UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: TraCS (Trabajo, Colaboración y Seguimiento): Herramienta de seguimiento colaborativa de tratamientos de pacientes con discapacidad
Autores: Martín Nicolás Trejo, Leandro Nicolás Vilas
Director: Lic. Ivana Harari
Codirector: Lic. Ana Paola Amadeo
Carrera: Licenciatura en Sistemas

Resumen

La presente Tesina de Grado consiste en desarrollar una aplicación multiplataforma enfocada en el proceso de seguimiento y tratamiento de pacientes con discapacidad. De esta manera buscamos poder realizar una transformación en la forma en la que actualmente se realiza el seguimiento del día a día de la vida de una persona con discapacidad, pasando del método tradicional a uno informatizado que permita la integración y centralización de la información, junto a la interacción en tiempo real de los participantes del seguimiento, para poder tomar decisiones rápidamente, velando por el bienestar y contención del paciente. Mediante el uso de la aplicación, el monitoreo se producirá de manera colaborativa, sincronizada y en tiempo real, involucrando tanto a todos los pilares vitales en el cuidado, contención y diagnóstico del paciente, como también al paciente en sí. A su vez buscamos que sea una herramienta que no se centre solo en un tipo de trastorno o discapacidad, sino que pueda ser adaptable a las necesidades de la familia y los profesionales involucrados en cada caso particular.

Palabras Claves

- TRACS - Groupware - Awareness
- Georreferenciación - Cloud Computing - Desarrollo multiplataforma - API - Aplicaciones Híbridas - Node JS - MongoDB - Socket IO - Ionic

Trabajos Realizados

En primer lugar se realizó una investigación de antecedentes acerca de la problemática y el contexto de los temas a desarrollar. Luego se hizo un relevamiento con usuarios finales para determinar las funcionalidades principales y necesarias. Después se programó una aplicación móvil basada en el concepto de groupware, llamada TraCS. Para esto también se desarrolló un servidor REST que provee datos. Se realizaron pruebas con usuarios para evaluar la experiencia de usuario y la utilidad de la aplicación

Conclusiones

Basándonos en los comentarios positivos recibidos por parte de los profesionales que harían uso de la aplicación y que se mostraron realmente interesados en empezar a usarla en su ámbito laboral, concluimos que logramos el objetivo de crear una herramienta que produzca una mejora en la forma de realizar el acompañamiento de un paciente, reduciendo los tiempos de respuesta y generando un acercamiento de los participantes de un tratamiento, fomentando así la comunicación y la participación de los mismos.

Trabajos Futuros

A lo largo del desarrollo de la aplicación surgieron un gran número de posibles funcionalidades que hemos decidido incluirlas en una lista de desarrollos futuros entre las que podemos destacar: tener un cronograma de sesiones a las que asiste el paciente, poder configurar la pantalla del paciente, tener recordatorios de horario de medicación, tener la información del paciente disponible de manera offline y poder configurar las plantillas de los mensajes de texto que envía el paciente.

Fecha de la presentación: Noviembre de 2016

Agradecimientos

Queremos agradecer a todas las personas que colaboraron a lo largo de este trabajo. A nuestras directoras Ivana Harari y Ana Paola Amadeo que nos ayudaron y guiaron en la definición y la realización de esta tesina. A Sofia Lukaszczuk que fue la persona que cumplió el rol de usuario final, planteando la problemática en base a la cual se desarrolló este trabajo y a su vez participó en las pruebas finales de la aplicación. A Federico Mazza que diseñó el logo y la paleta de colores usada en las distintas pantallas de la herramienta. A Luciana Pacheco por transmitirnos sus conocimientos en Experiencia de Usuario y ayudarnos a la hora de realizar los testeos. Por último a Yasmin Luz Quintanilla, Emiliano Moreno, Paula Castro y Julia Saborido que participaron en las jornadas de prueba de la aplicación y nos ayudaron a entender la importancia de los testeos con usuarios.

Personalmente, Leandro, me gustaría agradecer a Martín por bancarme durante toda la carrera y ser un gran amigo y compañero de trabajo. También a mi familia por siempre incentivarme, e insistirme, a estudiar y terminar la carrera. Y finalmente a todas las personas que se mostraron interesadas en este trabajo y nos motivaron a continuarlo y a realizarlo de la mejor manera posible.

En mi caso, Martín, me gustaría agradecer a mi familia, en especial a mis padres, Daniel y Susana y a mi hermano Julián que siempre estuvieron ahí presentes como sostén en todas las etapas de mi vida. A Sofía que además de ser una gran amiga nos dió una mano gigante para hacer este trabajo. A los amigos del colegio, que siempre están cuando uno los necesita y por último a todos los amigos que coseché a lo largo de mis años en la facultad, que por cierto son una de las mejores cosas que me llevo de mi paso por la universidad. Principalmente a Leandro, que más allá de ser un gran amigo, fue también un gran compañero durante el desarrollo de la tesina.

Para terminar nos gustaría hacer un agradecimiento a la Facultad de Informática, UNLP, que nos permitió estudiar a lo largo de estos años y nos formó como profesionales, demostrando el importante papel que cumple la educación pública en el país.

Índice de contenidos

Introducción

El tratamiento de un paciente al cual se le diagnostica algún tipo de discapacidad cognitiva involucra un grupo numeroso de personas empezando por los padres y familiares, profesionales de la medicina y la psicología y llegando a toda persona que tiene una interacción con él, como pueden ser los maestros del colegio al cual asiste o las personas que quedan a su cuidado cuando esté realizando alguna actividad recreativa fuera de su hogar.

Para poder desarrollar un buen seguimiento de la evolución del paciente es de vital importancia tener en consideración todo signo de mejoría o decaimiento que pueda presentar. Cada interacción que ocurre entre el paciente y alguno de estos participantes puede generar algún tipo de comportamiento o conducta que puede resultar relevante para la persona que está en contacto con el paciente en ese momento o para alguno de los demás involucrados. De la forma que se viene trabajando hoy en día, ante la ausencia de un canal universal por el cual pueda fluir esta información, muchas veces estos eventos quedan olvidados y no pueden ser tenidos en cuenta a la hora de una evaluación integral de mejora.

El tiempo es otro factor crucial que impacta directamente en la evolución del paciente. Coordinar los encuentros con los profesionales involucrados en un tratamiento suele ser una tarea complicada, principalmente por la poca disponibilidad de turnos. Esto genera que el encuentro entre estos y el paciente se disten en el tiempo, generando un asincronismo entre el acontecimiento de un hecho y el momento en el que se lo comunica al profesional. Más difícil aún es realizar encuentros entre los participantes para discutir opiniones y puntos de vista que cada uno tiene acerca del paciente viéndolo desde su disciplina.

Motivación

Como se puede apreciar en la situación planteada en la introducción, las tareas ligadas al seguimiento evolutivo del tratamiento de las personas con algún **trastorno o discapacidad** sigue realizándose de una manera clásica y asincrónica, sin aprovechar las grandes ventajas que ofrecen las nuevas tecnologías combinadas con el concepto de trabajo colaborativo o “groupware” [1,2,3].

Podemos definir groupware como la consecución de unos objetivos comunes, maximizando resultados, minimizando la pérdida de tiempo y, sobre todo, aportando conocimiento al resto de los integrantes de un equipo de trabajo. Cuando se adopta este tipo de dinámicas, el equipo desarrollará una serie de procesos internos que regulen su funcionamiento y los objetivos de sus proyectos. Normalmente, a estos procesos se les suele dar soporte mediante herramientas en línea que faciliten la interacción, la participación y, además, sirvan para realizar el seguimiento de tareas y facilitar el reporte de resultados.

Con el fin de poder dar una solución a la problemática planteada, decidimos enfocar el desarrollo de la presente Tesina de Grado en la creación de **TRACS** - **Tra**bajo, **C**olaboración y **S**eguimiento -, una aplicación multiplataforma enfocada en agilizar y facilitar el proceso de seguimiento y tratamiento de pacientes con discapacidad. De esta manera se busca poder realizar una transformación en la forma en la que actualmente se realiza el seguimiento del día a día de la vida de un paciente, pasando del método tradicional a uno informatizado que permita la mayor optimización del mismo.

Objetivos

Mediante el uso de la aplicación, el monitoreo se producirá de manera colaborativa, sincronizada y en tiempo real, involucrando tanto a todos los pilares vitales en el cuidado, contención y diagnóstico del paciente, como también al paciente en sí. A su vez buscamos que sea una herramienta que no se centre solo en un tipo de trastorno o discapacidad, sino que pueda ser adaptable a las necesidades de la familia y los profesionales involucrados en cada caso particular.

Uno de los beneficios que buscamos obtener mediante TRACS es la integración entre los diferentes actores que intervengan en el tratamiento del paciente. Así se lograría una mejora considerable en la velocidad de transferencia de la información a través del canal de comunicación, ayudando a tomar decisiones mucho más rápidas ante cualquier situación que se presente en pos de mejorar la contención del paciente. Adicionalmente, creemos que la aplicación reducirá de gran manera el tiempo que los usuarios tienen que emplear a la hora de reunirse entre ellos para transmitir sus visiones y opiniones acerca del estado del paciente, o presentar información que ellos mismos podrán transmitir y hacer visible a través de esta herramienta.

Para el desarrollo de la misma contamos con el apoyo de la Lic. en Psicología Sofía Lukaszczuk que se desempeña como acompañante terapéutico de chicos con discapacidad. Ella nos ayudó a comprender las formas en que se desarrolla el seguimiento evolutivo en la actualidad, dificultades y mejoras posibles, y a su vez participó en la fase de análisis de requerimientos y en las posteriores etapas de testeo.

Estructura de la Tesis

La estructura de la presente tesis se divide en 10 capítulos abarcando los siguientes temas:

Capítulo 1 - La tecnología celular

Se enfoca en dar un vistazo general de la forma en que la tecnología celular influye en nuestra vida, su crecimiento a lo largo de los años y una resumida cronología desde su creación hasta la

actualidad. A su vez se realiza una descripción del sistema operativo Android, dando a conocer las diferentes versiones que se lanzaron desde su aparición en el mercado.

Capítulo 2 - Proceso de seguimiento y tratamiento de pacientes con discapacidad

En una primer sección se remarcan las falencias de la metodología utilizada actualmente para el seguimiento evolutivo de pacientes con discapacidad, destacando la forma en que TRACS trata de solventarlos. Luego se puede encontrar un análisis de las aplicaciones con propósitos similares a nuestra herramienta.

Capítulo 3 - Background

Se realiza una descripción de los conceptos y técnicas de desarrollo que consideramos vitales a la hora de idear y desarrollar TRACS.

Capitulo 4 - Elecciones para el desarrollo

En este capítulo explicamos las tecnologías que fueron elegidas para desarrollar tanto el servidor como la aplicación móvil, tomando como base al análisis realizado en el Capítulo 3.

Capítulo 5 - Relevamiento de requerimientos

Durante esta sección se detalla el proceso de elicitación de requerimientos, comentando un poco las reuniones que se realizaron con el usuario final, conjuntamente con la explicación de las técnicas utilizadas para llevarlo a cabo.

Capítulo 6 - Desarrollo

Se relata cómo se llevó a cabo el desarrollo de la aplicación, comenzando en la definición del proyecto, llegando a la puesta en producción. A su vez se pueden encontrar algunos ejemplos de código de funcionalidades que encontramos destacables.

Capítulo 7 - Funcionalidades Desarrolladas

En este capítulo se detallan todas las funcionalidades con las que cuenta la versión de TRACS contemplada en el presente trabajo.

Capítulo 8 - Tests de usuario

Se describe el proceso de pruebas de la aplicación con cinco usuarios diferentes y las conclusiones que se obtuvieron una vez que se finalizó con el mismo.

Capítulo 9 - Conclusión

Como su nombre lo indica, esta sección contiene la conclusión final de todo el proceso de elaboración del presente trabajo. Además se listan las funcionalidades que decidimos incluir en desarrollos futuros.

Capítulo 10 - Bibliografía

Se detalla toda la bibliografía utilizada para el desarrollo de este proyecto.

1. La tecnología celular

Durante las últimas décadas, las comunicaciones han jugado un papel muy importante en la sociedad, ya que permiten la constante interrelación y comunicación entre personas, sociedades, empresas y demás actores del mundo moderno. Se podría decir que sin comunicaciones, la vida como la conocemos no podría existir. Con el paso del tiempo, la necesidad de estar cada vez más comunicados se hizo mayor, y es por ello que empresas y fabricantes relacionados al mundo de la telefonía se encuentran en una constante búsqueda por evolucionar y ofrecer cada vez mejores equipos y servicios.

Según la UIT (Unión Internacional de Telecomunicaciones) hoy en día existen casi 7.000 millones de teléfonos celulares [4], prácticamente lo mismo que el número de habitantes del planeta. El imperio de los móviles está haciendo sombra al uso de las computadoras de escritorio, y es que hay varias características de los smartphones que los hace perfectos para distribuir y consumir contenido [5], y provocar un acercamiento y compromiso con el usuario mucho mayor de lo que puede hacerlo otros dispositivos. En primer lugar, al ser dispositivos portables y valiosos para el individuo, tienden a permanecer con la persona y ser encendidos repetidas veces durante el día. Por lo tanto, ofrecen la oportunidad de brindar contenido y notificaciones importantes inmediatamente y que la persona tome decisiones en base a la mismas y según el contexto donde se encuentre. Segundo, las aplicaciones móviles son una forma barata, conveniente y personal de mantenerse actualizado en cualquier lugar. Tercero, la conectividad facilita y agiliza el intercambio de información entre personas. Además, la habilidad de los smartphones para usar sensores internos para inferir contextos como la posición geográfica, el movimiento o incluso las emociones, posibilita realizar un seguimiento continuo y automatizado de la persona. Por ejemplo, se han desarrollado muchas aplicaciones comerciales que proveen información, consejos y herramientas interactivas de monitoreo para asistir a las personas a manejar el stress, mejorar el humor, seguir una dieta saludable, incrementar la actividad física y dejar de fumar.

Hoy en día muchas de las empresas que ofrecen productos o servicios tienen una o varias aplicaciones móviles para poder generar un lazo constante con el usuario y mantenerlos conectados, ofreciéndoles contenido de interés y actualizaciones constantes.

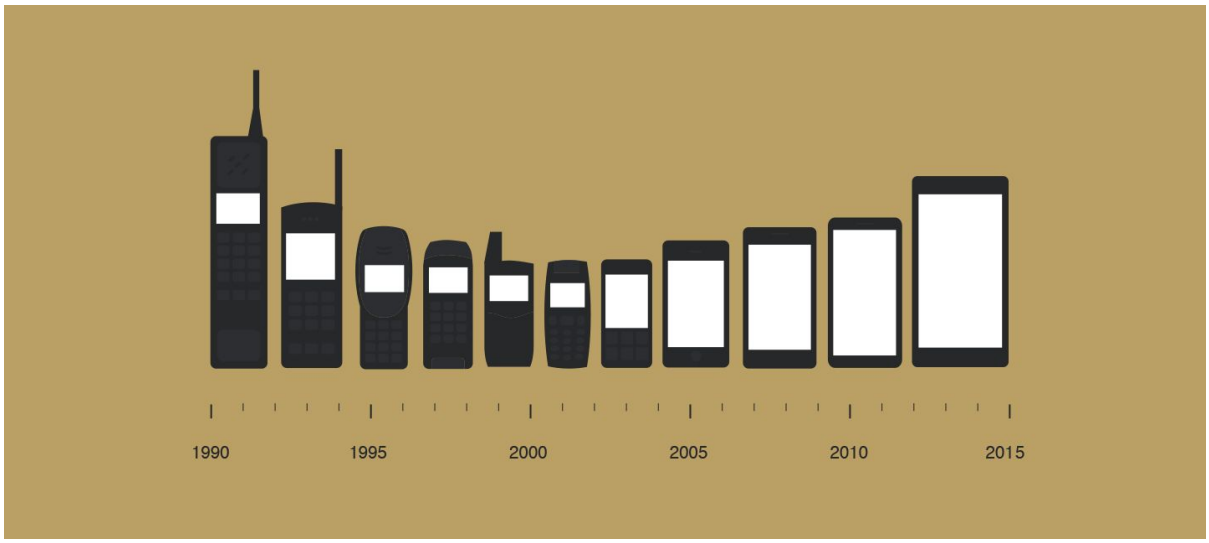
Por estas razones pusimos nuestro foco en una aplicación móvil, para que los participantes puedan estar en contacto todo el tiempo de una forma práctica, cómoda y ubicua, sin tener que dedicarle tiempo a sentarse en una computadora de escritorio a revisar las novedades.

Para poder mostrar de una manera más clara cómo la tecnología móvil impacta en nuestro día a día, decidimos incluir una breve cronología describiendo su evolución, comenzando desde la invención del teléfono llegando hasta los smartphones actuales.

1.1 Un poco de historia

Desde 1854, año que podría definirse como la fecha de creación del mercado de las telecomunicaciones gracias a la invención del teléfono, no se ha dejado en ningún momento de innovar [6]. Y si bien la invención del teléfono fue un acontecimiento histórico y tuvo gran impacto a nivel mundial, que permitió reducir las grandes distancias llevando a las personas la posibilidad de estar comunicadas con mayor frecuencia y facilidad, no fue hasta la llegada de los teléfonos celulares en 1972 que el teléfono se convirtió en un artefacto del cual es imposible desprenderse, alterando, además de la manera de comunicarse, la forma en que se comporta socialmente el individuo que lo utiliza.

Con el paso de los años, los teléfonos celulares evolucionaron (y lo siguen haciendo) de una manera drástica. Comenzaron siendo "ladrillos" (llamados de esta manera por su gran tamaño) y analógicos, para terminar siendo pequeños y digitales con incontables e increíbles funciones.



Evolución de los teléfonos celulares [1]

Primera generación de celulares

1 http://www.lomasnuevo.net/wp-contentupl/2015/10/rsz_mobiltelefonens_evolution.jpg

La primera generación de teléfonos celulares surge con la aparición en el mercado mundial del conocido "ladrillo" (DynaTac 8000X) a fines de los años 80. Estos equipos tenían tecnología analógica para uso restringido de comandos de voz.

Segunda generación de celulares

Surge en la década de los 90, con teléfonos celulares con tecnología digital y con beneficios como duración extendida de la batería, y mayor definición y calidad de sonido. Estos teléfonos, ya tenían la posibilidad de enviar y recibir mensajes de texto (SMS) aunque en esa época no fue una herramienta muy utilizada. A fines de la década se produjo el auge de los teléfonos celulares; la gente común se agregó a la lista de usuarios, favorecidos por el precio y la competencia entre las diferentes compañías.

Tercera generación de celulares

En esta generación se unen las tecnologías anteriores con las nuevas incorporadas en los teléfonos celulares. Surge una masificación, y ahora los mismos poseen un chip (tarjeta SIM), donde se encuentra toda la información. El consumo se extendió notablemente, y una de las causas fue la existencia en el mercado de teléfonos GSM de "bajo rango", como ser los Nokia 1100, Sagem XT, Motorola C200, Alcatel, todos con precios muy bajos.

Junto a estos equipos existían una variedad infinita de modelos de teléfonos con cámaras de foto, y algunos que hasta permitían filmar algunos minutos, pantalla color, conexión rápida a Internet (tecnología EDGE), envío de mensajes multimedia (MMS) y acceso a casilla de e-mail (POP3).

Smartphones y 4G

No cabe duda que la aparición en el mercado del standard de comunicaciones 4G, cambió para siempre el modo en que los usuarios de teléfonos celulares usan su dispositivo. A tal punto esto es así que la telefonía de consumo de entretenimiento tal como la conocemos en la actualidad no podría existir. La unión del smartphone, una impresionante mezcla entre teléfono y computadora, y este nuevo estándar de comunicaciones, sin duda alguna rompió el esquema de consumo de contenidos al cual estábamos acostumbrados desde hace años, ya que gracias a la velocidad de transmisión de datos que puede alcanzar 4G, podemos consumir sin ninguna clase de problemas contenidos de video en alta definición, música en streaming y mil cosas más, además que por supuesto todo lo que tenga que ver con nuestro trabajo lo podremos realizar varias veces más rápido y sin tantas complicaciones, en el lugar y momento en donde nos encontremos, puntos que hoy son vitales para el desenvolvimiento diario de millones de personas alrededor del mundo.

1.2 Historia y evolución del sistema operativo Android

Cualquier persona que tenga un smartphone hoy día habrá tenido que elegir entre dos sistemas operativos mayoritarios. Por un lado tenemos iOS, de Apple, y por otro su principal competidor, Android, desarrollado por Google. Lo curioso de esto es que Android es un sistema operativo que podemos ver en diferentes marcas de telefonía desde su nacimiento, en noviembre de 2007, que es cuando vio la luz su versión beta. Desde entonces, no ha dejado de evolucionar y hoy es a nivel mundial el líder indiscutido por cantidad de usuarios, y lo mismo sucede en la Argentina, ya que se encuentra en el 85% de los equipos contra el 6% que se lleva iOS, a través de la línea de dispositivos iPhones [7, 8].

Android 1.0

Lanzado el 22 de octubre de 2008, el HTC Dream también conocido por entonces como Google Phone fue el primer dispositivo en incorporar el sistema operativo de Google. Este incluyó la primera versión del Android Market, un Navegador Web y algunas aplicaciones para dar soporte a los servicios de Google más populares como Google Maps, con Latitude y Street View, Google Sync para sincronizar Gmail, Contactos y Calendario, Google Search, Google Talk y YouTube.

Por otro lado, incluyó una aplicación capaz de acceder a los servidores de correo de terceros con soporte para los estándares POP3, IMAP4, y SMTP. Tampoco faltó el reproductor de archivos multimedia, que por entonces no era capaz de reproducir video.

Android 1.5: Cupcake

Con la introducción de **Android 1.5** el **30 de abril de 2009**, se empezó a oír el nombre de **Cupcake** en referencia a la primera actualización importante del sistema operativo de Google. La misma le dio un poco más de pulido al SO en algunas áreas, pero sus principales características fueron la introducción del teclado virtual en pantalla y la posibilidad de insertar widgets.

Además se incluyeron otras funciones bastante demandadas por los usuarios como copiar y pegar en el navegador, la grabación de vídeo y reproducción en formatos MPEG-4 y 3GP, la capacidad de subir videos a YouTube directamente, transiciones animadas entre las pantallas, la opción de auto-rotación, auto-sincronización y soporte para Bluetooth.

Android 2.0: Eclair

Lanzada el **26 de octubre del 2009**, la actualización **de Android 2.0 Eclair** debutó en noviembre de ese mismo año en los Motorola Droid y se trató de un hito muy importante para la plataforma que dio paso al crecimiento exponencial y la atención de las masas. Una de las características más destacadas fue su integración social, permitiendo sincronizar los contactos de Facebook, y más tarde, Twitter, dándole a los usuarios la posibilidad de contar con todos sus contactos de sus redes en un solo lugar.

En cuanto a la interfaz de usuarios, también se realizaron mejoras que recayeron básicamente en las animaciones en las transiciones y su fluidez general. Finalmente, una de las mayores novedades de **Android 2.0** fue **Google Maps** que recibió el servicio de navegación GPS gratuito.

Android 2.2 Froyo

Lanzada el **20 de mayo de 2010**, **Android 2.2 Froyo** fue una de las actualizaciones que consagró al sistema operativo como la competencia de **iOS 4** de Apple, dotando a los terminales Android con un notable incremento de la velocidad de todo el sistema, tanto en sus aplicaciones como en la navegación de Internet. Incorporó el motor de Java 8 y aumentó la velocidad gracias al compilador JIT (Just-In-Time) que permite iniciar las solicitudes más rápido y mejorar el rendimiento general del sistema.

Por último cabe destacar otras características incluidas como la opción para mover las aplicaciones a las tarjeta microSD, una pantalla de inicio ligeramente modificada, nuevos widgets, más mejoras en la galería de fotos y un puñado de características de Exchange.

Android 4.0: Ice Cream Sandwich

La llegada de **Android 4.0 Ice Cream Sandwich** el **19 de octubre de 2011** significó un importante paso en la evolución de Android que no solo vio renovada casi por completo su interfaz de usuario con el nuevo diseño Holo, sino que volvió a integrar el sistema operativo en sus versiones para Tablets y Smartphones. La nuevo interfaz de usuario se mostró como la evolución y perfeccionamiento de las ideas de Android 3.0 dándole un poco de esa mirada limpia y futurista.

Pero no todo en Android 4.0 tuvo que ver con el diseño, Google incluyó algunas mejoras que hoy usamos a diario como la posibilidad de acceder a las aplicaciones directamente desde la pantalla de bloqueo y Google Chrome como navegador por defecto que permitió abrir hasta 15 pestañas y realizar la sincronización automática con los marcadores de la versión de escritorio.

Android 4.1: Jellybean (versión mínima de Android para la cual fue desarrollada TRACS)

Presentada el **27 de junio de 2012,** el objetivo primordial de **Android Jelly Bean** fue mejorar la estabilidad, funcionalidad y rendimiento de la interfaz de usuario, para lo cual se implementó el núcleo de linux 3.0.31 y una serie de mejoras en lo que se llamó Project Butter, que permitió aumentar hasta 60 FPS las transiciones en la interfaz de usuario, dando una experiencia realmente fluida.

También se mejoró notablemente la barra de notificaciones, que ahora permite realizar más acciones desde esta, cómo realizar llamadas o acceder a diferentes opciones y mostrar información proveniente de las aplicaciones que lanzan la notificación.

Android 4.4: KitKat

Esta versión no ofrece una lista enorme de cambios radicales y grandes transformaciones en la funcionalidad como lo que vimos cuando se lanzó Ice Cream Sandwich. En cambio, el propósito principal de KitKat [9] es el comienzo de una estrategia de Google para llevar la última versión de Android a todos los dispositivos, tanto los de calidad superior como los de baja gama. Esto es importante, dado que a medida que pasa el tiempo, los teléfonos celulares se quedan corriendo alguna versión vieja de Android, lo que incrementa la fragmentación del sistema operativo y da a los usuarios una experiencia de Android inconsistente.

Con KitKat, Google redujo el sistema operativo para que pueda ejecutarse en muchos más dispositivos, lo que ayuda a cerrar la brecha entre los dispositivos de gama baja y gama alta.

Android 5.0: Lollipop

Lanzado el 12 de Noviembre de 2014, Android 5.0 Lollipop presentó, dentro de sus cambios, una interfaz de usuario renovada con una serie de innovaciones y nuevas funcionalidades, pero lo que se destacó fue su nuevo diseño Material Design, con colores llamativos y un diseño intrépido.

Con respecto a la batería, Lollipop extiende la vida útil del dispositivo hasta 90 minutos con Project Volta. En cuanto a las notificaciones, nos trae nuevas formas de controlar la forma en la que nos aparecen y cómo las recibimos gracias a los perfiles que se configuran por hora y por aplicaciones. Sumado a esto, ahora las notificaciones aparecen en un sólo lugar para un acceso más rápido y sencillo.

Otra modificación importante fue la sustitución del Dalvik por parte de ART (Android Runtime), la nueva máquina virtual de Google diseñada para entregar un mejor rendimiento de las aplicaciones.

Material Design

Google lo define como un lenguaje visual que sintetiza los principios clásicos del buen diseño con la innovación y las posibilidades de la tecnología y la ciencia [10]. El objetivo de Material es desarrollar un sistema base que permita una experiencia de usuario unificada en las distintas plataformas y tamaños de dispositivos.

Es un diseño donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal. Recibe su nombre por estar basado en objetos materiales [11]. Piezas colocadas en un espacio (lugar) y con un tiempo (movimiento) determinado.

Precisamente este diseño basado en objetos es una manera de intentar aproximarse a la realidad, algo que en un mundo donde todo es táctil y virtual es difícil. Intenta guiarse por las leyes de la física, donde las animaciones sean lógicas, los objetos se superpongan pero no puedan atravesarse el uno al otro.

Material Design tiene sus propias normas para casi todos los detalles y se mantienen independientemente del tamaño de pantalla. Esta transversalidad es uno de los puntos fuertes que lo caracterizan y lo hacen tan innovador.

Sabiendo el alcance que podía llegar a tener una aplicación en Android, y dado que conocíamos mejor la manera en que el usuario interactúa con una aplicación Android y los lineamientos de Material Design, decidimos desarrollar nuestra herramienta para este sistema operativo y concentrarnos en la compatibilidad con las versiones más usadas en el mercado (Android 4.1 Jellybean en adelante).

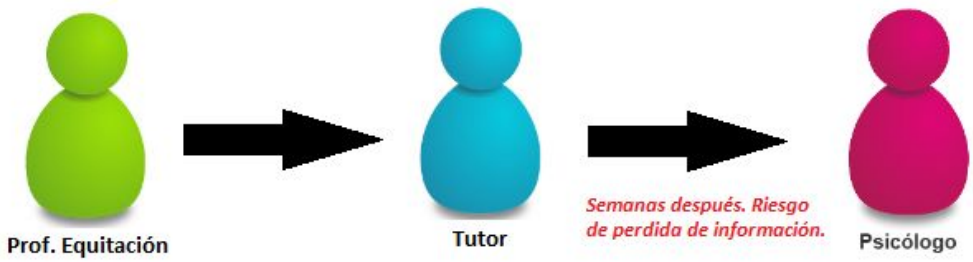
2. Proceso de seguimiento y tratamiento de pacientes con discapacidad

Como explicamos anteriormente, la metodología de evaluación y seguimiento de la evolución de un paciente con algún trastorno cognitivo que se está utilizando en la actualidad cuenta con una serie de problemáticas que impactan directamente en los resultados del tratamiento. Los mismos pueden resumirse en:

- Falta de comunicación entre los participantes
- Asincronismo entre el suceso de un avance/retroceso y la toma de decisiones
- Pérdida de información, al existir casos en que la misma debe ser transmitida al participante competente a través de intermediarios

Otro punto problemático es la forma en que se viene trabajando hoy en día es el almacenamiento y documentación de la información. Esto se debe a la ausencia del uso de alguna herramienta que permita llevar un control y almacenar tanto las observaciones mencionadas en el párrafo anterior, como las notas que se vayan realizando en cada encuentro que se de con el paciente. Este proceso se viene realizando en la mayoría de los casos en papel. Para representar esto decidimos emplear el siguiente ejemplo:

El paciente tiene una reacción positiva durante la clase de equitación terapéutica hecho que resulta de vital importancia para el análisis del psicólogo de cabecera. En este caso el profesor de equitación le comunicará acerca de ello a los padres y luego ellos se lo deberán comentar al psicólogo de cabecera en la próxima sesión a la que tengan que asistir con su hijo, estando la misma acordada para dentro de 15 días. Como uno puede ver, este proceso puede demorar semanas y hay posibilidades de que la información se pierda en el camino si no se la registra correctamente.



Representación de flujo de información sin TRACS

Con TRACS, nuestro objetivo es centralizar la información y generar un canal único de interacción para acortar la brecha comunicacional en tiempo y espacio. De esta forma, el profesor podrá ingresar esta información y la misma estará disponible en tiempo real tanto para el Psicólogo como también para los demás participantes del tratamiento que estén dados de alta en la aplicación. Así se reducen los tiempos de comunicación, se mitigan los riesgos de pérdida de detalle y a su vez se acorta el flujo de transmisión. Esto último resulta importante para los casos en que la información que se quiere transmitir de un profesional a otro resulta ser de un carácter demasiado técnico como para ser manipulada por los familiares. Otra ventaja que se puede destacar es la posibilidad de volver a revisar estos datos en un futuro para realizar algún tipo de informe, ya que siempre estarán disponibles en la aplicación.

Representación de flujo de información utilizando TRACS

2.1 Antecedentes

En el marco de nuestra investigación de antecedentes, encontramos algunas aplicaciones que, si bien no resolvían la problemática en su totalidad, implementaban algunas de las funcionalidades que nos interesaba tener para mejorar la interacción entre el equipo.

La primera que nos pareció interesante destacar y comentar por la concentración de la información y el manejo de roles, es la aplicación ***Conectar Igualdad: Plataforma web de trabajo colaborativo y reportes*** [12], que si bien estuvo vigente hasta el 2013 no deja de ser un buen ejemplo para observar y tener en cuenta. Es una plataforma donde los usuarios pueden subir y compartir contenido educacional en un repositorio centralizado, para después consultarlo a través de la biblioteca de medios, informes, noticias, etc.

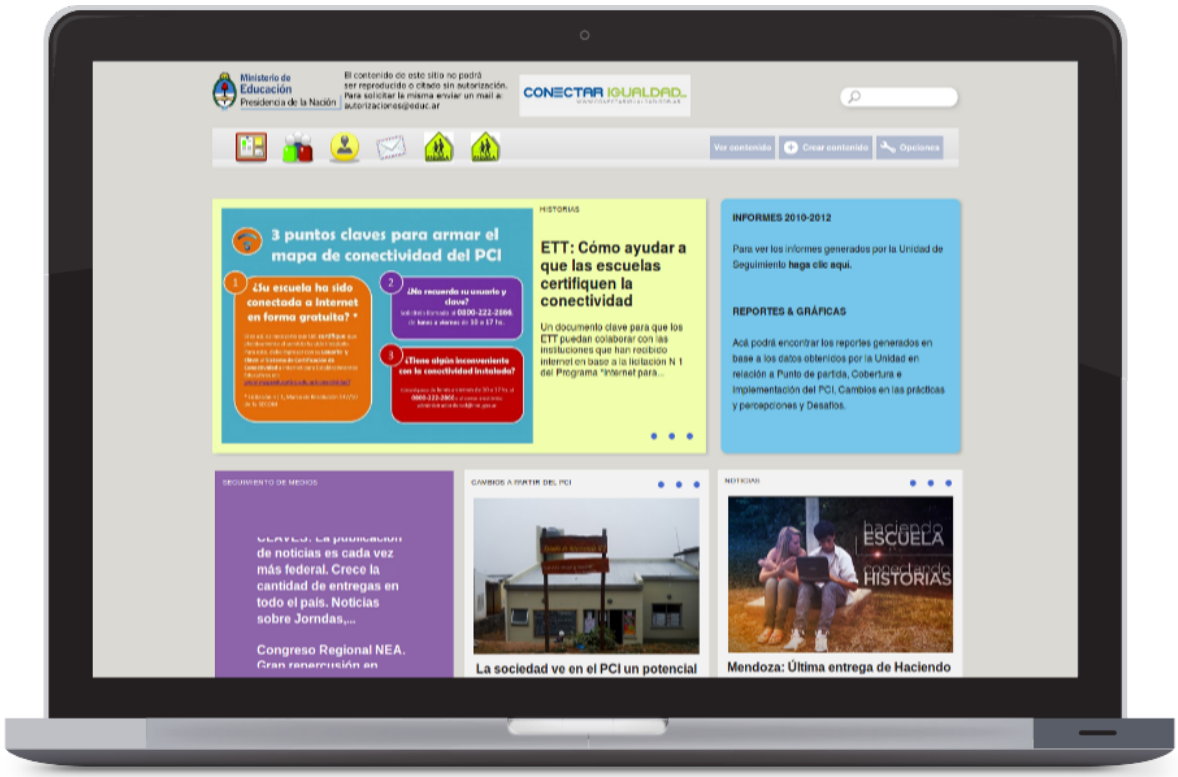


Imagen de ejemplo de la plataforma conectar igualdad [2]

El proyecto plantea una solución web que permite al cliente gestionar las tareas de "Evaluación y Seguimiento" constituyéndose también como una base de recursos que facilita el trabajo y que permite ver reportes en línea, con un completo sistema de permisos que da cuenta de los diversos roles y tareas. El desarrollo se realizó con Drupal CMS, una plataforma libre de gestión de contenidos.
En el sistema se registra el trabajo realizado en el campo, esto supone carga de distintos tipos de encuestas, audios e imágenes. Más allá de los usuarios que utilizan el sistema para organizar y sistematizar su trabajo (con diversos roles), existen roles como el de Director de Escuela o Referentes Territoriales que ingresan al mismo para visualizar los resultados que refieren a su área de trabajo.

La segunda aplicación que no podíamos dejar de mencionar por su funcionalidad basada en georreferenciación es la del ***Botón de Alerta de la Provincia de Buenos Aires*** [13].

2 https://www.gcoop.coop/sites/www.gcoop.coop/files/conectar-plataform-trabajo.png

Consiste fundamentalmente en un botón que el usuario puede pulsar en caso de emergencia (robo, incendio, o cualquier situación de riesgo) para activar una comunicación con un número de emergencias (911, ambulancias o cualquier persona de confianza) y al mismo tiempo enviar mensajes (ya sea vía SMS, correo electrónico o publicaciones en redes sociales) con información georreferenciada del usuario a una lista de contactos. La misma estuvo activa entre Abril de 2014 y Junio del 2016

Imagen ejemplo Botón de Alerta BS. AS.[3]

Otra aplicación interesante por la forma de manejar y presentar la información de los pacientes es ***Patient tracker*** [14], una herramienta colaborativa pero local (no distribuida) que puede ser utilizada por doctores y enfermeros para recolectar datos de sus pacientes con el objetivo de poder trackear su información general.

[3] http://www.portinos.com/wp-content/uploads/2014/01/01_28_BotonAlerta_6.jpg

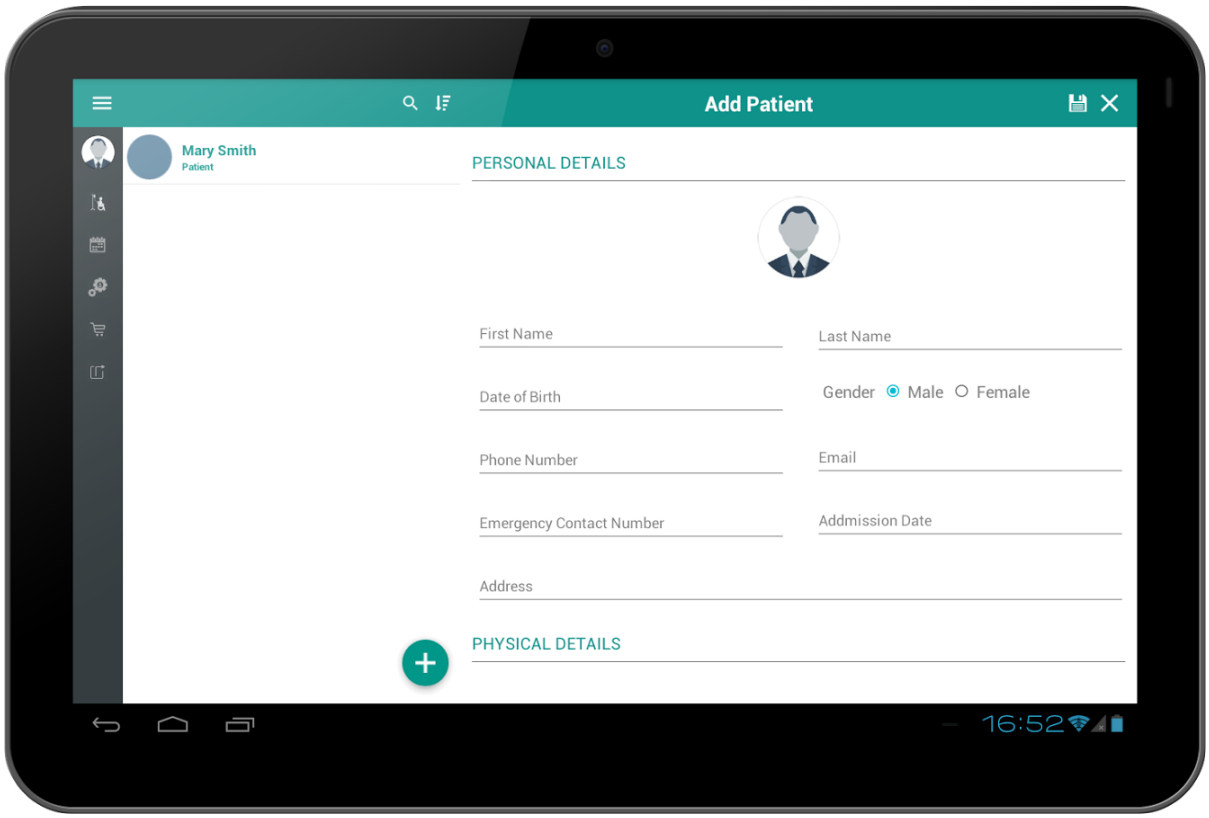


Imagen ejemplo Patient Tracker[4]

En cada visita del paciente, se puede almacenar datos relacionados a los medicamentos que se le recetan y sus signos vitales. Toda esta información es guardada localmente por la aplicación, por lo cual los enfermeros o doctores pueden en cualquier momento chequear el último estado de la evaluación del paciente, su registro de visitas y las medicinas que fueron recetadas conjuntamente con el cronograma de dosaje.

La aplicación más completa y pulida de integración que encontramos para el tratamiento de pacientes es ***Vocera Communication System*** [15], una herramienta multiplataforma utilizada actualmente en el Hospital de Reading, Pennsylvania.

4 https://lh3.googleusercontent.com/5598lZyCoC7Jgy6b8Tc5GBdcLCSAaxoEG4uzPMPfSEXzjIM3j5wk1jPOqnadO6fXrQ=h900-rw

Vocera Communication Badge[5]

A través de celulares y el Vocera Communication Badge (un dispositivo manejado por comandos de voz desarrollado por la empresa) permite conectar en tiempo real a los diferentes individuos a cargo de un paciente, pudiendo estar estos tanto dentro como fuera del hospital. Esta herramienta fue diseñada especialmente para empoderar a los empleados mediante la conexión instantánea entre ellos y las personas con las que trabajan y la información requerida en un momento dado. Además, la misma cuenta con perfiles de usuarios, grupos, manejo y conexiones de llamadas y la habilidad de conectarse a sistemas existentes de telefonía, alarmas y alertas para poder comunicar información crítica.

Desde una perspectiva más académica y orientada a la investigación, encontramos algunas tesis de grado y congresos que vienen al caso y son importante mencionarlos.

La tesis ***Juego educativo móvil colaborativo*** [16] está basada en tres conceptos fundamentales, aprendizaje, ludificación y colaboración. Propone un juego para que los alumnos o participantes adquieran conocimientos a partir de información relativa a la ubicación donde se encuentren mediante un dispositivo móvil.

Nos resultó relevante mencionarla porque varios de los conceptos principales de TRACS están presentes y, si bien se aplican para un objetivo distinto, forma parte del abanico de aplicaciones móviles que promueven la colaboración entre los participantes utilizando georreferenciación.

En la tesis ***Construcción de una Aplicación basada en tecnologías web y dispositivos móviles para la gestión del conocimiento en Comunidades de prácticas*** [17], los autores presentan una

[5] https://i.ytimg.com/vi/ODQPEwBRYvY/maxresdefault.jpg

aplicación tanto móvil como web para facilitar la circulación y la comunicación de conocimientos entre los miembros de organizaciones denominadas Comunidades de Práctica, o COP. Básicamente se trata de un servicio de microblog mediante el cual, de manera sencilla y amigable, los usuarios pueden escribirse mensajes cortos y compartir información publicada en la web mediante un canal RSS (Really Simply Syndication). También, por medio de tags se permite asociar conceptos relevantes del dominio a la información difundida, y cuenta con un sistema de recomendación que permite sugerir novedades a los usuarios que puedan serles de interés o que se encuentren cercanas a su ubicación geográfica.

Esta tesis combina el concepto de georreferenciación para proveer información contextualizada a la ubicación del usuario, junto a una base puramente colaborativa intentando acortar la brecha comunicacional y temporal para la transmisión de conocimientos, dos temas sobre los que ahondamos en TRACS.

En el marco de nuestra investigación también encontramos algunos trabajos y presentaciones en congresos que valen la pena comentar dado que abordan temas que nos resultaron interesantes para el planteo y resolución de nuestra herramienta.

El trabajo ***Taxonomía de mecanismos de awareness*** [18] elaborado para el CACIC del año 2013, define awareness como el entendimiento de las actividades de otros, que proporciona un contexto para la propia actividad. En el marco de un grupo de trabajo mediado por tecnología, el awareness entre los integrantes del grupo se mantiene mediante el acceso a información (por parte de todos los miembros del grupo) sobre cada uno de sus miembros; como por ejemplo: la ubicación de otros participantes en el espacio compartido (¿donde están trabajando?), sus acciones (¿qué están haciendo?), la historia de interacción (¿lo que lo han hecho?), y sus intenciones (¿qué van a hacer?).
El concepto de awareness [18] es fundamental en TRACS, ya que todo gira alrededor de la idea de lograr que las personas que integran el grupo de tratamiento estén todo el tiempo al tanto de los cambios y las novedades sobre el tratamiento del paciente, ya sea si cambió el diagnóstico, si hubo una modificación en la medicación, o si simplemente se incorporó un participante nuevo al grupo. Este trabajo nos resultó interesante porque profundiza el concepto de awareness y lo clasifica en distintos tipos desglosandolo en 4 características básicas, arquitectura auxiliar (constituida por las partes de los mecanismos de awareness que cumplen con un rol específico dentro del sistema), modo de awareness (conjunto de eventos que permite una descripción del estado de situaciones de cooperación), propagación del mensaje (qué personas del grupo recibirán la información), tipos de aplicación (descomponiendo los sistemas colaborativos a través de una matriz espacio-temporal), tipos de presentación (situaciones colaborativas que pueden ocurrir en la interfaz de los usuarios) y métodos de presentación (componentes gráficos o herramientas que se muestran en la interfaz del usuario y dan soporte a la diferente información de awareness).

Cambiando el foco hacia los conceptos de cloud computing y colaboración, el trabajo ***Desarrollo de aplicaciones colaborativas para Cloud Computing*** [19] propone que un entorno de trabajo colaborativo debe ser capaz de brindar a los usuarios la posibilidad de acceder a la información desde cualquier lugar, en cualquier momento y desde cualquier tipo de dispositivo, características

que definen a las aplicaciones del tipo anywhere. El cloud computing surge como una solución para soportar este tipo de aplicaciones que requieren cierta omnipresencia tanto de la información como del usuario, lo define como un modelo de servicios escalables bajo demanda para la asignación y el consumo de recursos de cómputo, donde la característica básica es que los recursos informáticos, tales como infraestructura, plataforma y aplicaciones, son ofrecidos y consumidos como servicios a través de Internet sin que los usuarios tengan que tener ningún conocimiento de lo que sucede detrás. Todo ello se realiza de manera fiable y segura, con una escalabilidad elástica, que es capaz de atender fuertes cambios en la demanda no previsibles a priori, sin que esto suponga un incremento en los costos de gestión.

En este contexto plantea la construcción de una aplicación web para crear un ambiente colaborativo de trabajo para cátedras de la universidad, semejante al desarrollado en TRACS, montada sobre el servicio de cloud computing de Google, Google App Engine (GAE). Si bien en nuestro caso no utilizamos GAE sino Heroku [20] como plataforma host de nuestro servidor, resulta interesante la profundización del concepto y la arquitectura de cloud computing en términos de escalabilidad, disponibilidad, potencia y facilidad de uso para marcar tendencia hacia esta forma de utilizar los recursos computacionales.

Aca le agregaría un párrafo que sintetice la investigación realizada diciendo que no hay algo en el mercado que sea como TRACS, que características tiene que lo diferencian.

Como se pudo observar a lo largo de este capítulo, en estos últimos años se han desarrollado aplicaciones enfocadas al trabajo colaborativo ya sea tanto orientado a la educación como también al ámbito de la medicina. A su vez se puede observar el uso creciente de la georreferenciación como herramienta de contextualización de perfiles o localización de usuarios que estén en alguna situación apremiante. Más allá de esto, ninguna de las aplicaciones analizadas logra agrupar todos los conceptos que abarca TRACS ni tampoco cumplen el objetivo final que la misma posee. En la siguiente tabla podemos observar una comparación entre nuestra aplicación y las anteriormente listadas, basándonos en sus características más destacables:

	Aplicaciones				
	Conectar Igualdad	Botón de alerta	Patient Tracker	Vocera	TRACS
Permite la comunicación entre los participantes					
Fomenta la colaboración de los usuarios					
Utiliza georeferenciacion					
Utiliza notificaciones en tiempo real					
Funciona como repositorio centralizado de informacion					
Esta orientada a la medicina					
Es adaptables a diferentes escenarios					
Brinda funcionalidad adaptada al paciente					
Es multiplataforma					
Es soportada por dispositivos móviles					

Tabla comparativa entre TRACS y aplicaciones analizadas

A diferencia de las aplicaciones orientadas a la medicina que fueron mencionadas, TRACS no solo involucra al personal experto que participa en el tratamiento (médicos, psicólogos) sino que también incluye a toda persona con la que el paciente tenga contacto (familiares, maestros), posibilitando así una mejor apreciación de las diferentes situaciones y avances que se van dando a lo largo del tiempo, ya sea en las sesiones con los profesionales como también en el día a día en su hogar. A su vez podemos destacar el hecho de que nuestra aplicación involucra al paciente brindándole una serie de funcionalidades, situación que no sucede en los casos de Patient Tracker o Vocera. Otra característica que diferencia a TRACS de las herramientas analizadas es su capacidad de poder representar diferentes tipos de tratamientos, independientemente del tipo de trastorno que presente el paciente o los actores intervinientes.

3. Background

3.1 Conceptos contemplados durante el desarrollo

En esta sección se detallan los conceptos que consideramos vitales a la hora de desarrollar una aplicación que cumpliera el propósito buscado. Los mismos van desde metodologías que fomentan la comunicación entre las personas hasta técnicas de desarrollo de aplicaciones.

3.1.1 Groupware

El término "**groupware**" (en español *conjunto de programas informáticos colaborativos*) se refiere al uso de métodos y herramientas de software que permiten que los usuarios realicen trabajos colectivos a través de las redes. Podemos decir que son procesos intencionalmente grupales soportados por software [21].

Por lo tanto *groupware* hace referencia a las diversas y variadas aplicaciones que contribuyen a una única y misma meta: permitir que usuarios separados geográficamente trabajen en equipo. El trabajo en equipo puede ser llevado a cabo compartiendo información o creando e intercambiando datos informatizados. En la mayoría de los casos, groupware se refiere a diversas herramientas como las de mensajería y teleconferencias, agendas compartidas, documentos del área de trabajo compartida, herramientas de intercambio de información (foros electrónicos) y herramientas de workflow.

3.1.2 Georreferenciación

La georreferenciación [22, 23, 24] es un proceso que permite determinar la posición de un elemento en un sistema de coordenadas espacial diferente al que se encuentra. Existen por tanto dos sistemas de coordenadas: el sistema origen y el sistema destino. Este proceso es determinado con una relación de posiciones entre elementos espaciales en ambos sistemas, de manera que, conociendo la posición en uno de los sistemas de coordenadas es posible obtener la posición homóloga en el otro sistema. La georreferenciación se utiliza frecuentemente en los sistemas de información geográfica (SIG) para relacionar información vectorial e imágenes raster de las que se desconoce la proyección cartográfica, el sistema geodésico de referencia, o las distorsiones geométricas que afectan a la posición de los datos

Los SIG son una integración organizada de hardware, software y datos geográficos diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y

gestión. Permiten hacer un análisis exhaustivo del territorio en los ámbitos más diversos. Son herramientas versátiles, con un amplio campo de aplicación en cualquier actividad que conlleve un componente espacial.

El uso de herramientas como Google Maps o Google Earth ha implicado un salto cualitativo en cuanto a georreferenciación. Ya no se trata solamente de datos geográficos limitados a los especialistas de las geociencias y SIG. Ahora la georreferenciación tiene un impacto sociológico dado que se realiza sobre todos los contenidos sociales presentes en el mundo. Esto está acelerando la aparición de una web geosemántica. Del mismo modo, la masificación y evolución constante de la georreferenciación se ha visto impulsada por el uso de mashups en sitios Web 2.0, permitiendo la localización de contenidos digitales (vídeo, noticias, modelados 3D, etc.) en cartografía digital, dentro de lo que se ha venido a llamar la Información Geográfica Voluntaria.

3.1.3 Awareness

El término awareness se define en inglés como el sustantivo de “estar al tanto”, “tener conocimiento” o “estar informado”. En el contexto de los sistemas de trabajo colaborativo mediado por tecnología (Computer Suported Colaborative Work - CSCW), un grupo puede ser visto como un conjunto de individuos que interactúan directamente o por medio de artefactos compartidos y que se perciben a sí mismos como un grupo. En gran parte, estas percepciones se logran a través de mecanismos de awareness.

En el marco de un grupo de trabajo mediado por tecnología, el awareness entre los integrantes del grupo se mantiene mediante el acceso a información (por parte de todos los miembros del grupo) sobre cada uno de sus miembros; como por ejemplo: la ubicación de otros participantes en el espacio compartido (¿donde están trabajando?), sus acciones (¿qué están haciendo?), la historia de interacción (¿lo que lo han hecho?), y sus intenciones (¿qué van a hacer?).

3.1.4 Cloud Computing

Es un modelo de servicios escalables bajo demanda para la asignación y el consumo de recursos de cómputo [25]. La característica básica de este modelo es que los recursos y servicios informáticos, tales como infraestructura, plataforma y aplicaciones, son ofrecidos y consumidos como servicios a través de Internet sin que los usuarios tengan que tener ningún conocimiento de lo que sucede detrás.

Una de las principales ventajas para las organizaciones que deciden incorporar a sus actividades servicios prestados a través de Internet es la posibilidad de reducir sus gastos de personal técnico, instalaciones, software y, sobre todo, de tareas de mantenimiento; de esta manera el retorno de

la inversión es inmediato, ya que no es necesaria pre-instalación ni configuración alguna. Todo ello se realiza de manera fiable y segura, con una escalabilidad elástica, que es capaz de atender fuertes cambios en la demanda no previsibles a priori, sin que esto suponga un incremento en los costos de gestión.

Vale la pena destacar que a la hora de trabajar con Cloud Computing, existen tres modelos de servicio:

- **SaaS (Software as a Service):** El concepto de SaaS ha existido desde hace mucho tiempo, pero quizás en estos últimos años tomó un poco más de notoriedad. Básicamente se trata de cualquier servicio basado en la web como por ejemplo el Webmail de Gmail, Google Docs, Dropbox, entre otros. En este tipo de servicios se accede normalmente a través del navegador sin prestar atención al software en sí. Todo el desarrollo, mantenimiento, actualizaciones, copias de seguridad son responsabilidades del proveedor. Un ejemplo puede ser una organización que contrate una aplicación de correo electrónico para sus 30 empleados. La aplicación no podrá ser modificada por la empresa contratante ni sus usuarios a excepción de posibles configuraciones de usuario o personalizaciones que le permita el proveedor. La aplicación se encontrará alojada en las infraestructuras cloud del proveedor y el usuario no tendrá ningún control sobre las mismas. Algunos ejemplos son Gmail o Salesforce.
- **PaaS (Platform as a Service):** en este caso se contrata un servicio que le permite a uno alojar sus propias aplicaciones en una plataforma que dispone de herramientas de desarrollo para que el usuario pueda elaborar una solución. El proveedor ofrece el uso de su plataforma que a su vez se encuentra alojada en sus infraestructuras. Es un modelo que reduce de gran manera la complejidad a la hora de desplegar y mantener aplicaciones ya que las soluciones PaaS gestionan automáticamente la escalabilidad usando más recursos si fuera necesario. Los desarrolladores aún así tienen que preocuparse de que sus aplicaciones estén lo mejor optimizadas posibles para consumir menos recursos posibles. Algunos ejemplos son Google App Engine, Heroku, Windows Azure, entre otras.
- **IaaS (Infraestructure as a Service):** Con este servicio se tendrá mucho más control que con PaaS, aunque a cambio de eso el usuario debe encargarse de la gestión de la infraestructura. La principal diferencia es que el contratante de este servicio será el encargado de escalar sus aplicaciones según sus necesidades, además de definir aspectos tales como capacidad de procesamiento, de almacenamiento y / o de comunicaciones. Algunos ejemplos son Amazon Web Services o Kio Networks.

Modelos de servicios con ejemplos

Un usuario puede adoptar uno o más de estos modelos según sus necesidades. La decisión vendrá condicionada por dónde desea centrar sus esfuerzos y expertise: en las aplicaciones, en las plataformas y/o en las infraestructuras tecnológicas. Qué elementos le aportan valor a su negocio y por lo tanto quiere seguir implicado más de cerca en su evolución y cuales no le suponen un valor diferencial y prefiere contratar a un proveedor especializado.

3.1.5 Health IT (HIT)

Es el área de la informática que involucra el diseño, desarrollo, creación, uso y mantenimiento de los sistemas de información para la industria del cuidado médico [26,27]. Estos sistemas automatizados e interoperables reducen costos, mejoran la eficiencia y reducen errores, conjuntamente proveyendo una mejora en el servicio y cuidado del paciente. El componente central de esta área es el Registro Electrónico de Salud (EHR por sus siglas en inglés). Un EHR es el registro médico digital oficial del individuo el cual es compartido a lo largo de múltiples instalaciones y agencias. Los otros elementos esenciales de HIT son:

- **Registros Médicos Electrónicos** (EMR por sus siglas en inglés): es el registro de salud de un individuo dentro de las instalaciones del proveedor del cuidado de la salud.
- **Registro Personal de Salud (PHR):** que es el registro mantenido por el propio paciente.
- **Organización de la Información de la Salud Regional (RHIO):** se encarga de controlar las comunicaciones entre los otros elementos y unificarlos geográficamente.

3.1.6 Responsive Web Design

Responsive web design es un enfoque usado para solucionar los problemas de diseño para la gran diversidad de resoluciones y dispositivos existentes actualmente en el mercado [28, 29, 30]. Éste requiere centrarse en el contenido y en el cliente, en su experiencia de usuario. Por ejemplo, si el usuario deja de trabajar con su equipo de escritorio y quiere continuar navegando en la misma página web a través de una tablet o smartphone, no tiene que encontrar inconveniente alguno.

Esta técnica busca eliminar la necesidad de diseños diferentes y nuevos desarrollos para distintas resoluciones y por el contrario, sugiere que el mismo desarrollo debe dar soporte y responder a la necesidad del contexto sobre el que se esté ejecutando la aplicación, teniendo en cuenta parámetros como el tamaño de la pantalla, tipo de dispositivo u orientación. La página web debe de tener la capacidad de adaptarse a cada dispositivo, creando una solución única.

A nivel implementación, el Responsive Web Design tiene tres conceptos claves:

- Uso de Media Queries ofrecidos por CSS3. Estos van a permitir aplicar estilos condicionalmente teniendo en cuenta parámetros de la pantalla.
- Diseño web fluido. Se trata de layouts definidos en porcentajes que se ajustan a los anchos de la pantalla
- Elementos fluidos dentro de los layouts mencionados, como son las fuentes, imágenes o elementos multimedia.

Al crear un sitio con esta técnica, solo se va a necesitar de una unica version HTML y CSS que funcionara adecuadamente en cualquier tipo de dispositivo y resolución, eximiendo al desarrollador de preocuparse que su sitio web se vea idéntico en los diferentes dispositivos.

3.1.7 Desarrollo multiplataforma

Se refiere al desarrollo de aplicaciones móviles que pueden ser usadas en varias plataformas móviles. Es uno de los pilares fundamentales de una tendencia en crecimiento conocida en el mundo de los negocios como BYOD (Bring Your Own Device) [31, 32]. Este tipo de desarrollo posibilita que una compañía pueda desarrollar una aplicación en una plataforma nativa (iOS, Android, Windows Mobile, etc) o bien hacerla sobre un solo ambiente posibilitando que la aplicación pueda ser ejecutada en diferentes plataformas nativas.

Como la mayoría de las técnicas de desarrollo, posee ventajas y desventajas. Dentro de las ventajas podemos mencionar que estos tipos de implementaciones son muy útiles, ya que disminuyen costos y a su vez aumentan la velocidad en que las aplicaciones son desarrolladas. Adicionalmente, las herramientas multiplataforma móviles son por lo general fáciles de usar, ya que están basadas en lenguajes comunes de programación, incluyendo CSS, HTML y JavaScript.

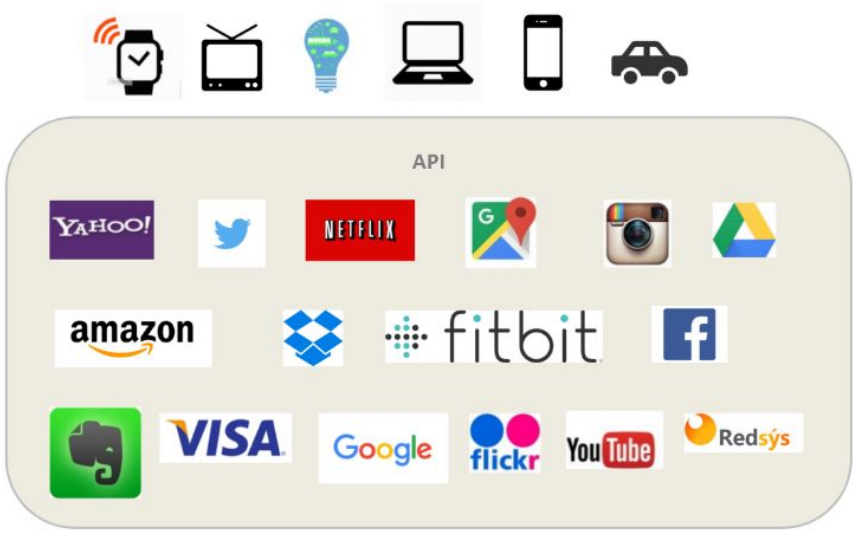
En cuanto a las desventajas podemos mencionar que los sistemas operativos móviles se actualizan frecuentemente, hecho que provoca que las aplicaciones desarrolladas tengan que actualizarse al mismo tiempo para ser compatibles con ellos. Además, los tiempos de renderizado pueden ser mayores ya que cada sistema operativo necesita un set de código separado.

La idea de este tipo de desarrollo es que una aplicación o producto de software pueda trabajar correctamente en más de un ambiente digital específico utilizando el mismo código.

3.1.8 API

Una API (**Application Programming Interface**) está conformada por un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas [33, 34]. En términos generales, las APIs hacen posible la interconexión de módulos y aplicaciones, facilitando el acceso a sus backends y permitiendo la reutilización de servicios. Es importante distinguir API de "servicio", siendo una API la manera con la que se interactúa y se consume dicho servicio. Haciendo una analogía con un ejemplo cotidiano, una API puede representar el enchufe de nuestra casa y el servicio la electricidad que nos proporcionada por la empresa distribuidora.

En la actualidad, hay una media de dos dispositivos conectados a Internet por persona, pero hacia el año 2020 se prevé que el número se duplique superando los cuatro por habitante del planeta. Todos estos dispositivos consumen APIs de manera proactiva de diferentes proveedores, algunos de los cuales pueden ser visualizados en la siguiente imagen.



Ejemplos de dispositivos y proveedores[6]

El mayor beneficio que presenta esta tecnología es permitir hacer uso de funciones ya existentes en otro software (o de la infraestructura ya existente en otras plataformas) para no estar, de

[6] https://www.paradigmadigital.com/wp-content/uploads/2016/02/1.png

alguna forma, reinventando la rueda constantemente, reutilizando así código que se sabe que está probado y que funciona correctamente. En el caso de herramientas propietarias (es decir, que no sean de código abierto), son un modo de hacer saber a los programadores de otras aplicaciones cómo incorporar una funcionalidad concreta sin por ello tener que proporcionar información acerca de cómo se realiza internamente el proceso.

3.1.9 API REST

En el modelo tradicional, el servidor web atiende las peticiones HTTP del cliente y devuelve una respuesta que puede ser un archivo HTML, un archivo (por ejemplo un archivo de estilos CSS o un script Javascript), o un objeto JSON. Para esto, mantiene en memoria una sesión por cada cliente conectado y toma decisiones en base a la misma por cada petición. Este modelo resulta costoso de escalar [35], ya que a medida que aumenta la cantidad de clientes, se requiere más y más memoria (generalmente RAM) para mantener las sesiones activas, lo cual muchas veces lleva a limitar la cantidad de clientes conectados al mismo tiempo y a desperdiciar potencia de cómputo.

La tendencia actual apunta a exponer servicios stateless (sin estado), es decir que entre dos llamadas cualesquiera, el servicio pierde todos sus datos y no podemos esperar que "nos recuerde". Esta es la clave de la arquitectura de servicios REST (**RE**presentational **S**tate **T**ransfer). Si quiero que un servicio REST me recuerde, el cliente debe pasarle quien soy en cada llamada, pudiendo hacerse mediante un usuario y una contraseña, un token o cualquier otro tipo de credenciales.

En una API REST la idea de "servicio" como tal desaparece. Lo que tenemos son recursos, accesibles por identificadores (URIs) utilizando una interfaz uniforme (Uniform interface). Sobre esos recursos podemos realizar acciones, generalmente diferenciadas a través de verbos HTTP distintos (GET, POST, PUT, etc.).

3.1.10 Aplicación móvil

Una aplicación móvil [36, 37], es una aplicación informática desarrollada para ser ejecutada a través de un dispositivo móvil inteligente, tablet u otro para el cual se desee implementar. Estas se encuentran en tiendas, por medio de las cuales son accedidas por el público que desee usarlas.

Es importante aclarar, que no todas las aplicaciones móviles tienen las mismas características, ni son del mismo tipo. Hay 3 tipos de aplicaciones móviles, nativas, web e híbridas, a continuación veremos las ventajas y desventajas de cada una.

Aplicaciones Nativas

Las aplicaciones nativas son aquellas desarrolladas bajo un lenguaje y entorno de desarrollo específico, como pueden ser Android, iOS o Windows, lo cual permite que su funcionamiento sea muy fluido y estable para el sistema operativo que fue creada. Pero como todo, tiene sus ventajas y desventajas.

- **Ventajas**
    - Utilización de los recursos tantos del sistema como del hardware.
    - Acceso completo al dispositivo
    - Mejor experiencia de usuario
    - La actualización es constante
    - Permite ser publicada en tiendas para su distribución.
    - En su mayoría, no necesitan estar conectadas a Internet para su funcionamiento

- **Desventajas**
    - Solo pueden ser utilizadas por un dispositivo que cuente con el sistema para el cual fue desarrollada.
    - El código del cliente no es reutilizable entre las distintas plataformas
    - Requiere de un costo para distribuirla en una tienda, y dependiendo el sistema, para el uso del entorno de desarrollo.

Aplicaciones web

Son aquellas desarrolladas usando lenguajes para el desarrollo web como lo son *html*, *css* y *Javascript* y un framework para el desarrollo de aplicaciones web, como por ejemplo *jquery mobile*, *Sencha, Kendo UI*, entre otros.. Se podría decir que este tipo de aplicaciones es muy usada para brindar accesibilidad a la información desde cualquier dispositivo, sin importar el sistema operativo, ya que solo se necesita contar con un navegador para acceder a esta.

- **Ventajas**
    - Pueden ser utilizadas desde cualquier dispositivo sin importar el sistema operativo.
    - Costo de desarrollo mínimo en comparación con las nativas.
    - Código reutilizable en múltiples plataformas
    - El usuario siempre dispone de la última versión
    - No requieren de ninguna aprobación para su publicación.
    - No necesitan instalación
- **Desventajas**
    - No pueden ser publicadas en plataformas para su distribución
    - No utilizan los recursos del sistema ni del dispositivo de manera óptima
    - Requieren de conexión a internet
    - Experiencia de usuario y tiempo de respuesta menor que una app nativa
    - Requieren de mayor esfuerzo en promoción y visibilidad

Aplicaciones híbridas

Por último están las aplicaciones híbridas, como su nombre lo indica tienen un poco de cada tipo de las aplicaciones ya nombradas. Este tipo de aplicaciones se desarrolla utilizando lenguajes de desarrollo web y un framework dedicado para la creación de aplicaciones híbridas, como por ejemplo *apache cordova* [38], *titanium appcelerator* [39]*, Steroids*, entre otros. La facilidad que brinda este tipo de desarrollo es que no hay un entorno específico el cual hay que utilizar para su desarrollo y la mayoría de las herramientas son de uso gratuito, también pudiendo integrarlo con herramientas de aplicaciones nativas

Hoy en día existen varios frameworks para este tipo de aplicaciones, siendo los más conocidos **Adobe Phonegap** [40], **Ionic** [41,42] y **Titanium Appcelerator**. Analicemos las diferencias entre cada uno:

- **Adobe Phonegap** es una distribución open source del framework Apache Cordova, que permite acceder a las APIs nativas del teléfono mediante una interfaz unificada para los distintos sistemas operativos. Provee un servicio de compilación en la nube (Phonegap Build), que nos permite exportar nuestro código html, css y javascript, a los formatos de las aplicaciones nativas de Android, iOS y Windows Phone sin tener cada compilador en nuestra máquina. Toda la vista como el look & feel y el comportamiento de la aplicación dependen exclusivamente del programador, ya que no provee funcionalidad ni componentes gráficos de ningún tipo.

- **Ionic** también es un framework open source que utiliza Apache Cordova como base, pero además provee una gran cantidad de componentes visuales que facilitan y aceleran mucho el proceso de construcción de la aplicación, ya que el desarrollador no tiene que preocuparse de programar comportamientos básicos de una aplicación móvil, como puede ser un menú desplegable con un swipe o un conjunto de tabs. Algo interesante de Ionic es que, usando los componentes que provee el framework, el look & feel de la aplicación cambia según el sistema operativo para el cual se compila, de manera que se adapte a los patrones básicos de interacción con el usuario, como lo haría una aplicación nativa. Además proporciona una SDK Javascript completa, construida utilizando el framework MVC AngularJs [17,18], para interactuar con los componentes visuales y los recursos nativos del teléfono (como pueden ser el gps, la cámara y el teclado), haciéndolo muy flexible a la hora de desarrollar y testear.

- Por último tenemos **Titanium Appcelerator**, una plataforma integral de desarrollo, cuenta con un IDE para organizar los proyectos y debuggear, provee notificaciones push y analytics, un editor gráfico de los componentes visuales de la aplicación y automatización de tests, todo out-of-the-box. Lo negativo es que es privativo y pago (no es económico), no existe una gran comunidad de desarrolladores como para otros frameworks open source, y además algunas acciones hay que realizarlas con markups específicos que no son nativos de tecnologías web.

- **Ventajas**

    - Uso de los recursos del dispositivo y del sistema operativo
    - El costo de desarrollo puede ser menor que el de una nativa
    - Son multiplataforma
    - Permite distribución a través de las tiendas de su respectiva plataforma.

- **Desventajas**

    - La documentación puede ser un poco escasa y desordenada
    - Experiencia de usuario y diseño visual propios de la aplicación web, no siempre relacionados con el sistema operativo sobre el que corre.

4. Elecciones para el desarrollo

Hoy en día existen multitud de lenguajes y frameworks para el desarrollo de aplicaciones, tanto para el lado del cliente como del servidor. A continuación haremos un análisis sobre las opciones y decisiones que tomamos en lo que respecta a herramientas de desarrollo.

4.1 Servidor

Para nuestro servidor decidimos implementar una API REST que solo devuelve objetos JSON, de esta manera el cliente está desacoplado y se encarga de cargar todo el contenido estático, archivos HTML, estilos CSS y scripts Javascript, y depende del servidor solamente para recuperar el contenido dinámico.

Para implementar una API REST existen frameworks muy conocidos y testeados que se utilizan en la industria hace bastante tiempo y se sabe que funcionan bien, como Spring en Java, que tan solo con un par de anotaciones podemos tener nuestro servicio REST disponible para ser consumido por cualquier cliente. Otro framework muy utilizado y que conocíamos es Symfony con PHP, el cual ha ganado mucho terreno en aplicaciones de gran escala y con arquitecturas complejas.

Si bien hubiéramos podido usar alguno de estos frameworks e ir por el camino de lo conocido y lo seguro, decidimos experimentar e investigar un poco otras herramientas más actuales y no tan maduras. Es por esto que decidimos utilizar Node.js [43,44], un entorno de ejecución montado sobre el motor Javascript V8 de Chrome que permite correr código Javascript del lado del servidor, junto con Express [43] que nos permite montar un servidor web muy fácilmente sobre la arquitectura de Node.js.

Además del desafío personal, Node.js está creciendo muy rápidamente y no puede dejar de evaluarse como una opción, en pocos años ya tiene más de 190.000 módulos desarrollados por programadores de la comunidad [45], un número que supera todo el repositorio de librerías de PERL en 20 años y las de Java Maven Central. Grandes empresas están confiando en Node.js para sus aplicaciones a gran escala, Netflix [46,47] potencia toda su interfaz gráfica con Node.js , Paypal [48] implementó una de sus aplicaciones con más tráfico con esta tecnología, y hasta la NASA [49] está usando Node.js en sus trajes espaciales!

Para la persistencia de datos se nos presentó un escenario similar, usar algún motor SQL con el cual ya hubiésemos trabajado, como podían ser MySQL o DB2, o experimentar con tecnología NoSQL que, en lugar de almacenar los datos en tablas con constraints, nos permite guardar

objetos JSON en colecciones que no tienen ningún tipo de restricción de tipos de datos ni de estructura. Así decidimos utilizar MongoDb [50, 51] como motor de base de datos NoSQL, principalmente porque hay mucha documentación y comunidad sobre esta tecnología, y porque utilizando la librería Mongoose [52] podemos modelar "clases" de objetos con propiedades y métodos que son persistidos en MongoDb y sobre las cuales podemos realizar operaciones CRUD a través de una interfaz muy simple (importante ya que las operaciones crudas sobre MongoDb son bastante genéricas y pueden tornarse complicadas).

De esta manera, tenemos la ventaja de una aplicación full-stack Javascript donde manejamos los mismos objetos JSON desde la base de datos hasta el cliente, sin ninguna transformación, por lo que no necesitamos un mapeador objeto-relacional ni ninguna otra abstracción, simplificando el proceso de desarrollo y debugging.

En las siguientes secciones analizaremos más en detalle cada una de las tecnologías utilizadas.

4.1.1 Node JS

Así como un browser permite correr código Javascript en el cliente, Node JS lo hace del lado del servidor. Es una plataforma open source construida sobre el **motor Javascript de Chrome V8,** destinada a la construcción rápida de aplicaciones de red veloces y escalables. Utiliza un modelo asincrónico **event-driven, non-blocking IO**, donde las operaciones abrir/leer/escribir sobre dispositivos y recursos (sockets, filesystem, etc), no bloquean al thread que hace el pedido, sino que solo lo marca para ser notificado cuando el nuevo dato o evento haya sido recuperado y esté disponible. Este modelo resulta ideal para aplicaciones real-time que corren a través de dispositivos distribuidos.

4.1.2 Express

Express es un framework web que funciona como un componente de Node, y que provee al programador de herramientas y estructuras que hacen que el desarrollo de aplicaciones sea más fácil y rápido. Ofrece un sistema de vistas unificado que permite el uso de cualquier template engine (como pueden ser Jade, Handlebars o Twig), además de una serie de utilidades para enviar respuestas en varios tipos de formatos, transferir archivos, rutear URLS, entre otras cosas. En comparación con otros frameworks como Django o Ruby on Rails, Express es extremadamente pequeño.

La filosofía que sigue es que cada aplicación varía enormemente en sus requerimientos e implementaciones, y un framework liviano permite construir exactamente lo que necesitás y nada más. Tanto la comunidad de Express como la de Node están enfocadas en achicar funcionalidad,

es decir más módulos que resuelvan una pequeña parte del problema en lugar de frameworks monolíticos.

4.1.3 MongoDb

MongoDB es un sistema de base de datos NoSQL multiplataforma orientado a documentos, de esquema libre. Esto significa que cada entrada o registro puede tener un esquema de datos diferente, con atributos que no tienen por qué repetirse de un registro a otro. Está escrito en C++, lo que le confiere cierta cercanía al bare metal, o recursos de hardware de la máquina, de modo que es bastante rápido a la hora de ejecutar sus tareas. Está licenciado como GNU AGPL 3.0, de modo que se trata de un software de licencia libre. Funciona en sistemas operativos Windows, Linux, OS X y Solaris.

Las características que más se destacan de MongoDB son su velocidad y su rico pero sencillo sistema de consulta. Se podría decir que alcanza un balance perfecto entre rendimiento y funcionalidad, incorporando muchos de los tipos de consulta que utilizaríamos en nuestro sistema relacional, pero sin sacrificar rendimiento.

En MongoDB, cada registro o conjunto de datos se denomina documento. Los documentos se pueden agrupar en colecciones, las cuales se podría decir que son el equivalente a las tablas en una BBDD relacional (sólo que las colecciones pueden almacenar documentos con formatos diferentes entre sí, en lugar de estar sometidos a un esquema fijo). Se pueden crear índices para algunos atributos de los documentos, de modo que MongoDB mantendrá una estructura interna eficiente para el acceso a la información por los contenidos de estos atributos.

Los distintos documentos se almacenan en formato BSON, o Binary JSON, una versión modificada de JSON que permite búsquedas rápidas de datos. BSON guarda de forma explícita las longitudes de los campos, los índices de los arreglos, y demás información útil para el escaneo de datos. Es por esto que, en algunos casos, el mismo documento en BSON ocupa un poco más de espacio de lo que ocuparía de estar almacenado directamente en formato JSON. Pero una de las ideas claves en los sistemas NoSQL es que el almacenamiento sea barato, y es mejor aprovecharlo si así se introduce un considerable incremento en la velocidad de localización de información dentro de un documento.

Aún así, los motores NoSQL no son siempre la mejor opción, se debe evaluar cada caso en particular y elegir considerando también sus desventajas [53,54]: al estar basados en documentos son muy permisivos y no imponen restricciones en la estructura de los datos como sí hacen los motores SQL, lo que puede llevar a errores de consistencia; no existe la posibilidad de realizar joins, la única forma de realizar esto es mediante la concatenación de múltiples consultas, en

ocasiones se tiende a denormalizar y tener redundancia de datos para mejorar la perfomance y evitar el overhead programático; no existen restricciones de integridad como claves foráneas, por lo que un documento puede hacer referencia a documentos que no existen, o puede ser eliminado dejando huérfanos a otros; el concepto de transacción de SQL donde se ejecutan todas las actualizaciones o no se ejecuta ninguna no existe en NoSQL, requieren control programático para alterar varios documentos manteniendo la consistencia; para aplicaciones productivas, un tema no menor es la madurez y el soporte de la tecnología, en el caso de MongoDb que entró al mercado en el 2009, aún no hay desarrolladores y administradores experimentados que puedan dar buen soporte ante un problema.

4.1.4 Mongoose

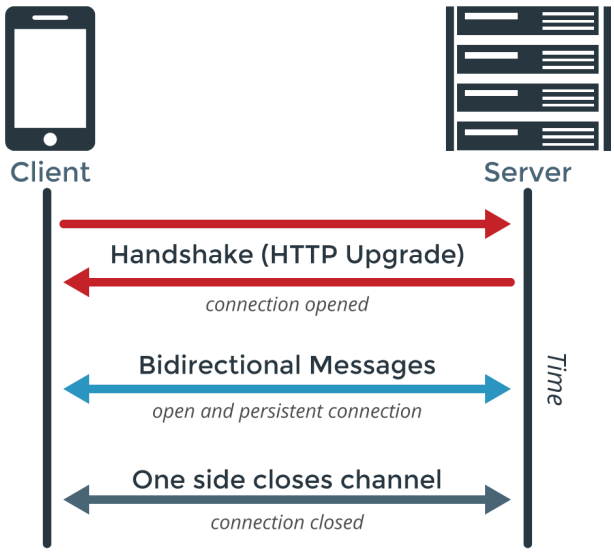
Mongoose es una librería de object data modeling (ODM) que provee un riguroso entorno para modelar la información, imponiendo las estructuras de objetos necesarias mientras mantiene la flexibilidad que hace poderoso a MongoDb.

Con Mongoose podemos definir esquemas (similares a clases) que contienen propiedades y métodos, que luego servirán para instanciar objetos y realizar operaciones sobre ellos. Algo interesante que provee out of the box es la posibilidad de tipar y poner restricciones sobre las propiedades de los esquemas, como si fuera un modelo SQL. De esta forma, podemos decir que una propiedad debe ser de tipo String, que no puede tener más de cierta cantidad de caracteres y no puede estar vacía al momento de persistirse, simplemente definiéndolo en el esquema y sin tener que realizar validaciones a mano.

4.1.5 Socket.IO

Desde el principio de la web, las comunicaciones HTTP eran dirigidas por el cliente hacia el servidor. Hace ya algún tiempo que existen tecnologías (conocidas como “Push” o “Comet”) que permiten al servidor enviar datos al cliente en el momento que hay nuevos datos disponibles. Una de las técnicas para simular una conexión iniciada por el servidor es el Long Polling, de esta forma el cliente abre una conexión HTTP con el servidor, el cual la mantiene abierta hasta que se envíe una respuesta. Existen otras técnicas como Flash Sockets, solicitudes XHR de varias partes y los denominados htmlfiles [55, 56].

Sin embargo, todas estas soluciones requieren muchas peticiones HTTP, lo que no las hace aptas para aplicaciones de baja latencia. En el 2011 se introdujo la especificación de WebSocket [57], que define una API para establecer conexiones persistente entre el cliente y el servidor, para que ambas partes puedan empezar a enviar datos en cualquier momento.



Comunicación utilizando WebSocket[7]

Para nuestra aplicación necesitábamos construir un chat que permitiese la comunicación instantánea entre los diferentes participantes de un tratamiento, para esto utilizamos Socket.IO. Socket.IO es una librería Javascript de comunicación bidireccional basada en eventos para aplicaciones web de tiempo real. Está basada en WebSockets, pero también abstrae distintas técnicas de comunicación en una sola API, incluyendo AJAX Long Polling y Flash Sockets. Permite que los desarrolladores envíen y reciban datos sin preocuparse por la compatibilidad cross-browser ya que determina cuál es la tecnología que se encuentra mejor soportada para cada caso. Resulta ideal para cualquier proyecto en el que tanto el cliente como el servidor puedan usar la misma librería.

4.2 Aplicación móvil

Teniendo en cuenta las ventajas y limitaciones de las distintas formas de desarrollo móvil (nativo, web e híbrido), optamos por hacer una aplicación híbrida, principalmente porque estábamos familiarizados con las tecnologías de base (Html, Css y Javascript), y porque combinan la versatilidad de una aplicación web junto a la posibilidad de acceder a los sensores y recursos nativos del dispositivo donde se ejecuta. Además, si bien ahora solo apuntamos a dispositivos con Android, nos posibilita reusar el código para llevarlo a otros sistemas operativos como iOS y Windows Phone.

Finalmente decidimos usar **Ionic** como framework principal, ya que toda nuestra experiencia con tecnologías web nos servía para arrancar el proyecto, y por la cantidad de componentes visuales e interactivos que provee, los cuales se suman a los servicios pre construidos de AngularJs [58,59].

[7] https://www.pubnub.com/wp-content/uploads/2014/09/WebSockets-Diagram.png

Además, cuenta con una comunidad gigante de desarrolladores de todo el mundo que dan soporte y bug fixing al proyecto.

A continuación analizaremos en detalle cada una de las tecnologías mencionadas anteriormente.

4.2.1 Ionic

Ionic es un framework de desarrollo destinado a la creación de aplicaciones móviles híbridas. Este tipo de aplicaciones son esencialmente pequeños sitios webs corriendo en un browser dentro de un entorno que tiene acceso a la capa de la plataforma nativa. Hay que pensar a Ionic como el framework front end que maneja todo el "look and feel" e interacciones de interfaz de usuario que la aplicación requiere.

A diferencia de un framework responsivo (como pueden ser Bootstrap o Foundation), Ionic incluye todos los elementos de interfaz de usuario nativos y capas que uno podría obtener con un SDK en iOS o Android.

Al ser Ionic un framework HTML5, necesita de un wrapper nativo como **Apache Cordova** para poder correr como una aplicación nativa. Cordova es el framework base que nos permite utilizar, mediante Javascript, las APIs nativas del teléfono para acceder a los sensores, información y las conexiones de red.

Como segundo pilar de Ionic está **AngularJs**, un framework Javascript MVC que nos permite estructurar de manera modular nuestro código, a la vez que provee servicios fáciles de utilizar para realizar invocaciones AJAX, utilizar el historial de navegación, realizar inyección de dependencias y testing. Una de las características principales es el **two-way data binding**, que permite que tanto el modelo como la vista estén sincronizados sin importar de qué lado haya cambiado la información, y sin necesidad de agregar código para interactuar con el DOM.

4.2.2 Bower

Cada vez que se quiere iniciar un proyecto web se necesita de librerías, plugins, fuentes, entre otros paquetes de terceros. Esto implica la necesidad de recurrir a un buscador, escribir el nombre del paquete que se desea, ir al sitio web o repositorio del proyecto, descargarlo, descomprimirlo y colocarlo en alguna ubicación de nuestro proyecto. Luego si se lanza una nueva versión y deseamos instalarla se debe repetir el procedimiento.

Esta tarea resulta muy tediosa, es por ello que se creó Bower [60, 61]. Este gestor que corre sobre Git [62], se encarga de **buscar, descargar y descomprimir** los paquetes que se requieran con un

simple comando en la consola (bower install). De igual manera para actualizar estas dependencias (bower update).

Bower está enfocado y optimizado para el *front-end* por lo que el manejo del árbol de dependencias es plano, en lugar de anidado como npm (el gestor de paquetes de node.js), esto evita problemas de versionamiento de las librerías además de una cantidad innecesaria de dependencias por versión.

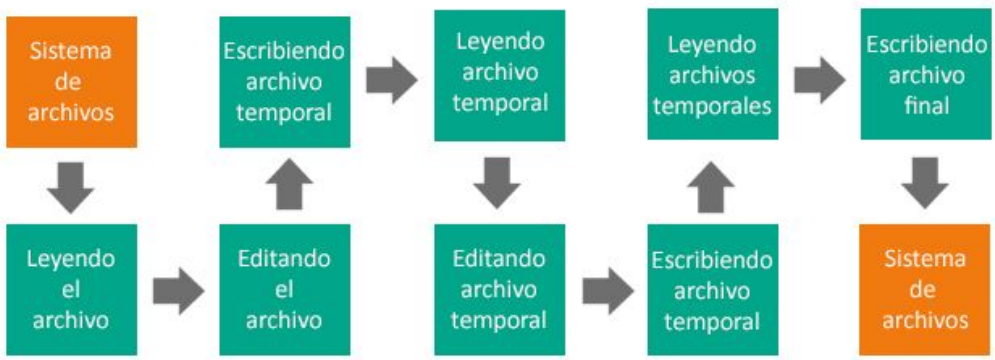
4.2.3 Automatizador de tareas

En el desarrollo web hay tareas que se hacen de manera repetitiva y manual, muchas veces en cada proyecto. como por ejemplo concatenar archivos Javascript, compilar estilos, minificar archivos, correr tests, recargar la aplicación cuando se hace algún cambio en un fuente, etc, etc.

Se debe poner en automático todo lo que se pueda, entonces un automatizador de tareas hace exactamente eso. Hoy en día existen 2 grandes automatizadores, Grunt y Gulp [63], ambos escritos en Javascript, corren sobre node.js y requieren crear tareas e instalar plugins. Entonces, qué diferencia a uno del otro.

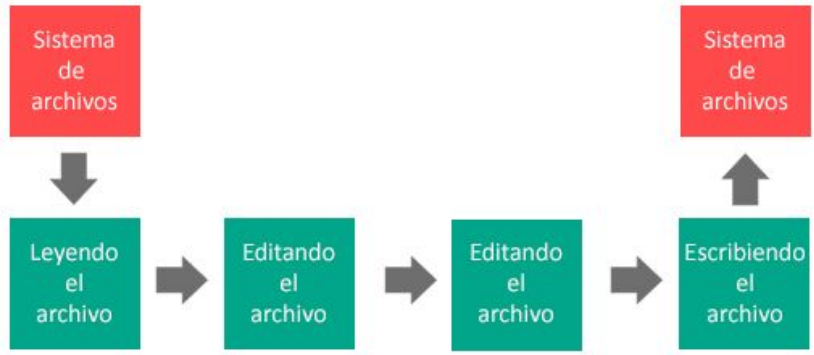
La diferencia principal entre Gulp y Grunt yace en cómo llevan a cabo internamente las tareas. Pongamos un ejemplo real para explicar esto, supongamos que queremos compilar un archivo SASS [64] a CSS y luego queremos minimizarlo para reducir su tamaño. Grunt maneja estas tareas usando archivos intermedios los cuales son operaciones I/O a disco. El archivo SASS se compila y luego se escribe en un archivo temporal, este archivo es usado por el autoprefixer [65] y luego el producto final se escribe a un archivo destino. Gulp realiza todas estas tareas en memoria, de manera que solo se hace una escritura a disco cuando todas las tareas fueron completadas. Las escrituras a disco son mucho más lentas, por lo que Gulp consigue una ventaja de velocidad en la ejecución de las tareas (tarda aproximadamente la mitad).

En la siguiente imagen veremos como Grunt.js manipula los archivos al realizar sus tareas:



Manipulación de archivos al utilizar Grunt.js[8]

Y en la siguiente se puede observar como lo realiza Gulp.js:



Manipulación de archivos al utilizar Gulp[9].js

Como podemos ver, aunque los 2 hicieron la misma tarea Gulp.js no escribe archivos temporales en el disco duro, realizó 4 operaciones contra 8 de Grunt.

Por otro lado, hace mucho más tiempo que se usa Grunt, por lo que la comunidad de usuarios y desarrolladores es notablemente mayor. Grunt tiene actualmente 37.000 descargas por día, contra 23.000 de Gulp, y cuenta con una base de más de 4000 plugins, mientras que Gulp sólo llega a un poco más de 1200 [63]. Aún así, ambas comunidades son muy activas y amables, por lo que es muy probable que para cualquiera de las 2 herramientas tengamos soporte y una respuesta a las inquietudes que puedan surgir.

Como último punto de comparación, podemos hablar sobre la sintaxis de cada una. Los argumentos son los siguientes: Gulp es un buen ejemplo de que “code over configuration” puede ser importante cuando la configuración se torna confusa; otros opinan que si bien Gulp es más fácil de leer, es más difícil escribir tareas ya que el piping puede ser confuso.

Para nuestro proyecto decidimos usar Gulp, principalmente por su velocidad y porque nos resultó más natural de leer y configurar que Grunt.

[8] https://frontendlabs.io/wp-content/uploads/2014/08/grunt-file-manipulation.png

[9] https://frontendlabs.io/wp-content/uploads/2014/08/gulp-file-manipulation.png

4.2.4 Genymotion

Al existir un gran número de dispositivos móviles en el mercado, los desarrolladores de aplicaciones se ven obligados a tener que dar soporte para cada uno de ellos. Es complicado y casi imposible tener a disposición muchos teléfonos, cada uno con diferentes tamaños de pantalla, resoluciones, características de hardware, versión de Android, etc. Es cierto que lo preferible es tener un dispositivo físico, ya que podremos probar nuestro proyecto en un teléfono/tablet real, pero en caso de no tenerlo, y a efectos de mayor practicidad uno se ve obligado a usar un emulador. Al ser el emulador AVD (Android Virtual Device) de Android excesivamente lento e ineficiente, nos dispusimos a buscar una alternativa mejor. Luego de analizar varias opciones, decidimos utilizar Genymotion.

Genymotion es un emulador que funciona en Windows, Mac y Linux, y aprovecha la arquitectura x86 para ejecutar de forma fluida y rápida distintos dispositivos Android, permitiendo probar todo tipo de aplicaciones y juegos [66, 67].

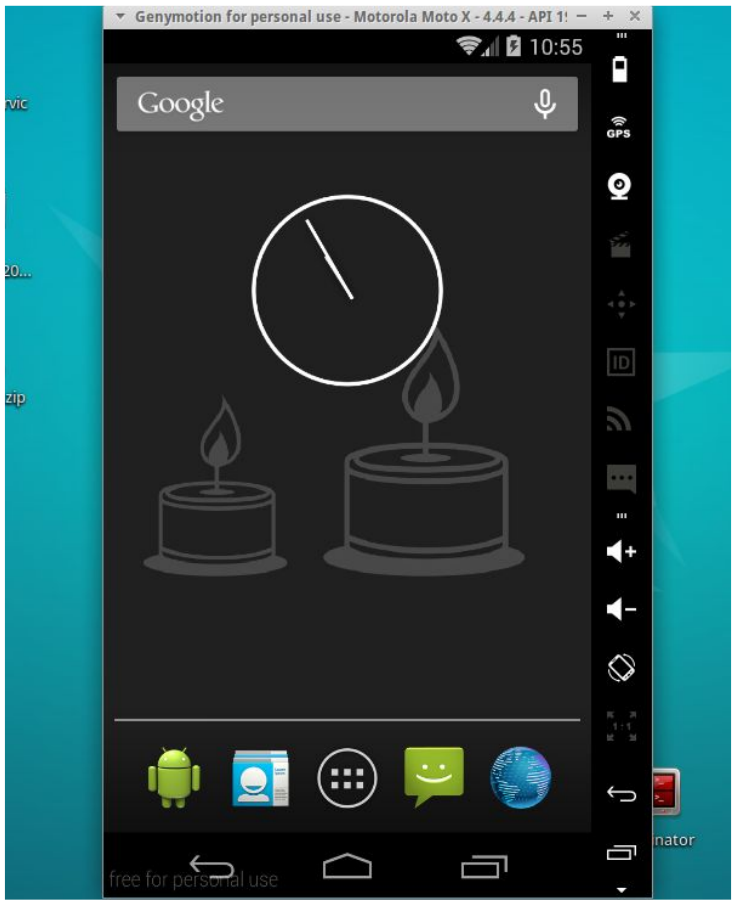


Imagen de Genymotion abierto en la computadora

Uno de los principales usos de Genymotion es facilitar el desarrollo de aplicaciones Android. Casi más de 900.000 usuarios registrados usan sus máquinas virtuales para crear aplicaciones usando Eclipse o intelliJ. Se integran perfectamente con el adb [68] (Android Debug Bridge), línea de comandos y los diferentes entornos de desarrollo.

Genymotion está basado en el uso de máquinas virtuales x86 optimizadas para correr sobre Virtualbox. Los creadores han conseguido crear una interfaz simple capaz de soportar distintas funcionalidades accesibles a cualquier usuario, sin olvidar a los desarrolladores.

Al descargarlo, nos permite instalar de manera rápida y sencilla cualquiera de las máquinas virtuales que emulan dispositivos como Nexus 4, Galaxy Nexus, Galaxy S4, Xperia Z, etc... para distintas configuraciones de Android 2.3, 4.1, 4.2, 4.3 y 4.4 , además de diferentes resoluciones de pantalla.

Entre todas las características que aporta el emulador se encuentran el uso de la conexión a internet del usuario, la simulación de ubicaciones GPS con el widget que facilita la búsqueda y el posicionamiento sobre un mapa, simulación de la cámara, estado de la batería, rotación del dispositivo, etc.

4.3 APIs

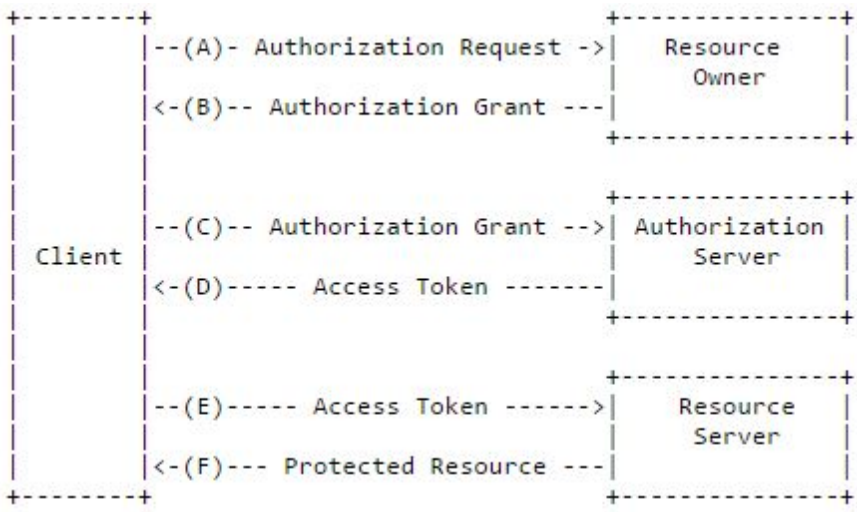
Las APIs integradas en TRACS son una parte muy importante de la aplicación ya que nos permitieron acceder a funcionalidades desarrolladas por terceros sin tener que volver a codificar una solución para el mismo problema. Esto nos sirvió para acortar tiempos y para mejorar la calidad del código, ya que sabemos que estamos utilizando herramientas que están testeadas y funcionan bien.
A continuación se encontrarán enumerados las APIs que utilizamos en el desarrollo de TRACS acompañados de una breve descripción de su funcionalidad y una especificación de la tarea que cumple en nuestra aplicación.

4.3.1 Google OAuth 2.0

OAuth 2.0 [69] (*Open Authorization*) es un framework que permite autorización segura de una API de modo estándar y simple para aplicaciones de escritorio, móviles y web.

OAuth 2.0 permite a un usuario del sitio **“A”** (proveedor de servicio) compartir su información con el sitio **“B”** (consumidor) sin compartir toda su identidad. Para desarrolladores de consumidores, OAuth2 es un método de interactuar con datos protegidos y publicarlos. Para desarrolladores de

proveedores de servicio, OAuth 2.0 proporciona a los usuarios un acceso a sus datos al mismo tiempo que protege las credenciales de su cuenta.



Flujo básico de OAuth 2.0

El flujo básico de OAuth 2.0 ilustrado en la figura anterior describe la interacción entre todos los roles y está compuesto por los siguientes pasos:

A. El cliente solicita autorización al dueño del recurso. La petición de autorización puede hacerse directamente al propietario (como se muestra), o preferentemente indirectamente a través del servidor de autorización como intermediario.
B. El cliente recibe un permiso de autorización, el cual es una credencial representando la autorización del propietario del recurso.
C. El cliente solicita un token de acceso autenticandose con el servidor de autorización y presentando el permiso.
D. El servidor de autorización autentica al cliente y valida el permiso. Si es válido emite un token de acceso

El flujo de autorización con Google OAuth es básicamente el mismo, y una vez que obtenemos un token de acceso podemos hacer peticiones autenticadas a las APIs que hayamos dado permiso, por ejemplo Google+, Drive o Gmail.

En TRACS el usuario se loguea directamente con una cuenta de Google (su email de Gmail), una vez que autoriza el acceso a la aplicación, mediante la API de Google+ obtenemos su información básica de perfil, nombre, email e imagen, la cual es enviada al servidor y es utilizada para crear un nuevo usuario en nuestra base de datos. Esto nos simplificó mucho la complejidad de un registro y el acceso a la aplicación, ya que hoy en día la mayoría de las personas tienen una cuenta de Google.

4.3.2 Google Drive

Drive es un servicio de almacenamiento en la nube que nos permite guardar de forma segura imágenes, videos, documentos de texto y todo tipo de archivos.
Google provee una API para interactuar con estos archivos y carpetas, pudiendo realizar consultas complejas, modificar metadata de los archivos, manejar permisos de acceso compartido y restringir la visibilidad.

En TRACS utilizamos la API de Drive para la versión web, donde el usuario logueado puede crear documentos privados o compartidos utilizando la interfaz de Google Docs, y luego visualizar un listado con todos los informes. Si bien para esto existe una API REST, Google sugiere utilizar la API Javascript para facilitar la ejecución de las operaciones de una forma segura. Lamentablemente, la API está pobremente documentada y es difícil encontrar las definiciones y métodos que se requieren para hacer cambios o consultas en archivos. Esto complicó un poco el desarrollo de algunas funcionalidades, particularmente la de modificación de permisos de un documento (para que todos los integrantes del equipo puedan verlo o sólo el propietario), y la consulta de carpetas existentes. Investigando la API REST, nos dimos cuenta que varios de los parámetros y nombres de los métodos coincidían, por lo que hicimos un trabajo de prueba, error y debugging para dar con los formatos de consulta para obtener el mismo resultado mediante Javascript.

4.3.3 Google Maps

Google Maps permite ver y usar una gran variedad de contenido en mapas, como datos de relieve, imágenes, fichas de empresas, indicaciones de tráfico, opiniones y otra información relacionada que proporciona Google y sus usuarios.

Mediante la API de Maps se pueden marcar puntos en un mapa, trazar caminos con indicaciones, desplegar información y mostrar capas personalizadas. Está muy bien documentada y hasta están especificadas todas las clases que componen un mapa (como pueden ser la clase Map, Marker, LatLng, TransitDetails), y que nos permiten configurar en detalle la información que se mostrará.

Para TRACS necesitábamos una forma de poder ubicar al paciente y mostrarle a los participantes dónde se encontraba. Utilizando el GPS del teléfono, cualquier persona en TRACS puede enviar un alerta georreferenciada de ayuda que llegará a todo el resto del equipo. Luego, a través de un mapa de Google Maps, se despliega un marcador que muestra la posición del paciente junto al camino más corto para llegar ahí y poder brindar ayuda.

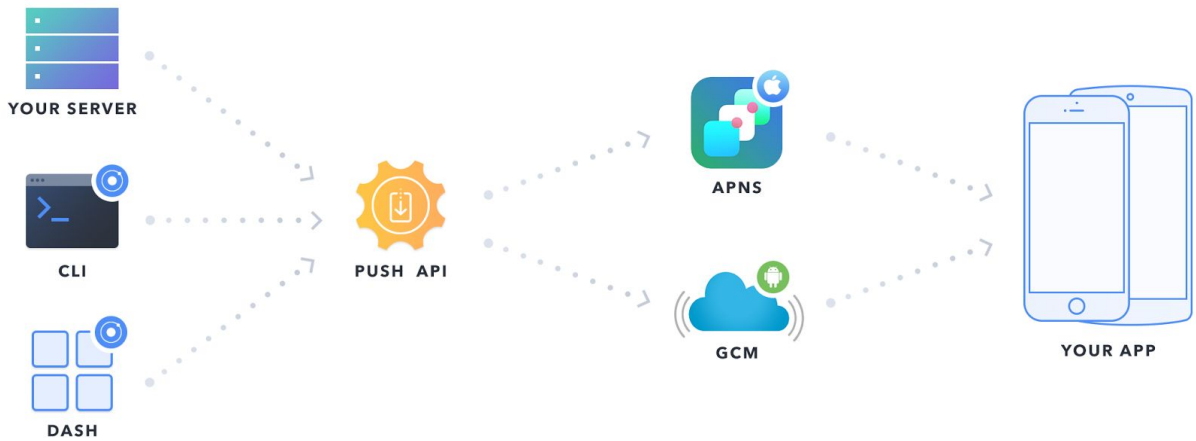
4.3.4 Push notifications (Ionic Push)

Las push notifications (o server push notifications), son el envío de información desde una aplicación servidor hacia otra aplicación cliente sin un pedido explícito (como ocurriría con pull notifications), ya que se originan directamente en el servidor.

En la computación móvil resultan un recurso muy importante e interesante, ya que no requieren que una aplicación esté abierta para poder recibir los mensajes, lo que permite que un smartphone reciba y muestre información de redes sociales, o alertas de mensajes de texto aún si la pantalla del dispositivo está bloqueada y la aplicación cerrada.

Son una forma muy conveniente de hablarle directamente al usuario para mantenerlo conectado y activo con la aplicación, ya que no son atrapadas por filtros de spam, ni olvidadas en una casilla de mensajes entrantes. Permiten promocionar productos y ofertas, notificar sobre eventos que al usuario le pueden llegar a interesar y hasta enviar contenido multimedia.

Ionic provee la plataforma Ionic Push para enviar push notifications a través de una API REST, la cual se configura a través de un dashboard al que podemos acceder registrándonos.



Arquitectura push notifications

La figura anterior ilustra la arquitectura de la plataforma y el flujo de datos para enviar push notifications mediante Ionic Push. Nuestra aplicación servidor se comunica mediante HTTP con la PUSH API de Ionic a la cual se le pasan todos los parámetros de la notificación, como puede ser el título, el mensaje principal, una imagen, un archivo de audio e inclusive un payload; luego, la API envía la notificación a los servidores de cloud messaging configurados, APNS (Apple Push Notification Service) para aplicaciones en iOS, GCM (Google Cloud Messaging) para aplicaciones en Android. En nuestro caso sólo configuramos GCM ya que apuntamos a dispositivos con Android, la configuración es bastante simple, basta con habilitar el uso de la API GCM y obtener una API KEY a través de la consola de desarrolladores de Google, que luego se utiliza para permitir que la PUSH

API de Ionic pueda enviar notificaciones en nombre de nuestra aplicación. Una vez enviada la notificación, los usuarios la reciben en sus teléfonos avisando sobre algún evento de la aplicación.

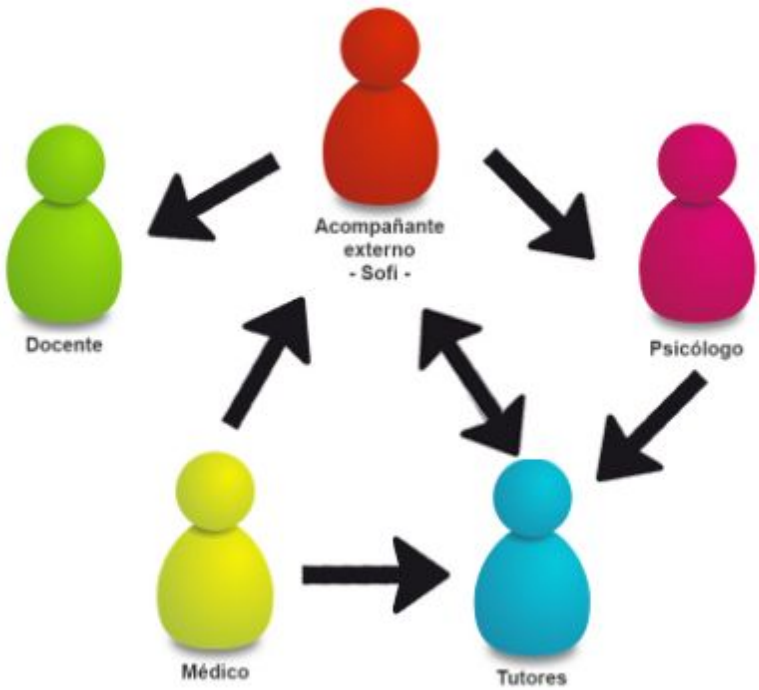
En TRACS, un usuario recibe una notificación push cuando un participante de algún equipo de tratamiento (sea el paciente o no) envía un alerta georreferenciada. De esta forma, todos los integrantes del equipo se enteran del evento y pueden responder inmediatamente ante una emergencia.

5. Relevamiento de requerimientos

Como filosofía de desarrollo y relevamiento adoptamos la metodología de Diseño Centrado en el Usuario (DCU) [70], cuya premisa es que para garantizar el éxito de un producto hay que tener en cuenta al usuario en todas las fases del diseño. El objetivo del DCU es la creación de productos que los usuarios encuentren útiles y usables; es decir, que satisfagan sus necesidades teniendo en cuenta sus características**.**

Partiendo de esta idea, desde el primer momento tuvimos una serie de reuniones con la Lic. en Psicología Sofía Lukaszczuk que trata a pacientes con discapacidad, como por ejemplo Autismo y TGD. Ella cumplió el rol de usuario final de la aplicación.

En la reunión inicial nos comentó cómo desarrolla actualmente su día a día laboral conjuntamente a todos los participantes intervinientes y las dificultades que el mismo presenta en términos organizativos, de falta de comunicación como también de pérdida de información. A grandes rasgos, nos describió que el circuito en el cual se transmite la información está compuesto por los siguientes actores e interacciones:



Circuito descripto por el usuario

Al mostrarle nuestra propuesta, se mostró muy a gusto con los beneficios que podría darle a su trabajo diario y a su vez en cómo podría incrementar el conocimiento del estado del paciente. A su

vez nos propuso funcionalidades que desde su punto de vista serían útiles dentro de la herramienta.

A partir de este momento empezamos a delinear las funcionalidades que podría llegar a incluir nuestra aplicación. Cabe destacar que para definir las mismas utilizamos la técnica de **Historias de Usuarios** o **User Stories [71]**, un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, usualmente un usuario [72]. Están compuestas por una descripción escrita que será utilizada para planificar y posteriormente descomponer los detalles con el dueño del producto, las conversaciones propiamente dichas con el dueño del producto y las pruebas que han de determinar si las historias están finalizadas o no [73].

La parte más importante de las historias de usuario es la conversación que se genera entorno a las mismas. Estas notas representan los requerimientos del cliente y típicamente se las escribe de la siguiente manera:

Como *<tipo de usuario>*, quiero *<algun objetivo>* para poder *<alguna razón>*

Uno de los beneficios principales de las Historias de Usuario es que pueden ser escritas en diferentes niveles de detalle. Es posible escribir historias que cubren múltiples funcionalidades. Estas historias más grandes son llamadas Épicas y dado que típicamente no pueden ser finalizadas en una iteración, se las divide en múltiples Historias de Usuario [73].

Si comparamos esta técnica con las formas tradicionales de especificación de requerimientos podemos encontrar las siguientes ventajas y desventajas:

Ventajas

- Enfatizan la comunicación verbal por sobre la escrita. La desventaja de documentar los requerimientos de software es que los clientes pueden obtener por resultado lo que el equipo de desarrollo ha interpretado, y no precisamente lo que el cliente necesita

- Las mismas pueden ser comprendidas tanto por los clientes y/o usuarios finales como por los miembros del equipo de desarrollo [74]. Los documentos de especificación de requerimientos de software contienen detalles técnicos que pueden resultar difíciles de comprender para los clientes y usuarios finales.

- Son compatibles con el desarrollo iterativo, hecho que posibilita que no sea necesario escribir todas las historias antes de comenzar un proyecto, sino que pueden escribirse un conjunto de historias, desarrollarse (codificarlas y probarlas) y luego continuar definiendo otro conjunto.

Desventajas

- En proyectos grandes con muchas historias de usuario, se hace más difícil establecer y entender las relaciones entre las historias. El autor sugiere armar matrices de trazabilidad

entre las historias implementadas en cada iteración y los casos de prueba creados para verificarlas.

- Si bien las historias de usuario fomenta la comunicación cara a cara y promueven el aprendizaje por medio de la participación activa de los clientes y/o usuarios finales, en proyectos grandes con múltiples equipos de desarrollo distribuidos geográficamente, si no se documenta cierto tipo de información el conocimiento puede perderse

Siguiendo los conceptos detallados en los párrafos anteriores, definimos una serie de historias de usuario para ser desarrolladas en la versión de TRACS contemplada en el presente trabajo. Como se podrá ver a continuación, cada funcionalidad está definida para un tipo de usuario específico. Actualmente la aplicación cuenta con tres perfiles, que más allá de que comparten la mayoría de las funcionalidades, tienen sus propias particularidades. Los mismos se pueden encontrar listados a continuación:

- **Usuario Administrador:** Se le asignará este tipo de perfil a la persona que inicialmente dé de alta al paciente. Lo que diferencia a este usuario de los demás tipos de usuarios es la capacidad de dar de baja participantes del tratamiento relacionado a los pacientes que haya creado.

- **Usuario pariente:** A la hora de añadir un participante al tratamiento se va a poder elegir la opción definirlo como pariente o no del paciente en cuestión. Sí es creado como pariente, la única diferencia que este participante va a tener con los demás es que no va a poder ser capaz de visualizar las opiniones hechas por los profesionales acerca del paciente, ubicadas en la sección de **Perfil Detallado** del mismo.

- **Usuario general:** Los participantes que cuentan con este perfil son los dados de alta una vez que el paciente fue creado y que además no son definidos como pariente del paciente. Van a contar con los mismos permisos que el Usuario administrador con la excepción de la baja de participantes del tratamiento.

Como	Quiero	Para poder
Usuario	**Administrar un log de notas**	**Registrar cualquier actividad del paciente que valga la pena recordar a la hora de hacer una evaluación formal mensual.**
Usuario	**Emitir alertas georeferenciadas**	**Dar aviso a los profesionales y/o tutores de alguna situación crítica relacionada al paciente**
Usuario no padre	**Crear informes compartidos**	**Poder realizar informes colaborativos entre los diferentes participantes del tratamiento**
Usuario no	**Crear informes privados**	**Poder confeccionar de una manera**

padre		**más cómoda y rápida los informes necesarios**
Usuario	**Administrar información relacionada a la Historia Vital de mi paciente**	**Poder tener conocimiento del tratamiento que realizaba el paciente antes de empezar a ser tratado por su grupo actual de trabajo**
Usuario	**Administrar información relacionada al diagnóstico de mi paciente**	**Poder conocer el diagnóstico actualizado del paciente con el que se está trabajando**
Usuario	**Administrar información relacionada a la medicación de mi paciente**	**Poder conocer la medicación que tiene recetada actualmente el paciente con el que se está trabajando**
Usuario	**Recibir alertas georeferenciadas**	**Conocer el lugar y momento en que se da una situación crítica y en caso de poder, acudir al lugar.**
Usuario Administrador /Usuario	**Administrar los participantes de un tratamiento**	**Recrear en la herramienta al equipo que está participando en el tratamiento de mi paciente**
Usuario	**Tener a disposición un canal de comunicación global/individual.**	**Comunicarme tanto de manera global o individual con alguno de los demás actores.**
Usuario	**Ver actualizaciones de otros perfiles**	**Ver en tiempo real los diferentes inputs que hagan los diferentes usuarios.**
Paciente	**Poder visualizar contactos de personas cercanas**	**Poder realizar llamadas rápidas y enviar mensajes predeterminados en caso de emergencia**
Paciente	**Emitir alertas georeferenciadas**	**Avisarles a los participantes de mi tratamiento mi ubicación en caso de una emergencia**
Usuario	**Configurar el orden en que aparecen los contactos cercanos de mi paciente**	**Facilitar el uso de la aplicación al paciente**
Usuario	**Ver y editar el perfil detallado de mi paciente**	**Conocer la información completa del paciente con el que estoy trabajando y modificar algún dato en caso de que sea necesario.**
Usuario no padre	**Escribir y visualizar opiniones sobre el estado actual de mi paciente y**	**Dar a conocer a los otros participantes mi opinión sobre el**

		paciente en cuestión y a su vez ver cual es la percepción de ellos acerca de él.
Usuario	**Ver y editar información de contacto de mi paciente**	**Tener disponible un número a donde llamar en caso de que haya algún inconveniente con el paciente.**

En base a algunas de estas funcionalidades desarrollamos un prototipo primitivo para ser mostrado al usuario en una segunda reunión, siguiendo el modelo iterativo del DCU (investigación y análisis de los usuarios, diseño y prototipado, y evaluación). De esta manera pudimos mostrarle un set reducido de funcionalidades, tales como el logueo utilizando su cuenta de Gmail mediante el protocolo OAuth2 y el ingreso de un nuevo paciente al sistema, para así obtener su feedback.

De esta demostración surgieron puntos importantes, como por ejemplo el modo en que tendría que mostrarse la información de pacientes, ingresos de información que tendría que realizar cada participante y la posibilidad de poder tener un espacio compartido en el cual se pudieran coordinar las sesiones entre el paciente y los profesionales intervinientes. Hicimos especial foco en la vista del paciente, para definir una interfaz que sea entendible y fácil de usar.

A continuación se pueden ver algunos modelos que fueron maquetados en base a los requerimientos del usuario. Los mismos fueron usados como base para comenzar con el desarrollo de todas las funcionalidades de TRACS, que podrán ser vistas en un apartado del capítulo.

Maquetas iniciales de pantallas a ser utilizadas por los participantes del tratamiento

El momento en que requerimos mayor asistencia fue a la hora de diseñar la pantalla que debe ser utilizada por el paciente, ya que había que buscar la mejor manera de incrementar la facilidad y simplicidad de uso. Sofía nos aconsejó enfocarnos en una interfaz con componentes grandes y llamativos representados por imágenes, evitar el uso de texto y lo más importante, posibilitar que las acciones disponibles no implicarán efectuar más de dos taps para poder realizarlas.

En base a esto, creamos el mockup que se puede apreciar en la imagen A lo largo del desarrollo, la misma fue mutando hasta su diseño final que puede ser visto en el Capítulo 7 del presente trabajo.

Maqueta inicial de pantalla a ser utilizada por el paciente

6. Desarrollo

Una vez realizada una primera iteración sobre los requerimientos, y teniendo ideadas las principales funcionalidades de la aplicación, pusimos manos a la obra en el desarrollo de TRACS.

Lo primero que hicimos fue definir la arquitectura de la aplicación. Para esto hicimos un diagrama simplificado de las partes macro de la herramienta, que nos permitió ver la interacción general del usuario con la aplicación, y de la aplicación con otros servicios.

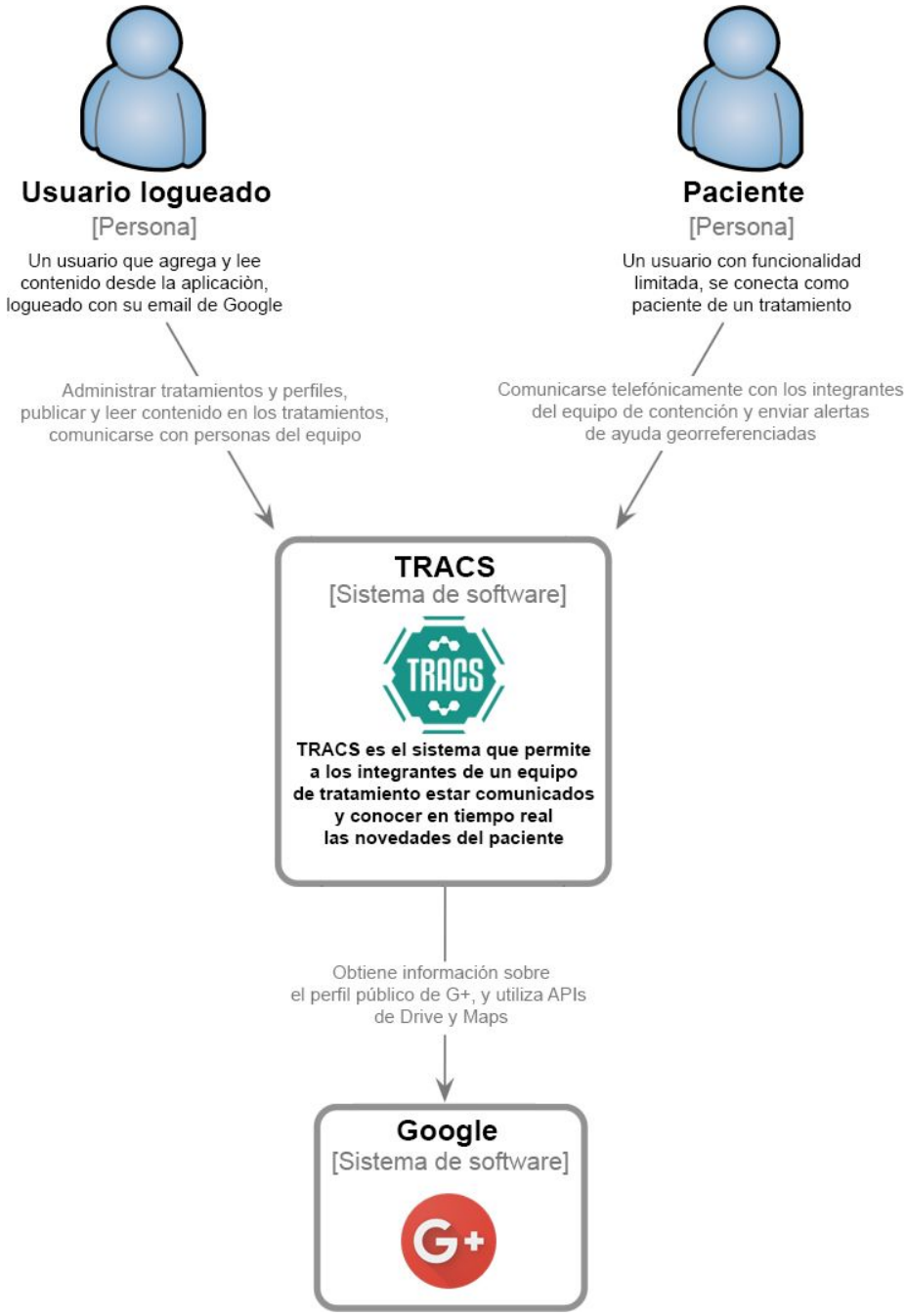


Diagrama simplificado de los componentes de la herramienta

Una vez que definimos la interacción con la aplicación y los servicios externos que usaría, empezamos a desglosarla en componentes más detallados, definiendo en este punto las tecnologías y herramientas con las que trabajaríamos.

Para ésto expandimos el diagrama anterior con el fin de visualizar fácil y rápidamente cómo íbamos a trabajar, y para estar enfocados y alineados en lo mismo.

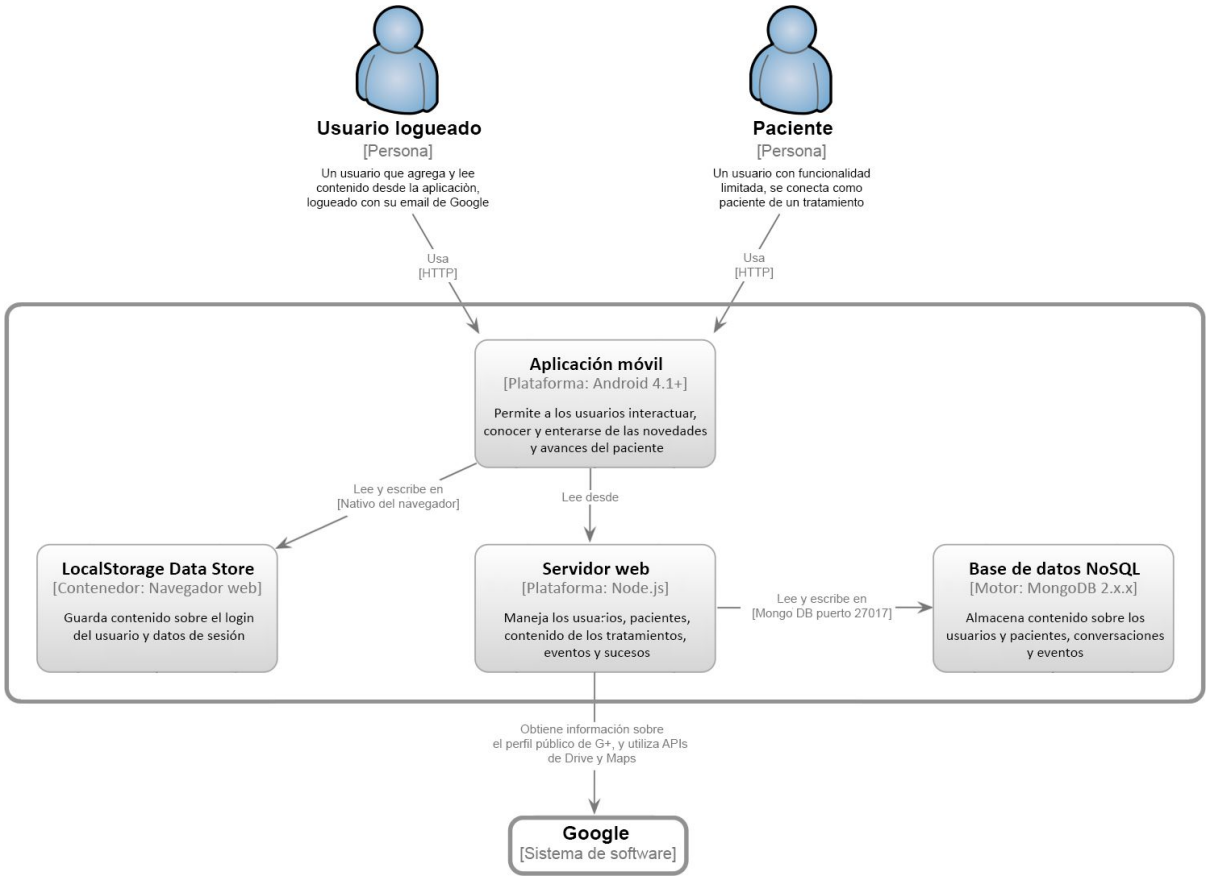


Diagrama extendido de los componentes de la herramienta

Así logramos definir la forma de comunicación entre los distintos componentes de la aplicación, y del usuario con la aplicación en sí.

Decidimos realizar una aplicación móvil, utilizando el framework Ionic para desarrollo híbrido, que sería compatible con versiones de Android iguales o superiores a 4.1 Jellybean. Ésta sería el principal punto de contacto del usuario con TRACS, mediante la cual, teniendo una conexión a internet, podría realizar la mayor parte de las funcionalidades planteadas, las de seguimiento, georreferenciación y comunicación.

Al ser una aplicación híbrida, el código corre dentro de un navegador embebido, el cual dispone de un pequeño almacén persistente de datos (LocalStorage [75]), que nos permite guardar hasta 5mb de información. De esta manera, la aplicación móvil lee y escribe sobre el LocalStorage datos simples sobre el usuario y la sesión en particular, y además recupera listas y objetos JSON más complejos desde un servidor web.

El servidor, montado sobre el ecosistema de Node.js y Express, es un simple servidor web REST que devuelve datos sobre un recurso solicitado, y persiste información en una base de datos NoSQL impulsada por el motor MongoDB. Expone varios endpoints a través de los cuales se

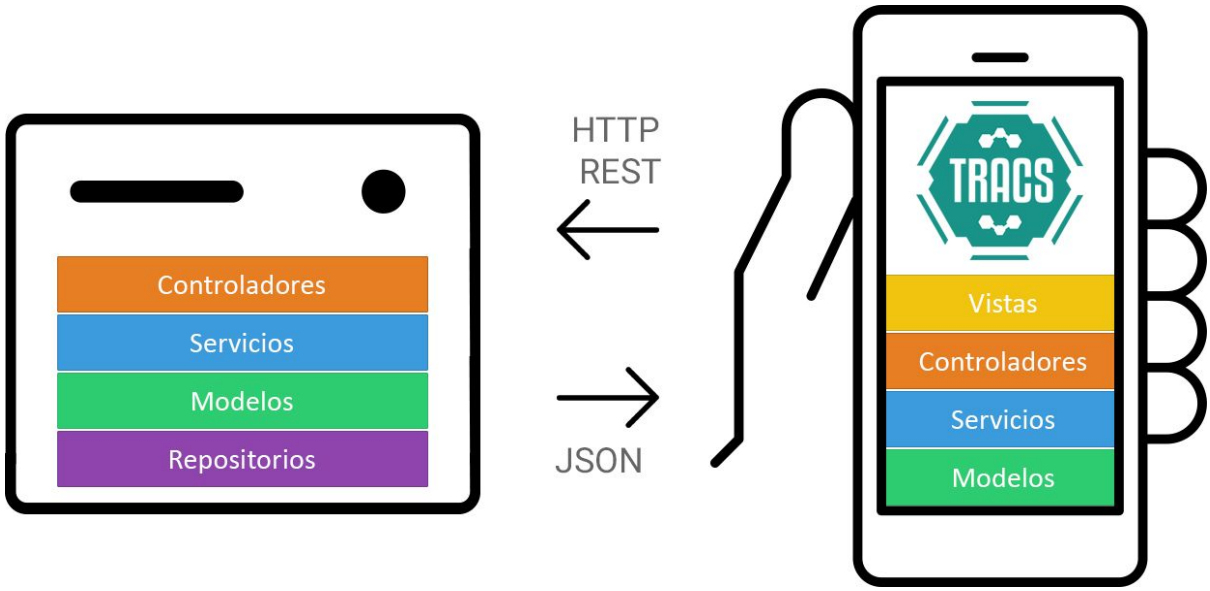
pueden realizar operaciones CRUD (Create, Retrieve, Update, Delete) sobre los pacientes y toda la información asociada a ellos, datos de perfil, información detallada sobre el diagnóstico, participantes del tratamiento, contactos de emergencia, mensajes del chat y notificaciones. También maneja información sobre los usuarios logueados, manteniendo referencias al perfil de Google+ y datos de contacto.

Para ambas aplicaciones, tanto el cliente como el servidor, se propuso una arquitectura siguiendo el patrón MVSC (**Model View Service Controller**), una variante del tan conocido MVC, que propone una arquitectura basada en capas que separan el código en función de sus responsabilidades o conceptos, con el fin de ayudarnos a crear aplicaciones con mayor calidad.

El **modelo** es una abstracción de los objetos que manejamos como componentes de la aplicación, por ejemplo un paciente o una notificación, encapsulan métodos y propiedades que representan el estado de ese objeto. Los objetos del modelo también son persistidos en la base de datos para después ser consultados y recuperados desde los servicios.

La capa de **servicios** contiene la lógica para modificar e interactuar con los objetos del modelo, es decir la "lógica de negocio", funciona como intermediario entre la capa de persistencia y los datos enviados desde los controladores para ejecutar alguna acción.

Los **controladores** actúan de nexo entre la entrada del usuario desde la vista, y los servicios a los cuales la envía para obtener algún resultado, no manipula directamente los datos. En el caso de nuestro servidor, al no proveer una vista, los controladores sólo actúan como routers, es decir, exponen los endpoints a los cuales el cliente puede conectarse para recuperar o modificar un recurso. Por el lado del cliente, los controladores funcionan como proveedores de datos para las vistas, y a la vez procesan la entrada del usuario enviándola al servicio que se comunicará con el servidor.



6.1 Comenzando los proyectos

Desde el inicio del proyecto nos servimos de una serie de herramientas para hacer el flujo de trabajo más fácil y dinámico, para poder compartir el código y tratando siempre de tener visibilidad sobre lo que estaba haciendo el otro para no pisarnos en la distribución de tareas.

Una vez que decidimos las tecnologías con las cuales haríamos el desarrollo, tanto del cliente como del servidor, nos dispusimos a instanciar ambos proyectos. Para esto aprovechamos las ventajas de **Yeoman** [76], un cliente por consola de scaffolding que nos ayuda a construir rápidamente aplicaciones con una arquitectura modular, sin preocuparnos por configuraciones manuales y facilitando el linting, testeo y minificación del código.

Yeoman provee un ecosistema de generadores. Un generador es básicamente un plugin que puede ser ejecutado mediante el comando "**yo**" para armar el esqueleto completo de un proyecto. El workflow de Yeoman comprende tres tipos de herramientas para construir una aplicación: la de scaffolding (yo), la de construcción y compilación (Gulp, Grunt, etc) y el gestor de paquetes y dependencias (como npm y Bower).

Para TRACS utilizamos 3 generadores diferentes: **generator-ionic-gulp** [77] para construir nuestro proyecto móvil, que provee las dependencias básicas para un proyecto Ionic y tareas preconfiguradas para fácilmente compilar y concatenar archivos de estilo SASS, minificar el código Javascript y construir y correr la aplicación en un emulador; también usamos **generator-angular** [78] para iniciar la aplicación web, ya que incluye todas las dependencias elementales para crear y correr un proyecto básico con AngularJs; por último utilizamos **express-generator** [79], que genera

rápidamente un esqueleto de aplicación “hola mundo” con express y sus dependencias básicas para poder construir un servidor REST.

Luego de crear los proyectos, lo primero que hicimos fue disponibilizar el código de cada uno en un VCS (Version Control System). Para esto elegimos **Git** [80, 81], pero antes de hablar directamente sobre Git, nos pareció interesante hacer una referencia general al concepto de control de versiones. Es un sistema que registra los cambios realizados sobre un archivo, o conjunto de archivos, a lo largo del tiempo, de modo que se puedan recuperar versiones específicas en el futuro. Este tipo de herramientas permite, entre otras cosas, revertir archivos o el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez un fragmento de código que puede estar causando problemas, etc. Hoy en día existen tres tipos de VCS [80]:

- **Sistemas de control de versiones locales:** Un método de control de versiones usado por mucha gente es copiar los archivos a otro directorio, indicando la fecha y hora en que lo hicieron. Este enfoque es muy común porque es muy simple, pero también tremendamente propenso a errores.

- **Sistemas de control de versiones centralizados:** Estos sistemas, como CVS, Subversion, y Perforce, tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos desde ese lugar central. Esta configuración ofrece muchas ventajas, especialmente frente a SCVs locales. Por ejemplo, dando la posibilidad de saber en que están trabajando los otros colaboradores del proyecto. Sin embargo, también tiene serias desventajas. La más evidente es el punto único de fallo, que representa tener un servidor centralizado que en caso de sufrir algún tipo de desperfecto técnico, durante ese tiempo nadie podría colaborar o guardar cambios versionados de aquello en lo que están trabajando.

- **Sistemas de control de versiones distribuidos:** En este tipo de SCV, los clientes no sólo descargan la última instantánea de los archivos, sino que replican completamente el repositorio. Así, si un servidor desaparece, y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo. Cada vez que se descarga una instantánea, en realidad se hace una copia de seguridad completa de todos los datos

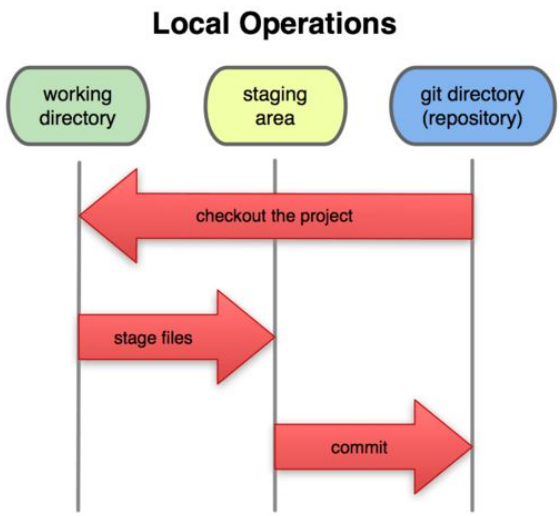
Git es un sistema de control de versiones distribuido cuyo objetivo es permitir mantener eficientemente una gran cantidad de código en proyectos que involucran a numerosos programadores. Es eficiente basándose en que cada programador almacena una copia completa

del repositorio en su máquina de forma local, incluido el historial de cambios. Esto implica que muchas de las operaciones realizadas sobre el código fuente no tienen lugar en la red, permitiendo que la velocidad de proceso dependa únicamente en los recursos locales.

Los archivos en Git pueden pasar por 3 estados diferentes:

- **Confirmado (committed)**: Los datos están almacenados de manera segura en la base de datos local
- **Modificado (modified):** Se ha modificado el archivo pero todavía no fue confirmado a la base de datos
- **Preparado (staged):** El archivo modificado fue marcado en su versión actual para que vaya en la próxima confirmación.

Esto nos lleva a las tres secciones principales de un proyecto:



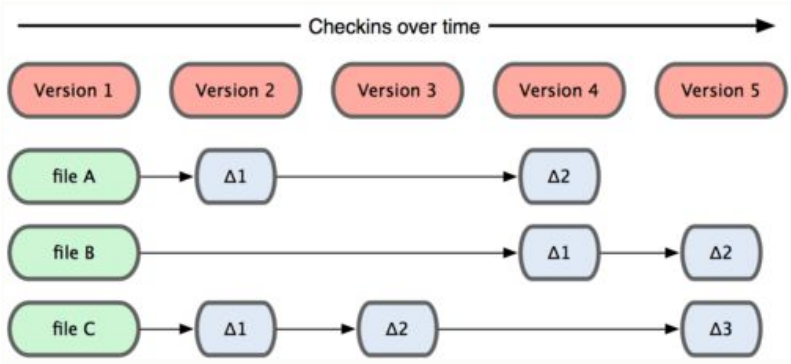
Directorio de trabajo, área de preparación y directorio de Git.

➔ **El directorio de Git:** es donde almacena los metadatos y la base de datos de objetos para el proyecto. Es la parte más importante, y es lo que se copia cuando se clona un repositorio desde otro ordenador.

➔ **El directorio de trabajo:** es una copia de una versión del proyecto. Estos archivos se obtienen de la base de datos comprimida en el directorio de Git, y se colocan en disco para que se puedan usar o modificar.

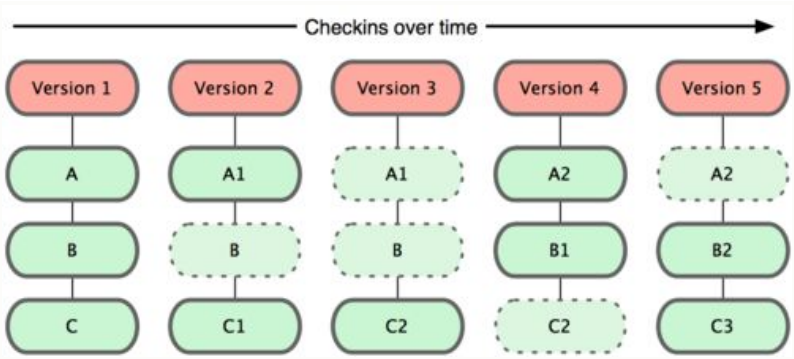
➔ **El área de preparación:** es un archivo sencillo, generalmente contenido en el directorio de Git, que almacena información acerca de lo que va a ir en la próxima confirmación.

La principal diferencia entre Git y cualquier otro SCV es la forma en la que modela sus datos. Conceptualmente, la mayoría de los otros sistemas almacenan la información como una lista de cambios en los archivos, modelan la información que almacenan como un conjunto de archivos y las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo.



Almacenamiento de datos como cambios en una versión de la base de cada archivo[10]

Git no no lo hace de esta manera, sino que realiza un conjunto de instantáneas de un mini sistema de archivos. Cada vez que se confirma un cambio, o se guarda el estado de un proyecto, se hace una foto del aspecto de todos los archivos en ese momento, y guarda una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, Git no almacena el archivo de nuevo, sólo un enlace al archivo anterior idéntico que ya tiene almacenado.



Almacenamiento de datos como instantáneas del proyecto a través del tiempo.[11]

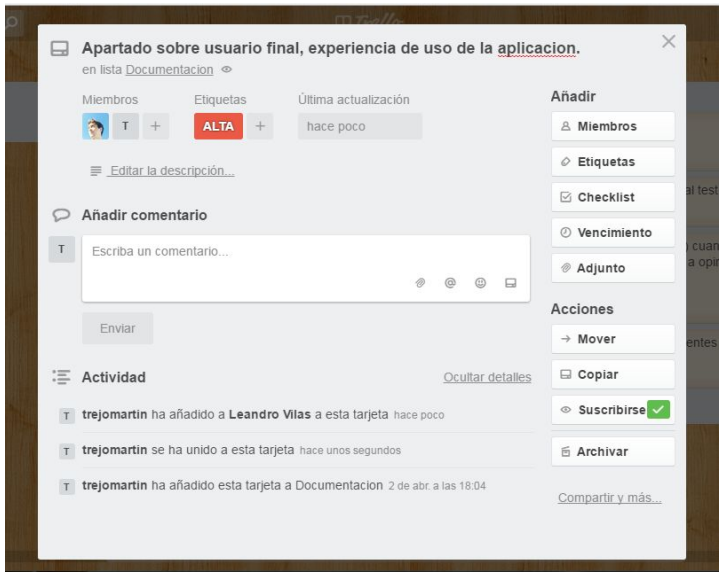
A la hora de la organización y seguimiento del proyecto decidimos utilizar **Trello**. Trello es una herramienta de colaboración cuyo objetivo primordial es la organización de proyectos. Esto lo realiza esquematizando dichos proyectos en lo que llama Tableros dentro de los cuales se pueden crear listas de diferentes índoles en las cuales se incluyen las tareas a realizar por cada integrante del equipo.

De esta manera pudimos llevar un control sobre las tareas que cada uno iba realizando y el estado de las mismas. La herramienta da la posibilidad de asignar prioridades, participantes, escribir

[10] https://git-scm.com/book/en/v2/book/01-introduction/images/deltas.png

[11] https://git-scm.com/book/en/v2/book/01-introduction/images/snapshots.png

comentarios y definir deadlines. En la imagen se puede observar un ejemplo de una tarea utilizada durante nuestro proyecto.



Ejemplo de tarea en Trello

A continuación se puede observar una imagen del tablero que utilizamos durante el desarrollo del proyecto, conjuntamente con un grupo de tareas que había generadas en ese momento.

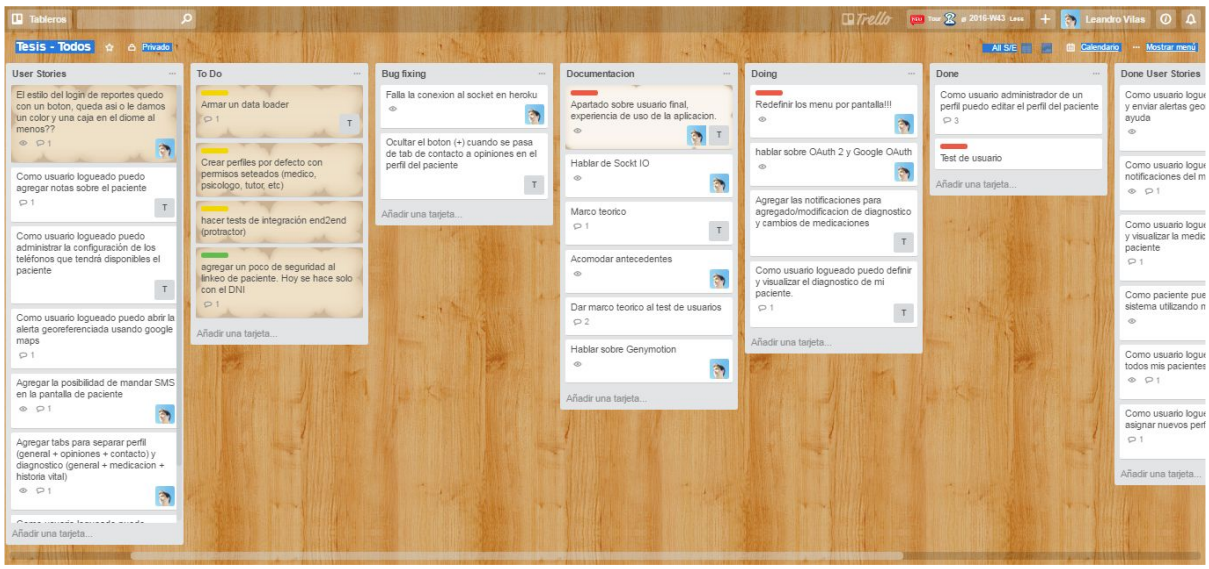


Imagen del tablero de Trello durante el desarrollo

6.2 Definición del modelo de clases

Antes de empezar a codificar TRACS planteamos un modelo de clases de manera de tener una guía y un esqueleto de los componentes de la aplicación y cómo iban a interactuar entre sí. En la

siguiente imagen se puede ver el diagrama UML de clases que representa la conformación final de nuestro modelo de datos. Para que sea más fácil de comprender, las clases tienen sus nombres y atributos en castellano, y a su vez tienen el nombre real en inglés de la implementación.

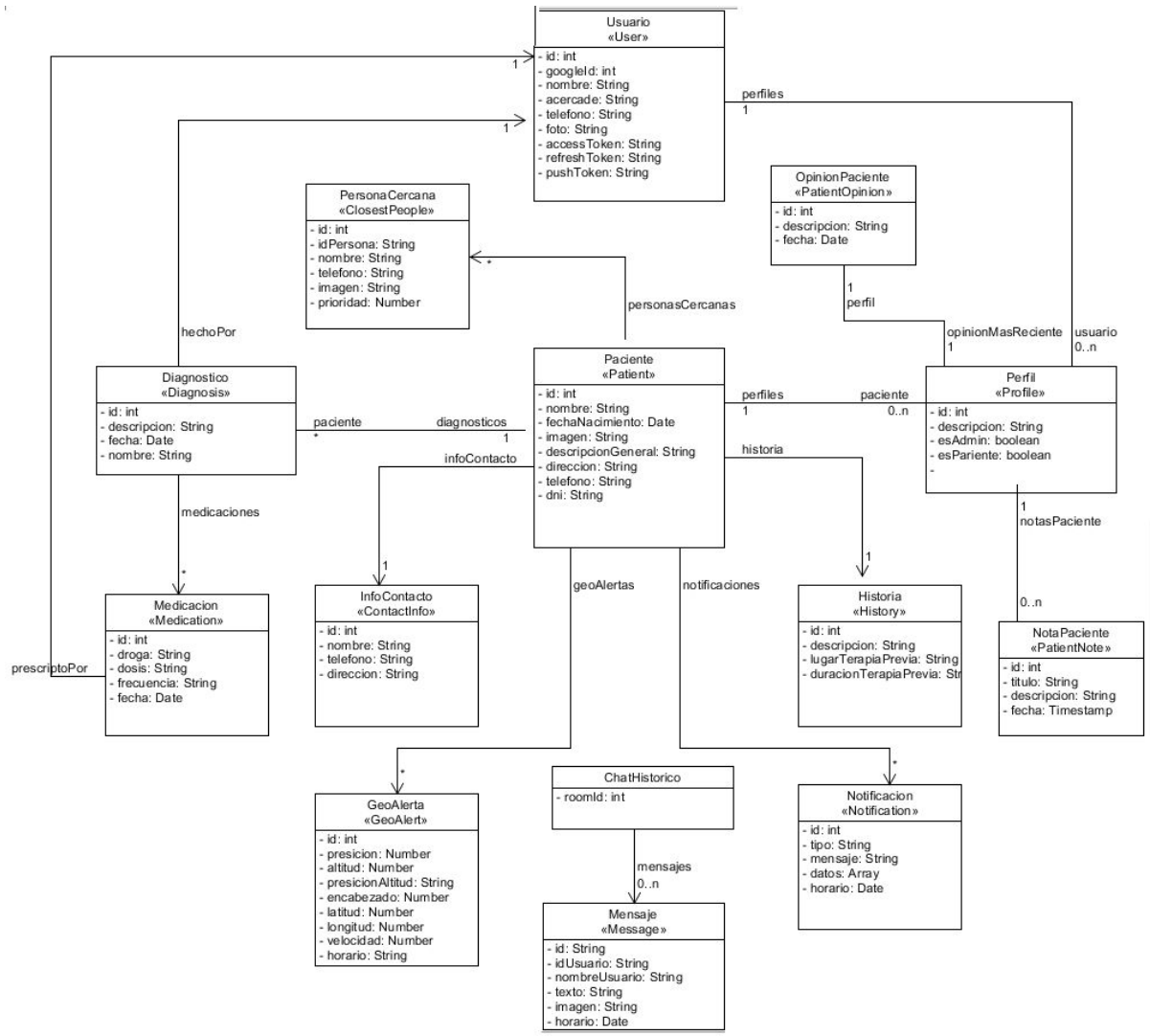


Diagrama de clases de TRACS

6.3 Ejemplos de código

Nos pareció bueno mostrar y comentar algunas secciones del código que pueden llegar a resultar interesantes de ver, ya sea por la estructuración del código, su funcionalidad o por la interacción con APIs.

Ejemplo de una invocación al servidor

Cuando un cliente realiza una petición al servidor, un ruteador interpreta la url invocada y, si puede resolver la ruta, ejecuta una acción asociada a un controlador.

```
78
79          /**
80           * Modifica el nombre y descripcion de un diagnóstico
81           * @param   {object}  updatedDiagnosis diagnostico modificado
82           * @param   {number}  diagnosisId      id del diagnostico a modificar
83           * @returns {promise} una promesa con el diagnostico modificado
84           */
85          function updateDiagnosis(updatedDiagnosis, diagnosisId){
86               return $http.put(DiagnosisEndpoint + "/updateDiagnosis/"+ diagnosisId, updatedDiagnosis).then(function (result) {
87                  return result.data;
88              }, function(error) {
89                  $log.error("Ocurrió un error al modificar el diagnostico del paciente " + diagnosisId, error);
90              });
91          }
92
```

Invocación del cliente

routers/DiagnosisRouter.js

```
1   /*jshint bitwise: false, camelcase: true, curly: true, eqeqeq: true, globals: false, freeze: true, immed: true, nocomma: true, newcap:
    true, noempty: true, nonbsp: true, nonew: true, quotmark: double, undef: true, unused: true, strict: true, latedef: true*/
2
3   /* globals require, module */
4
5   var express = require("express"),
6       router = express.Router(),
7       DiagnosisController = require("../controllers/DiagnosisController");
8
9       router.get("/diagnosisMedication/:id", DiagnosisController.getDiagnosisMedications);
10      router.get("/:id", DiagnosisController.getDiagnosis);
11      router.put("/updateDiagnosis/:id", DiagnosisController.updateDiagnosis);
12      router.put("/addDiagnosisMedication/:id", DiagnosisController.addDiagnosisMedication);
13
14      router.delete("/deleteDiagnosisMedication/:diagnosisId/:medicationId", DiagnosisController.deleteDiagnosisMedication);
15
16  module.exports = router;
17
```

Ruteador que interpreta la solicitud

El controlador que fue determinado para resolver la petición recupera los parámetros de invocación del método accediendo a las variables del objeto “req”, que encapsula un request http con toda su información, headers, body y variables pasadas por GET. Luego, envía la información al servicio adecuado para resolver la “lógica de negocios” de la acción solicitada. Si el servicio responde sin errores, el controlador resuelve con un estado 200 OK y el resultado del servicio; si ocurrió algún error, devuelve un 500 con la excepción capturada para que el cliente pueda tomar alguna acción en base al error.

controllers/DiagnosisController.js

```
/*jshint bitwise: false, camelcase: true, curly: true, eqeqeq: true, globals: false, freeze: true, immed: true, nocomma: true, newcap:
true, noempty: true, nonbsp: true, nonew: true, quotmark: double, undef: true, unused: true, strict: true, latedef: true*/

/* globals require, module, console */

var moment = require("moment"),
    DiagnosisService = require("../services/DiagnosisService"),
    MedicationService = require("../services/MedicationService");

var DiagnosisController = {};


DiagnosisController.getDiagnosis = function (req, res) { ··· };

DiagnosisController.getDiagnosisMedications = function (req, res) { ··· };

DiagnosisController.updateDiagnosis = function (req, res){
    "use strict";

    var diagnosisId = req.params.id;
    var updatedDiagnosis = req.body;

    DiagnosisService.updateDiagnosis(diagnosisId, updatedDiagnosis).then(function(diagnosis) {

        res.status(200).jsonp(diagnosis);

    }, function (err) {
        return res.status(500).send(err.message);
    });
};


DiagnosisController.addDiagnosisMedication = function (req, res){ ··· };

DiagnosisController.deleteDiagnosisMedication = function(req, res){ ··· };

module.exports = DiagnosisController;

```

Controlador que atiende la petición

Un servicio ejecuta la lógica y las acciones requeridas para resolver el pedido del cliente. Para esto manipula las entidades del modelo, que son las que tienen las propiedades y métodos propios de cada una, y las persiste o recupera desde la base de datos según sea necesario. Finalmente devuelve al controlador la respuesta que será enviada al cliente.

services/DiagnosisService.js•

```
/*jshint bitwise: false, camelcase: true, curly: true, eqeqeq: true, globals: false, freeze: true, immed: true, nocomma: true, newcap:
true, noempty: true, nonbsp: true, nonew: true, quotmark: double, undef: true, unused: true, strict: true, latedef: true*/

/* globals require, module, console */

require("../models/Diagnosis");
require("../models/Medication");

var mongoose = require("mongoose"),
    logger = require("../utils/Logger"),
    Diagnosis = mongoose.model("Diagnosis"),
    Medication = mongoose.model("Medication"),
    MedicationService = require("./MedicationService"),
    Patient = mongoose.model("Patient"),
    NotificationsService = require("./NotificationsService");

var DiagnosisService = {};


/* ··· */
DiagnosisService.getDiagnosis = function(diagnosisId){ ··· };


DiagnosisService.addDiagnosisMedications = function (diagnosisId, reqMedication) { ··· };

DiagnosisService.updateDiagnosis = function (diagnosisId, reqDiagnosis){
    "use strict";

    return Diagnosis.findOne({_id: diagnosisId}).then(function (diagnosis) {

        diagnosis.name = reqDiagnosis.name;
        diagnosis.description = reqDiagnosis.description;
        diagnosis.save();

        return NotificationsService.createNotificationForPatientId(diagnosis.patient, "Se ha modificado el diagnóstico",
        "patient.diagnosis.updated");

    }, function (error) {
        logger.error("No se pudo actualizar el diagnostico " + diagnosisId, error);
        return error;
    });
```

Servicio que resuelve la lógica de negocios

models/Diagnosis.js

```
/*jshint bitwise: false, camelcase: true, curly: true, eqeqeq: true, globals: false, freeze: true, immed: true, nocomma: true, newcap:
true, noempty: true, nonbsp: true, nonew: true, quotmark: double, undef: true, unused: true, strict: true, latedef: true*/

/* globals require, module */

var mongoose = require("mongoose");
var Schema = mongoose.Schema;

var DiagnosisSchema = new Schema({

    name: { type: String, required: true },
    description: { type: String, required: true },
    date: { type: Date, required: true },
    patient: {type : Schema.Types.ObjectId, ref : "Patient"},
    medications: [{type : Schema.Types.ObjectId, ref : "Medication"}],
    madeBy: {type : Schema.Types.ObjectId, ref : "User"}

});

module.exports = mongoose.model("Diagnosis", DiagnosisSchema);
```

Esquema de una clase realizada con Mongoose

Ejemplo de configuración de sockets para mensajería

Una de las funciones que no podíamos dejar de mencionar por el desafío que representó, principalmente porque no estábamos muy familiarizados con la programación de sockets, es la de la mensajería.

El servidor expone una serie de sockets que representan las “salas”, por cada paciente hay una sala, la global, donde todos los participantes del tratamiento pueden leer y escribir mensajes. Para lograr esto, el servidor escucha ciertos canales utilizando el método “**on**”, como puede ser el envío de un mensaje a una sala, en nuestro caso representado con el nombre “send:message”. Entonces cuando algún cliente envía un mensaje al canal “send:message”, mediante la función ”**emit**”, indica a qué sala pertenece ese mensaje y el servidor es el encargado de recibirlo, encolarlo y enviarlo al cliente para ser mostrado en pantalla mediante otro “emit” por el canal “message”.

```
152         function sendMessage(msg) {
153             roomCurrentMsg[msg.room].messages.push(msg);
154             socket.in(msg.room).emit("message", msg);
155         }
156
157         // Escucha en el canal "join:room" usado cuando u
158         socket.on("join:room", joinRoom);
159
160         // Escucha en el canal "leave:room" usado cuando
161         socket.on("leave:room", leaveRoom);
162
163         // Escucha en el canal "send:message" usado cuand
164         socket.on("send:message", sendMessage);
165
166     });
```

Ejemplo de configuración de sockets en el servidor

El cliente también está suscrito a este último canal, de manera que cuando recibe un evento de este tipo actualiza su lista de mensajes añadiendo el mensaje nuevo emitido por el servidor.

```
vm.sendTextMessage = function () {

    if (vm.message !== "") {

        var msg = {
            "room": vm.currentRoom,
            "userName": vm.currentUser,
            "userId": vm.user._id,
            "text": vm.message,
            "picture": vm.user.picture,
            "time": moment()
        };

        vm.messages.push(msg);
        $ionicScrollDelegate.scrollBottom();

        vm.message = "";

        SocketService.emit("send:message", msg);
    }
};

/**
 * This listens for messages sent by other users
 * that it's displayed in the view.
 */
SocketService.on("message", function (msg) {
    vm.messages.push(msg);
    $ionicScrollDelegate.scrollBottom();

});
```

Configuración de sockets del lado del cliente

Ejemplo de notificaciones push

Otra funcionalidad interesante para ver es la de notificaciones push, que hacen que cuando se ejecuta desde el cliente un alerta georreferenciada todos los participantes del tratamiento reciban en sus teléfonos una notificación avisando del evento.

```
34          /**
35           * Envia un alerta georreferenciada para un paciente
36           * @param   {object}  geoAlert un objeto con los datos de posicionamiento
37           * @param   {string}  patientId el id del paciente para el alerta
38           * @returns {promise} una promesa con el resultado del envio del alerta
39           */
40          function sendGeoAlert(geoAlert, patientId) {
41              // Transformación para evitar bugs de datos que no llegan al servidor (rarísimo)
42              var alertObject = {
43                  coords: { latitude: geoAlert.coords.latitude, longitude: geoAlert.coords.longitude },
44                  timestamp: geoAlert.timestamp
45              };
46              return $http.put(imAPatientEndpoint + "/sendGeoAlert/" + patientId, alertObject).then(function (result) {
47                  return result;
48              }, function(error) {
49                  $log.error("Ocurrió un error al enviar la alerta, intentalo de nuevo ", error);
50                  return error;
51              });
52          }
53
54          /**
55           * Recupera la posición actual del usuario
56           * @returns {promise} una promesa con la posición del usuario
57           */
58          function getMyPosition() {
59              var options = {
60                  timeout: 80000,
61                  enableHighAccuracy: true,
62                  maximumAge: 10000
63              };
64
65              return $cordovaGeolocation.getCurrentPosition(options).then(function (position) {
66                  return position;
67              }, function (error) {
```

Invocación al servidor con información de la posición geográfica

Cuando el servidor recibe el pedido de enviar un alerta georreferenciada, recupera los datos de posicionamiento enviados desde el cliente y utiliza la API de Ionic Push para disparar la notificación a los teléfonos de los participantes los cuales son identificados por un token.

```
var pushData = {
    tokens: [],
    payload: {
        "patient": {
            _id: patient._id,
            name: patient.name,
            picture: patient.picture,
            phoneNumber: patient.phoneNumber
        },
        "coordinates": {
            latitude: geoAlert.coords.latitude,
            longitude: geoAlert.coords.longitude
        }
    }
};
// Recupera los tokens push de casa usuario participante
// para mandarle una notificacion
pushData.tokens = patient.profiles.map(function(profile) {
    return profile.user.pushToken;
});

var options = {
    url: "https://api.ionic.io/push/notifications",
    method: "POST",
    json: true,
    headers: {
        // Ionic API KEY
        "Authorization": "Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
    },
    body: {
        "tokens": pushData.tokens,
        "profile": "tracs_push",
        "notification": {
            "title": "Alerta de paciente",
            "message": patient.name + " necesita ayuda!!",
            "android": {
                "message": patient.name + " necesita ayuda!!",
                "payload": pushData.payload
```

Procesamiento en el servidor del alerta georreferenciada

6.4 Puesta en producción

Una vez terminado el desarrollo tanto de la aplicación móvil como del servidor, quisimos disponibilizar la API en la nube para poder hacer los testeos de una manera más real, sin tener que levantar el ambiente de forma local. Para esto decidimos subir el servidor a un servicio de cloud computing que nos permitiera alojar la aplicación y que la expusiera bajo una url pública. Si bien teníamos varias opciones como Google App Engine, Heroku y Windows Azure, nos quedamos con Heroku porque ofrece un plan gratuito, y además es muy simple y fácil de usar, con un par de comandos teníamos nuestro servidor corriendo y disponible en la nube. Lo malo es que no ofrece almacenamiento gratuito de datos persistentes, por lo que buscamos otra opción para alojar la

información. El servicio que utilizamos para esto fue mLab, que permite crear bases de datos MongoDb y exponerlas en una url pública con usuario y contraseña.

De esta manera, el servidor alojado en Heroku quedó integrado con la base de datos en mLab [82], luego
conectamos nuestra aplicación móvil al servidor "productivo" y pudimos hacer los testeos teniendo sólo el celular y sin depender de nuestras computadoras.

Además, de esta forma la API de TRACS quedó expuesta en un dominio público (*https://gentle-tor-20108.herokuapp.com*) para que cualquier persona que quiera utilizar los datos proveídos por ella pueda hacerlo y pueda consumir los servicios para el desarrollo de nuevas aplicaciones.

A continuación analizaremos más en detalle las herramientas utilizadas.

6.3.1 Heroku

Heroku es una plataforma de cloud computing que permite construir, entregar, monitorear y escalar aplicaciones [83]. Como dice en su página oficial, provee una vía rápida para pasar de una idea a una URL, evitando los dolores de cabeza relacionados a aspectos de infraestructura. Ofrece un servicio ***PaaS*** (Platform as a Service) en donde actualmente se pueden desplegar aplicaciones desarrolladas en Ruby, Node.js, Java, Python, Clojure y Scala.

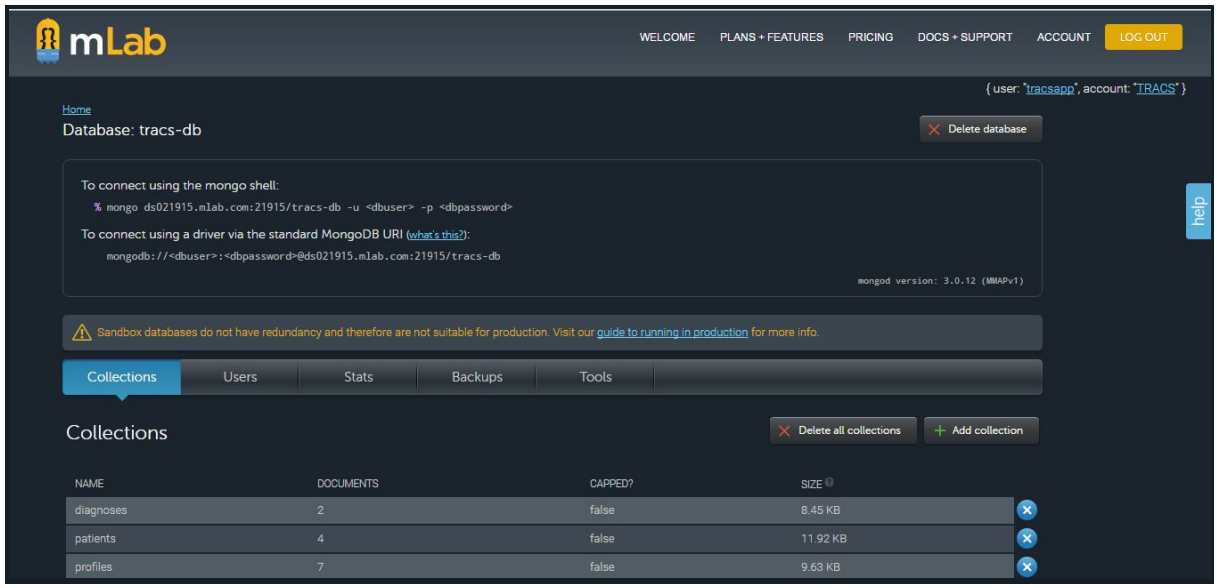
Inicialmente, a cada aplicación se le asigna un dyno, forma en que Heroku llama a sus servidores ligeros, donde se ejecuta el comando indicado por el usuario. El uso de dynos permite una gran escalabilidad para la aplicación, ya que en cualquier momento se puede contratar más o menos dynos según se requiera.

El deploy de la aplicación se realiza a través de Git. La primera vez puede resultar un proceso algo engorroso, pero una vez esté todo configurado, el resto de actualizaciones que se realicen al proyecto se suben con solo hacer un push al repositorio de Heroku mediante ***git push heroku master***. Con esto ya tendremos el código actualizado en el repositorio de Heroku y automáticamente se reiniciará la aplicación con la nueva versión.

6.3.2 mLab

mLab es un servicio de bases de datos en la nube que tiene como características principales el aprovisionamiento automático y escalado de bases de datos MongoDB, backup y recuperación, un servicio de monitoreo y alerta 24x7.

Hoy en día la plataforma Database-as-a-Service de mLab brinda soporte a cientos de miles de base de datos a través de AWS, Azure y Google y le permite a los desarrolladores enfocarse un 100% en los productos, dejando de lado las cuestiones operativas. Esta herramienta nos dio la posibilidad de subir y administrar nuestra base de datos a la hora de realizar las pruebas con la aplicación desplegada en Heroku, desligándonos totalmente de nuestro entorno local. A continuación se puede ver una imagen del administrador de base de datos que ofrece mLab:



7. Funcionalidades Desarrolladas

A lo largo de este capítulo se van a describir las funcionalidades con las que cuenta hoy en día la herramienta. Como punto de partida decidimos listar las historias de usuario que fueron generadas y desarrolladas antes de comenzar con las especificación funcional de las mismas.

7.1 Descripción de funcionalidades

A manera organizativa, dividimos la especificación de cada funcionalidad teniendo en cuenta sí forman parte de la versión **móvil** o **desktop**. A su vez, para el caso de las funcionalidades contenidas en la versión móvil, hicimos una distinción entre la **Vista Paciente** y **Vista Participante**.

➔ **TRACS Móvil**

- Vista Participante
- Vista Paciente

➔ **TRACS Desktop**

A su vez, decidimos incluir a modo orientativo referencias a los elementos más comunes con los que que uno puede encontrarse a la hora de interactuar con las diferentes pantallas de la aplicación:

Botones genéricos

ID Imagen	Acción	Ubicación en pantalla	Imagen
1	Añadir nuevo elemento	Cuadrante inferior derecho	
2	Submitir formulario	Cuadrante superior derecho	
3	Editar formulario	Cuadrante superior derecho	
4	Enviar alerta - Participante	Cuadrante superior derecho	
5	Ingresar a sala de chat grupal	Cuadrante superior derecho	

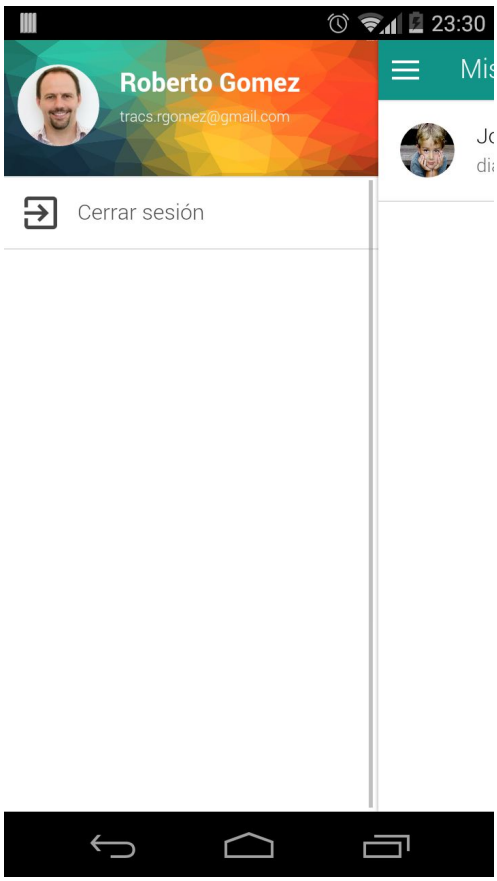
Menú lateral

El contenido del menú lateral va a ser contextual, es decir, va a depender de la pantalla en la que el usuario esté situado. En base a esto, van a ser posibles dos escenarios diferentes:

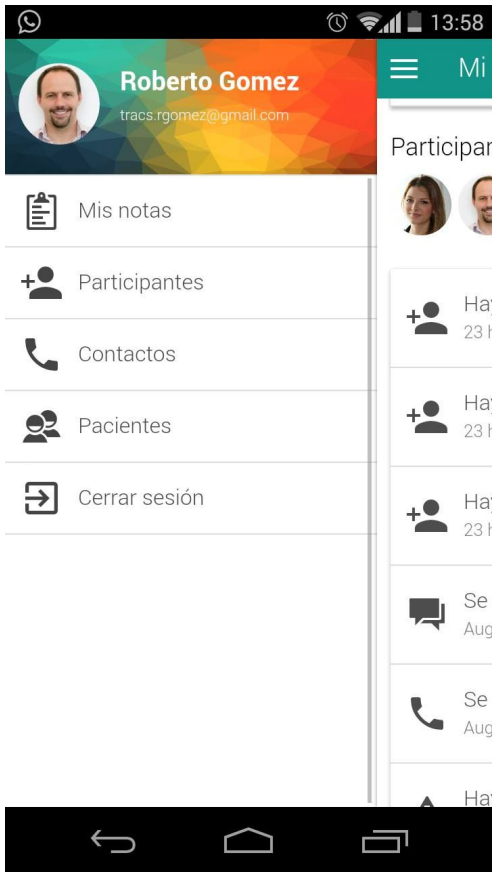
- ➔ Cuando el participante esté en la pantalla inicial donde se muestra el listado de sus pacientes, el menú solo va a contener la opción de Cerrar Sesión.

- ➔ En caso de que el participante esté situado en una sección diferente a la mencionada en el párrafo anterior, además de la opción de Cerrar Sesión, va a disponer de las siguientes:
  - ❏ **Mis Notas:** Acceso al administrador de notas personales
  - ❏ **Participantes:** Acceso al administrador de participantes
  - ❏ **Contactos:** Acceso al administrador de contactos de emergencia del paciente
  - ❏ **Pacientes:** Acceso al listado inicial de pacientes

Esto se puede ver reflejado en la siguientes imágenes:



Opciones del menú en pantalla inicial



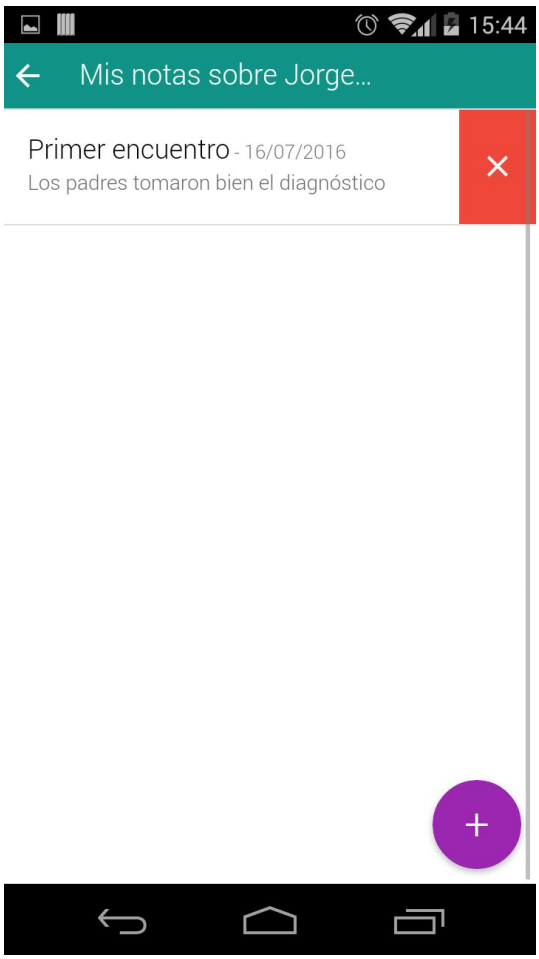
Opciones del menú en las demás pantallas

7.2 TRACS Móvil

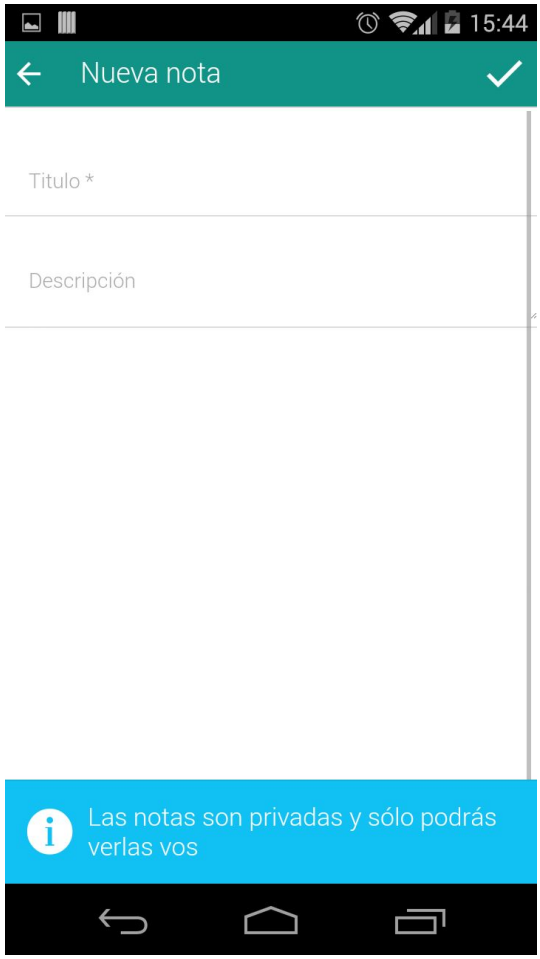
Vista Participante

Manejo de notas personales

Como	Quiero	Para poder
Usuario	**Administrar un log de notas**	**Registrar cualquier actividad del paciente que valga la pena recordar a la hora de hacer una evaluación formal mensual.**



Pantalla listado de notas



Pantalla alta de notas

La aplicación provee al usuario la posibilidad de mantener un log personal de notas. Esta funcionalidad es de gran ayuda ya que le permite al usuario tener un registro de eventos que se dan en el día a día con el paciente y que tal vez llegan a ser importantes para, por ejemplo, incluir a la hora de realizar un informe mensual sobre la evolución (en caso que el usuario sea el psicólogo) o mencionar en la siguiente sesión a la que el paciente asista (en caso de que el usuario sea un familiar). Es importante destacar que estas notas son personales, es decir que van a ser sólo visibles por el usuario que las ingresa. Para dar noción de esto al usuario, decidimos incluir un zócalo informativo en la pantalla de alta.

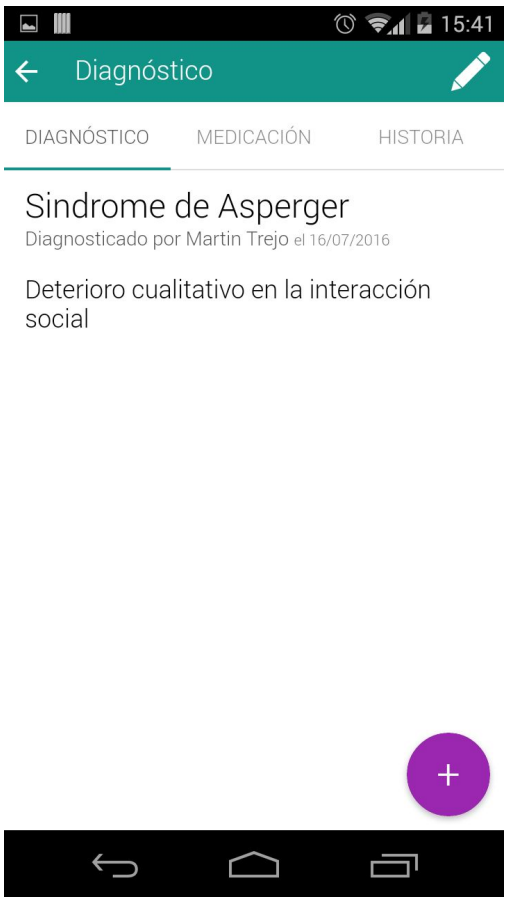
Para poder acceder a esta funcionalidad, habiendo ya seleccionado un paciente de la lista, se tiene que seleccionar la opción Notas desde el menú desplegable, que llevará al usuario a la sección en donde se encuentran listadas sus notas personales. Allí va a tener la posibilidad de elegir una nota para visualizar, agregar nuevas y borrar las ya existentes.

Para agregar nuevas notas, tiene que presionar el botón de agregar, y a continuación se mostrará la pantalla referida al alta de las mismas. Una vez que los campos mandatorios sean completados, el usuario deberá enviar la creación, finalizando de ese modo la adición de una nueva nota en su log.

A la hora de visualizar notas, solo se tiene que seleccionar alguna de las notas que se encuentren ya listadas. De este modo, el usuario tendrá visible todo el contenido de su nota y a su vez tendrá la posibilidad de editarla.

Sección de diagnóstico del paciente

Como	Quiero	Para poder
Usuario	**Administrar información relacionada al diagnóstico de mi paciente**	**Poder conocer el diagnóstico actualizado del paciente con el que se está trabajando**



Una de las funcionalidades de mayor utilidad que ofrece Tracs es la sección de Manejo de Diagnostico del paciente. El ingreso a esta sección se realiza desde el muro principal del paciente, oprimiendo el botón ***Diagnóstico***. Al ingresar uno se encuentra con tres tabs, una de las cuales pertenece al diagnóstico. En la misma los participantes del tratamiento pueden mantener información acerca del diagnóstico actual del paciente.

Se va a poder visualizar el nombre del trastorno diagnosticado, una descripción del mismo y por último, quien lo realizó. Cabe destacar que más allá de que la aplicación pueda almacenar más de un diagnostico, solo se mostrara el que fue añadido más recientemente.

Pantalla visualización de diagnóstico

Sección de medicación del paciente

Como	Quiero	Para poder
Usuario	**Administrar información relacionada a la medicación de mi paciente**	**Poder conocer la medicación que tiene recetada actualmente el paciente con el que se está trabajando**

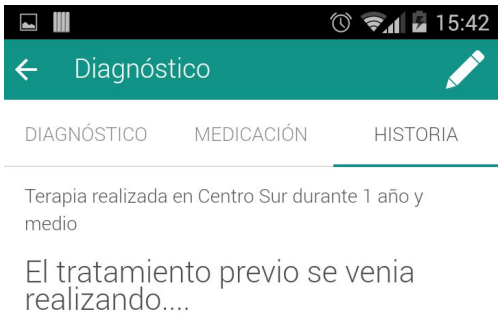
El manejo de la información relacionada a la medicación del paciente se va a realizar dentro de la sección de Diagnóstico, más específicamente en la segunda tab. Los participantes tendrán la capacidad de ingresar el nombre de la droga recetada, dosificación y fecha y persona que la prescribió.

Pantalla visualización de medicaciones

Sección de Historia del paciente

Como	Quiero	Para poder
Usuario	**Administrar información relacionada a la Historia Vital de mi paciente**	**Poder tener conocimiento del tratamiento que realizaba el paciente antes de empezar a ser tratado por su grupo actual de**

		trabajo

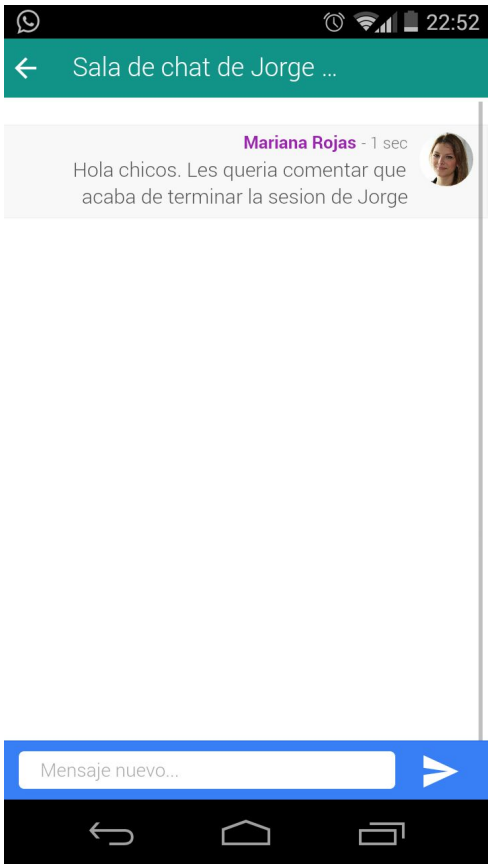


Por último en la sección de diagnóstico será posible visualizar la tab de ***Historia*** del paciente. En la misma los participantes van a ser capaces de conocer información acerca del tratamiento que el paciente venía realizando antes de comenzar a estar a cargo del actual equipo de trabajo. Se puede visualizar el centro al cual asistía, por cuanto tiempo lo hizo y una descripción general con los detalles que brindan valor agregado a la hora de encarar el nuevo tratamiento.

Pantalla visualización de Historia Vital

Canal de comunicación global (Chat)

Como	Quiero	Para poder
Usuario	**Tener a disposición un canal de comunicación global/individual.**	**Comunicarme tanto de manera global o individual con alguno de los demás actores.**



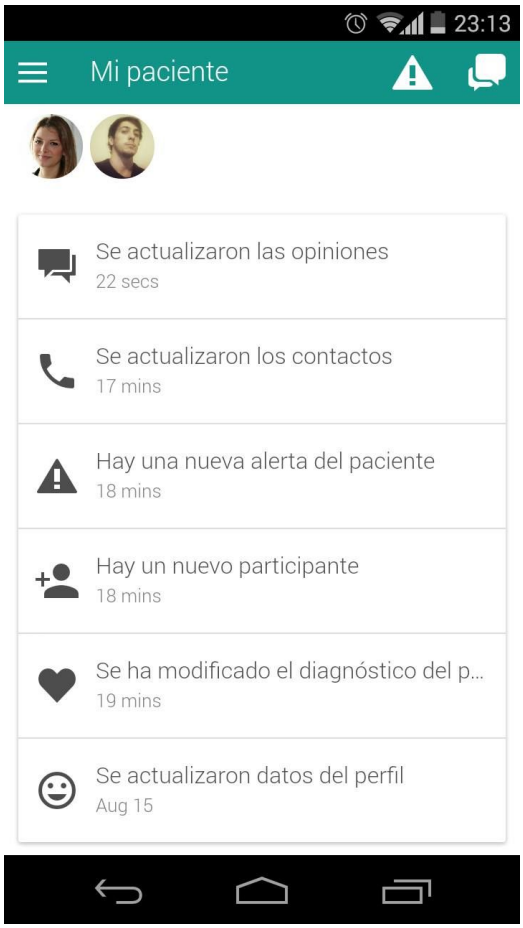
Pantalla de sala de chat

Con el objetivo de ofrecerle a los participantes la posibilidad de comunicarse entre ellos sin tener que dejar de usar la aplicación. Para poder cumplir con esa funcionalidad decidimos construir un chat global mediante el cual se posibilita la interacción entre los usuarios asignados al paciente en cuestión. El mismo lo desarrollamos utilizando Socket IO, herramienta ya mencionada en el capítulo xx.

Además de permitir la comunicación en tiempo real, los usuarios van a poder revisar los mensajes enviados en el pasado, ya que el chat construido tiene la funcionalidad de mostrar mensajes archivados.

Visualización de notificaciones en tiempo real

Como	Quiero	Para poder
Usuario	**Ver actualizaciones de otros perfiles**	**Ver en tiempo real los diferentes inputs que hagan los diferentes usuarios.**



Listado de actualizaciones de perfiles en el muro del paciente.

Dentro de la aplicación hay determinadas acciones que provocan la generación de notificaciones. Es gracias a esta configuración que los usuarios tienen la posibilidad de ver en tiempo real las actualizaciones que otros participantes realicen en la información referida al paciente.

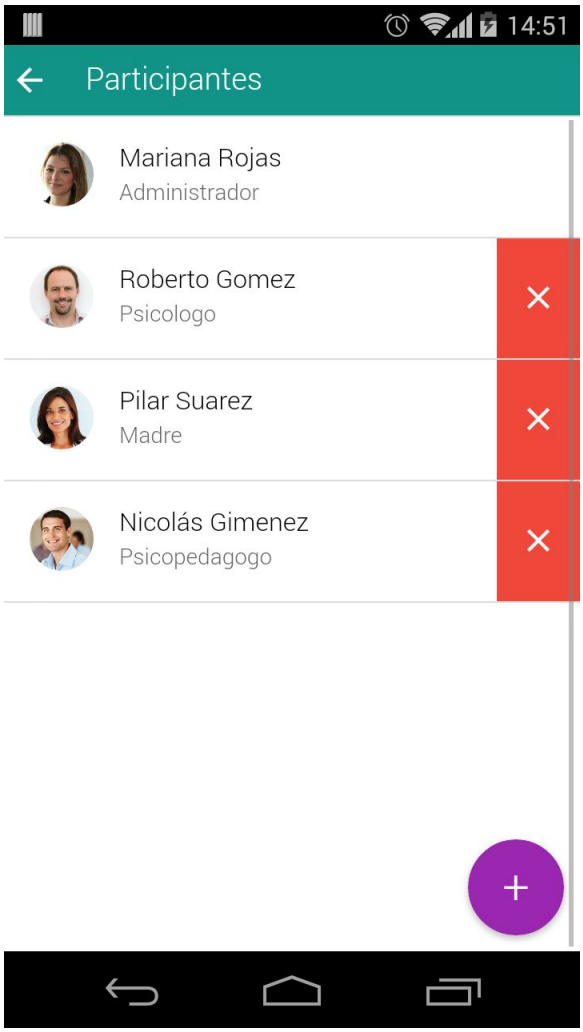
Cuando el usuario seleccione una de estas notificaciones, será llevado a la pantalla que se condice con la acción realizada. A continuación se listaran los eventos que disparan las notificaciones y la notificación que cada uno tiene asignado:

- Modificar datos del perfil del paciente: ***Se actualizaron datos del perfil***
- Agregar una opinión: ***Se actualizaron las opiniones***
- Actualizar la información de contactos: ***Se actualizaron los contactos***
- Alertas georeferenciadas: ***Hay una nueva alerta del paciente***
- Agregar un nuevo participante: ***Hay un nuevo participante***
- Actualización del diagnóstico: ***Se ha modificado el diagnóstico del paciente.***

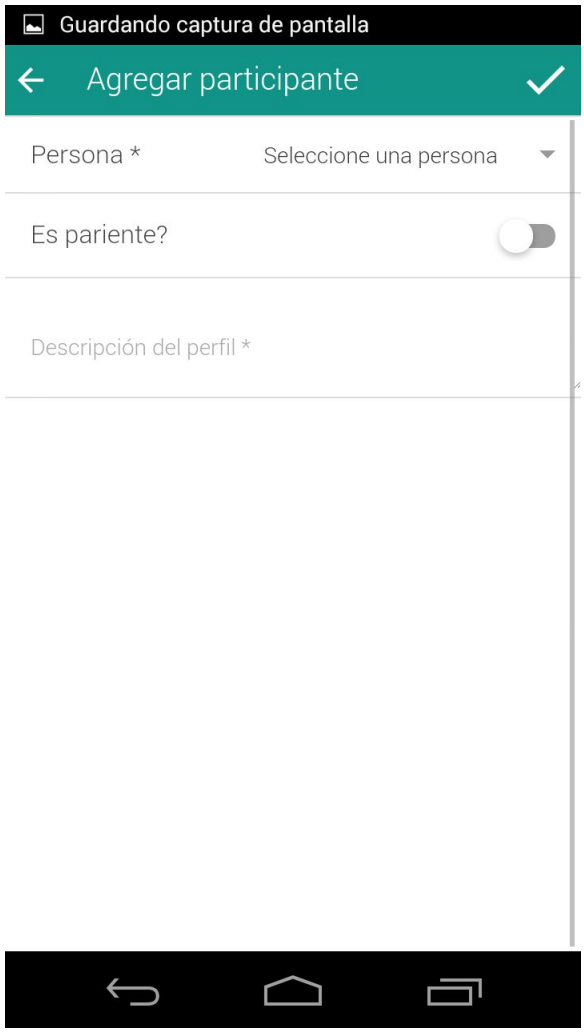
Estas notificaciones se pueden ver listadas en el muro principal del paciente seleccionado**.** Por cuestiones de practicidad el usuario solo tendrá visibles las últimas 15 actualizaciones.

Manejo de participantes

Como	Quiero	Para poder
Usuario Administrador /Usuario	**Administrar los participantes de un tratamiento**	**Recrear en la herramienta al equipo que está participando en el tratamiento de mi paciente**



Listado participantes del tratamiento



Pantalla de alta de participante

Con el objetivo de poder representar en la herramienta al equipo de trabajo encargado del seguimiento del paciente en la vida real, los usuarios ya asignados a un tratamiento van a ser capaces de asignar nuevos participantes al mismo. Una vez que un usuario es asignado como participante de un paciente, ya va a ser capaz de seleccionarlo del listado inicial y realizar todas las funciones disponibles según su perfil.

A la hora de dar de alta a un nuevo participante, se va a poder definir si la persona es pariente o no del paciente y el rol que va a cumplir en el equipo de trabajo. Cabe destacar que para que un usuario pueda ser seleccionable como participante, el mismo tuvo que haber iniciado sesión en la aplicación al menos una vez.

En cuanto a la baja a un participante, como ya se explicó en párrafos anteriores, el único perfil que va a tener disponible dicha funcionalidad es el Usuario Administrador. Una vez que el usuario sea excluido del tratamiento, ya no tendrá seleccionable al paciente en cuestión.

Tanto para las acciones de alta y eliminación de participantes, el usuario va a tener que acceder a la sección **Participantes,** a través del menú desplegable. Allí van a aparecer listados los participantes que están participando actualmente del tratamiento. En caso de que el usuario sea

administrador, a su vez tendrá la opción de eliminar, accion que va a ser realizada al apretar la cruz ubicada a la derecha de la información del participante.

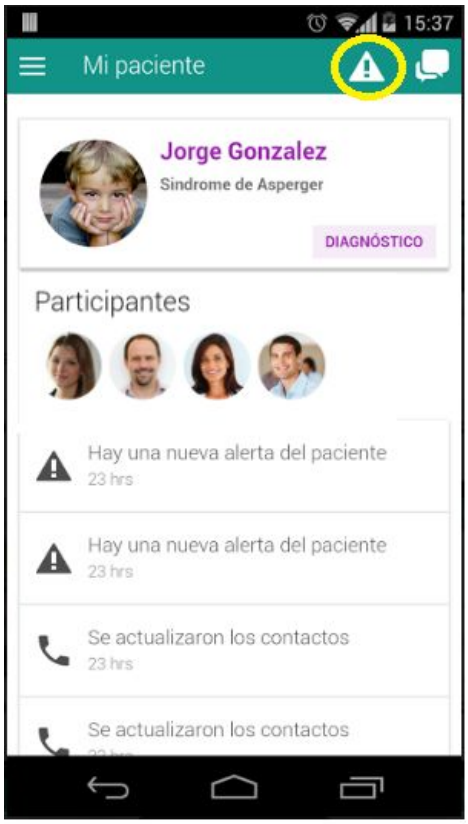
A la hora de querer saber qué personas están participando en el tratamiento de su paciente, los usuarios van a contar con dos opciones:

- Ingresando a través de la opción ***Participantes*** ubicada en el menú desplegable. En esta sección los usuarios van a poder ver el nombre de los participantes y el rol que están cumpliendo en el tratamiento en cuestiòn.

- Ingresando al muro principal del paciente seleccionado debajo de la imagen del paciente en cuestión. A diferencia de lo explicado en la opción anterior, en este caso solo se pueden ver las fotos de los participantes, sin información adicional.

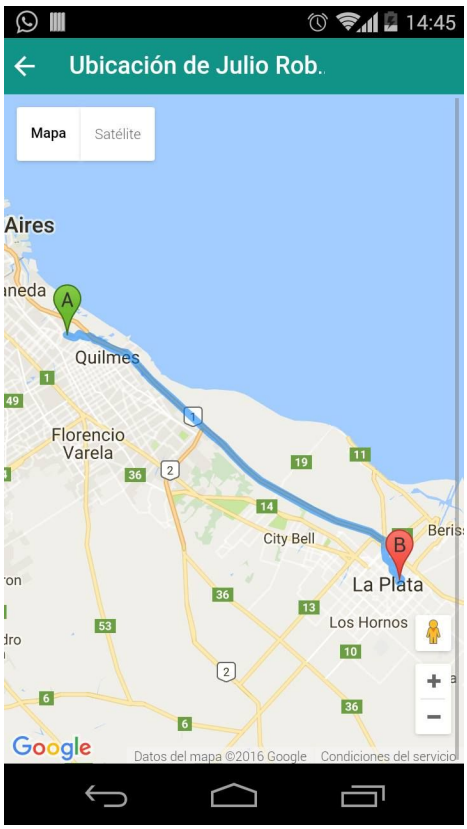
Emisión y recepción de alertas georreferenciadas

Como	Quiero	Para poder
Usuario	**Emitir alertas georeferenciadas**	**Dar aviso a los profesionales y/o tutores de alguna situación crítica relacionada al paciente**
Usuario	**Recibir alertas georeferenciadas**	**Conocer el lugar y momento en que se da una situación crítica y en caso de poder, acudir al lugar.**



Botón que emite el alerta

Notificación de alerta recibida por usuario



Trazado de ruta en el mapa

En caso de ser necesario los usuarios van a tener la posibilidad de emitir alertas georeferenciadas que van a ser recibidas por las demás personas que estén asignados al mismo tratamiento. De esta manera todos los participantes van a estar al tanto de esta ocurriendo una situación que requiere de su atención.

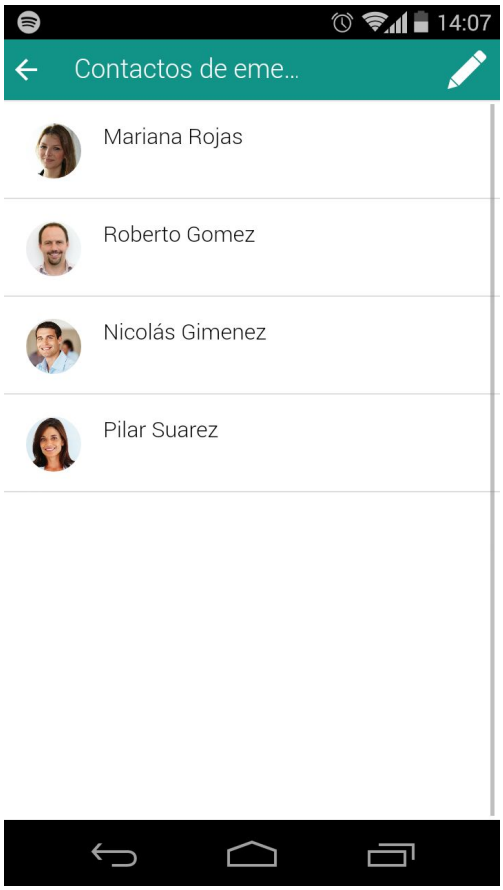
Para emitir una alerta, el usuario debe oprimir el botón relacionado a dicha función localizado en el muro del paciente. Una vez realizada esta acción, los usuarios recibirán una notificación en tiempo real en la que tendrán dos opciones, llamar al paciente o visualizar la ubicación desde donde se emitió el alerta. En el mapa se van a encontrar las siguientes referencias:

- **Punto A:** Locación en la cual se encuentra el usuario que recibe el alerta
- **Punto B:** Locación en la cual se encuentra el usuario que emitió el alerta

A su vez, para brindar mayor información al usuario sobre cómo llegar al lugar en el cual se emitió el alerta, va a aparecer trazada una ruta que conecta ambos puntos. En la imagen xx.x, podemos ver el ejemplo de un usuario que abre la notificación de un alerta emitida en La Plata (Punto B) situado en la ciudad de Quilmes (Punto A).

Configuración de contactos de emergencia del paciente

Como	Quiero	Para poder
Usuario	**Configurar el orden en que aparecen los contactos cercanos de mi paciente**	**Facilitar el uso de la aplicación al paciente**



Listado de contactos cercanos del paciente

Los participantes van a ser capaces de poder determinar cuatro personas que van ser mostrados como contactos de emergencia en la vista del paciente. Van a ser estos los participantes a los que el paciente va a poder llamar o mandar mensajes en caso de que el este atravesando una situación que requiera la asistencia de alguna de las personas designadas como contacto.

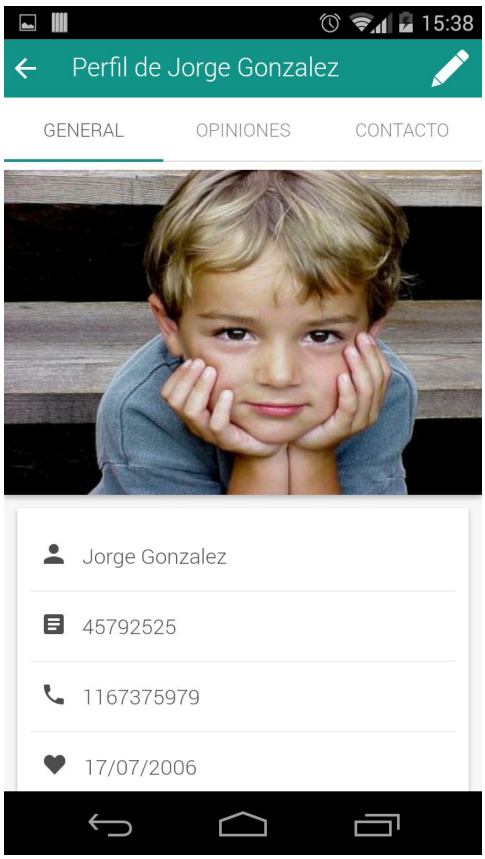
Se podrán seleccionar como máximo cuatro personas, las cuales deben estar participando del tratamiento al que pertenece el paciente para poder ser seleccionables. No es necesario que se elijan las cuatro, ya que puede haber contactos que queden desasignados.

Para realizar esta acción, el usuario tendrá que acceder a la sección Contactos, a través del menú lateral. Una vez allí visualizará el listado de los participantes que actualmente están designados como los contactos de emergencia del paciente. Para modificar esta lista, tendrá

que oprimir el boton de edicion e ingresar los participantes elegidos en el orden que le parezca más adecuado.

Perfil detallado de paciente.

Como	Quiero	Para poder
Usuario	**Ver y editar el perfil detallado de mi paciente**	**Conocer la información completa del paciente con el que estoy trabajando y modificar algún dato en caso de que sea necesario.**



Perfil de paciente - General

Como vimos anteriormente, cuando uno selecciona un paciente, ingresa al muro general en el cual se visualizan las notificaciones, participantes del tratamiento y una muestra reducida sobre los datos personales del paciente. En caso de que el usuario desee conocer información más específica acerca del paciente con el cual está trabajando, tendrá que acceder a su perfil detallado.

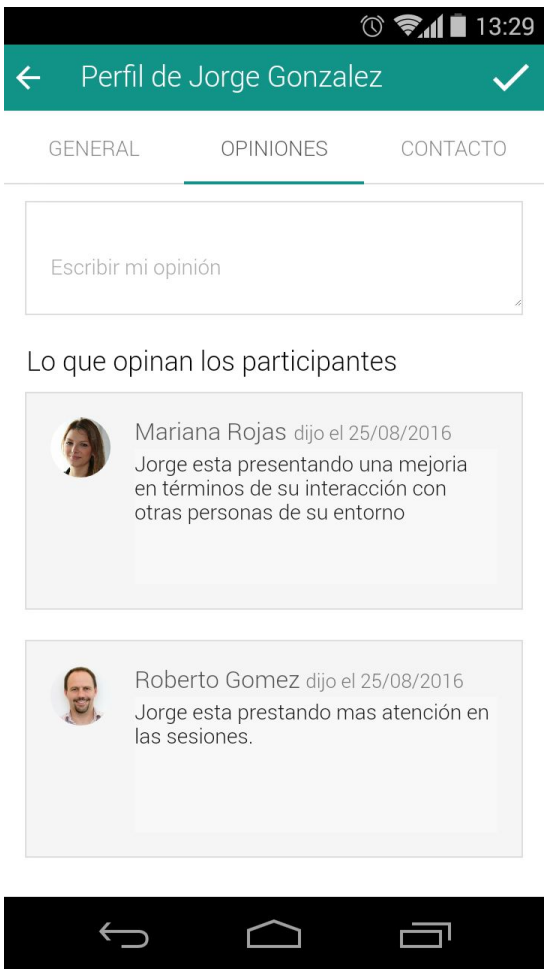
Cuando ingrese a esta seccion encontrara tres tabs una de las cuales, ***General***, va a contener información personal niño. Allí se encontrará con datos como teléfono, fecha de nacimiento, DNI y una breve descripción acerca de él. Además de visualizar, el usuario también será capaz de editar dicha información, y en algunos de los casos, completar los datos que pudo haber dejado vacíos al dar de alta al paciente.

Para acceder a esta pantalla el usuario va a tener que seleccionar a un paciente, y una vez en su muro, oprimir el nombre o imagen del mismo.

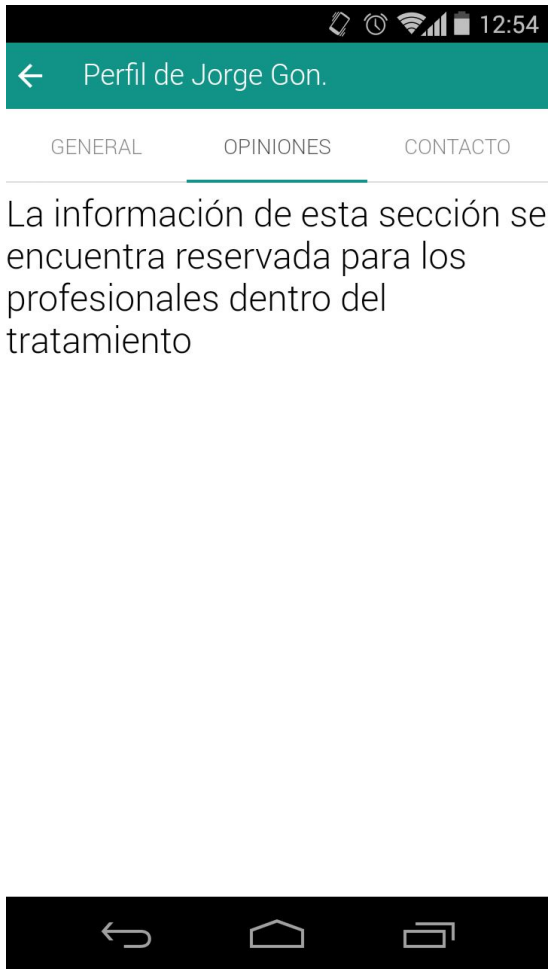
Sección de opiniones de participantes

Como	Quiero	Para poder
Usuario	**Escribir y visualizar opiniones sobre el estado actual de mi paciente y**	**Dar a conocer a los otros participantes mi opinión sobre el paciente en cuestión y a su vez ver**

		cual es la percepción de ellos acerca de él.



Perfil de paciente - Opiniones vista por usuario no pariente



Perfil de paciente - Opiniones vista por usuario pariente

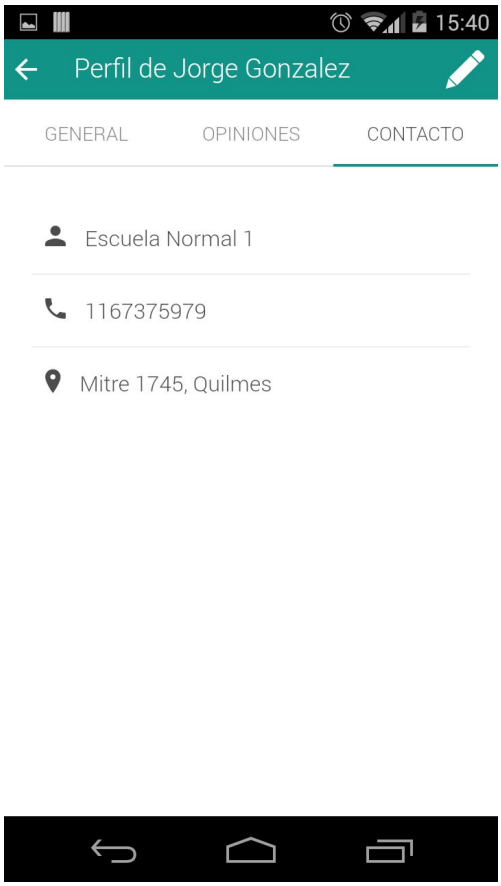
Otra de las funcionalidades que va encontrar el usuario en la sección del Perfil Detallado del paciente es la posibilidad de escribir opiniones personales acerca de su visión actual del paciente. Las mismas podrán ser vistas por los participantes del tratamiento que no sean parientes del paciente involucrado. Esto resulta de vital importancia ya que todos los profesionales involucrados en el tratamiento podrá tener una visión generalizada de la visión que cada uno tiene del paciente en cuestión y no quedarse solo con su punto de vista.

Esta funcionalidad se encuentra en la tab ***Opiniones***, dentro del Perfil Detallado del paciente. Más allá de que la aplicación pueda almacenar más de una opinión de cada participante, solo se mostrará la que fue escrita más recientemente. Se visualiza la imagen y nombre del participante que emitió su opinión, fecha en que fue realizada y el texto de la misma.

En caso de que el participante haya sido dado de alta como pariente del paciente, sólo verá una leyenda informativa que le informara que esta sección está reservada para los profesionales asignados al tratamiento.

Seccion de informacion de contacto del paciente

Como	Quiero	Para poder
Usuario	**Ver y editar información de contacto de mi paciente**	**Tener disponible un número a donde llamar en caso de que haya algún inconveniente con el paciente.**



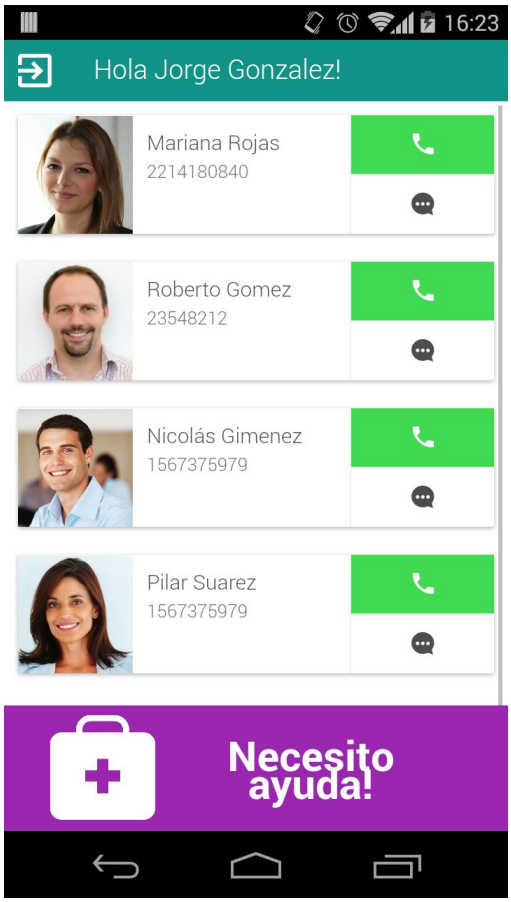
Perfil de paciente - Información de contacto

En la última tab de la sección del perfil detallado del paciente, el usuario va a tener la posibilidad de almacenar datos sobre una persona o establecimiento al cual se pueda contactar en caso de que algún participante quisiera hacer alguna consulta específica acerca del paciente que tenga asignado.

En la imagen relacionada podemos ver el ejemplo de la información de contacto del colegio al cual el paciente se encuentra asistiendo. Como se puede apreciar en la misma, los datos que se pueden ingresar son el nombre del contacto, el teléfono y la dirección.

Vista Paciente

La interfaz gráfica de la aplicación cuando uno se loguea como paciente fue desarrollada para que sea lo más simple e intuitiva posible. Luego de evaluar varias opciones en conjunto con nuestro stakeholder, se decidió que el diseño de la misma quedará como se muestra en la siguiente imagen



Pantalla a ser utilizada por el paciente

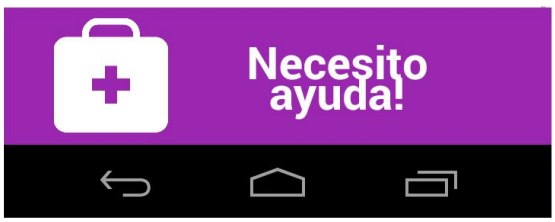
Como se puede apreciar, la pantalla fue diseñada con la premisa de que el paciente no tenga que realizar más de un click para cumplir con la función deseada. También se intentó lograr que las funcionalidades disponibles quedaran lo más explícitas y claras posibles. En esta vista, las acciones que el paciente podrá realizar son:

- Emitir alertas georreferenciadas
- Llamar a alguno de los contactos designados
- Enviar mensaje a alguno de los contactos designados

Dichas funcionalidades serán detalladas a continuación.

Emisión de alertas georreferenciadas

Como	Quiero	Para poder
Paciente	**Emitir alertas georreferenciadas**	**Avisarles a los participantes de mi tratamiento mi ubicación en caso de una emergencia**



Botón para enviar alertas referenciadas desde pantalla del paciente

Los pacientes de un tratamiento van a tener la capacidad de enviar una alerta georeferenciada en los casos que requieran la asistencia inmediata de algún usuario que esté participando en su tratamiento.

Para emitir dicha alerta, el paciente tendrá que oprimir el boton ***Necesito Ayuda*** ubicado en la sección inferior de la pantalla. Una vez emitida, los participantes recibirán una notificación en tiempo real en su teléfono, quedando así alerta sobre las necesidades de su paciente.

Visualización de contactos de emergencia

Como	Quiero	Para poder
Paciente	**Poder visualizar contactos de personas cercanas**	**Poder realizar llamadas rápidas y enviar mensajes predeterminados en caso de emergencia**

Información contenida inicialmente en la tarjeta de contacto cercano

Información contenida en la tarjeta de contacto cercano al oprimir la opción de mensajes

Cuando el paciente ingrese a la aplicación, va a tener visibles a cuatro de los participantes de su tratamiento que son considerados como los contactos de emergencia que el mismo tendrá disponibles para llamar o enviar mensajes en caso de que lo requiera. Como se vio con anterioridad, las personas que son visibles en esta sección deben ser seteadas previamente por algún usuario.

Como se puede ver en las imágenes anteriores**,** en la parte derecha de cada una de las tarjetas en las que se muestran la info del participante designado como contacto de emergencia, el paciente va a encontrar dos botones

- Si oprime el superior, el paciente va a realizar una llamada al participante correspondiente. En cambio, al oprimir el inferior, en el lugar donde se muestra la información del participante van a aparecer dos textos predeterminados.

- Si oprime el inferior, el paciente visualizará dos textos predeterminados, los cuales podrán ser enviados vía SMS al participante en cuestión. Para hacerlo, solo debe seleccionar el mensaje deseado y éste será enviado automáticamente

7.3 TRACS Desktop

Aplicación web para carga de informes

Además de la aplicación móvil que se viene describiendo en las páginas anteriores, TraCS le brinda a los usuarios la posibilidad de cargar diferentes informes sobre el paciente a través de sus computadoras utilizando una versión web. La decisión de separar la escritura de informes de la aplicación móvil se tomó con el fin de favorecer la comodidad del usuario en temas de escritura (resulta más sencillo y rápido escribir textos extensos utilizando el teclado físico que a través de un dispositivo móvil) y además para aprovechar y utilizar las funcionalidades que ofrece la API de Google Drive a la hora del manejo de archivos, tanto privados como compartidos.

Los participantes de un tratamiento van a tener la posibilidad de ingresar al sistema informes que regularmente realizan manualmente. Estos informes van a poder ser definidos como:

- **Privados:** Solo visibles y editables por el usuario que lo redacta.

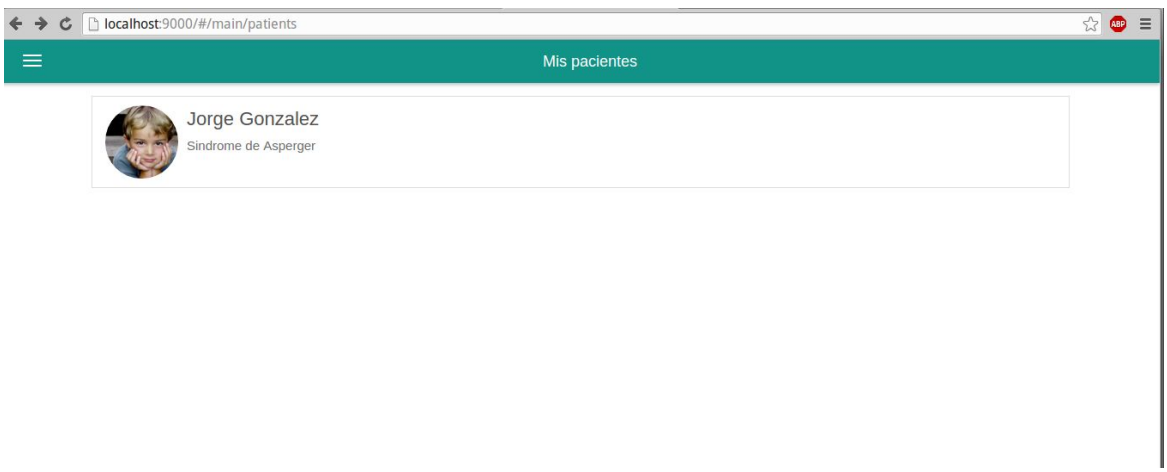
- **Compartidos:** Van a ser visibles y editables para todos los participantes del tratamiento que no hayan sido asignados como parientes del paciente.

Dichos informes serán guardados en el Drive de cada usuario interviniente. Para realizar dicho almacenamiento de manera ordenada, por cada paciente que el participante tenga asignado, decidimos generar una estructura de archivos comprendida por las siguientes carpetas:

➔ **TRACS - reportes - *ID INTERNO PACIENTE***: Se van a almacenar las carpetas que contendrán los reportes privados y compartidos del paciente:

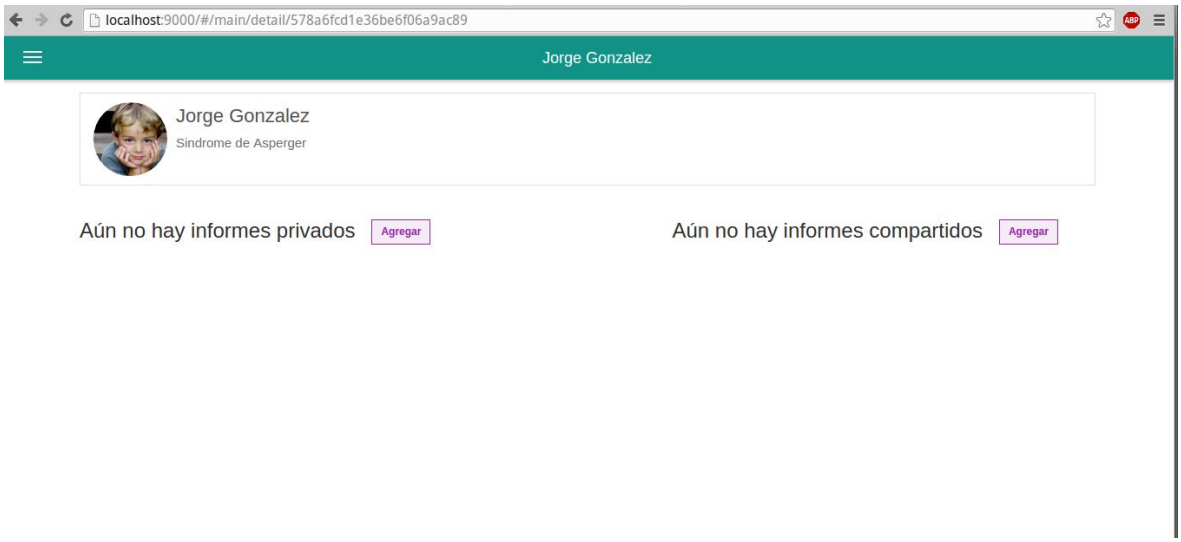
- **TRACS - compartido - *ID INTERNO PACIENTE***: Va a contener los informes compartidos sobre un paciente en particular
- **TRACS - privado - *ID INTERNO PACIENTE***: Va a contener los informes privados sobre un paciente en particular

El uso y estilo de la plataforma web es muy similar al que el usuario encuentra en la aplicación móvil. En la pantalla inicial va a tener que loguearse con su usuario de google que viene utilizando en la herramienta. Una vez logueado, va a ser llevado a una pantalla en la que se verán listados todos los pacientes que tiene asignado.



Lista de pacientes asignados al participante

Una vez que seleccione a uno de sus pacientes, se mostraran en pantalla dos columnas en las que estarán listados los informes privados y compartidos asociados al paciente elegido. En caso de que aún no se hayan redactado algún tipo de informe sobre él, aparecerá la leyenda ***"Aún no hay informes privados/compartidos"*** dependiendo del caso.



Listado de informes asociados al paciente (privados o públicos)

Para crear un informe nuevo, el usuario deberá cliquear el botón Agregar, evento que provocará que se muestre en pantalla el editor de texto de Drive. En dicha pantalla podrá redactar el informe que luego será almacenado en las carpetas especificadas en párrafos anteriores. Una vez que el usuario vuelva a la pantalla principal del paciente, verá allí listado el informe que acaba de crear.

En cuanto a la visualizacion y edicion de reportes, el participante solo deberá seleccionar el informe deseado y a partir de ese momento ya podrá leerlo y realizar las modificaciones que crea pertinentes.

8. Tests de usuario

Podemos definir al test de usuario como una prueba de usabilidad que se basa en la observación y análisis de cómo un grupo de usuarios reales utiliza e interactúa con una aplicación, ya sea web o móvil. Como resultado se puede determinar las sensaciones que tienen los usuarios con la aplicación, el grado de usabilidad y también los problemas de uso con los que se encuentran, con el fin de poder solucionarlos en la próxima iteración.

Los objetivos del test parten al entender que se quiere saber de un determinado producto y marcan quién debe participar en el test, qué tareas realizar, qué datos recoger, qué equipo utilizar, qué medir, qué se considera éxito, cómo analizar la información, qué hacer con la información una vez analizada, etc. Los objetivos ambiguos son una forma de empezar, pero deben concretarse hasta establecer la forma de medirlos:

- **Interés general.** Ejemplo: ¿Serán los usuarios capaces de seleccionar correctamente las opciones del menú?
- **Interés concreto.** Ejemplo: ¿Serán capaces de crear una factura? ¿De modificar una factura ya creada? ¿De enviar la factura en formato PDF por correo electrónico a un cliente?...
- **Forma de medirlo.** Ejemplo: Errores en la selección. Tiempo para completar la tarea. Tareas realizadas.

No se puede preparar un test de usuario con rigor sin conocer antes el perfil del potencial usuario del producto. Los perfiles son una responsabilidad compartida y nunca exclusiva de los especialistas en usabilidad que realizan el test. Un buen perfil identifica dos tipos de características en los usuarios:

- Lo que comparten.
- Lo que los diferencia

Un perfil de usuario debería contener:

- **Descripción general.** Ejemplo: Abogados / Médico / Psicólogo

- **Características relevantes para el test.** Ejemplo: Con experiencia informática. Con trabajo. Que hablen inglés. Que sepan hacer facturas.

- **Características concretas que comparten y sean relevantes para el test.** Ejemplo: Con trabajo desde hace dos años como autónomos. Que utilicen PC y usen habitualmente el producto de la competencia.

- **Características concretas que no comparten y sean relevantes para el test.** Ejemplo: Que usen el producto de la competencia «poco» (0-2 veces por semana) o «mucho» (más de 4 veces por semana).

A la hora de preparar el test es conveniente elaborar un guión en el que se describa: qué le va a decir a cada participante; que le va a pedir que haga; cómo va a hacerlo; cuánto tiempo estima necesario para cada paso en la prueba. No es obligatorio que siga de forma estricta el guión establecido, su función es orientativa.

Es esencial que los participantes del test intenten realizar las tareas que realizará un usuario de ese producto. Siempre hay más tareas que tiempo, por tanto es fundamental decidir qué tareas se van a incluir en el test. Para la selección de las mismas, es necesario priorizar aquellas que podrían llegar a ocasionar problemas de usabilidad.

Se sugiere que las tareas estén incluidas dentro de escenarios. Estos le dicen brevemente al participante lo que tienen que hacer en el test, describiendo las tareas sin artificialidad, usando palabras de usuario y de forma que todos lo entiendan sin ambigüedad. Algunos ejemplos podrían ser:

- Acabas de terminar un caso para un cliente y quieres emitir una factura y mandarla por correo electrónico.

- Un cliente te pide que le modifiques el IVA de la factura que le mandaste el pasado mes.

La selección de las palabras en los escenarios es fundamental, no se deben dar pistas que adulteren el resultado de una tarea. Si lo hacés, el participante puede que realice la tarea sin entender qué hace, y vas a generar productos que funcionan bien en los test pero que van a fallar en el mercado.

8.1 Momentos previos a la prueba

Antes de enfrentar al usuario con la aplicación se debe establecer un ambiente amigable y confortable. Es recomendable empezar explicando que el objetivo de la prueba es evaluar la calidad de uso del sitio, nunca la habilidad ni capacidad del participante. Algo que hay que dejarles en claro es que si el cometen algún tipo de fallo durante la prueba, no será culpa suya, sino del diseño.

Al usuario se le debe instar a que durante la prueba piense en voz alta. Debe tratar de decir todo lo que se le pase por la cabeza al momento de estar probando la aplicación. De hecho, durante el test, si el usuario pasa demasiado rato en silencio mirando la interfaz, es necesario hacerle preguntas para poder conocer en qué está pensando.

El participante deberá hacer lo que le pida el evaluador, expresando qué problemas encuentra, qué no entiende o qué cree que significa cada elemento. Además, el participante debe entender que la misión del evaluador es la de observador silencioso, el evaluador no debe responder ni ayudar al usuario en la consecución de tareas.

Antes de comenzar la prueba, nunca se debe caer en la tentación de explicar al usuario la aplicación a evaluar, ya que de lo que se trata es de comprobar el grado en que la misma resulta auto-explicativo, claro y fácil de comprender.

8.2 Durante las pruebas

Durante la realización de la tarea, justo antes de que el usuario vaya a realizar una acción como es hacer clic, el evaluador puede interrumpir momentáneamente al usuario y preguntarle: ***¿qué cree va a encontrar o a pasar cuando haga clic en botón?***, para dejarlo continuar una vez haya respondido.

Si el usuario se atasca y no consigue terminar la tarea, se le dará las gracias y se pasará a la siguiente tarea. Siempre hay que tener en cuenta que no es un problema del participante, el único que debería sentir cierto grado de frustración por el hecho es el desarrollador.

Otro aspecto importante a tener en cuenta es que los usuarios no son diseñadores ni expertos en usabilidad. No hay que preguntarles acerca de qué diseño consideran más adecuado. Los usuarios y su comportamiento indican problemas de diseño, no su posible solución.

8.3 Testing de TRACS

Teniendo en cuenta los conceptos citados en la sección anterior y tomando como ejemplo el testeo realizado en el artículo ***Del Telegrama a los Tweets: Investigación sobre la Interacción del Adulto Mayor con las Redes Sociales y Aplicaciones Google considerando Aspectos de Usabilidad y Accesibilidad Web*** [84], realizamos la prueba de nuestra aplicación con cinco personas, las cuales se desempeñan en diferentes disciplinas y poseen distintos niveles de manejos de la tecnología. En base a estas dos variables, decidimos dividir al grupo de testers en dos categorías:

- **Usuario con conocimiento Sintáctico:** La persona está muy familiarizada con la tecnología y las aplicaciones de punta, pero no está involucrado en el ambiente ni en las tareas para las cuales va a ser utilizada la aplicación.

- **Usuario con conocimiento Semántico:** Es la persona que actualmente se encuentra realizando las tareas que busca automatizar la herramienta, sabe como realizar el trabajo pero no está tan emparentado con la tecnología.

A su vez, para realizar una mejor valoración de los resultados de las pruebas, decidimos hacer una clasificación de los participantes de acuerdo a su background tecnológico. Decidimos entonces realizar la siguiente categorización

- **Básico:** Usa aplicaciones que vienen instaladas por default, llamada, mensajería, etc
- **Intermedio**: Utiliza aplicaciones tales como Facebook, Twitter, Whatsapp, etc
- **Alto**: Descarga aplicaciones a gusto, conoce de las prestaciones del teléfono, etc

En la siguiente tabla se encuentran enumerados los voluntarios que nos ayudaron a realizar las pruebas pertinentes:

Nombre	Edad	Profesión	Manejo Tecnología
Yasmin Luz Quintanilla	**26**	**Psicóloga**	**Alto**
Sofia Lukaszczuk	**25**	**Psicóloga**	**Intermedio**
Emiliano Moreno	**25**	**Estudiante dirección de cine**	**Intermedio**
Julia Saborido	**30**	**Diseñadora en comunicación visual**	**Intermedio**
Paula Castro	**24**	**Diseñadora en comunicación visual**	**Alto**

Listado de participantes de las pruebas

Como se puede ver, los dos primeros casos pertenecen al grupo de ***Usuario Semántico***, ya que son personas que utilizarían la aplicación en su dia a dia laboral, y los últimos tres representan a un ***Usuario Sintáctico***.

En cuanto a la realización de las pruebas, decidimos estructurar el test de la siguiente manera, estimando una duración de aproximadamente 30 minutos por participante:

Breve entrevista

En esta sección le describimos acotadamente al participante el propósito y objetivo de TRACS, además de realizarle una serie de preguntas con el fin de conocer un poco acerca de él, el trabajo que desempeña y su relación con la tecnología en el día a día. Algunas de las preguntas fueron:

1. ***Contame a qué te dedicas. Objetivos, procesos y tareas.***
2. ***¿Cómo te enterás sobre las novedades del tratamiento de un paciente?***
3. ***¿Con qué frecuencia usás el celular? ¿Por lo general para qué lo usas?***
4. ***¿Cuánto usás la tecnología en los tratamientos?***

Realización de pruebas

En esta etapa, lo primero que hicimos fue presentarles a los usuarios un escenario en el que potencialmente se verían utilizando la aplicación durante su trabajo diario. Después de leer y

comprender el mismo, empezaron con la ejecución de las tareas que decidimos incluir en esta sección. Las mismas fueron:

1. **Crear un paciente**
2. **Agregar diagnóstico**
3. **Agregar una mediación**
4. **Agregar dos participantes al tratamiento (uno que sea pariente del paciente)**
5. **Editar el teléfono del paciente**
6. **Agregar una nota**
7. **Configurar contactos de emergencia para el paciente**
8. **Enviar un alerta de ayuda**

Durante la realización de las pruebas, uno de nosotros se ocupaba de interactuar con el participante, auxiliarlo en caso de que necesitase ayuda o responder inquietudes y el otro se limitaba a tomar notas de situaciones que se pudieran llegar a considerar importantes para tener en cuenta a la hora de sacar las conclusiones finales.

Imagen tomada durante las pruebas de Yasmin Luz Quintanilla

Adicionalmente de la documentación escrita, los participantes nos permitieron videograbar los testeos, lo que nos sirvió luego para poder apreciar sus reacciones a la hora de hacer las tareas. También grabamos la pantalla del celular, lo que nos facilitó mucho ver el camino que trazaron para ir completando los escenarios propuestos y sacar conclusiones acerca de las fallas y mejoras del diseño planteado.

Imagen tomada durante las pruebas de Sofía Lukaszczuk

En la siguiente tabla se pueden observar las notas que tomamos durante la realización de las pruebas. Esta información es la que se utilizó a la hora de tomar las conclusiones finales por cada escenario, las cuales serán detalladas en la sección ***Resultado del test***

Escenarios	Participantes				
	Usuario 1 Yasmin Luz Quintinilla	Usuario 2 Sofia Lukaszczuk	Usuario 3 Julia Saborido	Usuario 4 Emiliano Moreno	Usuario 5 Paula Castro
1- Crear paciente	Encontró el botón de creación rápidamente y posteriormente completó la tarea sin inconvenientes.	Completó la operación rápidamente, pero trató de completar también los campos opcionales	Asoció el boton al alta de paciente. Agregó al paciente correctamente	Visualizó el botón de crear Terminó la tarea sin problemas	Completó la operación sin problemas Le llamó la atención que cuando fue a editar la fecha de nacimiento, el
2- Agregar diagnóstico	Entró a la seccion de diagnóstico sin problemas, en un solo intento. Creo el diagnostico	No tuvo problemas al realizar la tarea	Encontró la sección de un intento Agregó correctamente	Encontró el label del diagnóstico Agregó el diagnóstico sin problemas	Entendió el botón de diagnóstico como una etiqueta. Cuando tuvo que
3- Agregar medicación	Visualizo instanteamente la tab de Medicación y creó la entrada rápidamente	Intentó agregar una medicación antes de agregar el diagnóstico, no se dió cuenta que debía haber un diagnóstico primero	Encontró la tab sin problemas Agregó la medicación rápidamente Sugirió agregar alternativas para ingresar la dosificación	Fue a la tab correspondiente Agregó la medicación sin inconvenientes	Agregó la medicación rápidamente y sin problemas. También eliminó una y esperaba una confirmación, no que se borre sin avisar
4- Agregar 2 participantes	Encontró la opcion correspondiente en el menu lateral rapidamente. Realizo la tarea sin problemas	Se confundió agregar participante con paciente (tal vez por mala lectura o no quedó bien clara la semántica de cada uno), pensó que tenía que volver al listado principal. Se demoró bastante	Duda sobre donde agregar a los participantes No atinó a apretar los participantes en el muro del paciente. Finalmente entro al menú y agregó los participantes correctamente. No tuvo en cuenta el menú para buscar la opción porque en un principio cuando lo revisó en la pantalla inicial, vio que era solo para cerrar sesión.	Buscó mucho tiempo una forma de agregar el participante dentro de la sección de diagnóstico. No atinaba a volver al muro principal del paciente. Cuando regresó, lo primero que hizo fue ir al menú y elegir la opción correcta.	No le quedó claro que desde el muro se podía acceder a la parte de participantes ni que el menú estába asociado al contexto de un paciente en particular. Tardó un poco en encontrar la opción en el menú
5- Editar teléfono del paciente	Tardó unos segundos en ubicar la sección. Una vez encontrada, intentó realizar la modificación inline, en lugar de oprimiendo el botón de editar. Más allá de esto, asoció el diseño del botón con la acción de editar y realizar la modificación del perfil del paciente. Comentó que la modificación inline le hubiese parecido mejor.	Primero entró a editar los datos del centro de comunicación. Necesitó ayuda para encontrar la forma de editar el teléfono	Fue directamente a la tab de contacto. Intuye que como el paciente es chico, tiene que agregar el contacto de la madre/padre en esta tab. Después de agregar la info de contacto, vió el detalle de paciente y ahí sí edito el teléfono en el lugar esperado. Identificó el botón editar y lo modifico	Fue a la info del paciente Trato de hacer la edición inline Luego identificó el botón de editar y agregó el dato correctamente	Entró bien al detalle del paciente y clickeó en la tarjeta para editar el teléfono. Luego encontró rápidamente el lápiz de edición.

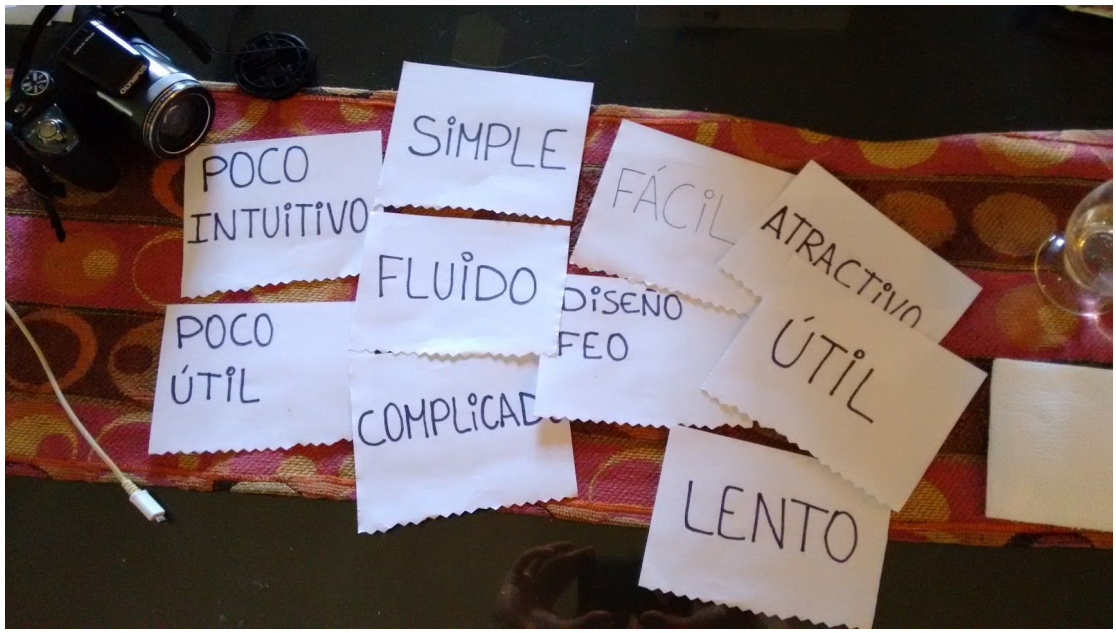
Notas tomadas durante las pruebas - Escenario 1 a 5

Escenarios	Participantes				
	Usuario 1 Yasmin Luz Quintinilla	Usuario 2 Sofia Lukaszczuk	Usuario 3 Julia Saborido	Usuario 4 Emiliano Moreno	Usuario 5 Paula Castro
6- Agregar nota	Pensó durante unos segundos pero pudo encontrar la opcion en el menu lateral y creó la nota sin inconvenientes. No visualizò el footer que indica que las notas son privadas, ya que no supo decirnos para quienes iban a ser visibles las notas creadas por ella	No supo dónde agregar nota, entró en todas las secciones accesibles desde el muro del paciente. Esperaba encontrarlo en la tarjeta del paciente o en el diagnóstico. Entró también al chat. Le llevó bastante tiempo y necesitó ayuda para completar la tarea.	Tardó en encontrar las notas. Se confunde con las opiniones en la info detallada del paciente Luego encuentra la opción en el menú, pero pasa el mismo problema del contexto. No se da cuenta que esas notas van a ser para el paciente seleccionado al principio. No vio el cartel indicando que las notas eran privadas. Piensa que lo ven todos, salvo los padres.	Encontró las notas en el menú Agregó correctamente Considera que las notas son públicas, no tuve en cuenta el footer.	Primero entró a las opciones clickeable desde el muro, la sección de paciente y diagnóstico. Después lo encontró en el menú. Le prestó atención al cartel de privacidad.
7- Configurar contactos de emergencia para el paciente	Encontró rapidamente la opcion en el menu, cuando pensabamos que iba a ser una de las acciones que mas tiempo iba a llevarle. Designó sin inconvenientes los contactos.	Para editar los contactos de emergencia primero entró a la sección de participantes. Igual lo encontró rápido y pudo completarlo	Cree que los contactos de emergencia van en la tab de Contacto Luego va al menú y encuentra la opción Agrega sin problemas	Encontró rapidamente la opcion en el menu lateral Agregó los contactos de emergencia sin problemas	Primero tocó el botón de alerta, luego la sala de chat y finalmente lo encontró en el menú
8- Enviar un alerta de ayuda	Envió el alerta rapidamente. Pudo relacionar correctamente el icono con la funcionalidad	El alerta la buscó primero en el menú y luego la encontró rápidamente	Encuentra rápido el botón y envia el alerta	Envió el alerta rapidamente. Pudo relacionar correctamente el icono con la funcionalidad	Completó la tarea rápidamente

Notas tomadas durante las pruebas - Escenario 6 a 8

Feedback

En esta sección no solo les preguntamos sus sensaciones luego del test, si pudieron realizar las tareas o si notaron dificultad en alguno de los escenarios. Además de esto, decidimos implementar una actividad en la que les brindamos una serie de tarjetas compuestas por palabras que describen positiva o negativamente los aspectos de la herramienta, sin haber un límite de tarjetas seleccionables. Una vez finalizada su elección les pedimos que nos explicaran las razones por las cuales las habían seleccionado. Este método resulta interesante porque el participante puede expresar su opinión sin estar tan sesgado por tener al entrevistador de frente, puede elegir libremente sin sentirse presionado.



Tarjetas utilizadas durante la etapa de feedback

A continuación se puede apreciar una gráfica que muestra la elección que realizó cada uno de los participantes:

	Participante				
Tarjeta	**Yasmin Luz Quintanilla**	**Sofia Lukaszczuk**	**Emiliano Moreno**	**Julia Saborido**	**Paula Castro**
Poco Intuitivo					
Útil					
Simple					
Diseño Feo					
Fluido					
Fácil					
Atractivo					
Lento					
Poco útil					
Complicado					

Tarjetas seleccionadas por cada participante

Cierre

Como un gesto de agradecimiento al finalizar con las pruebas les entregamos a cada participante una Gift Card para que pueda destinar el dinero incluida en ella en lo que desearan.

Una de las Gift Cards que fueron entregadas a los participantes

8.4 Resultados del test

En base a lo observado en cada una de las pruebas, sacamos una conclusión general de cada uno de los escenarios, en los cuales destacamos los puntos altos y bajos de cada uno. Esto se puede apreciar en la siguiente tabla:

Escenarios	Conclusión
1- Crear paciente	No se presentaron problemas a la hora de realizar esta tarea. Los cinco participantes identificaron el botón y completaron la misma sin dificultad
2- Agregar diagnóstico	Todos los participantes pudieron realizar la tarea con éxito. En algunos casos se generaron confusiones porque pensaban que este diagnóstico iba a coincidir con el diagnóstico general agregado en el Alta Paciente.
3- Agregar medicación	Tarea realizada sin problemas, en partes porque se encuentra en la tab subsiguiente a la de Diagnóstico. Solo se presentó un caso en el que se trató de agregar una medicación sin antes haber dado de alta el diagnóstico relacionado
4- Agregar 2 participantes	Fue una de las tareas que representó confusión. En un solo caso se dio el acceso sin reiterados intentos ni búsquedas. En los demás casos se tardó considerable tiempo en encontrar la funcionalidad correcta y a su vez en asociar al menú lateral con opciones relacionadas al paciente. Identificamos los primeros problemas de contextualización, ya que la mayoría de los usuarios creían que las opciones presentes en el mismo eran generales de la aplicación y que no afectaban directamente al paciente seleccionado
5- Editar teléfono del paciente	En dos de los casos se dió la confusión entre el teléfono del paciente y la información de Contacto (en uno de ellos se necesito ayuda nuestra para completar la tarea). En todos los casos se trato de editar el teléfono de una manera in-line (apretando el elemento y editando en la misma línea) en lugar de oprimir el botón de editar. Más allá de esto el botón fue identificado al instante luego de ver que la edición que se intentó en un principio no era posible
6- Agregar nota	En la mayoría de los casos se encontraron dificultades para hallar la opción. Solo en un caso se dio la asociacion del menu con opciones adicionales sobre el paciente. Los demás buscaron la opción dentro de las secciones de Información

	Detallada y Diagnóstico antes pensar sobre el menú. Esto se dio principalmente por el problema de contextualización mencionado anteriormente. Opinan que sería preferible tener la opción disponible en el mismo muro del paciente. Otra cosa que notamos es que el footer indicando que las notas son privadas no fue tenido en cuenta por cuatro de los cinco testers, no fue tan efectivo como pensábamos
7- Configurar contactos de emergencia para el paciente	En este caso también se notaron confusiones entre Contactos de Emergencias y la tab de Contacto dentro de la sección Información Detallada. Solo en dos de las cinco pruebas la tarea se realizó sin intentos extras.
8- Enviar un alerta de ayuda	El botón para enviar el alerta fue identificado sin problemas en todos los casos

Conclusiones obtenidas por cada escenario testeado

A su vez decidimos evaluar el uso de la aplicación por cada uno de los participantes basándonos en los siguientes conceptos:

- **Eficacia**: El usuario logra o no realizar la tarea

- **Eficiencia:** Si lo logra realizar la tarea, en cuanto tiempo lo hace, cantidad de intentos que necesita para finalizar, secciones a las que entra antes de ingresar a la correcta, cantidad de veces que pide asistencia

- **Satisfacción:** Opiniones del usuario acerca de su experiencia durante el uso de la aplicación.

Para poder medir la eficiencia de una forma correcta decidimos trazar umbrales en términos de tiempo, cantidad de intentos antes de terminar y ayudas requeridas que cada tipo de usuario tendría que respetar para considerar la prueba como exitosa. A su vez tuvimos en cuenta al parámetro de ***background tecnológico*** citado al principio de la sección. En la siguiente tabla se puede ver el detalle de estos umbrales:

Tarea	Tiempo (en minutos)			Cantidad de intentos antes de terminar			Ayudas requeridas		
	Básico	Intermedio	Alto	Básico	Intermedio	Alto	Básico	Intermedio	Alto
1- Crear paciente	1:30	1:00	0:40	1	1	1	0	0	0
2- Agregar diagnóstico	1:30	1:00	0:40	2	1	1	1	0	0
3- Agregar medicación	1:10	0:50	0:30	1	1	1	0	0	0
4- Agregar 2 participantes	2:00	1:20	1:00	3	2	1	1	0	0
5- Editar teléfono del paciente	1:30	1:00	0:40	2	2	1	1	1	0
6- Agregar nota	1:30	1:00	0:40	2	1	1	1	0	0
7- Configurar contactos de emergencia para el paciente	2:00	1:20	1:00	3	2	2	2	1	1
8- Enviar un alerta de ayuda	0:30	0:20	0:10	2	1	1	1	0	0

Umbrales determinados para cada tipo de usuario

Basándonos en lo mencionado anteriormente, estuvimos en condiciones de poder sacar las siguientes conclusiones respecto a cada participante:

Usuario	Eficiencia	Eficacia	Satisfaccion
Yasmin Luz Quintanilla	En general pudo realizar las tareas en el tiempo y forma esperados. No requirió ayuda para completar las tareas	Las tareas fueron completadas con exito	Se mostró conforme con la idea y las funcionalidades presentadas en la aplicacion. Se la notó entusiamada y con ganas de poder utilizarla en su día a día laboral
Sofia Lukaszczuk	Solo requirió ayuda en dos de los ocho escenarios planteados, el resto de las tareas las pudo resolver sin intervención nuestra pero sí realizando varios intentos antes de poder finalizarlas	Las tareas fueron completadas con exito	Le pareció una idea útil para comunicarse rapidamente y estar en contacto entre participantes. Considera que podría ser aplicada en su trabajo. Destacó la forma en que la herramienta podría acortar los tiempos de tareas que involucren a más de un actor
Emiliano Moreno	Pudo realizar las tareas en el tiempo y la forma esperados. No fue necesario brindarle ayuda para hacer ninguna de las 8 tareas	Las tareas fueron completadas con exito	Considera que la herramienta puede tener un gran grado de utilidad entre los profesionales que desempeñan este tipo de tareas y a su vez para los familiares de los pacientes involucrados. Le parecio simple y practica, mas que nada por notar un diseño muy similar al de las aplicaciones de Google
Julia Saborido	Pudo completar las tareas, pero le complicó un poco la falta de contexto con respecto a las opciones presentes en el menú. Le dificulto aun mas visualizar las opciones en la pantalla inicial, en la que la unica opcion es la de cerrar sesión	Las tareas fueron completadas con exito	Le pareció muy conveniente disponer de la información en tiempo real, sin la necesidad de tener que dirigirse a un lugar fisico determinado. Destacó la mejora que esta herramienta produciria en la relación entre los participantes del tratamiento. Le pareció facil de seguir y manejar y destacó la simpleza en la carga de los formularios.
Paula Castro	Si bien realizó casi todas las tareas dentro de los parámetros de eficiencia esperados, tuvo varios reintentos provocados por el desorden y las jerarquías de información confusas.	Las tareas fueron completadas con exito	Lo cree útil para que los profesionales puedan tener un registro digital en tiempo real y puedan mantener una comunicación fluida. Cree que puede ser útil para la familia, para sentirse acompañada por los profesionales.

Conclusiones sobre cada participante basándose en Eficiencia - Eficacia - Satisfacción

En términos generales podemos concluir que los resultados de los testeos fueron satisfactorios ya que todos los escenarios propuestos fueron completados dentro de los parámetros esperados. Como principal mejora en una próxima iteración, consideramos que habría que hacer foco en mejorar la contextualización de las operaciones disponibles para cada paciente, ya que los usuarios encontraron dificultades a la hora de definir el alcance de cada una. También habría que

revisar la jerarquía de la información, de manera que se muestre lo más relevante primero y más fácil de acceder.

9. Conclusión

Luego de realizar el presente trabajo pudimos comprender las dificultades comunicacionales y organizacionales a las que se enfrentan las personas que integran el equipo encargado de llevar el control y seguimiento de un paciente con algún tipo de trastorno cognitivo.

La creación de esta herramienta fue un desafío para nosotros ya que representó la realización de un desarrollo desde cero incluyendo las fases de entrevistas con usuarios, elicitación de requerimientos, realización de mockups pensando en la experiencia de usuario y usabilidad para facilitar su uso, e investigar y programar con tecnologías con las que no estábamos familiarizados y que fuimos aprendido a medida que trabajamos sobre las funcionalidades.

Decidimos cambiar el enfoque clásico del modelo en cascada por una metodología que permitiera una mayor inclusión del usuario durante las etapas de desarrollo. Es por esto que optamos por un método ágil, compuesto por un mayor número de iteraciones de corta duración en las cuales se creaba un prototipo contemplando un grupo definido de historias de usuario el cual, una vez finalizado, era presentado al usuario para conocer su opinión y validar los cambios.

Después de la etapa de testing, obtuvimos comentarios favorables de parte de Sofía y una colega que también participó de las pruebas, mostrándose realmente interesadas en empezar a usar la aplicación en su ámbito laboral en el corto plazo ya que consideran que su trabajo se vería beneficiado mediante el uso de la misma. También tuvimos la posibilidad de participar en una Jornada de Accesibilidad organizada en la universidad, donde pudimos exponer el trabajo realizado, recibiendo un feedback positivo por parte de varios de los presentes, lo cual resultó gratificante ya que significa que el desarrollo realizado puede tener un impacto positivo en la sociedad.

Con esto dicho, podemos concluir que logramos el objetivo de crear una herramienta que produzca una mejora sustancial en la forma de realizar el acompañamiento de un paciente, reduciendo los tiempos de respuesta y generando un acercamiento de los participantes de un tratamiento, fomentando así la comunicación y la participación de los mismos.

Como reflexión final podemos decir que todo este trabajo no hubiese sido posible sin la formación obtenida a lo largo de estos años pasados dentro de la facultad. Fueron varias las materias cursadas que vienen al caso destacar y que nos dieron bases sólidas de conocimientos para poder encarar y afrontar un problema: en Proyecto de Software aprendimos las ventajas del uso del patrón MVC para la estructuración del código en capas, posibilitando que el mismo sea escalable y fácil de mantener; Ingeniería de Software nos ayudó a comprender las distintas metodologías y etapas del proceso de desarrollo de una aplicación desde cero; y en Programación Orientada a Objetos aprendimos cómo abstraer y encapsular el sistema en clases para favorecer la delegación de comportamiento, las buenas prácticas y la utilización de patrones para resolver problemas comunes de programación.

9.1 Trabajos a futuro

A lo largo del desarrollo de la aplicación surgieron un gran número de posibles funcionalidades que dado a la falta de tiempo para completarlas han quedado fuera del alcance de la versión presentada en el presente trabajo. Es por ello que hemos decidido incluirlas en una lista de desarrollos y mejoras a ser implementados a futuro. Las mismas se encuentran listadas a continuación:

- **Perfiles predeterminados:** A la hora de dar de alta un participante se le podrá asignar perfiles predeterminados (***Psicólogo***, ***Médico***, ***Familiar***, ***Maestro***) los cuales definirán las funcionalidades que el mismo tendrá disponible en la aplicación.

- **Cronograma de sesiones a las que asiste el paciente:** Cada participante podrá estar al tanto de los días y horarios en los cuales el paciente asiste a una sesión con un profesional o realiza alguna actividad recreativa. Esto será de gran ayuda a la hora de poder coordinar las sesiones con sus pacientes de la manera más eficiente posible.

- **Configuración adaptada de la pantalla destinada al paciente:** La idea es darle al usuario la posibilidad de configurar la interfaz utilizable por el paciente dependiendo del tipo y grado de discapacidad que el mismo posea. Un ejemplo puede ser definiendo el tamaño o forma de mostrar los contactos disponibles, colores de los elementos, entre otros.

- **Recordatorio de horario de medicación:** Esta funcionalidad estaría destinada a los participantes que sean dados de alta como familiares. Les va a permitir tener un cronograma de los horarios en los cuales deben suministrarle los medicamentos al paciente.

- **Información offline, ahora es todo vía internet:** Se busca que los usuarios tengan disponible al menos una parte de la información almacenada en la aplicación de manera local, sin la necesidad de contar con una conexión a internet.

- **Templates de mensajes para el paciente:** La idea con esta mejora es que los participantes puedan configurar plantillas para los mensajes que el paciente tiene disponibles para enviar a sus contactos.

- **Templates para respuestas automáticas de mensajes de pacientes:** Los destinatarios de los mensajes enviados por el paciente van a tener la capacidad de poder responder los mismos con plantillas pre configuradas dentro de la aplicación.

- **Notificaciones push para la mensajería y las notificaciones:** Con este desarrollo se busca extender la funcionalidad de notificaciones push utilizada a la hora del envío de Alertas Georreferenciadas, para los mensajes recibidos en alguno de los chats grupales de los tratamientos en los que esté involucrado el participante, como así también a los cambios que generan una entrada en el muro de los pacientes que tenga asignado.

- **Posibilidad de ver informes en aplicación móvil:** El objetivo es que los usuarios puedan visualizar los informes que se hayan confeccionado a lo largo del tiempo en sus dispositivos móviles, dejando así de depender de estar conectados a su pc de escritorio para tener acceso a los mismos.

- **Marcar como "atendida" un alerta georreferenciada**: Esta funcionalidad permitirá que una vez que un participante decida hacerse cargo del alerta recibida, pueda marcarla como "atendida", acción que notificará a los demás usuarios que el paciente será asistido a la brevedad.

- **Generación de reportes automáticos:** El fin de esta mejora es darle a los usuarios la oportunidad de poder obtener reportes de los objetos que son almacenados en la aplicación, ya sean notas, opiniones, medicaciones, etc. Un ejemplo podría ser, obtener todas las notas que se hicieron sobre un paciente durante un mes determinado.

Para extender, mejorar o modificar TRACS, el código se encuentra disponible públicamente en Github bajo la licencia GNU GPLv3 que permite la libre distribución del mismo y la realización de trabajos derivados bajo la misma licencia. A continuación se encuentran disponibles los links de los repositorios para ser forkeados o clonados:

- **TRACS Server:** https://github.com/leito9008/tracs-server
- **TRACS Móvil:** https://github.com/leito9008/tracs-client
- **TRACS Web:** https://github.com/leito9008/tracs-desktop

10. Referencias bibliográficas

[1] Manjarres, Juan Jose, Cabrera Juan Esteban, Latorre Daniela, Reyes Johan, Costa Rica, Universidad del Valle. (Disponible en: https://sites.google.com/a/correounivalle.edu.co/software-colaborativo-en-internet-uva-univalle/caracteristicas-software-colaborativo-de-internet-1. Consultado el 14 de junio de 2015)

[2] Tramullas, J., Garrido-Picazo, P., Sánchez-Casabón, A. 2011. España, Universidad de Zaragoza, Escuela Universitaria Politécnica de Teruel. (Disponible en: http://www.researchgate.net/publication/241344157_Groupware_y_software_social_propuesta_de_marco_de_evaluacin_analtica_para_herramientas_de_software_libre. Consultado el 14 de junio de 2015)

[3] Velasco, Juan J., A. 2011. España, Sevilla. (Disponible en: http://hipertextual.com/archivo/2011/01/diez-herramientas-colaborativas/. Consultado el 14 de junio de 2015)

[4] Clarín, A. 2015 (Disponible en: http://next.clarin.com/mundo-misma-cantidad-celulares-gente_0_1432056851.html. Consultado el 7 de octubre de 2016)

[5] Dennison L, Morrison L, Conway G, Yardley L. 2013. ***Opportunities and Challenges for Smartphone Applications in Supporting Health Behavior Change: Qualitative Study.*** Southampton, Reino Unido. Academic Unit of Psychology, University of Southampton.

[6] Informaticahoy (Disponible en: http://www.informatica-hoy.com.ar/telefonos-celulares/La-evolucion-telefonos-celulares.php. Consultado el 23 de julio de 2016)

[7] La Nación, A. 2016 (Disponible en: http://www.lanacion.com.ar/1900814-android-vs-ios-quien-gana-la-batalla. Consultado el 7 de octubre de 2016)

[8] Mastromarino, Francisco. A.2013 (Disponible en: http://androidzone.org/2013/05/historia-de-android-la-evolucion-a-lo-largo-de-sus-versiones/. Consultado el 20 Marzo del 2016)

[9] Mitroff, Sarah., Garzon, Juan. A.2013 (Disponible en: http://www.cnet.com/es/analisis/google-android-kitkat/. Consultado el 20 Marzo del 2016)

[10] Google, A. 2015 (Disponible en: https://material.google.com/#. Consultado el 28 de septiembre de 2016)

[11] Enrique Pérez, A. 2014 (Disponible en: http://www.elandroidelibre.com/2014/11/que-es-material-design.html. Consultado el 28 de septiembre de 2016)

[12] GCOOP, A. 2013 (Disponible en http://gcoop.coop/conectar-igualdad-plataforma-web-de-trabajo-colaborativo-y-reportes. Consultado el 10 Diciembre del 2015)

[13] Minutouno, A. 2014 (Disponible en http://www.minutouno.com/notas/318729-como-funciona-el-boton-alerta-celulares-del-gobierno-la-provincia. Consultado el 10 Diciembre del 2015)

[14] Google Playstore. (Disponible en https://play.google.com/store/apps/details?id=com.wli.patient_track. Consultado el 8 de Diciembre del 2015)

[15] Vocera. A. 2015. (Disponible en http://www.vocera.com/vocera-communication-system. Consultado el 8 de Diciembre del 2015)

[16] Apezteguía, Matías. (2014). ***Juego educativo móvil colaborativo*** (Tesis de grado). Facultad de Informática UNLP, La Plata

[17] Lacomba, Agustín Eduardo. (2013). ***Construcción de una Aplicación basada en tecnologías web y dispositivos móviles para la gestión del conocimiento en Comunidades de prácticas*** (Tesis de grado). Facultad de Informática UNLP, La Plata

[18] Herrera Alexander, Rodríguez Darío, García-Martínez Ramón. 2013. ***Taxonomía de Mecanismos de Awareness***. Programa de Maestría en Ingeniería de Sistemas de Información. UTN-FRBA - Laboratorio de Investigación y Desarrollo en Espacios Virtuales de Trabajo - Grupo de Investigación en Sistemas de Información. Universidad Nacional de Lanús.

[19] María Murazzo, Nelson Rodríguez, Daniela Villafañe, Daniel Gallardo. 2013. ***Desarrollo de aplicaciones colaborativas para Cloud Computing.*** XVIII Congreso Argentino de Ciencias de la Computación. Departamento e Instituto de Informática – FCEFyN – UNSJ.

[20] Heroku, A. 2016 (Disponible en: https://www.heroku.com/what. Consultado el 10 de Junio de 2016)

[21] Peter Johnson-Lenz, Trudy Johnson-Lenz. 1981. ***Consider the Groupware: Design and Group Process Impacts on Communication in the Electronic Medium***. New Jersey Institute of Technology, Newark, New Jersey.

[22] Dávila Martínez, Francisco J., Camacho Arranz, Elena. 2012. **Georreferenciación de documentos cartográficos para la gestión de Archivos y Cartotecas**. Santander, España, Instituto Geográfico Nacional.

[23] Compilador J. Fallas. 2010. **Geoprocesamiento Análisis de geodatos**. Posgrado en Gestión de Áreas Protegidas y Desarrollo Ecorregional. Universidad para la Cooperación Internacional. UCI.

[24] Mangiaterra, A. - Grupo de Geodesia Satelital Rosario. 2014. ***Geografía y georreferenciación. Aplicación de GPS en la enseñanza***. Argentina, AMSAFE Rosario.

[25] Siliconhosting.com, A. 2013 (Disponible en: https://siliconhosting.com/blog/2013/07/cloud-iaas-paas-y-saas/. Consultado el 1 de septiembre de 2016)

[26] HealthIt.gov (Disponible en: http://healthit.gov/patients-families/health-it-and-health-care-quality. Consultado el 16 de septiembre de 2016)

[27] Margaret Rouse, A. 2016 (Disponible en: http://searchhealthit.techtarget.com/definition/Health-IT-information-technology. Consultado el 16 de septiembre de 2016)

[28] Adrián Alonso Vega. ***Responsive Web Design: Interfaces Web Adaptables al dispositivo empleando HTML5 y CSS3 .*** Madrid, España. Universidad de Alcalá

[29] Ethan Marcotte. 2011. ***Responsive Web Design***. Estados Unidos, A book Apart.

[30] Ben Frein. 2012. ***Responsive Web Design with HTML5 and CSS3.*** Inglaterra, Packt Publishing.

[31] Kony (Disponible en: http://www.kony.com/resources/glossary/cross-platform-mobile-development. Consultado el 25 de septiembre de 2016)

[32] Techopedia (Disponible en: http://www.techopedia.com/definition/30026/cross-platform-development. Consultado el 25 de septiembre de 2016)

[33] Marcos Merino, A. 2014 (Disponible en: https://www.paradigmadigital.com/dev/api-management-que-es-y-para-que-sirve/. Consultado el 12 de octubre de 2016)

[34] Marcos Merino, A. 2014 (Disponible en: http://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/. Consultado el 12 de octubre de 2016)

[35] Tomas, Eduard. A.2014. (Disponible en: http://www.desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html. Consultado el 20 de Febrero de 2016)

[36] Pimienta, Pedro, A. 2014. Colombia. (Disponible en: https://deideaaapp.org/tipos-de-aplicaciones-moviles-y-sus-caracteristicas/. Consultado el 19 de marzo de 2016)

[37] Lancetalent.com, A. 2016. (Disponible en: https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/. Consultado el 19 de marzo de 2016)

[38] The Apache Software Foundation, A. 2016. (Disponible en: https://cordova.apache.org/docs/en/latest/guide/overview/. Consultado el 21 de marzo de 2016)

[39] Appcelerator, A. 2016. (Disponible en: http://docs.appcelerator.com/. Consultado el 21 de marzo de 2016)

[40] Adobe, A. 2016 (Disponible en: http://docs.phonegap.com/. Consultado el 21 de marzo de 2016)

[41] Drifty Co, A. 2016 (Disponible en: http://ionicframework.com/docs/. Consultado el 21 de marzo de 2016)

[42] Jeremy Wilken. ***Ionic IN ACTION*.** Versión 9. Estados Unidos, Manning Publications

[43] Mike Cantelon,Marc Harter. 2014. ***Node.js in Action***. Estados Unidos. Manning Publications

[44] Manuel Kiessling. 2012. ***The Node Beginner Book, A comprehensive Node.js tutorial***

[45] Prime, Richard. A.2015. (Disponible en: http://www.information-age.com/rise-nodejs-and-why-it-will-rule-enterprise-software-development-least-decade-123460405/. Consultado el 28 de Octubre del 2016)

[46] Xiao, Yunong. A.2014. (Disponible en: http://techblog.netflix.com/2014/11/nodejs-in-flames.html. Consultado el 28 de Octubre del 2016)

[47] Oliver, Kiran "CK", A. 2015 (Disponible en: http://thenewstack.io/netflix-uses-node-js-power-user-interface/. Consultado el 28 de Octubre del 2016)

[48] Harrell, Jeff. A.2013. (Disponible en: https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal/. Consultado el 28 de Octubre del 2016)

[49] Nodejs. A.2016. (Disponible en https://nodejs.org/static/documents/casestudies/Node_CaseStudy_Nasa_FNL.pdf. Consultado el 28 de Octubre del 2016)

[50] Kristina Chodorow. 2013. ***MongoDB: The Definitive Guide***. 2 ed. Estados Unidos, O'Reilly.

[51] Karl Seguin. ***The Little MongoDB Book***

[52] Mongoose. A.2016. (Disponible en http://mongoosejs.com/. Consultado el 20 de Febrero del 2016)

[53] Craig Buckler, A. 2015 (Disponible en: https://www.sitepoint.com/sql-vs-nosql-differences/. Consultado el 25 de octubre de 2016)

[54] Easylearning , A. 2016 (Disponible en: http://blog.easylearning.guru/what-is-mongodb-the-advantages-disadvantages-of-mongodb/. Consultado el 25 de octubre de 2016)

[55] Kelleher, Fionn. A.2014. (Disponible en: https://nodesource.com/blog/understanding-socketio/. Consultado el 22 de Marzo del 2016)

[56] Kitamura, Eiji. Ubl, Malte. A. 2010. (Disponible en http://www.html5rocks.com/es/tutorials/websockets/basics/. Consultado el 22 de Marzo del 2016)

[57] I. Fette, Google Inc., A. Melnikov. 2011. ***RFC 6455*** - ***The WebSocket Protocol***

[58] Ari Lerner. 2013. ***ng-book, The Complete Book on Angular.*** FullStack.io

[59] Dan Wahlin. 2014. ***AngularJS in 60 Minutes.***

[60] Bower, A. 2016 (Disponible en: http://bower.io/. Consultado el 21 de marzo de 2016)

[61] Alvarez, Miguel A., A. 2015. (Disponible en: http://www.desarrolloweb.com/articulos/uso-bower-gestor-dependencias.html. Consultado el 21 de marzo de 2016)

[62] Git, A. 2016 (Disponible en: https://git-scm.com/. Consultado el 2 de abril de 2016)

[63] Hongkiat.com., A. 2015. (Disponible en: http://www.hongkiat.com/blog/gulp-vs-grunt/. Consultado el 21 de marzo de 2016)

[64] Catlin Hampton, Weizenbaum Natalie, Eppstein Chris, A. 2016 (Disponible en: http://sass-lang.com/documentation/file.SASS_REFERENCE.html. Consultado el 2 de abril de 2016)

[65] Css-tricks, A. 2013 (Disponible en https://css-tricks.com/autoprefixer/. Consultado 15 de julio de 2016)

[66] Mario Pérez Esteso, A. 2014. (Disponible en: https://geekytheory.com/genymotion-un-rapido-y-eficiente-emulador-android/. Consultado el 22 de abril de 2016)

[67] Txema Rodríguez, A. 2014. (Disponible en: http://www.xatakandroid.com/roms-android/genymotion-el-emulador-mas-rapido-de-android. Consultado el 22 de abril de 2016)

[68] Android Studio, (Disponible en: https://developer.android.com/studio/command-line/adb.html. Consultado el 15 julio 2016)

[69] Hardt, Dick, Microsoft. 2012. ***RFC 6749*** - ***The OAuth 2.0 Authorization Framework***

[70] Yusef Hassan Montero, Sergio Ortega Santamaría (Disponible en: http://www.nosolousabilidad.com/manual/3.htm. Consultado el 12 de Marzo de 2016)

[71] María Paula Izaurralde. 2013. ***Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario***. Córdoba, Argentina. Universidad Tecnológica Nacional Facultad - Regional Córdoba. Especialización en Ingeniería en Sistemas de Información

[72] Cohn, M, A.2008. ***Mountain Goat Software*** (Disponible en: https://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template. Consultado el 2 de noviembre de 2016)

[73] Cohn, M, A.2012. ***Mountain Goat Software*** (Disponible en: http://www.mountaingoatsoftware.com/topics/user-stories. Consultado el 2 de noviembre de 2016)

[74] Cohn, M. 2009. ***User Stories Applied for Agile Software Development***. Indiana, Addisson-Wesley.

[75] W3schools.com, A. 2016 (Disponible en: http://www.w3schools.com/html/html5_webstorage.asp. Consultado el 5 de octubre de 2016)

[76] Yeoman, A. 2015 (Disponible en: http://yeoman.io/. Consultado el 2 de junio de 2016)

[77] Thomas Maximini, A. 2015 (Disponible en: https://github.com/tmaximini/generator-ionic-gulp. Consultado el 2 de junio de 2016).

[78] Yeoman, A. 2015 (Disponible en: https://github.com/yeoman/generator-angular. Consultado el 2 de junio de 2016).

[79] Express, A. 2015 (Disponible en: http://expressjs.com/es/starter/generator.html. Consultado el 2 de junio de 2016).

[80] Chacon, Scott. Straub, Ben. 2014. ***Pro Git***. Estados Unidos. Apress

[81] Roche, Emiliano. A.2013 (Disponible en https://barradevblog.wordpress.com/2013/01/21/que-es-gitgithub/. Consultado 12 de mayo de 2016)

[82] mLab, A. 2016 (Disponible en: https://mlab.com/company/. Consultado el 8 de Julio de 2016)

[83] Hugo Trigoso, A.2016 (Disponible en: http://kawcode.com/tutoriales/que-es-heroku/. Consultado el 10 de Junio de 2016)

[84] F.Javier Díaz, Ivana Harari. 2015. ***Del Telegrama a los Tweets: Investigación sobre la Interacción del Adulto Mayor con las Redes Sociales y Aplicaciones Google considerando Aspectos de Usabilidad y Accesibilidad Web***. La Plata, Argentina, Laboratorio de Investigaciones en Nuevas Tecnologías Informáticas LINTI. Fac.de Informática, Universidad Nacional de La Plata