UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de:

Ingenieros de Sistemas

TEMA:

ANÁLISIS, DISEÑO Y DESARROLLO DE UN SISTEMA DE GESTIÓN DE CAMAS E INDICADORES DE CALIDAD PARA EL HOSPITAL GENERAL DOCENTE DE CALDERÓN

AUTORES:

JONATHAN JOSÉ ACHIG CARBO

LUIS EDUARDO PADILLA CRIOLLO

TUTOR:

DANIEL GIOVANNY DÍAZ ORTIZ

Quito, agosto del 2020

CESIÓN DE DERECHOS DE AUTOR

Nosotros, JONATHAN JOSÉ ACHIG CARBO, con documento de identificación N°1720115359, y LUIS EDUARDO PADILLA CRIOLLO , con documento de identificación N°1723535520, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación con el tema: "ANÁLISIS, DISEÑO Y DESARROLLO DE UN SISTEMA DE GESTIÓN DE CAMAS E INDICADORES DE CALIDAD PARA EL HOSPITAL GENERAL DOCENTE DE CALDERÓN", mismo que ha sido desarrollado para optar por el título de INGENIEROS DE SISTEMAS en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

JONATHAN JOSÉ
ACHIG CARBO
CI: 1720115359

LUIS EDUARDO
PADILLA CRIOLLO
CI: 1723535520

Quito, agosto del 2020

DECLARATORIA DE COAUTORÍA DEL TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, con el tema: ANÁLISIS, DISEÑO Y DESARROLLO DE UN SISTEMA DE GESTIÓN DE CAMAS E INDICADORES DE CALIDAD PARA EL HOSPITAL GENERAL DOCENTE DE CALDERÓN realizado por JONATHAN JOSÉ ACHIG CARBO y LUIS EDUARDO PADILLA CRIOLLO, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerados como trabajo final de titulación.

Quito, agosto del 2020

DÍAZ ORTIZ DANIEL GIOVANNY

CI: 1716975501

Dedicatoria

Dedico este trabajo a Dios por darme salud y fortaleza de seguir adelante en cada paso que doy, por guiar mi camino y que pese a las adversidades me lleno de sabiduría para continuar.

A mis padres Guillermo y Elizabeth por el amor, paciencia y apoyo incondicional en darme los medios para poder culminar mis estudios, con todo el amor del mundo ya que este logro también es de ellos.

A mi hermano Jordan como muestra de inspiración y perseverancia en lograr todo lo que nos propongamos es con esfuerzo y sacrificio.

A mi novia Raiza fuente de amor incondicional quien lleno de energías y esperanzas mi sueño de culminar mis estudios.

A mi familia en general quienes han estado en los buenos y malos momentos siempre para apoyarme.

Jonathan José Achig Carbo

Dedicatoria

Dedico este trabajo a todas las personas que me apoyaron a seguir adelante en cada momento de mi vida, en especial a mis padres que estuvieron junto a mí desde el primer día impulsándome a seguir adelante, y ayudándome tanto en lo económico como en lo moral para culminar mis estudios.

Luis Eduardo Padilla Criollo

Agradecimientos

Agradecemos a la Universidad Politécnica Salesiana que ha contribuido en nuestra formación profesional y personal, a nuestro tutor de proyecto de titulación el Ingeniero Díaz Ortiz Daniel Giovanny por habernos orientado y motivado para poder culminar nuestro trabajo.

Jonathan José Achig Carbo

Luis Eduardo Padilla Criollo

Índice

Índice de tablas

Índice de figuras

Resumen

El presente proyecto aplica todos los conocimientos adquiridos en la carrera, siendo el principal objetivo la automatización del proceso asignación de cama que en la actualidad se realiza manualmente. Bajo este contexto se desarrolla una aplicación web que sirva como base al Hospital Docente General de Calderón como una herramienta de apoyo para realizar el ingreso y seguimiento de pacientes dentro de sus unidades, con la finalidad de controlar digitalmente la información generada por cada uno de los usuarios de unidades y subunidades; permitiéndoles así contar con una plataforma web en la que se pueda acceder mediante intranet y en cualquier momento, facilitando el registro de información y seguimiento de pacientes dentro del hospital. Por otro lado, el hospital contará con valiosa información que le permitirá tomar decisiones basadas en reportes de calidad generados por la aplicación.

Abstract

This project applies all the knowledge acquired in the college career, being the main objective to automate the process of assigning beds that is currently carried out manually. Under this background, a web application has been developed to the "Calderón General Teaching Hospital" as a support tool for admitting and monitoring patients inside their units, the purpose is to control the information generated by each of units and subunits users; allowing them to have a web platform that can be accessed through the intranet and at any time, this will facilitate the registration and monitoring of the information of patients in the hospital. On the other hand, the hospital will have valuable information that allows it to make decisions based on quality reports generated by the application.

Introducción

Antecedentes

El presente trabajo desarrolla una plataforma tecnológica que permite mantener el control de camas dentro de las unidades médicas, esto debido a que en la actualidad el registro se lleva de manera manual sin tener un control por usuario sobre la información generada en cada área del hospital incrementando los tiempos de atención a los pacientes y creando un descontrol sobre la consolidación de reportería e indicadores que permitan evaluar el desempeño o nivel de servicio en cuanto a la atención hospitalaria y estadía de pacientes.

Como consecuencia de esto se han buscado herramientas dentro del mercado que sirvan como solución a los problemas establecidos, el objetivo es ofrecer la mejor atención y servicio a la unidad médica. Dentro de la validación realizada tenemos las siguientes opciones:

TICBLUE realizó el desarrollo de una plataforma tecnológica para la Unidad de Gestión Centralizada de Camas (UGCC), permitiendo conocer el estado real de la ocupación de camas en toda la unidad hospitalaria, esta plataforma permite la administración de sus recursos mediante la gestión y comunicación de áreas dentro de la unidad médica; basado en el flujo de procesos internos. La plataforma hace énfasis en el monitoreo y niveles de calidad de servicio, permitiendo visualizar el mapa de ocupación de todas las camas, mismas que se pueden encontrar en uso, libres o bloqueadas, incluso permite dar seguimiento a los pacientes con larga estadía (TICBLUE, s.f.).

La UGCC tiene como objetivo la mejora del servicio, mediante el control de registros hospitalarios con la obtención de registros estadísticos, minimiza los tiempos de gestión para personal interno, mejora la gestión clínica para disminuir la compra de camas, fomentando el trabajo integrado de la red y mejorando la coordinación entre los establecimientos de alta y baja complejidad en la derivación de pacientes (TICBLUE, s.f.).

El Software SIGICAM está basado en inteligencia artificial y algoritmos que contribuyen al proceso de asignación de camas, traslado o derivación de pacientes, utilizando técnicas de investigación de operaciones para detectar restricciones y variables críticas que incidan directamente en la gestión con el objetivo de transparentar la información, evidenciando así el tiempo de estadía de un paciente y también tener un listado de camas disponibles para pacientes que requieran atención. Esta herramienta permite integrar toda la información y poder mostrarla, generando una gran base de pacientes recolectada para generar reportería necesaria, esto gracias a la inteligencia artificial utilizada para tomar decisiones sobre recursos hospitalarios (UNIVERSIDAD DE VALPARAÍSO, 2019).

Con estos antecedentes, se determina la necesidad de desarrollar un Sistema de Información (SI) que ayude en la automatización del proceso de asignación de camas y que se ajuste a la infraestructura tecnológica en hardware y software con la que cuenta la institución.

Justificación del tema

El desarrollo de nuevas tecnologías y de la inclusión de estas para el crecimiento del Hospital General Docente de Calderón exige a sus autoridades implementar Sistemas de Información, automatizando sus procesos internos, con el fin de obtener información precisa y de calidad para la posterior toma de decisiones acerca del servicio y atención en el área de hospitalización, mismas que a corto plazo serán vitales para mantener indicadores de calidad para sus pacientes.

Por lo cual se ve la necesidad de contar con un sistema para la Gestión de Camas con el que se automatizará el registro de ingreso, salida y reingreso de pacientes. Mejorando de esta forma el flujo de información, ya que se registrará todas las atenciones recibidas por doctores y paso por las diferentes áreas del hospital durante su estadía, logrando así determinar el flujo de atención recibida.

Con esta información recopilada durante la atención de pacientes se podrá obtener indicadores de calidad en el servicio médico como disponibilidad de camas, rotación de camas, ingresos, reingresos todo en relación con la atención recibida en el Hospital. Estos indicadores servirán para una posterior toma de decisiones y así poder cumplir con los estándares de calidad dentro del Hospital.

La aplicación será desarrollada en herramientas libres con licencia GNU con el fin de cumplir con el decreto presidencial No. 1425 Art.- 4, utilizando herramientas ya integradas dentro del Hospital con licencias GNU tanto en motor de base de datos cómo en lenguajes de programación. Con el empleo de estas herramientas libres no se limita la implementación del sistema para la Gestión de Camas, debido a que no es necesaria la compra de licencias.

Objetivos

Objetivo general

Analizar, diseñar y desarrollar un Sistema de Gestión de Camas e indicadores de calidad para el Hospital General Docente de Calderón.

Objetivos específicos

Analizar la información recopilada sobre el proceso de gestión de camas dentro del hospital.

Diseñar una interfaz gráfica acorde con el sistema ya implantado dentro del hospital que sea amigable con el usuario e intuitivo para quien lo usa.

Desarrollar el sistema de Gestión de Camas dividido adecuadamente en módulos que permitan la automatización del proceso de asignación de camas.

Desarrollar indicadores de calidad respecto a los días de estada, giro de cama y el reingreso de pacientes.

Realizar pruebas de diseño y funcionalidad a fin de verificar eficientemente el software diseñado para que se acople de la mejor manera a los procesos del hospital.

Metodologías de desarrollo

El desarrollo de software implica llevar acabo un sinnúmero de actividades, que no necesariamente tienen que ver con la creación de código puro para el software, entre las que se destaca la correcta elección de una metodología de desarrollo que imponga una estructura y un orden estandarizado para todo el proceso de desarrollo. Esta metodología no debe ser elegida indistintamente del proyecto de software, más bien cada metodología es aplicable entre varias características por la envergadura del proyecto y por la destreza y flexibilidad de los integrantes del proyecto de desarrollo, es por esto por lo que una correcta elección de la metodología dictará un mejor camino hacia la culminación exitosa del proyecto.

Hay que tomar en cuenta que la utilización de una metodología de desarrollo es necesaria e indispensable en los medianos y grandes proyectos, así se asegura la organización de las actividades y la documentación del proyecto estará gestionada de mejor manera.

En este punto la elección de una metodología tradicional no es aplicable por el tiempo y la flexibilidad requerida para el desarrollo del proyecto, por este motivo seleccionar una metodología de desarrollo ágil es lo adecuado por las características que estas presentan, adaptables adecuadamente al proyecto. En la Tabla 1 encontramos las características más relevantes de tres metodologías de desarrollo ágil, con el objetivo de comparar y seleccionar la más adecuada.

Tabla 1 Metodologías Ágiles

CARACTERÍSTICAS	SCRUM	PROGRAMACIÓN EXTREMA (XP)	KANBAN
Indicada para proyectos donde la comunicación con el cliente es constante		X	
Enfocado para proyectos de mediana y gran envergadura	X		X
Indicado para proyectos pequeños y medianos		X	
Integración de código constante		X	
Entregas con funcionalidades completas	X		
Programación en parejas		X	
Utilización constante de historias de usuario.		X	
Pruebas Unitarias establecidas por el cliente al final del proyecto.		X	
Pruebas en cada fase o ciclo del proyecto	X		X
Contratos de trabajo un tanto flexibles.	X	X	X
Refactorización de código		X	
Estándares de programación Indispensables.	X	X	
Cliente in-situ		X	

Nota: esta tabla contiene las diferencias entre tres metodologías de desarrollo ágil.

Para el éxito del proyecto es indispensable la aplicación conjunta y equilibrada de todas las características de cada metodología, además de tener una capacitación constante del equipo de trabajo, de esta manera se asegura de alguna forma llegar a la conclusión exitosa del proyecto.

Cabe recalcar que antes de realizar la elección de una metodología adecuada que se adapte al proyecto es indispensable la realización de un análisis previo de las características del proyecto, de esta manera se asegura que la elección de la metodología se realizó adecuadamente. En la

Figura 1 se describe de manera simplificada la metodología XP y sus características, así como su estrecha relación y cooperación conjunta.

Características metodología XP

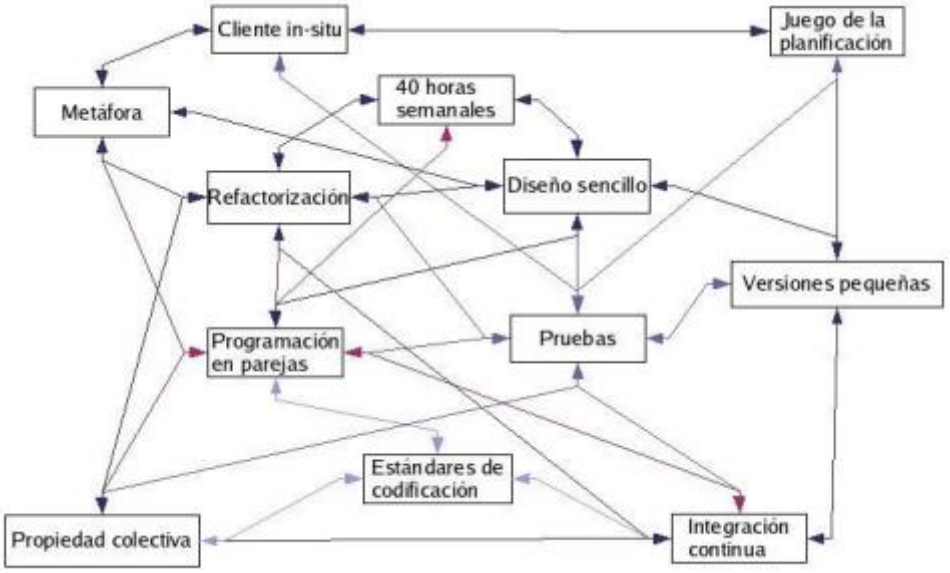

Figura 1 Metodología XP

Fuente: (Torres & López, 2013)

Mediante la Figura 1 se tiene una idea más dinámica de la importancia de aplicar todas las características de cada metodología, en este caso de XP, por el hecho de que las características se apoyan entre sí para lograr un mejor resultado. Es así como la aplicación conjunta y equilibrada es importante. Un ejemplo es la importancia que existe entre el equipo de trabajo y la aplicación de estándares de codificación y estas en la integración continua de código, la importancia reside en que, para la correcta integración de código, que se presenta de manera constante en la metodología XP, los programadores (equipo de trabajo) necesariamente deben aplicar los estándares de programación que facilitan la tarea de integración. De esta manera se relacionan todas las características.

Por las características del proyecto y del equipo de trabajo, se concluye que la elección de la metodología ágil XP es la más adecuada por cumplir ciertos parámetros necesarios en el

proyecto. Cabe recalcar que de nada sirve realizar la elección adecuada de una metodología si no se van a aplicar las “reglas” que esta conlleva, por lo que es fundamental tratar de seguir con el rigor del caso los parámetros establecidos en esta metodología.

Capítulo 1

Marco Teórico

1.1. Hospital General Docente de Calderón

El crecimiento económico del país ha derivado en la creación de diversos proyectos entre estos están los centros de salud y hospitales. Estos últimos sin embargo al tener una envergadura de proporciones mayores tienen necesidades más amplias que los centros de salud, es así como ha transcurrido cerca de cuatro años desde que el Hospital General Docente de Calderón abrió sus puertas a la comunidad del norte de Quito el 15 de Julio del 2015, Consulta Externa fue la primera área en atender y hoy en día cuenta con áreas como: Emergencia, Laboratorio, Imagen y Hospitalización.

Actualmente el Hospital tiene distribuidas 157 camas en las áreas de: medicina interna pediatría, gineco obstetricia y cirugía, fuera de las 48 camillas dispuestas para emergencia. En el transcurso del crecimiento de los requerimientos del Hospital se tiene la necesidad de implementar una adecuada estrategia de automatización del negocio para que las diferentes áreas del hospital estén debidamente coordinadas y el servicio brindado sea medible por indicadores de gestión (HOSPITAL GENERAL DOCENTE DE CALDERÓN, 2018).

Actualmente la información en las diferentes unidades del hospital se registra en hojas de cálculo, las mismas que se comparten en red para que así la información sea actualizada de forma manual en cada uno de los turnos. Si bien se ha dado una solución mediante recursos en red, existen grandes inconvenientes con el proceso actual.

1.1.1. Organigrama

Permite describir el modelo organizacional del núcleo de operaciones de manera matricial, debido a que de esta forma se reúne las habilidades de cada unidad especializada y de cómo

interaccionan para resolver un problema dentro del hospital, esto es representado en la Figura 2 que ayuda y facilita a entender los lineamos estratégicos implementados.

Estructura del HGDC basada en el modelo matricial

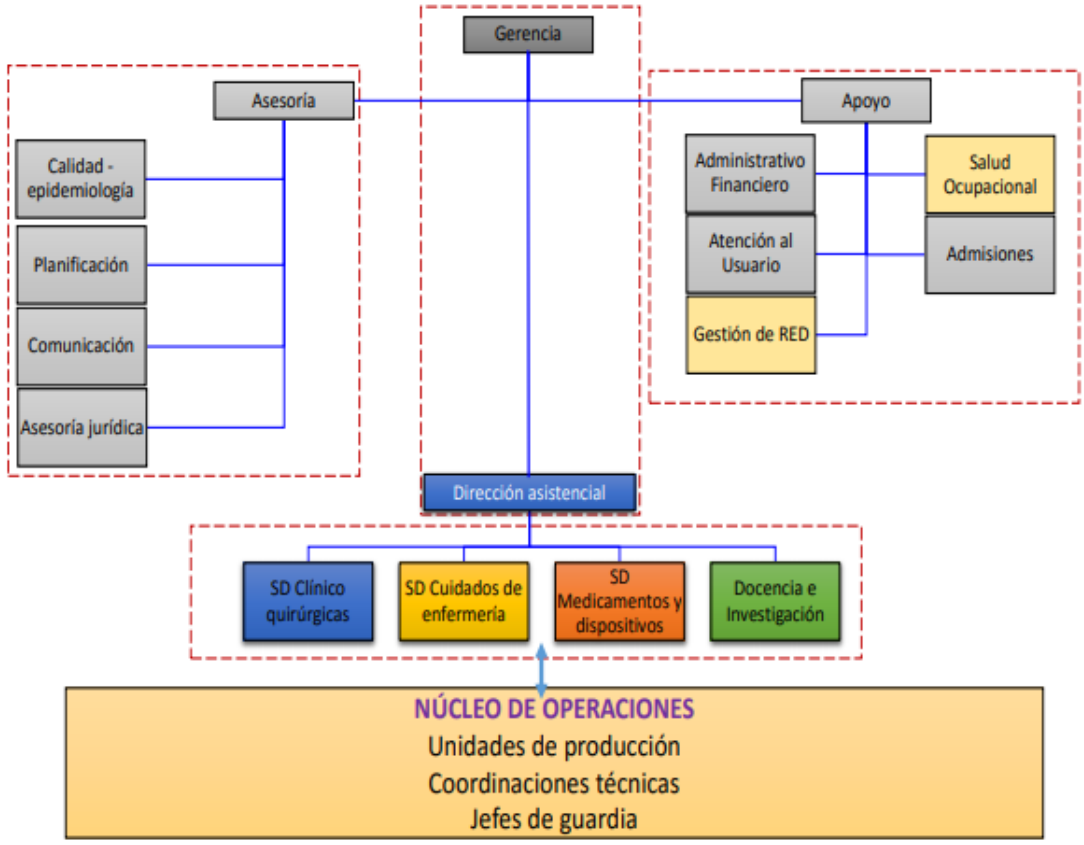

Figura 2 Estructura del HGDC

Fuente: (HOSPITAL GENERAL DOCENTE DE CALDERÓN, 2018)

El área de Procedimientos clínicos – Quirúrgicos menores que es parte de Procesos Complementarios del Hospital General Docente Calderón fue designada para la implementación en modo prueba del Sistema de Camas (SISCAM), por cumplir con las características del hospital, pero en un entorno un poco más controlado y de menor dimensión, en comparación a todo el hospital.

1.1.2. Procesos ingreso paciente en HGDC

Para el ingreso de pacientes en del HGDC se establece los actores, los procesos y su relacionamiento, representado en la Figura 3 determinando así el flujo a seguir con los diferentes posibles resultados dependiendo del camino que el paciente siguió dentro del hospital.

Ingreso paciente

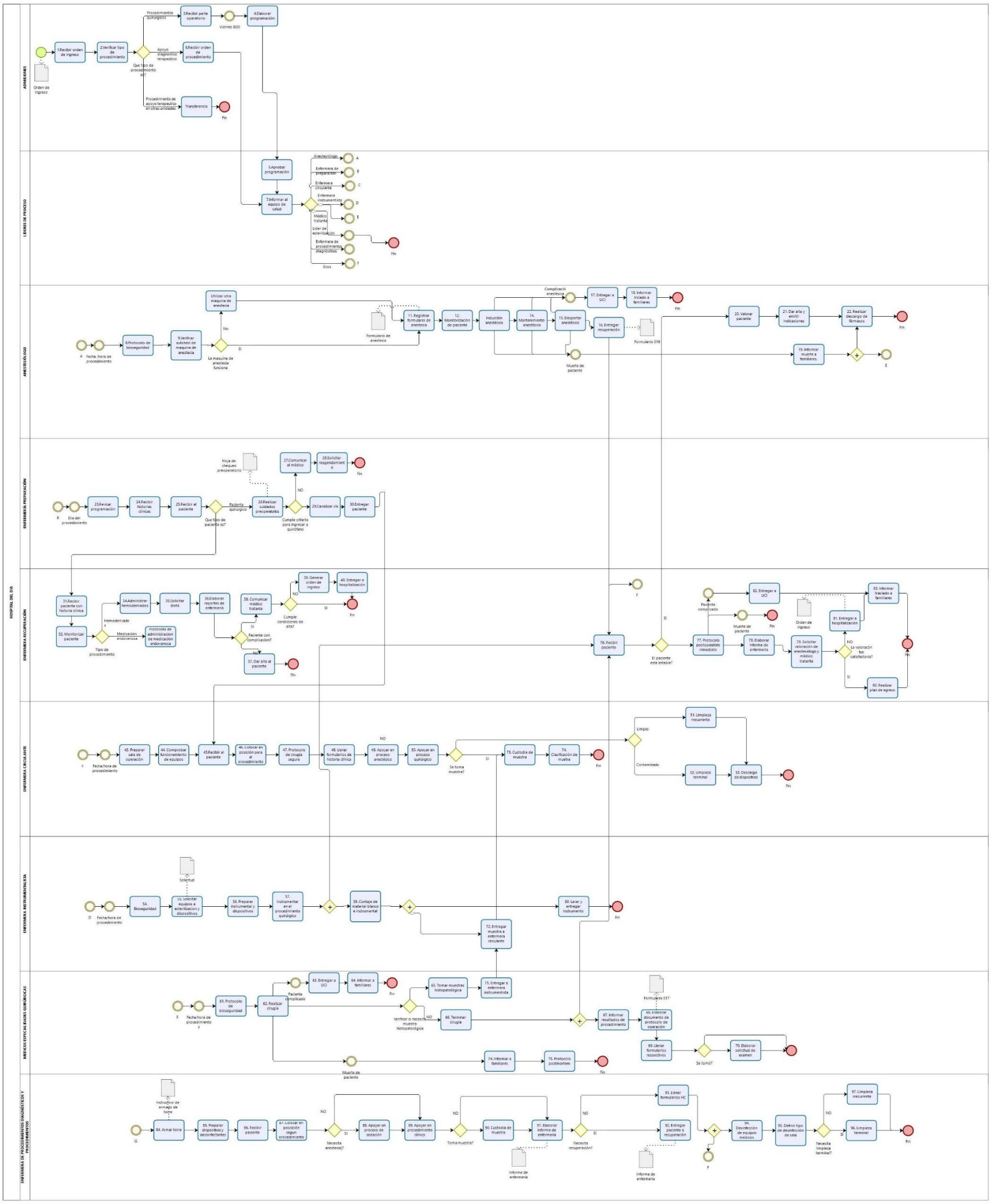

Figura 3 Flujo ingreso de paciente HGDC

Elaborado por: (HOSPITAL GENERAL DOCENTE DE CALDERÓN, 2018)

1.1.3. Proceso asignación de cama HGDC

Para asignación de camas en del HGDC se considera la procedencia del paciente y si cuenta con requisitos mínimos para ser atendido, representado en la Figura 4 debido a que los recursos a usar para cada procedimiento son diferentes, además de tener trazabilidad de atenciones en los pacientes.

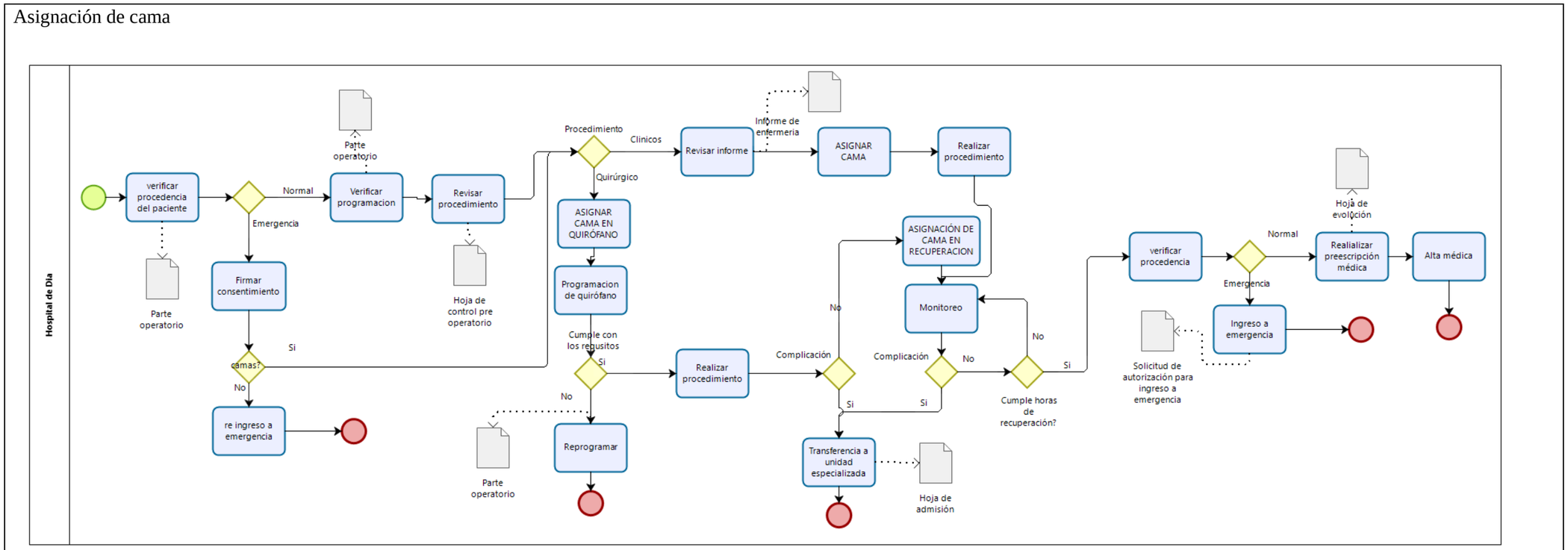

Figura 4 Flujo asignación de cama

Elaborado por: Jonathan Achig y Luis Padilla

1.2. Modelo Vista Controlador (MVC)

Es un patrón de diseño que divide una aplicación de software en tres módulos, como se lo puede visualizar en la Figura 5, estos módulos están bien definidos e identificados en base a su funcionalidad. Los elementos son:

- El Modelo: es un conjunto de clases, mismos que representa la información del mundo real o lógica del negocio que el sistema tiene que procesar, para nuestro ejemplo: el ingreso, salida y egreso de pacientes, giro de camas, días de estada de pacientes, etc., sin tomar en cuenta la forma en la que está la data, los mecanismos usados para procesar la información, es decir, independiente de otra entidad dentro de la aplicación.

- Modelo del Dominio: es un conjunto de clases modeladas para analizar el o los problemas que se plantea resolver. El modelo del dominio no debe guardar relación externa o diferente a la que ya contiene.

- Modelo de la Aplicación: conjunto de clases que tienen relación con el modelo del dominio, teniendo conocimiento de las vistas ya que implementan herramientas necesarias con el fin de anunciar sobre cambios que pueden a ver sido ejecutados en el modelo del dominio. Este modelo también es conocido como coordinador de la aplicación.

- Las Vistas: es un grupo de clases que se encarga de mostrar al usuario la información contenida en el modelo. Un modelo puede estar asociado con una o varias vistas relacionadas con el mismo modelo.

- El Controlador: es el objeto que administra el flujo del control de mensajes dentro de las capas de la aplicación, por ejemplo: datos ingresados por el usuario u opciones seleccionadas en el menú de la aplicación. Partiendo de los mensajes generados, el controlador es quien se encarga de abrir, cerrar vistas y modificar el modelo. El

controlador es quien tiene el acceso a las vistas y al modelo, pero las vistas y el modelo no conocen de la presencia del controlador (Bascón Pantoja, 2004).

Relación entre módulos MVC

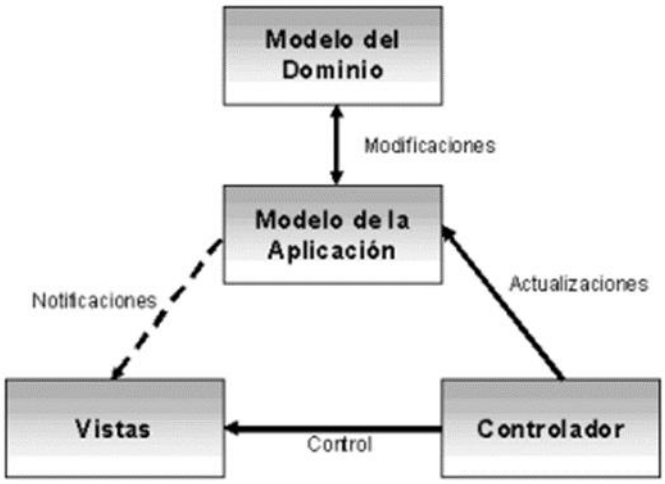

Figura 5 Modelo Vista Controlador

Fuente: (Bascón Pantoja, 2004)

1.3. Programación Orientada a Objetos (POO)

Es un paradigma de programación creado para lograr resultados más eficientes en los procesos internos de los programas informáticos. Este paradigma se basa en los siguientes aspectos de programación: herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento. Además, de usar objetos como entidades que encapsulan datos y funciones en la obtención de resultados.

Los elementos principales dentro de la programación funcional son las funciones, dichas funciones generan datos que solo son utilizados por las mismas funciones luego de ser pasados entre sí, se debe mencionar que los valores generados no son de ninguna utilidad fuera de las funciones.

Es importante considerar este paradigma al momento de desarrollar el sistema de control de camas ya que esto aporta a la construcción de un sistema con calidad con fuerza en extensión, mantenimiento y reutilización de código (Greiner, Acosta, Dapozo, & Estayno, 2012).

1.4. Base de Datos Relacional

Es una base de datos que permite representar datos en tablas, relacionándolas entre sí, proporcionando almacenamiento y acceso a varios puntos de datos. Los registros de las filas cuentan con un ID único o clave primaria, mientras que los datos de las columnas tienen atributos lo cual permite establecer relación entre tablas. Por tal motivo toma el nombre de modelo relacional (Suarez, 2008).

La relación de atributos entre dos tablas se lo realiza mediante un identificador único, permitiendo interconectar datos tal como se representa en la Figura 6.

Diagrama BBDD

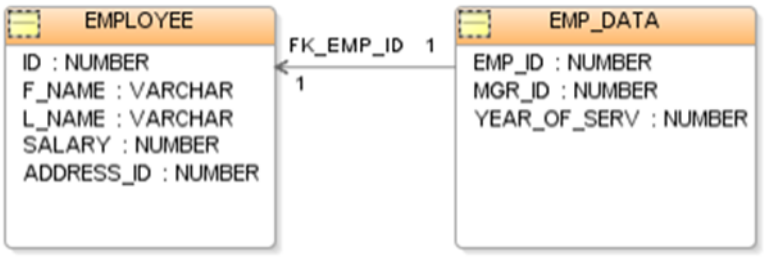

Figura 6 Modelo relacional

Fuente: (ACADEMIC, s.f.)

1.5. Interfaz de Programación de Aplicaciones con Transferencia de Estados Representacional (API REST)

Una API es un mecanismo de tipo backend para realizar la conexión entre dos aplicaciones de software, para de esta manera asegurar un intercambio de datos entre si aplicando un formato estándar. Ahora bien, REST es un conjunto de condicionamientos o reglas que permitan la

comunicación eficiente mediante protocolo HTTP utilizando JSON y XML como lenguaje de intercambio de información.

Esta tecnología permite crear aplicaciones web eficientes permitiendo el crecimiento horizontal de una manera correcta, siendo capaz de contestar las llamadas realizadas de cualquier URL sin importar la estructura del frontend. Por lo que desarrollar una API en el backend aplicando REST permite su utilización en cualquier dispositivo que utilice sistemas operativos diferentes como son: IOS, Android o un navegador web.

Con el paso de los años API REST ha ido remplazando la tecnología SOAP, que cuenta con características similares necesitando de una mayor complejidad, es así como empresas y aplicaciones reconocidas mundialmente como son Facebook o Google han optado por aplicarla a sus plataformas. Entre las principales ventajas de esta tecnología es necesario mencionar las siguientes:

- Cliente servidor, el cliente ni el servidor necesitan recordar estados previos de su comunicación, lo que permite de alguna manera que ninguno de los dos se preocupe como se manejan sus datos.
- Operaciones más importantes en esta tecnología: POST, GET, PUT y DELETE.
- Objetos REST manejados con URLs: El URL se utiliza como identificador único de cada recurso y facilita la interacción.
- Interfaz uniforme, las cuatro operaciones identificadas con una URL permiten la interacción uniforme entre cliente y servidor.
- Sistema de capas, el servidor puede utilizar un sistema de capas (jerárquica) entre sus componentes que realicen una determinada funcionalidad, permitiendo mejorar rendimiento, escalabilidad y seguridad (BBVAOPEN4U, 2016).

1.6. Indicadores de Calidad

Los indicadores de calidad permiten determinar el valor obtenido después de cuantificar las atenciones médicas y uso de camas en el Hospital. Entre los indicadores se encuentran:

- Trazabilidad de paciente: indicador con el cuál se cuantifica las atenciones médicas recibidas por un paciente.
- Ingreso: indicador que permite cuantificar los procedimientos médicos realizados a un paciente en el hospital.
- Reingreso: indicador que permite cuantificar los ingresos a un mismo procedimiento médico que tuvo un paciente.
- Giro de cama: indicador que permite cuantificar las veces que se ocupó una cama después de un procedimiento médico.

Mediante estos indicadores el hospital puede tomar decisiones en cuanto a sus requerimientos en número camas o en determinar si los pacientes están recibiendo un adecuado tratamiento (HOSPITAL GENERAL DOCENTE DE CALDERÓN, 2018).

1.7. Herramientas de desarrollo

Las herramientas por usar son de código abierto, que permiten una mejor colaboración con el equipo de desarrollo permitiendo crear, depurar y gestionar los avances y entregas del software.

1.7.1. Conector a Base de Datos (JDBC)

Es un API (Interfaz de programación de aplicaciones), que se compone de objetos y métodos permitiendo desarrollar interfaces con el objetivo de establecer comunicación con la base de datos, mediante el uso de aplicaciones JAVA.

Independientemente del sistema operativo (SO) donde este implementado el sistema debe permitir la conexión a BBDD mediante JAVA, interfaces JAVA y métodos de conexión hacia un modelo especifico de BBDD.

Esta API permite que cualquier programa escrito en lenguaje java ejecute instrucciones SQL, JDBC hace posible que una solo aplicación ejecute una base de datos en diferentes plataformas, enviando instrucciones directamente a la BBDD (Vanegas, 2005). Entre sus características tenemos:

- Orientada a ejecutar comandos SQL.
- Permite que cada comando SQL pase directamente al controlador.
- Es homogéneo al resto de APIs de JAVA.
- Debe mantener lo más sencillo posible los casos de acceso a las bases de datos.
- La utilización de esta herramienta es indispensable en el sistema para asegurar la correcta comunicación con la BBDD y a demás implementar comandos SQL directamente sobre la base.

1.7.2. AngularJS

Es un framework MVC usado para el desarrollo WEB (FRONT END), permitiendo crear aplicaciones de página única (SPA). Al aplicar MVC separamos en capas el diseño y la lógica, con la ventaja de gestionar llamadas, manipular contenido, recuperar datos y permitir la comunicación con el servidor. Logrando que el contenido se actualice sin que la capa lógica sepa lo ocurrido en la capa visual, permite su implementación dentro de diferentes plataformas como son: sitios web, aplicaciones de escritorio y dispositivos móviles (Mafla, 2019). Sus características son:

- Es open source.

- Permite el desarrollo ordenado y sencillo.
- Permite dar un mantenimiento adecuado a futuro.
- Esta apoyado por Google.
- Utiliza el patrón de desarrollo MVC.

La aplicación de angular dentro del sistema se debe a la fácil creación de instrucciones web que permitan la generación de las funcionalidades adecuadas dentro del sistema.

1.7.3. Git

Sistema de control de versiones, herramienta orientada a la gestión del versiona miento de un producto de desarrollo de software enfocado principalmente en los cambios que se generan en el desarrollo estableciendo control sobre los mismos.

Git fue concebido para mantener la eficiencia en la gestión de versiones estableciendo confiabilidad además de eficiencia en el proceso de desarrollo en un equipo de trabajo, sus características principales son:

- Potente en el mantenimiento de versiones.
- Tiene licencia de libre uso.
- Es independiente de repositorios centrales.
- Gestiona historial de versiones.

Esta herramienta nos permite llevar un adecuado historial de desarrollo mediante el registro de versiones que es indispensable en la programación XP para detectar algún error dentro del proceso de desarrollo. En este caso la utilización de Git fue elegida por la familiaridad del equipo de trabajo con esta herramienta (ATLASSIAN, s.f.).

1.7.4. Gradle

Herramienta usada en la automatización para compilar código, tomando como prioridad la flexibilidad y rendimiento. Utiliza Kotlin DSL o Groovy para escribir sus scripts de compilación.

Aparte de JAVA nos permite usar otros lenguajes debido a su flexibilidad representado en la Figura 7 se establece el sistema de gestión de dependencias. Gradle es rápido y personalizable debido a que completa las tareas reutilizando las salidas de previas ejecuciones, procesando entradas que presentan cambios en simultáneo (Álvarez Caules, 2015). Entre sus características se tiene:

- Permite compartir los resultados de la compilación del sistema.
- Se encarda de descargar y administrar las dependencias transitivas.
- Detecta las clases afectadas por los cambios realizados.

Esta herramienta ayuda a llevar una correcta organización del código y detectar afectaciones a clases que integran el sistema.

Gradle

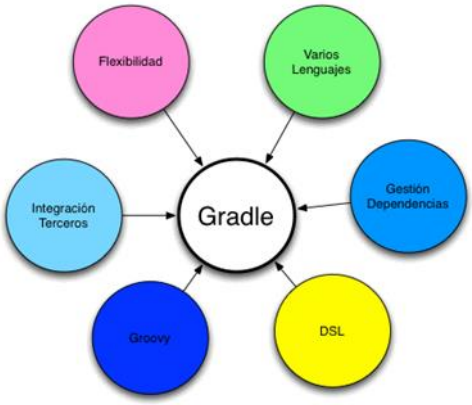

Figura 7 Características de gradle

Fuente: (HOSPITAL GENERAL DOCENTE DE CALDERÓN, 2018)

1.7.5. Objeto de Acceso de Datos (DAO)

Es un objeto que suministra una interfaz común, abstracta y gráfica entre la aplicación y los dispositivos de almacenamiento de datos. DAO actúa como intermediario entre las aplicaciones y la BBDD asociada ocultando todos los detalles correspondientes al almacenamiento.

Esta herramienta en necesaria para el almacenamiento de datos, actuando como intermediaria en la generación de información del sistema. (Blancarte, 2018)

1.7.6. Spring data

Es útil para crear consultas restrictivas sobre las entidades del repositorio. El mecanismo de tiras de los prefijos find…By, read…By, query…By, count…By, y get…By desde el método y comienza a analizar el resto de esta. La cláusula de introducción puede contener otras expresiones, como una Distinct para establecer un indicador distinto en la consulta que se creará. Sin embargo, el primero By actúa como delimitador para indicar el inicio de los criterios reales.

Su principal característica es la simplificación al desarrollador de la persistencia de datos contra distintos repositorios de información, lo cual nos ayuda a simplificar el trabajo de desarrollo (SPRING, s.f.).

1.7.7. Metodología eXtreme Programming (XP)

XP es una metodología enfocada en desarrollo ágil, programación en parejas y en la continua comunicación con el cliente, está enfocado para proyectos con requerimientos cambiantes siendo el cliente parte clave del desarrollo como integrante del equipo. A demás prioriza la optimización de recursos (si se implementa con éxito) y la generación de un buen ambiente de trabajo. Entre sus principales características tenemos:

- Interacción constante entre el equipo de trabajo y el cliente.
- Programación en parejas.

- Se considera una metodología con planificación abierta y flexible.
- Enfocada a respuestas rápidas en los cambios.
- El uso de historias de usuarios es indispensable.
- Adecuada para proyectos con requerimientos imprecisos y cambiantes.
- Integración de código constante.

1.7.7.1. Prácticas en la metodología XP

Para la implementación de XP es indispensable la aplicación de prácticas que aumentaran la probabilidad de éxito del proyecto. A continuación, se describen algunas de las prácticas más relevantes:

- Entregas pequeñas: las entregas que se realizan deben ser constantes por lo que en su mayoría no van a contar con todas las funcionalidades especificadas, sin embargo, son consideradas entregas de valor por parte del cliente ya que si son operativas.
- Refactorización: el objetivo de esta práctica es evitar la duplicación de código, la simplificación de código y flexibilizarlo ante cualquier cambio que se genere.
- Pruebas: se deben programar pruebas unitarias antes de cada integración de código y para cada historia de usuario y ejecutadas constantemente durante la generación del sistema, estas pruebas deben ser planificadas junto con el cliente.
- Programación en parejas: en esta metodología se ha comprobado que la programación en parejas es altamente eficaz a la hora de detectar errores y fallas de programación, así se equilibra la diferencia de conocimientos entre los programadores, se debe tener en cuenta que la revisión constante de código se debe dar por parte de los 2 integrantes del equipo.

- Integración continua: la integración de código generado por cada equipo de trabajo se debe implementar de inmediato una vez que se encuentre lista, por lo que al contar con varias parejas de programadores se puede llegar a integrar código varias veces al día.

- Horas de trabajo: es altamente recomendable no sobrecargar de trabajo al equipo del proyecto, por lo que cuarenta horas semanales se consideran adecuadas. Estas horas de trabajo se determinan de acuerdo con la estimación, previamente realizada, de esfuerzo requerido para cada tarea.

- Cliente en el sitio: el cliente debe participar activamente de principio a fin en el proceso de desarrollo para asegurar una correcta interpretación de los requerimientos y de los cambios que se puedan generar en el proceso, el cliente es quien guía la implementación correcta del modelo de negocio en el sistema. Es muy recomendable que al no contar con la presencia del cliente directo se asigne un representante que esté al tanto de todo el modelo de negocio y que disponga con el tiempo suficiente para el proyecto, y no planificar con el cliente reuniones parciales cada cierto tiempo.

- Estándares de programación: la aplicación de estándares de programación por parte de los programadores es indispensable en el curso del proyecto, en especial al momento de la interpretación de código y la integración de código diseñado por los distintos integrantes del proyecto (Joskowicz, 2008).

1.7.7.2. Fases de la metodología

Para implementar XP se debe contemplar seis fases que permiten aumentar el éxito en la utilización de la metodología, estas fases son las siguientes:

- Fase I Exploración: en esta primera fase el equipo de desarrollo en coordinación con el cliente procede a crear genéricamente historias de usuario, ya que de esta forma se

tendrá una idea de las funcionalidades esperadas por el cliente y el resultado de esto será parte del primer entregable.

Además, en este periodo de tiempo de no más de unas pocas semanas el equipo de trabajo tiene la tarea de explorar e informarse sobre las características de las herramientas a utilizarse durante el desarrollo. Se crean prototipos para armar la arquitectura que ira dando forma al sistema. Cabe recalcar que el tiempo está definido por la familiaridad previa del equipo de trabajo sobre las herramientas utilizadas.

- Fase II Planificación de la Entrega: para esta fase las historias de usuario toman especial importancia en el contexto de que marcarán el tiempo de cada entregable, es así como para determinar la cantidad de tiempo que se empleará en el desarrollo de cada historia de usuario se debe establecer un punto de esfuerzo determinado genéricamente como una semana ideal de desarrollo. Cada historia de usuario no debería superar los 3 puntos de esfuerzo en su creación, de esta manera al sumar los puntos se puede estimar fechas para generar un cronograma de entregables lo más exacto posible.

  Esta no es la única forma de generar un cronograma de entregables, también se puede realizar mediante el alcance del sistema donde es necesario determinar una velocidad de desarrollo para de esta forma establecer cuantas historias de usuario se puede terminar antes de cumplir una fecha seleccionada. Cabe recalcar que para superar esta fase no se debería emplear más allá de unos pocos días.

- Fase III Iteraciones: en esta fase se genera un plan de iteraciones donde en la primera de ellas el cliente debe seleccionar aquellas historias de usuario que considere fundamentales para el sistema y que tienen prioridad sobre las otras. En tanto el equipo de desarrollo tiene la tarea de generar la arquitectura del sistema basándose en las historias de usuario seleccionadas previamente por el cliente, esto no siempre se puede

cumplir ya que al ser el usuario quien escoja las historias no se asegura una selección adecuada de las mismas.

Para la elaboración del plan de iteraciones se debe tomar en cuenta varios elementos correspondientes a la generación de las iteraciones, los cuales pueden ser los siguientes:

- HU no consideradas
- Agilidad del desarrollo del proyecto
- Pruebas de aceptación no favorables
- Tareas no concluidas correctamente en iteraciones preliminares

Las tareas correspondientes deben ser asignadas a un programador responsable siendo este parte de una pareja de desarrolladores.

- Fase IV Producción: en esta fase se considera programar pruebas complementarias a las ya realizadas, además de comprobaciones sobre el rendimiento del sistema previo al traslado al ambiente del cliente, todo esto considerando al mismo tiempo cambios a la versión actual por el hecho de que si el cliente lo requiera se deben agregar nuevas funcionalidades o características al sistema.

  Posiblemente se pueda disminuir el tiempo de cada iteración hasta llegar a una semana. Y conjuntamente se pueden documentar las sugerencias y propuestas para su posterior implementación.

- Fase V Mantenimiento: para esta fase se debe considerar la adición de nuevos integrantes especializados al equipo de desarrollo para generar una mejor revisión sobre el mantenimiento del sistema.

En tanto la primera versión del sistema se encuentra en etapa de producción es necesario mantener funcionando el sistema al igual que seguir generando las iteraciones correspondientes, se debe tomar en cuenta que todo esto probablemente afecte la velocidad con la cual se desarrollaba el sistema y se lo colocaba en producción.

- Fase VI Muerte del Proyecto: suele ocurrir al finalizar el proyecto bajo las siguientes situaciones:
    - Cuando los recursos destinados al proyecto se agotan o suspenden súbitamente.
    - Cuando las características y funcionalidades del sistema no cumplen la expectativa del interesado, es decir el cliente.
    - Cuando la generación de beneficios del sistema no cumple lo esperado.
    - Cuando ya no se tiene más historias de usuario que desarrollar y el cliente ya no agrega más características al sistema, esta última es la esperada para el éxito del sistema, donde además luego se deben considerar satisfacer las necesidades del cliente en medida del rendimiento que debe tener el sistema y la confiabilidad de este (Joskowicz, 2008).

Capítulo 2

Análisis y Diseño

2.1. Requerimientos

Los requerimientos son indispensables para la creación y posterior codificación del sistema, estos requerimientos se dividen en funcionales y no funcionales ayudando a describir la estructura y funcionalidad del sistema.

2.1.1. Requerimientos funcionales

Permite establecer las cualidades que debe tener la aplicación, con el fin de satisfacer los requerimientos del usuario. A continuación, se detalla los siguientes:

- El sistema debe permitir la generación de nuevos usuarios y la asignación de perfiles.
- El sistema debe permitir la generación de nuevas unidades y subunidades por parte de los administradores.
- El sistema debe permitir la generación de nuevos estados de cama y tipos de cama.
- El sistema debe permitir la generación e ingreso de nuevas camas siempre y cuando el lote de camas cuente con el registro de bodega y con todas las especificaciones establecidas.
- El sistema debe permitir la transferencia de paciente hacia otras dependencias del hospital siempre y cuando se cuente con la capacidad requerida para su atención.
- El sistema debe permitir el ingreso de pacientes, siempre y cuando se informe que el paciente cuente con una historia clínica.
- El sistema debe permitir dar de alta fácil y rápidamente a un paciente sin que se tenga que llenar formularios en ese mismo instante.

- El sistema debe permitir la visualización de métricas de calidad en la atención a los pacientes.
- El sistema debe permitir establecer pistas de auditoria.
- El sistema debe permitir la fácil búsqueda de información con respecto a las camas y a los pacientes que las ocupan.
- Se debe permitir la visualización del recorrido realizado dentro de la unidad por parte del paciente.
- Se debe visualizar todos los formularios requeridos en el tratamiento de un paciente, y no necesariamente deben ser llenados en ese instante, se tiene que verificar cada formulario antes de su ingreso.
- Se deben numerar las camas y establecer un mapa de la unidad donde están dispuestas las camas.
- El sistema debe permitir que las camas de quirófano sean visualizadas por separado, con ingresos y egresos de pacientes.

2.1.2. Requerimientos no funcionales

Permite establecer las características generales y las restricciones que debe tener la aplicación, y la presentación de los datos mediante la interfaz del sistema. A continuación, se detalla los siguientes:

- El sistema debe estar diseñado sobre plataformas libres.
- El sistema debe ser web.
- Se deben implementar adecuadas prácticas de desarrollo.
- El sistema debe ser modular.

- El sistema debe contar con interfaces estándar.
- El sistema debe poder ejecutarse en equipos con características medias.
- El sistema debe estar respaldado con los manuales correspondientes a cada módulo.
- La probabilidad de falla no debe ser superior al 1% de todas las transacciones realizadas.
- El sistema debe ser ejecutable sobre plataforma Linux, preferiblemente Ubuntu desde su versión 14.
- Los respectivos permisos de acceso al sistema únicamente podrán ser modificados por el administrador.
- La información enviada a la BBDD debe estar debidamente encriptada.
- El sistema debe desplegar mensajes de éxito o error de ser el caso.
- El sistema debe ser capaz de soportar varios ingresos de datos a la vez.
- El sistema debe ser interactivo.
- El sistema debe tener como mínimo un 99% de disponibilidad.
- El sistema debe ser fácil de usar y no superar las 8 horas para su total aprendizaje.

2.2. Diagramas casos de uso

Ayudan a describir los pasos que tienen que seguir los diferentes usuarios del sistema para la ejecución de procesos. Adicional de describir el rol de cada actor (usuario del sistema), definir los elementos del sistema (operaciones) y determinar su relación.

2.2.1. Creación de usuario

El actor administrador por su perfil puede crear usuarios, teniendo que ingresar datos básicos del usuario a crear (nombre, apellido, nickname, email, cargo), determinar el perfil correspondiente al cargo y así guardar la transacción (Ver Figura 8) dando como resultado la notificación de creación o no creación del usuario.

Descripción de pasos para crear usuario

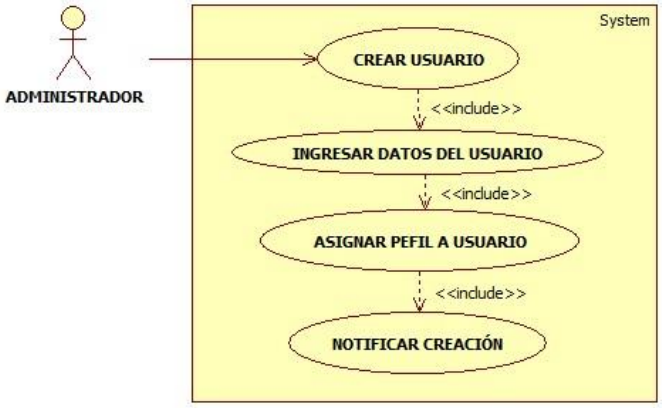

Figura 8 CU crear usuario

Elaborado por: Jonathan Achig y Luis Padilla

2.2.2. Creación de unidad

El actor administrador por su perfil puede crear unidades, teniendo que ingresar datos referenciales de la unidad, para después asignar un responsable en base a los usuarios ya creados para guardar la transacción (Ver Figura 9) dando como resultado la notificación de creación o no creación de la unidad.

Descripción de pasos para crear unidad

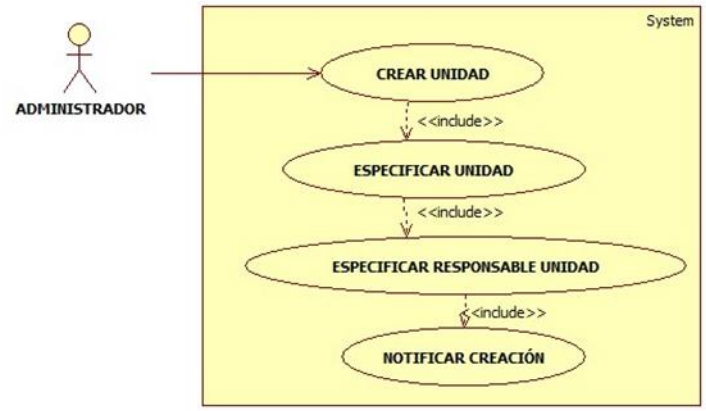

Figura 9 CU crear unidad

Elaborado por: Jonathan Achig y Luis Padilla

2.2.3. Creación de subunidad

El actor administrador por su perfil puede crear subunidades, previo a ejecutar esta acción se tiene que haber creado unidades para que después se ingrese datos referenciales de la nueva subunidad, de igual manera se debe asignar un responsable en base a los usuarios ya creados para guardar la transacción (Ver Figura 10) dando como resultado la notificación de creación o no creación de la subunidad.

Descripción de pasos para crear subunidad

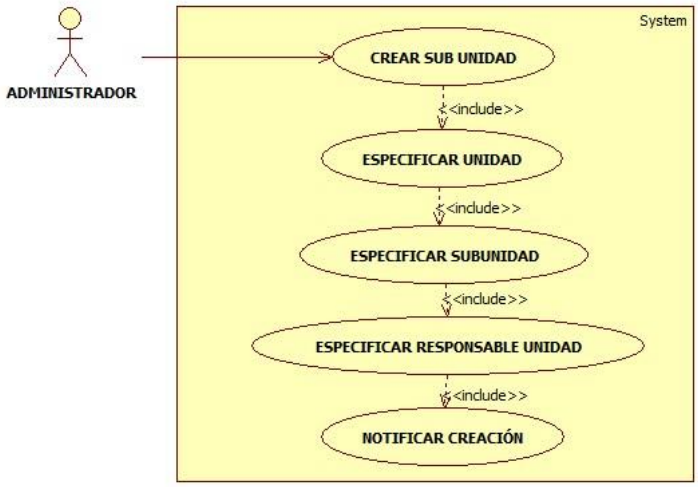

Figura 10 CU crear subunidad

Elaborado por: Jonathan Achig y Luis Padilla

2.2.4. Crear cama

El actor administrador por su perfil puede crear camas, previo a ejecutar esta acción se tiene que haber creado unidades y subunidades para que después se ingrese datos referenciales de la nueva cama, debido a que de igual manera se debe asignar un responsable en base a los usuarios ya creados para guardar la transacción (Ver Figura 11) dando como resultado la notificación de creación o no creación de la subunidad.

Descripción de pasos para crear cama

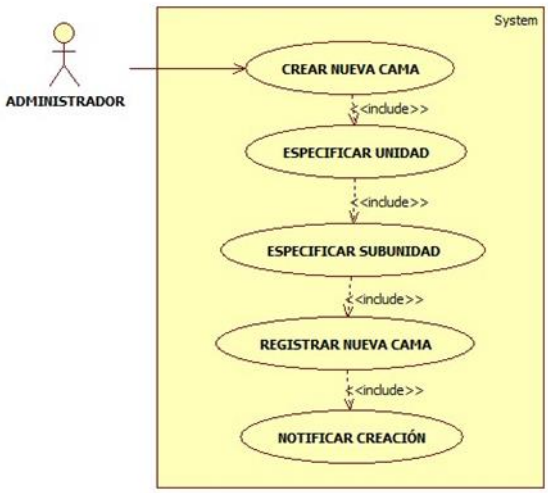

Figura 11 CU crear cama

Elaborado por: Jonathan Achig y Luis Padilla

2.2.5. Crear estado de cama

El actor administrador por su perfil puede crear estados de cama, con el fin de poder determinar el uso de la cama se describe el estado y se procede a guardar (Ver Figura 12) dando como resultado la notificación de creación o no creación del estado de cama.

Descripción de pasos para crear estados de cama

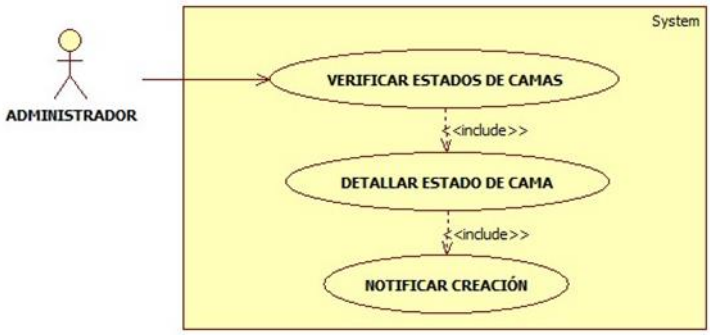

Figura 12 CU crear estado cama

Elaborado por: Jonathan Achig y Luis Padilla

2.2.6. Ingresar paciente

El actor usuario por su perfil puede ingresar pacientes para registrar el nuevo procedimiento a recibir, previo al ingreso valida si tiene atenciones pasadas para tener visibilidad de su historia clínica. Si paciente es nuevo deberá ingresar los datos básicos (nombre, apellidos, CI, nacionalidad, tratamiento) para que pueda ser asignado a una unidad y subunidad según corresponda, como paso final deberá guardar el paciente (Ver Figura 13) dando como resultado la notificación de ingreso o no ingreso del paciente.

Descripción de pasos para ingresar paciente

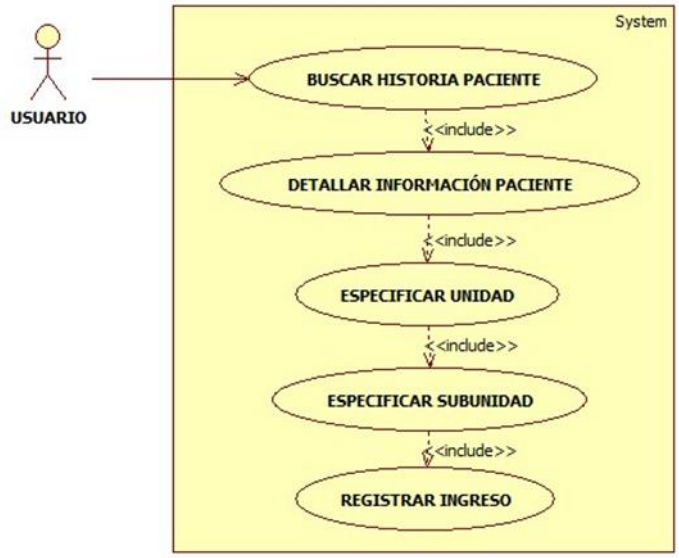

Figura 13 CU ingresar paciente

Elaborado por: Jonathan Achig y Luis Padilla

2.2.7. Obtener informes

El actor usuario por su perfil puede generar informes, previo a elegir el tipo de informe y fechas de inicio – fin de la información (Ver Figura 14) dando como resultado la generación del reporte para su posterior impresión.

Descripción de pasos para obtener informe

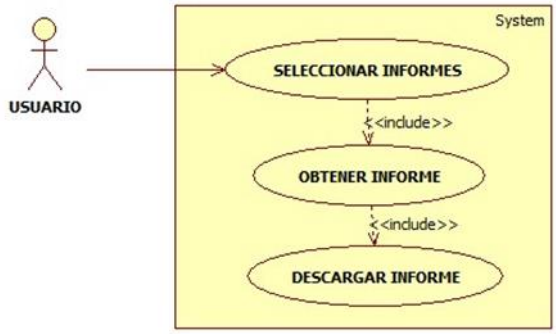

Figura 14 CU obtener informe

Elaborado por: Jonathan Achig y Luis Padilla

2.2.8. Transferir paciente

El actor usuario por su perfil puede transferir pacientes, para esto es importante definir el ingreso del paciente (Ver Figura 13) ya que de esto se tomará la transferencia a la unidad de origen, después de haber registrado la atención en la unidad / subunidad (Ver Figura 15) dando como resultado la transferencia del paciente.

Descripción de pasos para transferir paciente

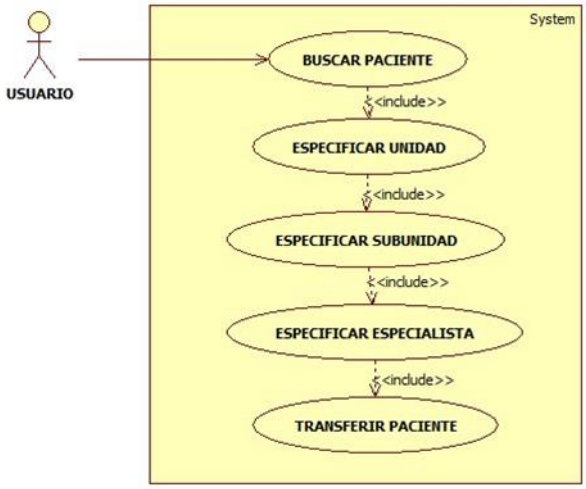

Figura 15 CU transferir paciente

Elaborado por: Jonathan Achig y Luis Padilla

2.3. Historias de usuarios

Describen los requisitos para ejecutar un proceso, especificando los requisitos que cada una necesita para su correcta ejecución y establecer los mensajes a desplegar por el sistema.

2.3.1. Ingresar aplicación

Para el ingreso a la aplicación se debe contar con conexión a intranet e ingresar el usuario y contraseña asignados al usuario, ya que de esto dependerá el flujo de pantallas y mensajes que el sistema desplegará (Ver Tabla 2) con la finalidad que el usuario visualice la pantalla correcta o en su caso sepa que falló y proceda a corregir.

Tabla 2 HU: Ingreso a la aplicación

HISTORIA DE USUARIO	
NÚMERO: 1	PERFIL: ADMINISTRADOR
NOMBRE: INGRESO A LA APLICACIÓN	
PRIORIDAD: ALTA	RIESGO: ALTO
FLUJO NORMAL: 1) Se ingresa a página principal o login desde la intranet 2) Como requisitos es necesario digitar por teclado usuario y contraseña 3) La visualización del menú se desplegará acorde al perfil asignado al usuario	
FLUJO ALTERNATIVO: 1.a) Visualiza un mensaje de error cuando el servicio de la aplicación no esté disponible 2.a) Si se digita mal el usuario o contraseña, se visualizará un mensaje de alerta indicando lo sucedido y solicitará que se vuelve a ingresar las credenciales 2.b) Si las credenciales del usuario fueron ingresadas correctamente y se visualiza el mensaje de usuario inactivo se deberá solicitar al administrador del sistema activar el usuario 3.a) Si las opciones del perfil no corresponden al cargo, se deberá notificar al administrador del sistema la verificación de los permisos	

Nota: tabla en la que se detalla los flujos para ingreso a la aplicación.

2.3.2. Crear usuarios

Para la creación de usuarios la aplicación valida el acceso a dicho módulo por perfil del usuario, de ser el perfil necesario debe ingresar los datos solicitados en el formulario, ya que de esto dependerá el flujo de pantallas y mensajes que el sistema desplegará (Ver Tabla 3) con la finalidad que el usuario identifique los pasos a seguir.

Tabla 3 HU: Creación de usuarios

HISTORIA DE USUARIO	
NÚMERO: 2	PERFIL: ADMINISTRADOR
NOMBRE: CREACIÓN DE USUARIOS	
PRIORIDAD: ALTA	RIESGO: MEDIO
FLUJO NORMAL: 1) Dentro de las opciones del menú se debe seleccionar la opción crear usuario 2) Se debe ingresar la información básica requerida por el sistema 3) Notifica sobre creación del usuario	
FLUJO ALTERNATIVO: 2.a) Si falta información dentro de los parámetros requeridos, solicitará validar los campos resaltados en rojo 2.b) Si el usuario ya se encuentra registrado, el sistema notificará la no creación del usuario	

Nota: tabla en la que se detalla los flujos para crear usuarios.

2.3.3. Crear estado de cama

Para el ingreso de estados de cama la aplicación valida el acceso a dicho módulo por perfil del usuario, de ser el perfil necesario debe ingresar los datos solicitados en el formulario, ya que de esto dependerá el flujo de pantallas y mensajes que el sistema desplegará (Ver Tabla 4) con la finalidad que el usuario identifique los pasos a seguir.

Tabla 4 HU: Creación estados de cama

HISTORIA DE USUARIO	
NÚMERO: 3	PERFIL: ADMINISTRADOR
NOMBRE: CREAR ESTADO DE CAMA	
PRIORIDAD: ALTA	RIESGO: ALTO
FLUJO NORMAL: 1) Dentro de las opciones del menú se debe seleccionar la opción crear estados de cama 2) Se debe ingresar la información básica requerida por el sistema para crear un estado de cama 3) Notifica sobre creación del nuevo estado de cama	
FLUJO ALTERNATIVO: 2.a) Si falta información dentro de los parámetros requeridos, solicitará validar los campos resaltados en rojo 2.b) Si el estado de cama ya se encuentra ingresado, el sistema notificará la no creación del estado de cama	

Nota: tabla en la que se detalla los flujos para crear estados de cama.

2.3.4. Crear cama

Para el ingreso de camas la aplicación valida el acceso a dicho módulo por perfil del usuario, de ser el perfil necesario debe ingresar los datos solicitados en el formulario, ya que de esto dependerá el flujo de pantallas y mensajes que el sistema desplegará (Ver Tabla 5) con la finalidad que el usuario identifique los pasos a seguir.

Tabla 5 HU: Creación de camas

HISTORIA DE USUARIO	
NÚMERO: 4	PERFIL: ADMINISTRADOR
NOMBRE: CREACIÓN DE CAMAS	
PRIORIDAD: MEDIA	RIESGO: MEDIO
FLUJO NORMAL: 1) Dentro las opciones del menú se debe seleccionar la opción crear cama 2) Se debe ingresar la información básica requerida por el sistema para crear una cama	

HISTORIA DE USUARIO
3) Notifica sobre creación de la nueva cama
FLUJO ALTERNATIVO: 2.a) Si falta información dentro de los parámetros requeridos, solicitará validar los campos resaltados en rojo 2.b) Si la cama ya se encuentra ingresada, el sistema notificará la no creación de la cama

Nota: tabla en la que se detalla los flujos para crear camas.

2.3.5. Crear unidad / subunidad

Para el ingreso de nueva unidad o subunidad la aplicación valida el acceso a dicho módulo por perfil del usuario, de ser el perfil necesario debe ingresar los datos solicitados en el formulario, ya que de esto dependerá el flujo de pantallas y mensajes que el sistema desplegará (Ver Tabla 6) con la finalidad que el usuario identifique los pasos a seguir.

Tabla 6 HU: Crear unidad y subunidad

HISTORIA DE USUARIO	
NÚMERO: 5	PERFIL: ADMINISTRADOR
NOMBRE: CREAR UNIDAD Y SUBUNIDAD	
PRIORIDAD: ALTA	RIESGO: MEDIO
FLUJO NORMAL: 1) Dentro las opciones del menú se selecciona la opción crear unidad / subunidad 2) Se debe ingresar la información básica requerida por el sistema para una unidad / subunidad 3) Notifica sobre creación de la nueva unidad / subunidad	
FLUJO ALTERNATIVO: 2.a) Si falta información dentro de los parámetros requeridos, solicitará validar los campos resaltados en rojo 2.b) Si la unidad / subunidad se encuentra ingresada, el sistema notificará la no creación de la unidad / subunidad	

Nota: tabla en la que se detalla los flujos para crear unidades y subunidades.

2.3.6. Ingresar paciente

Para el ingreso de pacientes la aplicación valida el acceso a dicho módulo por perfil del usuario, de ser el perfil necesario debe ingresar los datos solicitados en el formulario, ya que de esto dependerá el flujo de pantallas y mensajes que el sistema desplegará (Ver Tabla 7) con la finalidad que el usuario identifique los pasos a seguir.

Tabla 7 HU: Ingreso paciente

HISTORIA DE USUARIO	
NÚMERO: 6	PERFIL: ADMINISTRADOR
NOMBRE: INGRESO PACIENTE	
PRIORIDAD: ALTA	RIESGO: MEDIO
FLUJO NORMAL: 1) Dentro las opciones del menú se selecciona la opción ingresar paciente 2) Se debe ingresar la información básica requerida por el sistema para un paciente 3) Notifica sobre el ingreso del paciente	
FLUJO ALTERNATIVO: 2.a) Si falta información dentro de los parámetros requeridos, solicitará validar los campos resaltados en rojo 2.b) Si el paciente se encuentra ingresado, el sistema notificará el no ingreso del paciente	

Nota: tabla en la que se detalla los flujos para registrar pacientes.

2.3.7. Obtener informes

Para obtener informes la aplicación valida el acceso a dicho módulo por perfil del usuario, de ser el perfil necesario debe seleccionar los datos solicitados en el formulario, ya que de esto dependerá el flujo de pantallas y mensajes que el sistema desplegará (Ver Tabla 8) con la finalidad que el usuario identifique los pasos a seguir.

Tabla 8 HU: Obtener informes

HISTORIA DE USUARIO	
NÚMERO: 7	PERFIL: ADMINISTRADOR
NOMBRE: OBTENER INFORMES	
PRIORIDAD: ALTA	RIESGO: BAJO
FLUJO NORMAL: 1) Dentro las opciones del menú se selecciona la opción obtener informes 2) Se debe ingresar los parámetros requeridos por el sistema para obtener un informe 3) Notifica sobre la generación del informe	
FLUJO ALTERNATIVO: 2.a) Si los parámetros están mal ingresados, solicitará validar nuevamente los datos ingresados 2.b) Si el informe se encuentra fuera de los parámetros, el sistema lo indicará mediante un mensaje de alerta	

Nota: tabla en la que se detalla los flujos para generar reportes.

2.3.8. Transferencia de pacientes

Para el ingreso a la aplicación se debe contar con conexión a intranet e ingresar el usuario y contraseña asignados al usuario, ya que de esto dependerá el flujo de pantallas y mensajes que el sistema desplegará (Ver Tabla 9) con la finalidad que el usuario visualice la pantalla correcta o en su caso sepa que falló y proceda a corregir.

Tabla 9 HU: Transferencia de pacientes

HISTORIA DE USUARIO	
NÚMERO: 8	PERFIL: ADMINISTRADOR
NOMBRE: TRANSFERENCIA DE PACIENTES	
PRIORIDAD: ALTA	RIESGO: BAJO
FLUJO NORMAL: 1) Dentro las opciones del menú se selecciona la opción transferir paciente 2) Se debe ingresar al paciente antes de transferir a otra área	

HISTORIA DE USUARIO
3) Posterior a esto se debe ingresar la información básica requerida por el sistema para transferir un paciente 4) Notifica sobre la transferencia del paciente
FLUJO ALTERNATIVO: 2.a) Si el paciente no está registrado antes notificará, para que sea ingresado antes de la transferencia 2.b) Si falta información dentro de los parámetros requeridos, solicitará validar los campos resaltados en rojo

Nota: tabla en la que se detalla los flujos para transferir pacientes.

2.4. Diagramas de secuencia

Especifican la interacción entre objetos del sistema mediante mensajes, determinando el tiempo de vida de la acción a ejecutar.

2.4.1. Creación usuario

Permite establecer la relación entre los tres objetos que interactúan al momento de crear un cliente los que son: usuario, SISCAM y BBDD intercambiando mensajes (Ver Figura 16) finalizando con la respuesta de éxito o negación sobre la petición realizada.

Interacción para creación de usuario

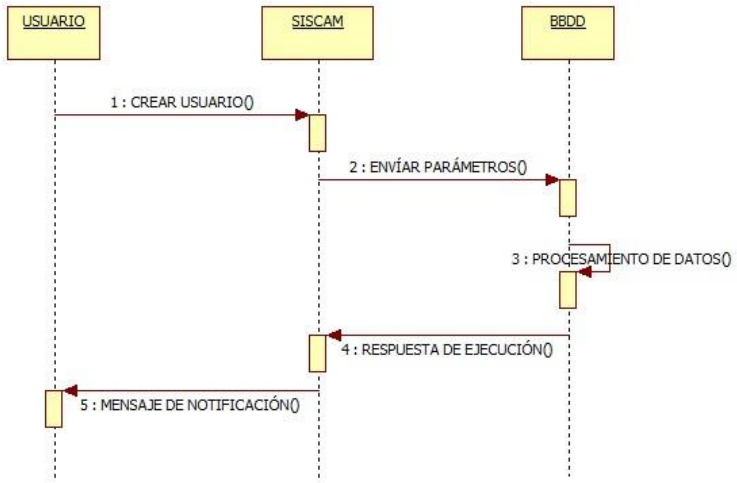

Figura 16 Diagrama de secuencia creación de usuario

Elaborado por: Jonathan Achig y Luis Padilla

2.4.2. Creación unidad

Permite establecer la relación entre los tres objetos que interactúan al momento de crear una unidad los que son: usuario, SISCAM y BBDD intercambiando mensajes (Ver Figura 17) finalizando con la respuesta de éxito o negación sobre la petición realizada.

Interacción para creación de unidad

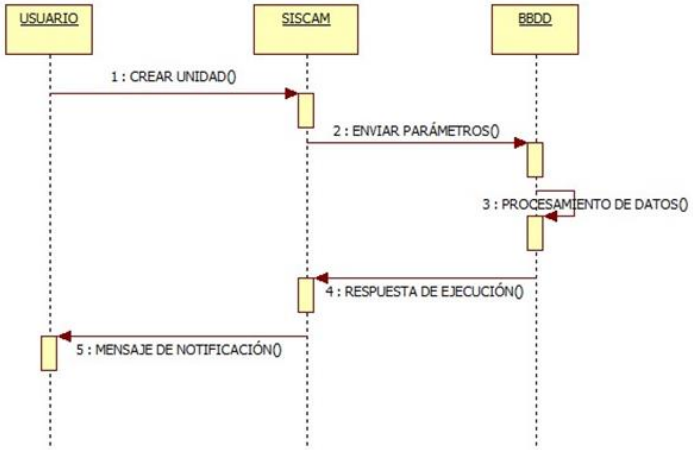

Figura 17 Diagrama de secuencia creación de unidad

Elaborado por: Jonathan Achig y Luis Padilla

2.4.3. Creación subunidad

Permite establecer la relación entre los tres objetos que interactúan al momento de crear una subunidad los que son: usuario, SISCAM y BBDD intercambiando mensajes (Ver Figura 18) finalizando con la respuesta de éxito o negación sobre la petición realizada.

Interacción para creación de subunidad

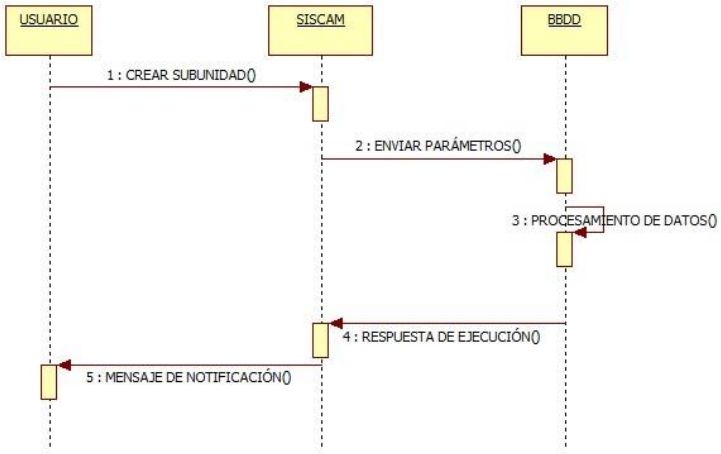

Figura 18 Diagrama de secuencia creación de subunidad

Elaborado por: Jonathan Achig y Luis Padilla

2.4.4. Creación cama

Permite establecer la relación entre los tres objetos que interactúan al momento de crear una cama los que son: usuario, SISCAM y BBDD intercambiando mensajes (Ver Figura 19) finalizando con la respuesta de éxito o negación sobre la petición realizada.

Interacción para creación de cama

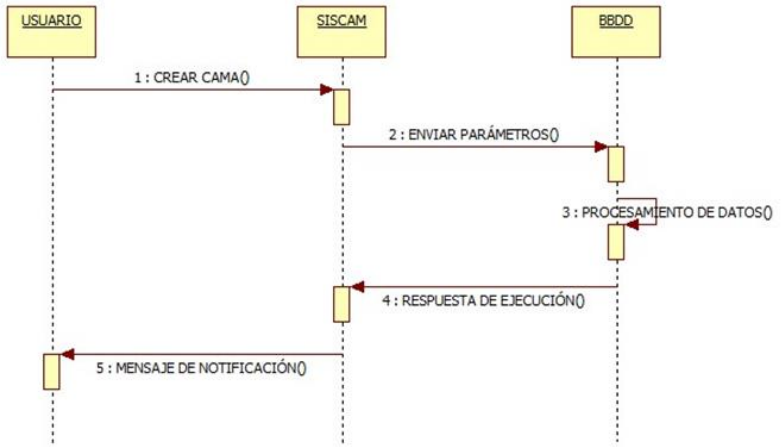

Figura 19 Diagrama de secuencia creación de cama

Elaborado por: Jonathan Achig y Luis Padilla

2.4.5. Creación estados de cama

Permite establecer la relación entre los tres objetos que interactúan al momento de crear un estado de cama los que son: usuario, SISCAM y BBDD intercambiando mensajes (Ver Figura 20) finalizando con la respuesta de éxito o negación sobre la petición realizada.

Interacción para creación estados de cama

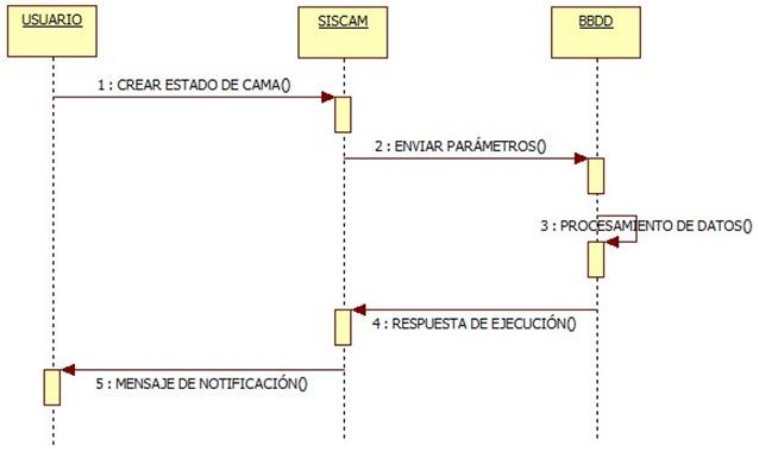

Figura 20 Diagrama de secuencia creación estados de cama

Elaborado por: Jonathan Achig y Luis Padilla

2.4.6. Ingreso de paciente

Permite establecer la relación entre los tres objetos que interactúan al momento de ingresar un paciente los que son: usuario, SISCAM y BBDD intercambiando mensajes (Ver Figura 21) finalizando con la respuesta de éxito o negación sobre la petición realizada.

Interacción para ingreso de paciente

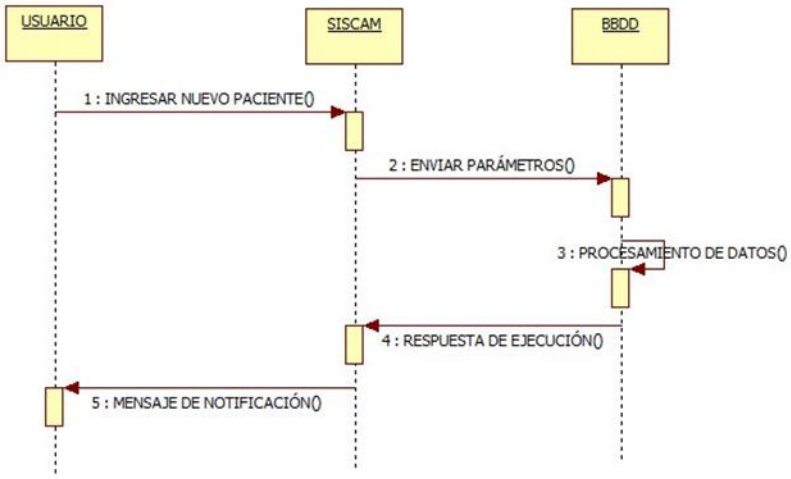

Figura 21 Diagrama de secuencia ingreso de paciente

Elaborado por: Jonathan Achig y Luis Padilla

2.4.7. Obtener informes

Permite establecer la relación entre los tres objetos que interactúan al momento de obtener informe los que son: usuario, SISCAM y BBDD intercambiando mensajes (Ver Figura 22) finalizando con la respuesta de éxito o negación sobre la petición realizada.

Interacción para obtener informe de camas y pacientes

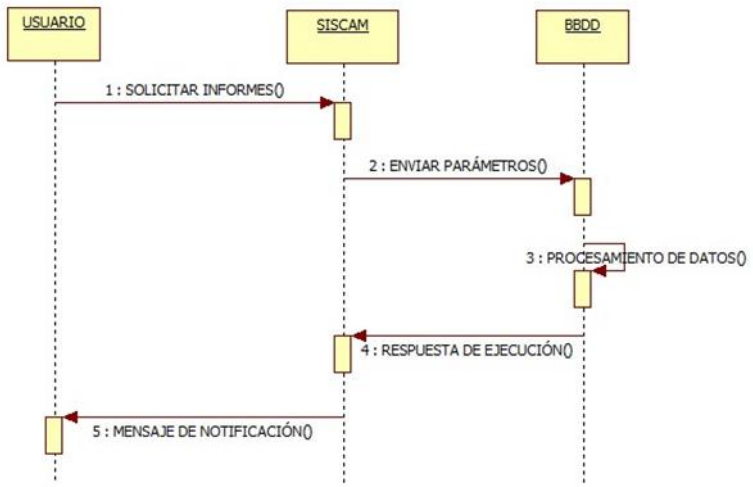

Figura 22 Diagrama de secuencia para obtener informe de camas y pacientes

Elaborado por: Jonathan Achig y Luis Padilla

2.4.8. Transferencia de paciente

Permite establecer la relación entre los tres objetos que interactúan al momento de realizar la transferencia de pacientes los que son: usuario, SISCAM y BBDD intercambiando mensajes (Ver Figura 23) finalizando con la respuesta de éxito o negación sobre la petición realizada.

Interacción para transferencia de paciente

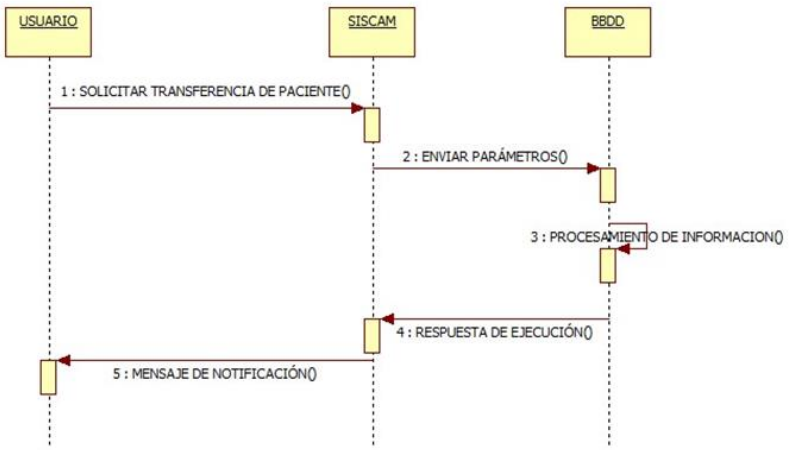

Figura 23 Diagrama de secuencia transferencia de paciente

Elaborado por: Jonathan Achig y Luis Padilla

2.5. Diseño de base de datos

Mediante el correcto diseño de la base de datos, modelamos la forma en que la información será almacenada y procesada para su posterior visualización. En la Figura 24 se describe las relaciones que tienen las tablas de la base de datos de una manera lógica identificando entidades, especificando atributos y especificando la relación que tiene cada una de ellas. Mientras que en la Figura 25 se describe de manera física asegurando la integridad de los datos al evitar repeticiones innecesarias de datos, transformando las entidades en tablas, los atributos en columnas y las instancias en filas.

Diagrama Lógico de Base de Datos

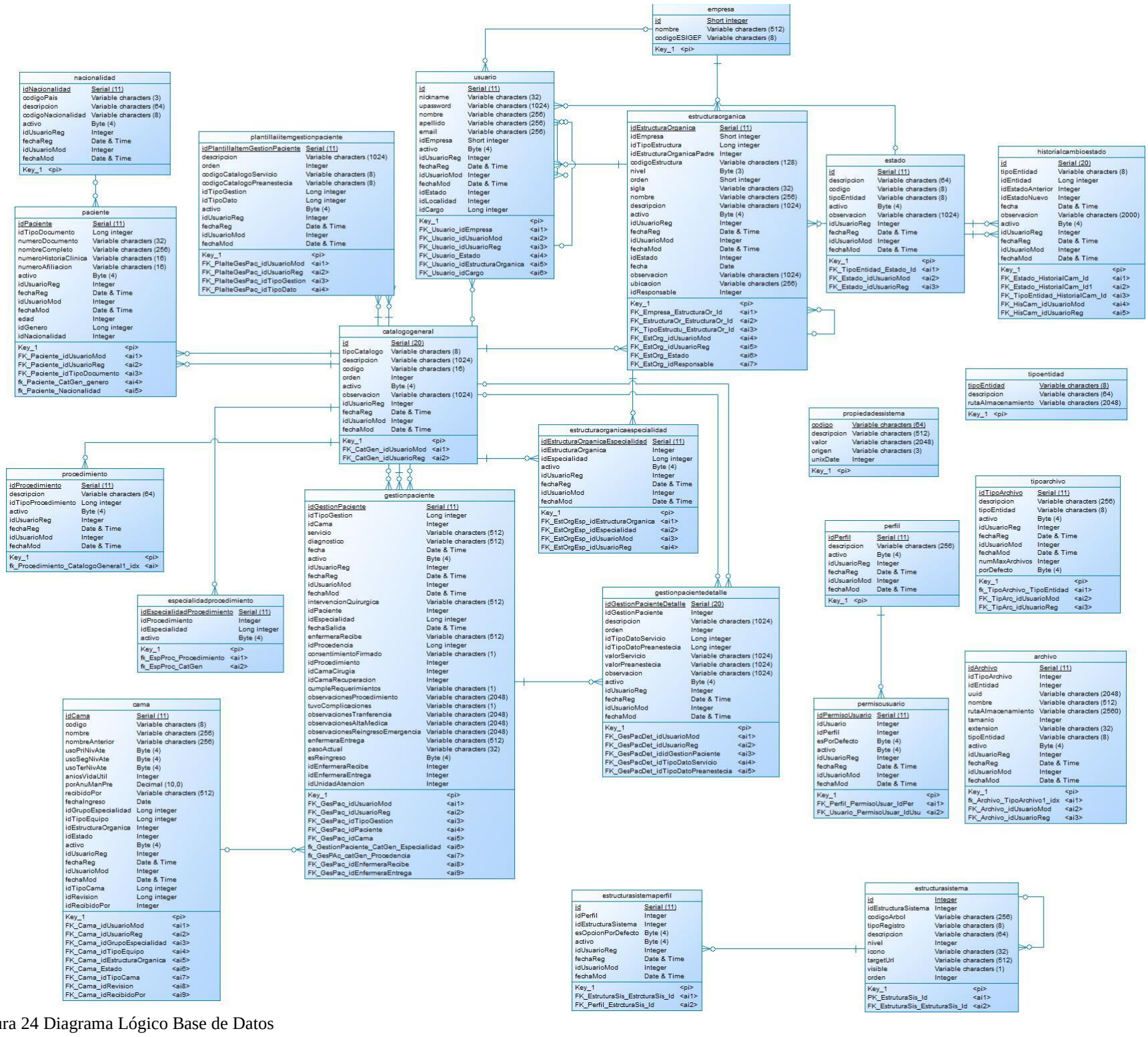

Figura 24 Diagrama Lógico Base de Datos

Elaborado por: Jonathan Achig y Luis Padilla

Diagrama Físico Base de Datos

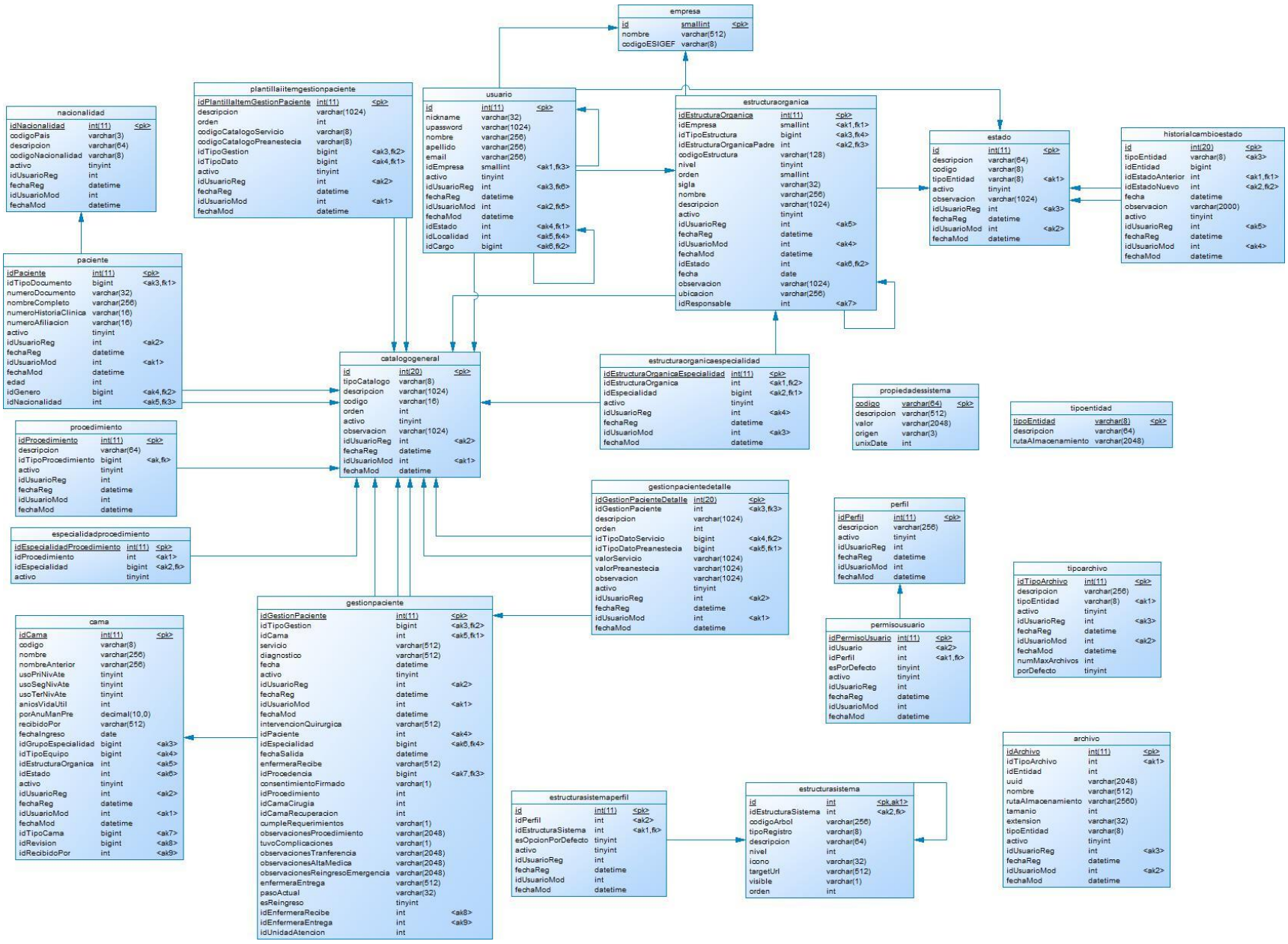

Figura 25 Diagrama Físico Base de Datos

Elaborado por: Jonathan Achig y Luis Padilla

2.6. Diagrama navegacional

Representa la estructura del contenido en el sistema, presentando la navegación entre módulos determinando el número de pantallas (Ver Figura 26) que el sistema contiene en base a los diagramas de casos de uso definidos previamente.

Diagrama Navegacional

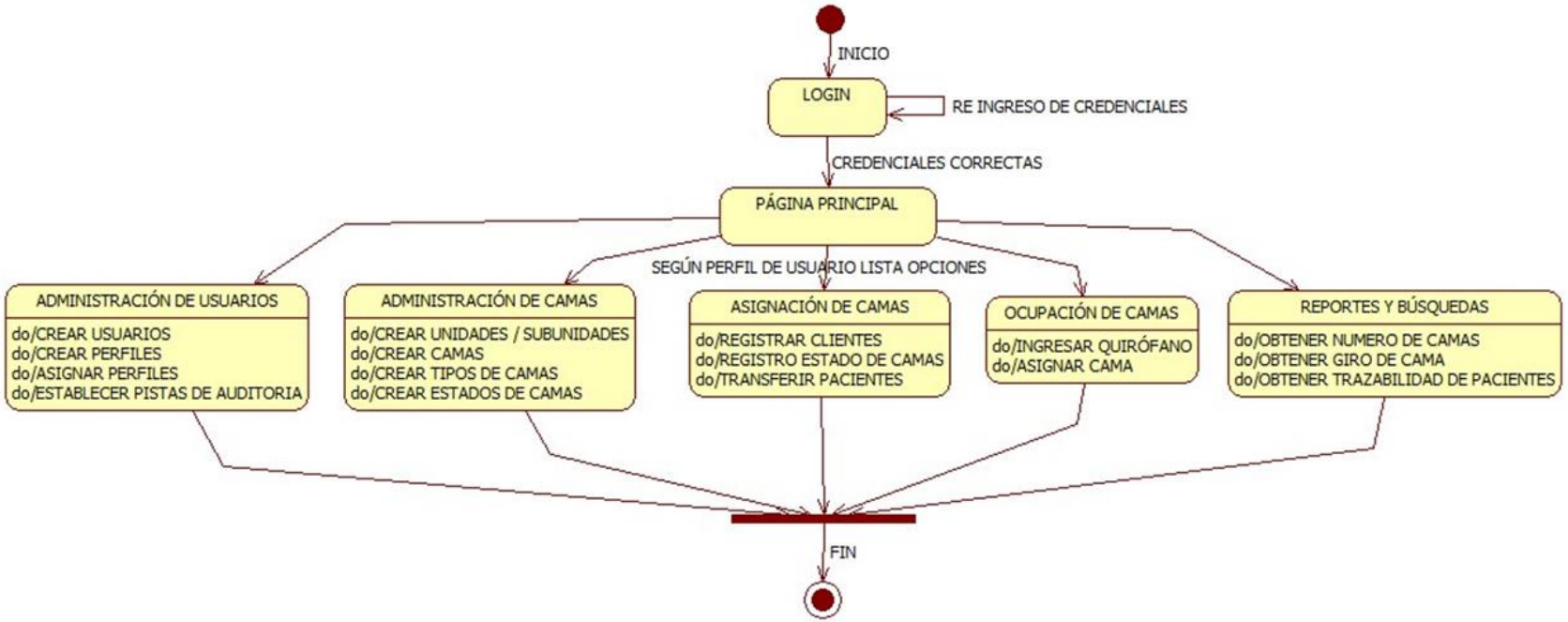

Figura 26 Diagrama de navegación

Elaborado por: Jonathan Achig y Luis Padilla

Capítulo 3

Construcción y Pruebas

3.1. Descripción diagrama de despliegue

El diagrama de despliegue está estructurado bajo las capas de Backend y Frontend descritas en la Figura 27, estableciendo como elementos principales servidor de aplicaciones Tomcat, servidor web Nginx y como motor de base de datos MySql.

Diagrama de despliegue

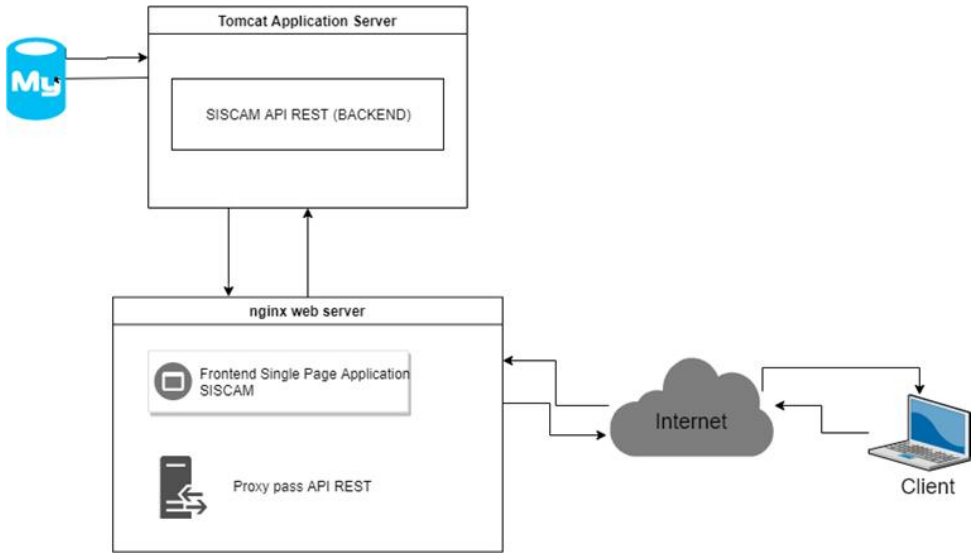

Figura 27 Diagrama de despliegue

Elaborado por: Jonathan Achig y Luis Padilla

3.2. Backend

La capa de backend almacena la lógica del negocio, representado en servicios que serán publicados y consumidos con las peticiones de los usuarios mediante un frontend. Entre los componentes se tiene:

3.2.1. Web layer

La Web Layer contiene la lógica de presentación almacenada en el servidor web, dando respuesta a las peticiones de los usuarios, bajo la siguiente estructura:

- HTTP Listener: dentro de la web layer el primer componente usado es el HTTP listener el cual permite establecer comunicaciones seguras entre clientes y la unidad principal a través de uno o más oyentes. Los escuchas HTTP deben estar asociados a un socket que tiene una dirección IP, un número de puerto, un nombre de servidor y un servidor virtual, todos estos predeterminados.

- Controler Layer: este componente se encarga de gestionar las órdenes enviadas por el usuario, solicitando información al Modelo y enviándolos a la Vista. Este controlador está realizado bajo Spring Boot que permite la creación de aplicaciones independientes con servicios web Restful basadas en Spring, mismo que se basa en el patrón de diseño MVC.

3.2.2. Business layer

El Business Layer contiene los servicios necesarios para ejecutar la lógica del negocio (Ver Figura 28), recibiendo y enviando respuesta luego de su procesamiento, bajo la siguiente estructura:

- Business Services: para este componente de servicio los archivos de clase se utilizan para escribir la lógica del negocio, separando los archivos de clase @RestController y @Services aunque es posible colocarlos en una sola capa, una buena práctica es separarlos en capas distintas.

- Repository Services: este componente está basado en el framework de Spring Data que se encuentra dentro de Spring, el objetivo que tiene es el de simplificar el

desarrollo de la persistencia de datos versus distintos repositorios de información. Además de facilitar el uso de tecnologías de acceso a datos.

- Model: componente de Java basado en JPA (Java Persistence API) usado para relacionar la POO (Programación Orientada a Objetos) y las bases de datos relacionales. Mediante JPA se puede gestionar el almacenamiento, actualización, eliminación y recuperación de datos, de bases de datos relacionales a objetos JAVA y viceversa.

- Mysql (SISCAM DB): es un gestor de base de datos, basado en MVC, que puede llegar a comunicar de manera eficiente varios clientes y servidores al mismo tiempo entre sí y manteniendo su rendimiento como programa. Este gestor está desarrollado en lenguaje SQL, usado para consultar y renovar datos en la gestión de una base de registros. En la Figura 28 se representa los componentes antes descritos:

Diagrama Back End

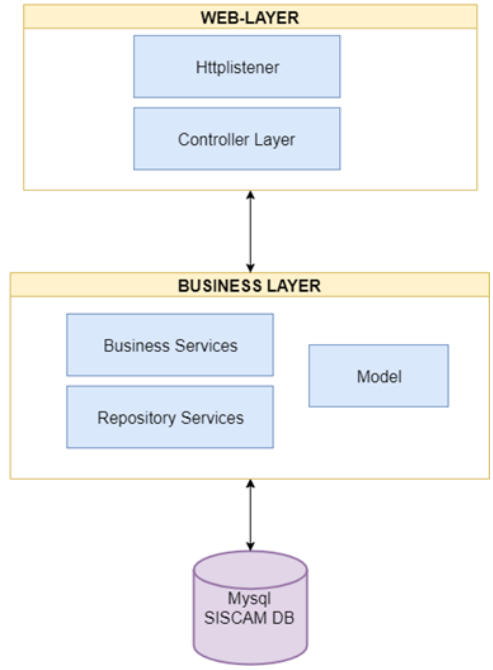

Figura 28 Diagrama Back End

Elaborado por: Jonathan Achig y Luis Padilla

3.3. Frontend

La capa de frontend interactúa de manera directa con el usuario, desplegando una interfaz gráfica mediante la que se puede enviar y recibir información (Ver Figura 29). Entre los componentes se tiene:

3.3.1. Single page application

Es la interacción de la aplicación cargada en una sola página, permitiendo el acceso a datos mediante una interfaz dinámica y el almacenamiento de datos de sesión mediante el storage del navegador, bajo la siguiente estructura:

- UI, HTML y CSS: En este componente se establecen las herramientas utilizadas para la elaboración de la interfaz de usuario (UI) basada en lenguaje de programación HTML con sus correspondientes librerías y los componentes visuales de CSS, corridos sobre un navegador web que lo soporte.

- Controllers: para los controladores dentro del nodo, usamos AngularJS del lado del cliente debido a que es una aplicación web. AngularJS nos ayuda en mantener una jerarquía de componentes dentro de la aplicación, siendo más simple la configuración de directivas. Los componentes son widgets que contienen y controlan el conocimiento de cómo se ven, cómo actúan y cómo interactúan con el exterior.

- Navigation APIs: los componentes para la interfaz de usuario usamos PrimeNG es una colección de UI perteneciente a Angular. Teniendo varios widgets de código abierto y de licencias MIT de uso gratuito, ofreciendo una selección de temas prefabricados y componentes de UI para la presentación de datos, entradas de formularios, menús, gráficos y superposiciones.

Diagrama Front End

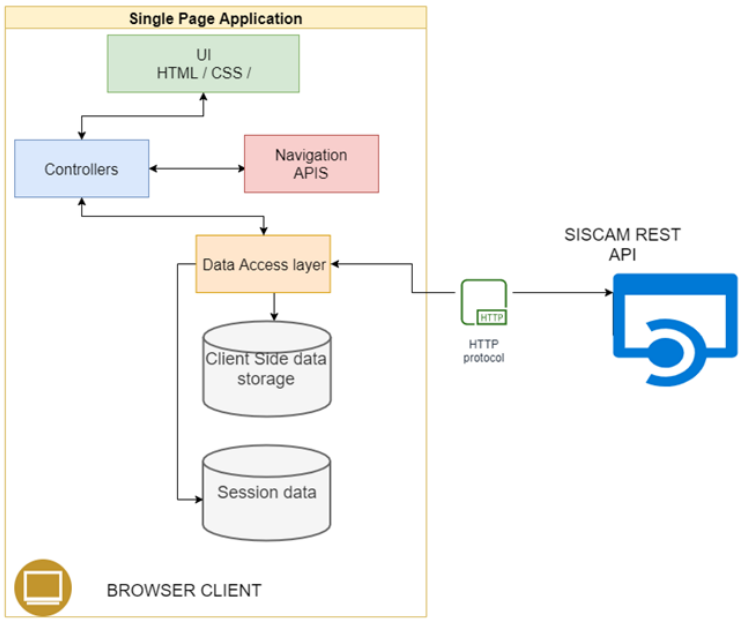

Figura 29 Diagrama de Front End

Elaborado por: Jonathan Achig y Luis Padilla

3.4. Diccionario de datos

Mediante el diccionario de datos se describe los componentes de la base de datos, tales como: tablas, campos, atributos que permitan la comprensión de la información almacenada y procesada por la aplicación SISCAM. Ver Anexo A

3.5. Métodos más importantes

Debido al diseño de la aplicación y al objetivo de optimizar código se ha generado elementos transversales, dichos elementos están asociados a diferentes componentes del sistema para, de esta manera controlar de forma genérica varios componentes con un solo elemento (método).

Un componente transversal tiene como finalidad atravesar las diferentes capas que conforman un sistema informático para de esta manera implementar la lógica que llevan consigo y que pueden abarcar funciones comunes entre las cuales se puede mencionar las siguientes:

- La autenticación de usuario

- Los permisos
- La comunicación
- La validación de datos
- El almacenamiento en memoria caché
- La gestión de errores

Es así como realizando un análisis del código se pude definir como métodos más importantes a elementos transversales como son:

- Filtro validación de usuario: durante el uso del sistema, es necesario verificar si el usuario que ingresó o que pretende ingresar es un usuario válido, siendo necesario contar con un elemento que se encargue de dicha función de forma constante, ya que puede ocurrir que la sesión caduque o que haya intentos de acceso por parte de terceros. Por lo que se encarga de verificar que el usuario que hace login sea válido y esté registrado en el sistema.

  La librería se encarga de generar un token y verificar mediante un filtro si este cumple con todas las características de este (si la clave de token es correcta) de este modo se vuelve un token válido. A continuación, el código para llamar a la librería:

```
org.springframework.security.web.authentication.www.BasicAuthenticationF
ilter;
```

  Para Jwapi de Json web token se realiza:

```
<dependency>
	<groupId>io.jsonwebtoken</groupId>
	<artifactId>jjwt-api</artifactId>
	<version>0.10.5</version>
</dependency>
```

La autenticación por tokens es una tendencia en la programación web, el protocolo a usar es HTTP, mismo que lleva un token o firma cifrada en su cabecera por petición. Permitiendo al API identificar al usuario, el token no se almacena en el servidor, sino en el lado del cliente siendo el API el encargado de descifrar y redireccionar el flujo del proceso.

Para esto también se ha empleado la Autenticación con Json Web Token para controlar los inicios de sesión u otras características de autenticación.

- Filtro JWTAuthorizationFilter.java: se encarga de filtrar las peticiones siendo un componente propio de aplicaciones JAVA, además de otras, que por cada petición que llegue al servidor ejecuta cierta lógica programada para verificar si estas son válidas, sino son rechazadas. El método principal es doFilterInternal, descrito a continuación:

```
@Override
    protected void doFilterInternal(HttpServletRequest request, HttpServ
letResponse response, FilterChain chain)
            throws IOException, ServletException {
        LOG.info("Executing doInternalFilter");
        String header = request.getHeader("Authorization");
        if(!requiresAuthentication(header)) {
            chain.doFilter(request, response);
            return;
        }
        AuthenticationInfo authentication = null;
        if(jwtService.validate(header)) {
            String userName = jwtService.getUsername(header);
            Integer userId = jwtService.getUserId(header);
            authentication = new AuthenticationInfo( userName, null, jwt
Service.getRoles(header));
            authentication.setUserId(userId);
            SecurityContextHolder.getContext().setAuthentication(authent
ication);
            chain.doFilter(request, response);
        }else {
            LOG.error("token incorrecto");
            onUnsuccessfulAuthentication(request, response, null);
        }
    }
```

Este método es propio de la clase BasicAuthenticationFilter que en esencia se encarga de procesar los encabezados de autorización de una solicitud HTTP, dicha clase es importada de la librería descrita anteriormente.

Una vez ejecutado el método antes descrito, y si la autenticación de usuario fue aceptada, este se encarga de enviar la petición a los demás componentes del sistema, dando información respecto al elemento con la anotación.

A demás esté método se encarga de verificar si la sesión del usuario ha caducado, ya que el token está programado para caducar después de un cierto periodo de tiempo con el fin de evitar ataque de negación de servicios.

- Validación de campos requeridos: en el ingreso de información como en la autenticación se presenta el caso de campos requeridos que no pueden ser ignorados por el usuario ya que son indispensables para el manejo interno de la información asociada, a estos se los define como campos requeridos u obligatorios, estos campos se presentan en todas las interacciones de ingreso de información en el sistema. Este elemento se encarga de verificar los campos obligatorios en todos los campos del sistema.

  Las librerías y anotaciones por usar se describen en las siguientes líneas de código:

```
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
@Retention(RetentionPolicy.RUNTIME)
@Target(FIELD)
public @interface Required {
    String customFieldName() default "";
}
```

  Las anotaciones proporcionan datos sobre un programa que no forma parte del programa en sí. No tienen efecto directo sobre el funcionamiento del código que anotan.

Y estás anotaciones permiten incrustar información adicional en un archivo fuente, todo esto sin afectar las acciones que el programa está realizando, pero puede ser utilizado varias veces por las herramientas del sistema.

La semántica del programa no es afectada por las anotaciones de forma directa, pero sí afectan la forma en que las herramientas y las bibliotecas tratan los programas, lo que a su vez puede afectar la semántica del programa en ejecución. Las anotaciones se pueden leer desde archivos de origen, archivos de clase o reflexivamente en tiempo de ejecución.

Esta lógica genérica creada para este método fue concebida con las anotaciones en java, requiere la siguiente estructura:

```
public static void validateRequiredFields(Object o)  throws SiscamExcept
ion{
        for(Field field :   o.getClass().getDeclaredFields()) {
            if(!field.isAnnotationPresent(Required.class))
                continue;
            Required requerido = getRequiredAnnotation(field);
            try {
                if(String.class.equals(field.getType()) ) {
                    Object obj = getMethodFromField(o, field).invoke(o);
                    Assert.hasText(
                            obj == null? null: obj.toString(),
                            getRequiredMessage(field, requerido));
                }else {
                    Assert.notNull(getMethodFromField(o, field).invoke(o
), getRequiredMessage(field, requerido));
                }
            } catch (IllegalAccessException | InvocationTargetException
e) {
                e.printStackTrace();
            }catch (IllegalArgumentException e) {
                e.printStackTrace();
                throw new SiscamException(e.getMessage(), Boolean.TRUE);
            }
        }
    }
```

En este caso cuando el usuario intenta enviar o guardar información de una ventana y esta posee uno o varios campos requeridos se llama a un método, estructurado de forma genérica para aceptar cualquier texto ingresado, que se encarga de verificar si el usuario ha ingresado texto, esto sucede cada vez que el usuario intenta realizar dicha acción, de esta manera se optimiza código al utilizar un solo método para todos los casos de campos requeridos.

3.6. Pruebas de caja negra (flujo alternativo)

Es una de las técnicas de prueba de software que sin tomar en cuenta la estructura del código nos permite verificar la funcionalidad, validando escenarios o detalles de implementación del software en base a las entradas y salidas del sistema, sin considerar la estructura interna del programa. En las Tablas 10, 11 y 12 se describen las condiciones a realizar numeradas por el número de intentos descritos por las respuestas SI (S) y NO (N), dando como resultado las acciones que se ejecutarán marcadas con una (X). A continuación, las pruebas realizadas:

3.6.1. Loguear

Descripción del caso: el sistema mostrará una alerta de usuario o contraseña incorrecta al momento que las credenciales ingresadas no corresponden a ninguna de las registradas, caso contrario se muestra el menú principal acorde al perfil.

Técnica de prueba de caja negra: Requerimiento funcional / Casos de Uso (CU)

Caso 1.1 Referencia de entrada: ingreso de usuario (nickname) y contraseña correctos. Resultado esperado (Salida): ingreso al menú principal del sistema acorde al perfil del usuario registrado (Ver Figura 30).

Caso 1.2 Referencia de entrada: ingreso de usuario (nickname) y contraseña incorrectos / no registrados. Resultado esperado (Salida): mensaje de alerta indicando que las credenciales son

incorrectas, por lo que no se puede ingresar al menú principal y pide volver al login (Ver Figura 31).

Tabla 10 Test caja negra login

Condiciones	1	2
¿Usuario correcto?	S	N
¿Password correcto?	S	S
Acciones	**1**	**2**
Ingresa al sistema	X	
Usuario o password incorrecto		X

Nota: en la tabla se detalla las acciones ejecutadas para login

Mensaje éxito login

Figura 30 Mensaje éxito login

Elaborado por: Jonathan Achig y Luis Padilla

Mensaje alerta login

Figura 31 Mensaje alerta login

Elaborado por: Jonathan Achig y Luis Padilla

3.6.2. Crear usuario

Descripción del caso: al crear un usuario validará que todos los campos como requeridos dentro del formulario sean ingresados, caso contrario indicará el campo que haga falta.

Técnica de prueba de caja negra: Requerimiento funcional / CU

Caso 2.1: Referencia de entrada: ingresamos nombre de usuario, apellidos, nombres, empresa, perfil, estado, cargo y unidad grabando al final del procedimiento, el campo que no se ingresa es email. Resultado esperado (Salida): mensaje de alerta indicando que el campo email es requerido (Ver Figura 33).

Caso 2.2: Referencia de entrada: ingresamos nombre de usuario, email, apellidos, nombres, empresa, perfil, estado, cargo y unidad grabando al final del procedimiento. Resultado esperado (Salida): mensaje de éxito indicando que usuario fue grabado (Ver Figura 32).

Caso 2.3: Referencia de entrada: ingresamos un nombre de usuario ya registrado en el sistema, email, apellidos, nombres, empresa, perfil, estado, cargo y unidad grabando al final del

procedimiento. Resultado esperado (Salida): mensaje de alerta indicando que el usuario ya existe, por lo que debo grabar con otro nombre.

Tabla 11 Test caja negra crear usuario

Condiciones	**1**	**2**	**3**
¿Ingresa todos los campos requeridos?	N	S	S
¿Repite un usuario ya registrado?	N	N	S
Acciones	**1**	**2**	**3**
Graba usuario		X	
Mensaje de verificar datos ingresados	X		X

Nota: en la tabla se detalla las acciones ejecutadas para crear usuario

Mensaje éxito crear usuario

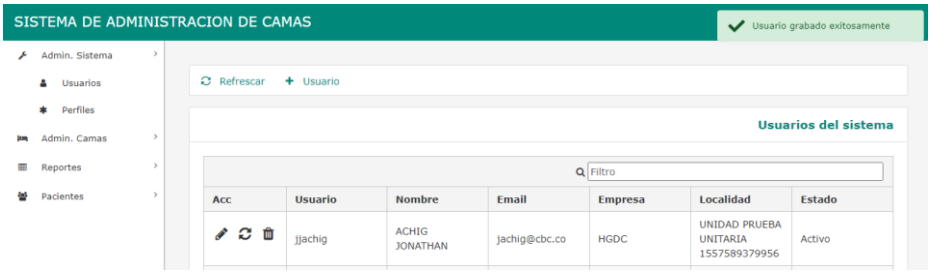

Figura 32 Mensaje éxito crear usuario

Elaborado por: Jonathan Achig y Luis Padilla

Mensaje alerta crear usuario

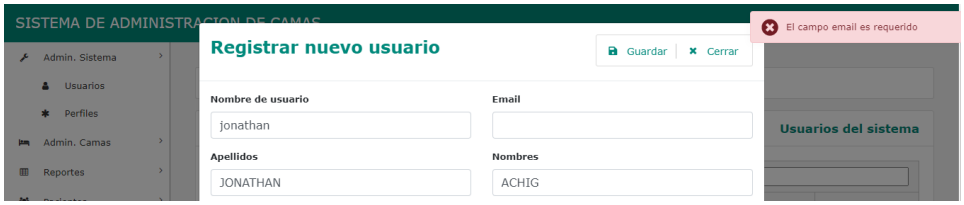

Figura 33 Mensaje alerta crear usuario

Elaborado por: Jonathan Achig y Luis Padilla

3.6.3. Ingresar paciente

Descripción del caso: en el módulo Paciente se tiene ingreso paciente, pidiendo como primer parámetro ingreso de cédula si lo encuentra se despliega resumen de atenciones, caso contrario podemos registrar nuevo procedimiento.

Técnica de prueba de caja negra: Requerimiento funcional / CU

Caso 3.1: Referencia de entrada: ingresamos número de cédula del paciente que recibe primera atención dentro del hospital. Resultado esperado (Salida): mensaje de alerta indicando que el paciente es nuevo y que se ingrese el nuevo procedimiento.

Caso 3.2: Referencia de entrada: ingresamos número de cédula del paciente que ya presenta atenciones previas en el hospital. Resultado esperado (Salida): mensaje de éxito indicando que paciente existe, desplegando el resumen de atenciones pudiendo validar cada una o generar un nuevo procedimiento.

Caso 3.3: Datos de entrada: registro de un nuevo procedimiento en el que constan los siguientes datos: número de cédula, enfermera que recibe, nombre del paciente, procedencia, número de afiliación, fecha e ingreso, fecha de salida, número de historia clínica, especialidad, edad, género, nacionalidad y lista de chequeo con una encuesta rápida al paciente antes de ingresar al

procedimiento. Resultado esperado (Salida): mensaje de éxito indicando que procedimiento sobre paciente fue grabado (Ver Figura 34).

Caso 3.4: Referencia de entrada: registro de un nuevo procedimiento en el que constan los siguientes datos: número de cédula, nombre del paciente, número de afiliación, fecha e ingreso, fecha de salida, número de historia clínica, especialidad, edad, género, nacionalidad y lista de chequeo, los datos que no se ingresa son: enfermera que recibe y procedencia, seguido pasamos a llenar una encuesta rápida al paciente antes de ingresar al procedimiento. Resultado esperado (Salida): mensaje de alerta indicando que campos de enfermera que recibe y procedencia son necesarios (Ver Figura 35).

Tabla 12 Test caja negra ingresar paciente

Condiciones	**1**	**2**	**3**	**4**
¿Ingresa cédula del paciente?	S	S	S	S
¿Ingresa enfermera y procedencia de paciente?			N	S
Acciones	**1**	**2**	**3**	**4**
Paciente nuevo ingresar información	X			
Paciente existe y visualiza las atenciones		X		
Paciente y procedimiento ingresado con éxito			X	
Alerta enfermera que recibe y procedencia son necesarios				X

Nota: en la tabla se detalla las acciones ejecutadas para ingresar paciente

Mensaje éxito ingresar paciente

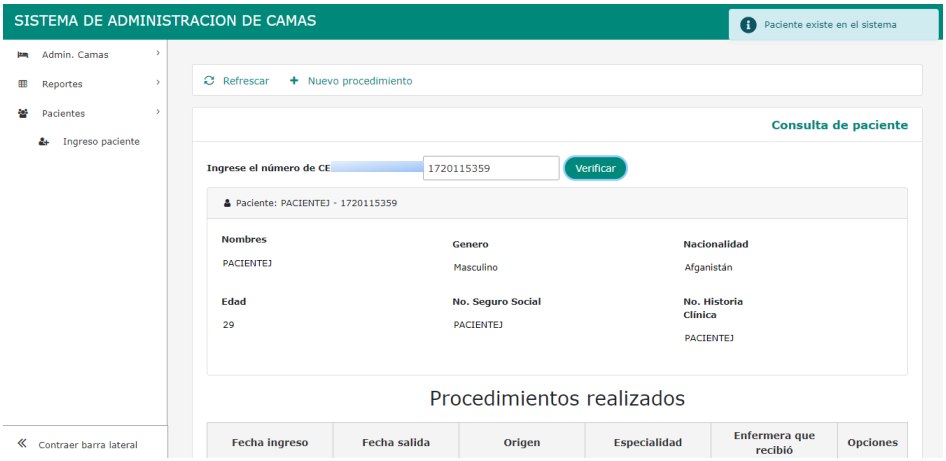

Figura 34 Mensaje éxito ingresar paciente

Elaborado por: Jonathan Achig y Luis Padilla

Mensaje alerta ingresar paciente

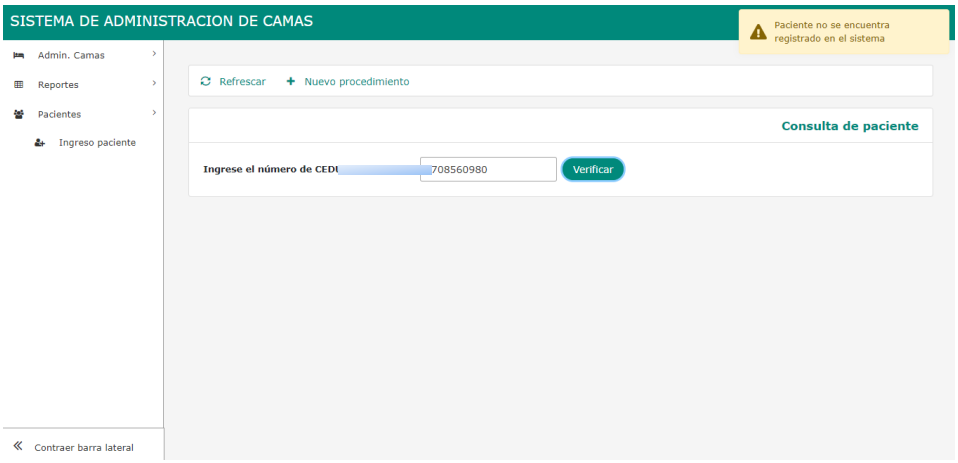

Figura 35 Mensaje alerta ingresar paciente

Elaborado por: Jonathan Achig y Luis Padilla

3.7. Pruebas de concurrencia

Determinan el comportamiento de una o varias actividades ejecutadas dentro de un intervalo de tiempo, realizadas de manera simultánea bajo varios escenarios posibles.

3.7.1. Jmeter

Apache JMETER permite testear el rendimiento de recursos sean estos dinámicos o estáticos, para aplicaciones WEB. Permitiendo simular cargas de datos al servidor, incrementar tráfico red poniendo a prueba el tiempo de respuesta y analizar su rendimiento.

Sus características más importantes son:

- Permite cargar y probar el tiempo de respuesta en aplicaciones, bajo varios protocolos: WEB, HTTP, HTTPS, entre otros.
- Pruebas IDE con varias funciones que permite: grabar, crear y depurar el plan de pruebas.
- Un informe HTML dinámico listo para presentar.
- Portabilidad completa y pureza al 100% JAVA.

3.7.2. Prueba ingreso de usuarios

En la prueba de ingreso de usuarios se realiza bajo los siguientes parámetros:

- 300 hilos
- 90 segundos
- 1 bucle

Se realiza peticiones HTTP a los módulos de PÁGINA PRINCIPAL, LOGIN e INGRESO USUARIOS. Realizando un total de 900 tests obteniendo una media de 289 peticiones, el tiempo de respuesta máximo es de 2757s, el porcentaje de peticiones con error son del 65,22% debido a que los datos enviados en cada prueba son varios unos correctos y otros incorrectos, sobre el rendimiento el sistema atiende 10 peticiones por segundo. En cuanto al rendimiento en kb se tiene un resultado de 6.56kb/s (Ver Figuras 36, 37).

Gráfico de resultados ingreso usuarios

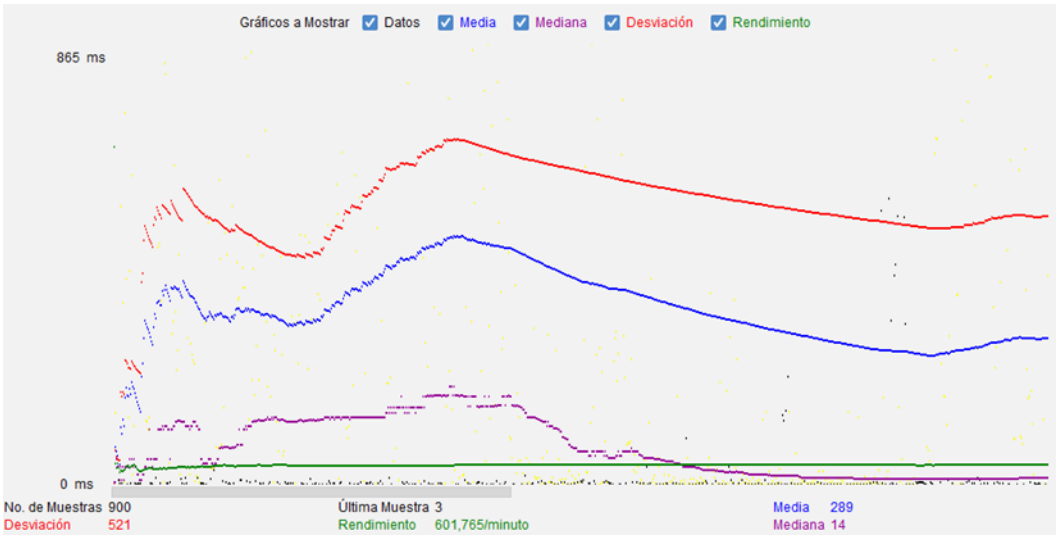

Figura 36 Gráfico de resultados ingreso usuarios

Elaborado por: Jonathan Achig y Luis Padilla

Resumen ingreso usuario

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Están...	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Byt...
Petición HT...	300	4	1	92	6,76	0,00%	3,3/sec	3,70	0,39	1131,0
Petición HT...	300	694	7	2757	702,90	97,67%	3,3/sec	1,25	0,80	383,4
Petición HT...	300	170	1	2201	249,86	98,00%	3,4/sec	1,65	2,77	504,9
Total	900	289	1	2757	521,62	65,22%	10,0/sec	6,59	3,94	673,1

Figura 37 Resumen ingreso usuario

Elaborado por: Jonathan Achig y Luis Padilla

3.7.3. Prueba ingreso pacientes

En la prueba de ingreso de usuarios se realiza bajo los siguientes parámetros:

- 500 hilos
- 120 segundos
- 2 bucles

Se realiza peticiones HTTP a los módulos de PÁGINA PRINCIPAL, LOGIN e INGRESO USUARIOS. Realizando un total de 3000 tests obteniendo una media de 231 peticiones, el

tiempo de respuesta máximo es de 5884s, el % de peticiones con error son del 65,40% debido a que los datos enviados en cada prueba son varios unos correctos y otros incorrectos, sobre el rendimiento el sistema atiende 25 peticiones por segundo. En cuanto al rendimiento en kb se tiene un resultado de 16.21kb/s (Ver Figura 38, 39).

Gráfico de resultados ingreso usuarios

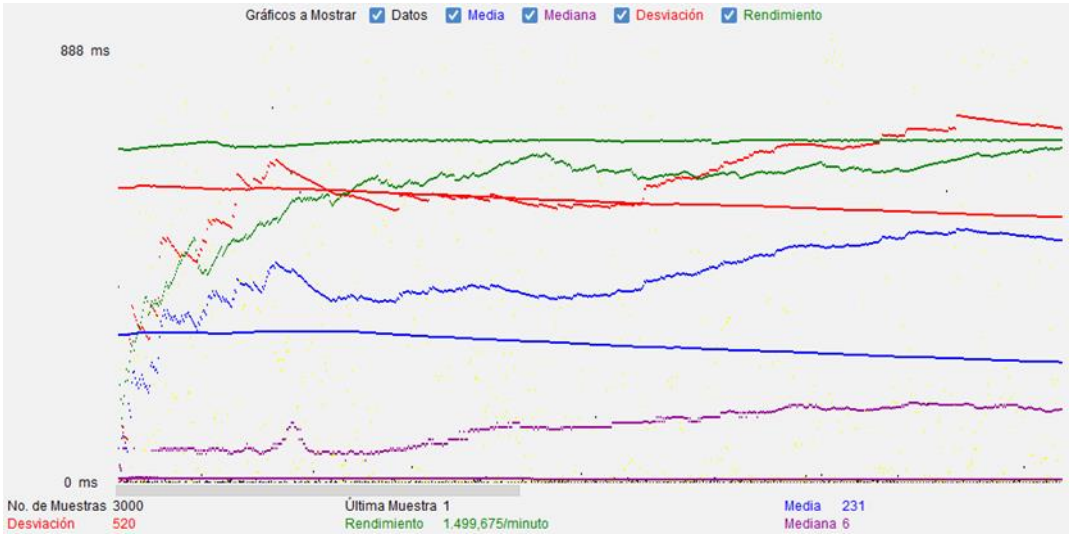

Figura 38 Gráfico de resultados ingreso usuarios

Elaborado por: Jonathan Achig y Luis Padilla

Resumen ingreso pacientes

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Están...	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Byt...
Petición HT...	1000	2	0	113	5,84	0,00%	8,3/sec	9,20	0,97	1131,0
Petición HT...	1000	553	6	3467	704,14	98,30%	8,3/sec	3,09	1,98	379,0
Petición HT...	1000	136	1	5884	389,89	97,90%	8,4/sec	3,95	6,90	482,3
Total	3000	231	0	5884	520,53	65,40%	25,0/sec	16,21	9,80	664,1

Figura 39 Resumen ingreso pacientes

Elaborado por: Jonathan Achig y Luis Padilla

3.7.4. Pruebas de obtención de informes de indicadores de calidad

En la prueba de obtención de indicadores de calidad se realiza bajo los siguientes parámetros:

- 200 hilos

- 60 segundos
- 2 bucles

Se realiza peticiones HTTP a los módulos de PÁGINA PRINCIPAL, LOGIN e INGRESO USUARIOS. Realizando un total de 1200 tests obteniendo una media de 299 peticiones, el tiempo de respuesta máximo es de 3701s, el % de peticiones con error son del 66,42% debido a que los datos enviados en cada prueba son varios unos correctos y otros incorrectos, sobre el rendimiento el sistema atiende 20 peticiones por segundo. En cuanto al rendimiento en kb se tiene un resultado de 12.99kb/s (Ver Figura 40, 41).

Gráfico de resultados obtención de indicadores de calidad

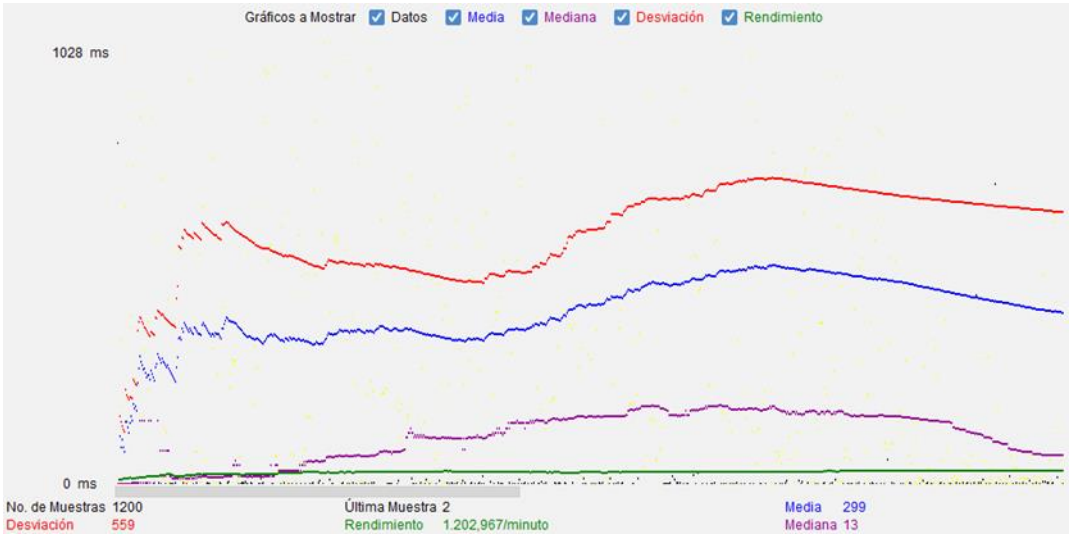

Figura 40 Gráfico de resultados obtención de reportes de indicadores de calidad

Elaborado por: Jonathan Achig y Luis Padilla

Resumen obtención reportes de indicadores de calidad

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Están...	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Byt...
Petición HT...	400	3	0	64	6,22	0,00%	6,7/sec	7,38	0,78	1131,0
Petición HT...	400	753	6	3701	761,00	99,50%	6,7/sec	2,42	1,59	370,5
Petición HT...	400	142	1	1071	204,01	99,75%	6,7/sec	3,21	2,02	488,6
Total	1200	299	0	3701	559,48	66,42%	20,0/sec	12,99	4,36	663,4

Figura 41 Resumen obtención reportes de indicadores de calidad

Elaborado por: Jonathan Achig y Luis Padilla

Conclusiones

- Realizando un análisis de requerimientos y aplicando la metodología ágil de desarrollo XP se obtuvo como resultado un software de fácil uso e intuitivo que cuenta con cinco módulos que permiten desarrollar de principio a fin el proceso de asignación de camas dentro del Hospital General Docente de Calderón.

- La aplicación correcta de la metodología de desarrollo permite conocer el proceso a desarrollarse dejando la documentación correspondiente para desarrollar un nuevo módulo que se acople perfectamente al sistema desarrollado.

- El uso de herramientas libres permitió desarrollar el sistema con todas las funcionalidades requeridas, sin embargo, fue necesaria la capacitación en ciertas áreas del desarrollo con herramientas de código abierto.

- Como resultado de las pruebas de estrés aplicadas al software, se concluye que la creación de servicios en el backend permite mantener estables los tiempos de respuesta sobre las peticiones realizadas (GET, POST, HTTP).

- El sistema corresponde efectivamente a las peticiones realizadas en cada acción, de ingreso, búsqueda y modificación, de esta manera se concluye que el software se acopla adecuadamente al diseño del flujo del proceso para el cual fue creado.

- Luego de recibir resultados positivos en las pruebas correspondientes es factible considerar en un futuro la implementación del sistema como un proyecto viable dentro del Hospital General Docente de Calderón.

Recomendaciones

- Obtener información técnica sobre las herramientas de desarrollo a fin de capacitar adecuadamente al equipo de trabajo.
- A pesar de tener conocimientos sobre el desarrollo de software es necesario recurrir a recomendaciones del docente tutor para obtener una guía más acertada sobre ciertos aspectos relacionados a las buenas prácticas de desarrollo.
- Se recomienda automatizar un mayor número de procesos que tengan relación a la asignación de camas, que se acoplen al software desarrollado, y generen intercambio de datos para de esta menara aumentar el alcance del sistema y dar un mejor seguimiento a los procesos del hospital.
- Se recomienda la capacitación de conocimientos asociados a procesos de salud, previo al desarrollo de software hospitalario.
- Se recomienda establecer una comunicación constante con el personal a cargo del proceso a automatizar para apegarse a los lineamientos del mismo.

Glosario de términos

SI: Sistema de Información

MVC: Modelo Vista Controlador

SISCAM: Sistema de Camas

DSL: Domain Specific Language

JWT: JSON Web Token.

HTTP: Protocolo de transferencia de hipertexto.

HTTPS: Protocolo seguro de transferencia de hipertexto.

Kb: Kilobyte, unidad de almacenamiento.

HTML: HyperText Markup Language

IDE: Integrated Development Environment

API: Application Programming Interface

REST: Representational State Transfer

CU: Caso de Uso

HU: Historia de Usuario

Lista de referencias

Artículos académicos

Suarez, E. (2008). *¿Qué es una base de datos relacional?*

Sitios web

ACADEMIC. (s.f.). Obtenido de https://esacademic.com/dic.nsf/eswiki/151563

Álvarez Caules, C. (1 de 04 de 2015). *ARQUITECTURAJAVA*. Obtenido de https://www.arquitecturajava.com/que-es-gradle/

ATLASSIAN. (s.f.). *ATLASSIAN*. Obtenido de ATLASSIAN: https://www.atlassian.com/es/git/tutorials/what-is-git

Bascón Pantoja, E. (12 de 2004). *DOCPLAYER.* Obtenido de https://docplayer.es/53405274-El-patron-de-diseno-modelo-vista-controlador-mvc-y-su-implementacion-en-java-swing.html

BBVAOPEN4U. (23 de marzo de 2016). *BBVA API_MARKET*. Obtenido de BBVA API_MARKET: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

Blancarte, O. (10 de 12 de 2018). *Oscar Blancarte Software Architect*. Obtenido de Oscar Blancarte Software Architect: https://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/

Greiner, C., Acosta, J. C., Dapozo, G., & Estayno, M. (octubre de 2012). *REPOSITORIO INSTITUCIONAL DE LA UNLP.* Obtenido de REPOSITORIO INSTITUCIONAL DE LA UNLP: http://sedici.unlp.edu.ar/handle/10915/23734

HOSPITAL GENERAL DOCENTE DE CALDERÓN. (07 de 2018). *HOSPITAL GENERAL DOCENTE DE CALDERÓN.* Obtenido de https://www.hgdc.gob.ec/images/DocumentosInstitucionales/Plan%20Estratgico%20HGDC%202018%20-%202022.pdf

Joskowicz, J. (10 de 2 de 2008). *Instituto de Ingeniría Eléctrica.* Obtenido de Instituto de Ingeniría Eléctrica: https://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf

Mafla, S. M. (20 de 2 de 2019). *Repositorio Digital Universidad Técnica del Norte.* Obtenido de Repositorio Digital Universidad Técnica del Norte: http://repositorio.utn.edu.ec/bitstream/123456789/9017/1/04%20ISC%20502%20TRABAJO%20DE%20GRADO.pdf

SPRING. (s.f.). *SPRING*. Obtenido de SPRING: https://spring.io/projects/spring-data

Suarez, E. (2008). *¿Qué es una base de datos relacional?*

TICBLUE. (s.f.). Obtenido de http://ticblue.com/Elit.html

Torres, P. L., & López, E. (12 de 11 de 2013). *Metodologías Ágiles en el Desarrollo de Software.* Obtenido de ISSI: http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf

Vanegas, C. A. (1 de 12 de 2005). *Sistema de Revistas Científicas.* Obtenido de Sistema de Revistas Científicas: https://revistas.udistrital.edu.co/index.php/vinculos/article/view/4066/5728

Anexos

Para revisar los anexos del presente proyecto, por favor diríjase al CD.

Anexo A: Diccionario de datos

Anexo B: Manual de usuario

Anexo C: Manual técnico