Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

SANIDAPP: SISTEMA DE COMUNICACIÓN Y ORGANIZACIÓN PERSONAL EN UNA INSTITUCIÓN SANITARIA

Alumna: Hernández Muriel, Eguzkine

Directora: Armendariz Leunda, Ana Jesus

Curso: 2018-2019

Fecha: 22 de Julio de 2019

Resumen

SanidApp es un sistema dirigido a pacientes y profesionales sanitarios para mejorar su comunicación y organización. Esta aplicación ha sido desarrollada para realizar acciones conjuntas entre profesional-profesional y profesional-paciente, además de ofrecer una amplia información laboral sobre los expertos sanitarios, así como la posibilidad de enviar mensajes entre sus usuarios.

Este documento recoge todas las fases por las que ha pasado el proyecto Sistema de Comunicación y Organización Personal en una Institución Sanitaria. Se presenta la planificación previa a la realización del proyecto, posteriormente se analizan la captura de requisitos, el diseño y el desarrollo, que es la parte más extensa, finalizando con las pruebas, las conclusiones, que incluye una reflexión personal sobre la experiencia vivida durante el proyecto, y el trabajo futuro.

Palabras clave

Sistema de gestión sanitaria, comunicación, chat, agenda profesional.

Laburpena

SanidApp paziente eta osasun-profesionalen antolaketa eta komunikazioa hobetzeko zuzenduta dagoen sistema da. Aplikazio hau profesionalen eta pazienteen arteko jarduerak egiteko garatuta dago. Gainera, osasun-jakitunen informazio-laborala ikusteko eta aplikazio erabiltzaileen artean mezuak bidaltzeko aukera ematen du.

Dokumentu honek Osasun-Enpresa baten Komunikazio eta Antolakuntza Sistema proiektuaren atal guztiak biltzen ditu. Proiektuaren aurretiko planifikazioa aurkezten da, ondoren proiektuaren eskakizunak, diseinua eta garapena aztertzen da, parte hau luzeena izanda. Amaitzeko, probak eta ondorioak aztertzen dira, honen barruan proiektuaren garapenean izandako esperientziaren gogoeta pertsonala egiten da, baita etorkizunerako lanari buruzkoa.

Hitz gakoak

Osasun-kudeaketa sistema, komunikazioa, txat, agenda profesionala.

Summary

SanidApp is a system aimed at patients and healthcare professionals to improve their communication and organization. This application has been developed to carry out joint actions between professional-professional and patient-professional, in addition to offering a wide range of job information about health experts, as well as the possibility of sending messages between their users.

This document covers all the phases of the project Communication System and Personal Organisation in a Healthcare Institution. First of all, it presents the previous planning to the accomplishment of the project. Second of all, the capture of requirements, the design and the development are analyzed, what is the most extensive part. Finally, it finishes with the tests, conclusions, which includes a personal reflection on the experience lived during the project, and the future work.

Keywords

Health management system, communication, chat, professional agenda.

Índice de contenido

Índice de tablas

Índice de ilustraciones

1.- Introducción

¿Cuántas veces hemos pensado que el sistema sanitario está en decadencia? Es posible que, en algún momento, cualquier persona haya tenido una ráfaga de impotencia cuando piensa en la situación actual de la sanidad, cuando algún familiar o amigo está en una situación difícil y no pueden hacer nada por eso persona, porque simplemente no tienen medios o tiempo. Pero realmente, poca gente se para a pensar en qué es lo realmente necesario para esos profesionales sanitarios que están dando lo máximo para que el paciente se sane.

En los últimos 50 años, España se ha colocado en uno de los países más importantes en el ámbito de la medicina gracias a la relación obtenida en coste-eficacia **(1)**. Esto se debe a la buena organización y la gestión que se ha conseguido llevar durante el periodo de cambio desde que se impusieron los impuestos para la seguridad social, y conseguir así, una sanidad pública. Este proceso tuvo su comienzo en 1883 cuando se decidió formar una política que protegiera de alguna forma a la clase obrera, pero no tuvo una estabilidad propia, en cuanto a sanidad se refiere, hasta que se implantó definitivamente un modelo integrado de protección social, en 1963 **(2)**. Hoy en día, todos los ciudadanos contribuimos a la llamada Seguridad Social, la que se encarga de proporcionarnos sanidad y justicia entre otros servicios de forma "gratuita".

Si nos centramos en el paciente y analizamos sus necesidades podemos darnos cuenta de que la exigencia de información está más que presente. El paciente ante una enfermedad, sea leve, grave, temporal o duradera, siente la necesidad de saber la causa y el desarrollo que tendrá la misma. Por lo que la posición del médico ante la necesidad de información del paciente, deberá ser abierta y comprensiva.

Por otro lado, cada vez son más frecuentes las citas telefónicas entre paciente y profesional sanitario (normalmente enfermeros y médicos). Estas consultas son accesibles desde la página web de los propios servicios (públicos). Con esto, se está consiguiendo que las listas de espera no sean tan largas y así optimizar el tiempo y los resultados de las personas que necesiten una atención especial y en persona.

Basándonos en el hecho de que la comunicación es indispensable para una buena y óptima organización, he querido desarrollar un sistema en el que esta sea más fluida y natural teniendo en cuenta en la era en la que vivimos, la Tecnológica.

2.- Descripción de Objetivos del Proyecto

Dentro de esta sección, se encuentran todos los apartados a tener en cuenta antes de comenzar con el proyecto. Se basa en llevar una organización para conseguir llegar a los objetivos planteados en un tiempo determinado. Se hace una descripción de los objetivos del trabajo a realizar, las características que tendrá la aplicación, el plan de desarrollo (en qué partes se repartirá el proyecto), la planificación temporal, estimación de recursos, estimación económica y prevención de riesgos.

2.1.- Objetivos

El objetivo de este proyecto es el de optimizar la gestión y el tiempo de trabajo de los profesionales sanitarios. Tras llevar una investigación para conseguir ver las necesidades actuales de estos trabajadores, se ha encontrado con que la comunicación dentro de las instituciones sanitarias es bastante escasa. Así mismo, queriendo reforzar esta faceta de la sanidad, se propone el siguiente proyecto.

Hasta ahora, la forma de comunicación dentro de un hospital, ambulatorio, centro de salud o residencia ha sido de forma electrónica o personal, es decir, los profesionales pueden mandar e-mails o comunicarse cara a cara. Para optimizar la forma electrónica de este sistema, este proyecto se caracteriza por ser una aplicación de mensajería instantánea, entre otras.

Con esta aplicación se podrán enviar y recibir mensajes de forma instantánea. También los usuarios registrados tendrán acceso a todos esos profesionales que estén dados de alta en el sistema y podrán seleccionar al profesional sanitario para ver su trayectoria, especialidad y lugar de trabajo actual.

Además del chat entre dos profesionales, se podrán crear eventos con el objetivo de hacer preparatorios para pruebas, operaciones y actividades en el que haga falta una organización entre varios trabajadores. *Ejemplo: Estamos en la semana anterior a una operación importante y el equipo se necesita juntar para tomar apuntes de la organización tanto de material, como de personal. Existe la posibilidad de que algunas personas no asistan a esa reunión por el simple hecho de que no se han enterado. Creando el evento y vinculándolo a esos profesionales se podrá llevar a cabo la convocatoria para la reunión, donde se indique el lugar, el momento, la duración y las personas que deberán asistir.*

Los eventos, podrán ser de cualquier índole, ya que estos también podrán compartirse con pacientes. Podría darse el caso que el profesional sanitario quiera informar al paciente de que al siguiente día tiene una cita programada, por ello le vinculará al evento haciendo que le aparezca la nueva cita en el perfil del paciente.

En cuanto a los perfiles de los usuarios, serán totalmente públicos a los demás usuarios de la aplicación. Los usuarios profesionales tendrán acceso a otros perfiles profesionales y podrán visualizar la información sobre su trayectoria, entre otras. Aunque, esa información queda en mano de cada profesional, ya que será opcional incluirla en la aplicación. Por otro lado, los perfiles de pacientes solo estarán disponibles para los usuarios profesionales, con la finalidad de que estos puedan contactar con los pacientes en un momento necesario.

2.2.- Características de la aplicación

Para poder acceder a la aplicación, es indispensable que la persona quiera hacerlo y esté en disposición de un número de trabajador facilitado por la institución servidora de Salud o sea paciente de esta institución.

2.2.1.- Datos necesarios para el acceso

Los datos necesarios para poder acceder a la aplicación son los siguientes:

- Código del trabajador.
- Nombre y apellidos.
- E-mail.
- Contraseña nueva para el futuro acceso.

2.2.2.- Características

Las características de las funciones que tiene que cumplir la aplicación propuesta son:

- Buscador de contactos Profesionales:
    - Por nombre.
    - Por apellidos.
    - Lugar de trabajo actual.
    - Especialidad sanitaria.

- Buscador de contactos Pacientes:
    - Por nombre.
    - Por apellidos

- Chat
    - Profesional - Profesional.
    - Profesional – Paciente.
    - Paciente – Profesional.
    - Únicamente se comparten mensajes instantáneos.

- Eventos:
    - Nombre
    - Lugar
    - Descripción
    - Fecha
    - Hora de comienzo
    - Vincular usuarios.

- Calendario
    - Eventos vinculados al usuario actual.
    - Visualizar la información completa expuesta por el creador del evento.

- Perfil Profesional
    - Foto
    - Nombre y apellidos
    - E-mail.
    - Trayectoria *(Opcional)*
    - Lugar de trabajo actual (*Opcional)*
    - Especialidad *(Opcional)*

- Perfil Paciente
    - Foto
    - Nombre y apellidos
    - E-mail.

2.3.- Plan de desarrollo

Este proyecto se desarrollará mediante la metodología **Scrum**. Esta metodología se basa en ponerse objetivos en tiempos razonables para poder cumplirlos. No se trata de objetivos definitivos, ya que puede que ocurran dificultades o cambios a lo largo del proyecto, tanto en implementación, pruebas, estudio y/o documentación.

Esta metodología consta de diferentes etapas dispuestas por el creador del proyecto para llevar una mejor organización en cuanto a tiempo/objetivos. Aun así, la parte de implementación, pruebas y prototipos puede cambiar debido al desarrollo propio del proyecto (cambio de especificaciones, etc.).

En caso de no poder finalizar un **Sprint** a tiempo (esto es muy probable que ocurra en el momento de la implementación y las pruebas), se sumará tiempo a ese objetivo y se retrasarán los objetivos que se dispongan después de este. Aunque habrá que tener en cuenta que no exceda del tiempo total que hay para desarrollar el proyecto. Estos objetivos se plantean al principio del proyecto.

Por lo tanto, el proyecto se dividirá en tres grandes grupos: diseño, implementación y pruebas. En el primer grupo, entrará toda la realización de los diseños necesarios para la implementación; en el segundo grupo, la implementación. Basándose en los diseños creados anteriormente, se implementará la aplicación de forma más fácil. Finalmente, en el grupo de pruebas se realizarán las pruebas necesarias en cuanto a programación, usabilidad, etc. para comprobar que la aplicación funciona realmente.

La planificación del proyecto se repartirá según la siguiente Estructura de Descomposición del proyecto *Ilustración 1: EDT*.

2.3.1.- Estructura de Descomposición del Trabajo (EDT)

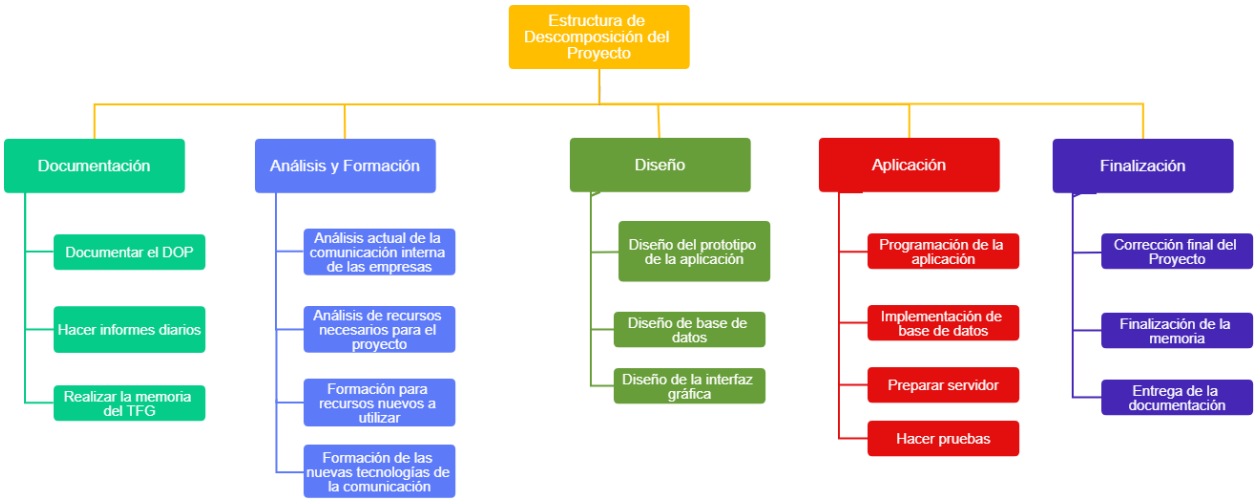

Ilustración 1: EDT

2.3.2.- Descripción de actividades del EDT

En este apartado se explican las actividades propuestas en la *Ilustración 1: EDT*.

Documentación

A continuación se detallan las actividades propuestas para la documentación.

Documento de Objetivos del Proyecto (DOP)
Actividad: Crear y desarrollar el documento de objetivos del proyecto **Tiempo de trabajo estimado:** 40 horas **Descripción:** Este documento recoge los objetivos principales del proyecto. La realización, el tiempo necesario y la gestión de cada uno. **Entrada:** - **Salida:** Documento de Objetivos del Proyecto (DOP) **Recursos necesarios:** Un ordenador y acceso a internet (Drive, Microsoft Word, etc.).

Tabla 1: DOP

Realizar informes diarios
Actividad: Hacer apuntes diarios donde queda escrito lo que se ha hecho cada día. **Tiempo de trabajo estimado:** 15 min / día **Descripción:** Estos informes se harán para realizar una memoria del proyecto más exacta. En cuanto a tiempo real dedicado a los objetivos dispuestos. **Entrada:** - **Salida:** Apuntes diarios **Recursos necesarios:** Un ordenador y acceso a internet (Drive, Microsoft Word, etc.).

Tabla 2: Realizar informes diarios

Realizar la memoria del Proyecto
Actividad: Realizar una memoria para el Trabajo de Fin de Grado **Tiempo de trabajo estimado:** 82 horas **Descripción:** En este documento se recogerán todos los detalles técnicos y teóricos del proyecto. **Entrada:** - **Salida:** Memoria del TFG **Recursos necesarios:** Un ordenador y acceso a internet (Drive, Microsoft Word, Cacoo etc.).

Tabla 3: Realizar la memoria del Proyecto

Análisis y Formación

Mediante estas tablas se explican las actividades propuestas para el análisis y formación del proyecto.

Análisis de la comunicación interna actual de las instituciones
Actividad: Hacer un análisis de la comunicación interna de las instituciones. **Tiempo de trabajo estimado:** 10 horas **Descripción:** Se hará un análisis de las carencias y fortalezas de la actual comunicación interna en instituciones, sobre todo en sanidad. **Entrada:** - **Salida:** Introducción de la memoria. **Recursos necesarios:** Un ordenador y acceso a internet (Drive, Microsoft Word, etc.).

Tabla 4: Análisis de la comunicación

Análisis de recursos necesarios para el proyecto
Actividad: Hacer un análisis de los recursos necesarios a utilizar **Tiempo de trabajo estimado:** 5 horas **Descripción:** Hacer un estudio para saber qué tipo de tecnologías serán necesarias para la realización del proyecto y conseguirlas. **Entrada:** - **Salida:** Los recursos necesarios. **Recursos necesarios:** Un ordenador, recursos económicos y acceso a internet.

Tabla 5: Análisis de recursos

Formación para recursos nuevos a utilizar
Actividad: Aprender a usar los nuevos recursos. **Tiempo de trabajo estimado:** 9 horas **Descripción:** En caso de necesitar recursos no usados hasta el momento, así como, programas nuevos, lenguajes no estudiados, formarse para su buena utilización. **Entrada:** Recursos nuevos **Salida:** Aprendizaje de los nuevos recursos. **Recursos necesarios:** Un ordenador, recursos económicos y acceso a internet.

Tabla 6: Formación para recursos

Formación en tecnologías de la comunicación
Actividad: Aprender sobre diferentes tecnologías de la comunicación. **Tiempo de trabajo estimado:** 10 horas **Descripción:** Leer sobre los diferentes métodos de comunicación electrónicos, para reforzar las facetas que más carencias se encuentren. **Entrada:** Artículos sobre comunicación electrónica **Salida:** Aprendizaje recursos necesarios para la comunicación. **Recursos necesarios:** Un ordenador y acceso a internet.

Tabla 7: Formación en tecnologías de la comunicación

Diseño

Estas tablas explican las actividades en cuanto a diseño previstas para el proyecto.

Diseño de prototipos de la aplicación
Actividad: Diseñar el manejo de la aplicación **Tiempo de trabajo estimado:** 7 horas **Descripción:** Teniendo en cuenta diferentes factores, como la facilidad de entendimiento que debe tener un programa, hacer un diseño de la usabilidad de la aplicación. **Entrada:** Programa necesario para el simulacro o papel y lápiz. **Salida:** Diseño de usabilidad del proyecto (Aplicación) **Recursos necesarios:** Un ordenador, acceso a internet, papel y lápiz.

Tabla 8: Diseño de prototipos

Diseño de bases de datos
Actividad: Diseñar la base de datos **Tiempo de trabajo estimado:** 10 horas **Descripción:** Se hará un diagrama para ver la necesidad de algunas entidades y la dependencia que surgen entre ellas. **Entrada:** **Salida:** Diagrama de base de datos **Recursos necesarios:** Un ordenador, Cacoo y acceso a internet.

Tabla 9: Diseño de bases de datos

Diseño de la interfaz gráfica
Actividad: Diseñar la interfaz gráfica de la aplicación. **Tiempo de trabajo estimado:** 7 horas **Descripción:** Después de definir qué tipo de público usará la aplicación, se hará un diseño de la interfaz gráfica y se probará con diferentes personas. **Entrada:** **Salida:** Diseño de interfaz gráfica. **Recursos necesarios:** Un ordenador y acceso a internet o papel y lápiz.

Tabla 10: Diseño de la interfaz

Aplicación

En esta sección se explican las tablas de las actividades relacionadas con la aplicación.

Programación de la aplicación
Actividad: Implementación de la aplicación a desarrollar. **Tiempo de trabajo estimado:** 183 horas **Descripción:** Después de definir qué tipo de público usará la aplicación, se hará un diseño de la interfaz gráfica y se probará con diferentes personas. **Entrada:** - **Salida:** Programación de la aplicación **Recursos necesarios:** Un ordenador, Eclipse + plugins, servidor y acceso a internet.

Tabla 11: Programación de la aplicación

Implementación de base de datos
Actividad: Implementar la base de datos. **Tiempo de trabajo estimado:** 10 horas **Descripción:** Según el diseño realizado anteriormente, implementar la base datos. **Entrada:** Diseño de base de datos **Salida:** Base de datos implementada **Recursos necesarios:** Un ordenador, programa MySQL y acceso a internet.

Tabla 12: Implementación de base de datos

Implementación gráfica de la aplicación
Actividad: Implementar la parte de gráfica de la aplicación **Tiempo de trabajo estimado:** El tiempo dedicado a esta parte se complementa con *Programación de la aplicación*. **Descripción:** Según el diseño de la parte gráfica, realizar la implementación juntando la programación de la propia aplicación y la gráfica. **Entrada:** Diseño de la interfaz gráfica **Salida:** Implementación de la interfaz gráfica **Recursos necesarios:** Un ordenador, diseño de la interfaz, implementación de la aplicación, servidor y acceso a internet.

Tabla 13: Implementación gráfica de la aplicación

Preparar servidor
Actividad: Preparar el servidor a utilizar **Tiempo de trabajo estimado:** 20 horas **Descripción:** Se construye un servidor mediante Sockets implementado en Eclipse. **Entrada:** - **Salida:** Servidor listo para la simulación. **Recursos necesarios:** Un ordenador, acceso a internet.

Tabla 14: Preparar servidor

Hacer pruebas
Actividad: Hacer pruebas de la aplicación. **Tiempo de trabajo estimado:** 95 horas **Descripción:** Hacer pruebas tanto de la implementación de la aplicación, como de la interfaz definitiva. **Entrada:** Aplicación programada. **Salida:** Resultados de diferentes pruebas realizadas. **Recursos necesarios:** Un ordenador, Eclipse, aplicación terminada y acceso a internet.

Tabla 15: Hacer pruebas

Finalización

Las siguientes tablas describen las actividades relacionadas con la finalización del proyecto.

Corrección final del proyecto
Actividad: Corrección final de la memoria del Trabajo de Fin de Grado con la tutora. **Tiempo de trabajo estimado:** 2 horas **Descripción:** El trabajo se corrige por última vez con la tutora. **Entrada:** Memoria del Trabajo de Fin de Grado **Salida:** Memoria del Trabajo de Fin de Grado. **Recursos necesarios:** Un ordenador, TFG y acceso a internet.

Tabla 16: Corrección final del proyecto

Finalización de la memoria
Actividad: Finalización de la memoria por parte del alumno. **Tiempo de trabajo estimado:** 10 horas **Descripción:** El trabajo se corrige por última vez. **Entrada:** Memoria del Trabajo de Fin de Grado **Salida:** Memoria del Trabajo de Fin de Grado. **Recursos necesarios:** Un ordenador, TFG y acceso a internet.

Tabla 17: Finalización de la memoria

Entrega de la documentación
Actividad: Entregar la documentación completa del proyecto **Tiempo de trabajo estimado:** 2 horas **Descripción:** En esta documentación entran todos los diagramas necesarios para la explicación de la implementación, interfaz gráfica, etc. **Entrada:** Diagramas, interfaz gráfica. **Salida:** Documentación del proyecto. **Recursos necesarios:** Un ordenador, todos los diseños definitivos y acceso a internet.

Tabla 18: Entrega de la documentación

2.4.- Planificación temporal

En esta parte de la documentación, se ha hecho un desglose de las horas necesarias para la realización completa del proyecto. Más adelante se explicará mediante unas tablas en qué orden se ha de realizar cada tarea y el tiempo total que llevará cada una. Además, podremos saber según el tiempo estimado de cada actividad, el tiempo extra del que se dispondrá. Con todo esto, se conseguirá organizar las actividades, calcular las rutas óptimas de trabajo, las dependencias que pueden surgir entre diferentes tareas tratando así, cada actividad de manera individual y su relación con las demás. Esto ayudará a la toma decisiones anticipadas y efectivas, prevención de errores y una mayor integración conjunta de las actividades.

La planificación temporal está dividida en dos partes principales: las actividades iniciales y finales y las basadas en la metodología **Scrum**. Esta última parte, son ciclos completos (diseño + implementación + pruebas) de diferentes actividades que debe contener la aplicación. Por lo tanto, el flujo final del proyecto es **Actividad** - **Scrum** - **Actividad**.

Por otro lado, se explicará más adelante, mediante un diagrama de Gantt, el tiempo de cada actividad y el tiempo total para calcular la fecha estimada de la finalización del proyecto.

2.4.1.- Planificación en horas

A continuación, se presenta la planificación realizada en horas.

Actividad

Se visualiza la siguiente tabla con el fin de saber la cantidad de horas necesarias para las primeras tareas del proyecto.

Actividad	**Estimación (Horas)**
1.- Inicio	**26**
1.1.- Planteamiento del tema 1.2.- Selección del Software a utilizar 1.3.- Instalación del Software 1.4.- Reunión con la tutora	10 5 9 2
2.- Formación	**13**
2.1.- Charlas con personas ligadas al proyecto 2.2.- Investigación sobre el tema	3 10
3.- Documento de Objetivos del Proyecto	**42**
3.1.- Creación del documento 3.2.- Reunión con la tutora	40 2
Total horas 1ª Actividad	**81**

Tabla 19: Actividad 1 estimación de horas

Scrum

Esta parte es la más complicada y que más tiempo llevará realizarla. El desarrollo del proyecto se centra en esta parte, ya que cada apartado es un ciclo completo que consta de diseño, implementación y pruebas.

1.- Registrar un Usuario	**Estimación (Horas)**
1.1.- Diseño	**7**
● Diseño a papel de la base de datos ○ Tabla Usuario ○ Tabla UsuarioPaciente ○ Tabla UsuarioProfesional ○ Tabla Profesionales ○ Tabla Pacientes ○ Tabla FotoSistema ○ Tabla FotoUsuario ● Diseño a papel de las ventanas ○ Login ○ Nueva Cuenta	**5** **2**
1.2.- Implementación	**20.5**
● Implementar las tablas de la base de datos. ● Crear página "Login". ● Crear página "NuevaCuenta". ● Comprobar que el usuario no está previamente registrado. ● Comprobar que el profesional está en la BD de "Profesionales". ● Comprobar que el paciente está en la BD de "Pacientes". ● Guardar los datos del nuevo Usuario	**5** **7** **5** **0.5** **0.5** **0.5** **2**
1.3.- Pruebas	**5**
● Intentar registrar un Usuario, tanto "Profesional" como "Paciente", que no esté en la BD de la institución. ● Intentar registrar un Usuario, tanto "Profesional" como "Paciente" que ya esté registrado en SanidApp. ● Registrar un usuario "Profesional" ● Registrar un usuario "Paciente". ● Intentar registrar un usuario sin los datos cumplimentados (formulario). ● Comprobar el correcto funcionamiento de los botones.	
TOTAL	**32.5**

Tabla 20: Registrar un usuario estimación de horas

2.- Inicio	**Estimación (Horas)**
2.1.- Diseño	**0.5**
● Diseño a papel de las ventanas ○ Inicio	**0.5**
2.2.- Implementación	**12**
● Crear la ventana “Inicio”. ● Poner los botones a que redirijan a otra ventanas. ● Cargar el nombre de la persona que ha ingresado en la App. ● El botón *Calendario* abra la ventana “Calendario”. ● El botón *Notificaciones* abra la ventana “Notificaciones”. ● El botón *Buscar Contactos Profesionales* abra la ventana para buscar a contactos profesionales. ● El botón *Buscar Contactos Pacientes* abra la ventana para buscar a contactos pacientes. ● El botón Mi Perfil redirija al usuario a ver y editar su perfil. ● Guardar en una clase única del sistema los datos del profesional.	**5** **1** **0.5** **0.1** **0.1** **0.1** **0.1** **0.1** **5**
2.3.- Pruebas	**10**
● Enseñar en un JLabel el nombre de la persona que ha ingresado. ● Cada botón del JPanel principal lleve a su ventana correspondiente. ● Al entrar en una ventana no se pierden los datos del Usuario.	
TOTAL	**22.5**

Tabla 21: Inicio estimación de horas

3.- Buscar Contacto	**Estimación (Horas)**
3.1.- Diseño	**1**
● Diseño a papel de las ventanas ○ BuscarContacto	**1**
3.2.- Implementación	**8.5**
● Crear Panel de Filtros. ● Crear JTable donde se ven los resultados de la búsqueda. ● Ver la Imagen de perfil del contacto en JTable de búsqueda. ● Hacer la búsqueda correctamente. ● Ver Contacto	**3** **3** **0.5** **1.5** **0.5**
3.3.- Pruebas	**4.5**
● Buscar un contacto sin filtros y que te enseñe todos los contactos. ● Buscar un contacto con filtros y que los resultados cuadren. ● Darle al botón de buscar y que el JTable se limpie y aparezca la nueva búsqueda.	
TOTAL	**14**

Tabla 22: Buscar Contacto estimación de horas

4.- Ver Mi Perfil	**Estimación (Horas)**
4.1.- Diseño	**0.5**
● Diseño a papel de las ventanas ○ Ver Mi Perfil	**0.5**
4.2.- Implementación	**13.5**
● Visualizar correctamente los datos. ● Visualizar la foto de perfil ● Implementación para cambiar foto de perfil. ● Las imágenes han de ser guardadas en la base de datos. ● Añadir Trayectoria. ● Cuando se añada una "Trayectoria" que salte un mensaje de precaución.	**5** **2** **2** **2** **2** **0.5**
4.3.- Pruebas	**10**
● Los datos que se visualizan son los del usuario que ha ingresado. ● Para cambiar la foto se abre un navegador de archivos. ● Si se escoge una foto se reemplaza al momento. ● Si no se escoge ninguna foto, se queda igual. ● La imagen se guarda en la base de datos. ● Guardar los campos escritos correctamente en la base de datos. ● Al añadir trayectoria te salga el mensaje de precaución. ● Añada bien la trayectoria y aparezca bien en la lista al momento.	
TOTAL	**24**

Tabla 23: Ver Perfil estimación de horas

5.- Calendario	**Estimación (Horas)**
4.1.- Diseño	**3**
● Diseño a papel de las ventanas ○ Ver Eventos ○ Añadir Eventos ● Diseño en base de datos ○ Tabla Eventos ○ Tabla RelacionEventos	**1** **2**
4.2.- Implementación	**22**
● Visualizar correctamente toda la pantalla. ● Selección de datos en la base de datos usando filtros. ● Selección de datos sin filtros. ● Añadir evento. ● Vinculación del evento con otros usuarios ● Estudio de modo de utilización de JCalendar. ● Vinculación en dependencias de JCalendar.	**5** **5** **3** **2** **4** **2** **1**
4.3.- Pruebas	**10**
● Visualizar únicamente los eventos del usuario actual. ● Añadir correctamente los eventos nuevos. ● Vincular correctamente a los contactos.	
TOTAL	**35**

Tabla 24: Calendario estimación de horas

5.- Chat	**Estimación (Horas)**
5.1.- Diseño	**2**
● Diseño de la base de datos la tablas: ○ Chat ○ Mensaje ● Diseño a papel de las ventanas ○ Chat	**1** **1**
5.2.- Implementación	**34**
● Mandar un mensaje mediante servidor. ● Visualizar al momento el mensaje enviado en pantalla. ● Guardar mensaje en base de datos. ● Guardar mensaje en Notificaciones ● Abrir canales para recibir mensajes. ● Implementar servidor (hilo mensajes)	**10** **2** **5** **2** **10** **5**
5.3.- Pruebas	**15**
● Servidor recibe el paquete completo. ● Servidor guarda correctamente los datos en la base de datos. ● Servidor identifica al receptor. ● Servidor prepara el paquete para redirigirlo. ● Servidor manda el paquete. ● Usuario recibe el mensaje enviado por otro usuario. ● Se visualiza en la pantalla correctamente.	
TOTAL	**51**

Tabla 25: Chat estimación de horas

5.- Notificaciones	**Estimación (Horas)**
5.1.- Diseño	**2.5**
● Diseño de la base de datos la tablas: ○ Notificación ● Diseño a papel de las ventanas ○ Notificaciones	**2** **0.5**
5.2.- Implementación	**11.5**
● Implementación de pantalla. ● Creación del modelo Notificaciones para panel de resultados. ● Implementar sentencias SQL. ● Vincular con pantalla Chat. ● Ejecutar sentencias en servidor.	**2** **5** **2** **0.5** **2**
5.3.- Pruebas	**6**
● Usuario pueda recibir en cualquier momento notificaciones ● Tras pulsar la notificación abra el chat correspondiente al mensaje. ● Si el servidor no puede redirigir el mensaje al usuario receptor, ejecute la sentencia contra la tabla Notificación. ● El modelo visualice correctamente los datos.	
TOTAL	**20**

Tabla 26: Notificaciones estimación de horas

Es necesario apuntar, que algunos hitos de la metodología Scrum se han tenido que repetir dos veces, uno para la parte de profesional y otra para la parte de paciente, porque no se ha previsto que se pueda hacer de otra manera. Por lo tanto, las horas de implementación y de pruebas de los hitos Inicio, Buscar Contacto, Ver Perfil y Calendario se han de duplicar.

Inicio (12h + 10h) x 2 = 44 horas.

Buscar Contacto (8.5h + 4.5h) x 2 = 26 horas

Ver Mi Perfil (13.5h + 10) x 2 = 47 horas

Calendario (22h + 10h) x 2 = 64 horas

Las horas finales de la parte Scrum es: 279.5 horas

Actividad

A continuación, en esta tabla se exponen las horas necesarias para realizar las últimas tareas del proyecto.

Actividad	Estimación (Horas)
6.- Documentación final	**82**
6.1.- Hacer un manual de instrucciones 6.2.- Acabar de documentar la memoria 6.3.- Reunión con la tutora	10 70 2
7.- Finalización del proyecto	**22**
7.1.- Corrección final del proyecto 7.2.- Pruebas definitivas 7.3.- Entrega del proyecto 7.4.- Reunión con la tutora	10 8 2 2
8.- Total de horas	**464.5**

Tabla 27: Actividad 2 estimación de horas

2.4.2.- Planificación en días

En esta sección de la planificación se estructuras las tareas en días para saber qué funciones ha de realizar el desarrollador en cada momento.

Actividad

En esta tabla se muestra la planificación en días de las tareas que corresponden a la primera actividad.

Actividad	Fecha de inicio	Fecha de finalización
1.- Inicio	**15/04/2019**	**17/04/2019**
1.1.- Planteamiento del tema 1.2.- Selección del software 1.3.- Instalación del software 1.4.- Reunión con la tutora	15/04/2019 16/04/2019 16/04/2019 17/04/2019	15/04/2019 16/04/2019 16/04/2019 17/04/2019
2.- Formación	**18/04/2019**	**21/04/2019**
2.1.- Charlas con las personas ligadas al proyecto 2.2.- Investigación sobre el tema	18/04/2019 18/04/2019	19/04/2019 21/04/2019
3.- Documentos de Objetivos del Proyecto	**22/04/2019**	**06/05/2019**
3.1.- Creación del documento 3.2.- Reunión con la tutora	22/04/2019 06/05/2019	29/04/2019 06/05/2019
8.- Estimación total	**15/04/2019**	**6/05/2019**

Tabla 28: Actividad 1 planificación de días

Scrum

A continuación se muestra la planificación en días de la parte realizada con la metodología Scrum.

Registrar un usuario	Fecha de inicio	Fecha de finalización
Diseño	**07/05/2019**	**09/05/2019**
Implementación	**10/05/2019**	**15/05/2019**
Pruebas	**16/05/2019**	**16/05/2019**

Tabla 29: Registrar un usuario planificación de días

Inicio	Fecha de inicio	Fecha de finalización
Diseño	**17/05/2019**	**17/05/2019**
Implementación Profesional	**17/05/2019**	**20/05/2019**
Pruebas Profesional	**21/05/2019**	**22/05/2019**
Implementación Paciente	**23/05/2019**	**26/05/2019**
Pruebas Paciente	**26/05/2019**	**27/05/2019**

Tabla 30: Inicio, planificación de días

Buscar Contacto	Fecha de inicio	Fecha de finalización
Diseño	**28/05/2019**	**28/05/2019**
Implementación Profesional	**28/05/2019**	**29/05/2019**
Pruebas Profesional	**30/05/2019**	**30/05/2019**
Implementación Paciente	**31/05/2019**	**01/06/2019**
Pruebas Paciente	**02/06/2019**	**02/06/2019**

Tabla 31: Buscar Contacto planificación de días

Ver mi Perfil	Fecha de inicio	Fecha de finalización
Diseño	**03/06/2019**	**03/06/2019**
Implementación Profesional	**03/06/2019**	**05/06/2019**
Pruebas Profesional	**06/06/2019**	**07/06/2019**
Implementación Paciente	**08/06/2019**	**10/06/2019**
Pruebas Paciente	**10/06/2019**	**11/06/2019**

Tabla 32: Ver mi Perfil planificación de días

Calendario	Fecha de inicio	Fecha de finalización
Diseño	**12/06/2019**	**12/06/2019**
Implementación	**12/06/2019**	**15/06/2019**
Pruebas	**16/06/2019**	**17/06/2019**

Tabla 33: Calendario planificación de días

Chat	Fecha de inicio	Fecha de finalización
Diseño	**18/06/2019**	**18/06/2019**
Implementación	**18/06/2019**	**23/06/2019**
Pruebas	**23/06/2019**	**25/06/2019**

Tabla 34: Chat planificación de días

Notificaciones	Fecha de inicio	Fecha de finalización
Diseño	**25/06/2019**	**25/06/2019**
Implementación	**26/06/2019**	**27/06/2019**
Pruebas	**28/06/2019**	**28/06/2019**

Tabla 35: Notificaciones planificación de días

Actividad

Finalmente, se estructuran los días para la realización de las últimas tareas del proyecto.

Actividad	Fecha de inicio	Fecha de finalización
Documentación Final	**29/06/2019**	**10/07/2019**
Hacer un manual de instrucciones Acabar de documentar la memoria Reunión con la tutora	29/06/2019 01/07/2019 10/07/2019	30/06/2019 09/07/2019 10/07/2019
Finalización del Proyecto	**10/07/2019**	**19/07/2019**
Corrección final del proyecto Pruebas definitivas Entrega del proyecto Reunión con la tutora	10/07/2019 12/07/2019 19/07/2019 19/07/2019	11/07/2019 13/07/2019 19/07/2019 19/07/2019

Tabla 36: Actividad 2 planificación de días

2.4.3.- Planificación en días en diagrama Gantt

A continuación se hace una representación gráfica a través de diagramas Gantt, los cuales sirven para representar la planificación definida anteriormente.

Actividad 1

Este es el diagrama Gantt del periodo de la Actividad 1.

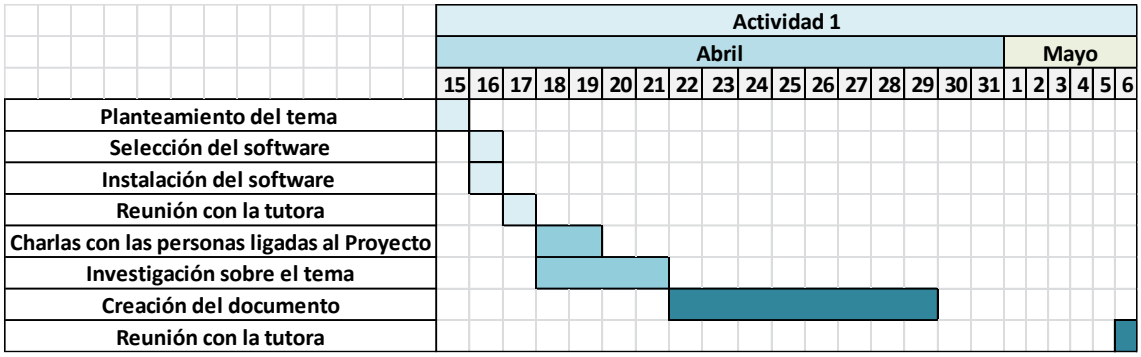

Ilustración 2: Actividad 1 Gantt

Scrum

A continuación se presentan los diagramas Gantt correspondientes a la planificación Scrum.

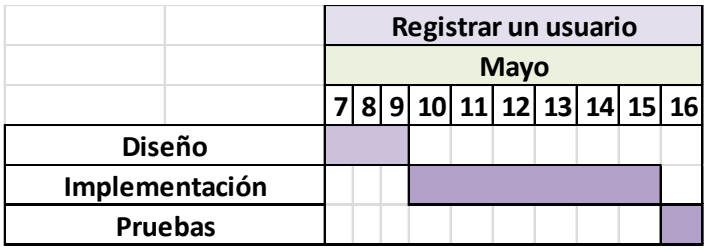

Ilustración 3: Registrar un usuario Gantt

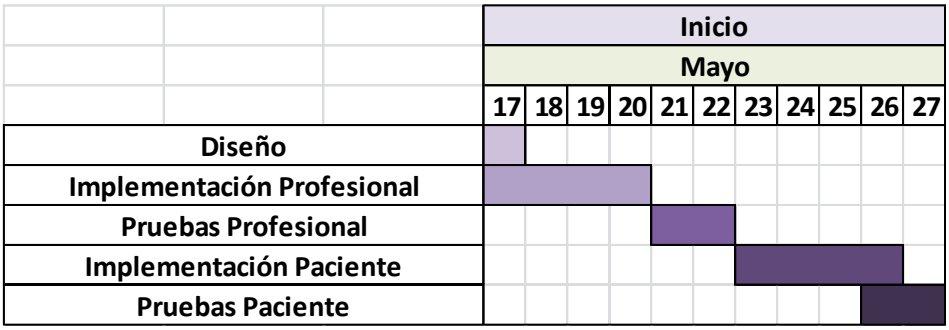

Ilustración 4: Inicio Gantt

	Buscar Contacto					
	Mayo				Junio	
	28	29	30	31	1	2
Diseño						
Implementación Profesional						
Pruebas Profesional						
Implementación Paciente						
Pruebas Paciente						

Ilustración 5: Buscar Contacto Gantt

	Ver mi Perfil								
	Junio								
	3	4	5	6	7	8	9	10	11
Diseño									
Implementación Profesional									
Pruebas Profesional									
Implementación Paciente									
Pruebas Pacientes									

Ilustración 6: Ver mi Perfil Gantt

	Calendario					
	Junio					
	12	13	14	15	16	17
Diseño						
Implementación						
Pruebas						

Ilustración 7: Calendario Gantt

	Chat							
	Junio							
	18	19	20	21	22	23	24	25
Diseño								
Implementación								
Pruebas								

Ilustración 8: Chat Gantt

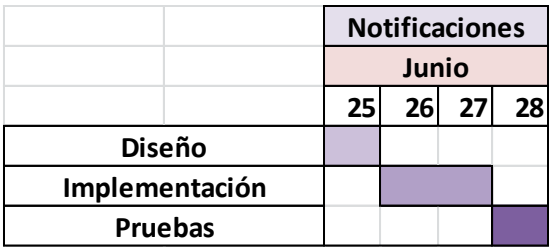

Ilustración 9: Notificaciones Gantt

Actividad 2

Este es el diagrama Gantt correspondiente a las tareas marcadas en la Actividad 2.

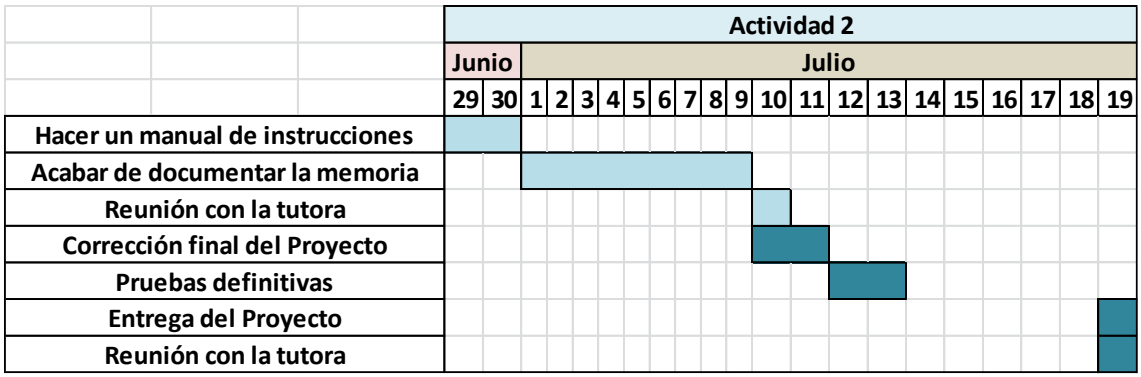

Ilustración 10: Actividad 2 Gantt

2.5.- Estimación de recursos

En este apartado se estudiará la necesidad de los diferentes recursos a utilizar en este proyecto. Se incluirán todos los necesarios para realizar las actividades planteadas anteriormente.

- Un ordenador: El ordenador será el recurso base para usar todos los demás. En este dispositivo se instalarán todos los programas necesarios para programar, diseñar los diagramas, las bases de datos, crear los documentos necesarios para la memoria…

- Internet: Es el segundo recurso indispensable que se necesitará para poder acceder a los siguientes. En la red se encuentra acceso a todos los recursos disponibles para poder programar, crear diagramas y realizar los documentos necesarios.

- Eclipse: La programación se hará con esta herramienta. Como el lenguaje que se utilizará para hacer la aplicación es Java, este recurso es muy intuitivo ya que tiene muchas opciones para utilizar ejemplos que otros programas no tienen.

Ilustración 11: Eclipse logo

- Cacoo: Es un programa gratuito en línea. Se usará, básicamente, para crear diagramas (diagrama de clases, base de datos, flujos…). Es muy fácil de usar, aparte de que te da la opción de personalizar plantillas que serán parecidas a los diagramas que se necesitarán.

Ilustración 12: Cacoo logo

- GitHub: Se utilizará para guardar el proyecto. Esta herramienta permite crear ramas para mantener diferentes versiones del proyecto. Así cuando se haga una parte del proyecto (y sea definitiva) se podrá converger con la parte principal del proyecto y así ir creando la aplicación a desarrollar por partes.

Ilustración 13: GitHub logo

- MySQL Worckbench: Este programa se usará para crear las bases de datos necesarias para el proyecto. Es una programa que crea bases de datos relacionales, además, da la opción de crearlas a mano, sin ningún tipo de lenguaje necesario (sin comandos). Este programa es muy útil para acceder a las bases de datos mediante Eclipse.

Ilustración 14: MySQL logo

- Plugin Swing para Eclipse **(3)**: Es un paquete necesario para la edición de pantallas en el programa Eclipse. Este plugin permite implementar funciones en el desarrollo del proyecto para hacer más fácil la programación de las pantallas.

- Microsoft Excel: Este programa se han utilizado para realizar los diagramas Gantt que se han expuesto anteriormente en la planificación temporal.

Ilustración 15: Excel logo

- Microsoft Word: Esta herramienta es un editor de texto. Se ha utilizado para realizar y documentar la memoria del proyecto.

Ilustración 16: Word logo

2.6.- Estimación económica

En este apartado se tendrá en cuenta todos los recursos a utilizar para calcular sus costes. Dentro de los recursos entran, el coste de las herramientas, el sueldo que un trabajador cobraría por este proyecto (teniendo en cuenta la estimación temporal planteada) y otros recursos.

- Coste personal

  Tras hacer un breve estudio de los salarios actuales e identificar qué tipo de profesionales son los necesarios para desarrollar este proyecto, se ha concluido que el coste del desarrollador del proyecto es de 25€/hora.

  Por lo tanto, teniendo en cuenta que el proyecto tendrá una duración de 4 meses, y las horas previstas para la creación completa y funcionamiento de este serán 464.5 horas, el coste personal será de **11.612,5€**.

- Tecnología Software y Hardware

  Todos los recursos Software planteados que se van a utilizar en este proyecto, serán gratuitos, por lo que no supondrán coste ninguno. En cambio, las herramientas Hardware a utilizar sí que tendrán un coste.

  El ordenador que se usará para el proyecto es un *Lenovo Idea G50-80 i7-5500U* cuyo coste fue de **555€**. Teniendo en cuenta que la vida para este recurso es de unos 4 años y que el proyecto tendrá una duración de 4 meses, el coste proporcional que se le atribuye al trabajo por este recurso es de **46.25€.**

- Otros gastos

  Otros gastos a tener en cuenta son la luz, la red contratada para internet y el móvil y el coste del transporte.

  En cuanto a la luz, se tendrá en cuenta la tarifa que se tiene contratada en el lugar de trabajo habitual donde se desarrollará el proyecto. Aquí entrará la carga del ordenador, el acceso a la red de internet, y la luz artificial. En este caso se paga *0.12158 €/kWh* [precio Endesa], por lo que teniendo en cuenta que, de media, se trabajará 4 horas diarias y que el proyecto tendrá una duración de 4 meses, el precio final en cuanto a luz para todo el proyecto será de **58'3584€**.

  Por otro lado, la fibra contratada de internet es de 120€/mes. En este precio entran 4 líneas móviles, televisión, fibra óptica y línea fija. Teniendo en cuenta que sólo se utilizarán una línea móvil y la red a internet, el precio será de 80€/ mes (consultar mejor más adelante), por lo que el presupuesto de luz para los 4 meses será de **240€**.

  Finalmente, teniendo en cuenta se irán una media de 3 veces al mes a la universidad, el precio del transporte requerido será de **46'08€.** El precio Llodio-San Mamés es de 3'84€ ida y vuelta.

COSTE TOTAL DEL PROYECTO: 11.612'5€ (coste personal) + 30'83€ (ordenador) + 58.35€ (luz) + 240€ (internet + móvil) + 46'08€ (transporte) = **11.987'76€**

2.7.- Prevención de riesgos

En esta parte de la documentación se aclaran los diferentes riesgos que entrañan todo lo planteado anteriormente. Cada riesgo tendrá una descripción donde se explicará en qué consiste, el plan de prevención que se utilizará para que el riesgo no suceda, un plan de contingencia para que en caso de que suceda el riesgo descrito, se sepa cómo actuar y finalmente la probabilidad y el impacto que supondrán.

- **Problemas de recursos humanos**

Descripción	Padecer una enfermedad, accidente o lesión que no permita seguir adelante con el proyecto planteado, por no poder realizar las actividades necesarias.
Prevención	Llevar una vida sana y no exponerse a riesgos innecesarios en el tiempo que dure el proyecto, así como, salir con poca ropa los días en los que hay bajas temperaturas.
Plan de contingencia	Parar el flujo de actividades y retrasar todo lo que sea posible hasta sanarse por completo. Aun así, el tiempo de espera de las actividades no deberá exceder del tiempo total del proyecto.
Probabilidad	Baja
Impacto	Dependiendo de la enfermedad o lesión sufrida, el impacto será de un nivel u otro. En cualquier caso, siempre será el mínimo gracias al plan de contingencia planteado.

Tabla 37: Problemas de recursos humanos

- **Fallo del ordenador**

Descripción	Por cualquier fallo que haya tenido el ordenador, el proyecto y todos sus complementos se han perdido.
Prevención	Hacer continuamente copias en GitHub, Drive, y Local. En las copias entrarán tanto diagramas (bases de datos, clases, flujos...), como implementación de la base de datos y la programación pertinente a la aplicación.
Plan de contingencia	En caso de rotura del ordenador, la copia del proyecto estará en diferentes sitios online y podrá ser descargado para poder seguir adelante en cualquier otro computador.
Probabilidad	Baja
Impacto	Medio. Tendrá como consecuencia el coste económico de otro ordenador y el coste temporal de instalar todos los programas necesarios en el nuevo dispositivo, por lo que, la previsión para realizar algunas tareas tendrían que ser retrasadas.

Tabla 38: Fallo del ordenador

- **Recursos adicionales**

Descripción	La previsión de recursos en principio no supone ningún coste adicional al que se ha tenido hasta el momento de plantear y comenzar el proyecto. Pero es posible que según se vaya desarrollando se necesiten más recursos adicionales.
Prevención	Tener una buena previsión de recursos a utilizar y ver claramente en alcance que tendrá el proyecto.
Plan de contingencia	En caso de necesitar un recurso adicional tanto por accidente o por no haberlo previsto, se intentará conseguir el de menor coste sin empeorar la calidad/tiempo para seguir adelante con el proyecto.
Probabilidad	Media
Impacto	Medio. Se tendrá en cuenta si el recurso a utilizar es realmente necesario o se puede seguir adelante sin él.

Tabla 39: Recursos adicionales

- **Error en la estimación temporal**

Descripción	Durante el proyecto puede que haya diferentes contratiempos temporales debido a problemas que vayan surgiendo (sobre todo en implementación). Es posible que se haya estimado que cierta tarea vaya a llevar un tiempo determinado y esta se alargue.
Prevención	Ajustarse a cada tarea planteada para realizarla en el tiempo estimado. También tener en cuenta el tiempo de holgura que tendrá cada tarea.
Plan de contingencia	En caso de no poder cumplir con los plazos planteados para la tarea en cuestión, cada tarea tendrá que tener un límite de días en los que se podrá retrasar para que no afecte a la entrega final del proyecto. En el fatal caso que la entrega final se tenga que retrasar, se presentará en la siguiente convocatoria.
Probabilidad	Media
Impacto	Alto, retrasa la entrega del proyecto en el plazo planteado.

Tabla 40: Error en la estimación temporal

3.- Análisis de antecedentes

SanidApp es una aplicación que se ha desarrollado con el objetivo de hacer más amena y fácil la comunicación dentro de una institución sanitaria. Tras hacer un breve estudio sobre la comunicación dentro de este tipo de instituciones, se ha descubierto que la comunicación digital es insuficiente. Por lo que, basándose en diferentes Apps, así como, **Whatsapp**, **Telegram**, **LinkedIn**, **Google Calendar**, etc. se ha creado un programa capaz de comunicar a pacientes y profesionales de la misma institución sanitaria, además de dar la posibilidad de poder llevar una mejor gestión de las tareas que tengan a lo largo del mes.

Es importante destacar que esta aplicación está desarrollada desde cero. Se han utilizado ideas basadas en las aplicaciones mencionadas anteriormente, pero el código de implementación se ha realizado íntegramente por la persona desarrolladora de este proyecto.

3.1.- WhatsApp

WhatsApp (4) es una aplicación de mensajería instantánea que permite comunicarse mediante mensajes escritos y de voz. Para poder utilizar esta aplicación, es necesario que las personas que se vayan a comunicar la tengan instalada en su dispositivo móvil o se conecten a través de la plataforma web. Además de poder enviar mensajes antes mencionados, esto es, escritos y de voz, también se podrán enviar emoticonos, imágenes, documentos, contactos, ubicaciones, etc.

Esta aplicación no es de código abierto, por lo que no se ha podido acceder a la implementación desarrollada, para realizar las diferentes tareas, así como, enviar un mensaje, crear un grupo, etc.

La idea destacada de esta aplicación que ha sido fuente de inspiración para el desarrollo de SanidApp, ha sido la forma de comunicación entre usuario-usuario. Por otra parte, para conseguir esto, se ha implementado un servidor mediante *Sockets* para el envío y recepción de mensajes.

3.2.- Telegram

Telegram (5) es una aplicación de mensajería instantánea parecida a la anteriormente explicada, *WhatsApp*. La diferencia es que esta, es más abierta en cuanto a usuarios. En este caso no es necesario tener registrado previamente el contacto telefónico de la persona con la que se quiere contactar, sino que simplemente se requiere el nombre de usuario. Con ellos se pueden enviar mensajes (hablados y escritos), imágenes, emoticonos, documentos, etc.

La idea principal recogida para este trabajo, es la de buscar a usuarios mediante el nombre. En SanidApp la forma de buscar a usuarios no es con el número de teléfono, sino que se podrán aplicar unos filtros propios de profesionales de la sanidad, así como, nombre, apellidos, especialidad médica y lugar de trabajo. Con esto se conseguirá un mayor abanico de posibilidades en cuanto la búsqueda de usuarios en caso de no saber un dato concreto. *Ejemplo: Se quiere contactar con un médico traumatólogo que está en un hospital en concreto. Gracias a los filtros se podrán ver todos los usuarios traumatólogos que están registrados en ese hospital, pudiendo así, ver su perfil, ver su trayectoria e iniciar una conversación de mensajes instantáneos.*

3.3.- LinkedIn

LinkedIn (6) es una red social dedicada a crear perfiles profesionales. La actividad principal de esta, es compartir perfiles de personas para que instituciones, que también tienen su propio perfil, puedan ver y contactar con ellas. En esta web, se podrá compartir la trayectoria de un profesional con el objetivo de que diferentes profesionales e instituciones tengan acceso a ella.

La idea que se ha cogido de esta red social para el proyecto, es precisamente la de poder compartir o incluir en el perfil de un usuario la trayectoria laboral completada hasta el momento. En SanidApp, es útil para saber con qué profesional contactar, por ejemplo, en caso de necesitar un consejo médico. Basándose en la trayectoria profesional publicada en su perfil, se podrá saber si la persona goza o no de esos conocimientos.

3.4.- Google Calendar

Google Calendar (7) es una de las aplicaciones más usadas de la institución Google. Esta prestación permite visualizar un calendario donde se pueden añadir, editar y compartir eventos. Estos eventos son personalizables pudiendo especificar la ubicación, la duración, los participantes, etc. además de compartir mediante e-mail dicho evento.

Se ha cogido como base Google Calendar, para que cada profesional o paciente pueda llevar una mejor gestión de sus tareas. Se podrán crear eventos vinculables a usuarios que estén dados de alta en la aplicación. *Ejemplo: El profesional responsable de un paciente tiene cita con él para unas pruebas un día en concreto. El profesional sanitario podrá crear el evento y podrá vincularlo a la otra persona. Lo mismo pasa entre dos profesionales.* Los eventos vinculables tan solo podrán crearlos los profesionales, los pacientes, en cambio, podrán crear eventos solamente para ellos mismos.

4.- Captura de requisitos

En este apartado de la documentación, se muestran los requisitos necesarios que hacen falta para el correcto funcionamiento del programa que se desarrollará. Esta es una parte muy significativa del proyecto, ya que una captura de requisitos bien estructurada hará la implementación más clara y rápida.

4.1.- Requisitos funcionales

Los requisitos funcionales recogen las definiciones de las funciones imprescindibles para la utilización de la aplicación:

- Para poder usar la aplicación es necesario que el usuario se registre en la misma.
- El usuario, si es paciente, tendrá que estar dado de alta en la base de datos de pacientes de la institución.
- El usuario, si es profesional, tendrá que estar dado de alta en la base de datos de profesionales de la institución.
- El usuario si es profesional sanitario, tendrá una aplicación con funcionalidades dirigidas a profesionales sanitarios.
- El usuario si es un paciente, tendrá una aplicación con funcionalidades dirigidas a pacientes.

4.2.- Requisitos no funcionales

Los requisitos no funcionales se encargan de definir el comportamiento y el manejo del sistema.

- La aplicación es accesible desde cualquier ordenador.
- Es necesario tener acceso a internet.
- Es fácil e intuitiva de manejar.
- La aplicación es rápida en responder, mientras no haya ninguna sobrecarga tipo software o algún problema hardware que impida la correcta carga de los datos (programa, base de datos y servidor).

4.3.- Casos de uso y jerarquía de actores

Los casos de uso se basan en hacer una demostración del funcionamiento del sistema mediante un diagrama de casos de uso, según el tipo de usuario que esté utilizando la aplicación.

Estos casos, se dividen en dos partes, paciente y profesional sanitario. Los casos de paciente y trabajador, se basan en el funcionamiento local del programa que se comunica con el servidor. Este está siempre en funcionamiento y atiende las peticiones del programa para procesar y enviar datos al cliente (programa local del paciente o profesional).

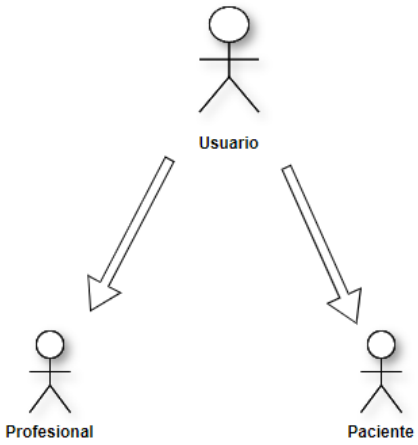

Ilustración 17: Jerarquía de actores

4.3.1.- Usuario genérico

Un usuario genérico es tanto un Usuario Profesional, como un Usuario Paciente. En el siguiente diagrama se visualizan las acciones que pueden llevar a cabo. Como anteriormente se ha mencionado.

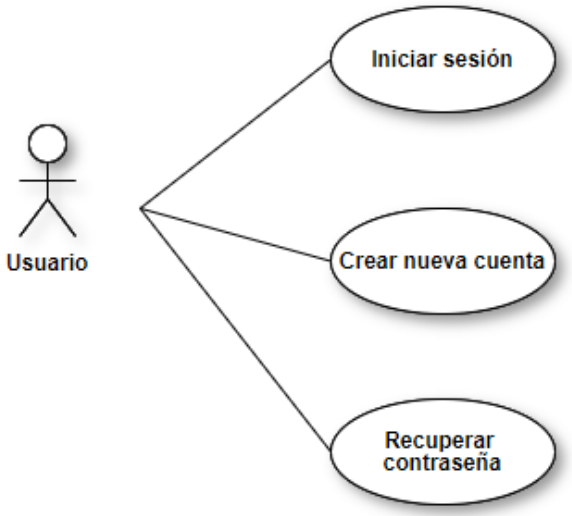

Ilustración 18: Casos de uso Usuario genérico

4.3.1.1.- Iniciar Sesión

Permite al usuario, tanto paciente como profesional, el acceso a la aplicación. Se hace una comprobación de datos para permitir la entrada a su cuenta personal.

Una vez dentro, dependiendo del tipo de usuario que sea, el sistema tendrá unas funciones u otras.

4.3.1.2.- Crear nueva cuenta

Permite al usuario, tanto paciente como profesional, el acceso a la aplicación. Se hace una comprobación de datos para permitir la creación de su cuenta personal.

Una vez cumplimentados y comprobados que los datos son correctos, el usuario se habrá creado una cuenta.

4.3.1.3.- Recuperar contraseña

En caso de que el usuario haya perdido la contraseña, en la pantalla de Login tiene opción de recuperarla. Se presentará una ventana emergente en el que tendrá que cumplimentar unos datos para que el sistema reconozca que es el verdadero usuario y así pueda restablecer una nueva contraseña.

4.3.2.- Usuario Profesional

A continuación, se presenta un diagrama que representa las funciones que puede realizar un Usuario Profesional que se da de alta en la aplicación SanidApp.

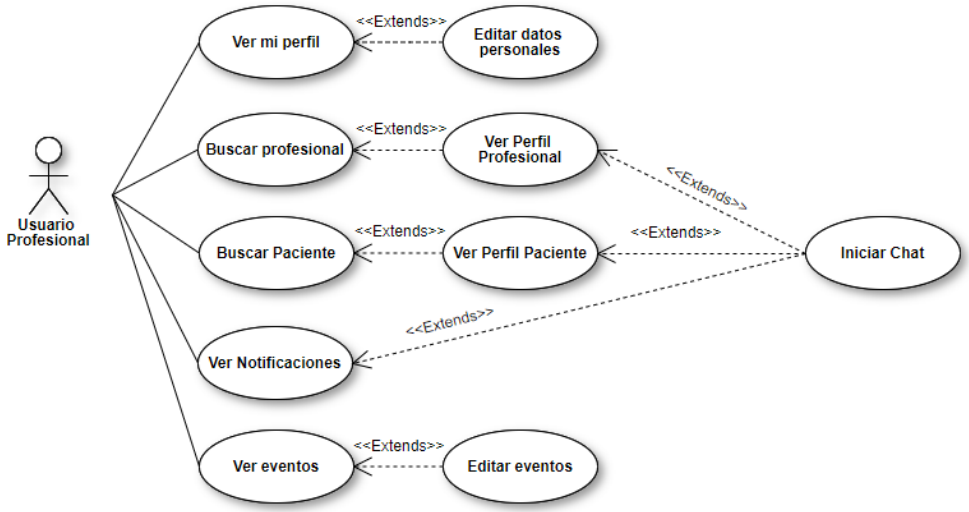

Ilustración 19: Casos de uso Usuario Profesional

4.3.2.1.- Ver Perfil

El usuario Profesional que ha iniciado sesión, puede ver los datos que tiene introducidos en la cuenta. Entre ellos se encuentran los parámetros nombre, apellidos, e-mail, lugar de trabajo actual, especialidad, foto de perfil y los datos de la trayectoria laboral que vaya introduciendo el usuario. Esta información será pública para que otros usuarios, tanto pacientes como profesionales, puedan verla.

Además de poder ver los datos personales, estos también serán editables. La contraseña se podrá modificar en este apartado. Incluso se podrán cumplimentar datos para mostrar la trayectoria profesional del usuario.

4.3.2.2.- Buscar Profesional

En la pantalla de inicio, el profesional tendrá la opción de buscar profesionales que estén dados de alta en la plataforma SanidApp. La búsqueda de usuarios se podrá realizar con unos filtros establecidos, entre ellos, nombre, apellidos, lugar de trabajo actual, o especialidad.

Se tendrá acceso a su perfil para ver sus datos y trayectoria profesional. Incluso se podrá iniciar un Chat de mensajería instantánea.

4.3.2.3.- Buscar Paciente

En este caso, el usuario podrá buscar pacientes que estén dados de alta en el sistema. Los filtros establecidos para buscar pacientes son, nombre y apellidos. El usuario profesional tendrá opción de ver los datos establecidos por el paciente, abrir un Chat y mandar mensajes instantáneos.

4.3.2.4.- Ver Notificaciones

El usuario profesional podrá ver sus notificaciones. Las notificaciones son avisos para mostrar que algún usuario, tanto paciente como profesional, ha escrito un mensaje o más al usuario que ha iniciado sesión. En caso de que le lleguen más de un mensaje de un mismo usuario en el tiempo que ha estado desconectado, la fecha y hora de la notificación corresponderá al último mensaje recibido.

Cuando el usuario pulse sobre una de las notificaciones, tendrá la opción de abrir chat con la persona que le ha escrito con el fin de ver los mensajes recibidos y/o responderle.

4.3.2.5.- Ver Eventos

El usuario tendrá una agenda donde podrá ver, añadir y borrar eventos. Estos eventos se presentarán mediante una lista ordenada donde tras clicar en uno de ellos se podrán ver más detalles, entre ellos su descripción y cuáles son los participantes del evento. Entre otros datos, el nombre, el lugar, la fecha y la hora del evento se mostrarán a primera vista.

Estos eventos pueden haberlos creado él mismo o puede que otros usuarios profesionales los hayan creado y vinculado al usuario actual, ya que tiene o puede participar en la actividad descrita.

Además, el usuario podrá crear nuevos eventos especificando la fecha, la hora, el nombre, el lugar y los participantes del evento. Los participantes podrán ser pacientes y profesionales.

4.3.3.- Usuario Paciente

Este es el diagrama correspondiente a las acciones que puede realizar un paciente en la aplicación SanidApp.

Ilustración 20: Casos de uso Usuario Paciente

4.3.3.1.- Ver mi Perfil

El usuario paciente podrá ver los datos que tiene introducidos en la plataforma. Además, podrá editarlos para que los usuarios profesionales puedan verlos.

4.3.3.2.- Buscar Profesional

El paciente puede buscar profesionales y ver los datos que el trabajador de la institución sanitaria haya introducido en el sistema. Además, el paciente tendrá opción de abrir Chat con el profesional para poder iniciar una conversación con él.

4.3.3.3.- Mis Notificaciones

El paciente tendrá, como en el caso del profesional, opción de ver notificaciones de mensajes que ha recibido en el tiempo que no ha estado conectado al sistema. Estas notificaciones, hacen referencia a mensajes que ha recibido de usuarios profesionales. Como anteriormente se ha explicado, la fecha y la hora de la notificación corresponde al último mensaje recibido de ese usuario profesional.

4.3.3.4.- Mis eventos

Permite al usuario paciente visualizar los eventos que tiene vinculados a su cuenta. Estos eventos pueden ser eventos que ha creado él mismo o eventos que otros profesionales le han vinculado (citas, operaciones, etc.).

En caso de que los eventos los haya creado el paciente, el usuario los podrá borrar sin ningún problema, pero sin embargo, si el evento ha sido creado por un profesional, este evento no se podrá borrar.

4.4.- Modelo de dominio

El modelo de dominio es una representación de las relaciones de clases que componen la aplicación. Estos diagramas no están sujetos a ningún lenguaje de programación, sino que, se representa de forma UML, el cual, se basa en visualizar las relaciones que hay entre las diferentes clases del sistema. Esas clases simbolizan personas, objetos, procesos, etc.

El objetivo de estos diagramas es comprender de forma abstracta cómo se relacionan y qué papel juega cada una de las entidades que forman el sistema.

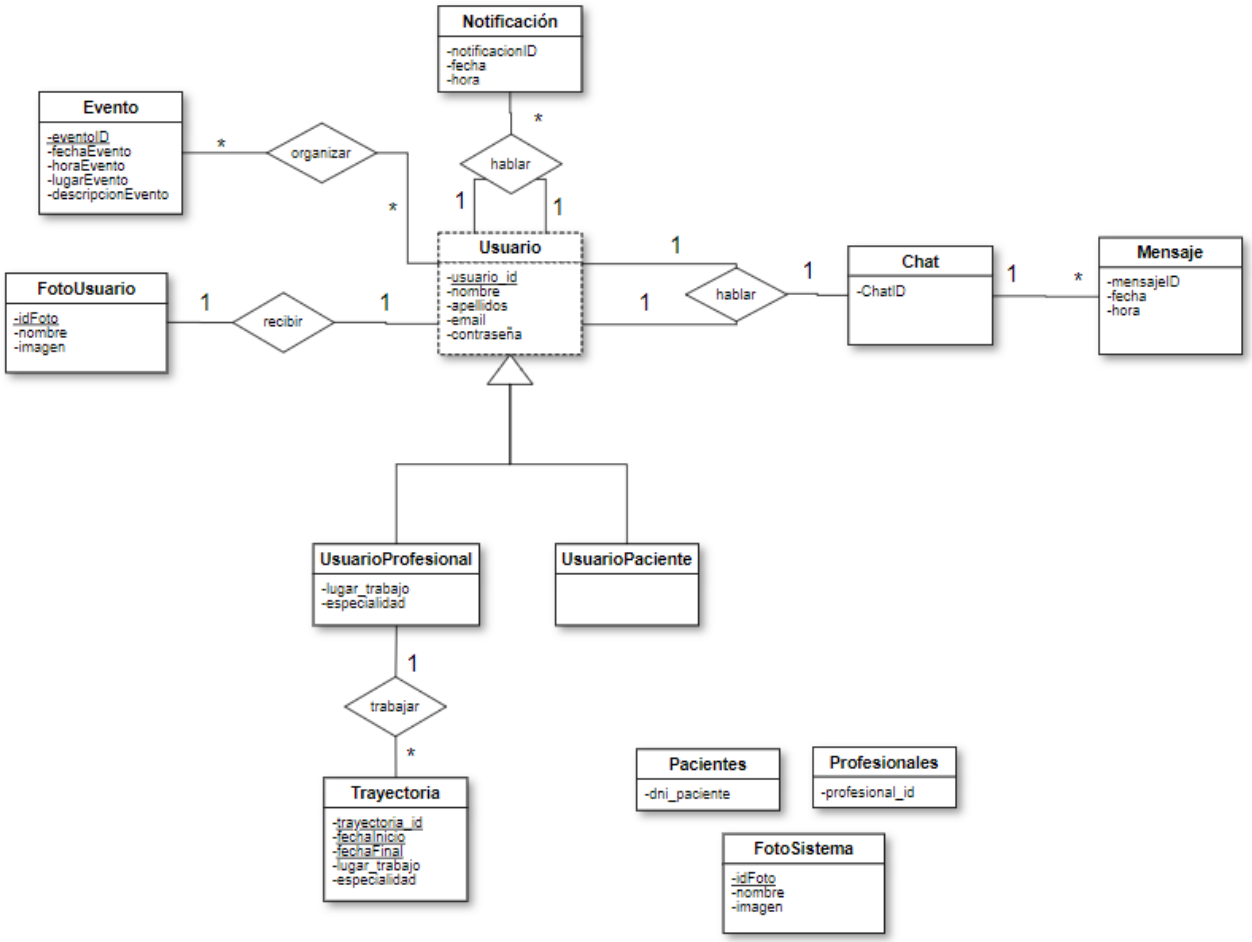

Ilustración 21: Modelo de dominio

4.4.1.- Usuario

En esta entidad se almacenan los datos comunes de un usuario paciente o profesional. Estos datos son los datos mínimos que se le pedirán al usuario al registrarse y que siempre tendrán que estar rellenados. Además, el nombre, apellidos e e-mail son los datos que estarán siempre públicos en caso que el usuario se dé de alta en el sistema.

Por otro lado, todos los datos serán editables, menos el **usuario_id**. En caso de los trabajadores, este dato deberá ser proporcionado por la institución sanitaria contratante de la aplicación. En cambio, en caso de los pacientes, el dato será el número de identificación de identidad.

4.4.2.- UsuarioProfesional

Esta entidad, además de contener los atributos propios de un usuario genérico, guarda también datos propios de un profesional sanitario. Estos datos podrán estar vacíos, en caso de que el profesional así lo prefiera.

4.4.3.- UsuarioPaciente

No guarda ningún dato adicional, pero es necesario mantenerla para que el sistema diferencie de un usuario profesional a un usuario paciente.

4.4.4.- Trayectoria

Las trayectorias son datos que introduce un usuario profesional para registrar los contratos que ha ido teniendo durante su vida laboral en las distintas instituciones. Se guarda el tiempo que ha estado en ese contrato, el puesto que ha ocupado, y el lugar de trabajo. Además cada trayectoria tendrá un código único, que será necesario para operaciones del sistema. Todas las trayectorias estarán vinculadas a un profesional.

4.4.5.- FotoUsuario

Dentro de esta entidad se guardan las imágenes de perfil que tienen todos los usuarios. Cada foto está vinculada a un usuario en concreto, tanto profesional, como paciente.

4.4.6.- Evento

Esta tabla guarda la información completa de un evento. El evento contiene un código identificativo y puede estar vinculado a uno o más usuarios.

Entre los datos que guarda, se encuentran el nombre del evento, el lugar, la fecha, la hora y una breve descripción que introduce el usuario creador del evento. Esta información es necesaria para que el usuario sepa que evento es, donde y cuando va a ser.

4.4.7.- Notificación

Las notificaciones guardan información básica de mensajes que le han llegado al usuario en su periodo de desconexión con el sistema. Cada notificación es identificativa con un código. Además, cada una de ellas guardará la información del usuario que ha mandado el mensaje y la fecha y la hora del último mensaje recibido de ese usuario.

4.4.8.- Chat

Es una entidad que relaciona un usuario con otro para con un identificador único para el chat. Cuando un usuario abra el chat por primera vez con otro usuario, se registrará para que se almacenen los mensajes de una forma más organizada.

4.4.9.- Mensaje

La tabla Mensaje almacena la información de cada mensaje que se manda por un chat. Cada mensaje tiene un código identificativo único, además de contener el código del Chat al que pertenece el mensaje. Esta entidad también guarda el texto del mensaje, la fecha y la hora en la que se envió. Tener en cuenta que cada mensaje está vinculado a dos usuarios, uno que será el emisor y otro que será el receptor.

4.4.10.- Pacientes

Dentro de esta entidad, se almacenan únicamente los números de identidad de los pacientes que son clientes de la institución sanitaria. Para que los pacientes se puedan hacer una cuenta en SanidApp, debe estar registrado como paciente de esta institución. Por lo tanto, a la hora de registrar un usuario Paciente se comprueba que el **usuario_id** introducido se encuentra en esta tabla.

El dato de esta tabla lo debe introducir la institución.

4.4.11.- Profesionales

Igual que en la tabla Pacientes, esta tabla guarda solamente los códigos de los trabajadores. Estos códigos debe introducirlos la institución que ha contratado la aplicación SanidApp. Para que un profesional se pueda registrar en el sistema debe ser trabajador de dicha institución. Por lo que a la hora de que un usuario se vaya a registrar, se comprueba que el **usuario_id** introducido, está en esta tabla.

4.4.12.- FotoSistema

Todas las imágenes necesarias para los gráficos del sistema se encuentran grabadas en esta tabla. Esta tabla es única, los datos guardados aquí no se cambian ni se cambiarán durante la utilización del programa.

5.- Análisis y diseño

En este apartado se hace un estudio del diseño y el análisis de la aplicación. Es una parte fundamental del trabajo previo a la implementación, ya que un buen análisis y diseño tendrán como resultado una programación más precisa. La finalidad de esta sección es estructurar todo lo recogido anteriormente en la captura de requisitos para realizar un diagrama de la base de datos y diagramas de flujo para representar los métodos principales.

Todo esto será necesario para realizar un buen y rápido desarrollo sin incidencias previsibles.

5.1.- Base de datos

La base de datos está formada por tablas y éstas por atributos. A continuación, se ha hecho un diagrama de esas tablas donde se ven los atributos de cada una para representar las relaciones que hay entre cada una de ellas.

Dentro de cada una se han diferenciado qué atributos son principales y cuáles secundarios. Además, se especifica qué relaciones hay entre cada una de las entidades. Esto se puede apreciar en el diagrama de la *Ilustración 22: Base de Datos*.

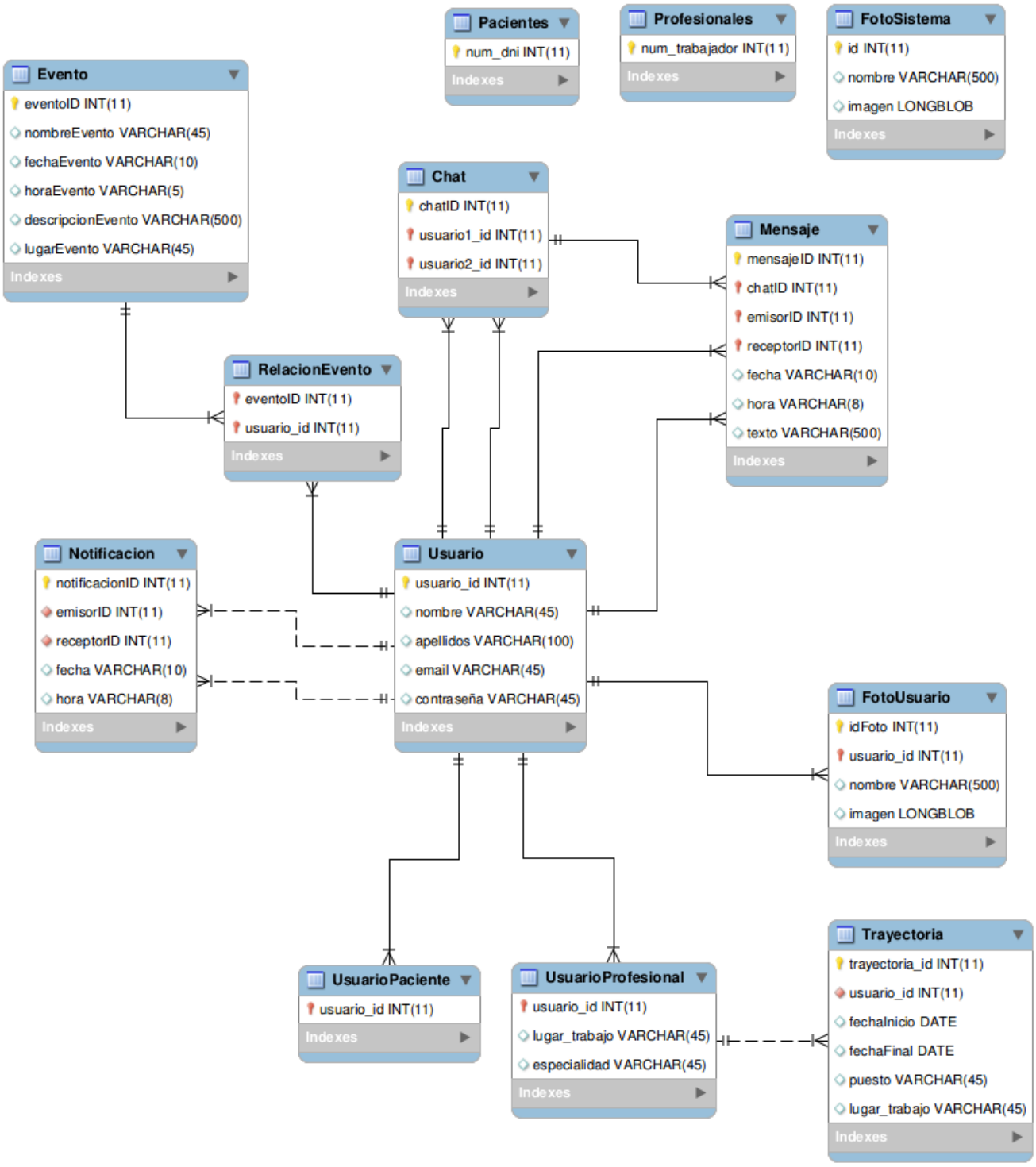

Ilustración 22: Base de Datos

En este diagrama, se puede observar el diseño de base de datos que se ha utilizado para el buen funcionamiento del sistema.

- **Profesionales:** Esta tabla guarda solamente los códigos de los trabajadores que están trabajando para la institución sanitaria contratante de la aplicación. En el momento en el un profesional se registra en el sistema, se comprueba en esta tabla que es trabajador de la entidad.

- **Pacientes**: Esta tabla guarda únicamente los números de los DNIs de los pacientes (clientes) que están registrados en la institución sanitaria. En el momento en el que un usuario se registra, se comprueba en esta tabla si el paciente pertenece a la institución.

- **FotoSistema:** Esta entidad guarda las imágenes que utiliza la aplicación. De esta forma, la primera vez que se accede a SanidApp, el sistema recupera las imágenes, las guarda en el sistema y las carga. Así no tiene que recuperarlas cada vez se quiera acceder a la aplicación.

- **Usuario**: La tabla Usuario guarda los datos que un paciente y un profesional tienen en común. Se ha realizado de esta forma para que, a la hora de la obtención de datos en el sistema, sea más fácil acceder. De esta forma el código de usuario establecido será fijo, en cambio el resto de atributos serán editables dentro del perfil de cada usuario, aunque siempre deberán estar cumplimentados.

- **UsuarioProfesional:** Esta tabla está vinculada a la tabla Usuario. Almacena atributos propios de un profesional sanitario. Hace referencia a la tabla Usuario mediante el código de usuario. Cuando un Usuario Profesional se crea una cuenta en SanidApp, este tiene la opción de cumplimentar estos datos para que el sistema pueda visualizar la especialidad y el lugar actual de trabajo del profesional.

- **Trayectoria**: En esta tabla están los datos que debe tener cada *Trayectoria*. Cada profesional puede añadir *Trayectoria*s a su perfil. Una *Trayectoria* es un dato que representa un intervalo de tiempo para indicar el lugar de trabajo y el puesto que se ha ocupado en el contrato indicado. Esta tabla complementa la información de un profesional y es opcional.

- **UsuarioPaciente**: Esta tabla guarda los datos de los pacientes que se hayan creado una cuenta en SanidApp. El atributo principal, es el número de usuario que a la vez hace referencia a la tabla Usuario.

- **FotoUsuario:** Esta entidad guarda la foto de perfil del usuario. Cada registro tiene un identificador de foto, además de estar relacionado con la tabla usuario, ya que está vinculada con un usuario en concreto.

- **Chat**: Esta tabla guarda únicamente un número identificativo para representar el chat y los números de los usuarios que pertenecen al chat. En el momento en el que un usuario inicia una conversación con otro usuario, en el sistema se crea un registro con los identificadores de los dos usuarios y el identificador del chat (auto-incrementable).

- **Mensaje**: En esta tabla se guarda cada envío que hace un usuario a otro usuario. Como atributos principales se encuentran el identificador del mensaje, el identificador del usuario emisor, el identificador del usuario receptor y el identificador del chat. En cuanto a los atributos secundarios, pero no menos necesarios, se encuentra el texto del mensaje, la fecha y la hora en la que se envió.

- **Notificación**: La notificación consta de información básica para que el usuario tenga conocimiento de que ha recibido un mensaje en su periodo de desconexión con el sistema. Cada notificación tendrá un identificador (auto-incrementable) y estará vinculada a un usuario emisor y un usuario receptor. Al registrar una nueva notificación, se hará una comprobación para saber si ya existe un registro con el mismo emisor y receptor. En este caso, se hará una actualización de la fecha y de la hora de este registro.

- **Evento**: En esta tabla se guarda únicamente la información que contiene un evento. Todos los eventos se identifican con un código único. Además, almacena el nombre del evento, el sitio donde tendrá lugar, la fecha, la hora y una descripción del mismo.

- **RelaciónEvento**: Esta tabla es creada para vincular los eventos con los usuarios. Cada evento está por lo menos vinculado a un usuario. Por lo que cada vez que un evento se vincule a un usuario, se guardará en esta tabla un registro nuevo con el identificador del evento y el identificador del usuario.

6.- Desarrollo

En este apartado, se detalla cómo se ha llevado a cabo el punto anterior. El desarrollo o implementación del proyecto, es la parte más complicada de este trabajo. Mediante esta sección se explicarán las pantallas, su construcción, el flujo que hay entre ellas y el código que se ha implementado para conseguirlas.

La construcción gráfica de las pantallas, se ha realizado de forma simple, ya que los usuarios que utilizarán la aplicación no tienen por qué estar habituados a la tecnología, por lo que se ha intentado implementar para que pueda ser lo más intuitiva posible.

El desarrollo de un proyecto conlleva una gran carga para el mismo. Hacerlo bien depende de haber realizado correctamente todos los puntos anteriores, aunque a la hora de implementar la aplicación se haya tenido que cambiar algunos apartados, la mayoría de ellos sirven de base para que el programa sea lo más preciso posible a la idea principal.

6.1.- Login

En esta sección, explicará todo lo relacionado con la pantalla Login. Es la página inicial de la aplicación. En esta pantalla se podrán realizar diferentes acciones, entre ellas, crear una nueva cuenta SanidApp, recuperar la contraseña, en caso de haberla olvidado, e iniciar sesión.

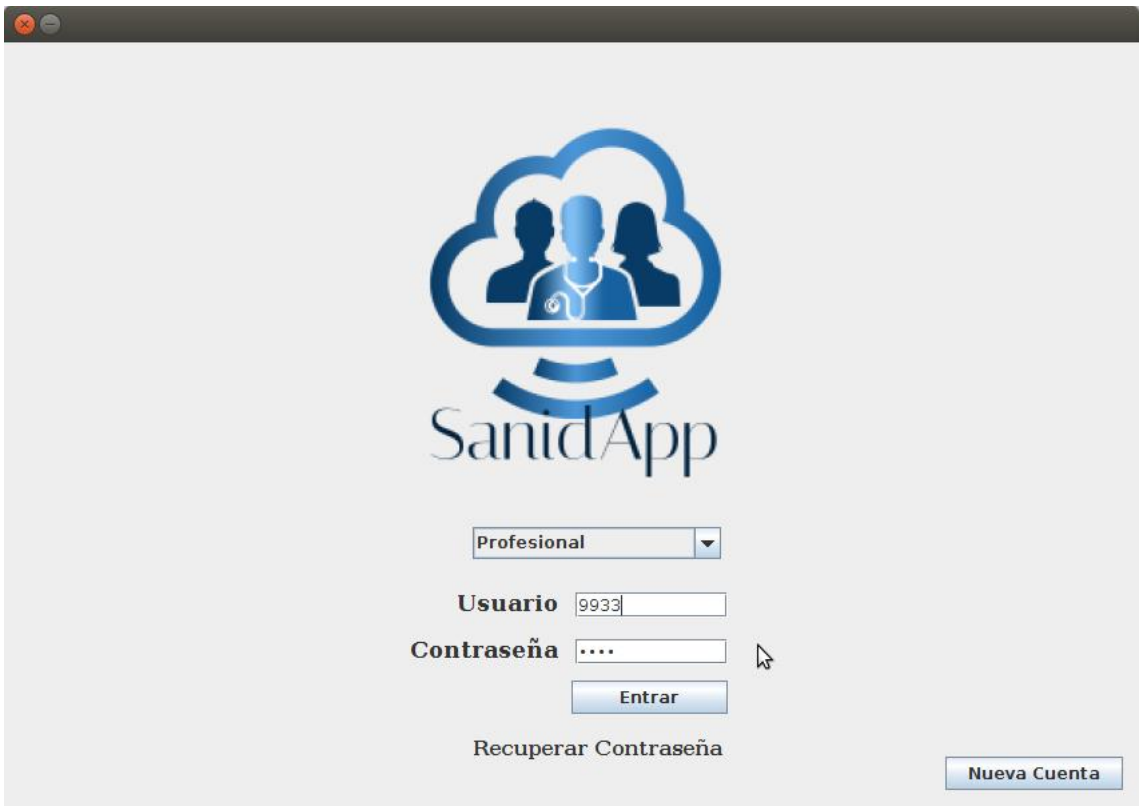

Ilustración 23: Pantalla Login

6.1.1.- Componentes

Esta pantalla, está construida sobre una estructura **JFrame**. En la *Ilustración 23: Pantalla Login* se visualizan claramente las acciones disponibles que tiene esta página. Para llevar a cabo estas acciones se han dispuesto para el usuario unos componentes gráficos del paquete **javax.Swing (3)**.

Logo

En primer plano se encuentra el logo de la aplicación SanidApp. Se ha construido un **JLabel** introduciendo en este, la imagen cargada de la tabla FotoSistema de la base de datos.

```
JLabel lblLogo = new JLabel();
//Conseguimos la imagen del logo
ImageIcon iconoOriginal = FotosDB.getInstancia().getFotoSistema("logo_icon.PNG");
int ancho = 290;
int alto=-1;
ImageIcon iconoEscala = new ImageIcon(iconoOriginal.getImage().getScaledInstance(ancho, alto, java.awt.Image.SCALE_SMOOTH));
lblLogo.setIcon(iconoEscala);
lblLogo.setBounds(283, 60, 311, 270);
contentPane.add(lblLogo);
```

La composición del logo consta de un profesional sanitario, en este caso un médico, y dos personas genéricas detrás de este. Todos ellos se encuentran dentro de una nube, la cual representa internet. El color que se ha utilizado para su relleno es el azul, ya que simboliza el agua, la cual conlleva un significado de pureza, limpieza, tranquilidad y reposo. Por ello, este color es muy utilizado en ámbitos sanitarios.

Ilustración 24: SanidApp logo

Elección de Rol

Existe un componente en esta pantalla de tipo **JComboBox**. Al desplegar este, se han introducido dos elementos: "Profesional" y "Paciente". Se ha de elegir uno de ellos para definir si el usuario quiere acceder a la aplicación como Paciente o como Profesional.

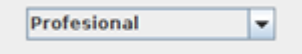

Ilustración 25: Elección de rol Login

```
final JComboBox<String> comboBox = new JComboBox<String>();
comboBox.setBounds(351, 363, 186, 24);
comboBox.addItem("Profesional");
comboBox.addItem("Paciente");
contentPane.add(comboBox);
```

Ilustración 26: Código elección de rol Login

Usuario

Para introducir el número de usuario necesario para acceder a la aplicación, se han definido dos componentes, una etiqueta de tipo **JLabel**, donde se le muestra al usuario dónde tiene que escribir su identificador, y **JTextField** necesario para recoger el parámetro identificador.

Ilustración 27: Usuario Login

```
JLabel lblUsuario = new JLabel("Usuario");
lblUsuario.setFont(new Font("DejaVu Serif", Font.BOLD, 17));
lblUsuario.setBounds(339, 411, 76, 19);
contentPane.add(lblUsuario);
```

Ilustración 28: Código JLabel Usuario Login

```
usuario_text = new JTextField();
usuario_text.setBounds(428, 412, 114, 19);
contentPane.add(usuario_text);
usuario_text.setColumns(10);
```

Ilustración 29: Código JTextField Usuario Login

Contraseña

A la hora de introducir la contraseña, como en el apartado anterior, también son necesarios dos componentes, un **JLabel,** para mostrar al usuario el espacio que tiene para introducir la contraseña, y un **JPasswordField,** el espacio donde se recoge el parámetro "contraseña". Al escribir en este último componente, por cada carácter introducido se visualizará un punto, con el objetivo de que la contraseña no se visualice en pantalla.

Ilustración 30: Contraseña Login

```
JLabel label = new JLabel("Contraseña");
label.setFont(new Font("DejaVu Serif", Font.BOLD, 17));
label.setBounds(304, 442, 111, 27);
contentPane.add(label);

contraseña_text = new JPasswordField();
contraseña_text.setBounds(428, 447, 114, 19);
contentPane.add(contraseña_text);
```

Ilustración 31: Código contraseña Login

Botón Entrar

Este componente es un botón de tipo **JButton**. Al pinchar sobre él, se comprobarán que el número de usuario y la contraseña introducidos son correctos y se dará paso a la ventana de Inicio de ese usuario.

El proceso de comprobación es el siguiente:

1. Si es Profesional la elección de rol que ha escogido el usuario.
   1.1. Si el usuario introducido tiene cuenta en SanidApp como profesional.
      1.1.1. Si la contraseña introducida coincide con la contraseña que le corresponde a este usuario en la base de datos.
         1.1.1.1. Se cargan los datos en el sistema.
      1.1.2. Si no es igual a la contraseña guardada en la base de datos, se visualizará por pantalla un mensaje informativo.

Ilustración 32: Usuario o contraseña incorrectos Login

   1.2. Si el número de usuario introducido no coincide con ningún usuario profesional, se muestra un mensaje de aviso.

Ilustración 33: Usuario no registrado Login

2. Si en cambio el usuario ha escogido la opción de Paciente, el proceso se repetirá contra la base de datos UsuarioPaciente y las comprobaciones serán las mismas.

```
btnEntrar.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        guardarFotoBD();
        if (comboBox.getSelectedItem().equals("Profesional")) {
            UsuarioDBProfesional usuarioProfesionalBD = UsuarioDBProfesional.getInstancia();
            //Comprobamos que el usuario introducido está en SanidApp registrado
            int usuarioPresente = usuarioProfesionalBD.estaElUsuario(usuario_text.getText());

            //comprobamos que el usuario que quiere entrar está previamente registrado
            if (usuarioPresente!=-1) {
                boolean correcto = UsuarioDB.getInstancia().comprobarUsuarioContraseña(usuario_text.getText(), new String (contraseña_text.getPassword()));

                //comprobamos que el usuario ha metido bien la contraseña
                if (correcto) {
                    //cargamos los datos en Java
                    usuarioProfesionalBD.cargarDatos(usuario_text.getText());
                    usuariobd.meterProfionales(generateNumeros());
                    InicioProfesional inicioPantalla = new InicioProfesional();
                    inicioPantalla.setVisible(true);
                    dispose();
                }else {
                    JOptionPane.showMessageDialog(null, "El usuario o contraseña introducidos no son correctos\n"+
                                                "Por favor, revísalos.");
                }
            }else {
                JOptionPane.showMessageDialog(null, "El usuario no se encuentra registrado.\n"+
                        "Por favor, cree una cuenta.");
            }
        }else {
            UsuarioDBPaciente usuarioPacienteBD = UsuarioDBPaciente.getInstancia();
            //Comprobamos que el paciente está en la base de datos
            int usuarioPresente = usuarioPacienteBD.estaElPaciente(usuario_text.getText());
```

Ilustración 34: Código Comprobación Login

Recuperar Contraseña

Se ha dispuesto para el usuario una etiqueta, la cual, pueda clicar, donde podrá recuperar la contraseña en caso de que no la recuerde. Es un componente de tipo **JLabel** que se le ha añadido un **MouseListener** para que el usuario al pasar el cursor por encima de esta etiqueta, agrande el tamaño de su texto y se introduzca dentro de un recuadro.

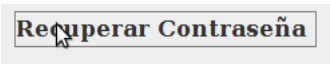

Ilustración 35: Recuperar Contraseña Login

```
final JLabel lblRecuperarContrasea = new JLabel("Recuperar Contraseña");
lblRecuperarContrasea.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
lblRecuperarContrasea.setBounds(351, 515, 226, 27);
lblRecuperarContrasea.addMouseListener(new MouseListener() {

    public void mouseReleased(MouseEvent e) {
        // TODO Auto-generated method stub

    }

    public void mousePressed(MouseEvent e) {
        // TODO Auto-generated method stub

    }

    public void mouseExited(MouseEvent e) {
        lblRecuperarContrasea.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
        lblRecuperarContrasea.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
        lblRecuperarContrasea.setBorder(null);

    }

    public void mouseEntered(MouseEvent e) {
        lblRecuperarContrasea.setCursor(new Cursor(Cursor.HAND_CURSOR));
        lblRecuperarContrasea.setFont(new Font("DejaVu Serif", Font.BOLD, 17));
        lblRecuperarContrasea.setBorder(LineBorder.createGrayLineBorder());

    }

    public void mouseClicked(MouseEvent e) {
        RecuperarContraseña recuperarContraseña = new RecuperarContraseña();
        recuperarContraseña.setVisible(true);
        dispose();

    }
});
contentPane.add(lblRecuperarContrasea);
```

Ilustración 36: Código botón Recuperar Contraseña

Botón crear nueva cuenta

Este componente de tipo **JButton**, es utilizado para que al clicar sobre él, lleve al usuario a una nueva pantalla donde pueda crear una nueva cuenta.

6.2.- Crear cuenta

En esta pantalla cualquier usuario se podrá crear una cuenta SanidApp. Tener en cuenta, que para poder realizar esta acción el usuario profesional tendrá que ser trabajador/a de la institución contratante de la aplicación. En el caso del Paciente, tendrá que ser paciente de la institución.

Esta pantalla se ha querido estructurar de una forma intuitiva, para que el usuario no tenga la impresión de que al usar esta aplicación sea muy difícil llevar las acciones a cabo.

Ilustración 37: Pantalla Nueva Cuenta

6.2.1.- Componentes

A continuación, se explican los componentes que se encuentran en la pantalla Crear Cuenta y las acciones que llevan a cabo cada uno de ellos.

Elije una opción

Igual que en el apartado anterior en la pantalla de **Login**, el usuario debe escoger si se quiere registrar como Paciente o como Profesional. Para ello, se ha dispuesto un **JComboBox** con las opciones "Paciente" y "Profesional".

En caso de que el usuario escoja la opción Paciente, el campo que corresponde a Especialidad queda deshabilitado a la escritura, ya que ese atributo pertenece únicamente a un trabajador sanitario. Además, el componente donde se requiere el número identificativo, cambia de texto.

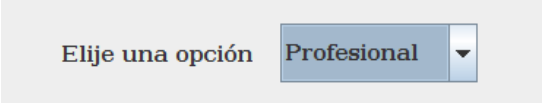

Ilustración 38: Elige una opción Crear Cuenta

```
final JComboBox comboBox = new JComboBox();
comboBox.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
comboBox.setBounds(321, 80, 135, 40);
comboBox.addItem("Profesional");
comboBox.addItem("Paciente");
comboBox.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        String selected = comboBox.getSelectedItem().toString();
        if (selected.equals("Profesional")) {
            numTrabajador_text.setEditable(true);
            especialidad_text.setEditable(true);
        }else {
            numTrabajador_text.setEditable(true);
            especialidad_text.setEditable(false);
            lblNumeroDeTrabajador.setText("Número de DNI");
            lblNumeroDeTrabajador.setEnabled(true);
            lblNumeroDeTrabajador.setToolTipText("Meta sólo los números de su DNI");
            numTrabajador_text.setToolTipText("Meta sólo los números de su DNI");
        }
    }
});
contentPanel.add(comboBox);
```

Ilustración 39: Código Elije una opción Crear Cuenta

Especialidad

Los componentes necesarios para que el usuario Profesional inserte la especialidad son un **JLabel** y un **JTextField**. El **JLabel,** es necesario para indicar el campo donde el usuario tiene que introducir el parámetro de su especialidad. El **JTextField,** es donde tiene que introducir su especialidad. Este campo, como anteriormente se ha explicado, quedará inhabilitado en caso de que sea un Paciente el que se quiere registrar.

Ilustración 40: Especialidad Profesional Crear Cuenta

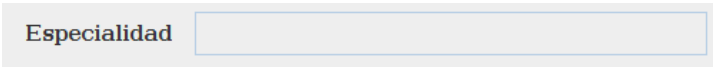

Ilustración 41: Especialidad Paciente Crear Cuenta

Identificador de usuario

En caso de que sea un Profesional el que se quiere registrar, el número identificativo ha de dárselo la institución. Sin embargo, si es un Paciente el que quiere registrarse, el número identificativo será su número de DNI. Estos componentes, como el anterior, se basan en una etiqueta tipo **JLabel** y un **JTextField** para que el usuario introduzca el parámetro.

Ilustración 42: Numero de trabajador Crear Cuenta

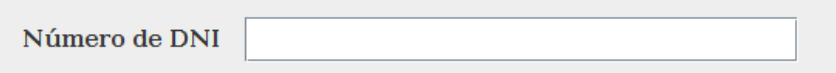

Ilustración 43: Número de DNI Crear Cuenta

Nombre – Apellidos – e-mail

Cada uno de estos grupos también está compuesto por un **JLabel** y por un **JTextField**. El **JTextField** colocado al lado de la etiqueta Nombre, recogerá el parámetro introducido por el usuario que corresponde a su nombre, el de Apellidos, a sus apellidos y finalmente, el de e-mail a su e-mail.

Ilustración 44: Nombre, apellidos, e-mail Crear Cuenta

Contraseña – Repetir Contraseña

Estos dos grupos son del mismo tipo y se requieren exactamente los mismos datos en cada uno de ellos.

Son del tipo **JPasswordField**, esto hace que cuando el usuario escriba en ese campo, no sea visible el texto que está escribiendo, sino que sólo se visualizarán un punto por cada carácter introducido.

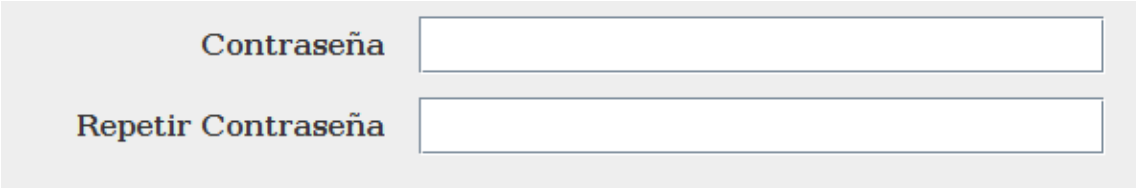

Ilustración 45: Contraseña Crear Cuenta

Cancelar

El usuario podrá cancelar el proceso de inscripción en cualquier momento y volver a la pantalla de **Login**.

Este componente es de tipo **JButton** y tiene aplicado un **ActionListener**, que hace volver a la pantalla de **Login**, si se pulsa sobre él, sin haber guardado ningún tipo de dato.

```
JButton cancelButton = new JButton("Cancel");
cancelButton.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
cancelButton.setActionCommand("Cancel");
cancelButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        new Login().setVisible(true);
        dispose();
    }
});
buttonPane.add(cancelButton);
```

Ilustración 46: Botón Cancelar Crear Cuenta

Crear Cuenta

Este componente es un **JButton**, el cual realiza la comprobación de datos que el usuario ha introducido anteriormente en todos los campos que se visualizan en la pantalla.

La comprobación que se lleva a cabo es en el siguiente orden:

1. Comprueba que los campos Nombre y Apellidos no están vacíos.
   1.1. Si están vacíos el proceso de comprobación para y se visualiza en pantalla un mensaje informativo.

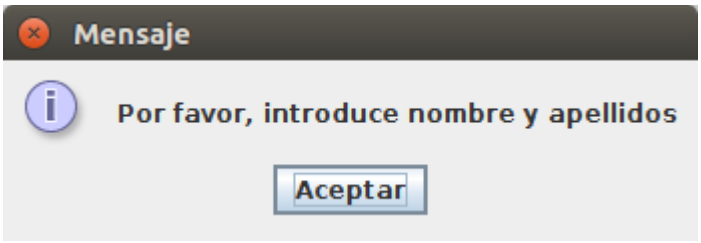

Ilustración 47: Mensaje nombre apellidos Crear Cuenta

2. Se comprueba que el e-mail introducido tiene la forma usual de un correo electrónico (ejemplo@ejemplo.com).
   2.1. Si el e-mail no es válido, el proceso para y se muestra en pantalla un mensaje informativo.

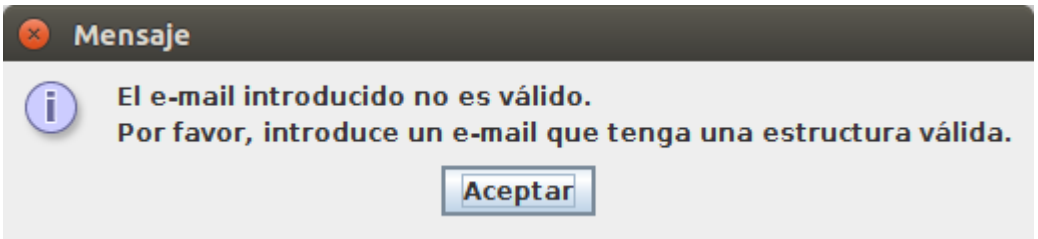

Ilustración 48: Mensaje e-mail incorrecto Crear Cuenta

3. Se comprueba que la contraseña introducida en el campo "Contraseña" contiene al menos un número (medidas de seguridad).
   3.1. Si no se cumple esta condición el proceso para y se muestra un mensaje informativo en pantalla.

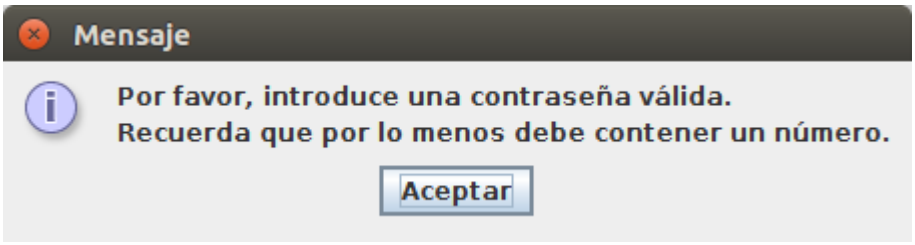

Ilustración 49: Mensaje contraseña no válida Crear Cuenta

4. Se comprueba que los campos "Contraseña" y "Repetir Contraseña" son iguales.
   4.1. En caso contrario, se visualiza un mensaje en pantalla informando del problema.

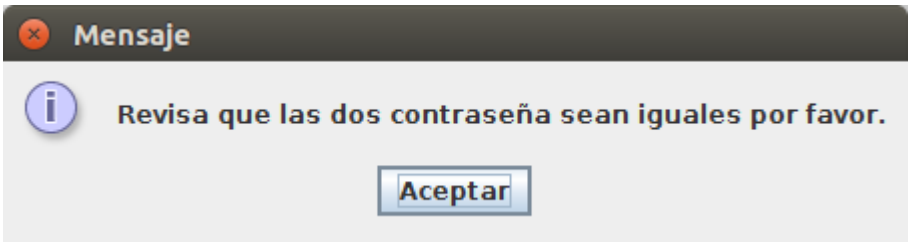

Ilustración 50: Mensaje contraseñas diferentes Crear Cuenta

5. Se comprueba el identificador de usuario introducido es numérico.
   5.1. Si no, se muestra el siguiente error en pantalla.

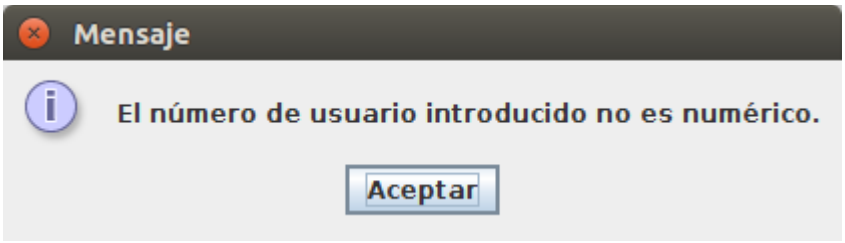

Ilustración 51: Mensaje identificador no numérico Crear Cuenta

6. Se obtiene la opción de rol que ha escogido el usuario para registrarse. Y se comprueba en las tablas de la base de datos respectivamente si actualmente es paciente o trabajador de la institución contratante.
   6.1. Si no es así, se visualiza un mensaje informante en pantalla.

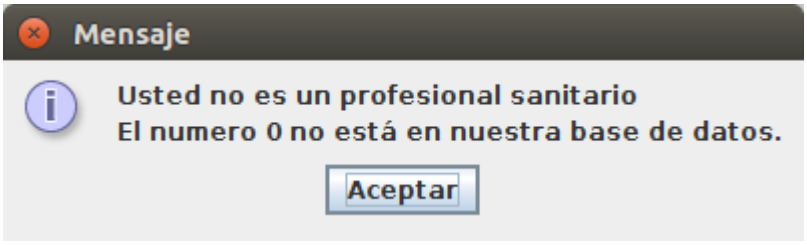

Ilustración 52: Mensaje usuario no profesional Crear Cuenta

7. Se comprueba que el usuario no tiene ya una cuenta creada como el rol especificado.
   7.1. Si es así, se informa por pantalla mediante mensaje.

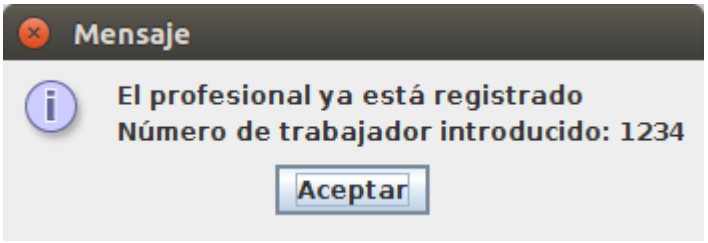

Ilustración 53: Mensaje profesional ya registrado

Finalmente, si todo el proceso de comprobación se ha llevado a cabo sin ningún incidente, se registrará el usuario en la base de datos, creando nuevos registros en la tabla **Usuario** y **UsuarioPaciente** (si es paciente) o en **Usuario** y **UsuarioProfesional** (si es profesional). Para finalizar, se mostrará en pantalla un mensaje de bienvenida.

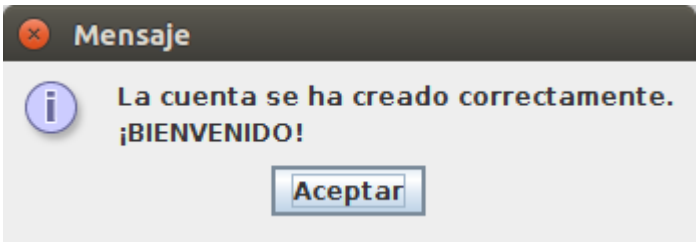

Ilustración 54: Mensaje Bienvenido

6.3.- Recuperar Contraseña

Esta pantalla se visualiza tras pulsar el botón "Recuperar Contraseña" de la ventana Login. Es un pequeño formulario que se le hace al usuario para comprobar sus datos y restablecer su contraseña.

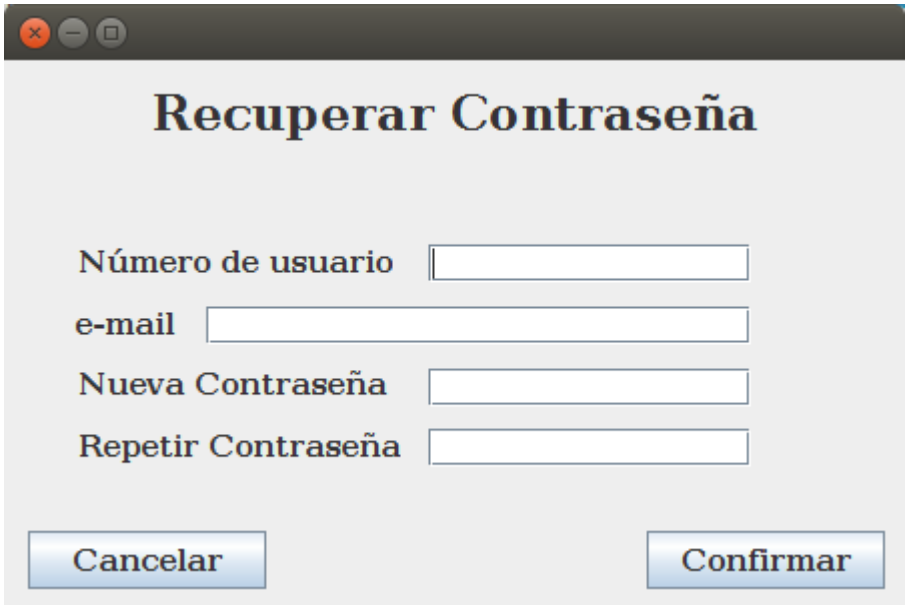

Ilustración 55: Pantalla Recuperar Contraseña

En este caso, los componentes que se han usado ya se han explicado en puntos anteriores, por lo que no se va a proceder a su explicación, sólo se va a describir el proceso de comprobación de datos para el restablecimiento de la contraseña.

El usuario debe introducir su número identificativo, que en caso que lo desconozca, deberá solicitarlo a la institución contratante, el e-mail y la nueva contraseña. Esta última la debe repetir en el último campo y los dos campos deben coincidir.

Si el usuario pulsa el botón **Cancelar**, volverá a la pantalla de **Login** sin haber restablecido la contraseña.

```
JButton buttonCancelar = new JButton("Cancelar");
buttonCancelar.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
buttonCancelar.setBounds(12, 235, 119, 29);
buttonCancelar.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        Login login = new Login();
        login.setVisible(true);
        dispose();

    }
});
contentPane.add(buttonCancelar);
```

Ilustración 56: Código Cancelar Recuperar Contraseña

En cambio, si pulsa el botón Confirmar, se procederá a la comprobación de los datos. En caso de que alguno de los puntos iniciales no se cumpla y salgan los mensajes informativos, el proceso de restablecimiento cesará:

1. Ningún campo debe estar vacío.
   1.1. En caso contrario, se visualizará un mensaje por pantalla indicando el error.

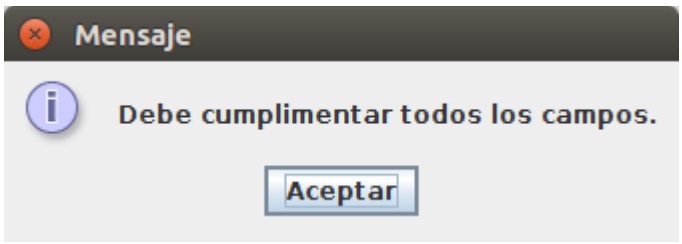

Ilustración 57: Mensaje cumplimentar datos

2. El texto introducido en Número de usuario, debe ser totalmente numérico.
   2.1. Si no es así, se mostrará en pantalla un mensaje informando la incidencia.

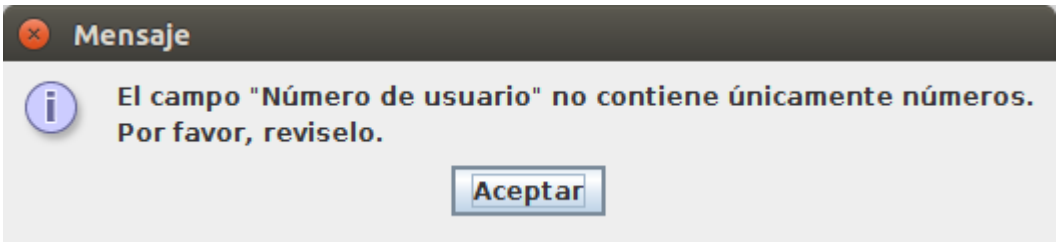

Ilustración 58: Mensaje usuario no numérico

3. Se comprueba que el e-mail y el identificador de usuario se corresponden.
   3.1. Si no se cumple, visualizará el siguiente mensaje.

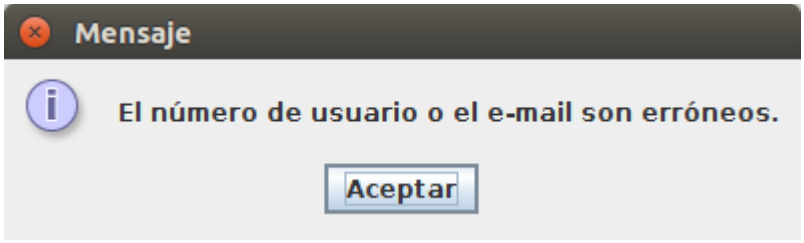

Ilustración 59: Mensaje usuario e-mail erróneos

4. Comprueba que la nueva contraseña contiene al menos un número.
   4.1. Si no es así, se mostrará el siguiente mensaje.

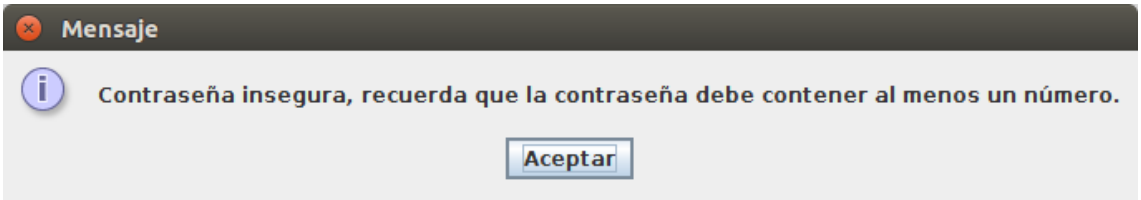

Ilustración 60: Mensaje contraseña insegura

5. El campo “Nueva Contraseña” y “Repetir Contraseña” deben ser exactamente iguales.
   5.1. Si esto no es así, se informará por pantalla.

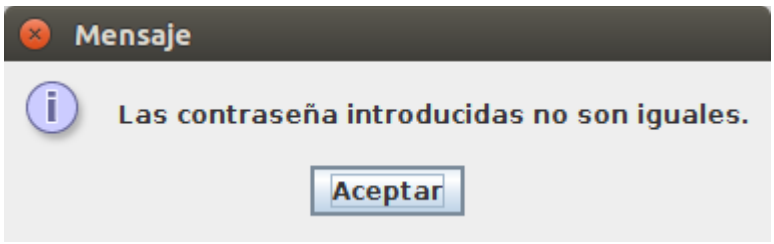

Ilustración 61: Mensaje contraseñas diferentes

Finalmente, si todo ha sido correcto, se hará una actualización de la contraseña del usuario y se informará por pantalla que el proceso de restablecimiento de la contraseña ha sido satisfactorio.

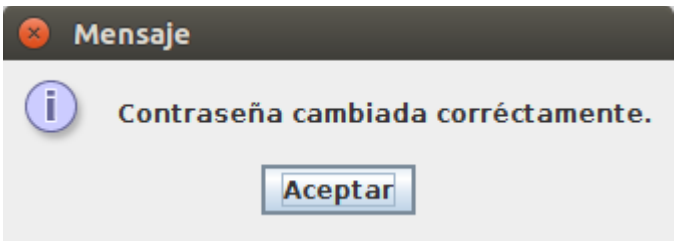

Ilustración 62: Contraseña modificada

6.4.- Inicio Profesional

En el momento en el que un usuario Profesional inicia sesión en SanidApp, entra en la ventana de Inicio de la aplicación. La pantalla de inicio es diferente para un usuario Profesional que para un usuario Paciente. En este caso, se explicará la pantalla de un profesional ya que contiene más acciones para realizar que la de un usuario Paciente.

En el momento en el que se validan los datos de acceso del usuario, se cargan todos los datos (nombre, apellidos, email…) en una clase única del sistema. Este proceso se ha realizado con el fin de no tener que ejecutar sentencias cada vez que se necesitan datos básicos del usuario.

Ilustración 63: Pantalla Inicio Profesional

6.4.1.- Componentes

En este apartado se realiza un análisis de los componentes existentes en la pantalla Inicio de un Profesional. Está compuesta por elementos de tipo **JLabel**, **JButton**, **JPanel** e **ImageIcon**. La tipografía utilizada y el tamaño de letra en todos los elementos **JLabel** y **JButton** es el mismo.

Por otro lado, las imágenes utilizadas en pantalla, salvo la imagen de perfil del usuario y el logo, se han extraído íntegramente de la página web de imágenes libres "pixabay.com" **(8)**.

JPanel Inicio

En este panel está incluida la información relevante del usuario que acaba de iniciar sesión. Mediante dos componentes **JLabel**, se muestra el nombre y la especialidad que tiene el profesional.

Por otro lado, mediante otro **JLabel**, se inserta la foto de perfil del usuario, que es cargada de la base de datos.

Finalmente, en este panel existen dos componentes **JButton** "Mi Perfil" y "Cerrar Sesión" que tienen implementado un **ActionListener**. Lo que hace este componente al ser pulsado es visualizar la pantalla del perfil completo del usuario que ha iniciado sesión o cerrar sesión, respectivamente.

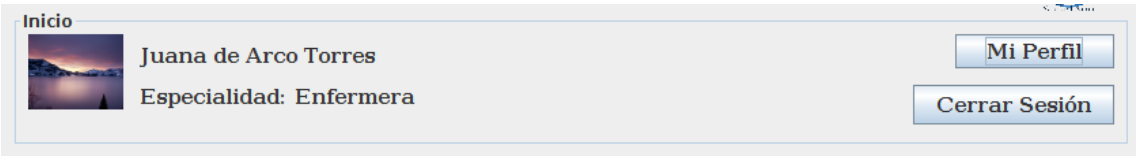

Ilustración 64: Datos del usuario Inicio

JPanel principal

Este es el panel principal de la pantalla, donde se encuentran los botones principales de la aplicación.

Estos botones usan, como anteriormente se ha explicado, elementos **JButton**. Cada uno de ellos tiene una acción vinculada mediante un **ActionListener**, por lo que cada vez que se pulsan llevará a una pantalla específica. Así mismo, las imágenes que utilizan están guardadas en la base de datos, en caso de ser la primera vez que se arranca la aplicación, son extraídas de ahí y guardadas en la máquina local en la que se esté utilizando la aplicación para no tener que estar accediendo continuamente a la base de datos.

Todos estos botones tienen aplicado un **ToolTipText** para que el usuario cuando pose el cursor sobre el botón obtenga una breve descripción de lo que realiza. Esto ayuda a los usuarios inexpertos en la aplicación a saber la acción que desempeña cada botón.

Ilustración 65: Panel botones Inicio

- **Notificaciones**: Este botón tiene como cometido abrir la pantalla de notificaciones para que el usuario pueda visualizar los contactos que le han escrito en su periodo de desconexión.
- **Agenda de Eventos**: Este botón visualiza la pantalla de Eventos que tiene el usuario guardados en su cuenta.
- **Buscar Contactos Profesionales**: Lleva al usuario a la pantalla para buscar un contacto Profesional y poder así, abrir su perfil, iniciar una conversación mediante chat, etc.

- **Buscar Contactos Pacientes**: Lleva al usuario a la ventana correspondiente de buscar Pacientes que estén dados de alta en la aplicación. Como en el caso anterior, para poder ver su perfil e/o iniciar una conversación vía chat.

Finalmente, en esta ventana, arriba a la derecha se visualiza el logo de la aplicación.

6.5.- Notificaciones

Esta pantalla es exactamente igual para un Paciente que para un Profesional. Cuando un usuario pulsa el botón "Notificaciones" en la pantalla de inicio, se abrirá esta ventana. Su cometido es visualizar mensajes de llegada.

En el periodo de desconexión del usuario, cuando otro usuario abra el chat y le mande un mensaje, le llegará una notificación. Esta notificación no visualiza el contenido del mensaje, tan solo muestra la información básica del usuario, la fecha y la hora en la que envió el mensaje. Para que el usuario pueda visualizar el contenido, pulsará sobre la notificación y seguidamente el botón "Abrir Chat".

Ilustración 66: Pantalla Notificaciones

6.5.1.- Componentes

Es una de las pantallas que usa un modelo para visualizar en filas la información de cada notificación. Se basa mayormente en un **JScrollPanel** donde tiene aplicados un modelo que extiende de la clase **AbstractTableModel**.

A continuación, se explican con más detalles estos elementos.

Título

El título de la ventana es un componente **JLabel**. Contiene una parte del texto fijo "Centro de Notificaciones de" y una parte dinámica. La parte dinámica varía según el nombre del usuario que está conectado.

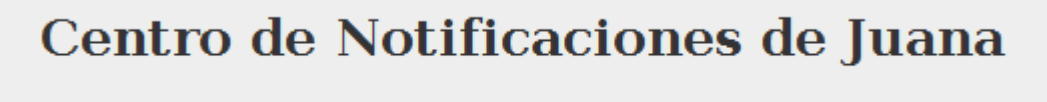

Ilustración 67: Titulo Notificaciones

```
JLabel lblCentroDeNotificaciones = new JLabel("Centro de Notificaciones de " + UsuarioProfesional.getInstancia().getNombre());
lblCentroDeNotificaciones.setHorizontalAlignment(SwingConstants.CENTER);
lblCentroDeNotificaciones.setBounds(12, 10, 826, 30);
lblCentroDeNotificaciones.setFont(new Font("DejaVu Serif", Font.BOLD, 25));
contentPanel.add(lblCentroDeNotificaciones);
```

Ilustración 68: Código título Notificaciones

JTable

Esta parte gráfica necesita de una clase **JTable** para ser aplicada al **JScrollPanel** donde se visualizan las notificaciones. Una clase **JTable** pertenece al paquete **Swing** y su cometido es estructurar información de un modelo para visualizarla en pantalla. Estos componentes se han utilizado en otras pantallas de la aplicación, por lo que el funcionamiento de este, se explicará en este apartado y en los siguientes solo se mencionarán.

```
private final JPanel contentPanel = new JPanel();
private ArrayList<LagNotificacion> notificacionesRecibidas = new ArrayList<LagNotificacion>();
private ModelNotificaciones modelNotificaciones = new ModelNotificaciones(notificacionesRecibidas);
private JTable jTable = new JTable(modelNotificaciones);
```

Ilustración 69: Código Forma panel Notificaciones

El modelo utilizado para la muestra de notificaciones, es una clase llamada **ModelNotificaciones**. Este modelo extiende de la clase **AbstractTableModel**, que se ha implementado para mostrar únicamente la información explicada anteriormente.

```
jTable.setRowHeight(64);
JScrollPane scrollPane = new JScrollPane(jTable);
scrollPane.setBounds(12, 78, 826, 447);
contentPanel.add(scrollPane);
```

Ilustración 70: Código definición JTable Notificaciones

Existen dos atributos principales para este tipo de modelos; el primero, un vector con los datos de cada fila, en este caso un **Vector de Notificaciones** y el segundo, un **Vector de String** donde se insertan los nombres de cada columna. En cada nodo del vector de notificaciones se guarda

una notificación. Las notificaciones son recuperadas de la base de datos y cargadas en clases tipo "**LagNotificacion**".

```
public class ModelNotificaciones extends AbstractTableModel{
    private Vector<LagNotificacion> data = new Vector<LagNotificacion>();
    private Vector<String> columnNames = new Vector<String>();

    public ModelNotificaciones(ArrayList<LagNotificacion> mensajes) {
        cargarDatos(mensajes);
        cargarColumnas();
    }

    private void cargarDatos(ArrayList<LagNotificacion> mensajes) {
        for(int i=0; i<mensajes.size(); i++) {
            data.add(mensajes.get(i));
        }

    }
```

Ilustración 71: Código modelo Notificaciones

```
private void cargarColumnas() {
    columnNames.add("Foto");
    columnNames.add("Nombre");
    columnNames.add("Apellidos");
    columnNames.add("Fecha del último mensaje");
    columnNames.add("Hora del úlltimo mensaje");
}
```

Ilustración 72: Código cargar columnas Notificaciones

Cuando se hayan cargado las notificaciones en el modelo, éste se cargará en el **JTable** instanciado en la ventana para que el usuario tenga visibles las notificaciones.

```
//SE INICIALIZAN LAS NOTIFICACIONES EN LA TABLA
notificacionesRecibidas = NotificacionesDB.getInstancia().getNotificaciones(UsuarioProfesional.getInstancia().getUsuario_id());
for(int i=0; notificacionesRecibidas.size()>i; i++) {
    modelNotificaciones.añadeElemento(notificacionesRecibidas.get(i));
}
```

Ilustración 73: Código iniciar Notificaciones

Finalmente, cuando el usuario abra el chat, tanto desde la pantalla de notificaciones, como buscando al contacto manualmente, la notificación que corresponde con ese chat se eliminará.

Botones

Los botones que se visualizan son "Volver" y "Abrir Chat". El botón de Volver retrocederá a la pantalla de inicio del usuario y el botón de Abrir Chat, abrirá la conversación correspondiente a la notificación seleccionada.

Ilustración 74: Botón Abrir Chat

```
JButton AbrirChatButton = new JButton("Abrir Chat");
AbrirChatButton.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
AbrirChatButton.setActionCommand("OK");
ImageIcon abrirChat = FotosDB.getInstancia().getFotoSistema("iniciarChat_icon.png");
ImageIcon abriChatEscala = new ImageIcon(abrirChat.getImage().getScaledInstance(32, -1, Image.SCALE_SMOOTH));
AbrirChatButton.setIcon(abriChatEscala);
AbrirChatButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        if (jTable.getSelectedRow()!=-1) {
            LagNotificacion elem = modelNotificaciones.getRow(jTable.getSelectedRow());
            int usuario_id = elem.getEmisorID();
            //Comprobamos que el usuario es un profesional
            if (UsuarioDBProfesional.getInstancia().estaElUsuario(""+usuario_id+"")!=-1) {

                //Conseguimos sus datos completos
                ContactoProfesional contacto = UsuarioDBProfesional.getInstancia().buscarContacto(usuario_id);
                PantallaChat pantallaChat = new PantallaChat(contacto);

                //Elimimamos la notificacion de nuestra lista
                NotificacionesDB.getInstancia().eliminarNotificacion(elem.getNotificacionID());
                pantallaChat.setVisible(true);
                dispose();

            }else {
                //El chat que queremos abrir es con un paciente

                //Conseguimos los datos que ha mandado el mensaje
                ContactoPaciente contacto = UsuarioDBPaciente.getInstancia().buscarContacto(usuario_id);
                PantallaChatPaciente pantallaChatPaciente = new PantallaChatPaciente(contacto);

                //Eliminamos la notificacion de nuestra lista
                NotificacionesDB.getInstancia().eliminarNotificacion(elem.getNotificacionID());
                pantallaChatPaciente.setVisible(true);
                dispose();
            }
        }else {
            JOptionPane.showMessageDialog(null, "Selecciona el chat que quieres abrir\npara ver las notificaciones.");
        }

    }
```

Ilustración 75: Código Abrir Chat Notificaciones

En caso de que el usuario no haya seleccionado previamente una notificación, se mostrará en pantalla un mensaje informativo.

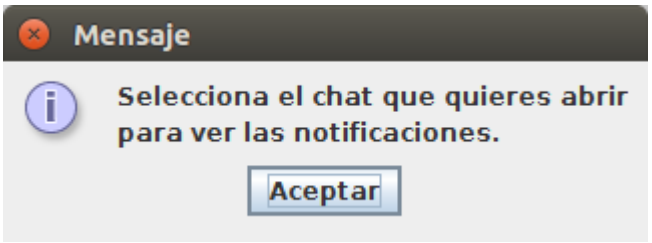

Ilustración 76: Mensaje selección notificación

6.6.- Agenda de Eventos

La agenda de eventos de un usuario es primordial para llevar una buena organización de las actividades diarias de un usuario. Gráficamente esta pantalla es igual en el caso de un Paciente y un Profesional.

Un usuario puede crear un Evento y vincularlo a otros usuarios, tanto profesionales como pacientes. En esta pantalla se visualizarán todos esos eventos.

Se ha dispuesto un panel de filtros para que, en caso de querer buscar un evento en especial, pueda el usuario encontrarlo de una manera más rápida. Además, el panel donde se visualizan los datos de los eventos, sigue una estructura **JTable** como anteriormente se ha explicado.

Ilustración 77: Pantalla ver Eventos

6.6.1.- Componentes

En esta sección se explicarán los componentes que completan la pantalla **Ver Eventos**.

Título

En primera plana se visualiza el título de la ventana "Eventos de <<Nombre>> <<Apellidos>>". El nombre y apellidos se cargarán de la clase única Usuario del sistema. En esta clase, se encuentran los datos personales del usuario y se completa en el momento en el que el usuario inicia sesión en SanidApp.

Ilustración 78: Título Eventos

Panel de filtros

Este panel se ha dispuesto, como anteriormente se ha comentado, para realizar una búsqueda específica de eventos. Está compuesto por un **JCalendar** (elemento extraído de la librería *com.toedter* ***(9)***), dos **JComboBox**, dos **JTextField** y un **JButton**.

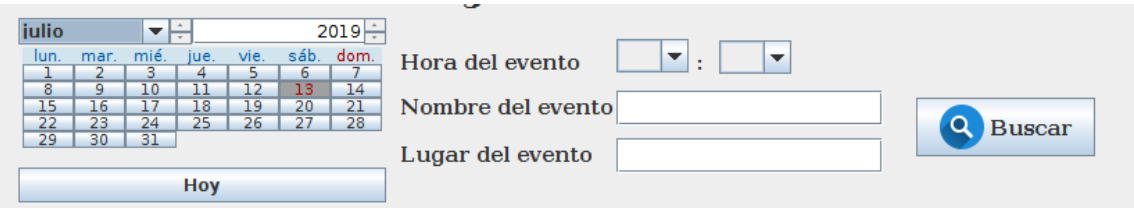

Ilustración 79: Panel filtros Eventos

El **JCalendar** se ha dispuesto para que el usuario pueda seleccionar la fecha exacta del evento. En la parte alta del elemento se visualiza un desplegable para que el usuario pueda elegir el mes y el año del calendario que quiere visualizar. En la parte central, se encuentra el calendario, donde cada día podrá ser clicado por el usuario para especificar el día. En la parte inferior del componente se encuentra un botón con el texto "Hoy", que, si se pulsa sobre él, el calendario marcará el día actual.

Ilustración 80: JCalendar Eventos

```
final JCalendar calendar = new JCalendar();
calendar.setBounds(12, 0, 265, 132);
calendar.setTodayButtonVisible(true);
calendar.setTodayButtonText("Hoy");
calendar.setWeekOfYearVisible(false);

panelFiltros.add(calendar);
```

Ilustración 81: Código JCalendar

Los dos **JComboBox** están dispuestos en este panel con el fin de buscar por hora un evento. Para ahorrar tiempo al desarrollador y minimizar los problemas del usuario al introducir la hora en forma de texto, se han introducido en el primer elemento todas las horas del día en formato "00-23" y en el segundo elemento los números de los minutos de cinco en cinco "00-55". De esta forma, cuando el sistema tenga que recoger la hora introducida, se creará un objeto tipo **String** donde se concatene el dato del **JComboBox** de la hora con el de minutos.

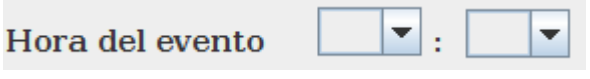

Ilustración 82: Hora Evento

```
final JComboBox comboBoxHora = new JComboBox();
comboBoxHora.setModel(new DefaultComboBoxModel(new String[] {"", "00", "01", "02", "03", "04", "05", "06", "07",
        "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23"}));
comboBoxHora.setFont(new Font("DejaVu Serif", Font.BOLD, 12));
comboBoxHora.setBounds(443, 15, 52, 24);
panelFiltros.add(comboBoxHora);

final JComboBox comboBoxMinuto = new JComboBox();
comboBoxMinuto.setModel(new DefaultComboBoxModel(new String[] {"", "00", "05", "10", "15", "20", "25",
        "30", "35", "40", "45", "50", "55"}));
comboBoxMinuto.setFont(new Font("DejaVu Serif", Font.BOLD, 12));
comboBoxMinuto.setBounds(517, 16, 52, 24);
panelFiltros.add(comboBoxMinuto);
```

Ilustración 83: Código hora Evento

Los dos **JTextField** se usan para que el usuario incluya en su búsqueda el nombre y el lugar del evento que quiere encontrar.

Ilustración 84: Nombre lugar Evento

Finalmente, el botón Buscar, ejecuta la búsqueda de eventos en la base de datos teniendo en cuenta los filtros introducidos por el usuario. Para visualizar correctamente los datos del resultado en el panel, se lleva a cabo un proceso.

1. Se limpia la tabla que contiene el **JScrollPanel** para introducir los nuevos datos.
2. Se guardan en elementos locales los datos introducidos en los filtros.
3. Se comprueba que los dos desplegables que corresponden a la hora no están vacíos.
   3.1. En caso de que uno esté completado y el otro desplegable no, se mantendrá el elemento **String**, donde se guarda la hora, vacío.
4. Se recupera de la base de datos los eventos que estén vinculados al usuario actual y que coincidan con los filtros establecidos.

```
public void actionPerformed(ActionEvent e) {
    tableModel.limpiarTabla();
    Date fecha = calendar.getDate();
    String fecha_string = Utils.convertirFecha(fecha);
    System.out.println("fecha seleccionada: " + fecha_string);
    String hora ="";
    if (!comboBoxHora.getSelectedItem().equals("") && !comboBoxMinuto.getSelectedItem().equals("")) {
        hora= comboBoxHora.getSelectedItem().toString() + ":"+comboBoxMinuto.getSelectedItem().toString();
    }

    System.out.println("hora: " + hora);
    String nombreEvento = textNombreEvento.getText();
    System.out.println("Nombre del evento: " + nombreEvento);
    String lugarEvento = textLugarEvento.getText();
    System.out.println("Lugar del Evento: " + lugarEvento);

    //Comprobamos los campos estan vacios
    ArrayList<Evento> eventos = new ArrayList<Evento>();
        eventos = EventoDB.getInstancia().getEventosConFiltros(UsuarioProfesional.getInstancia().getUsuario_id(),
                                            fecha_string, hora, nombreEvento, lugarEvento);
    System.out.println("eventos.size()="+eventos.size());
    for (int i=0; eventos.size()>i; i++) {
        Evento elem = eventos.get(i);
        tableModel.añadeElemento(elem);
    }
```

Ilustración 85: Código búsqueda Eventos

Panel de resultados

Este panel es un **JScrollPanel** donde se le aplica un **JTable** que a la su vez se personaliza con un modelo específico para visualizar la información de los eventos. Funciona exactamente como el panel explicado en el punto **JTable** de la sección *6.5.1* de la pantalla *Notificaciones*.

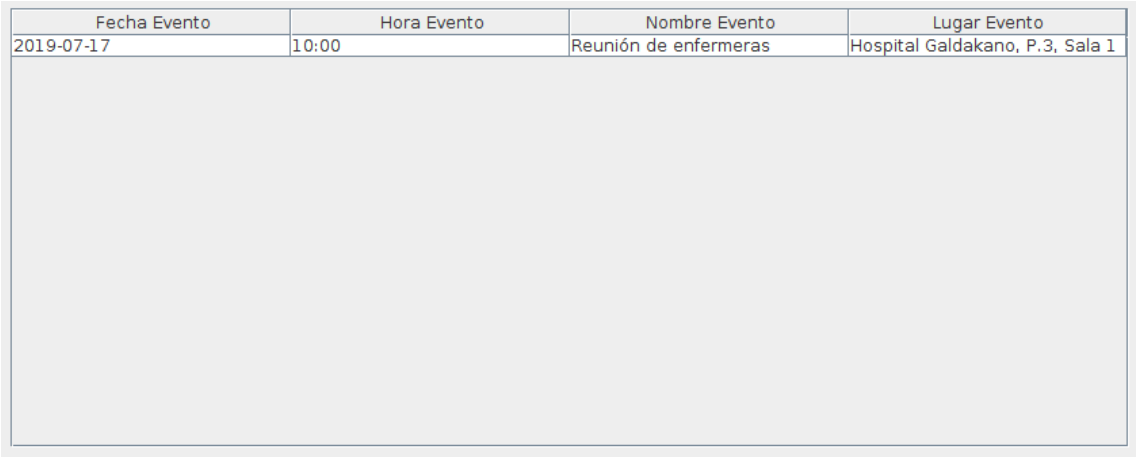

Ilustración 86: Panel resultados Eventos

Los datos de cada evento que se visualizan, son Fecha del Evento, Hora del Evento, Nombre del Evento y Lugar del Evento. Además, tras seleccionar uno de los eventos y pulsar sobre el botón "Ver detalles" se expondrán la descripción que haya incluido el creador del evento y los participantes vinculados a éste.

Panel inferior de botones

En este panel se encuentran los botones "Volver", "Borrar Evento", "Crear Evento" y "Ver detalles". Estos botones son funcionales para el usuario y están todos en la parte inferior de la pantalla para que sea fácil encontrarlos.

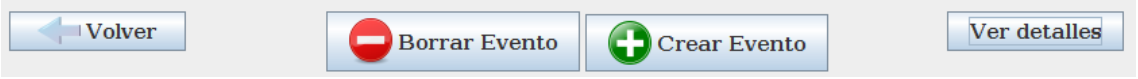

Ilustración 87: Panel botones Eventos

El botón **Volver** cierra la pantalla actual y devuelve al usuario a la pantalla de inicio.

El botón **Borrar Evento**, requiere que el usuario haya seleccionado previamente un evento del panel de resultados para poder eliminarlo de la base de datos. En caso de que sea un Paciente el que haya iniciado sesión en la aplicación, tan solo podrá borrar los eventos que haya creado él. En cambio, si el usuario actual es un profesional, podrá borrar cualquier evento, aunque no haya sido él el creador de este.

Si el usuario no ha seleccionado previamente un evento para eliminarlo, saldrá en pantalla un mensaje informativo.

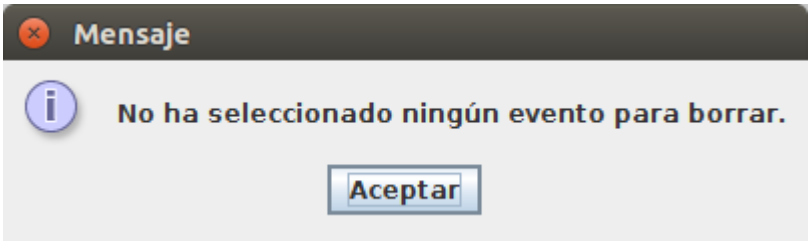

Ilustración 88: Mensaje evento no seleccionado

Si el usuario ha seleccionado un evento y ha pulsado el botón borrar, saldrá un mensaje volviendo a preguntar al usuario si realmente desea borrar ese evento. En caso de que éste responda "Sí", el evento se eliminará de la base de datos.

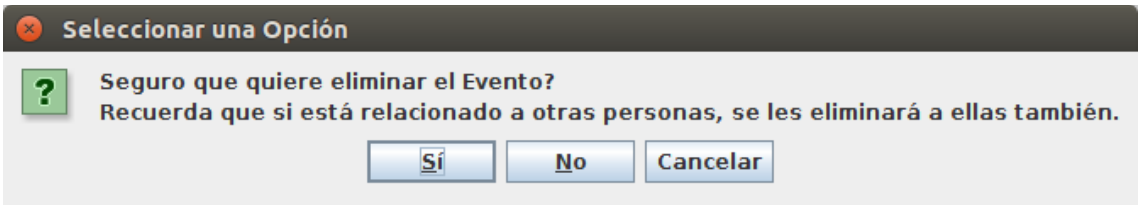

Ilustración 89: Mensaje interrogatorio de eliminación Evento

El botón **Crear Evento** está disponible para que el usuario pulse sobre él y se abra una ventana donde podrá crear y vincular un nuevo evento a otros usuarios.

El botón **Ver detalles** tan solo visualiza una ventana emergente para que el usuario pueda ver la descripción y los nombres y apellidos de los usuarios que están vinculados al evento previamente seleccionado.

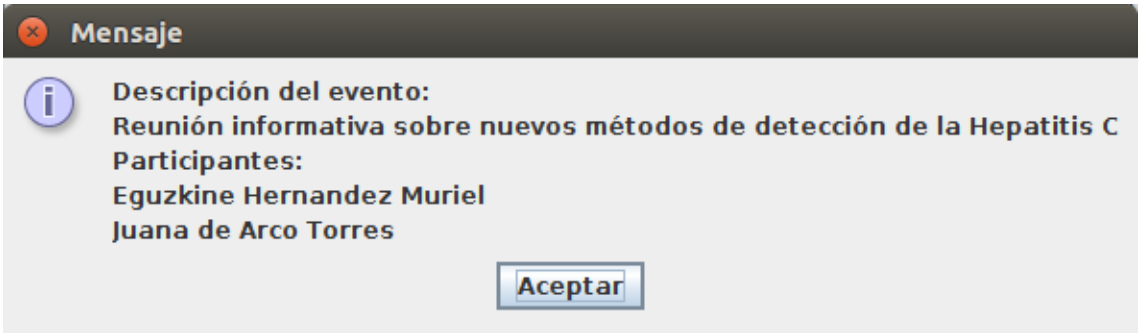

Ilustración 90: Mensaje Descripción Evento

6.7.- Añadir Evento

El usuario podrá crear un evento para añadirlo a su agenda personal y vincularlo directamente a otros usuarios. En el momento en el que vincula a un usuario, en la pantalla explicada anteriormente donde el usuario puede ver sus eventos, se visualizará el nuevo evento creado a los usuarios vinculados.

Al pulsar el botón **Añadir Evento** de la sección anterior, la pantalla que verá el usuario será la siguiente.

Ilustración 91: Pantalla Añadir Evento

En primer lugar, está el panel donde el usuario debe introducir los datos necesarios para crear el evento. Entre esos datos se encuentran el nombre del evento, el lugar del evento, una pequeña descripción de por qué se va a llevar a cabo, el día y la hora en la que tendrá lugar. No se va a proceder a explicar qué componentes se utilizan en este panel, ya que se han explicado en secciones anteriores.

El proceso que se lleva a cabo tras pulsar el botón **Añadir Evento**, es el siguiente:

1. Existe una variable local de tipo **ArrayList** donde se van incluyendo los usuarios vinculados al nuevo evento. Se borran todos los elementos de la lista.
2. Se guardan en variables locales los datos introducidos en pantalla por el usuario.
3. Se comprueban que los datos recogidos de pantalla no están vacíos.
   3.1. En caso contrario, se muestra un mensaje en pantalla informando del problema.
4. Se crea el evento en la base de datos y se vincula al usuario actual.

Una vez se haya realizado este proceso con éxito, el evento se registrará en la base de datos, concretamente en la tabla Evento. También se añadirá un registro nuevo en la tabla **RelacionEvento** con el identificador del mensaje y el identificador del usuario. Seguidamente, se habilitará un segundo panel para la búsqueda de usuarios que se quieran vincular al evento.

Ilustración 92: Panel vincular participantes

```
JButton btnAñadirEvento = new JButton("Añadir evento");
btnAñadirEvento.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
btnAñadirEvento.setBounds(662, 173, 149, 29);
btnAñadirEvento.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        metidos.clear();
        String nombreEvento = textFieldNombreEvento.getText();
        String lugarEvento = textFieldLugarEvento.getText();
        Date fecha = calendar.getDate();
        String fecha_String = utils.Utils.convertirFecha(fecha);
        String descripcion = textPaneDescripcion.getText();
        String hora = "";
        if (!comboBoxHora.getSelectedItem().equals("") && !comboBoxMinuto.getSelectedItem().equals("")) {
            hora = comboBoxHora.getSelectedItem() + ":" + comboBoxMinuto.getSelectedItem();
        }
        if (!hora.equals("")) {
            if (utils.Utils.comprobarCamposVaciosCrearEvento(nombreEvento, lugarEvento, descripcion)) {
                evento_id = EventoDB.getInstancia().crearEvento(UsuarioProfesional.getInstancia().getUsuario_id(),
                        nombreEvento, lugarEvento, fecha_String, hora, descripcion);
                EventoDB.getInstancia().añadirRelacionEvento(UsuarioProfesional.getInstancia().getUsuario_id(), evento_id);
                panel_1.setVisible(true);

            }else {
                JOptionPane.showMessageDialog(null, "Por favor, complete los campos necesarios para la creación del evento.");
            }
        }else {
            JOptionPane.showMessageDialog(null, "Lo siento, la hora introducida no es válida.");
        }

    }
});
panel.add(btnAñadirEvento);
```

Ilustración 93: Código Añadir evento

Los usuarios se podrán buscar por nombre, apellidos y en caso de los profesionales, también por especialidad. Según la opción escogida en el desplegable "Tipo de participante", la búsqueda se hará de Profesionales o Pacientes. Mencionar, que los eventos se pueden vincular tanto a pacientes como a profesionales en la misma transacción.

El proceso que sigue tras pulsar el botón buscar, es el siguiente:

1. Si no se ha rellenado ningún filtro de los disponibles, se buscarán todos los usuarios correspondientes a la elección que tenga en el desplegable **Tipo de participante**.

1.1. Si en cambio se ha rellenado algún campo de los filtros, la búsqueda se ajustará a esos requerimientos.

2. Los resultados acordes a los filtros, se mostrarán en el panel personalizado para ver los datos relevantes de los participantes encontrados.

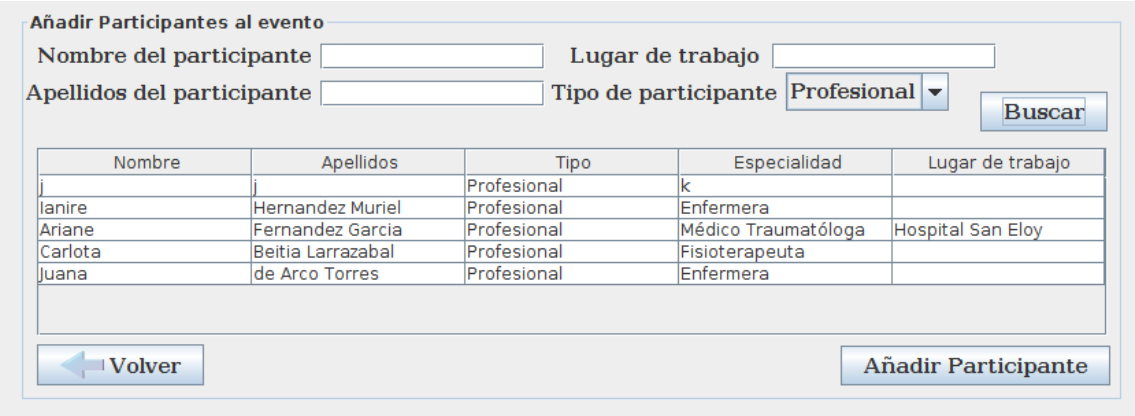

Ilustración 94: Panel buscar participantes

3. El usuario deberá seleccionar un participante del panel de resultados previamente, antes de pulsar el botón **Añadir Participante**. En este momento, el evento creado anteriormente, se vinculará al usuario seleccionado creando un registro en la tabla RelaciónEvento con el identificador del usuario y el identificador del evento. Además, este participante se añadirá a la lista local explicada anteriormente donde se guardan los usuarios vinculados. Esta acción se lleva a cabo para evitar que en los resultados de las búsquedas de participantes se visualicen los ya vinculados.

3.1. En caso de que el usuario pulse este botón sin haber seleccionado previamente un participante, se visualizará un mensaje de error.

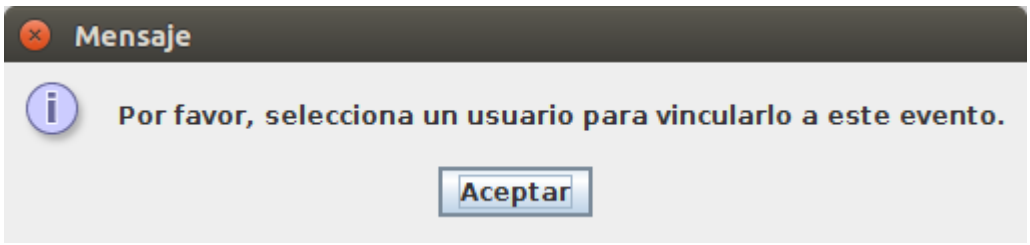

Ilustración 95: Mensaje selección de participante

Finalmente, una vez el usuario haya acabado de crear el evento y vincular participantes, si pulsa el botón **Volver**, se abrirá la pantalla para ver los Eventos. Entonces, se podrá visualizar en el panel la información del nuevo evento creado.

```
public void actionPerformed(ActionEvent e) {
    modelParticipantes.limpiarTabla();
    String nombreParticipante = textNombreParticipante.getText();
    String apellidosParticipante = textApellidosParticipante.getText();
    String opcion = (String) comboBox.getSelectedItem();
    String lugarTrabajo = textLugarTrabajo.getText();
    if (opcion.equals("Profesional")) {
        ArrayList<ContactoProfesional> contactosProfesionales = new ArrayList<ContactoProfesional>();
        if (utils.Utils.comprobarCamposVaciosParticipante(nombreParticipante, apellidosParticipante, lugarTrabajo)) {
            contactosProfesionales = UsuarioDBProfesional.getInstancia().buscarTodosLosContactosProfesionales(
                                        UsuarioProfesional.getInstancia().getUsuario_id());
        }else {
            contactosProfesionales = UsuarioDBProfesional.getInstancia().buscarContactosProfesionales(
                    UsuarioProfesional.getInstancia().getUsuario_id(), nombreParticipante, apellidosParticipante, lugarTrabajo, "");
        }
        System.out.println("El tamaño es: "+contactosProfesionales.size());
        if (contactosProfesionales.size()>0) {
            for (int i=0; i<contactosProfesionales.size(); i++) {
                int usuario_id = contactosProfesionales.get(i).getIdContacto();
                String nombre = contactosProfesionales.get(i).getNombre();
                String apellidos = contactosProfesionales.get(i).getApellidos();
                String tipo = "Profesional";
                String especialidad = contactosProfesionales.get(i).getEspecialidad();
                String lugar_trabajo = contactosProfesionales.get(i).getLugar_trabajo();
                Participante elem = new Participante(usuario_id, nombre, apellidos, especialidad,lugar_trabajo, tipo);
                boolean esta=false;
                for (int i2=0; metidos.size()>i; i2++) {
                    if (metidos.get(i2).getUsuario_id()==elem.getUsuario_id()) {
                        esta=true;
                        break;
                    }
                }
                if (!esta) {
                    modelParticipantes.añadeElemento(elem);
                }

            }
        }else {
            JOptionPane.showMessageDialog(null, "No hay resultados con los filtros establecidos");
```

Ilustración 96: Código visualizar participantes

6.8.- Buscar Contacto Profesional

Esta pantalla es necesaria para permitir al usuario buscar contactos profesionales. El objetivo de esta acción es facilitar la comunicación entre un profesional sanitario y un paciente.

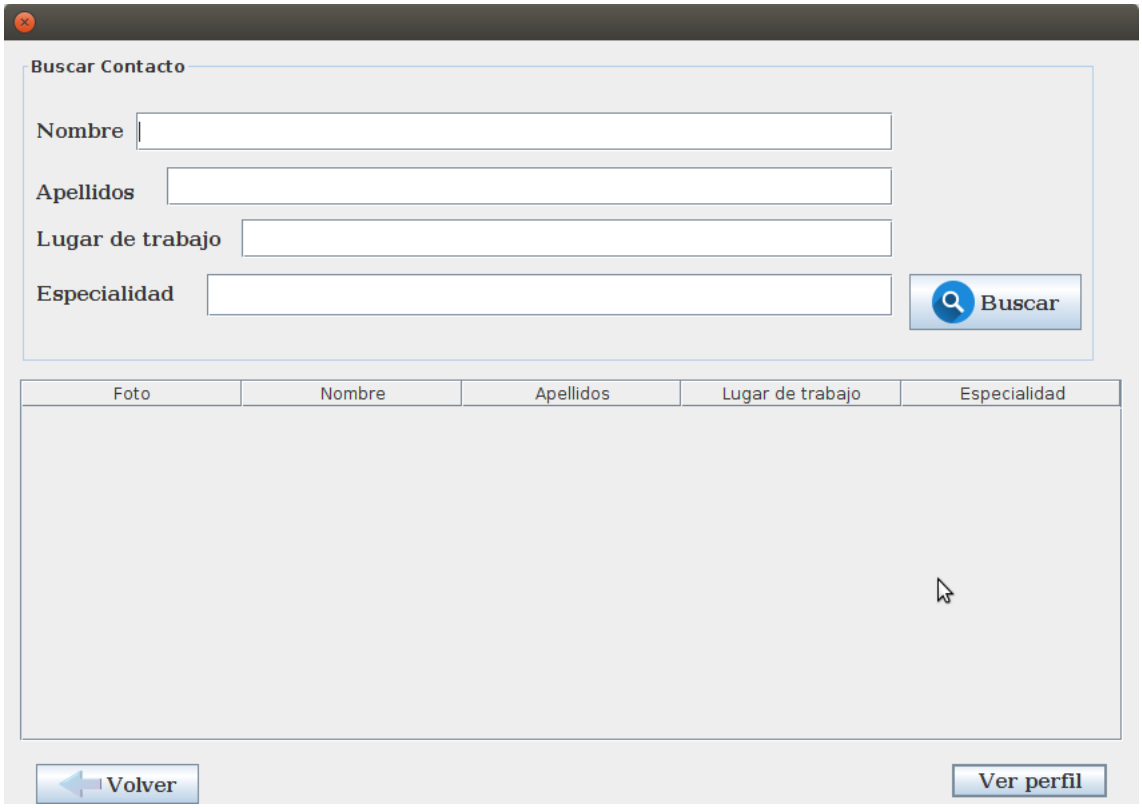

Ilustración 97: Pantalla buscar Contacto Profesional

La distribución de la pantalla consta de dos partes. La primera, en la parte superior de la pantalla, un panel de filtros para buscar contactos específicos. Los filtros disponibles para buscar a profesionales son: Nombre, apellidos, lugar de trabajo y especialidad. La segunda, en la parte inferior, consta de un **JScrollPanel** con la misma función que en los casos anteriores, visualizar ahí los resultados personalizados de los contactos buscados.

El usuario debe pulsar el botón **Buscar**, el cual, sigue el siguiente proceso:

1. Limpia el panel de resultados, por si hubiera alguno de una búsqueda anterior.
2. Comprueba que alguno de los campos que corresponden a los filtros esta completado. En ese caso, realiza una búsqueda que se ajuste a los atributos especificados.

```
String query="SELECT * FROM Usuario U, UsuarioProfesional UP WHERE "
        + "U.usuario_id=UP.usuario_id and U.usuario_id!="+usuarioActual+" and ";
if (!nombre.equals("")) {
    query = query + "U.nombre LIKE \"%" + nombre+ "%\"";
    if (!apellidos.equals("") || !lugarTrabajo.equals("") || !especialidad.equals("")) {
        query = query + " and ";
    }
}
if (!apellidos.equals("")) {
    query = query + "U.apellidos LIKE \"%" + apellidos + "%\" ";
    if (!lugarTrabajo.equals("") || !especialidad.equals("")) {
        query = query + " and ";
    }
}
if (!lugarTrabajo.equals("")) {
    query = query + "UP.lugar_trabajo LIKE \"%"+ lugarTrabajo+"%\"";
    if (!especialidad.equals("")) {
        query = query + " and ";
    }
}
if (!especialidad.equals("")) {
    query = query + "UP.especialidad LIKE \"%" + especialidad + "%\"";
}
query = query + ";";
```

Ilustración 98: SQL buscar contactos con filtros

2.1. En caso de que no se encuentren profesionales que se ajusten a los requerimientos del usuario, se mostrará un mensaje informativo:

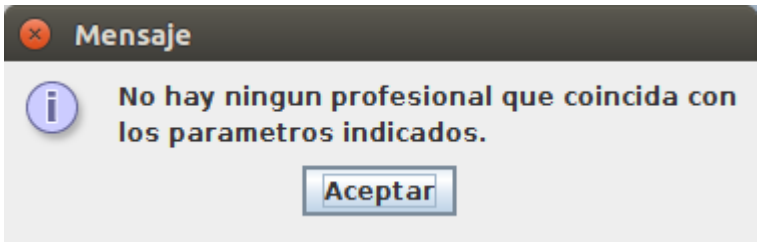

Ilustración 99: Mensaje usuario no encontrado

3. Si por el contrario el usuario no ha completado ningún campo que haga referencia al panel de los filtros, se realizará una búsqueda de todos los usuarios profesionales que se hayan dado de alta en la aplicación.
   3.1. En caso de que no encuentre ninguno, se visualizará un mensaje en pantalla explicando que no hay ningún usuario Profesional dado de alta en SanidApp.

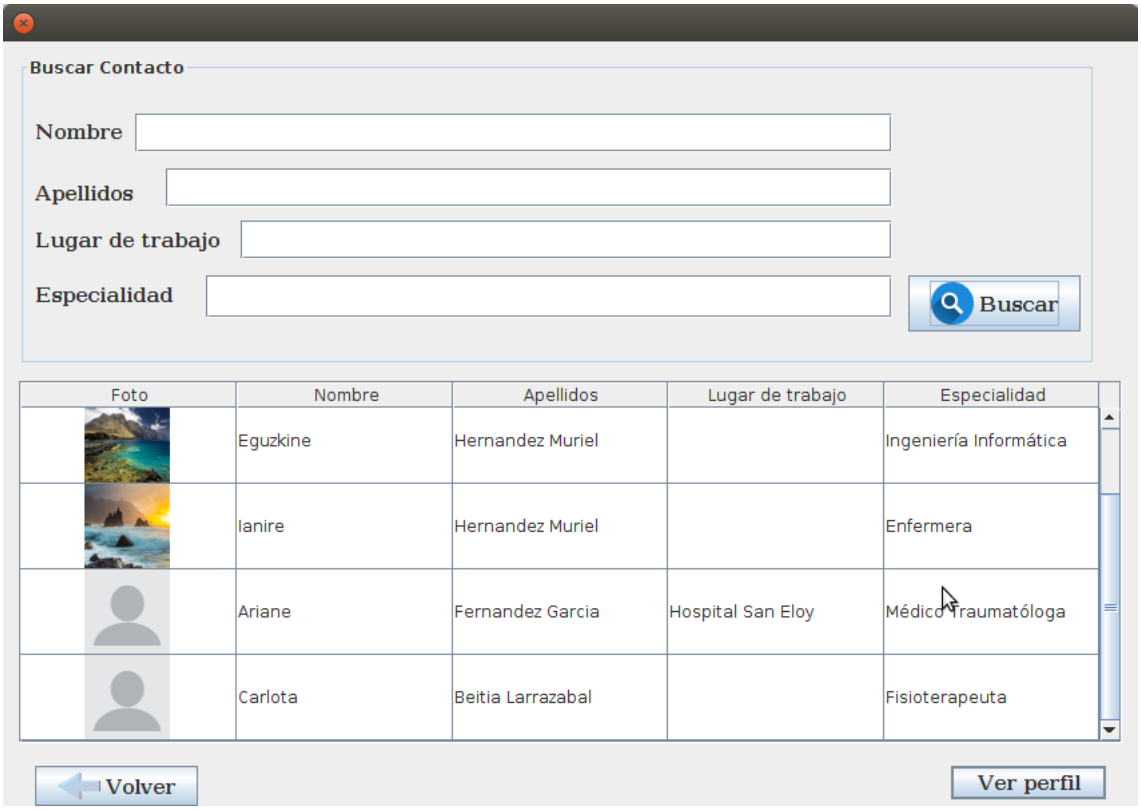

Ilustración 100: Pantalla resultados búsqueda Profesionales

En caso de querer visualizar el perfil de un contacto, el usuario deberá seleccionar previamente un usuario del panel de resultados de la búsqueda y seguidamente pulsar el botón **Ver Perfil**. Esto abrirá una ventana con el perfil del profesional. Si por el contrario, no ha seleccionado previamente un contacto, se informará al usuario del problema por pantalla.

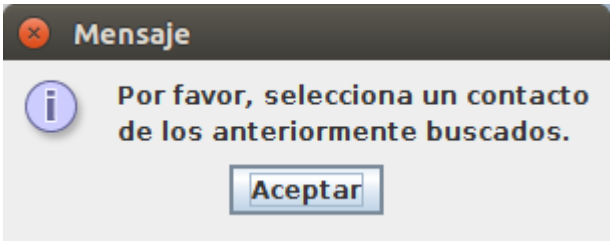

Ilustración 101: Mensaje selección de contacto

6.9.- Perfil de un Profesional

El perfil de un profesional visualizará los datos personales que el usuario haya permitido su visibilidad. En esta ventana se expondrá la imagen de perfil que el profesional se ha adjudicado, su nombre, sus apellidos, su lugar de trabajo actual (si lo ha indicado), y su especialidad. Además, el usuario tiene la opción de ver la trayectoria que el profesional ha incluido en su perfil. Finalmente, la última acción que puede realizar el usuario en esta ventana, es iniciar una conversación vía Chat con el contacto buscado.

Ilustración 102: Pantalla Perfil de Contacto

En cuanto a componentes, esta ventana se ha construido íntegramente con elementos **JLabel**, exceptuando el botón **Volver**. La imagen del profesional que se visualiza, está insertada en un **JLabel**, la cual se adapta a las medidas de éste.

También es de destacar, que las etiquetas **Ver trayectoria profesional** e **Iniciar Chat** son dinámicas. Es decir, si el usuario pasa por encima el cursor del ratón, los textos de estas etiquetas se harán más grandes, se pondrán en estilo **BOLD**, además la forma de cursor cambiará de ser una flecha a una mano, lo que hará entender al usuario que ese texto se puede clicar.

```
ImageIcon iconoEscala = new ImageIcon(iconoOriginal.getImage().getScaledInstance(ancho, alto, java.awt.Image.SCALE_SMOOTH));
verTrayectoria.setIcon(iconoEscala);
verTrayectoria.setFont(new Font("DejaVu Serif", Font.ITALIC, 17));
verTrayectoria.setBounds(12, 394, 392, 27);
verTrayectoria.addMouseListener(new MouseListener() {

    public void mouseReleased(MouseEvent e) {
        // TODO Auto-generated method stub

    }

    public void mousePressed(MouseEvent e) {
        // TODO Auto-generated method stub

    }

    public void mouseExited(MouseEvent e) {
        verTrayectoria.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
        verTrayectoria.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
        verTrayectoria.setBorder(null);
    }

    public void mouseEntered(MouseEvent e) {
        verTrayectoria.setFont(new Font("DejaVu Serif", Font.BOLD, 17));
        verTrayectoria.setCursor(new Cursor(Cursor.HAND_CURSOR));
        verTrayectoria.setBorder(LineBorder.createGrayLineBorder());
    }


    public void mouseClicked(MouseEvent e) {
        PantallaVertrayectoria verTrayectoria = new PantallaVertrayectoria(contacto);
        verTrayectoria.setVisible(true);
        dispose();


    }
});
contentPanel.add(verTrayectoria);
```

Ilustración 103: Código JLabel Ver Trayectoria

6.10.- Ver trayectoria Profesional

Esta parte de la aplicación, gráficamente hablando, contiene únicamente un panel donde se visualizan los datos de trayectoria del contacto buscado.

Las trayectorias, como anteriormente se ha expuesto, son datos sobre contratos que ha tenido o tiene el usuario en cuestión. Estos datos son introducidos por el mismo usuario mediante la ventana de su perfil personal, para que otros usuarios puedan acceder a esta información. Es importante mencionar que esta información ha de ser totalmente verídica.

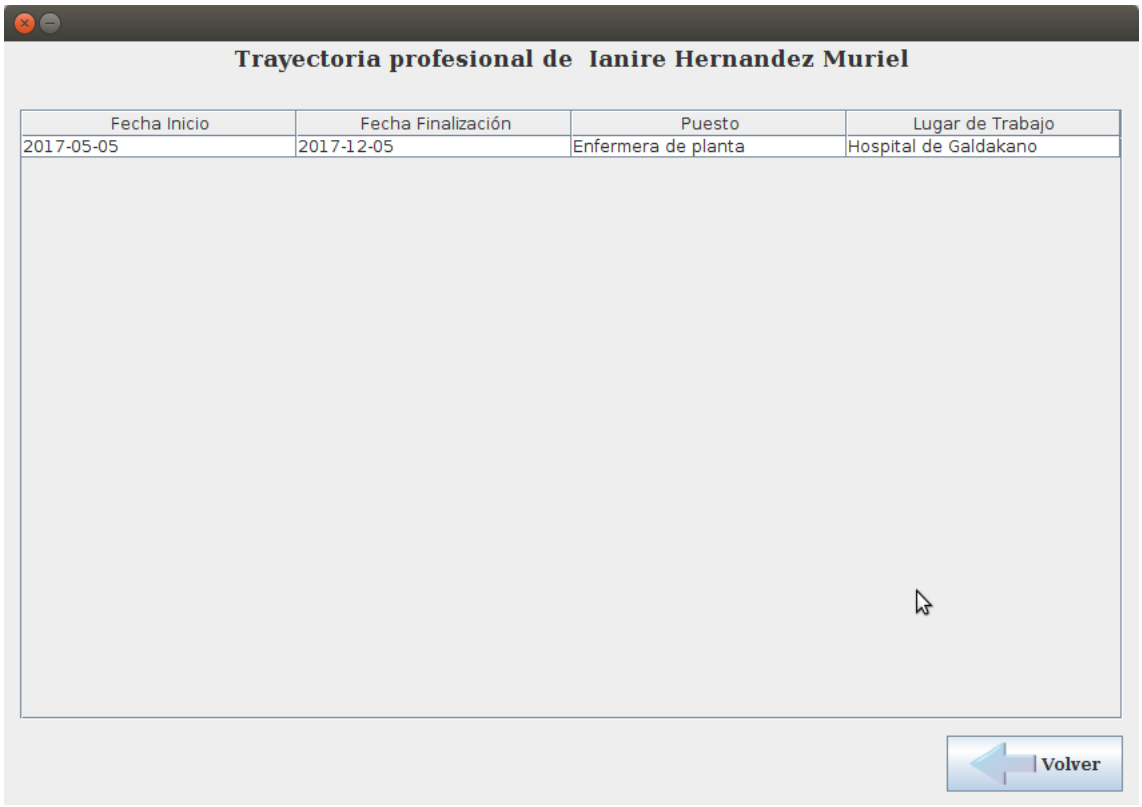

Ilustración 104: Pantalla ver trayectoria de contacto

6.11.- Chat

Esta ventana es una de las más complejas de la aplicación. Aquí, el usuario podrá mandar y recibir mensajes con el contacto seleccionado anteriormente. La parte gráfica que el usuario visualiza en esta ventana, consta de dos paneles principales.

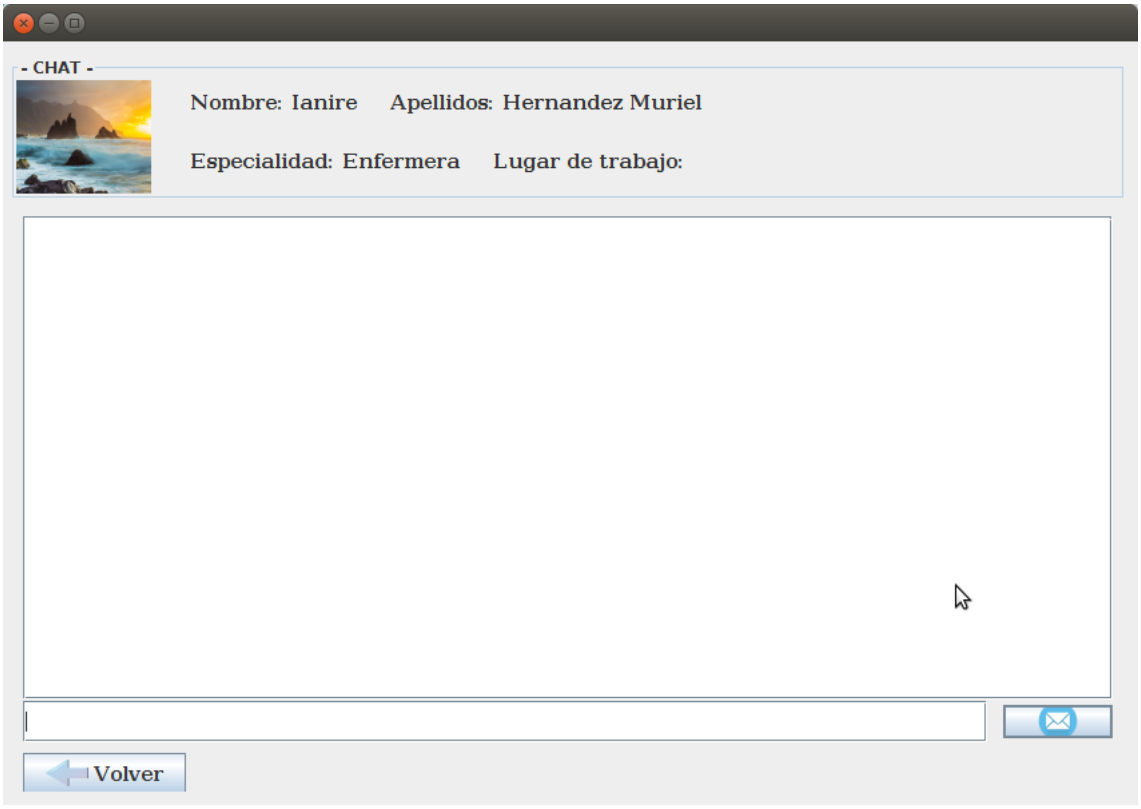

Ilustración 105: Pantalla Chat

En el primero de ellos, se encuentra la información básica del usuario con el que está manteniendo la conversación. A la izquierda del panel, la foto de perfil del usuario, un poco más a la derecha su nombre y sus apellidos. En caso de que sea con un profesional con el que tiene el chat abierto, también aparecerá la información de la especialidad y el lugar de trabajo.

En el segundo panel, se muestra la conversación de los dos usuarios. Los mensajes están ordenados por fecha, es decir, el mensaje más reciente enviado o recibido, aparecerá en la parte inferior del panel. Este es de tipo **JScrollPanel**, por lo que, si los mensajes no cupieran en la pantalla, el usuario podría subir y bajar la barra vertical para poder visualizar los mensajes más viejos.

Este panel, tiene aplicado un estilo de documento para poner en pantalla de forma diferente los mensajes enviados por el usuario que está actualmente en sesión y los mensajes que recibe del contacto. En este caso, los mensajes que se envían, van detallados con el nombre del emisor, la fecha y hora exactas en el que se han enviado.

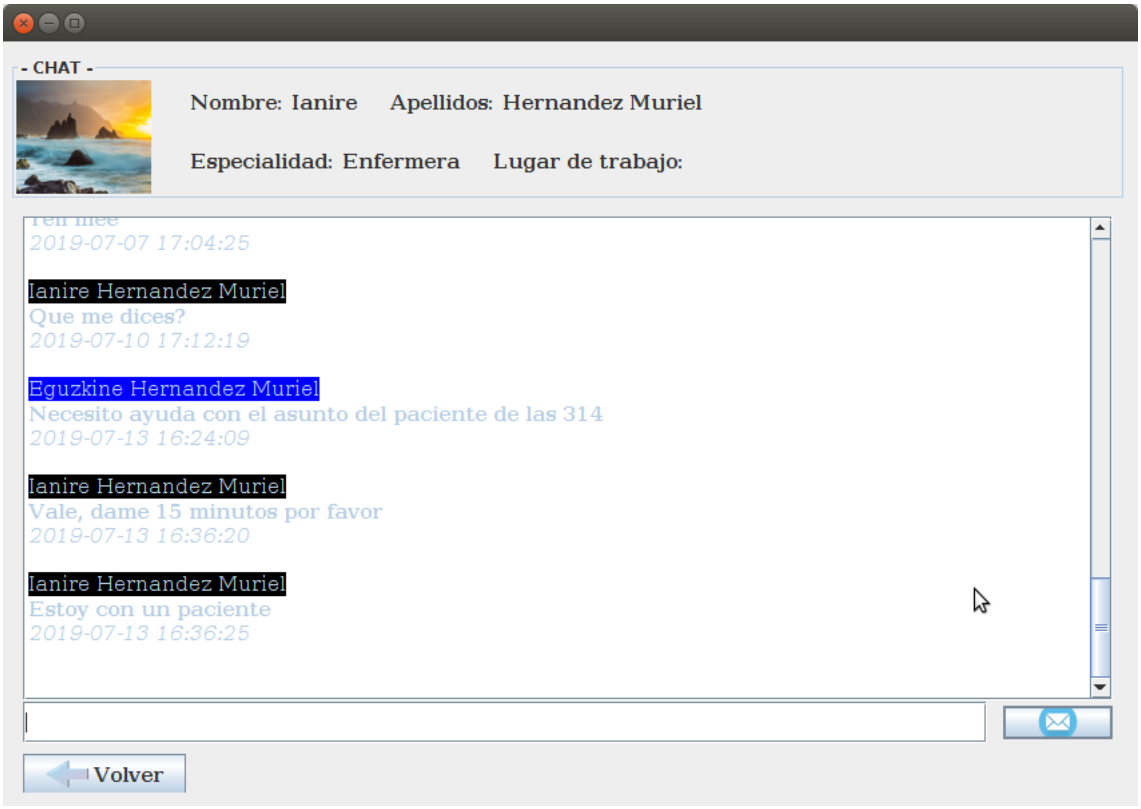

Ilustración 106: Pantalla Mensajes Chat

6.10.1.- Flujo de funcionamiento

En este apartado se explicará cual es el flujo que tiene un mensaje desde que el usuario le da al botón **Enviar** hasta que le llega al contacto en cuestión. Se mencionará también, el uso del servidor, pero la explicación de éste se dará más adelante.

Cuando un usuario inicia sesión en la aplicación manda un paquete al servidor (que está siempre en funcionamiento) para que éste tenga conocimiento de qué usuario se ha conectado. En el momento en el que al servidor le llega el paquete con la IP que está usando el usuario y su número identificador, lo guarda en una lista local que tiene. Esta lista contiene a los usuarios conectados en ese momento.

```
ArrayList<EnvioIP> usuariosConectados ;

public static void main (String[] args) {
    Servidor servidor = new Servidor();
}

public Servidor() {
    usuariosConectados = new ArrayList<EnvioIP>();

    //HILO PARA RECEPCION DE IPS
    Thread hiloIPs = new Thread(new ServidorAñadirIPs());
    hiloIPs.start();

    Thread hiloMensajes = new Thread(new ServidorMensajes());
    hiloMensajes.start();

    Thread hiloDesconexión = new Thread(new ServidorEliminarIPs());
    hiloDesconexión.start();

}
```

Ilustración 107: Código definición Hilos de Servidor

```
private class ServidorAñadirIPs implements Runnable{

    public void run() {
        try {
            while (true) {
                ServerSocket serverSocket = new ServerSocket(9999);
                Socket socketEnvioIPs = serverSocket.accept();

                ObjectInputStream flujo_entrada = new ObjectInputStream(socketEnvioIPs.getInputStream());
                EnvioIP envioIP = (EnvioIP) flujo_entrada.readObject();
                System.out.println("Usuario conectado: " + envioIP.getUsuario_id() + " " + envioIP.getIp());
                usuariosConectados.add(envioIP);
                serverSocket.close();
            }


        }catch (Exception ex) {
            System.out.println(ex.getMessage());
        }

    }

}
```

Ilustración 108: Código AñadirIP Servidor

Envío de mensajes

Si el usuario pulsa el botón enviar en la pantalla del chat, lo primero que hace el programa es visualizar el mensaje con el nombre de usuario y la fecha y hora exactas en pantalla. Seguidamente, se prepara el paquete con la información que tiene que enviar al servidor para que este lo redirija correctamente y lo envíe al destinatario correcto.

Para ello, se abre un **Socket** con la IP que está usando actualmente el usuario y el puerto por el que va a enviar el mensaje. El puerto es siempre el mismo, ya que así se ha dispuesto en la implementación de la aplicación.

```
public void actionPerformed(ActionEvent e) {
    try {
        //SE ESCRIBE EL MENSAJE EN LA PANTALLA
        Document doc = textPane.getStyledDocument();
        String nombreUsuario = UsuarioDB.getInstancia().getNombreUsuario(usuario_id);
        Date fechaAhora = new Date();

        try {
            doc.insertString(doc.getLength(), nombreUsuario+"\n", attrsNombreUsuario);
            doc.insertString(doc.getLength(), campoMensaje.getText()+"\n", attrsTextoUsuario);
            doc.insertString(doc.getLength(), formatoFecha.format(fechaAhora)+" " + formatoHora.format(fechaAhora)+"\n", attrsHora);
            doc.insertString(doc.getLength(), "\n", null);

        } catch (BadLocationException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        //SE PREPARA EL PUENTE POR EL QUE ENVIAR EL MENSAJE
        Socket socketEnviar = new Socket("IP", 9001);
```

Ilustración 109: Código enviar Mensaje

Seguidamente se crea el flujo de salida donde se va a escribir el paquete con la información del mensaje.

```
ObjectOutputStream flujo_salida = new ObjectOutputStream(socketEnviar.getOutputStream());

EnvioMensaje mensaje = new EnvioMensaje();
mensaje.setTexto(campoMensaje.getText());
campoMensaje.setText("");
mensaje.setFecha(formatoFecha.format(fechaAhora));
mensaje.setHora(formatoHora.format(fechaAhora));
mensaje.setEmisor(usuario_id);
mensaje.setReceptor(contacto.getIdContacto());
mensaje.setChat_id(chatID);
```

Ilustración 110: Código flujo salida enviar Mensaje

El paquete tendrá una estructura con los siguientes datos: el identificador del emisor, el identificador del receptor, el texto del mensaje, la IP, la hora, la fecha, el identificador del mensaje y el identificador del Chat. Todos estos datos son necesarios para que el servidor guarde correctamente el envío en la base de datos.

Una vez guardados todos estos datos en el objeto que se va a enviar por el flujo de salida, se escriben en éste, se envía y se cierra el **Socket**.

```
flujo_salida.writeObject(mensaje);
flujo_salida.flush();
flujo_salida.close();

socketEnviar.close();
```

Ilustración 111: Escribir en flujo de salido Mensaje

Llegada a Servidor

Cuando el servidor está arrancado tiene iniciado un hilo por el que tiene ejecutando una clase **Runnable** para así, en caso de que le llegue algún paquete pueda recibirlo. En este caso, el hilo que tiene abierto es el de recibir los mensajes.

```
Thread hiloMensajes = new Thread(new ServidorMensajes());
hiloMensajes.start();
```

Ilustración 112: HiloMensajes Servidor

Este hilo está siempre en funcionamiento para que en cualquier momento pueda coger los mensajes y redirigirlos al receptor correcto. Para ello, se ha implementado el siguiente proceso:

En primer lugar, se abre un **ServerSocket** definido con el mismo puerto por el que se envían los mensajes desde la aplicación. A continuación, se crea un **Socket**, que hace la función de puerta para aceptar todos los paquetes que lleguen por el **ServerSocket**. Se crea el flujo de entrada para poder leer los objetos que entran por el **Socket**.

```
private class ServidorMensajes implements Runnable {

    public void run() {
        while (true) {
            try {
                ServerSocket serverSocket = new ServerSocket(9001);
                Socket socketRecibirMensajes = serverSocket.accept();
                ObjectInputStream flujo_entrada = new ObjectInputStream(socketRecibirMensajes.getInputStream());
```

Ilustración 113: ServidorMensajes recibir Mensaje

El próximo paso es instanciar en una clase el paquete que le ha llegado y recuperar del objeto recibido con los datos que se han escrito en la aplicación local.

```
EnvioMensaje mensajeRecibido = (EnvioMensaje) flujo_entrada.readObject();
int chatID = mensajeRecibido.getChat_id();
int emisorID = mensajeRecibido.getEmisor();
int receptorID = mensajeRecibido.getReceptor();
String fecha = mensajeRecibido.getFecha();
String hora = mensajeRecibido.getHora();
String texto = mensajeRecibido.getTexto();
```

Ilustración 114: Descomposición Mensaje en Servidor

Se guarda el mensaje en la tabla **Mensaje** de la base de datos. Se crea un nuevo registro que especifique el usuario receptor, el usuario emisor, el texto del mensaje, la fecha, la hora y el identificador del chat.

```
MensajesDB.getInstancia().guardarMensajeNuevo(chatID, emisorID, receptorID, fecha, hora, texto);
```

Ilustración 115: Guardar mensaje Servidor

Una vez leído el identificador del receptor, se comprueba que ese usuario esté conectado.

```
boolean encontrado = false;
int i=0;
EnvioIP actual=null;
while (!encontrado && usuariosConectados.size()>i) {
    actual = usuariosConectados.get(i);
    if (receptorID==actual.getUsuario_id()) {
        encontrado = true;
    }
    i++;
}
```

Ilustración 116: Identificar usuario en contectados Servidor

Si el usuario que debe recibir el mensaje no está conectado, es decir, no se encuentra en la lista de conectados del servidor, se guarda la notificación en la base de datos. El proceso es el siguiente:

Comprueba que en la tabla **Notificación** de la base de datos, existe ya un registro con el identificador del emisor y el identificador del receptor. Es decir, que el usuario receptor tiene ya

un mensaje no leído de parte del mismo emisor. Si esto es así, se actualizará ese registro con la fecha y la hora del mensaje actual. En cambio, si no existe ninguna notificación, se creará un nuevo registro en la tabla Notificación con los datos recogidos en el servidor (identificador del emisor, identificador del receptor, la fecha y la hora).

```
public void guardarNotificacion(int emisorID, int receptorID, String fecha, String hora) {
    int notificacionID = HayMasNotificaciones(emisorID, receptorID);
    System.out.println("NOTIFICACIONID: " + notificacionID);
    if (notificacionID!=-1) {

        actualizarNotificaciones(fecha, hora, notificacionID);
    }else {
        String query = "INSERT INTO Notificacion(emisorID, receptorID, fecha, hora) VALUES (?, ?, ?, ?);";
        Connection conn = DBConector.getInstantzia().conn;
        PreparedStatement ps;
        try {
            ps = conn.prepareStatement(query);
            ps.setInt(1, emisorID);
            ps.setInt(2, receptorID);
            ps.setString(3, fecha);
            ps.setString(4, hora);
            ps.execute();
        } catch (SQLException e) {
            System.out.println("ERROR ne guardarNotificacion(): " + e.getMessage());
            e.printStackTrace();
        }

    }

}
```

Ilustración 117: Código guardarNotificación Servidor

```
private  int  HayMasNotificaciones(int emisorID, int receptorID) {
    int resultado =-1;
    String query = "SELECT * FROM Notificacion WHERE receptorID="+receptorID+" and emisorID="+emisorID+";";
    ResultSet rs = DBConector.getInstantzia().execSQL(query);
    try {
        if (rs.next()) {
            resultado = rs.getInt("notificacionID");
        }
    }catch (SQLException ex) {
        System.out.println("ERROR en actualizarNotificacion(): "+ex.getMessage());
        ex.printStackTrace();
    }
    return resultado;

}

private void actualizarNotificaciones(String fecha, String hora, int notificacionID) {
    System.out.println("ESTA ACTUALIZANDO LA NOTIFICACION");
    String query = "UPDATE Notificacion SET fecha=\""+fecha+"\", hora=\""+hora+"\" WHERE notificacionID="+notificacionID+";";
    DBConector.getInstantzia().execSQL(query);
}
```

Ilustración 118: Código comprobación existencia de Notificaciones

Reenvío a Usuario

Si se da el caso de que el receptor del mensaje está conectado a la aplicación, es decir, que está en la lista de conectados que guarda el servidor, redirige el mensaje a ese emisor.

El proceso que sigue, se basa en recuperar la IP que manda el receptor en el momento en el que se conecta a su aplicación. El servidor la guarda y redirige el mensaje a esa dirección. Para ello sigue estos pasos:

Primero, crea un **Socket** con la dirección IP que corresponde al usuario receptor recuperada de la lista de conectados del servidor, y se establece un puerto diferente, en este caso el 9002, para enviar el paquete recibido.

Segundo, se crea el flujo de salida estableciendo la salida del Socket y se escribe en este el objeto del mensaje.

Finalmente se abre el flujo de salida y se cierra el **Socket.**

```
//SE LE ENVIA EL MENSAJE
Socket socketEnviar = new Socket(actual.getIp(), 9002);
ObjectOutputStream flujo_salida = new ObjectOutputStream(socketEnviar.getOutputStream());

flujo_salida.writeObject(mensajeRecibido);
flujo_salida.flush();
flujo_salida.close();
socketEnviar.close();
```

Ilustración 119: Código reenvío Mensaje a receptor

Recepción del mensaje en aplicación

Como el mensaje que envía un usuario a otro se guarda cuando este llega al servidor, el hilo de llegada que ha de estar abierto en las aplicaciones locales y por el que llega el mensaje, solo estará abierto en la pantalla del Chat. Esto se hace así para que el usuario pueda visualizar el mensaje al momento en el que el servidor redirige el mensaje.

El hilo abierto pone en funcionamiento una clase definida e implementada en la pantalla del Chat. Esta clase, implementa un método **Runnable,** como en el caso del servidor y realiza su misma función. En este caso, se abre un **ServerSocket** con el mismo puerto (9002) por el que se ha redirigido el paquete en el servidor. Se acepta todo lo que venga por ese puerto y se lee el objeto de entrada por el flujo de entrada establecido.

```
public class RecibirMensaje implements Runnable{

    public void run() {
        try {
            while (true) {
                ServerSocket serverSocket = new ServerSocket(9002);
                Socket socket = serverSocket.accept();
                ObjectInputStream flujo_entrada = new ObjectInputStream(socket.getInputStream());

                EnvioMensaje mensaje = (EnvioMensaje) flujo_entrada.readObject();
                String texto = mensaje.getTexto();
                System.out.println(texto);
                int emisor_id = mensaje.getEmisor();
                String nombre_emisor = UsuarioDB.getInstancia().getNombreUsuario(emisor_id);
                String fecha = mensaje.getFecha();
                String hora = mensaje.getHora();
```

Ilustración 120: Código RecibirMensaje receptor

Una vez recogido el paquete del mensaje, se descompone para recuperar los datos. Así se extrae el texto del mensaje, el identificador del emisor, la fecha y la hora del mensaje. Con el identificador del emisor, conseguimos su nombre y apellidos ejecutando una sentencia contra la base de datos. Este último dato es necesario para visualizar el nombre del emisor del mensaje en pantalla.

Una vez recuperados todos los datos, se modifica el documento del panel donde se visualizan los mensajes y se inserta el nuevo mensaje con el estilo previamente definido.

```
Document doc = textPane.getStyledDocument();
socket.close();
try {
    doc.insertString(doc.getLength(), nombre_emisor+"\n", attrsNombreContacto);
    doc.insertString(doc.getLength(), texto+"\n", attrsTextoContacto);
    doc.insertString(doc.getLength(), fecha+" " + hora+"\n", attrsHora);
    doc.insertString(doc.getLength(), "\n", null);
} catch (BadLocationException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Ilustración 121: Código añadir mensaje en documento

6.12.- Buscar Contacto Paciente

Este apartado sigue exactamente los pasos y funciones de las secciones *6.8.- Buscar Contacto Profesional* *6.9.- Perfil de un Profesional* y *6.11.- Chat*. La única diferencia que existe es que los resultados de las búsquedas son de usuarios que se han registrado como pacientes.

Por ello, en la pantalla de buscar un paciente sólo habrá opción de buscar por nombre, apellidos. De este modo, los datos que se visualizarán en el panel de resultados de la búsqueda, serán la foto de perfil del usuario, su nombre y sus apellidos.

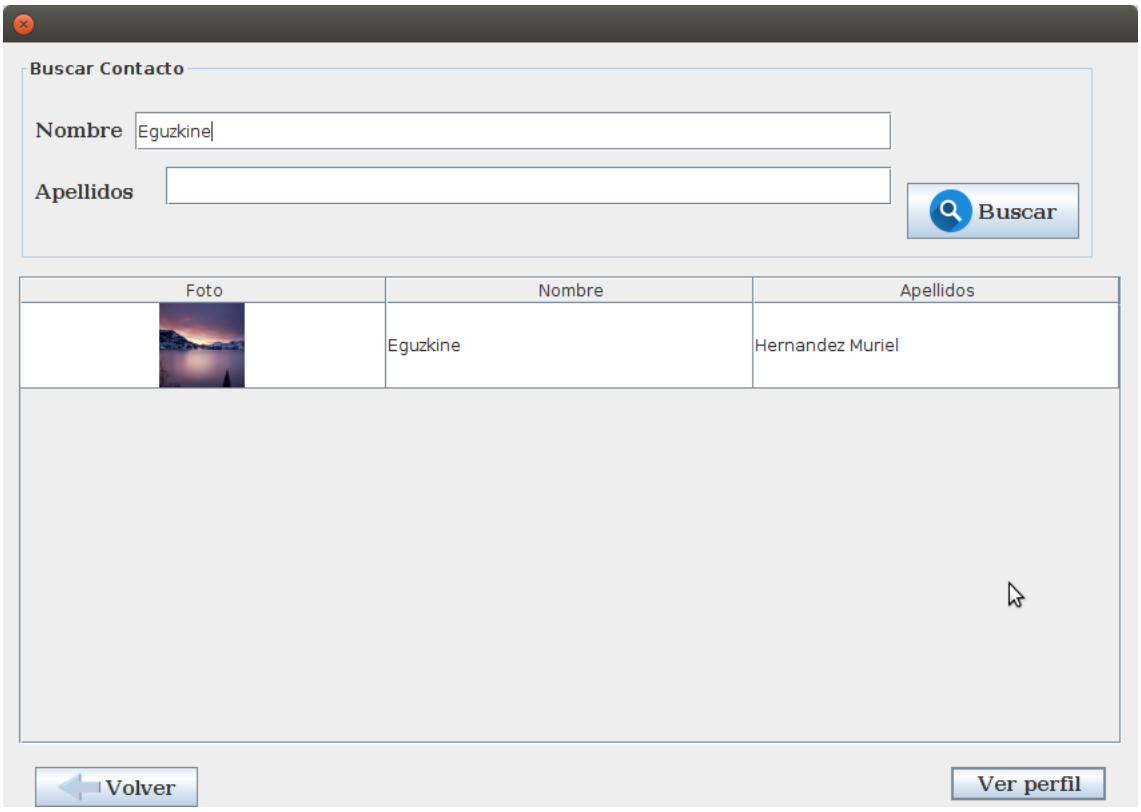

Ilustración 122: Pantalla buscar Paciente

En la página de su perfil se visualizarán los mismos datos de diferente forma distribuida y estará disponible el botón de **Abrir Chat**.

Ilustración 123: Pantalla Perfil Paciente

6.13.- Perfil del usuario actual

El usuario que está usando la aplicación, puede acceder a editar su perfil en la pantalla de inicio.

Ilustración 124: Pantalla Mi Perfil

En esta ventana el usuario podrá editar su nombre, sus apellidos, su e-mail, su contraseña y su foto de perfil. Además, si es un usuario profesional, también tendrá la opción de modificar su lugar de trabajo actual y su especialidad.

Para que el usuario pueda actualizar su foto de perfil, ha de pulsar el botón **Cargar Foto**. Esto hará que se abra una ventana para que el usuario busque en su ordenador una imagen. Los formatos que se admiten en esta aplicación, son JPG, JPEG y PNG. Cuando el usuario haya escogido la imagen y pulse el botón **Abrir**, esta se cargará en un elemento local llamado fichero de tipo **File**. Este archivo se convierte en un objeto tipo **ImageIcon** la cual se carga directamente en el **JLabel** destinado para visualizar la imagen de perfil del usuario.

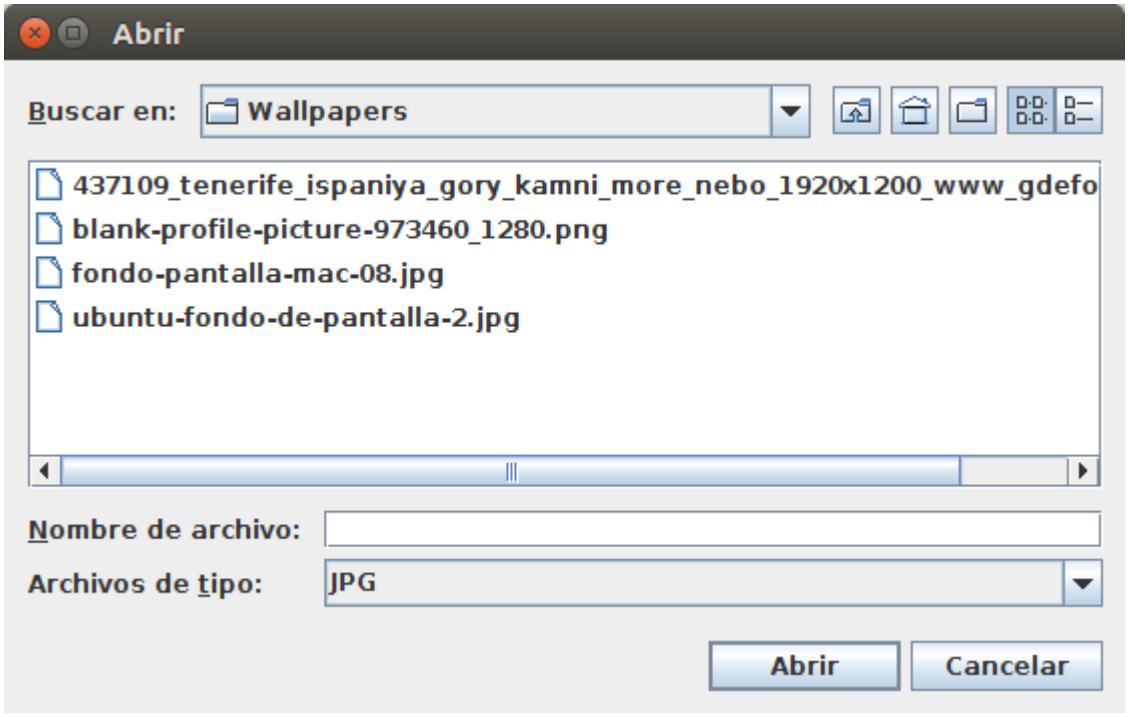

Ilustración 125: Buscador de imágenes

```
cargarFotobtn.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        int resultado;
        PantallaBuscarFichero buscarFich = new PantallaBuscarFichero();
        FileNameExtensionFilter filtro = new FileNameExtensionFilter("JPG", "jpg", "jpeg","JPEG", "png", "PNG" );
        buscarFich.pantalla.setFileFilter(filtro);

        resultado = buscarFich.pantalla.showOpenDialog(null);
        if (JFileChooser.APPROVE_OPTION == resultado) {
            fichero = buscarFich.pantalla.getSelectedFile();
            fotoPath = fichero.toString();
            try {
                ImageIcon icon = new ImageIcon(fichero.toString());
                ImageIcon icono = new ImageIcon(icon.getImage().getScaledInstance(lblFoto.getWidth(),
                                                    lblFoto.getHeight(), Image.SCALE_SMOOTH));
                lblFoto.setIcon(icono);

            }catch (Exception ex) {
                System.out.println("SALTA LA EXCEPCION DE LA FOTO");
                JOptionPane.showMessageDialog(null, "Algo ha ido mal en al cargar la foto.");
            }
        }

    }
});
```

Ilustración 126: Código modificación de imagén de pefil

Estos datos se guardarán en el momento en el que el usuario pulse el botón **Guardar cambios**. Para realizar la actualización de datos, se hace una comprobación de que los datos introducidos son correctos.

Primero, se comprueba que los campos **Nombre** y **Apellidos** no están vacíos. Si no es así, el proceso cesará y se visualizará el siguiente mensaje:

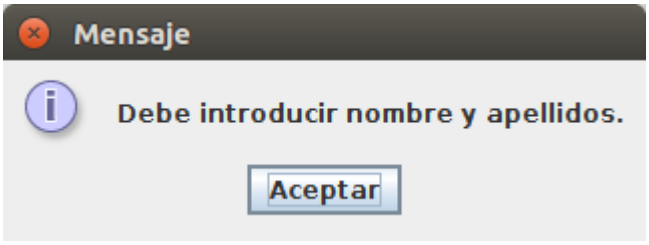

Ilustración 127: Mensaje introducir nombre y apellidos

Segundo, se comprueba que la contraseña introducida contiene al menos un número. Si no es así saldrá el siguiente mensaje:

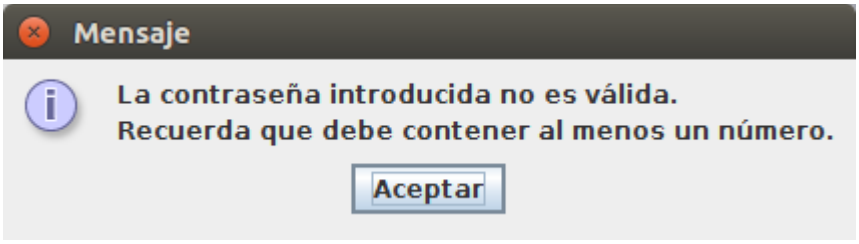

Ilustración 128: Mensaje contraseña no válida

Tercero, se comprueba que el e-mail introducido sigue las normas establecidas anteriormente. Si esto no se cumple saldrá el siguiente mensaje.

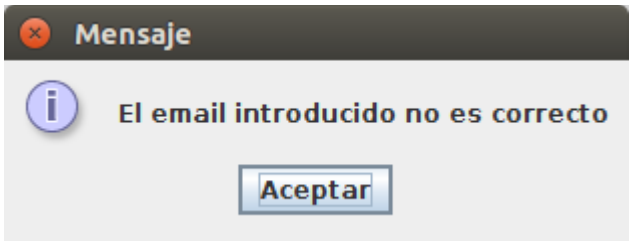

Ilustración 129: Mensaje e-mail incorrecto

En caso de que estos tres puntos se cumplen, los datos completos (no solo los modificados), se sobrescribirán con los campos que actualmente ve el usuario en pantalla. Además, estos datos también se actualizarán en la clase única que mantiene la aplicación para evitar la extracción continua de datos a la base de datos.

```
public void actionPerformed(ActionEvent e) {
    UsuarioDBProfesional usuarioDB = UsuarioDBProfesional.getInstancia();
    if (!nombre_text.getText().equals("") && !apellidos_text.getText().equals("")) {
        if (Utils.comprobarContraseña(new String(passwordField.getPassword()))){
            if (Utils.comprobarEmail(email_text.getText())) {
                usuarioDB.actualizarPerfil(UsuarioProfesional.getInstancia().getUsuario_id(), new String(passwordField.getPassword())
                        , nombre_text.getText(), apellidos_text.getText(), email_text.getText(), lugarTrabajo_text.getText(),
                        especialidad_text.getText(), fichero);
                UsuarioProfesional.getInstancia().actualizarPerfil(new String(passwordField.getPassword()),
                        nombre_text.getText(), apellidos_text.getText(), email_text.getText(),
                        lugarTrabajo_text.getText(), especialidad_text.getText(), fichero);
                JOptionPane.showMessageDialog(null, "Los datos se han guardado correctamente.");

            }else {
                JOptionPane.showMessageDialog(null, "El email introducido no es correcto");
            }
        }else {
            JOptionPane.showMessageDialog(null, "La contraseña introducida no es válida.\n"+
                    "Recuerda que debe contener al menos un número.");
        }
    }else {
        JOptionPane.showMessageDialog(null, "Debe introducir nombre y apellidos.");
    }
```

Ilustración 130: Código actualiza datos de perfil

Otra de las cosas que puede hacer el usuario dentro de su perfil, es ver, editar y borrar sus trayectorias profesionales. Para acceder a esto, deberá pulsar el botón **Trayectoria** situado en parte inferior izquierda de la pantalla.

6.14.- Trayectoria del usuario

Dentro de esta pantalla, el usuario actual podrá añadir, borrar y ver las trayectorias actuales que tiene en su perfil de SanidApp.

Lo primero que visualiza el usuario son las trayectorias que tiene actualmente incluidas en su perfil. Estos datos aparecen en un panel de resultado como los anteriormente explicados, pero que en este caso, se visualizan campos como fecha de inicio, fecha de finalización, puesto y lugar de trabajo.

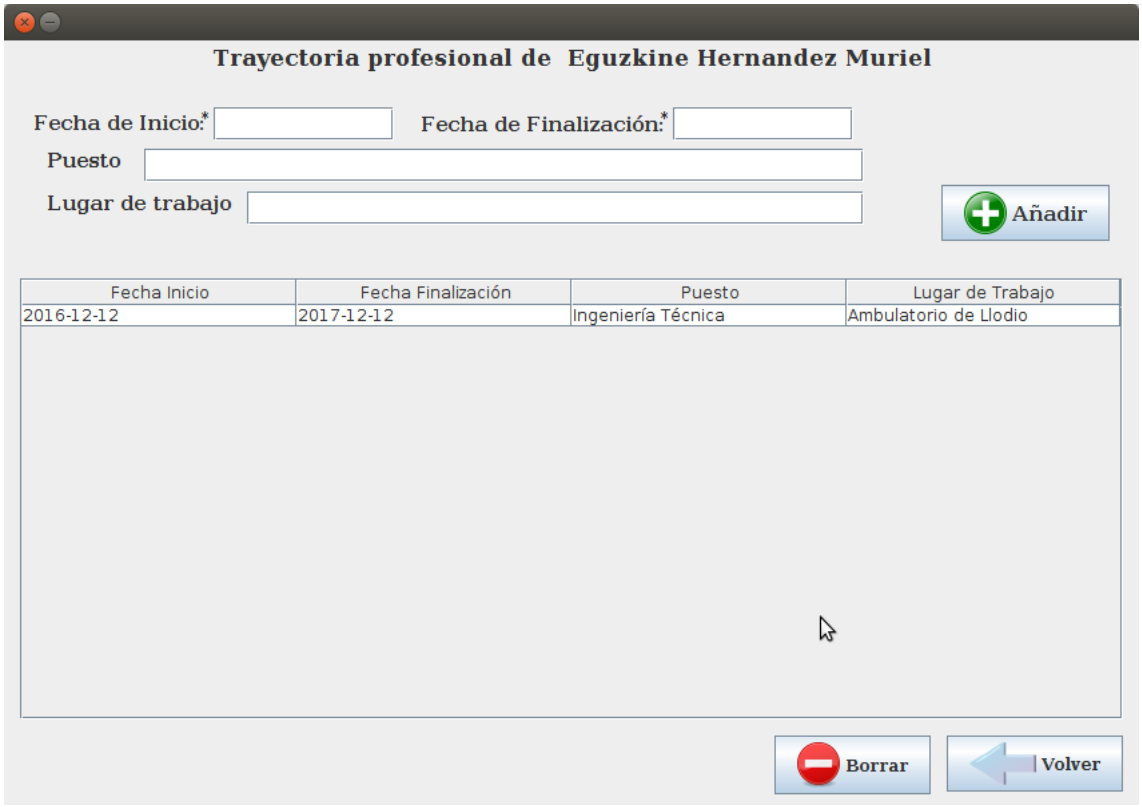

Ilustración 131: Pantalla Trayectorias del usuario

Añadir una Trayectoria

Para añadir una trayectoria deberá cumplimentar los campos que se disponen en el panel superior y seguidamente pulsar el botón Añadir. Esto realizará el siguiente proceso:

1. Comprueba que los campos requeridos están todos rellenados. Si no fuera así, el proceso cesaría y aparecería el siguiente mensaje:

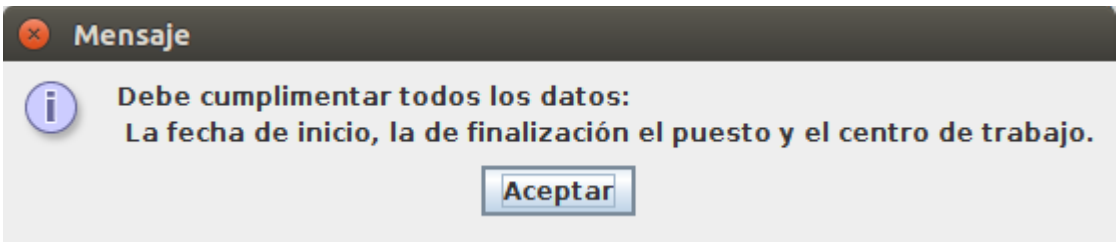

Ilustración 132: Mensaje cumplimentar datos Añadir Trayectoria

2. Se validan los formatos introducidos en los campos **Fecha Inicio** y **Fecha Final**, que corresponden a la fecha de inicio y final del contrato a añadir. El formato correcto en el que deben estar las fechas, es "yyyy-MM-dd". Si no fuera así, se mostraría el siguiente mensaje:

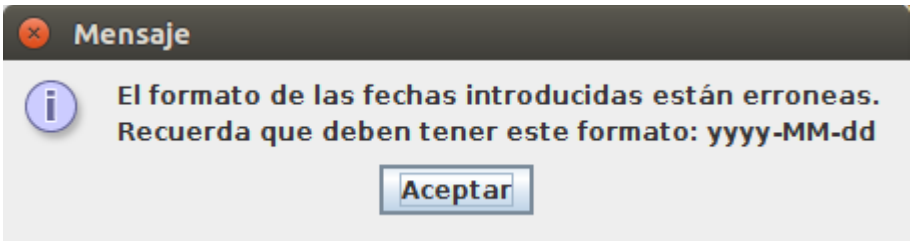

Ilustración 133: Mensaje formato fecha inválida Añadir Trayectoria

3. Una vez comprobado el formato de las fechas, el sistema se cerciora de que la fecha de inicio introducida por el usuario es anterior a la fecha final. Si esto no se cumpliese, se informaría mediante mensaje.

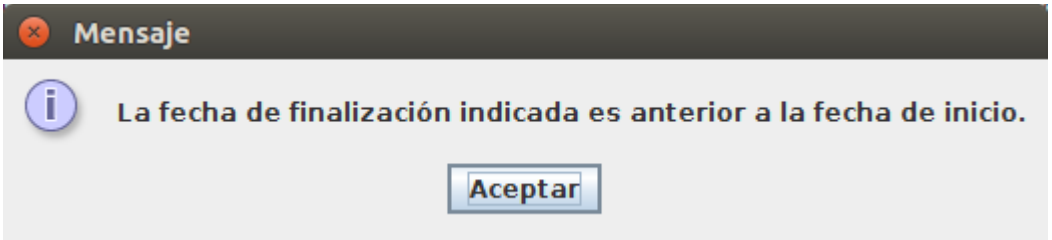

Ilustración 134: Mensaje fecha inicio y fecha final incorrectas

4. Finalmente, si es la primera vez que ha añadido una trayectoria desde que ha abierto esta ventana, sale un mensaje en pantalla indicando que toda la información introducida en la aplicación debe ser verídica.

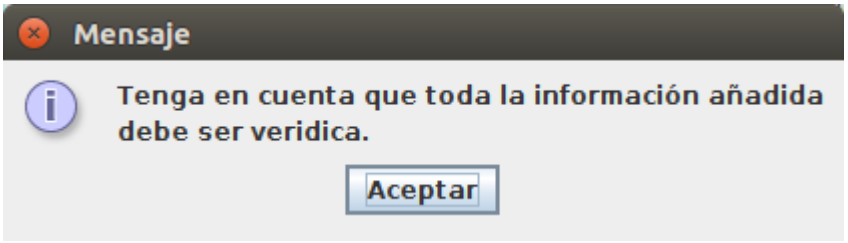

Ilustración 135: Mensaje de advertencia Añadir Trayectoria

Finalmente se recogen los valores de los campos introducidos por el usuario en el sistema y estos se introducen en la base de datos mediante una ejecución de inserción. Además, estos datos se incluyen en el modelo del panel de resultados para que aparezcan en pantalla.

```
public void actionPerformed(ActionEvent e) {
    //Comprobamos que todos los campos están completos
    if (Utils.comprobarCamposCompletosTrayectoria(textFechaInicio.getText(), textFechaFinal.getText(), textPuesto.getText(), textLugarTrabajo.getText())) {

        //Comprobamos la fecha
        if (Utils.validarFecha(textFechaInicio.getText()) && Utils.validarFecha(textFechaFinal.getText())) {
            //Las fechas son válidas.

            //Comprobamos que la fecha de incio sea menor o igual a la final
            System.out.println("Inicio: " + textFechaInicio.getText() + " Final: " + textFechaFinal.getText());
            if (Utils.comprobarOrdenFechas(textFechaInicio.getText(), textFechaFinal.getText())) {
                //Salta el mensaje de advertencia.
                if (cont==0) {
                    JOptionPane.showMessageDialog(null, "Tenga en cuenta que toda la información añadida\n"
                            + "debe ser veridica.");
                    cont++;
                }
                //Se crea la Trayectoria y se guarda en la BD
                String fechaInicio = textFechaInicio.getText();
                String fechaFinal = textFechaFinal.getText();
                String puesto = textPuesto.getText();
                String lugar_trabajo = textLugarTrabajo.getText();
                Trayectoria nueva = new Trayectoria (0, null, null, null, null);
                nueva.setFecha_final(fechaFinal);
                nueva.setFecha_inicio(fechaInicio);
                nueva.setPuesto(puesto);
                nueva.setLugar_trabajo(lugar_trabajo);
                int trayectoria_id = TrayectoriaDB.getInstancia().añadeTrayectoria(nueva, UsuarioProfesional.getInstancia().getUsuario_id());
                nueva.setTrayectoria_id(trayectoria_id);
                tableModel.añadeElemento(nueva);
```

Ilustración 136: Código Añadir Trayectoria

Eliminar una Trayectoria

Para eliminar una trayectoria, es necesario tener seleccionada una de las que aparecen en el panel de resultados. Si no fuera así la aplicación lanzaría un mensaje informando del error.

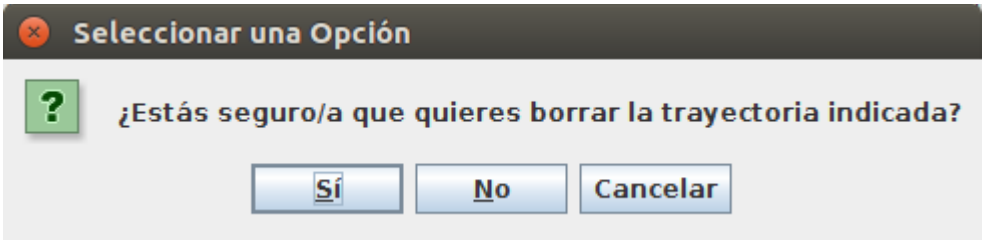

Ilustración 137: Confirmación de eliminación Trayectoria

En caso contrario, el sistema pregunta al usuario si realmente desea borrar ese registro. Si pulsa el botón **Sí**, la trayectoria seleccionada se eliminará de la base de datos y del modelo de resultados.

```
buttonBorrar.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {
        int index = table.getSelectedRow();
        if (index!=-1) { //Si tiene uno seleccionado
            Trayectoria eliminar = tableModel.getRow(index);
            tableModel.quitaElemento(index);
            int respuesta = JOptionPane.showConfirmDialog(null, "¿Estás seguro/a que quieres borrar la trayectoria indicada?");
            if (respuesta==0) {
                TrayectoriaDB.getInstancia().eliminarTrayectoria(eliminar);
                trayectorias.remove(index);
            }

        }else {
            JOptionPane.showMessageDialog(null, "Selecciona un elemento de la tabla para poder borrarlo.");
        }

    }
});
```

Ilustración 138: Código eliminación Trayectoria

6.15.- Cerrar sesión

En la pantalla de inicio el usuario tendrá la opción de cerrar la sesión en el momento en el precise. Cuando lo haga, el sistema mandará un paquete con el número identificador del usuario al servidor, para que este lo recoja y elimine al usuario de su lista de conectados. Este proceso, se realiza exactamente igual que en los puntos anteriores donde se explica detalladamente el funcionamiento de los sockets. La pantalla que tendrá entonces el usuario será de la **Login**.

```
JButton btnCerrarSesin = new JButton("Cerrar Sesión");
btnCerrarSesin.setFont(new Font("DejaVu Serif", Font.BOLD, 15));
btnCerrarSesin.setBounds(663, 54, 148, 29);
btnCerrarSesin.addActionListener(new CerrarSesion());
panel.add(btnCerrarSesin);
```

Ilustración 139: Código botón Cerrar Sesión

```
private class CerrarSesion implements ActionListener  {

    public void actionPerformed(ActionEvent e) {
        try {
            Socket miSocket = new Socket("127.0.0.1", 9003);
            Socket miSocket = new Socket("172.20.10.5", 9999);

            ObjectOutputStream flujoSalida = new ObjectOutputStream(miSocket.getOutputStream());

            EnvioIP envioIP = new EnvioIP();
            envioIP.setIp("127.0.0.1");
            envioIP.setIp("172.20.10.5");
            envioIP.setUsuario_id(UsuarioProfesional.getInstancia().getUsuario_id());

            flujoSalida.writeObject(envioIP);
            miSocket.close();
            Login login = new Login();
            login.setVisible(true);
            dispose();
        }catch (Exception ex) {
            System.out.println(ex.getMessage());
        }

    }

}
```

Ilustración 140: Código Cerrar Sesión

6.16.- Pantalla Inicio Paciente

El paciente tendrá la siguiente pantalla de inicio:

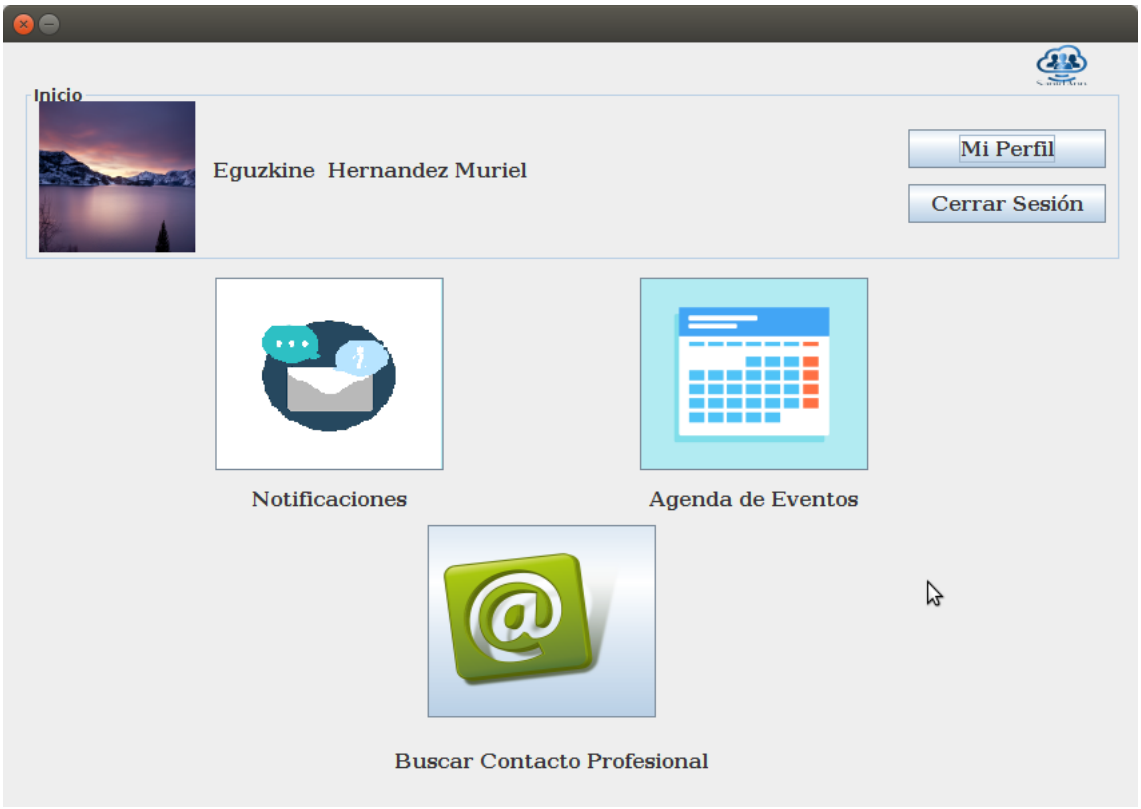

Ilustración 141: Pantalla Inicio Paciente

El funcionamiento de esta ventana y las que se mostrarán en caso de que accione los botones como en caso de los profesionales, será exactamente igual que en las secciones anteriores.

7.- Verificación y evaluación

A continuación, se detallan las pruebas definitivas hechas en el sistema. En cada prueba se documenta la acción realizada y se compara el resultado obtenido con el resultado esperado. Finalmente, se hará un balance para concluir si se ha conseguido llegar a los objetivos propuestos.

7.1.- Registrar un usuario

En esta tabla se hace un registro de todas las funciones y comportamientos que puede tener el usuario frente al sistema, al intentar registrarse y como responde este a las acciones planteadas.

Acción	Resultado esperado	Resultado obtenido
El usuario escoge la opción paciente	El texto "Número de trabajador" cambia y pasa a ser "Número de DNI" y el campo destinado a especificar la especialidad del profesional, queda deshabilitada.	Cuando el usuario cambia la opción de Profesional a Paciente, el texto donde hay que introducir el número identificador cambia y el campo de especialidad queda inhabilitado.
El usuario pulsa el botón "Crear" sin rellenar ningún campo.	Se comprueba que los campos están vacíos y se visualiza un mensaje de error campos vacíos de nombre y apellidos.	Se comprueba que los campos están vacíos y se visualiza un mensaje de error campos vacíos de nombre y apellidos.
El e-mail no tiene la estructura de un e-mail.	Se comprueba la estructura del e-mail y se visualiza un mensaje en pantalla.	Se muestra un mensaje de error en pantalla informando que el e-mail es erróneo.
La contraseña introducida para ser registrada no contiene ningún carácter numérico.	Se visualiza en pantalla un mensaje de error informando del problema	Se visualiza en pantalla un mensaje de error informando del problema.
Las contraseñas que deben ser iguales no lo son.	Se muestra un mensaje en pantalla anunciando el error.	Se muestra un mensaje en pantalla anunciando el error.
El número de usuario que se quiere registrar en SanidApp no corresponde a ningún profesional ni paciente de la institución.	Hace una consulta en las tablas "Pacientes" y "Profesionales" respectivamente y se visualiza un mensaje de error de usuario no encontrado.	Se realiza la consulta y se visualiza el mensaje de error de usuario no encontrado.

El número de usuario que se quiere registrar como paciente o profesional, ya está registrado en la aplicación.	Se comprueba en la tabla "Usuario" de la base de datos si el identificador indicado está. Se visualiza mensaje de error usuario ya registrado.	Se realiza correctamente la consulta y se visualiza el mensaje de error usuario ya registrado en pantalla.
El usuario introduce correctamente los datos necesarios para darse de alta en la aplicación.	Se crea un nuevo registro en las tablas respectivas al paciente o profesional y se visualiza un mensaje de bienvenida en pantalla.	Se introduce el nuevo usuario en la base de datos y se visualiza un mensaje de bienvenida en pantalla.
Se limpian los campos donde el usuario ha escrito cuando sale en pantalla un error	Los campos quedan vacíos.	Los campos se mantienen igual.

Tabla 41: Pruebas Registrar un usuario

7.2.- Iniciar sesión

Esta tabla analiza las acciones que puede realizar un usuario cuando quiere iniciar sesión, como debe responder el sistema y como realmente responde.

Acción	Resultado esperado	Resultado obtenido
El usuario introduce un número de usuario no registrado.	Se comprueba en la base de datos que el usuario está registrado. Se visualiza un mensaje de usuario no registrado en pantalla.	Se hace la comprobación en la base de datos y se visualiza el mensaje de usuario no registrado en pantalla.
El usuario registrado introduce su número identificativo correctamente, pero la contraseña incorrecta.	Se comprueba que la contraseña coincide con el número identificador. Se muestra un mensaje de error de contraseña en pantalla.	Se hace la comprobación en la base de datos y se visualiza el error de contraseña en pantalla.
El usuario introduce correctamente el número identificador y la contraseña.	Se comprueba en la base de datos que el usuario y la contraseña coinciden. Se abre la ventana de inicio de la aplicación correspondiente al usuario que ha iniciado sesión.	Se realiza la comprobación en la base de datos y se visualiza en pantalla la ventana de inicio del usuario que ha iniciado sesión.

Se limpian los campos donde el usuario ha escrito cuando sale en pantalla un error	Los campos quedan vacíos.	Los campos se mantienen igual.

Tabla 42: Pruebas Iniciar Sesión

7.3.- Recuperación de contraseña

Esta tabla recoge todas las acciones posibles que un usuario puede realizar en esta ventana. También se explica cómo debería funcionar el sistema en cada una de ellas y como realmente responde.

Acción	Resultado esperado	Resultado obtenido
El usuario en la página de Login pulsa el botón "Recuperar Contraseña"	La pantalla de Login se cierra y se visualiza la pantalla de Recuperar Contraseña	Se cierra la pantalla Login y se abre la pantalla de Recuperar Contraseña.
El usuario no complementa todos los campos.	Se comprueba que los campos están rellenados. Se muestra un mensaje de error de campos vacíos.	Se realiza la comprobación de los campos vacíos y se visualiza en pantalla el error de campos vacíos.
El identificador de usuario introducido no es numérico.	Se comprueba que el texto del identificador es numérico. Sale un mensaje de error usuario no numérico.	Se realiza la comprobación y se visualiza el error de usuario no numérico.
El usuario introduce un número identificador que no coincide con el e-mail introducido.	Se comprueba que el usuario y el e-mail coinciden con los datos de la base de datos. Se visualiza error usuario e e-mail.	Se realiza la comprobación. Se visualiza el mensaje de usuario o e-mail erróneo.
La nueva contraseña introducida no contiene mínimo un carácter que sea numérico.	Mensaje de informativo de contraseña inválida.	Se visualiza un mensaje de error contraseña inválida.
Las contraseñas introducidas que deben ser iguales, no lo son.	Mensaje en pantalla de contraseñas diferentes.	Se muestra un mensaje en pantalla de contraseñas diferentes.

El usuario introduce correctamente los datos para recuperar la contraseña.	Se modifica la contraseña adjudicada a ese usuario correctamente, se visualiza el mensaje "Contraseña cambiada satisfactoriamente", se cierra la pantalla actual y se abre la pantalla Login.	Se modifica la contraseña en la base de datos, se visualiza un mensaje para informar al usuario que el proceso se ha realizado correctamente, se cierra la pantalla actual y se abre la pantalla Login.
Si se visualiza algún mensaje de error, los campos escritos por el usuario se borran.	Campos vacíos para que el usuario pueda volver a introducir los datos.	Los campos se mantienen igual.

Tabla 43: Pruebas Recuperación de contraseña

7.4.- Pantalla de inicio

El usuario en la pantalla de inicio tiene poca interactividad, ya que todas las acciones posibles abren otras ventanas que realizan otras funciones.

Acción	Resultado esperado	Resultado obtenido
El usuario inicia sesión y se abre la pantalla Inicio.	Se cargan los datos del usuario acreditado correctamente en el sistema.	El usuario inicia sesión correctamente y la ventana se visualiza correctamente.
Usuario pulsa sobre un botón de la pantalla.	Sistema cierra la pantalla Login y se visualiza la pantalla correspondiente.	Desaparece la pantalla Login y se visualiza la pantalla correspondiente al botón pulsado.
En la pantalla se visualizan los datos del usuario.	Cuando el usuario inicia sesión se cargan los datos de la base de datos en el sistema y se visualizan en pantalla la foto de perfil del usuario, su nombre y sus apellidos. Si es profesional, también su especialidad.	Se obtienen correctamente los datos del usuario y se visualizan en pantalla.

Tabla 44: Pruebas Pantalla de inicio

7.5.- Ver Perfil

En esta tabla se presentan las pruebas de las acciones disponibles que tiene el usuario para poder realizar y como responde el sistema en cada una de ellas.

Acción	Resultado esperado	Resultado obtenido
Se visualizan los datos actuales del usuario.	Se visualizan los datos cargados en el sistema, tantos los datos de tipo String, como la foto de perfil.	Los datos se visualizan correctamente.
El usuario deja en blanco los campos nombre y apellidos.	Mensaje de error campos nombre y apellidos vacíos.	Mensaje de error campos nombre y apellidos vacíos.
El usuario modifica el email y no tiene una estructura correcta.	Mensaje de error en el e-mail.	Mensaje informando al usuario de que el e-mail no tiene una estructura correcta.
El usuario modifica su contraseña sin poner mínimo un carácter numérico.	Se visualiza en pantalla un mensaje de error de estructura de contraseña.	Se visualiza el error de contraseña en pantalla.
El usuario pulsa el botón cambiar foto de perfil	Se abre el navegador de archivos y solo se visualizan en pantalla las imágenes con los formatos indicados.	El usuario puede visualizar los archivos que tiene en el ordenador, pero únicamente con los formatos de archivos indicados.
El usuario selecciona una foto nueva de perfil.	La foto elegida en la pantalla del navegador de archivos se carga en la pantalla del perfil.	La foto que ha elegido el usuario se visualiza en pantalla donde se encontraba la foto anterior.
El usuario complementa correctamente los campos y pulsa el botón guardar datos.	Los datos se modifican en la base de datos y también en el sistema.	Los datos se sobrescriben en la base de datos y en el sistema.
El usuario pulsa el botón Trayectoria	La ventana para ver la trayectoria del usuario se abre y se cierra la actual.	Se visualiza la ventana de trayectorias del usuario actual. Se cierra la pantalla de su perfil.
Si se visualiza algún mensaje de error, los campos escritos por el usuario se borran.	Campos vacíos para que el usuario pueda volver a introducir los datos.	Los campos se mantienen igual.

Tabla 45: Pruebas Ver Perfil

7.6.- Trayectorias del usuario

En esta sección se analizan todas las funciones que el usuario puede hacer en la pantalla de sus trayectorias y como el sistema reacciona a cada una de esas actividades.

Acción	Resultado esperado	Resultado obtenido
Se abre la ventana de Trayectorias.	Se visualizan correctamente las trayectorias vinculadas al usuario de mayor a menor.	Los datos se visualizan correctamente.
El usuario pulsa el botón añadir pero no tiene ningún dato completado.	Mensaje de error campos vacíos.	Mensaje de error campos vacíos.
El usuario no introduce correctamente las fechas en el formato definido.	Mensaje de error Formato de fecha no válido.	Se visualiza en pantalla un mensaje el mensaje de error de formato de fecha no válido.
La fecha de inicio es posterior a la fecha final indicada.	Mensaje de error de fecha inicial posterior a fecha final.	El usuario visualiza en pantalla el mensaje de error correspondiente fecha inicial posterior a fecha final.
El usuario introduce correctamente los datos y pulsa el botón añadir.	Se introduce en la base de datos un nuevo registro con la nueva trayectoria vinculada al usuario actual. Se visualiza un mensaje en pantalla de advertencia. Todas las trayectorias insertadas deben ser verídicas.	Se registra la nueva trayectoria en la base de datos y se visualiza el mensaje de advertencia.
El usuario no selecciona ninguna trayectoria del panel de resultados y pulsa el botón borrar.	El sistema no debe reaccionar.	El sistema no reacciona.
El usuario selecciona una trayectoria y pulsa el botón borrar.	Debe salir un mensaje emergente volviendo a preguntar al usuario su decisión de borrar el dato.	Se visualiza el mensaje preguntando al usuario sobre su decisión.
El usuario pulsa Sí en el mensaje emergente.	Se borra la trayectoria seleccionada en la base de datos.	La trayectoria seleccionada se elimina en la base de datos y del panel de resultados.

El usuario declina la decisión de borrar la trayectoria seleccionada.	El sistema no ejecuta ningún cambio.	El sistema sigue igual y el dato no se elimina de la tabla de resultados.
Si se visualiza algún mensaje de error, los campos escritos por el usuario se borran.	Campos deben estar vacíos para que el usuario pueda volver a introducir los datos.	Los campos se mantienen igual.
El usuario pulsa el botón volver	Se deja de visualizar la ventana actual y se visualiza la pantalla de Inicio.	Se deja de visualizar la ventana actual se abre la ventana de inicio del usuario.

Tabla 46: Pruebas Trayectoria del usuario

7.7.- Notificaciones

Se procede a describir las funciones que el usuario puede realizar en la pantalla de Notificaciones del sistema y como este reacciona.

Acción	Resultado esperado	Resultado obtenido
Se abre la ventana de Notificaciones.	Se visualizan en pantalla las notificaciones vinculadas al usuario en pantalla.	Los datos se visualizan correctamente en pantalla.
El usuario pulsa el botón Abrir Chat sin seleccionar previamente una notificación.	Se visualiza en pantalla un mensaje de error.	Se visualiza en pantalla un error avisando al usuario que debe seleccionar previamente una notificación para abrir el chat que le corresponde.
El usuario selecciona una notificación y seguidamente pulsa el botón Abrir Chat	El sistema reconoce el chat que quiere abrir el usuario mediante la notificación. Se cierra la pantalla actual y se abre la pantalla de Chat que le corresponde.	El sistema cierra la pantalla de Notificaciones y abre la pantalla de chat que le corresponde al usuario emisor de la notificación.
El usuario pulsa el botón Volver.	El sistema cierra la pantalla de Notificaciones y se visualiza la pantalla de Inicio.	Se cierra la pantalla actual y se vuelve a la pantalla de Inicio.

Tabla 47: Pruebas Notificaciones

7.8.- Buscar Contactos

En esta sección se explicará cómo reacciona el sistema a las búsquedas realizadas por un usuario.

Acción	Resultado esperado	Resultado obtenido
El usuario pulsa el botón buscar sin rellenar ningún campo de los filtros ofrecidos.	Se buscan todos los usuarios (profesionales o pacientes, según en qué buscador estén) y se visualizan en el panel de resultados.	El sistema busca todos los contactos del tipo del buscador y los añade al panel de resultados.
El usuario completa algunos de los campos disponibles en los filtros y pulsa el botón Buscar.	El sistema busca usuarios que coincidan con los filtros establecidos. Si los encuentra los visualiza en el panel de resultados. Si no lo encuentra se visualiza un mensaje en pantalla de usuario no encontrado.	El sistema encuentra usuarios que coinciden con los filtros y los visualiza en el panel de resultados.
		El sistema no encuentra ningún usuario que coincida con los filtros establecidos. Se muestra en pantalla un mensaje de usuario no encontrado.
El usuario pulsa el botón Ver perfil sin seleccionar previamente un contacto.	El sistema lanza un mensaje informando al usuario que debe seleccionar previamente un contacto para visualizar su perfil.	El sistema visualiza el mensaje de aviso para avisar al usuario que debe seleccionar un contacto.
El usuario selecciona un contacto y pulsa el botón Ver Perfil.	El sistema carga los datos del usuario en cuestión en el sistema, se cierra la ventana actual y se visualiza la pantalla de perfil del usuario.	El sistema cierra la ventana actual, carga los datos del usuario seleccionado en la nueva ventana.
Si se visualiza algún mensaje de error, los campos escritos por el usuario se borran.	Campos vacíos para que el usuario pueda volver a introducir los datos.	Los campos se mantienen igual.
El usuario pulsa el botón Volver.	El sistema cierra la ventana actual y vuelve a la pantalla de Inicio.	El sistema cierra la ventana actual y vuelve a la pantalla de Inicio.

Tabla 48: Pruebas Buscar Contactos

7.9.- Ver perfil del Contacto

Esta tabla recoge todas las pruebas realizadas y sus resultados relacionados con los datos del contacto seleccionado en la búsqueda.

Acción	Resultado esperado	Resultado obtenido
Se abre la ventana de perfil de contacto.	Se deben visualizar todos los datos del perfil seleccionado en la búsqueda correctamente.	Se visualizan correctamente todos los datos del contacto. Incluido su foto de perfil.
En caso de que sea el perfil de un profesional, el usuario pulsa el botón Ver Trayectoria.	El sistema cierra la pantalla actual. Abre otra ventana donde se visualizan todas las trayectorias vinculadas con el contacto.	El sistema cierra la ventana actual y abre la ventana de las trayectorias del usuario.
El usuario pulsa el botón Abrir Chat.	La ventana actual se cierra y se visualiza la ventana del chat con los mensajes vinculados al usuario actual y al contacto.	Desaparece la ventana actual y se abre la ventana del chat donde se cargan correctamente los mensajes enviados hasta el momento.
El usuario pulsa el botón Volver.	Se cierra la ventana actual y vuelve a la pantalla de Inicio.	La ventana actual se cierra y regresa a la pantalla de Inicio.

Tabla 49: Pruebas Ver Perfil del Contacto

7.10.- Chat

Esta tabla recoge todos los resultados de las pruebas realizadas con la función del Chat.

Acción	Resultado esperado	Resultado obtenido
Se abre la ventana del chat	Se cargar los mensajes del usuario con el contacto en el panel de mensajes.	Se visualizan correctamente todos los datos y mensajes.
El usuario escribe un mensaje al usuario (desconectado) y pulsa el botón Enviar.	El mensaje se visualiza en el panel de mensajes con la fecha, la hora, el nombre y apellidos del emisor del mensaje. Este mensaje debe enviar por el Socket y llegarle al servidor. El servidor lo descompone, lo registra en la base de datos y añade una notificación al usuario receptor del mensaje. El campo donde ha escrito el texto se debe limpiar.	El mensaje se visualiza en pantalla. Se envía por el servidor y cuando le llega a este, guarda el mensaje en la base de datos, comprueba que el usuario no está conectado y registra una notificación con la fecha, la hora y los datos del emisor en la base de datos del receptor. El campo donde ha escrito el mensaje, queda limpio.
El usuario escribe un mensaje al usuario (conectado) y pulsa el botón Enviar.	El mensaje se visualiza en el panel de mensajes con la fecha, la hora, el nombre y apellidos del emisor del mensaje. Este mensaje llega al servidor y es redirigido por otro puerto al usuario según su IP. Se visualiza en la pantalla del usuario receptor.	El mensaje se visualiza en el panel de mensajes y se envía al servidor. Este comprueba que el usuario está conectado y redirige el mensaje a su aplicación. El usuario receptor del mensaje lo visualiza en su sistema local.
El usuario recibe un mensaje.	El panel de mensajes se actualiza y al final de este se visualiza el mensaje que le ha enviado el otro usuario.	El mensaje recibido se visualiza en el panel de mensajes. Se posiciona al final del panel.
La letra del mensaje debe ser de color negro.	El sistema debe visualizar el texto del mensaje en color negro.	El sistema visualiza el texto del mensaje en color azul clarito.
El usuario pulsa el botón volver.	El sistema abre la ventana de Inicio y cierra la actual	El sistema abre la ventana de Inicio y cierra la actual.

Tabla 50: Pruebas Chat

7.11.- Ver Eventos

En esta tabla, se explica cómo reacciona el usuario antes todas las acciones que realiza el usuario cuando ve o busca un evento.

Acción	Resultado esperado	Resultado obtenido
Se abre la ventana Eventos	El sistema carga en el panel de resultados de todos los eventos vinculados con el usuario.	El sistema visualiza correctamente los eventos.
El usuario pulsa el botón buscar sin completar ningún campo.	Se coge la fecha actual y se buscan los eventos de ese día que el usuario guarda en la base de datos.	Se visualizan en el panel de resultados los eventos que tiene para el día actual.
El usuario indica solo una parte de la hora (hora o minuto) y pulsa el botón Buscar.	El sistema identifica solo una parte de la hora pero no busca los eventos por hora, ya que no está completa.	El sistema no tiene en cuenta la hora o minuto seleccionado, pero sí cuenta los demás filtros. Visualiza los eventos relacionados en el panel.
El usuario completa correctamente el o los filtros que quiere aplicar a la búsqueda y pulsa el botón Buscar.	El sistema identifica los filtros y busca y visualiza los eventos que coincidan con los filtros definidos.	Se visualizan en el panel de eventos los resultados de la búsqueda según los filtros indicados.
El usuario pulsa Borrar Evento sin haber seleccionado previamente un evento.	El sistema muestra en pantalla un mensaje informativo de evento no seleccionado.	Se visualiza un mensaje en pantalla informando al usuario que tiene que seleccionar previamente un evento para poder borrarlo.
El usuario selecciona un evento y pulsa el botón Borrar.	El sistema lanza un mensaje para que el usuario reitere su decisión de borrar el evento.	Se visualiza un mensaje para que el usuario reitere su decisión de borrar el evento.
El usuario decide definitivamente borrar el evento.	El evento se borra de la base de datos y del panel de eventos.	El evento se borra de la base de datos y por lo tanto, se desvincula a todos los usuarios que tenía vinculados. También se elimina del panel de eventos.

El usuario selecciona un evento y pulsa el botón ver detalles.	El sistema muestra un mensaje emergente indicando la descripción del evento y los nombres y apellidos de las personas vinculadas.	Se visualiza un mensaje en pantalla indicando la descripción del evento seleccionado y las personas vinculadas.
Pulsa el botón añadir evento.	Se cierra la ventana actual y se abre la ventana encargada de añadir los eventos.	La pantalla actual se cierra se visualiza la ventana de añadir los eventos.
Tras cada búsqueda los campos de los filtros quedan vacíos.	Los campos de las búsquedas y la hora deben quedar vacíos.	Los filtros no se modifican tras la última búsqueda.

Tabla 51: Pruebas Ver Eventos

7.12.- Añadir eventos

En esta tabla se explican las acciones y los resultados del sistema vinculados con añadir un evento.

Acción	Resultado esperado	Resultado obtenido
Se abre la ventana Eventos	Solo se visualiza en pantalla el primer panel donde el usuario debe introducir información sobre el evento a crear.	Se visualiza el panel necesario para crear el evento.
El usuario pulsa el botón Crear Evento sin haber cumplimentado todos los campos.	Mensaje de campos no cumplimentados.	Se visualiza en pantalla un mensaje informando al usuario que debe cumplimentar todos los campos para la creación del evento.
El usuario complementar correctamente todos los campos.	El evento se registra en la base de datos vinculando al usuario actual y se visualiza un segundo panel para vincular más usuarios.	El evento se guarda en la base de datos con el usuario actual vinculado y se visualiza un segundo panel para vincular más participantes al evento.
El usuario pulsa el botón Buscar del segundo panel sin haber completado ningún campo de los filtros.	Se visualizan todos los contactos que no se hayan vinculado anteriormente al evento creado y que coincida con el tipo de usuario escogido.	Se visualizan en el panel de resultado de participantes todos los usuarios (pacientes o profesionales) del sistema, que no se hayan vinculado anteriormente al evento.

El usuario completa alguno de los campos de los filtros y pulsa el botón Buscar.	El panel de resultados se limpia y se añaden los usuarios que coincidan con los filtros establecidos.	Se borran los resultados de la búsqueda anterior y aparecen los nuevos resultados de los usuarios que coinciden con los filtros establecidos y que no se han vinculado ya al evento creado.
El usuario pulsa el botón Añadir Participante sin haber seleccionado previamente un participante de la tabla.	El sistema muestra un mensaje para que el usuario seleccione antes el participante que quiere vincular al evento.	Se visualiza en pantalla un mensaje que explica que el usuario debe seleccionar previamente un participante para vincularlo al evento.
El usuario selecciona un participante y pulsa el botón Añadir Participante.	El participante seleccionado es vinculado al evento y añadido a la base de datos. El participante se borra del panel de resultados.	Se guarda en la base de datos la vinculación con el participante seleccionado. Se elimina del panel de búsqueda este participante.
Después de cada búsqueda, los campos se vacían.	Se borra el texto de los filtros después de cada búsqueda.	El sistema no borra el texto de los filtros de búsqueda.

Tabla 52: Pruebas Añadir Eventos

8.- Conclusiones

Este apartado está destinado a hacer una valoración objetiva y personal del proyecto. Es una parte indispensable, ya que es el momento en el que hay que puntualizar los puntos positivos y negativos que se han vivido vinculados a este Trabajo de Fin de Grado.

Además, se anotan los cambios llevados a cabo durante todo el periodo de realización del proyecto.

8.1.- Cambios

Los cambios que se han realizado durante todo el proyecto, han sido de diferentes ámbitos. Se explicarán los cambios de planificación, herramientas utilizadas, de diseño y desarrollo.

8.1.1.- Planificación VS. Realidad

La planificación que se impuso en un principio se ha modificado. Al comienzo del proyecto, se propuso una planificación de las tareas en formato cascada. Es decir, primero, se haría todo el diseño de la aplicación, tanto bases de datos, como prototipos del sistema a desarrollar, seguidamente se haría la implementación completa y finalmente las pruebas definitivas. Pero más adelante, se concluyó que la metodología **Scrum** era más servicial para este proyecto.

Esta metodología se basa en poner pequeños objetivos para ir obteniendo resultados reales del proyecto. En este caso, la parte de diseño, implementación y pruebas se ha llevado a cabo en cada **Sprint**. Por lo que el cambio ha sido satisfactorio.

Planificación inicial.

Esta es la planificación que se propuso al inicio del proyecto.

Actividad	Fecha de inicio	Fecha de finalización
1.- Inicio	**15/04/2019**	**17/04/2019**
1.1.- Planteamiento del tema 1.2.- Selección del software 1.3.- Instalación del software 1.4.- Reunión con la tutora	15/04/2019 16/04/2019 16/04/2019 17/04/2019	15/04/2019 16/04/2019 16/04/2019 17/04/2019
2.- Formación	**18/04/2019**	**21/04/2019**
2.1.- Charlas con las personas ligadas al proyecto 2.2.- Investigación sobre el tema	18/04/2019 18/04/2019	19/04/2019 21/04/2019
3.- DOP	**22/04/2019**	**06/05/2019**
3.1.- Creación del documento 3.2.- Reunión con la tutora	22/04/2019 06/05/2019	29/04/2019 06/05/2019

4.- Diseño	**07/05/2019**	**13/05/2019**
4.1.- Diseño de la interfaz gráfica	07/05/2019	07/05/2019
4.2.- Diseño de usabilidad de la aplicación	08/05/2019	09/05/2019
4.3.-Diseño de base de datos	08/05/2019	09/05/2019
4.4.- Diseño de diagrama de clases	10/05/2019	10/05/2019
4.5.- Reunión con la tutora	11/05/2019	11/05/2019
4.6.- Cambio de diseño	11/05/2019	12/05/2019
4.7.- Documentar diseños	13/05/2019	13/05/2019
5.- Implementación	**14/05/2019**	**13/07/2019**
5.1.- Implementación de bases de datos	14/05/2019	14/05/2019
5.2.- Preparar el servidor	14/05/2019	14/05/2019
5.3.- Programar	14/05/2019	13/07/2019
5.4.- Programar parte gráfica	14/05/2019	13/07/2019
5.5.- Hacer pruebas	14/05/2019	13/07/2019
5.6.- Reunión con la tutora	28/06/2019	28/06/2019
5.7.- Cambios en implementación	28/06/2019	13/07/2019
6.- Documentación final	**28/06/2019**	**08/07/2019**
6.1.- Hacer un manual de instrucciones	28/06/2019	30/06/2019
6.2.- Acabar de documentar la memoria	01/07/2019	13/07/2019
6.3.- Reunión con la tutora	08/07/2019	08/07/2019
7.- Finalización del proyecto	**09/07/2019**	**19/07/2019**
7.1.- Corrección final del proyecto	09/07/2019	09/07/2019
7.2.- Pruebas definitivas	10/07/2019	13/07/2019
7.3.- Entrega del proyecto	15/07/2019	22/07/2019
7.4.- Reunión con la tutora	19/07/2019	19/07/2019
8.- Estimación total	**15/04/2019**	**19/07/2019**

Tabla 53: Planificación inicial

Planificación final.

La planificación final se muestra en las tablas *Tabla 28: Actividad 1 planificación de días, Tabla 29: Registrar un usuario planificación de días, Tabla 30: Inicio, planificación de días, Tabla 31: Buscar Contacto planificación de días, Tabla 32: Ver mi Perfil planificación de días, Tabla 33: Calendario planificación de días, Tabla 34: Chat planificación de días, Tabla 35: Notificaciones planificación de días* y *Tabla 36: Actividad 2 planificación de días*.

8.1.2.- Cambios en Herramientas

La modificación en las herramientas propuestas al principio de la documentación, ha sido remodelada posteriormente ya que se han llevado a cabo varias de las tareas de diferente forma.

Entre esas tareas, se encuentra la construcción del Servidor. Al inicio del proyecto se pensó construir un servidor mediante un apache-TomCat, pero finalmente se construyó manualmente

una clase en Eclipse, definida mediante Sockets e hilos para establecer la comunicación con los clientes de la aplicación.

8.1.3.- Cambios en Diseño

Los cambios realizados en diseño han sido mínimos. Sí es verdad que hubo que hacer un cambio importante de la base de datos a mitad del proyecto, ya que las tablas **UsuarioProfesional** y **UsuarioPaciente** se habían creado por separado y eso complicaba en gran parte el desarrollo e implementación de la aplicación.

Por lo tanto, obligaba en la implementación a realizar clases duplicadas para un paciente y un profesional, ya que las sentencias contra la base de datos no eran iguales.

Por ejemplo, si un usuario quería modificar la foto de perfil, obligaba al sistema a saber si el usuario actual era un profesional o un paciente. En cambio, con la nueva base de datos no es necesario saber qué tipo de usuario es, ya que se dispuso una tabla general con los atributos comunes de todos los usuarios, entre ellos el identificador de cada uno, lo que permite vincular la tabla de **FotoUsuario** con la de **Usuario**.

8.1.4.- Cambios en Desarrollo

Algunos de los cambios del desarrollo son, como anteriormente se ha detallado, la forma de definir el servidor, la creación de grupos en los chats y la creación de los eventos.

Servidor

Antes de empezar con el desarrollo que complementa el servidor, se había pensado incluir la opción de creación de grupos en un chat, como tiene WhatsApp. Pero, el problema surgió en el momento en el que se creó el servidor mediante hilos y sockets. Al definirlo así para que el flujo y la comunicación con la aplicación cliente fueran más sencillos, la idea de creación de los grupos quedó apartada.

Creador de eventos

A la hora de definir los eventos, se pensó en utilizar una API para conectar la aplicación del cliente con la aplicación de Google Calendar, para así poder vincular los eventos creados en SanidApp al calendario personal del usuario en la aplicación de Google.

Pero esto no se pudo hacer, ya que el tiempo que estaba dispuesto para realizar esta función no era suficiente para el aprendizaje del funcionamiento de la API y conectar ésta con la aplicación local. Por lo que, se dispusieron los eventos locales, es decir, sin conectarlo a la aplicación de Google Calendar y diseñándolo en la base de datos.

8.2.- Resultado

Se puede decir que el resultado final es satisfactorio. Es un proyecto en el que se ha trabajado con constancia, cumpliendo los hitos propuestos, y llevando a cabo las acciones relevantes en momentos apropiados.

El resultado es acorde a las horas que finalmente se han invertido en el proyecto. Sí que es verdad, que en caso de haber tenido un poco más de tiempo y algunos recursos que en estos momentos no se han poseído, quizás se hubieran mejorado algunas funcionalidades, pero eso es algo que siempre puede pensar un desarrollador.

Finalmente, la función de este proyecto es la de ayudar a organizar y comunicar a personal sanitario y pacientes con el objetivo de una mayor integración y aportación médica.

9.- Trabajo futuro

Este proyecto, aunque se ha llevado a cabo en su totalidad, actualmente desde la posición en la que me encuentro, puedo observar algunas futuras mejoras que se podrían llevar a cabo. Ha sido un proyecto duro en el que he trabajo el tiempo suficiente como para ir viendo la evolución que podría tener a partir de este momento.

9.1.- Diseño de la interfaz

El diseño de la interfaz es una parte muy importante en la aplicación. La base de un sistema intuitivo y fácil de manejar se basa en la interfaz gráfica, sobre todo para personas, que como en este caso, no tienen por qué ser entendidas en tecnología.

Considero que esta aplicación es bastante intuitiva, no obstante, los colores utilizados quizás sean mejorables. Por una parte, porque no se ha tenido en cuenta mayormente el color que la marca representa para realizar los iconos. Esto es algo mejorable sin duda.

9.2.- Perfil Paciente

Los usuarios identificados como pacientes, únicamente guardan su nombre y apellidos como información relevante. Si un profesional quiere buscar el perfil de un paciente, solamente tendrá disponible esa información. Por ello, como complemento a esto, me parece necesario incluir el centro de salud que tiene adjudicado el paciente de forma usual. Esto ayudará a los profesionales a identificar mejor al paciente en caso de querer contactar con él.

9.3.- Chat de un Usuario Profesional

Actualmente cuando un usuario profesional habla con otro usuario profesional, solamente pueden intercambiar mensajes escritos entre ellos. Por ello, una de las cosas que se pueden incluir en un futuro es:

1. **Compartir informes:** Cuando un paciente se hace una prueba en un centro sanitario, muchas veces los resultados no son transmitidos correctamente a los demás profesionales y acaban repitiendo pruebas innecesarias, cansando innecesariamente al paciente. Por lo tanto, poder compartir entre profesionales, informes de pacientes para evitarles así mayores incomodidades, sería de gran ayuda. Además, el centro de salud se ahorraría mucho tiempo. De todas formas, la difusión de estos informes quedaría registrada en un historial, ya que no se puede compartir informes de pacientes si no están vinculados directamente con los profesionales en cuestión.

9.4.- Notificaciones de mensajes

En el momento en el que el usuario recibe un mensaje, en el sistema actual no se notifica de ninguna forma emergente, a no ser que el usuario esté actualmente en la pantalla del chat del usuario que le ha enviado el mensaje. Por lo que una forma de mejorar este aspecto de información momentánea al usuario, es que se le notifique los mensajes que le llegan al momento aunque no esté en la pantalla del chat correspondiente.

Por ello, aplicar un patrón observable para que en el momento en el que le llegue un nuevo mensaje, lo notifique en pantalla.

10.- Reflexión personal

Este proyecto es, sin duda, el más importante realizado por mi hasta la fecha. Ha sido una experiencia en la que hay que pararse a pensar detenidamente, a mi parecer.

El Trabajo de Fin de Grado es el más significativo de la carrera, es con el que tienes que dar carpetazo a una fase de tu vida y has de hacerlo de una forma que represente todo el esfuerzo dedicado en los 4 años de carrera. Durante toda esta trayectoria me he esforzado inmensamente en sacar esta carrera adelante, ya que es a lo que me quería dedicar desde tercero de ESO. Al principio de esta etapa estuve a punto de dejar el grado. El primer cuatrimestre del primer año, todo se me hacía cuesta arriba, me esforzaba por aprobar los exámenes parciales y no lo conseguía, no veía salida a esa situación porque no había contemplado la posibilidad de verme suspendiendo, ya que había sido una persona que siempre sacaba buenas notas. Al tiempo descubrí que todo había sido causado por una asignatura en particular, programación. Yo nunca hasta la fecha había cursado una asignatura parecida, por lo que no acababa de entenderla por más que me esforzase y pensaba que la carrera no me gustaba. Pero decidí seguir adelante con las demás asignaturas y apuntarme a clases particulares para que alguien me explicara los entresijos de ésta. Finalmente, hoy es el día en el que puedo decir que es la asignatura o la parte de la informática que más me gusta. Por ello decidí basar una gran parte del proyecto de fin de carrera en este ámbito, combinado con bases de datos.

Este trabajo ha requerido mucho esfuerzo y me ha dado la oportunidad de darme cuenta cuán es la magnitud que puede tener un proyecto real. El tiempo dedicado en estos 4 meses al proyecto ha sido, como suelo decir, una auténtica montaña rusa. Ha habido días en los que he estado en lo más alto, porque veía que el tiempo dedicado ese día había cundido y había conseguido los objetivos previstos, pero muchos otros días la montaña rusa bajaba, y me encontraba en la más onda frustración, ya que el tiempo dedicado no había servido para nada.

La dedicación y tiempo empeñado ha tenido que estar muy estructurado y lo he seguido muy estrictamente. Durante la ejecución de este, mi situación ha sido un poco peculiar, ya que me he encontrado realizando prácticas voluntarias vinculadas con este grado en una empresa durante las mañanas, algunas tardes dando clases particulares a un estudiante de 4 de la ESO y trabajando en un bar los fines de semana. Por ello la planificación previa al proyecto me ha ayudado a realizarlo con constancia, y ser consciente de que no hacerlo así tendría como consecuencia no poder entregar el proyecto a tiempo.

Aun así, de toda experiencia se aprende y esta no iba a ser menos. Por ello, presento este trabajo con un buen sabor de boca. Me alegra haber aprendido cómo gestionar y organizar mi tiempo, además de saber amoldarme y sacar el máximo partido a situaciones no tan gratas como las explicadas anteriormente.

Finalmente, explicar de alguna forma el por qué he decidido relacionar mi Trabajo de Fin de Grado con el ámbito de la medicina. En mi familia hay muchas personas que se dedican actualmente, a trabajar en hospitales, residencias y ambulatorios como profesionales sanitarios. Ayudar a sanar a una persona que necesita atención sanitaria en un momento dado, es algo que admiro profundamente. Por ello, he querido ayudar a estos profesionales, desde mi situación actual, a poder llevar ciertas tareas de un modo más sencillo y para realizarlo, he intentado verter los conocimientos adquiridos durante toda la carrera en este proyecto. También me gustaría destacar que los próximos estudios que tengo pensados realizar, por la grata experiencia en este proyecto y por las inquietudes explicadas, es un máster en ingeniería biomédica.

Glosario

- **SQL**: "SQL (Structured Query Language) es un lenguaje de programación estándar e interactivo para la obtención de información desde una base de datos y para actualizarla." **(10)**

- **Plugin:** "Un plugin es aquella aplicación que, en un programa informático, añade una funcionalidad adicional o una nueva característica al software. **(11)**

- **Diagrama Gantt:** "El diagrama de Gantt es una herramienta para planificar y programar tareas a lo largo de un período determinado". **(12)**

- **Scrum:** "Se trata de planificar proyectos en pequeños bloques o Sprints, e ir revisando y mejorando el anterior".**(13)**

- **Sprint**: "El corazón de Scrum es un Sprint, es un intervalo prefijado durante el cual se crea un incremento de producto".**(14)**

Bibliografía

(1) ElEconomista.es, (2018, 20 septiembre). España tiene la sanidad más eficiente de Europa y la tercera del mundo. Recuperado 19 julio, 2019, de https://www.eleconomista.es/economia/noticias/9400270/09/18/Espana-tiene-la-sanidad-mas-eficiente-de-Europa-y-la-tercera-del-mundo-.html

(2) Seguridad Social: Historia de la Seguridad Social. (s.f.). Recuperado 17 julio, 2019, de http://www.seg-social.es/wps/portal/wss/internet/Conocenos/HistoriaSeguridadSocial

(3) Christopher Guindon (s.f.). Eclipse software repository | The Eclipse Foundation. Recuperado 19 julio, 2019, de https://download.eclipse.org/windowbuilder/latest/

(4) WhatsApp.com. (2019). *WhatsApp*. [online] Available at: https://www.whatsapp.com [Accessed 19 Jul. 2019].

(5) Telegram Web. Recuperado 19 julio, 2019, de https://web.telegram.org/

(6) LinkedIn. Recuperado 19 julio, 2019, de https://es.linkedin.com/

(7) Google Calendar (s.f.). Google Agenda. Recuperado 19 julio, 2019, de https://calendar.google.com/

(8) Pixabay - Más de 1 millón de imágenes gratis para descargar. Recuperado 21 julio, 2019, de https://pixabay.com/es/

(9) Maven Repository: com.toedter » jcalendar » 1.4. Recuperado 19 julio, 2019, de https://mvnrepository.com/artifact/com.toedter/jcalendar/1.4

(10) ¿Qué es SQL o lenguaje de consultas estructuradas? - Definición en WhatIs.com (s.f.). Recuperado 19 julio, 2019, de https://searchdatacenter.techtarget.com/es/definicion/SQL-o-lenguaje-de-consultas-estructuradas

(11) Definición de plugin — Definicion.de (s.f.). Definición de plugin — Definicion.de. Recuperado 19 julio, 2019, de https://definicion.de/plugin/

(12) ¿Qué es un diagrama de Gantt y para qué sirve? | OBS Business School. Recuperado 19 julio, 2019, de https://www.obs-edu.com/es/blog-project-management/diagramas-de-gantt/que-es-un-diagrama-de-gantt-y-para-que-sirve

(13) (2017, 19 mayo). Metodología SCRUM: ¿qué es y cómo aplicarlo en tu proyecto? Recuperado 19 julio, 2019, de https://www.sinnaps.com/blog-gestion-proyectos/metodologia-scrum

(14) Las 5 etapas en los “Sprints” de un desarrollo Scrum | OBS Business School. Recuperado 19 julio, 2019, de https://www.obs-edu.com/es/blog-investigacion/project-management/las-5-etapas-en-los-sprints-de-un-desarrollo-scrum