesionales. El mismo problema que en el sistema SAMI, esto se debe a la cantidad de datos que se obtienen en los endpoints expuestos, basta apreciar los Kb's para notar que la velocidad de transferencia de archivos es significativamente mayor que en los otros casos, y esto es posible siempre que el tamaño sean mayores que los demás.

10.4.1.3 Comparación de Resultados

Tabla P16 – Comparación SAREM vs SAMI

Label	Average	Median	90% Line	Min	Max	Throughput	KB/sec
especialidades-10	-8.47%	-10.34%	-13.51%	2.22%	-79.22%	-8.14%	-15.99%
consultas-especialidad-10	-42.86%	-41.04%	-49.25%	-16.67%	-49.39%	-40.02%	-1.00%
consultas-especialidad-medico-10	11.67%	15.00%	10.14%	6.52%	-1.52%	5.16%	-79.50%
profesionales-10	49.36%	57.75%	47.22%	51.52%	5.71%	43.83%	-16.29%
TOTAL	-13.43%	40.78%	-39.71%	4.44%	-46.40%	-47.77%	-43.66%

1000 Peticiones, Nivel de concurrencia 10

Label	Average	Median	90% Line	Min	Max	Throughput	KB/sec
especialidades-100	33.01%	40.95%	21.90%	10.20%	-15.99%	14.36%	6.39%
consultas-especialidad-100	44.39%	50.13%	41.65%	23.68%	30.37%	36.56%	2.13%
consultas-especialidad-medico-100	25.86%	15.84%	39.78%	-93.33%	69.63%	23.05%	-74.39%
profesionales-100	56.58%	64.83%	47.82%	66.09%	53.68%	53.20%	0.46%
TOTAL	47.36%	45.75%	46.47%	-93.18%	30.37%	31.81%	-18.05%

1000 Peticiones, Nivel de concurrencia 100

Label	Average	Median	90% Line	Min	Max	Throughput	KB/sec
especialidades-500	45.58%	43.09%	53.97%	-100.00%	-26.40%	-10.35%	-17.90%
consultas-especialidad-500	60.12%	61.55%	55.35%	-100.00%	51.39%	51.77%	28.45%
consultas-especialidad-medico-500	57.78%	61.90%	65.85%	19.30%	40.45%	24.71%	-287.45%
profesionales-500	69.98%	69.96%	67.10%	-100.00%	85.65%	83.11%	64.48%
TOTAL	63.64%	68.30%	62.23%	-100.00%	71.24%	48.40%	9.85%

1000 Peticiones, Nivel de concurrencia 500

Se realizaron cálculos sencillos para comparar el rendimiento de las APIs, para ello se efectuó para todos los casos excepto cantidad de respuestas por segundo y KB/s la diferencia entre los tiempos de cada sistema (Tiempo SAREM – Tiempo SAMI / MAXIMO(Tiempo SAREM, Tiempo SAMI).

Se expresa en verde los valores en los cuales SAREM obtuvo mejores tiempos en los tiempos de respuesta, los casos señalados en rojo implican que los tiempos mínimos son 0 (en la práctica imposible de lograr) ver **Tabla P16**.

Se puede extraer una serie de conclusiones a partir de los datos obtenidos, en particular ambos sistemas presentan la capacidad de escalar horizontalmente por lo cual ante demandas concurrentes mayores a 1000 se pueden instanciar nuevos servicios para reducir el tiempo de respuesta promedio en ambos casos.

El peor caso en SAREM y SAMI fue de 3 y 7 segundos de tiempo promedio respectivamente para el caso de consultas por especialidad con nivel de concurrencia 1000.

SAMI presenta un comportamiento más eficiente en el caso donde la concurrencia es de 10 procesos paralelo, para las pruebas listar especialidades y listar consultas por especialidades, SAMI presenta un mejor desempeño global. Logrando de esta forma para el caso de listar consultas por especialidad una mejora de 43 % sobre el sistema SAREM. Sin embargo, para el caso de listar Profesionales, SAREM brinda mejor comportamiento logrando una mejora del 50% en tiempos de respuesta promedio en comparación con SAMI.

En la mayoría de los casos SAMI presenta un mejor comportamiento ante los tiempos máximos, por lo cual la mediana es menor en prácticamente todos los casos.

Sin embargo al aumentar el número de procesos en paralelo, SAREM globalmente tiene un comportamiento más eficiente mejorando los tiempos de respuesta promedio, medio, mínimo y máximo en la mayoría de los casos. Esto puede atribuirse al servidor en sí mismo que debe aceptar las conexiones en paralelo. Consultado a los integrantes de SAMI, se utilizó Apache, en cambio en SAREM se utilizó NGINX como servidor minimalista con técnicas de caching.

Es interesante destacar que a medida que se establecen mayor cantidad de conexiones en paralelo SAREM presenta una ventaja mayor. Otro factor determinante puede ser el manejo que se realiza a nivel de implementación, consultando nuevamente a los integrantes de SAMI sobre la implementación del sistema se relevó que se utiliza Python como lenguaje para el desarrollo del backend, mientras que en SAREM se utiliza NodeJS[101], el cual presenta como bien se mencionó en la sección de implementación, un rendimiento superior cuando se trabaja con eventos concurrentes, ya que el lenguaje es orientado al mismo. No genera de esta forma bloqueos de recursos entre procesos incrementando operaciones de I/O entre otros detalles mencionados.

En orden promedio para niveles de concurrencia 500 procesos, SAREM presenta una mejora promedio de 63% de todos los casos de prueba. Logrando mejores resultados en el caso de listado de profesionales un 70 % más veloz que SAMI.

Se puede concluir que ambos sistemas presentan un rendimiento destacado dada la instancia brindada de forma gratuita en AWS, observar que únicamente con 1GB de memoria RAM y un solo procesador se logran resultados importantes. Es posible que otro factor determinante es el manejador de bases de datos seleccionado. Para SAREM se utiliza PostgreSQL 9.5[94] mientras que en SAMI MySQL [102], si bien puede ser un factor relevante, se dejará de lado esta evaluación puesto que la única forma justa de evaluar cuál es mejor RDBMS es utilizando la misma infraestructura e implementación y permutando únicamente el manejador.

Podemos concluir a partir de los resultados que SAREM presenta mejor rendimiento

cuando el sistema tiende a escalar en llamados concurrentes, sin embargo se debe tener en cuenta todos los componentes involucrados en el sistema, el lenguaje de programación seleccionado, los servicios, la infraestructura, entre otros problemas.

El beneficio de ambos sistema y gracias a ello el buen rendimiento de ambos es la utilización de tecnologías libres, esto es posible gracias al uso de un stack 100% opensource que permite a través de diversas herramientas, lograr resultados de alto rendimiento con hardware limitado.

11. Resultados

Luego de realizadas las pruebas al sistema podemos sacar las siguientes conclusiones:

- El proceso de pruebas dejó en evidencia déficit en cuanto a la realización de una API, el cual obligó a investigar alternativas a los problemas de tiempo de respuesta dado un volumen de datos importante.
- Se lograron mejorar los tiempos de respuestas y la performance global del sistema agregando un componente a la solución. Gracias a Nginx[85] se logra mejorar concurrentemente 60% aproximadamente los tiempos en comparación con un contexto de prueba local.
- El estudio de Nginx[85] así como de Jmeter[97] para el desarrollo de las pruebas resultó un aporte significativo tanto para conocimiento personal como para mejorar el comportamiento de la solución.
- Cuando el volumen de datos crece, la performance de la aplicación se degrada rápidamente, por lo cual gracias al proceso de pruebas, se lograron detectar debilidades importantes en el sistema.
- Se debe manejar estrategias de caching o de particionamiento de datos, reduciendo el tamaño del documento JSON solicitado, ya que el mismo presenta desventajas en la transferencia de los datos, así como en la generación de los mismos.
- Proceso de pruebas de performance, conocidos generalmente como “Pruebas de stress” deben considerarse en etapas tempranas de desarrollo, una debilidad del proyecto fue realizar este tipo de pruebas una vez finalizado el sistema.
- Al comparar SAREM con un proyecto similar se puede apreciar el buen rendimiento global que este posee ante una demanda concurrente importante. Los tiempos de respuesta son aceptables para los tiempos máximos definidos al inicio del proyecto.
- Las pruebas con comparación resultan útiles para detallar el rendimiento promedio del sistema, establecer fortalezas y debilidades del sistema. Se destaca que en ambos casos en la comparación con el Sistema SAMI, se reportaron que los casos de prueba con menor rendimiento coinciden, lo cual refleja que las problemáticas encontradas en ambos sistemas son las mismas, a diferencia que se resuelve con componentes de software distintos.
- El conjunto de pruebas en su totalidad satisface los objetivos marcados, a través de una comparación se pudo establecer que el rendimiento de SAREM es el deseado superando las expectativas planificadas en primer instancia.

12. Conclusiones y Desarrollos futuros

SAREM ha cumplido con la especificación inicial mostrando un alto nivel de rendimiento. Si bien surgieron dificultades en el transcurso del proyecto que dificultaron su culminación, como la pérdida de un integrante, supimos sobrellevar los obstáculos y lograr buenos resultados.

Aunque queda trabajo por hacer, creemos que el sistema es aplicable y adaptable a muchas instituciones del país y la región. Tenemos seguridad de que el sistema va a resolver los problemas de los usuarios a la hora de programar citas médicas, mejorará la gestión de los recursos en los centros de salud, y permitirá a los profesionales médicos tener un contacto más cercano con sus pacientes. Lo que deparara en una mejor calidad de atención.

12.1 Lecciones aprendidas

- SAREM surgió de la interacción de estudiantes de RRMM e Ingeniería en Computación. Esta experiencia nos permitió tomar conciencia sobre las dificultades que se pueden presentar al tener un cliente que no tiene ningún tipo de conocimiento informático, o que sus conocimientos son muy básicos.
- Nos dimos cuenta de lo importante que es la comunicación en un equipo y el trabajo en conjunto. Además somos conscientes de la importancia que tiene poder entender cuáles son las necesidades de los clientes y tener la habilidad de poder transmitirles, y hacerles ver las soluciones posibles junto con sus ventajas y desventajas.
- Aprendimos sobre estándares informáticos médicos, y su aplicación en la práctica, así como también cómo funcionan ciertas partes de los prestadores de servicios y sus funcionarios.
- Este proyecto nos permitió adquirir y mejorar nuestras habilidades de diseño e implementación de aplicaciones web y mobile. También tuvimos la oportunidad de trabajar con tecnologías de vanguardia.

12.2 Trabajo a futuro

Como trabajo a futuro podemos mencionar las siguientes mejoras al sistema:

- Mayor soporte de idiomas. Actualmente la aplicación soporta solo inglés y español.
- Implementar e integrar a SAREM un módulo de análisis de datos, que permita obtener estadísticas de asistencias a consultas, cancelaciones, horas de trabajo de profesionales médicos, inasistencias, etc.
- Aplicación móvil para iOS. Si bien la aplicación móvil se realizó con Ionic [24], lo que nos permite re-utilizar casi el mismo código utilizado para implementar la aplicación Android, la implementación de *Push Notifications* para iOS es diferente. Se debe contemplar esto y realizar pruebas pertinentes para asegurar su correcto funcionamiento.
- Integrar SAREM con un módulo que gestione la historia clínica electrónica de los pacientes. Si se tuviera esta información se podrían generar alertas de manera inteligente para ser enviadas a los pacientes. Estas alertas se generarían mediante la extracción de los datos de la historia clínica del paciente. También facilitaría la labor de los técnicos en RRMM y médicos eliminando el uso del papel.
- Integrar SAREM con un módulo de facturación. Esta integración permitiría que los pacientes pudieran pagar los tickets de las consultas y los estudios médicos mediante SAREM.
- Penalizar usuarios con comportamiento indebido. Implementar funcionalidades en SAREM que permitan limitar acciones a usuarios cuya conducta dentro del sistema sea incorrecta. Por ejemplo no permitirles agendar o cancelar turnos o directamente impedir que inicien sesión.
- Para funcionar adecuadamente en un centro de salud, SAREM necesitará un servicio a consumir que autorice cada paciente y el agendado de horas en línea, basado en la persistencia institucional y sus políticas.

13. Referencias

[1] Agesic, “SALUD.UY,” 2015. [Online]. Available: http://www.agesic.gub.uy/innovaportal/v/4422/19/agesic/que-es.html?idPadre=4425. [Accessed: 21-Feb-2016].

[2] M. and A. MSP, “Acuerdo de Cooperación Técnica e Interinstitucional para el Desarrollo del Programa SALUD.UY.” 2012.

[3] AGESIC, “Investigación sobre la Utilización de las Tecnologías de la Información y Comunicación en el Sector Salud en Uruguay.” .

[4] Jill Bowers, “NueMD Review,” 2014. [Online]. Available: http://medical-billing-services-review.toptenreviews.com/nuemd-review.html. [Accessed: 21-Feb-2016].

[5] NueSoft, “NueMD Overview.” [Online]. Available: http://www.nuemd.com/. [Accessed: 21-Feb-2016].

[6] “NueMD Pricing, Features, Reviews & Comparison of Alternatives.” [Online]. Available: https://www.getapp.com/industries-software/a/nuemd-practice-management/. [Accessed: 13-Mar-2016].

[7] Medical Practice Software, “Medical Practice Software: Benchmark Systems.” [Online]. Available: http://blog.whatasoftware.com/medical-practice-software-benchmark-systems/. [Accessed: 21-Feb-2016].

[8] “Benchmark Systems.” [Online]. Available: https://www.topadvisor.com/practice-management-software/benchmark-systems-profile. [Accessed: 13-Mar-2016].

[9] “CASMU implementa nuevo sistema de Historia Clínica Electrónica,” 25-Feb-2015. [Online]. Available: http://neturuguay.com/casmu-implementa-nuevo-sistema-de-historia-clinica-electronica/. [Accessed: 06-Feb-2016].

[10] “El Hospital Británico se mete de lleno en las prestaciones online,” 27-Dec-2010. [Online]. Available: http://www.infonegocios.biz/nota.asp?nrc=17941&nprt=1. [Accessed: 06-Feb-2016].

[11] “El Hospital Británico renovó su página web con un sistema de reservas online,” 28-Jan-2011. [Online]. Available: http://espectadornegocios.com/core.php?m=amp&nw=MzE1NA==. [Accessed: 06-Feb-2016].

[12] GeneXus Consulting, “ La primera Historia Clínica Digital del Uruguay,” 2012. [Online]. Available: http://www.genexusconsulting.com/comunidad/genexus-consulting-ampliacion-de-noticias/la-primera-historia-clinica-digital-del-uruguay. [Accessed: 21-Feb-2016].

[13] Agesic, “Gobierno Abierto - Sistema de Agenda Electrónica.” [Online]. Available: http://www.agesic.gub.uy/innovaportal/v/2422/1/agesic/sistema_de_agenda_electronica.html?menuderecho=16. [Accessed: 27-Feb-2016].

[14] IBM Corporation, “StrongLoop.” [Online]. Available: https://strongloop.com/. [Accessed: 15-Feb-2016].

[15] Google, “Google Cloud Messaging.” [Online]. Available: https://developers.google.com/cloud-messaging/gcm. [Accessed: 18-Feb-2016].

[16] “AMQP 0-9-1 library and client for Node.JS,” 2016. [Online]. Available: https://www.npmjs.com/package/amqplib. [Accessed: 18-Apr-2016].

[17] “Cloudinary,” 2016. [Online]. Available: http://cloudinary.com/. [Accessed: 18-Apr-2016].

[18] Sysnet, “OpenEMPI.” [Online]. Available: http://www.openempi.org/. [Accessed: 22-Feb-2016].

[19] “Node-gcm.” [Online]. Available: https://github.com/ToothlessGear/node-gcm. [Accessed: 18-Feb-2016].

[20] “AngularJS.” [Online]. Available: https://angularjs.org/. [Accessed: 15-Feb-2016].

[21] Oracle, “JSF,” 2016. [Online]. Available: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html. [Accessed: 18-Apr-2016].

[22] Microsoft, “ASP.NET MVC.” [Online]. Available: http://www.asp.net/mvc. [Accessed: 15-Feb-2015].

[23] “Amazon Web Services.” .

[24] “Ionic.” [Online]. Available: http://ionicframework.com/. [Accessed: 15-Feb-2016].

[25] “Apache Cordova.” [Online]. Available: https://cordova.apache.org/. [Accessed: 15-Feb-2016].

[26] StrongLoop, “LoopBack.” [Online]. Available: https://docs.strongloop.com/display/public/LB/LoopBack. [Accessed: 15-Feb-2016].

[27] ABTO Software, “WHY CHOOSE .NET FOR YOUR PROJECT.” [Online]. Available: http://www.abtosoftware.com/blog/why-choose-net-for-your-project. [Accessed: 15-Feb-2016].

[28] Kevin Purdy, “What is the .Net Framework, and why do I need it?,” 2011. [Online]. Available: http://lifehacker.com/5791578/what-is-the-net-framework-and-why-do-i-need-it. [Accessed: 15-Feb-2016].

[29] “jQuery.” [Online]. Available: https://jquery.com/. [Accessed: 15-Feb-2016].

[30] “Bootstrap.” [Online]. Available: http://getbootstrap.com/. [Accessed: 15-Feb-2016].

[31] Microsoft, “ADO .NET Entity Framework.” [Online]. Available: https://msdn.microsoft.com/library/bb399572(v=vs.100).aspx. [Accessed: 15-Feb-2016].

[32] Wikipedia, “Object-Relational Mapping.” [Online]. Available: https://en.wikipedia.org/wiki/Object-relational_mapping. [Accessed: 15-Feb-2016].

[33] Microsoft, “Microsoft SQL Server.” [Online]. Available: https://www.microsoft.com/es-es/server-cloud/products/sql-server/.

[34] Node.js Foundation, “Node.js.” [Online]. Available: https://nodejs.org/en/. [Accessed: 15-Feb-2016].

[35] Stephen Cleary, “Async Programming : Introduction to Async/Await on ASP.NET,” 2014. [Online]. Available:

https://msdn.microsoft.com/en-us/magazine/dn802603.aspx. [Accessed: 28-Apr-2016].

[36] Thomas L. Marquardt, “ASP.NET Thread Usage on IIS 7.5, IIS 7.0, and IIS 6.0,” 2007. [Online]. Available: https://blogs.msdn.microsoft.com/tmarq/2007/07/20/asp-net-thread-usage-on-iis-7-5-iis-7-0-and-iis-6-0/. [Accessed: 28-Apr-2016].

[37] StrongLoop - An IBM Company, “What Makes Node.js Faster Than Java?,” 2014. [Online]. Available: https://strongloop.com/strongblog/node-js-is-faster-than-java/. [Accessed: 28-Apr-2016].

[38] Jeremy Ashkenas, “CoffeeScript.” [Online]. Available: http://coffeescript.org/. [Accessed: 28-Apr-2016].

[39] Microsoft, “TypeScript - JavaScript that scales.” [Online]. Available: https://www.typescriptlang.org/. [Accessed: 28-Apr-2016].

[40] Ralf S. Engelschall, “ECMAScript 6.” [Online]. Available: http://es6-features.org/#Constants. [Accessed: 28-Apr-2016].

[41] Richard M. Stallman, “Why free software shouldn’t depend on Mono or C#,” 2009. [Online]. Available: https://www.fsf.org/news/dont-depend-on-mono. [Accessed: 28-Apr-2016].

[42] brett Contributions, “Microsoft’s Empty Promise,” 2009. [Online]. Available: https://www.fsf.org/news/2009-07-mscp-mono. [Accessed: 28-Apr-2016].

[43] Stefan Fidanov, “How to Elegantly Solve the Callback Hell of NodeJS and Express with Async.js,” 2015. [Online]. Available: https://www.terlici.com/2015/10/28/solving-node-callback-hell-asyncjs.html. [Accessed: 28-Apr-2016].

[44] Koa developers, “Koa - Next generation of web frameworks for Node.js.” [Online]. Available: http://koajs.com/. [Accessed: 28-Apr-2016].

[45] “Babel is a JavaScript compiler.” [Online]. Available: https://babeljs.io/. [Accessed: 28-Apr-2016].

[46] Node.js.org, “Node.js v6.0.0 Documentation Cluster.” [Online]. Available: https://nodejs.org/api/cluster.html. [Accessed: 29-Apr-2016].

[47] Apache, “ab - Apache HTTP server benchmarking tool.” [Online]. Available: http://httpd.apache.org/docs/2.0/programs/ab.html. [Accessed: 29-Apr-2016].

[48] Mozilla Developer Network, “JavaScript reference.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference. [Accessed: 30-Apr-2016].

[49] Microsoft, “C#.” [Online]. Available: https://msdn.microsoft.com/en-us/library/kx37x362.aspx. [Accessed: 30-Apr-2016].

[50] Microsoft, “.NET Framework 4.6 and 4.5.” [Online]. Available: https://msdn.microsoft.com/en-us/library/w0x726c2(v=vs.110).aspx. [Accessed: 30-Apr-2016].

[51] Pankaj Kumar Jha, “Why to choose AngularJS Javascript Framework for front-end web development?,” 2014. [Online]. Available: https://www.linkedin.com/pulse/20140613173601-45832080-why-to-choose-angularjs-javascript-framework-for-front-end-web-development. [Accessed: 23-Feb-2016].

[52] Pritam Bala, “The advantages and disadvantages of using Ionic framework, compared to native apps,” 2015. [Online]. Available: https://www.linkedin.com/pulse/advantages-disadvantages-using-ionic-framework-compared-pritam-bala. [Accessed: 15-Feb-2016].

[53] “Sass.” [Online]. Available: http://sass-lang.com/. [Accessed: 15-Feb-2016].

[54] Asier Marqués, “Conceptos sobre APIs REST,” 2013. [Online]. Available: http://asiermarques.com/2013/conceptos-sobre-apis-rest/. [Accessed: 15-Feb-2016].

[55] StrongLoop, “Express.js.” [Online]. Available: http://expressjs.com/. [Accessed: 15-Feb-2016].

[56] Alex Gorbatchev, “Comparing Express, Restify, hapi and LoopBack for building RESTful APIs,” 2014. [Online]. Available: https://strongloop.com/strongblog/compare-express-restify-hapi-loopback/. [Accessed: 15-Feb-2016].

[57] “LoopBack.” [Online]. Available: https://github.com/strongloop/loopback. [Accessed: 15-Feb-2016].

[58] “PostgreSQL.” [Online]. Available: http://www.postgresql.org/.

[Accessed: 18-Feb-2016].

[59] Wikipedia, “ACID,” 2015. [Online]. Available: https://es.wikipedia.org/wiki/ACID. [Accessed: 18-Feb-2016].

[60] “PostgreSQL Advantages.” [Online]. Available: http://www.postgresql.org/about/advantages/. [Accessed: 18-Feb-2016].

[61] MongoDB Inc., “MongoDB,” 2016. [Online]. Available: https://www.mongodb.org/. [Accessed: 03-Mar-2016].

[62] “Nodemailer.” [Online]. Available: https://www.npmjs.com/package/nodemailer. [Accessed: 18-Feb-2016].

[63] I. Pivotal Software, “RabbitMQ - Robust messaging for applications.” [Online]. Available: https://www.rabbitmq.com/. [Accessed: 01-May-2016].

[64] AWS, “What is AWS?” [Online]. Available: https://aws.amazon.com/what-is-aws/. [Accessed: 01-Mar-2016].

[65] Maninder Bains, “What is Amazon Cloud, Its Advantages and Why Should You Consider It,” 2014. [Online]. Available: http://www.netsolutionsindia.com/blog/what-is-amazon-cloud-its-advantages-and-why-should-you-consider-it/. [Accessed: 01-Mar-2016].

[66] “Angular translate.” [Online]. Available: https://angular-translate.github.io/. [Accessed: 03-Mar-2015].

[67] GitHub Inc., “GitHub.” [Online]. Available: https://github.com/. [Accessed: 23-Feb-2016].

[68] Software Freedom Conservancy, “Git.” [Online]. Available: https://git-scm.com/. [Accessed: 23-Feb-2016].

[69] “SAE-Introduccion.”

[70] El Senado y la Cámara de Representantes de la República Oriental del Uruguay, “Ley N^{O} 18.335 - PACIENTES Y USUARIOS DE LOS SERVICIOS DE SALUD,” 2008. [Online]. Available: https://datospersonales.gub.uy/wps/wcm/connect/urcdp/8ad60d62-ff5a-4e09-8c7f-

067509674cef/Descargar+Ley+N%C2%B0+18.335.pdf?MOD=AJPERES&CONVERT_TO=url&CACHEID=8ad60d62-ff5a-4e09-8c7f-067509674cef. [Accessed: 01-May-2016].

[71] Dra. Flavia Baladán, “Marco legal de la Historia Clínica Electrónica en Uruguay,” 2011. [Online]. Available: http://www.agesic.gub.uy/innovaportal/file/1652/1/marco_legal_baladan.pdf. [Accessed: 01-May-2016].

[72] El Senado y la Cámara de Representantes de la República Oriental del Uruguay, “Ley N^{o} 18.331 - PROTECCIÓN DE DATOS PERSONALES Y ACCIÓN DE "HABEAS DATA",” 2008. [Online]. Available: https://datospersonales.gub.uy/wps/wcm/connect/urcdp/f085d1b8-0a24-4070-9dad-adc87b7595f2/Descargar+Ley+N%C2%B0+18.331.pdf?MOD=AJPERES&CONVERT_TO=url&CACHEID=f085d1b8-0a24-4070-9dad-adc87b7595f2. [Accessed: 01-May-2016].

[73] PostgreSQL, “Integridad de datos.” [Online]. Available: http://www.postgresql.org/docs/9.5/static/applevel-consistency.html. [Accessed: 01-May-2016].

[74] IBM, “Total security in a PostgreSQL database,” 2009. [Online]. Available: http://www.ibm.com/developerworks/library/os-postgresecurity/. [Accessed: 01-May-2016].

[75] X. S. K. Zhang, *Security and Privacy for Mobile Healthcare Networks, Wireless Networks*. Springer; 1st ed. 2015 edition (November 10, 2015), 2015.

[76] “Concurrencia.” [Online]. Available: https://eva.fing.edu.uy/pluginfile.php/51120/mod_resource/content/2/11_Concurrencia.pdf. [Accessed: 01-May-2016].

[77] JBoss, “Keycloak.” [Online]. Available: http://keycloak.jboss.org/. [Accessed: 01-May-2016].

[78] Bruce Momjian, “Bruce Momjian publications.” [Online]. Available: http://www.enterprisedb.com/postgres-plus-edb-blog/author/bruce-momjian.

[79] PostgreSQL, “The pg_hba.conf File.” [Online]. Available: http://www.postgresql.org/docs/9.5/static/auth-pg-hba-conf.html. [Accessed: 01-May-2016].

[80] PostgreSQL, “Triggers PostgreSQL.” [Online]. Available: http://www.postgresql.org/docs/9.5/static/sql-createtrigger.html. [Accessed: 01-May-2016].

[81] PostgreSQL, “pg_dump.” [Online]. Available: http://www.postgresql.org/docs/9.5/static/app-pgdump.html. [Accessed: 01-May-2016].

[82] OpenSSL Software Foundation, “OpenSSL.” [Online]. Available: https://www.openssl.org/. [Accessed: 01-May-2016].

[83] Elizabeth Mohn, “Advanced Encryption Standard (AES),” 2015. [Online]. Available: http://eds.a.ebscohost.com/eds/detail/detail?sid=7d17300a-3ac1-44df-ae95-b1ac257663ee%40sessionmgr4001&vid=1&hid=4102&bdata=Jmxhbmc9ZXMmc2l0ZT1lZHMtbGl2ZQ%3d%3d#db=ers&AN=87323277. [Accessed: 01-May-2016].

[84] C. Anderson, “Docker,” edselc.2-52.0-84928741645, 2015.

[85] “NGINX,” 2016. [Online]. Available: https://www.nginx.com/. [Accessed: 05-Apr-2016].

[86] Docker, “Docker Swarm.” [Online]. Available: https://docs.docker.com/swarm/. [Accessed: 01-May-2016].

[87] “Docker.” [Online]. Available: https://www.docker.com/. [Accessed: 18-Feb-2016].

[88] Docker, “DockerHub.” [Online]. Available: https://hub.docker.com/explore/. [Accessed: 01-May-2016].

[89] Docker, “Dockerfile.” [Online]. Available: https://docs.docker.com/engine/reference/builder/. [Accessed: 01-May-2016].

[90] SOURCEFORGE.NET, “GitStats - git history statistics generator.” [Online]. Available: http://gitstats.sourceforge.net/. [Accessed: 06-Mar-2016].

[91] AAMC, “Medical Specialties,” 2016. [Online]. Available: https://www.aamc.org/cim/specialty/list/.

[92] “Chance - Utility library to generate anything random,” 2016.

[Online]. Available: http://chancejs.com/. [Accessed: 05-Apr-2016].

[93] “Amazon EC2 Instance Types,” 2016. [Online]. Available: https://aws.amazon.com/ec2/instance-types/. [Accessed: 05-Apr-2016].

[94] “PostgreSQL,” 2016. [Online]. Available: http://www.postgresql.org/docs/9.5/static/app-psql.html. [Accessed: 05-Apr-2016].

[95] Keymetrics, “PM2 - Advanced, production process manager for Node.js,” 2016. [Online]. Available: http://pm2.keymetrics.io/. [Accessed: 05-Apr-2016].

[96] “AMAZON Beanstalk,” 2016. [Online]. Available: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html. [Accessed: 05-Apr-2016].

[97] “Apache JMeter,” 2016. [Online]. Available: http://jmeter.apache.org/. [Accessed: 05-Apr-2016].

[98] “Round-trip delay time,” 2014. [Online]. Available: https://es.wikipedia.org/wiki/Round-trip_delay_time. [Accessed: 05-Apr-2016].

[99] “Nginx Caching,” 2014. [Online]. Available: https://serversforhackers.com/nginx-caching/. [Accessed: 07-Apr-2016].

[100] “Nginx caching guide.” [Online]. Available: https://www.nginx.com/blog/nginx-caching-guide/. [Accessed: 07-Apr-2016].

[101] Wikipedia, “Node.js,” 2016. [Online]. Available: https://es.wikipedia.org/wiki/Node.js. [Accessed: 10-Feb-2016].

[102] “MySQL.” [Online]. Available: https://www.mysql.com/. [Accessed: 18-Feb-2016].

14. Anexo

14.1 Poster Ingeniería deMuestra 2015

SAREM: Sistema de Alertas y Registros Médicos

Valentina Da Silva, Leonardo Clavijo, Victoria Aldaz, Silvana Arrambide, Fernanda Rodríguez, Ing. Lucía Grundel, Ing. F. Simini

InCo – NIB – UIBLH – H.Clínicas

Ingeniería de Muestra, 22 - 24 de octubre 2015.

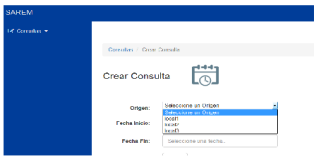
Características generales

Usuarios

- Agendado de consultas en tiempo real vía web.
- Notificaciones dinámicas a los usuarios por medio de Twitter o correo electrónico sobre recordatorios de consulta, promoción de lista de espera y eventos específicos prestadores de Salud.
- Digitalización parte diario.
- Asignación Médico Referencia.
- Usuarios acceden a listas de espera.

Profesionales y Administradores

- Administración de prestadores de salud en el sistema.
- Estrategias de conectividad con servicios externos OpenEMPI.
- Exportación de datos a sistemas externos.

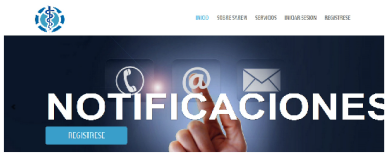


Usabilidad

El impacto global de un sistema en un entorno heterogéneo como en el área de la Salud, depende en gran medida de la experiencia de usuario final. Ídem para los profesionales orientados a Registros Médicos. La simplificación, impacta en reducción del tiempo de trabajo y optimización de recursos materiales, así como gestión de información sensible.

Alta disponibilidad

El sistema permite escalar rápidamente, brindando la posibilidad de desplegar diferentes componentes del sistema utilizando tecnologías PaaS, gratuitas o pagas. Dependiendo de la demanda de los usuarios se ofrece mecanismos de alta disponibilidad para satisfacerla. Es crucial mantener replicados repositorios de datos de uso intensivo, por ello el sistema brinda estrategias de recuperación y alta disponibilidad.



Integración SaludUY

Arquitectura: Combinación de tecnologías privativas y opensource. Aplicación web desplegada en Windows Azure, permitiendo escalabilidad vertical y horizontal a bajo costos. La baja cohesión de integración entre sistemas orientados al área de la Salud, motivaron la elección de una arquitectura Multitenant permitiendo la separación de información sensible entre prestadores de Salud.
Integración con **OpenEMPI** para el registro unificado de pacientes, con despliegue en la nube utilizando servicios PaaS **Openshift**.

Fundación Julio Ricaldoni
INGENIERÍA EN EL URUGUAY

FACULTAD DE INGENIERÍA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

14.2 Paper

SAREM Medical Appointments Management System to Optimize Patient Wellness and Uses of Clinical Resources

Leonardo Clavijo, Valentina Da Silva, Victoria Aldaz, María Fernanda Rodríguez, Silvana Arrambide, Lucia Grundel, Franco Simini

NIB, Facultad de Medicina e Ingeniería, Universidad de la República

Abstract This project stems from the need for a tool to provide a solution to the chaos in the Uruguayan health system, where patients have trouble to schedule medical appointments due to unavailability of slots, and where institutions waste resources because many of these patients do not attend their appointments. SAREM is a computer system that allows proper user of medical resources, a closer contact between the health professional and the patient, and better optimization of time for both health centers officials and their users; friendly for both desktop and mobile web browsers, in addition to an Android application with similar functionality. The system is implemented with open source technologies based on a Node.js Backend and Cloud Services (Amazon Web Services), allowing low costs, high reliability and easy installation on any existing platform. SAREM resulted in a solid and well tested product, with high acceptance and good feedback after being shown at Ingeniería deMuestra 2015 event to over 200 users. Although work remains to be done, we believe that the system is applicable and adaptable to many institutions in the country and the region, and we think it will solve user's problems at the time of scheduling their appointments.

I. INTRODUCTION

SAREM was conceived as a result of a joint analysis between the Medical and Engineering schools of UdelaR, in which the students from the Medical Records career posed a problem to be solved by the Engineering students. That problem was the difficulty for healthcare providers' staff as well as patients to schedule medical appointments.

SAREM emerged with the aim of optimizing the interaction of patients with health centers, and improve management of medical resources.

Since 2008, the Uruguayan health system has evolved with the implementation of the Integrated National Health System (SNIS). The system gives access to workers and retired to healthcare through the National Health Insurance (SNS). Also, from 2010 onwards their spouse and children are also included.

This have increased the number of people attending to Healthcare Providers. That, in turn, generate a variety of problems including finding slots for scheduling medical appointments, particularly for specialists in high demand. Currently, people line up early in the morning to get an appointment, generating dissatisfaction. Especially when there are no slots available or when staff is on a strike. Another source of problem is that patients do not assist to the consultation, so slots are actually available but patients who need them are not aware of that.

In order to improve the situation, the government agency AGESIC designed a plan for the computerization of the Uruguayan Healthcare System called salud.uy [1]. This included the adoption of the Electronic National Clinic History (HCEN), which has led to increase the adoption of information systems in Healthcare Providers.[2]

Nonetheless, the current state of computerization is mostly at an intermediate stage. Most of them continue their largely paper-based systems only storing a little of it to electronic means. [3]

14.3 Manual de Usuario Página Web

Para ingresar a la página se debe ir a la url: http://ec2-52-67-47-2.sa-east-1.compute.amazonaws.com/. Donde aparecerá una página como se muestra en la **Figura 1.**

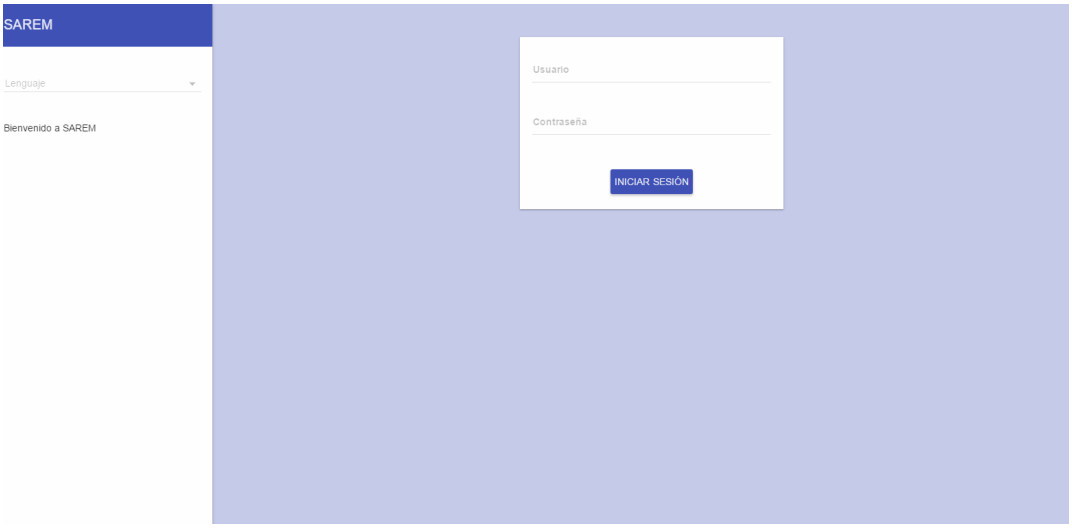


Figura 1 – Inicio Página Web

A continuación se describirán los casos de uso más importantes del sistema, con el fin de que su funcionamiento sea entendido correctamente.

14.3.1 Iniciar Sesión

Para iniciar sesión se debe ingresar el nombre de usuario y la contraseña en los campos que se muestran en la **Figura 2.** Luego se procede a seleccionar el botón “Iniciar Sesión”.

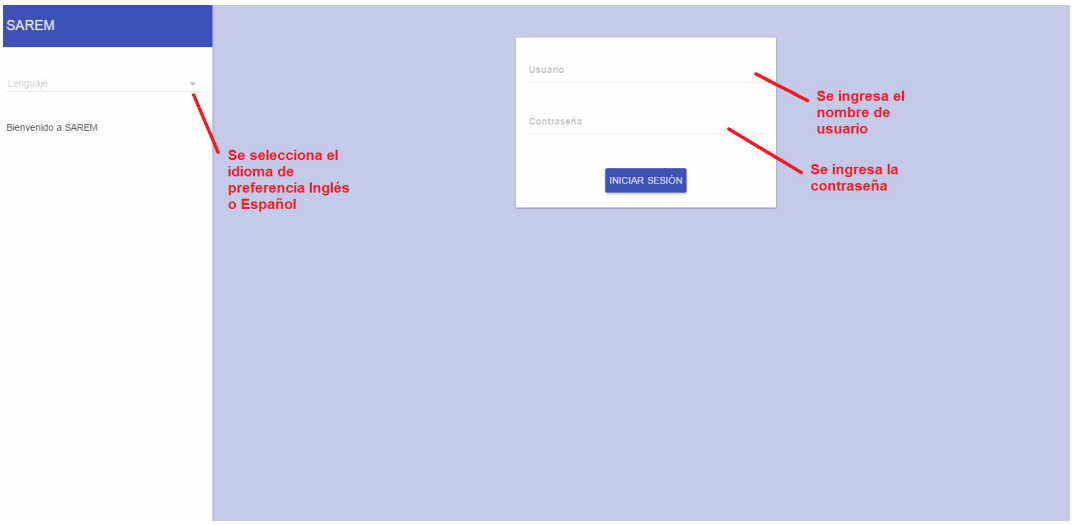


Figura 2 – Iniciar Sesión

Si el usuario y contraseña son correctos, se pasará a una página como la que se muestra en la **Figura 11** si el usuario es de tipo Administrador; si el usuario es de tipo Paciente se mostrara una página como la de la **Figura 12**, y si el usuario es de tipo Profesional se mostrar una página como la de la **Figura 13**.

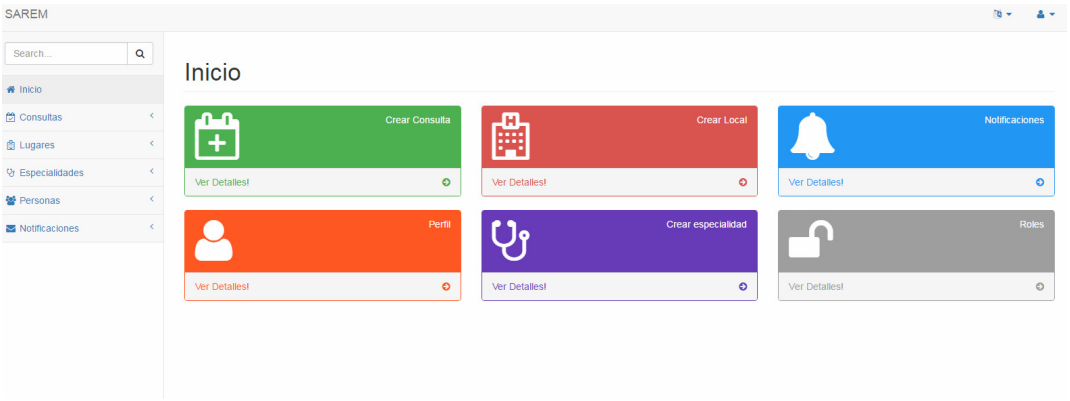


Figura 3 – Página de Inicio usuario Administrador

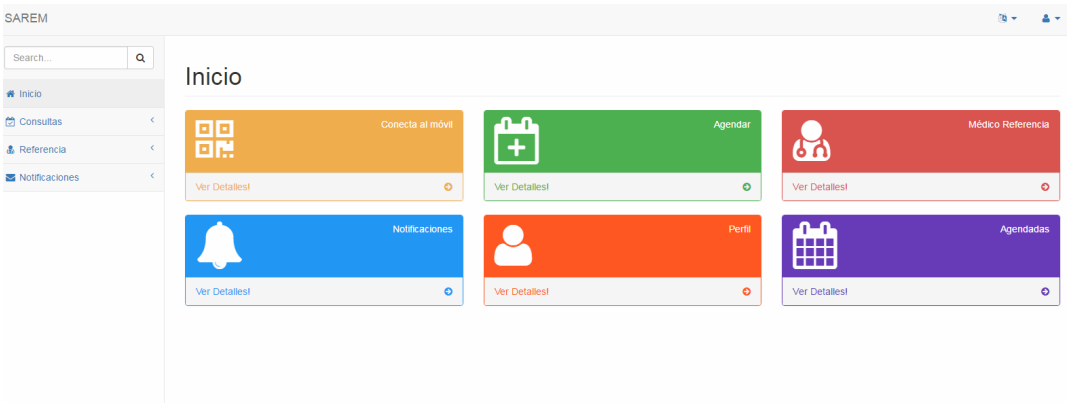


Figura 4 – Página de Inicio usuario Paciente

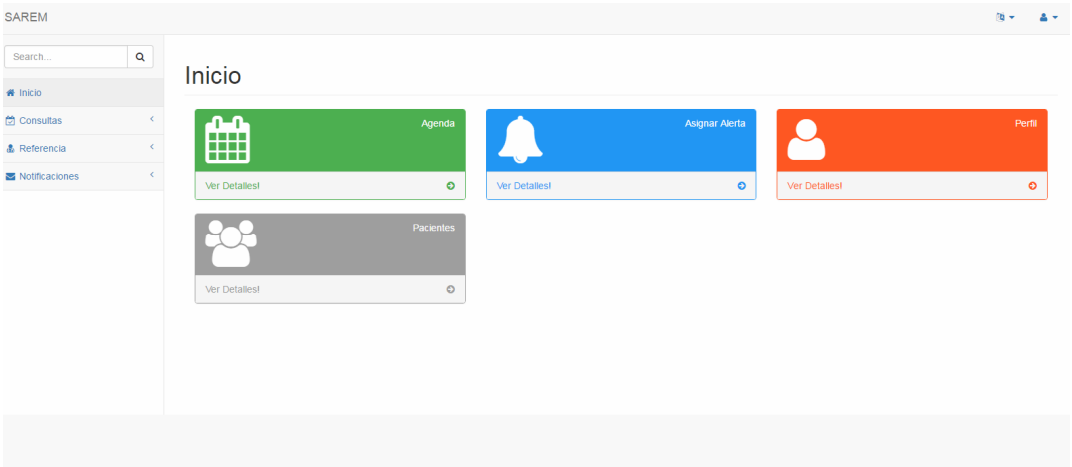


Figura 5 – Página de Inicio usuario Profesional

14.3.2 Caso de Uso Crear Consulta

Para Crear una consulta se debe seleccionar la opción “Crear” dentro de las opciones disponibles para las consultas del menú izquierdo, como muestra la **Figura 6**.

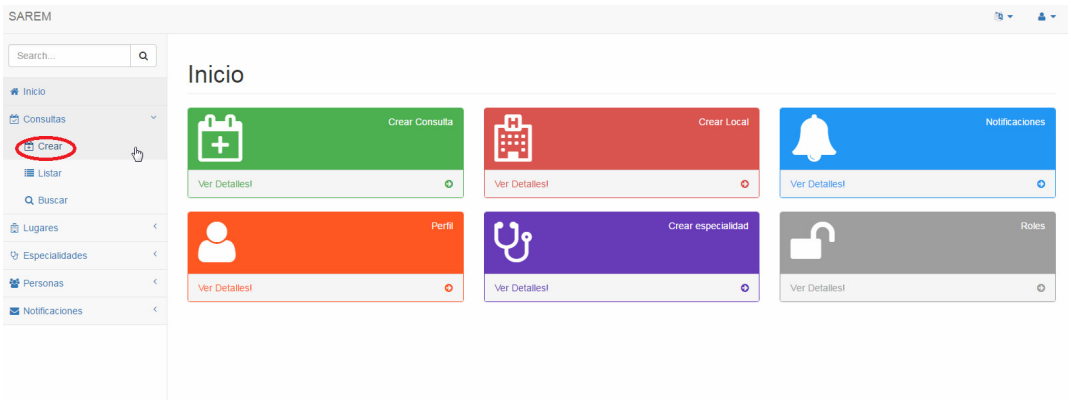


Figura 6 – Crear Consulta

Luego de seleccionar esta opción se visualizara en pantalla una imagen como la de la **Figura 7**. Donde se muestran los campos necesarios para crear la consulta. Entre ellos está el campo Fecha, en este campo se debe ingresar la fecha para la cual se creara la consulta. El campo Hora inicio indica a qué hora empezara la consulta. El campo Minutos turno indica cuantos minutos dura cada turno de la consulta. El campo Pacientes en consulta indica cuantos pacientes serán atendidos en la consulta y el campo Pacientes en lista de espera indica la cantidad de pacientes que podrán ingresar a la lista de espera. Por otra parte están los combos con opciones desplegables Especialidad, Local y Medico. Estos combos muestran las opciones disponibles que existen para crear la consulta especificando especialidad, local (lugar físico donde se realizara la consulta), y medico (profesional de la salud a cargo de la consulta).

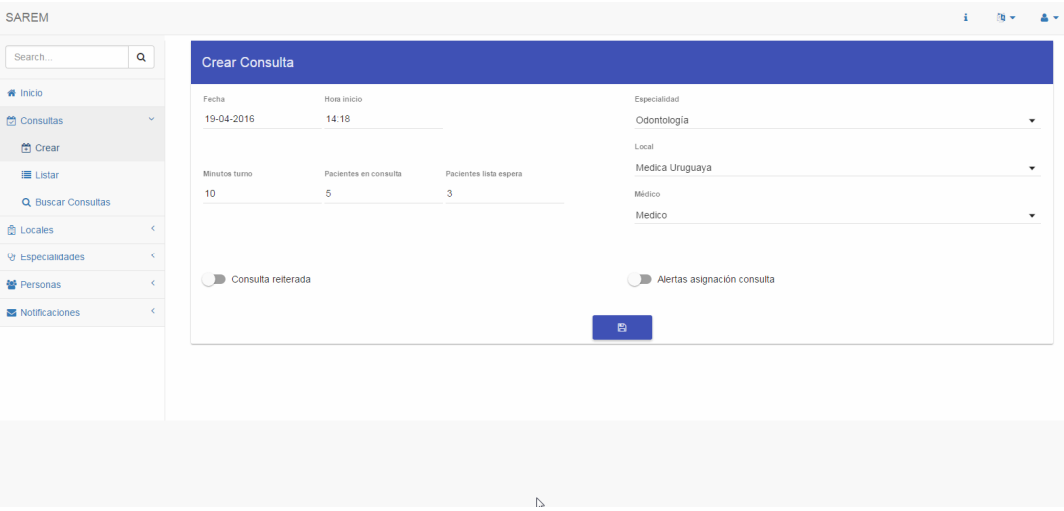


Figura 7 – Campos a ingresar para Crear Consulta

También se puede ver en la **Figura 7,** que existen dos opciones *Consulta reiterada* y *Alertas asignación consulta*. Si se activa la opción *Consulta reiterada*, se muestra en pantalla dos campos más *Repetir cada (días)* y *Hasta* como muestra la **Figura 8**. Esto permite que la consulta se cree también cada n días siendo n el número que se ingresa en el campo *Repetir cada (días)*. El campo *Hasta* indica hasta que fecha se crea la consulta, y actúa como interruptor de la frecuencia de creación de consultas.

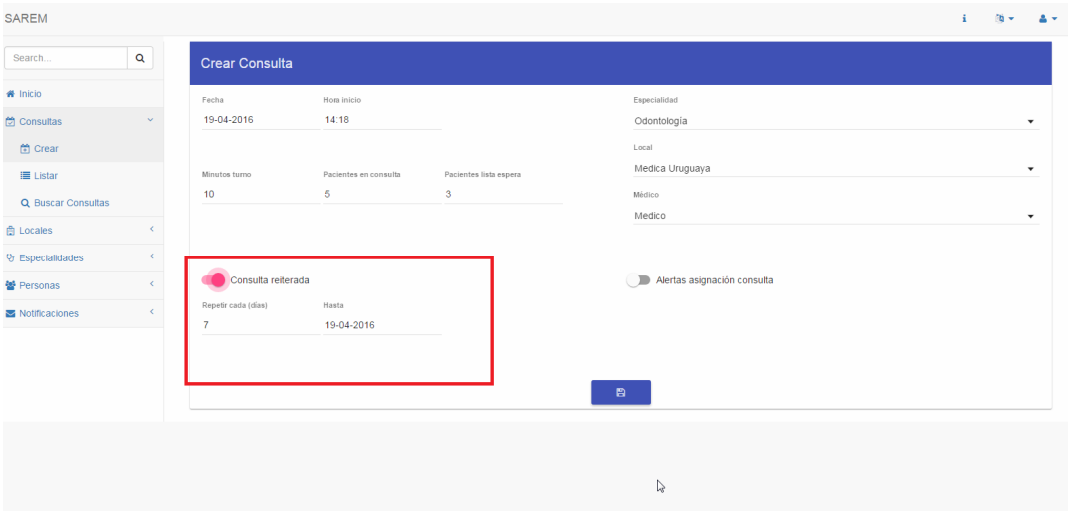


Figura 8 – Campos a ingresar para Crear Consulta, opción Consulta reiterada.

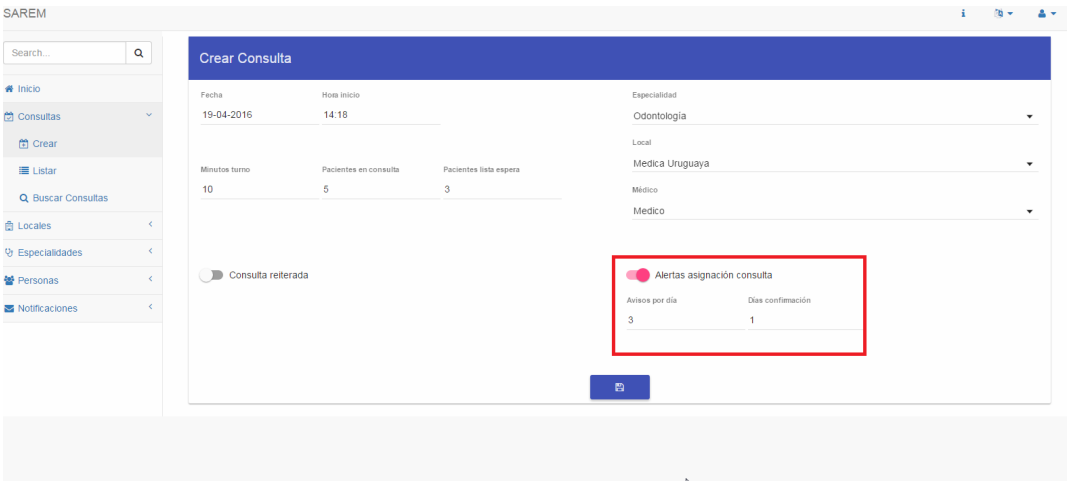


Figura 9 – Campos a ingresar para Crear Consulta, opción Alertas asignación consulta.

Si se activa la opción *Alertas asignación consulta,* se muestran en pantalla dos campos *Avisos por día* y *Días de espera de la confirmación* ver **Figura 9**. . En el campo *Avisos por día* se ingresa un número entero que indica la cantidad de notificaciones que le deben llegar a la persona por día. En *Días confirmación* también se ingresa un número entero que indica la cantidad de días antes de que se realice la consulta en que deben empezar a enviarse las notificaciones.

Luego de que se ingresan todos los datos necesarios se procede a seleccionar el botón guardar como muestra la **Figura 10**.

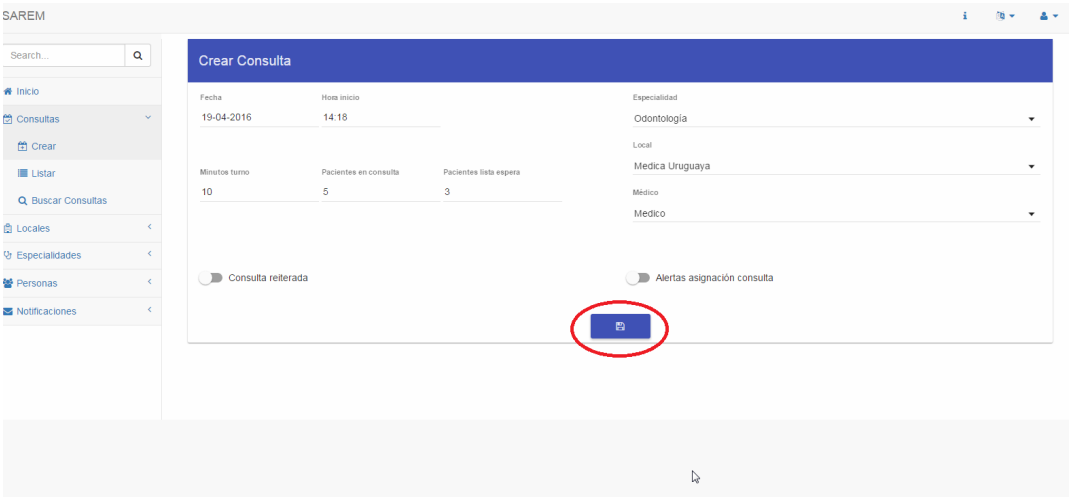


Figura 10 – Guardar Consulta.

14.3.3 Caso de Uso Listar Consultas

Para ver todas las consultas creadas en el sistema que aún están vigentes se debe seleccionar la opción del menú izquierdo Listar, luego de seleccionada esta opción se mostraran en una grilla todas las consultas creadas vigentes como lo muestra la **Figura 11**. Cada consulta que se muestra en la grilla tiene dos botones, uno para ver más información sobre la consulta y otro para eliminarla.

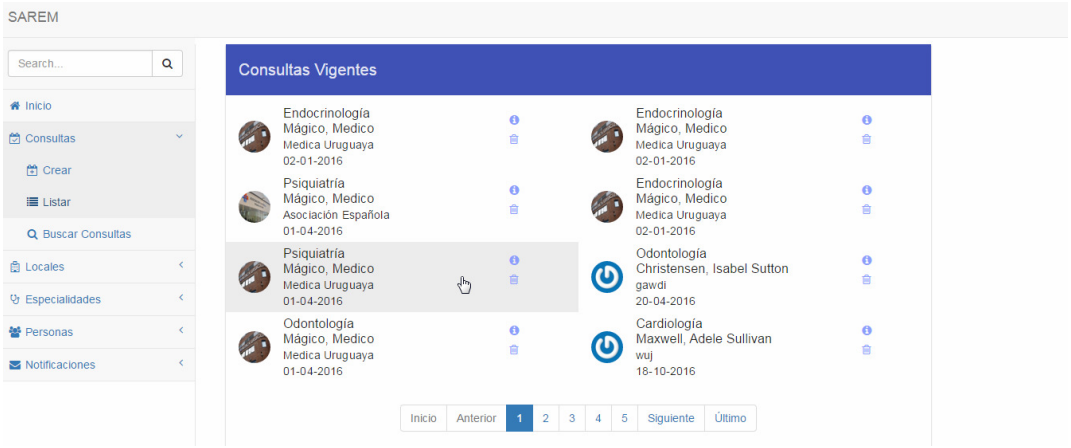


Figura 11 – Listar Consultas.

14.3.4 Caso de Uso Buscar Consulta

Para buscar una consulta para agendarse a un turno de la misma por ejemplo o simplemente para ver más información en detalle sobre una determinada consulta se debe seleccionar la opción del menú izquierdo “Buscar Consultas” como lo indica la **Figura 12**.

Una vez seleccionado el botón “Buscar Consutas”, se mostrará en pantalla un campo de texto seguido de dos botones. Si se ingresa un valor en el campo de texto y luego se selecciona el botón con la lupa, el sistema filtrara todas las consultas que posean el valor ingresado en alguno de sus campos. Si se quiere hacer una búsqueda más específica se puede seleccionar el segundo botón al lado de la lupa. Si se selecciona dicho botón se mostrar en pantalla opciones para seleccionar filtros de búsqueda por Especialidad, Médico, y Local.

La **Figura 12** muestra una imagen que corresponde a la activación de todos los filtros específicos.

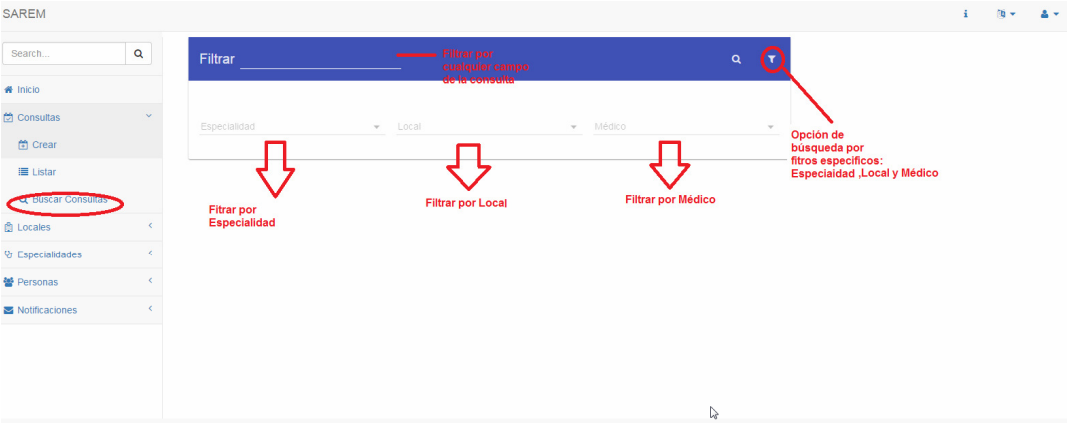


Figura 12 – Filtros para Búsqueda de Consultas.

14.3.5 Caso de Uso Agendar Turno en Consulta

Para agendar un turno en una consulta lo primero que se debe hacer es seleccionar la opción “Agendar” del menú izquierdo de la pantalla como se muestra en la **Figura 13**. Una vez seleccionada esta opción, se deberá buscar la consulta en la que se desea agendar un turno. Para realizar esto tendrá la posibilidad de buscar por cualquier campo de la consulta o utilizar filtros más específicos por Especialidad, Médico y Local. Si se activan estos filtros se mostrara en pantalla lo que aparece en la **Figura 13**. Una vez que se seleccionen los filtros deseados y se seleccione el botón con la imagen de la lupa para buscar, se mostrara en pantalla una grilla con las consultas que cumplen con los datos indicados en los filtros. En caso de que ninguna consulta cumpla con los criterios especificados, se le avisara de que no existen consultas que cumplan los requerimientos.

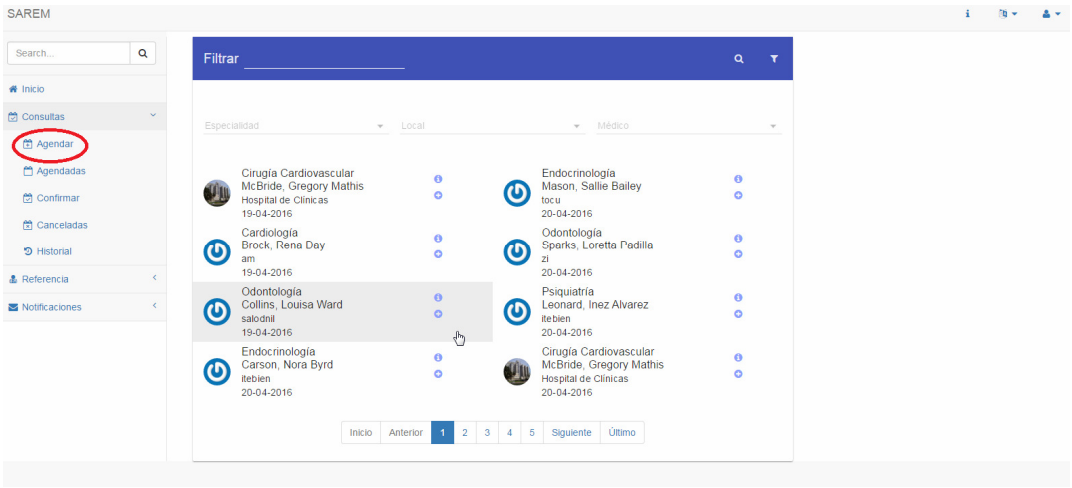


Figura 13 – Opciones de Búsqueda de Consultas.

En caso de existir consultas con turnos libres que satisfagan los filtros se mostrará en pantalla una grilla con las consultas como muestra la **Figura 13**. Se selecciona el icono con el símbolo de (+) para poder ver los turnos disponibles de la consulta.

Se procede a seleccionar un turno de los disponibles y luego se presiona el botón “Agendar” para completar el proceso de agendado. En caso de que se desee ver información más detallada sobre la consulta se puede apretar el botón con el icono de la (i) como lo muestra la **Figura 14**.

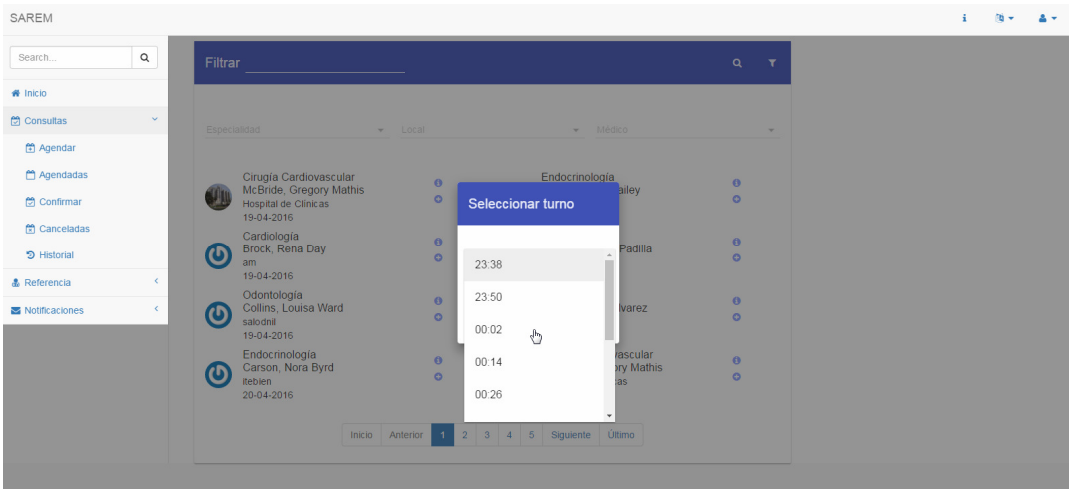


Figura 14 – Seleccionar Turno en Consulta.

14.3.6 Caso de Uso Ingresar a Lista de Espera de una Consulta

En caso de que se desee agendar un turno en una consulta que ya posee todos sus turnos ocupados, el sistema le dará la oportunidad de ingresar a su lista de espera (si esta posee lugares libres).

Como se muestra en la **Figura 15** aparecerá un icono con la imagen de un reloj que si se lo selecciona ingresara al usuario en la lista de espera de la consulta.

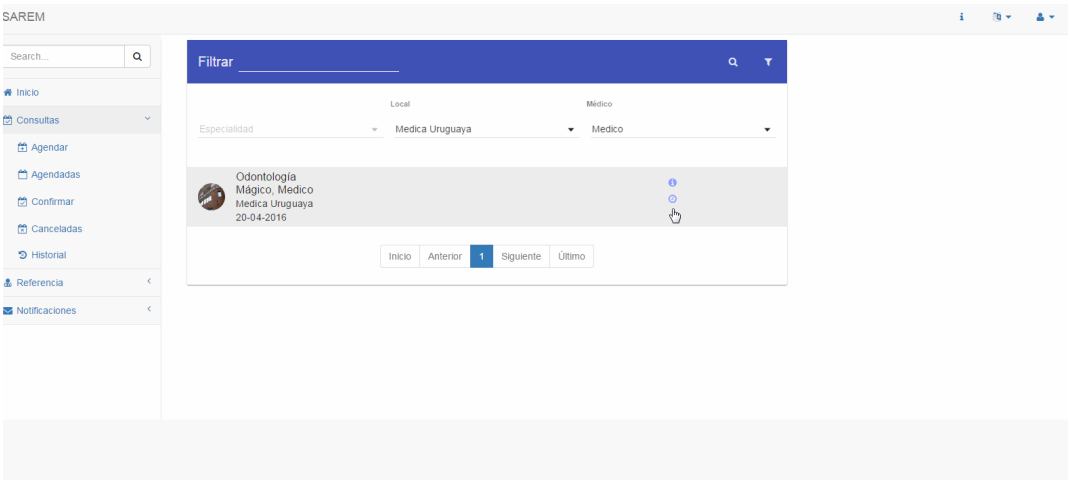


Figura 15 – Ingresar a la Lista de Espera de una Consulta.

14.3.7 Casos de Uso Ver Consultas Agendadas, Confirmadas, Canceladas e Historial

En el menú de la izquierda como muestra la **Figura 16** se pueden ver las opciones para ver todas las consultas agendadas, confirmadas (este caso aplica para cuando un determinado usuario está en la lista de espera de una consulta, se libera un turno, y se le avisa al usuario que existe un turno libre en la consulta y este confirma que desea tomar el turno), canceladas y el historial. Si se selecciona la opción consultas agendadas se le mostrara una grilla similar a la de la **Figura 11,** donde cada consulta de la grilla tendrá un icono en forma de cruz para cancelar la consulta y otro para ver más información sobre la consulta.

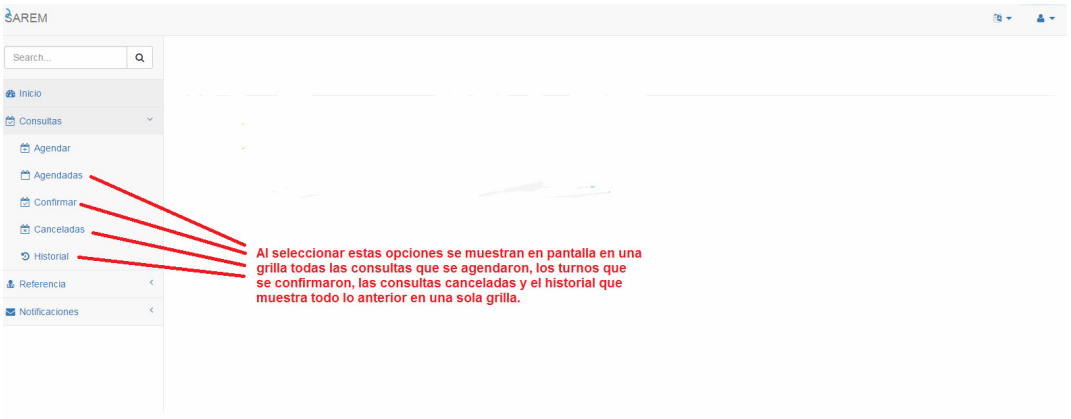


Figura 16 – Opciones para ver Consultas Agendadas, Confirmadas, Canceladas e Historial.

14.3.8 Casos de Uso Seleccionar Médico de Referencia y Cancelar Referencia

Para seleccionar un médico de referencia en caso de no tener uno, se puede seleccionar la opción de Médico de Referencia en el menú izquierdo como muestra la **Figura 17**. En el campo Personas se puede ingresar un valor que identifique al médico con el que desea referenciarse y luego presionar el botón con el icono de la lupa para facilitar la búsqueda.

Una vez obtenidos los resultados, se le mostrara una grilla con los médicos encontrados. Cada médico de la grilla tendrá dos iconos: un icono con la imagen de un doctor para seleccionar al médico y enviar la solicitud de referencia, y otro icono para ver información más detallada sobre el médico.

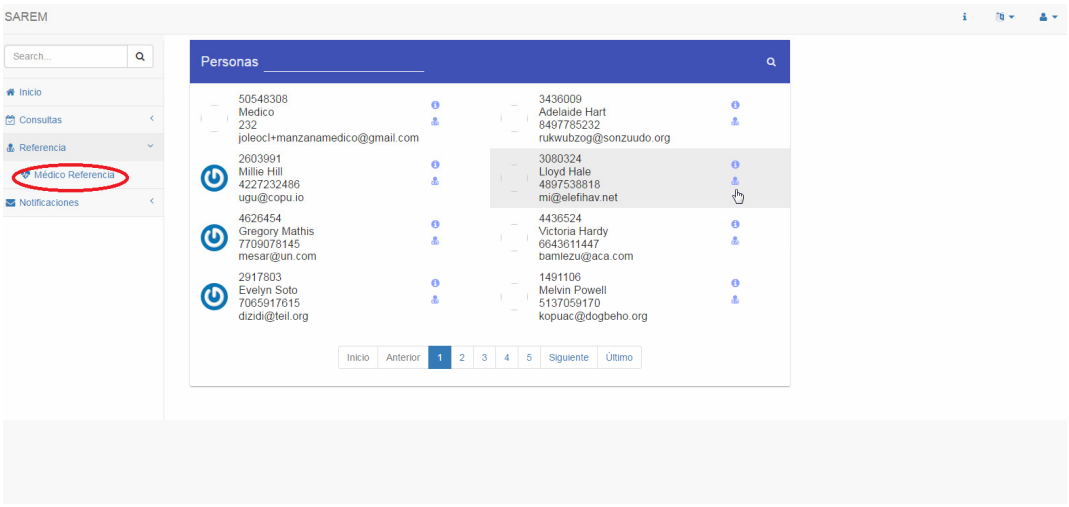


Figura 17 – Seleccionar Médico de Referencia.

En caso de que el usuario ya posea un Médico de Referencia se le mostrar una página con la información de su médico y un botón para eliminar la referencia como muestra la **Figura 18**.

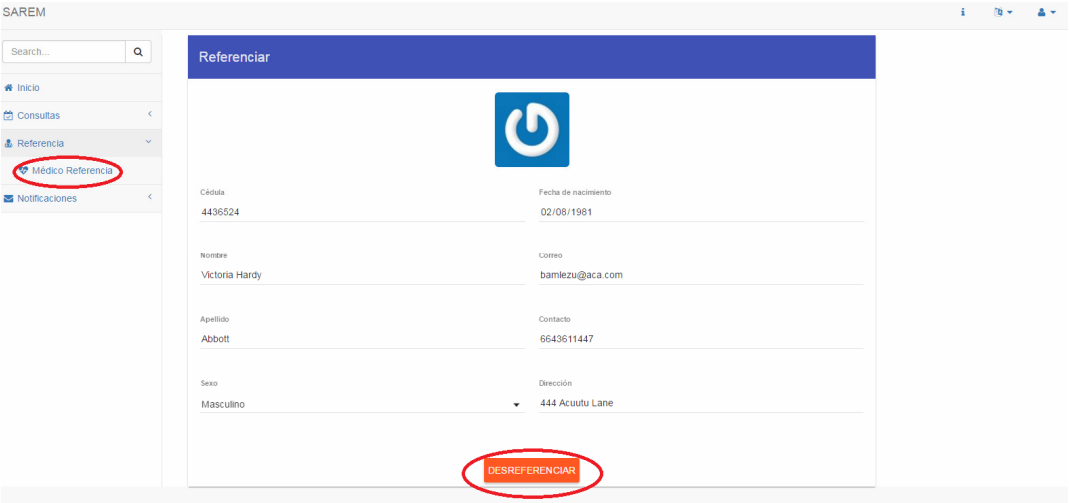


Figura 18 – Información Médico de Referencia.

14.3.9 Casos de Uso Aceptar y Denegar Solicitud de Referencia

Para aceptar o denegar una solicitud de referencia, un usuario de tipo Profesional debe seleccionar la opción del menú izquierdo de la pantalla Referencia > Pacientes, luego deshabilitar la opción *Mostrar confirmados* como lo indica la **Figura 19**. Una vez realizadas estas acciones, vera en pantalla una grilla con todos los pacientes que solicitaron referenciarse con él/ella, y tendrá la posibilidad de aceptar la referencia o denegarla.

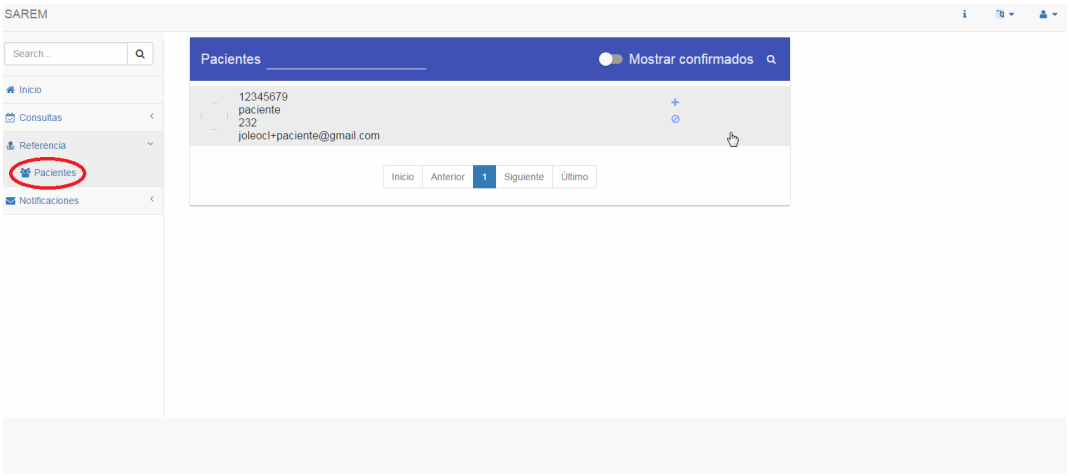


Figura 19 – Aceptar/Denegar Solicitud de Referencia.

14.3.10 Caso de Uso Crear Notificación

Para poder crear una Notificación se debe seleccionar la opción Notificaciones > Crear en el menú de la izquierda de la pantalla como lo muestra la **Figura 20**. Una vez seleccionada esta opción se le mostrará en pantalla un formulario con campos a completar para crear la Notificación. Entre ellos se encuentra el campo Título, Mensaje, la lista desplegable Sexo con opciones (Masculino, Femenino, Todos), la lista desplegable Rango de edades con opciones (Todos, Adolescente, Adulto, Adulto Mayor), el campo Frecuencia, el radio Difusión instantánea.

También se puede ver en pantalla un radio con nombre Móvil. Si activa este radio se muestra en pantalla una imagen correspondiente a la **Figura 21**. Esta opción permite personalizar la notificación que le llegara al usuario en su teléfono móvil, pudiéndose ingresar html con imágenes y estilos. El botón verde es para guardar la notificación.

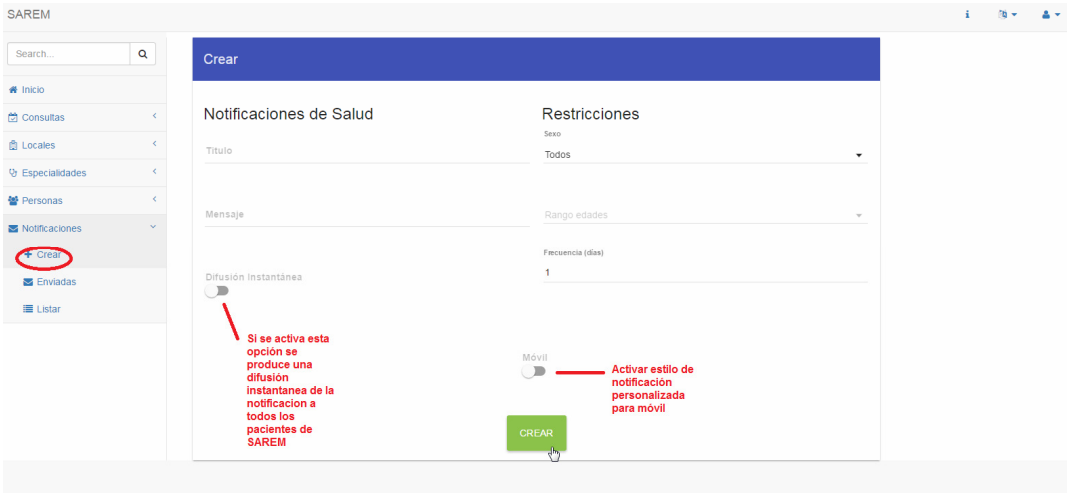


Figura 20 – Crear Alerta.

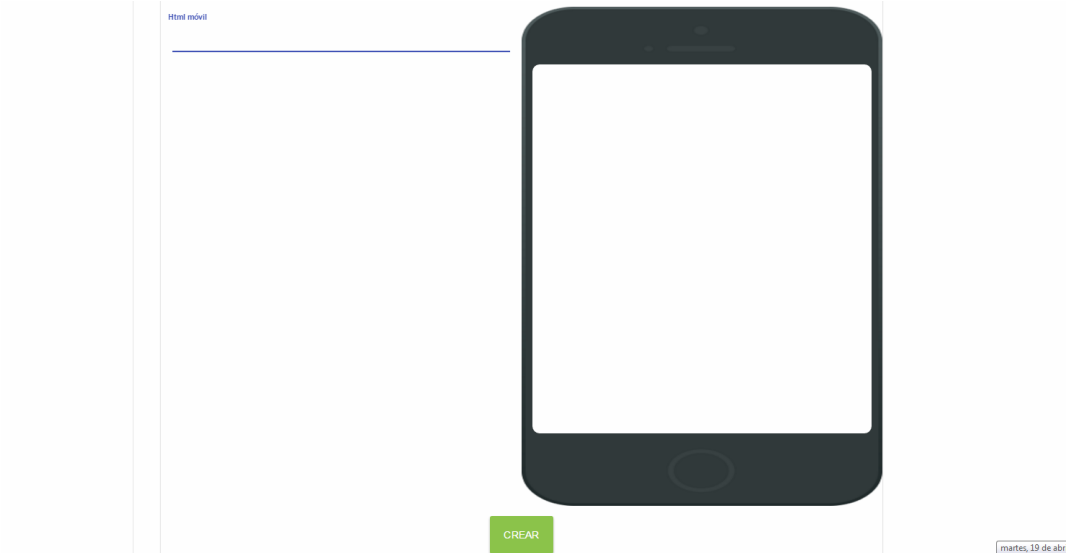


Figura 21 – Notificación Personalizada para móvil.

14.3.11 Caso de Uso Asignar Notificaciones a Pacientes

Para poder asignar notificaciones a pacientes se debe seleccionar la opción Notificaciones > Asignar en el menú izquierdo de la pantalla y luego activar el radio Mostrar confirmados, como se muestra en la **Figura 22.** En la imagen se puede ver una grilla con los pacientes que están referenciados con el medico en cuestión. Para cada paciente de la grilla existe un botón con símbolo de mensaje, que si se lo selecciona se le muestra en pantalla al usuario un modal con las notificaciones que le puede asignar al paciente de acuerdo a su sexo y rango etario como muestra la **Figura 23**.

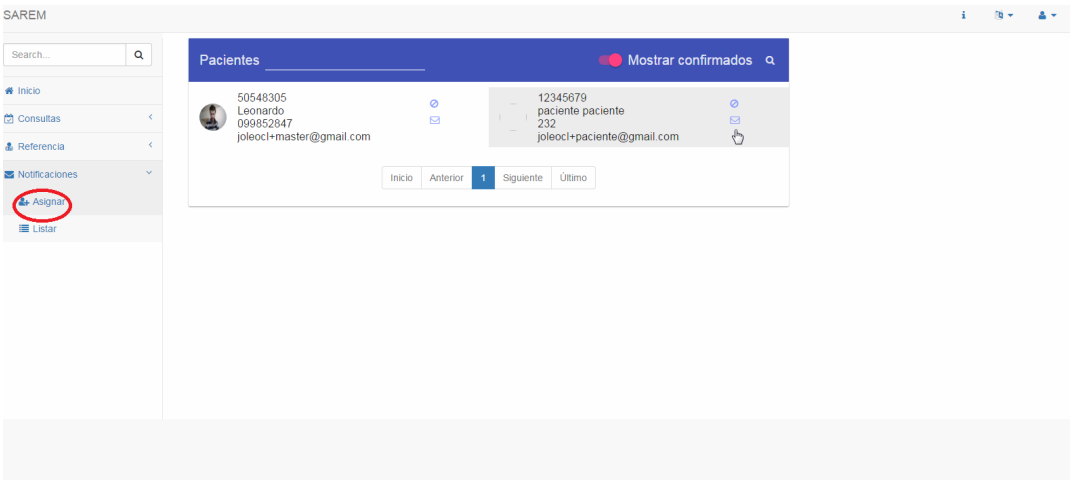


Figura 22 – Asignar Notificaciones a Pacientes.

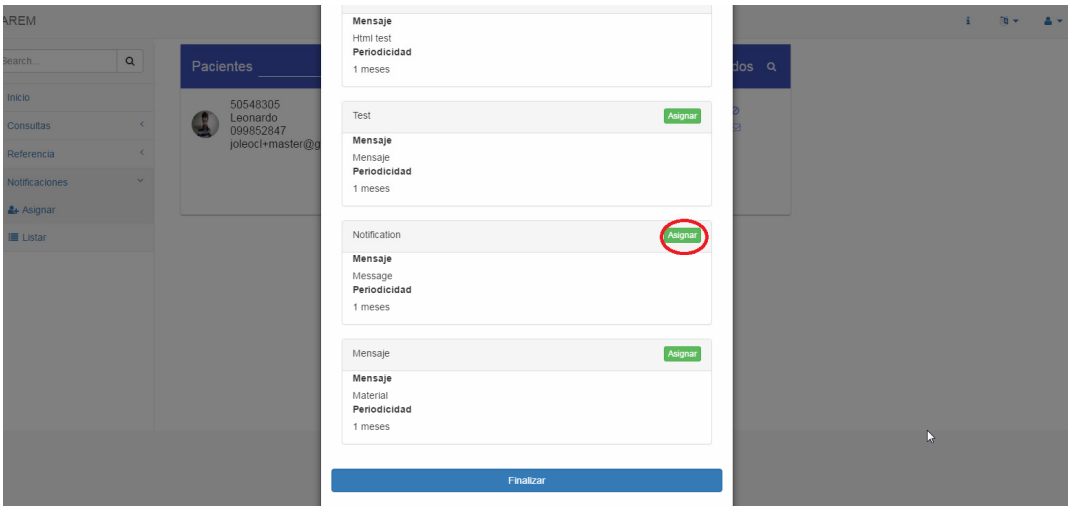


Figura 23 – Modal con Notificaciones para asignar a Paciente.

14.3.12 Caso de Uso Completar Parte Diario

Para ver todas los pacientes que ha atendido un usuario de tipo Profesional se debe seleccionar la opción Consultas > Calendario. Al seleccionar esta opción se le mostrará en pantalla una imagen como la de la **Figura 24**.

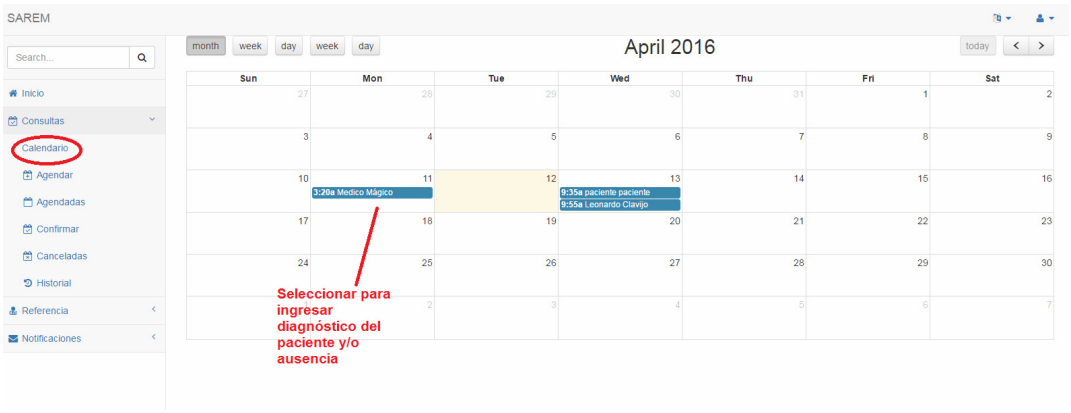


Figura 24 – Ver Calendario.

Si se selecciona alguno de los pacientes que se muestran en el calendario, se mostrara en pantalla un modal para ingresar el diagnóstico del paciente e indicar su ausencia o no al turno de la consulta.

Figura 25 – Ingresar Diagnóstico y Ausencia de Paciente.

14.3.13 Caso de Uso Crear Usuario

Para crear un usuario se debe seleccionar la opción del menú de la izquierda de la pantalla Personas > Crear. Al seleccionar esta opción se visualizara un formulario con campos a completar como muestra la **Figura 26**. Una vez completados estos campos se selecciona el botón “Crear” para finalizar de crear a la persona.

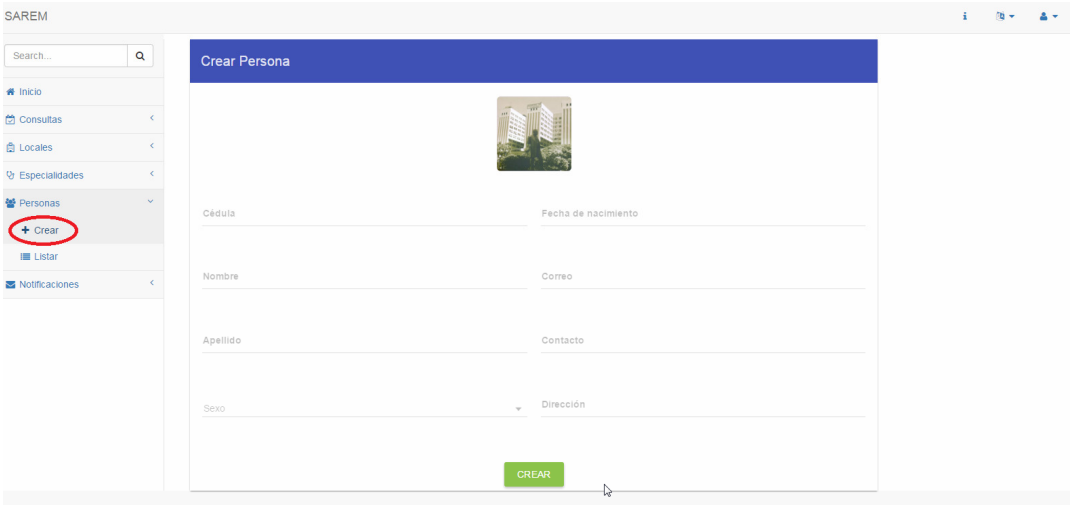


Figura 26 – Crear Usuario.

14.3.14 Caso de Uso Crear Local

Para poder crear los locales físicos donde se van a llevar a cabo las consultas se debe seleccionar la opción Locales > Crear del menú izquierdo de la pantalla. Una vez seleccionada esta opción se verá en pantalla una imagen como la de la **Figura 27.**

Se puede seleccionar el lugar desde el mapa, lo que provocara que los campos Dirección, Ciudad, Estado, Código Postal se completen automáticamente. O se podrán ingresar todos los datos del lugar de forma manual.

Una vez ingresados todos los datos, se debe seleccionar el botón “Crear” para finalizar de crear el local.

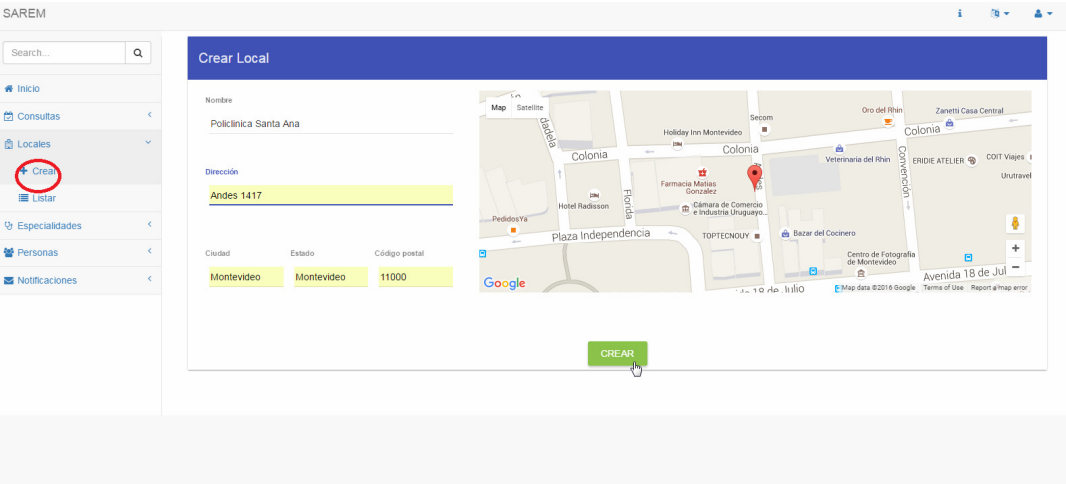


Figura 27 – Crear Local.

14.3.15 Caso de Uso Listar Locales

Para ver todos los locales creados en el sistema se debe seleccionar la opción del menú de la izquierda de la pantalla Locales > Listar. Una vez seleccionada esta opción se podrá visualizar una grilla como muestra la **Figura 28**, con todos los locales creados, teniendo la opción de ver más información sobre el local o eliminarlo.

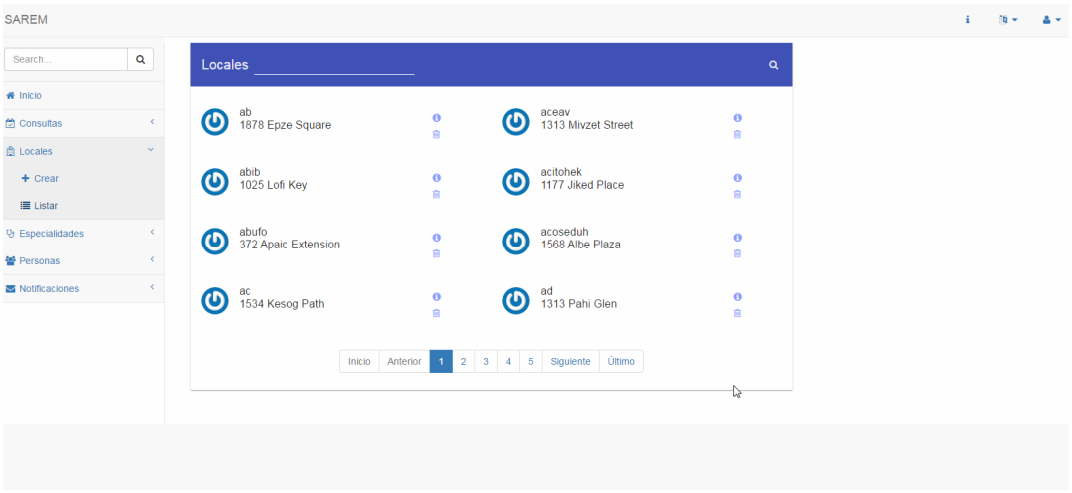


Figura 28 – Listar Locales.

14.3.15 Caso de Uso Crear Especialidad

Para poder crear una Especialidad se debe dirigir a la sección Especialidades > Crear del menú de la izquierda de la pantalla. Una vez seleccionada esta opción vera una imagen como la de la **Figura 29.** Para crear la Especialidad debe completar los campos Tipo y Descripción y luego seleccionar el botón verde.

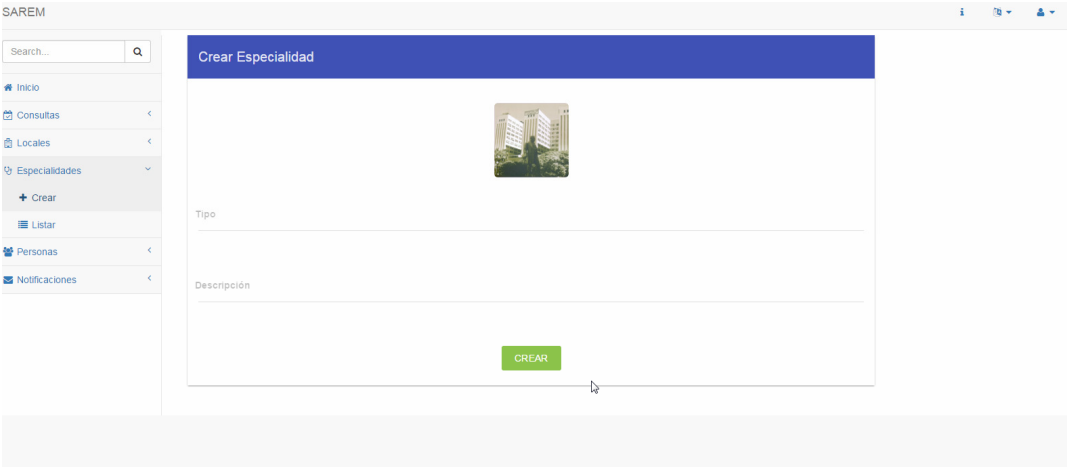


Figura 29 – Crear Especialidad.

14.3.16 Caso de Uso Listar Especialidades

Para poder ver todas las especialidades creadas se debe seleccionar la opción del menú izquierdo de la pantalla Especialidades > Listar. Una vez seleccionada esta opción se verá en pantalla una imagen como la de la **Figura 30**. Cada especialidad de la grilla tiene dos opciones una para eliminar la especialidad y otra para ver/modificar los datos de la especialidad.

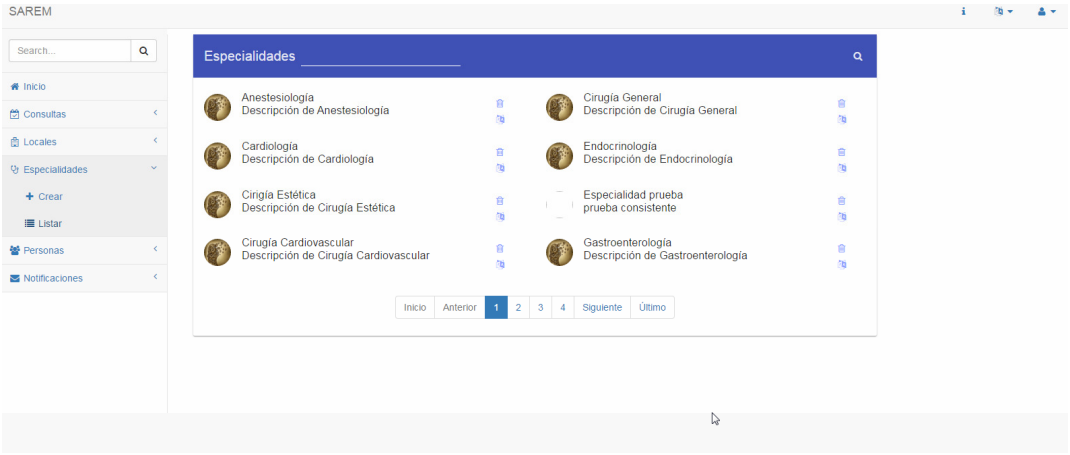


Figura 30 – Listar Especialidades.

14.3.17 Caso de Uso Asignar Roles a Usuarios

Para poder ver/modificar los roles de los usuarios en el sistema se debe seleccionar la opción del menú de la izquierda de la pantalla Personas > Roles. Una vez seleccionada esta opción se le mostrara en pantalla una grilla como la de la **Figura 31**. Cada usuario de la grilla tiene un botón con el icono de un candado, que si se lo selecciona abre un modal para que el usuario pueda ver/modificar los roles como lo muestra la **Figura 32**. En el modal se puede seleccionar para que el usuario tenga roles de Administrador, Paciente, Profesional, una vez seleccionado los roles correspondientes se debe apretar el botón Finalizar.

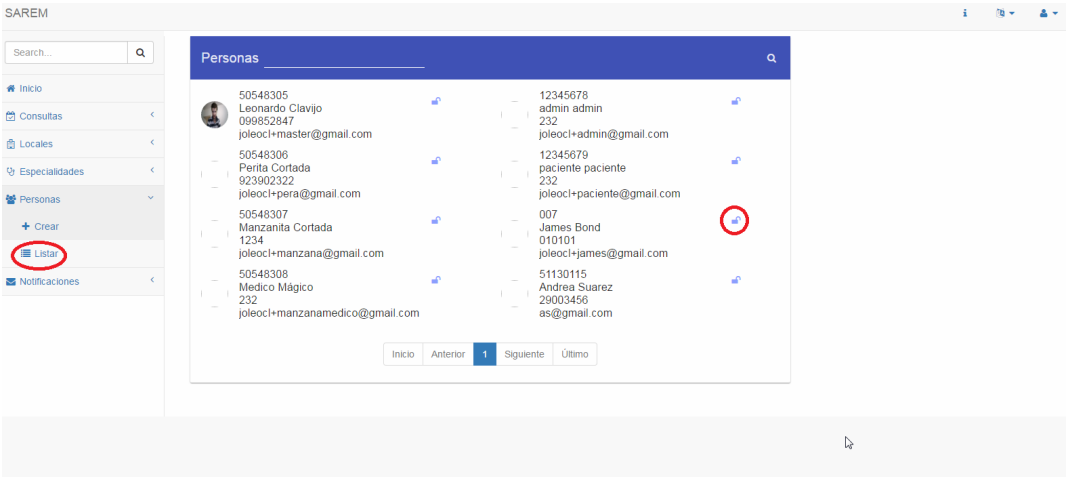


Figura 31 – Ver/Modificar roles de usuarios.

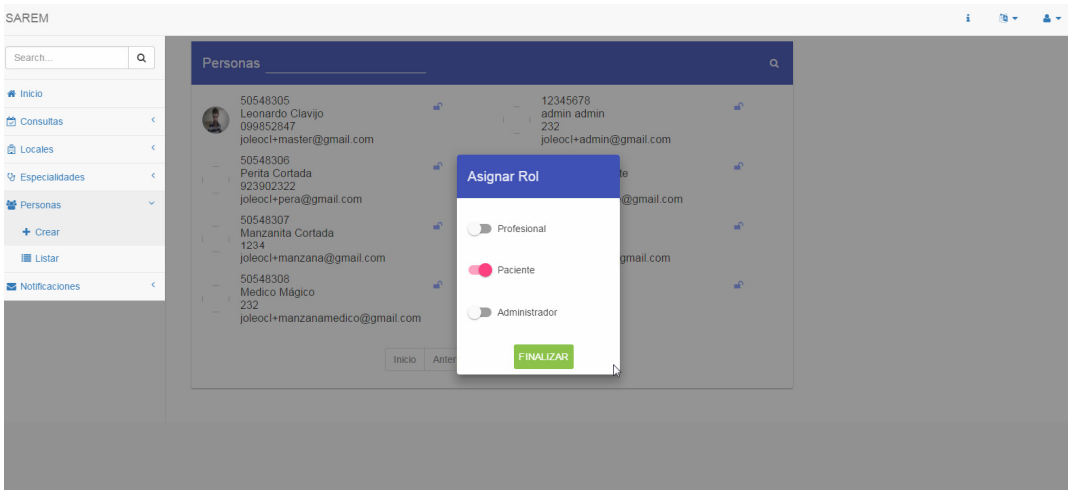


Figura 32 – Ver/Modificar roles de usuarios.

14.3.18 Caso de Uso Modificar Datos Personales

Para poder ver/modificar los datos personales, se debe seleccionar el menú de arriba del todo de la pantalla a la derecha que tiene el icono de la persona como lo muestra la **Figura 33**. Luego se debe seleccionar la opción Perfil. Una vez seleccionadas estas opciones se le mostrara al usuario un formulario con sus datos personales, pudiendo modificar todos menos su Cédula. Para guardar los cambios realizados se debe presionar el botón verde.

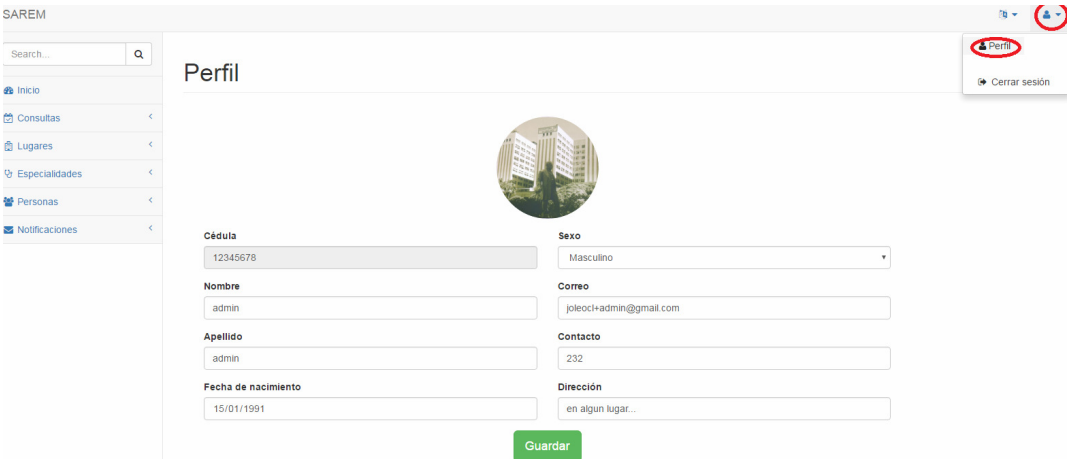


Figura 33 – Ver/Modificar Datos Personales.

14.4 Manual de Aplicación Móvil

14.4.1 Iniciar Sesión

Para iniciar sesión desde la aplicación móvil existen dos posibilidades:

1 – Ingresar el nombre de usuario y contraseña. Luego seleccionar el botón Login.

2 – Iniciar primero sesión desde la página web de SAREM. Luego seleccionar la opción *Conectar al móvil* como muestra la **Figura M2**. Una vez seleccionada esta opción se le mostrara al usuario un código QR como indica la **Figura M3**. A continuación desde la aplicación móvil seleccionar el botón Login QR, y por ultimo escanear el QR para iniciar sesión desde la aplicación móvil.

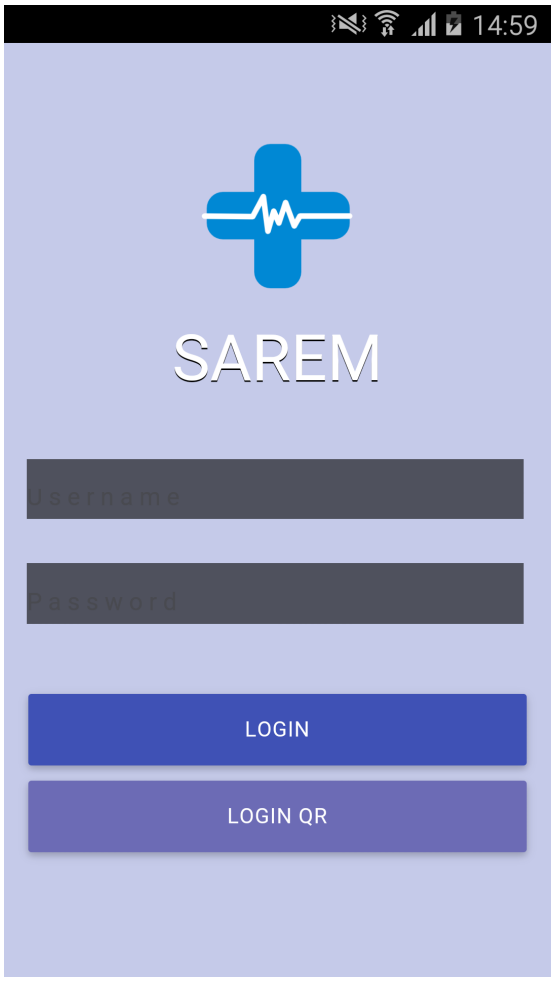


Figura M1 – Iniciar Sesión.

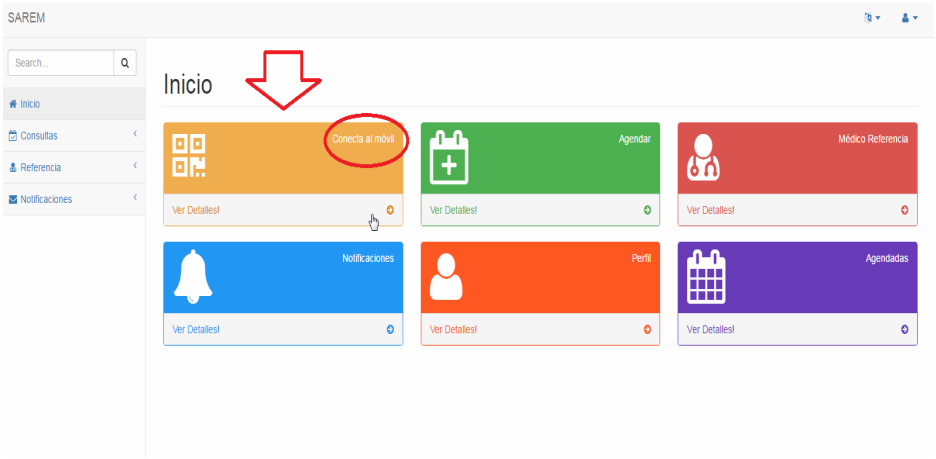


Figura M2 – Conectar al Móvil.

Figura M3 – Código QR.

14.4.2 Agendar Turno en Consulta

Para agendar un turno en una consulta lo primero que se debe hacer es seleccionar el icono del calendario como muestra la **Figura M4**. Luego se debe seleccionar el botón rojo indicado en la **Figura M5** para ver la lista de especialidades disponibles como lo indica la **Figura M6**. Al seleccionar una especialidad se muestran todas las consultas que existen en el sistema con turnos disponibles o lugares libres en su lista de espera ver **Figura M7**. Luego de seleccionar la consulta se debe seleccionar el turno que se desee y por ultimo apretar el botón verde como lo indica la **Figura M8.**

Figura M4 – Seleccionar Calendario.

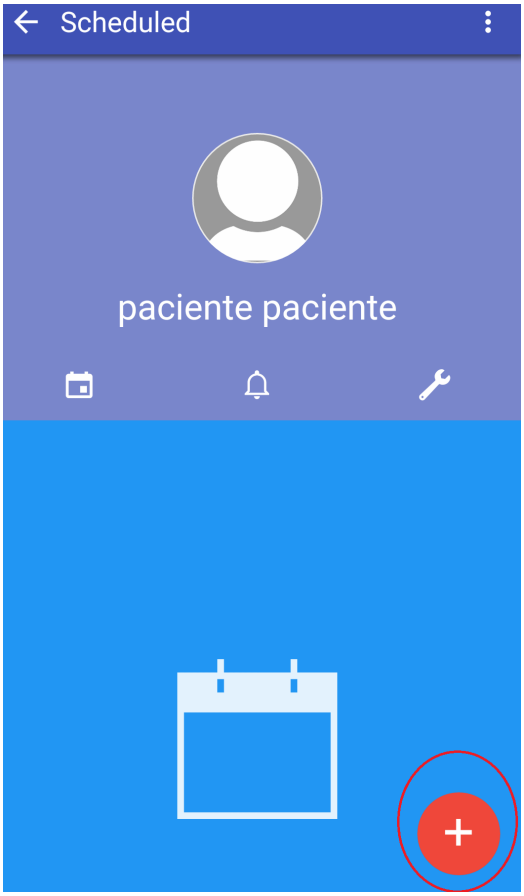


Figura M5 – Seleccionar botón para ver especialidades.

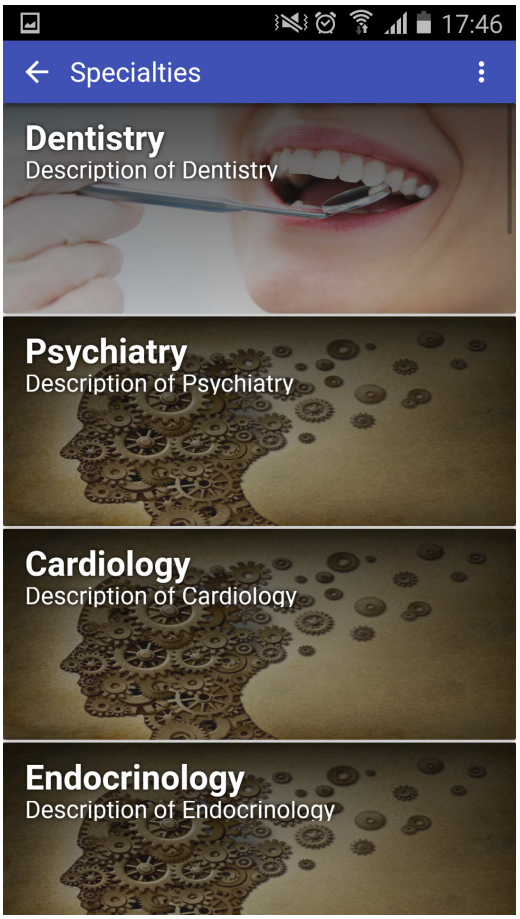


Figura M6 – Seleccionar Especialidad.

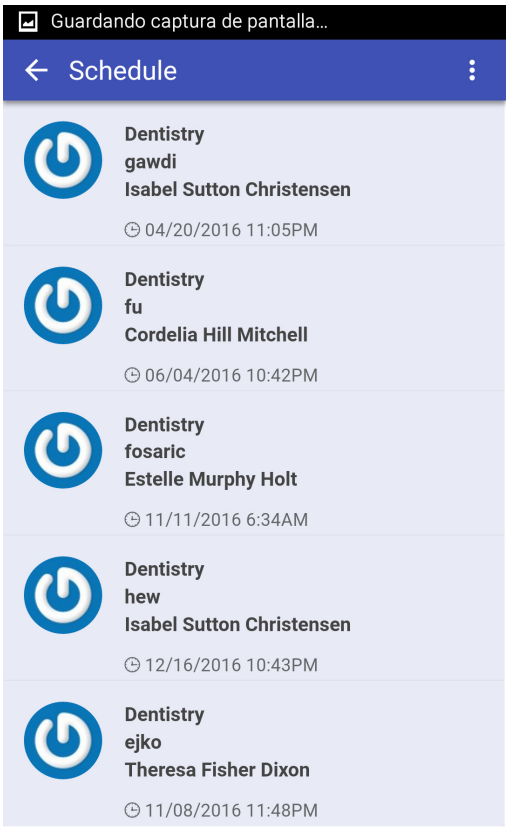


Figura M7 – Seleccionar Consulta dentro de Especialidad.

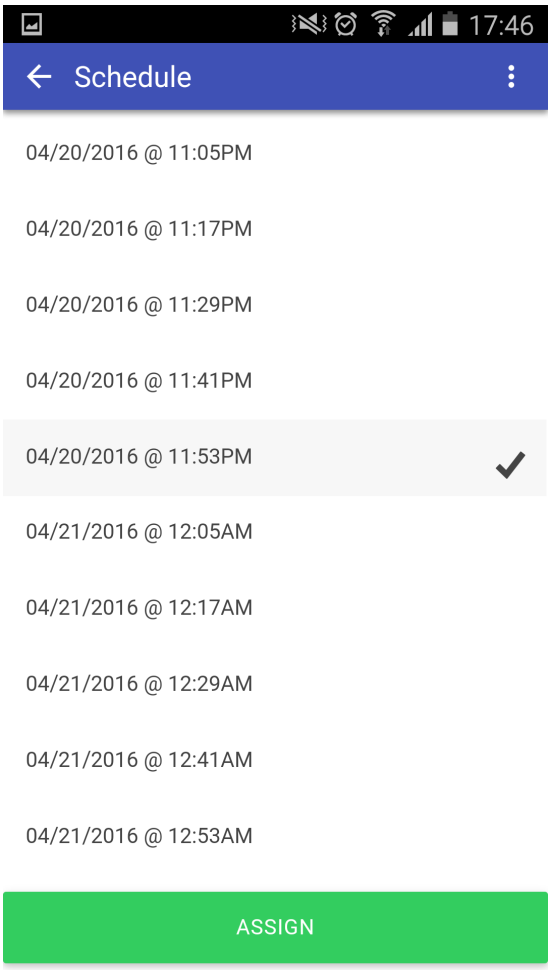


Figura M8 – Seleccionar Turno en Consulta.

14.4.3 Ver Notificaciones Asignadas

Para poder ver las notificaciones asignadas al usuario se debe seleccionar el icono de la campana como muestra la **Figura M9.** También se puede seleccionar cualquiera de las notificaciones que aparecen en la lista para verla en más detalle.

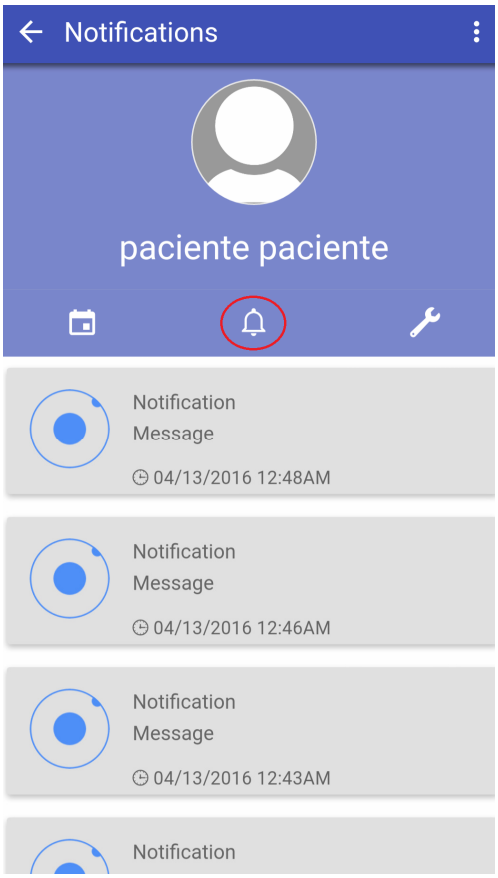


Figura M9 – Ver Notificaciones Asignadas.

14.4.4 Ver/Modificar Datos Personales

Para poder ver/modificar los datos personales se debe seleccionar el icono de la herramienta. Luego se muestran en pantalla los datos del usuario. Para persistir los cambios realizados se debe seleccionar el botón guardar. Ver **Figura M10** y **Figura M11**.

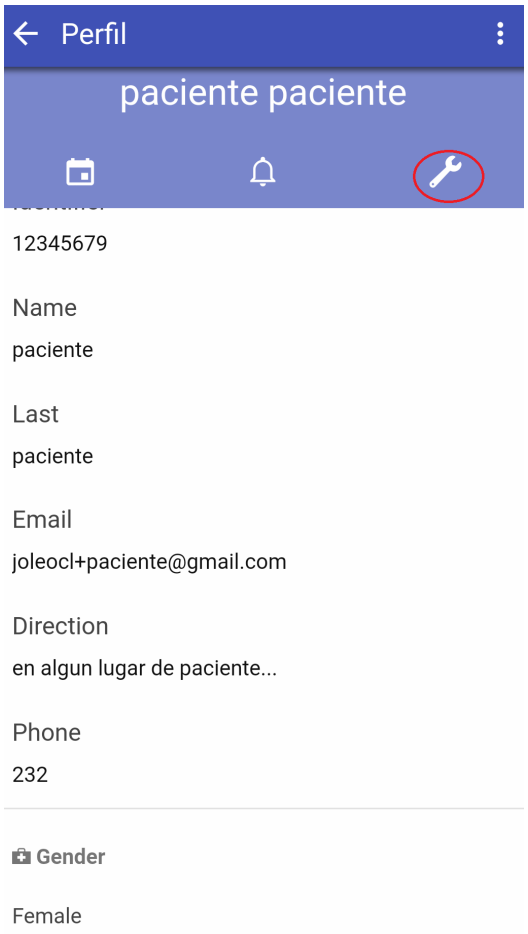


Figura M10 – Ver/Modificar Datos Personales.

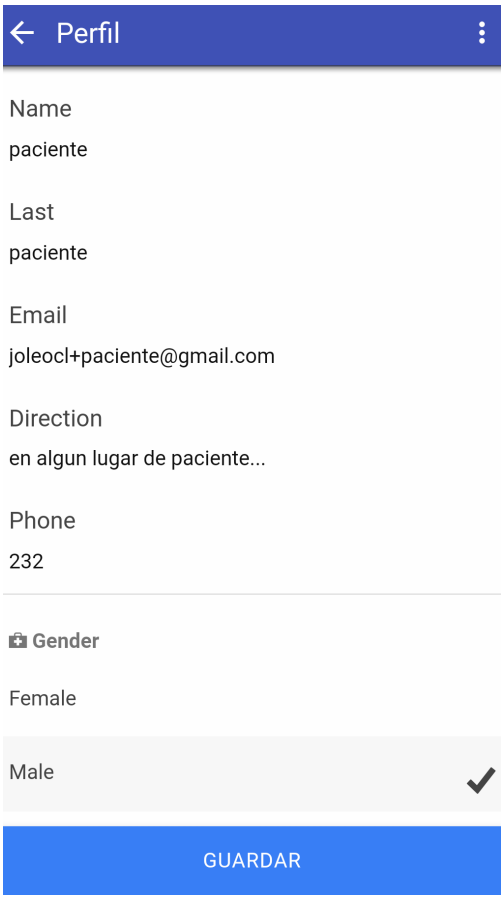


Figura M11 – Ver/Modificar Datos Personales.

14.4.5 Cancelar Turno en Consulta

Para poder cancelar un turno en una consulta lo primero que se debe hacer es buscar las consultas en las que se tienen turnos asignados. Para eso se debe seleccionar la opción Appointments (o Consultas en español) como indica la **Figura M12** en el menú lateral. Una vez seleccionada esta opción aparecerán en pantalla todas las consultas en las que el usuario tiene asignados turnos ver **Figura M13**. Se procede a seleccionar la consulta en la que se quiere cancelar el turno y luego se selecciona el botón cancelar ver **Figura M14**.

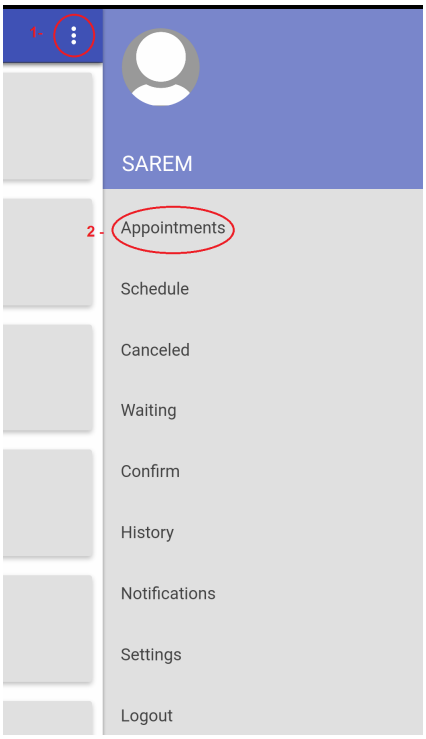


Figura M12 – Ver Consultas Agendadas.

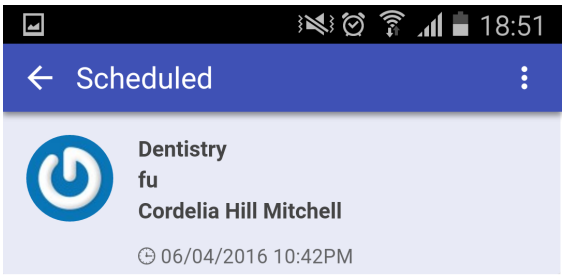


Figura M13 – Consultas en las que se Agendaron Turnos.

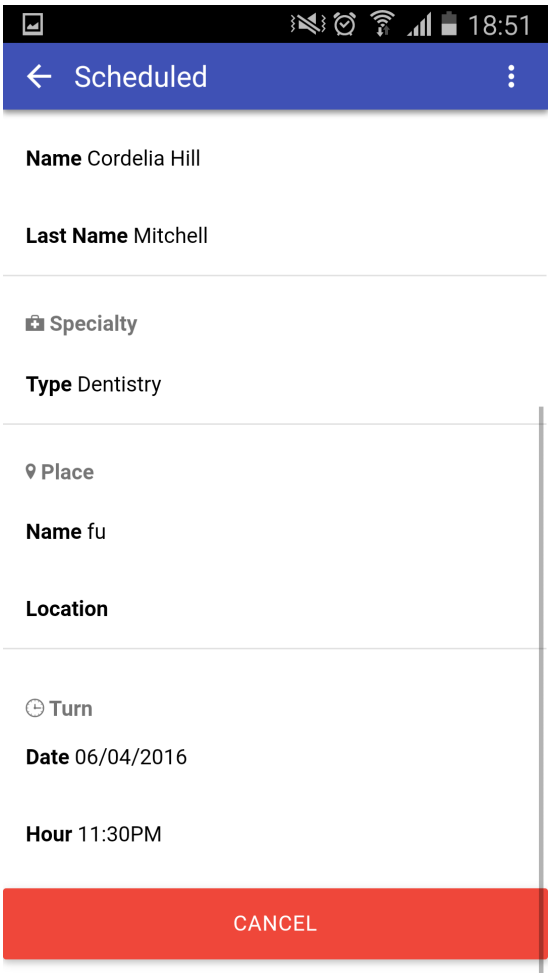


Figura M14 – Cancelar Turno en Consulta.

14.5 Proyecto de Grado SAMI

Facultad de Ingeniería - Universidad de la República

17-02-2016

S.A.M.I

Sistema de Agenda Médica Informática

Definición, especificación, proyecto e implementación de un sistema original para aumentar la eficiencia del agendado de horas de policlínica y el empoderamiento de personas atendidas en el primer nivel de atención.

Documentación del trabajo realizado por:

Gimena Bernadet - 4.503.539-1
Cristiano Coelho - 4.298.259-3
Emiliano Conti - 5.033.483-5

Como parte de los requerimientos para aprobar la asignatura
Proyecto de Grado de la Carrera de Ingeniería en Computación

Tutores: Ing. Lucía Grundel y Prof. Ing. Franco Simini
Núcleo de Ingeniería Biomédica de las Facultades de Medicina e Ingeniería

Trabajo realizado en colaboración con la Escuela Universitaria de Tecnología Médica (EUTM) con la participación de (nombres de las estudiantes, etc.) bajo la guía de la Prof. Saadia Zawadski, directora de la Licenciatura en registros Médicos, en la modalidad de
PIRIM - Proyecto Interdisciplinario de Registros Informáticos Médicos
Con la colaboración de las alumnas de Registros Médicos:
Micaela Ottonello - 4.373.104-6

Montevideo, URUGUAY marzo 2015 – febrero 2016

14.6 Bitácora del Proyecto

14.6.1 Planificación inicial

El proyecto inició a fines de febrero, con la búsqueda de proyecto de grado, llegando a una decisión sobre el tema a mediados de marzo.

Una vez definido el proyecto, los meses de abril, mayo, junio y mediados de julio fueron dedicados al relevamiento de requerimientos, análisis y diseño de la solución.

En un principio nuestro grupo estaba conformado por tres integrantes. Hicimos la planificación del proyecto pensando que contábamos con una persona más para la fase de implementación, con el fin de culminar en diciembre de 2015.

Decidimos dividir las tareas de la siguiente forma: Dos personas se encargaban del frontend de la aplicación realizando la implementación de los casos de uso de los usuarios de tipo Paciente, Profesional, Administrador y Diseñador. El otro compañero restante se encargaba de la implementación del backend y de la integración con OpenEmpi.

Acordamos de realizar reuniones una vez por semana presenciales o de manera online para la coordinación del grupo.

14.6.2 Acontecimientos más importantes

En el transcurso del proyecto ocurrieron varios acontecimientos importantes que afectaron al desarrollo del proyecto y extendieron su finalización. A continuación detallaremos los más importantes.

- Pérdida de un integrante

A principios de setiembre de 2015 uno de los integrantes decidió abandonar el proyecto.

Esta situación nos complicó, ya que contábamos con un recurso menos. Debido a esto decidimos recortar el alcance pautado inicialmente con las estudiantes de RRMM para poder llegar a diciembre de 2015.

Tomamos la decisión de eliminar al usuario de tipo Diseñador, que era el encargado de realizar la personalización del diseño de SAREM. Esto permitía que cada centro de salud pudiera customizar el Look & Feel de SAREM a su gusto. Esto era un plus para la aplicación, pero consideramos que no era un requerimiento tan importante y por eso decidimos no implementarlo.

También decidimos no implementar un módulo de análisis de datos que permitiría obtener estadísticas de asistencias a consultas, cancelaciones, horas de trabajo de profesionales médicos, inasistencias, etc. Si bien consideramos que era un

complemento interesante para SAREM, no era el modulo más importante ya que SAREM tenía el objetivo principal de resolver la dificultad de gestión de citas médicas.

- Presentación en Ingeniería deMuestra

Los días 22, 23 y 24 de octubre de 2015 se realizó el evento de Ingeniería deMuestra del cual fuimos participantes. Con gran esfuerzo, decidimos participar porque consideramos que era un evento importante para obtener retroalimentación del público.

En general, sentimos que el público se mostró a gusto con el proyecto y que lo consideraron de utilidad. Una de las críticas que nos hicieron fue en base al stack de tecnologías que utilizamos. Nosotros decidimos utilizar el stack de tecnologías de Microsoft. Al recibir las críticas en Ingeniería deMuestra sobre esto, nos planteamos la interrogante de si la elección que habíamos hecho era la correcta, y de si nos perjudicaría a la hora de defender el proyecto.

El proyecto ya estaba bastante avanzado en esa etapa y empezar de cero la fase de implementación del proyecto utilizando tecnologías open source podría consumir un tiempo considerable que atrasaría la finalización de SAREM.

En noviembre meditamos sobre esta decisión, y finalmente decidimos comenzar de cero la fase de implementación utilizando tecnologías open source a finales de este mes. También decidimos realizar una aplicación móvil para SAREM, ya que vimos que sería de gran utilidad para la implementación de notificaciones a los usuarios y que tener una interfaz receptiva no era suficiente. Entendimos que esta decisión atrasaría la finalización del proyecto para mediados o finales de marzo.

- Reunión con tutor del proyecto diciembre 2015

A principios de diciembre de 2015 tuvimos una reunión con Franco Simini nuestro tutor de proyecto, donde le comentamos la decisión de migrar SAREM para implementarlo con tecnologías open source. Él nos dio el visto bueno, y nos sugirió realizar mejoraras a SAREM para que contemplara el soporte a varios idiomas y realización de envió de notificaciones con retroalimentación de usuario.

14.7 Nginx Security Configuration

```
proxy_cache_path /tmp/nginx levels=1:2 keys_zone=my_zone:10m
inactive=60m;
proxy_cache_key "$request_method$request_uri";

server {
 root /var/www/app;
 index index.html index.htm;
 server_name ec2-52-67-47-2.sa-east-1.compute.amazonaws.com;

 listen 443 ssl;
 ssl_certificate /etc/nginx/ssl/nginx.crt;
 ssl_certificate_key /etc/nginx/ssl/nginx.key;


 location ~ ^/api/consultas/(search|findOne){
   if ( $request_method !~ ^(GET|POST|OPTIONS)$ ) {
        return 405;
   }
   proxy_pass http://0.0.0.0:3000$request_uri;
 }

location  ~
^/api/(especialidades|locales|notificaciones|images|translations){
   if ( $request_method !~ ^(GET|OPTIONS)$ ) {
        return 405;
   }
   proxy_pass http://0.0.0.0:3000$request_uri;
 }

location  ~
^/api/(personas)/.*/(referencia|espera|canceladas|consultas|history){
   if ( $request_method !~ ^(GET|POST|OPTIONS|DELETE)$ ) {
        return 405;
   }
   proxy_pass http://0.0.0.0:3000$request_uri;
 }

location  ~ ^/api/personas/(count|findOne){
   if ( $request_method !~ ^(GET|POST|OPTIONS)$ ) {
        return 405;
   }
   proxy_pass http://0.0.0.0:3000$request_uri;
 }

location  ~ ^/api/personas{
   if ( $request_method !~ ^(GET|OPTIONS)$ ) {
        return 405;
   }
   proxy_pass http://0.0.0.0:3000$request_uri;
 }

# account access
location  ~ ^/api/accounts/(roles|login|logout|password|reset){
   if ( $request_method !~ ^(GET|POST|OPTIONS)$ ) {
        return 405;
   }
   proxy_pass http://0.0.0.0:3000$request_uri;
 }
```

```
location  ~ ^/api/accounts/.*/persona{
   if ( $request_method !~ ^(GET|POST|OPTIONS|PUT)$ ) {
        return 405;
   }
   proxy_pass http://0.0.0.0:3000$request_uri;
 }


}
```