Integración de Módulos para la Programación de Quirófanos en las Clínicas y los Hospitales Colombianos con un Módulo Web

Jhonatan España Rodríguez
Alejandro José Marmolejo Salamanca

Universidad Icesi
Facultad de Ingeniería
Ingeniería de Sistemas
Santiago de Cali
2016

Integración de Módulos para la Programación de Quirófanos en las Clínicas y los Hospitales Colombianos con un Módulo Web

Jhonatan España Rodríguez
Alejandro José Marmolejo Salamanca

Documento de Proyecto de Grado

Juan Manuel Reyes García, M.Sc.
Víctor Javier Escallón Santamaría, M.Sc.

Universidad ICESI
Facultad de Ingeniería
Ingeniería de Sistemas
Santiago de Cali
2016

Tabla de contenido

Tabla de ilustraciones

Listado de Anexos

Anexo 1. Esquema de la base de la base de datos

Anexo 2. Diagrama de Deployment del Sistema

Anexo 3. Documento de Casos de Uso del Sistema

Anexo 4. Manual de Usuario

Anexo 5. Manual de Despliegue

Anexo 6. Documento de Requerimientos del Sistema

1. Motivación y Antecedentes

La planificación es el proceso de organizar tareas, controlando y optimizando recursos para lograr un objetivo (Pinedo, 2016). En el área de ingeniería, la planificación tiene un gran impacto a la hora de aumentar la eficiencia, minimizando los tiempos y costos de los servicios.

El sector hospitalario, al igual que muchos sectores productivos, se ha visto obligado a buscar la manera de aprovechar más eficientemente sus recursos más costosos. Los quirófanos, en específico, son uno de los recursos más costosos en las clínicas y hospitales, debido al problema que presenta la planeación eficiente de las cirugías electivas. Los principales problemas que presenta la planeación de los quirófanos son: el seguimiento inadecuado de los criterios de planificación de los quirófanos establecidos por el hospital, causando un mal uso de recursos y tiempo; los hospitales no tienen definidos unos criterios de programación de quirófanos que sean efectivos y que permitan una buena utilización de los recursos; la disponibilidad de los quirófanos no es usada adecuadamente, dejando tiempos muertos donde el quirófano no tiene ninguna cirugía planeada, y no hay una adecuada toma de decisiones acerca de la programación de las cirugías en los diferentes horarios de disponibilidad.

De acuerdo con la investigación doctoral del profesor Víctor Escallón del departamento de Ingeniería Industrial de la Universidad Icesi, actualmente, la planificación de las salas de quirófanos se hace con técnicas empíricas, estas son implementadas por las enfermeras de las clínicas con base en los criterios de dichas técnicas y la experiencia obtenida por la realización de este proceso. El uso de estas técnicas por parte de las enfermeras puede generar planes ineficientes, dando como resultado pérdidas de recursos, cancelaciones, aplazamiento de cirugías electivas y tiempos muertos en los quirófanos. Además de las técnicas que usan las clínicas del país para hacer las planeaciones, existen estudios de modelos y criterios que permiten lograr una mejor eficiencia en la planeación, sin embargo, estas planeaciones siguen siendo realizadas por personas que en algunas ocasiones no implementan de forma correcta y adecuada dichas técnicas, lo cual genera planeaciones ineficientes, concluyendo que estas personas requieren un apoyo sistematizado para mejorar el proceso de toma de decisiones relacionadas con la programación de cirugías.

2. Identificación del Problema

Como parte de la investigación del profesor Víctor Escallón en su tesis doctoral “Metodologías para la Programación de cirugías electivas en las IPS de alta complejidad”, se está desarrollando una estrategia que permite generar un conjunto de planeaciones de quirófanos dentro de las clínicas que atienden cirugías electivas en Colombia, estas planeaciones son caracterizadas por un manejo eficiente en los tiempos disponibles de los quirófanos y en los recursos con los que cuentan las clínicas. La generación de las planeaciones es posible gracias a los módulos desarrollados dentro de la investigación, que inician su ejecución con la entrada de la información referente a las solicitudes de cirugías y los recursos disponibles de la clínica. Como resultado, la estrategia sugiere un conjunto de planeaciones acompañadas por un conjunto de indicadores que sirven para establecer cuál de todas las planeaciones es la más óptima y por tanto la que se usará. La definición de dichos indicadores es parte de otro estudio dentro de la investigación doctoral del profesor.

En la investigación se han propuesto cinco módulos necesarios para poder implementar la estrategia planteada. Para que cada módulo funcione es necesario tomar como entrada la salida de otro de los módulos dentro de la estrategia. Debido a que no hay una comunicación directa entre los módulos, es imposible que estos funcionen como un único sistema, obligando al usuario final a transportar manualmente los datos de un módulo a otro. Además, el formato de salida de los módulos no concuerda con el formato de entrada del módulo siguiente, por lo tanto, la persona que utiliza los diferentes módulos de la estrategia debe hacer la conversión del formato de las salidas de un módulo al formato de las entradas de otro módulo e ingresar manualmente las entradas a cada módulo.

Los módulos presentados en la estrategia son:

- Base de datos, lugar donde se almacena toda la información relacionada con las cirugías, pacientes y recursos de la clínica.
- Módulo de modelo matemático, encargado de generar planeaciones de los quirófanos con la información de la Base de datos. En un principio este módulo estaba diseñado para ser implementado en el lenguaje de programación AMPL, luego de investigaciones se descubrió que el paquete de software IBM ILOG CPLEX combinado con el lenguaje de programación C++, permite relajar las restricciones y dar cierta flexibilidad en un modelo complejo.
- Módulo de simulación, encargado de simular las planeaciones generadas por el módulo matemático, este módulo puede ser implementado en sistemas de software dedicados a la simulación por eventos discretos como lo son: Promodel, Flexsim o Arena. Debido a la incapacidad de obtener la licencia de dichos sistemas de software, este módulo se implementó en C++.
- Módulo de indicadores, encargado de calcular y comparar con indicadores desarrollados dentro de la investigación los resultados obtenidos en el módulo de simulación. Este módulo está implementado en C++.

- Módulo heurístico, encargado de decidir el flujo de información durante el proceso de planeación de quirófanos. Además, basándose en los resultados del módulo de indicadores es capaz de decidir cuál es la planeación más eficiente. Este módulo está implementado en C++.

3. Definición del problema

La estrategia para generar la planeación de quirófanos carece de integración entre los diferentes módulos que la conforman, esto hace que la usabilidad y operatividad de la estrategia desarrollada sea muy baja.

4. Objetivos

4.1. Objetivo General

Desarrollar un sistema de software que sea usable y operativo para el usuario final, que integre los módulos de software requeridos para generar y simular las planeaciones de los quirófanos.

4.2. Objetivos Específicos

1. Analizar los sistemas matemáticos y de simulación que se requieren integrar para la generación, validación y visualización de las planeaciones de los quirófanos.
2. Diseñar e implementar una arquitectura que permita la integración de los diferentes sistemas de software necesarios para la elaboración de la aplicación.
3. Diseñar e implementar interfaces que permitan la comunicación entre los diferentes módulos de software.
4. Diseñar e implementar tres módulos: el primero para la administración de la base de datos, el segundo para el manejo de los indicadores utilizados en la estrategia de planeación y el último que permita la visualización de los resultados obtenidos tras el proceso de planeación de quirófanos.

5. Marco Teórico

5.1. Estilos Arquitectónicos

Según M. Shaw y D. Garlan (Garlan & Shaw, 1993), un estilo arquitectónico “define una familia de sistemas en términos de un modelo de organización estructural; un vocabulario de componentes y conectores, con limitaciones en la forma en que se pueden combinar”, es decir que los estilos arquitectónicos describen los componentes, su funcionamiento en tiempo de ejecución, control y transferencia de datos. Por estas características, se decidió desarrollar la arquitectura del sistema que se quiere desarrollar usando los siguientes estilos arquitectónicos:

5.1.1. Pipes & Filters

Es un estilo arquitectónico cuyos componentes y relaciones son filtros y tuberías respectivamente. En este estilo cada componente tiene un conjunto de entradas y salidas, que viajan a través de un flujo de datos llamado tubería. Este flujo de datos es leído y procesado por un componente llamado filtro. Cada filtro es una entidad independiente, que no comparte sus estados con las demás entidades, pero para poder enviar los datos por la tubería, los filtros adyacentes tienen que manejar el mismo formato de flujo de datos.

Como se puede apreciar en la imagen, en este estilo arquitectónico, el flujo de datos (tuberías) se direcciona hacia un solo lado, limitando que cada filtro solo espere entradas por un grupo de tuberías y envíe las salidas por otro grupo de tuberías.

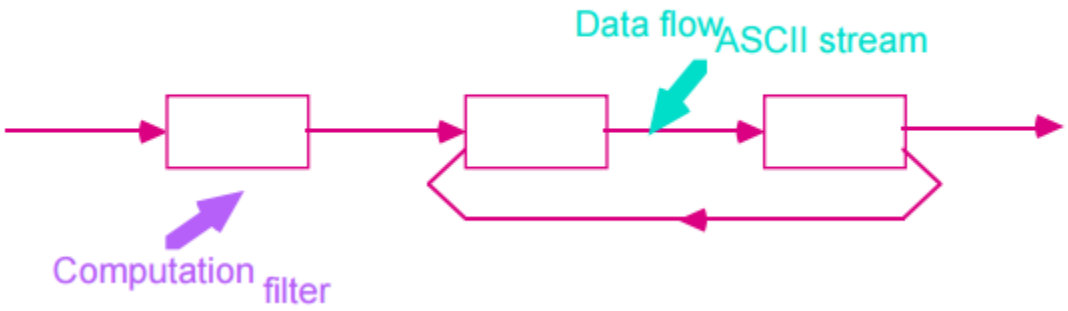


Ilustración 1: Pipes and Filters

5.1.2. REST

Representational State Transfer es el estilo de arquitectura de software del World Wide Web. Da un conjunto de restricciones coordinadas para diseñar componentes en un sistema distribuido de hipermedia que puede llevar una arquitectura de alta mantenibilidad y desempeño. Generalmente es usado para describir interfaces entre sistemas que utilicen el protocolo HTTP, permitiendo la transferencia de datos entre dichos sistemas.

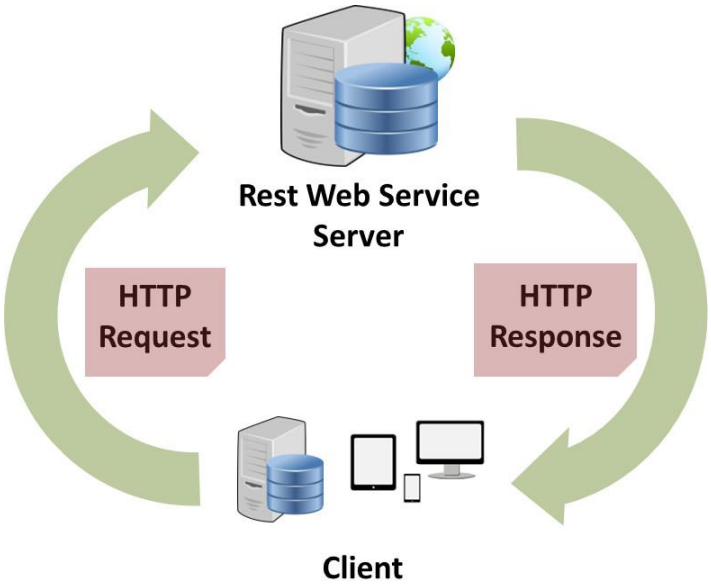


Ilustración 2: REST

5.1.3. Layers

El patrón de arquitectura por capas se utiliza para dividir sistemas de software complejos, en diferentes capas donde la superior consume servicios que la capa inmediatamente inferior ofrece, sin que la última tenga conocimiento de su capa superior. Además, cada capa oculta a las capas superiores, sus capas inferiores.

Algunas de las ventajas de utilizar un patrón arquitectónico de capas son: se inicia desde lo más abstracto hasta lo más complejo, reduce los niveles de complejidad, soporte de reutilización, posibilitan la estandarización de servicios, entre otros.

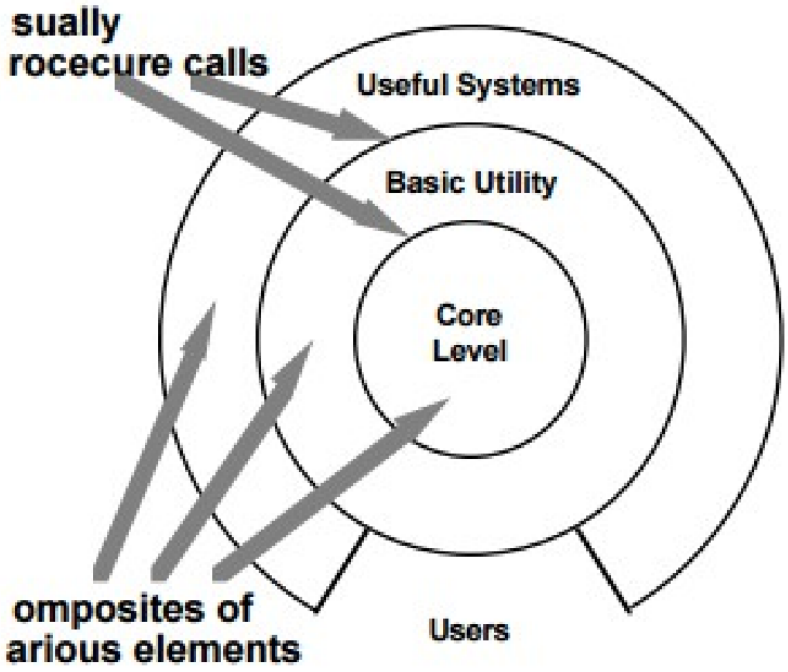


Ilustración 3: Layers

5.2. Protocolos y formatos de comunicación

Los protocolos de comunicación definen un conjunto de reglas y formatos que permiten la comunicación e intercambio de información entre dos entidades o sistemas. Estos protocolos definen por medio de estándares la sintaxis, semántica y sincronización de la comunicación entre las entidades.

5.2.1. SOAP (Monson-haefel, 2004)

Es un protocolo estándar que proporciona un mecanismo simple y ligero entre dos objetos, que están desarrollando diferentes procesos, para comunicarse entre sí por medio de un intercambio de datos XML. La información en SOAP es definida mediante un modelo de empaquetado de datos modular y un mecanismo de codificación de datos. La comunicación por medio de SOAP es posible a través del protocolo HTTP, usado para la transmisión de datos, y XML, usado para la codificación de los datos. La característica principal del porqué SOAP es tan útil, está en la forma que utiliza para comunicar diferentes aplicaciones que están corriendo en diferentes sistemas operativos, con diferentes tecnologías y en diferentes lenguajes de programación.

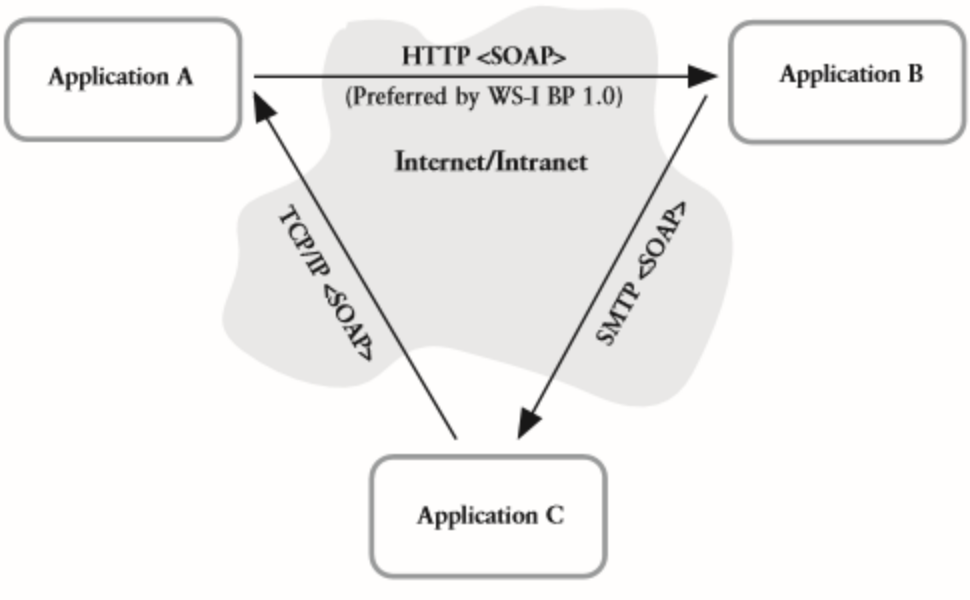


Ilustración 4: Protocolo SOAP

5.2.2. HTTP (Fielding, Irvine, & Gettys, 1999)

HTTP “Hypertext Transfer Protocol” es un protocolo perteneciente a la capa de aplicación del modelo OSI. Este protocolo es altamente usado en cada transacción de la WWW (World Wide Web) debido a que define la sintaxis y la semántica que utilizan los elementos de software dentro de casi todas las arquitecturas web. Al ser un protocolo orientado a transacciones, el flujo de datos y control del protocolo se basa en una petición hacia un servidor HTTP y una respuesta al usuario generada por el servidor. La información que es transmitida a través de este protocolo se le llama recurso y es localizada por medio de una URL (Uniform Resource Locator).

La petición HTTP está compuesto por un conjunto de métodos y encabezados que permiten identificar su propósito. HTTP tiene 8 métodos definidos, los cuales son acciones que se llevan a cabo sobre un recurso específico, los métodos son: HEAD, recuperación de meta-información; GET, recuperación de una representación de un recurso específico; POST, envío de datos para ser procesados; PUT, carga un recurso específico; DELETE, borra un recurso específico; TRACE; reenvío en el cuerpo de respuesta todos los datos que se enviaron en la solicitud; OPTIONS; recuperación de métodos HTTP que el servidor soporta y CONNECT, comprobar si se tiene acceso a un host.

HTTP es un protocolo sin estado, una gran desventaja en el desarrollo de aplicaciones web, donde a veces es necesario el conocimiento del estado de información de conexiones anteriores, por ello se usan las cookies para el manejo de información del cliente.

5.2.3. JSON (Crockford, 2013)

JavaScript Object Notation (JSON) es un formato de texto plano diseñado para el intercambio de datos de una forma entendible para el humano. JSON se caracteriza por ser un formato liviano debido a su fácil declaración de objetos en comparación a XML que usa etiquetas para declararlos. La declaración de objetos en JSON es derivada de un subconjunto del lenguaje JavaScript (definición de literales de objetos), la cual puede representar objetos de tipo primitivo (cadenas, números, booleanos y nulos) y estructurado (objetos y arreglos).

Debido al fácil uso de JSON por parte de los desarrolladores Web, se ha evidenciado un notable crecimiento en el uso de este formato, a veces posicionándose sobre XML. Es por eso que en el momento existen una cantidad considerable de lenguajes de programación que acepta el uso de JSON como por ejemplo: ActionScript, C, C++, C#, ColdFusion, Common Lisp, Delphi, E, Eiffel, Java, JavaScript, ML, Objective-C, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Lua y Visual FoxPro.

5.2.4. WSDL

Es un formato XML hecho para describir servicios web; describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Su principal ventaja es, además de eliminar ambigüedad, es un estándar hecho en XML y puede ser leído tanto por máquinas como por humanos.

5.2.5. XML (w3schools)

Extensible Markup Language es un lenguaje de marcas desarrollado por la W3C utilizado para transportar y almacenar datos de tal forma que sea leíble tanto por humanos como máquinas.

XML presenta varias ventajas, entre las cuales están: es extensible, lo cual quiere decir que se pueden añadir nuevas marcas sin que se genere complejidad, no es necesario crear un analizador específico para cada versión del lenguaje, su estructura es sencilla de entender y de procesar, tiene una alta flexibilidad para estructurar documentos, entre otros.

5.2.6. HTML (“HTML | MDN,” n.d.)

HyperText Markup Language es el lenguaje básico de casi todo el contenido web. HTML es el lenguaje con el que se escribe la estructura de un documento web. Además, es un estándar internacional con especificaciones que son reguladas por el W3C, comunidad internacional que se encarga de desarrollar estándares abiertos para asegurar el crecimiento a largo plazo de la web, y el WHATWG, comunidad interesada en evolucionar la web. Es considerado un "estándar viviente" y está, técnicamente, siempre bajo construcción. La versión actual de la especificación HTML se conoce como HTML5, reconocida por tener un espacio para gráficos generados en el momento de la visualización de la web.

6. Estado del Arte

6.1. Integrating Heterogeneous Components in Software Supply Chains (Hartmann et al., 2010)

En este estudio se discuten las implicaciones que conlleva la integración de componentes de software heterogéneos, es decir, que no poseen interfaces de combinación compatibles, en softwares de cadenas de suministro. Particularmente, se enfocan en el caso donde los componentes que se quieren integrar provienen de distintos proveedores y la generación de componentes llamados “glue components” que se usarán para dar solución a la incompatibilidad de los componentes que se requiere integrar.

Los *glue components* son componentes que se implementan entre los componentes del sistema de producción que se va a implementar, cuyas interfaces no son compatibles para establecer una comunicación directa entre ellos. Los glue components se encargarán de hacer una manipulación de los datos que entregan las interfaces de los componentes incompatibles, para que el servicio que ofrece aquel componente incompatible, sea consumido sin problema por el otro.

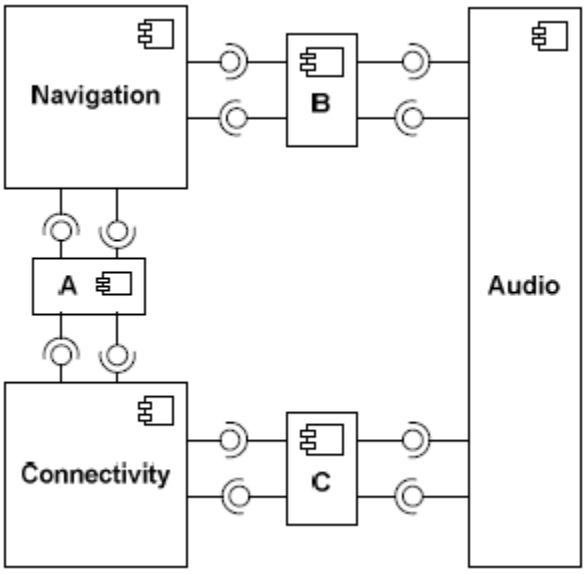


Ilustración 5: Diagrama de componentes de alto nivel de información y entretenimiento

En la gráfica presentada arriba, se quieren integrar un componente de navegación, uno de audio y uno de conectividad. Para el ejemplo, estos componentes son de distintos proveedores y no son compatibles entre ellos, por lo cual, es necesario integrarlos usando los glue components A, B y C.

Esta solución aplica parcialmente a nuestro proyecto, ya que sucede lo mismo en los módulos que son necesarios integrar en nuestra solución; son módulos de diferentes proveedores que no son compatibles. El problema en esta solución es que aplica para componentes, mientras

que en nuestro problema, son sistemas completos de software que se necesitan integrar y no se posee en este momento el conocimiento para saber si los sistemas a integrar poseen componentes que ofrecen servicios para integración con otros sistemas, además de que nuestra solución está enfocada a la programación de citas y no de cadenas de suministro.

6.2. An Evaluation of Hospital Information Systems Integration Approaches (Sabooniha, Toohey, & Lee, 2012)

Las organizaciones de la salud para proveer un servicio de alta calidad, se les hace obligatorio la manipulación de altos volúmenes de información. Bajo esta preocupación, la integración de los sistemas de información del área de la salud es necesaria, ya que estos sistemas pueden estar distribuidos o ser heterogéneos. Bajo este requerimiento, se realiza un estudio de los enfoques bajo los cuales la integración de HIS (Hospital Information Systems) se podría realizar.

Los enfoques estudiados son cuatro: Message-Oriented Integration, Application-Oriented Integration, Coordinated-Oriented Integration y Middleware-Oriented Integration.

Message-Oriented Integration se basa en un conjunto de mensajes estándar cuyo objetivo principal es permitir que varios subsistemas del HIS intercambien información. Este enfoque usa bases de datos, APIs e intercambios de datos para producir información.

Application-Oriented Integration suministra una capa de aplicaciones definidas y centralizadas, encima de las aplicaciones existentes con el fin de apoyar el flujo e intercambio de información y de control lógico entre ellas combinando aplicaciones relevantes y procesos.

Coordinated-Oriented Integration proporciona una visión consistente de la información contenida en los diferentes sistemas, aplicaciones y servicios subyacentes para los usuarios. Se puede suministrar mediante el uso de un sistema de front-end unificado o mediante la sincronización y la coordinación de los diversos sistemas o aplicaciones en la estación de trabajo del usuario.

Middleware-Oriented Integration define un conjunto de servicios, interfaces o métodos compartidos que soportan todo el sistema. Este enfoque proporciona la infraestructura para el intercambio de servicios funcionales y de información.

Tras analizar los enfoques en los que pueden ser integrados los sistemas, luego se pasa a identificar y comparar los distintos aspectos de las distintas soluciones que pueden ser útiles al seleccionar la solución de integración. Al haber una gran variedad de sistemas a integrar, es claro que un solo enfoque de los estudiados no es capaz de cubrir todos los requerimientos por los que los distintos sistemas se van a integrar, por lo cual, se deja la escogencia del enfoque a dependencia de los requerimientos de los sistemas a integrar.

Esta solución puede ayudar, más no aplica a nuestro problema, ya que dependiendo de los requerimientos de integración de los sistemas que debemos integrar, podemos escoger o definir enfoques que se acomoden a nuestras necesidades a la hora de integrar los módulos. No aplica porque el estudio realizado es hacia sistemas de manejo y despliegue de información, y en nuestro problema tratamos con simulaciones y evaluaciones de alternativas de las planeaciones de quirófanos resultantes.

6.3. DIMP: an interoperable solution for software integration and product data exchange (Wang & Xu, 2012)

Distributed Interoperable Manufacturing Platform (DIMP) es un ambiente de colaboración para la integración de los módulos heterogéneos (implementación hecha en otros lenguajes y con otros protocolos de intercambio de información) (Computer-aidded design, CAD; Computer-aided manufacturing, CAM y Computer numerical control, CNC). Este ambiente busca lograr que el sistema de la integración de todos los módulos sea interoperable, visible, portable y longevo. Para esto se construyó la solución implementando la arquitectura orientada a servicios (SOA) y STEP-NC como formato en la definición de datos, el cual facilita el intercambio de éstos entre los módulos.

Existen dos categorías de métodos para lograr interoperabilidad e intercambio de datos entre diferentes módulos de software, las cuales son enfoque centrado a los datos y enfoque centrado a los procesos. Para lograr cumplir los requerimientos de calidad del sistema se implementaron varios métodos ubicados en las dos categorías, entre ellos están: arquitectura orientada a tareas (perteneciente a los enfoques centrados en el proceso) y modelo de datos con formato neutral (perteneciente a los enfoques centrados en los datos).

Basados en los anteriores métodos escogidos para la implementación del sistema, se diseñó una arquitectura de tres capas, que son: la primera es una capa supervisora, la cual es la encargada de recibir las solicitudes de los usuarios, tomar decisiones y enviar los datos a las otras capas; la segunda es una capa de datos, la cual es la encarga de almacenar los datos referentes a la salida, el proceso del sistema y la asignación de tareas por cada módulo (en la imagen se identifican como Application); la última es una capa de almacenamiento de aplicaciones, cuya función es contener todos los módulos funcionales del sistema, los servicios son expuestos a través de un componente virtual de servicios, que se encarga de proveer la respuesta a la solicitud del usuario. En la siguiente figura se muestra la arquitectura anteriormente descrita.

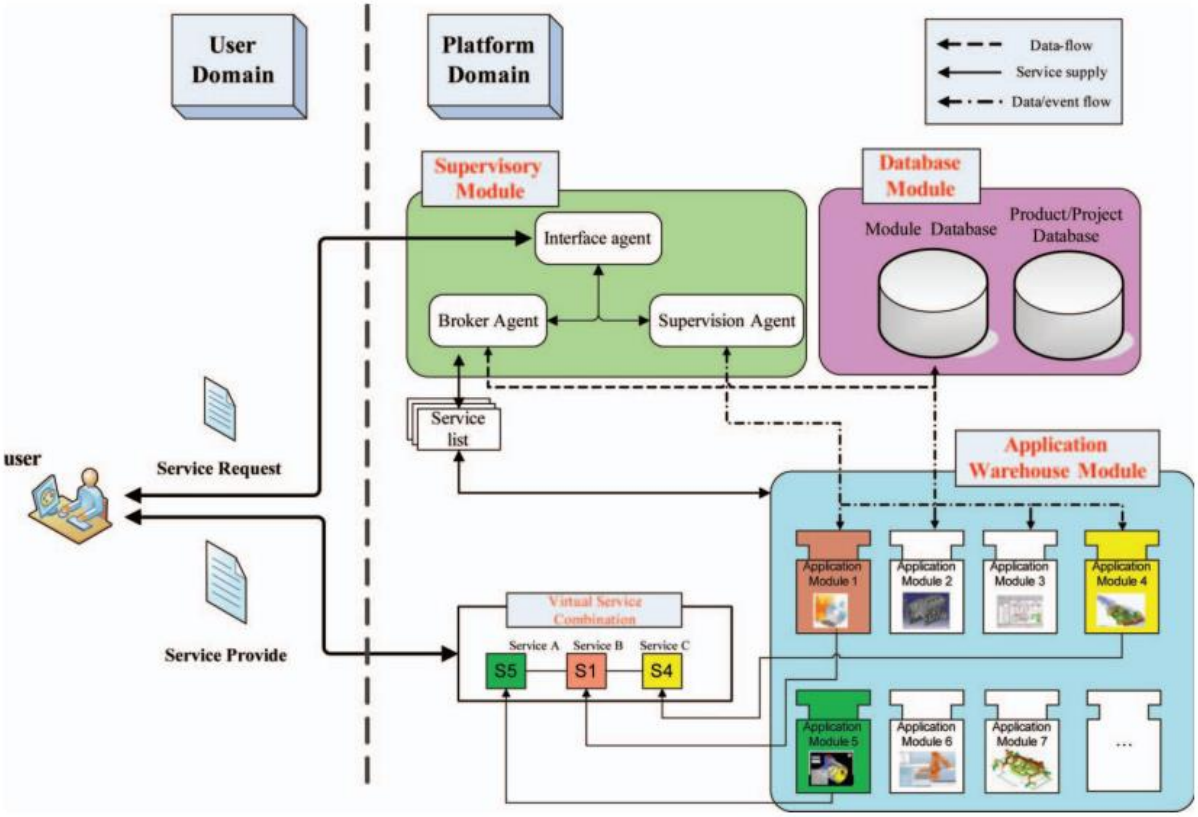


Ilustración 6: Arquitectura DIMP orientada a servicios

La solución planteada en este caso no aplica del todo al problema, dado que los formatos que utilizan para el intercambio de datos son neutrales, es decir que se ha escogido un formato y todos los módulos deberán de generar salidas y recibir entradas con el mismo formato.

6.4. Integrating GI with non-GI services (Treiblmayr, Scheider, Krüger, & von der Linden, 2012)

Los sistemas de información geográfica (GIS) son implementados en diferentes plataformas, lenguajes y formato de datos que otros sistemas que no son GI, por lo tanto, el intercambio de datos con otros sistemas es complicado. Los sistemas GIS sirven para analizar datos geográficos para la evaluación de posibles riesgos geológicos. Por ello, estos sistemas necesitan comunicarse e intercambiar datos con los sistemas que los recolectan y reciben.

Para facilitar la implementación de la integración entres sistemas GIS y no- GIS, la solución plantea el uso de la arquitectura orientada a servicios (SOA), a través de la exposición de Web Services que contiene la información del sistema que los expuso. Los estándares que permiten la comunicación y la exposición de estos servicios son Simple Object Access Protocol (SOAP) e Hypertext Transfer Protocol (HTTP) definidos anteriormente. Los servicios son expuestos en interfaces. Éstos son especificados por Web Service Description Language (WSDL), que está basado en extensible markup language (XML).

El diseño de la arquitectura planteado por la solución se compone por tres nodos de procesamiento, que son: Enterprise Resource Planning (ERP) encargados de la resolución de alertas geográficas y la recolección de datos para el sistema GIS; NetWeaver Java Enterprise Edition (NW JEE) que es el proveedor de la arquitectura orientada a los servicios y sirve como intermediario entre los sistemas GIS y no-GIS, y por último está el servidor ArcGIS que es el encargado de almacenar los servicios GIS. A continuación, se presenta el diagrama de los componentes de la arquitectura correspondiente a la solución.

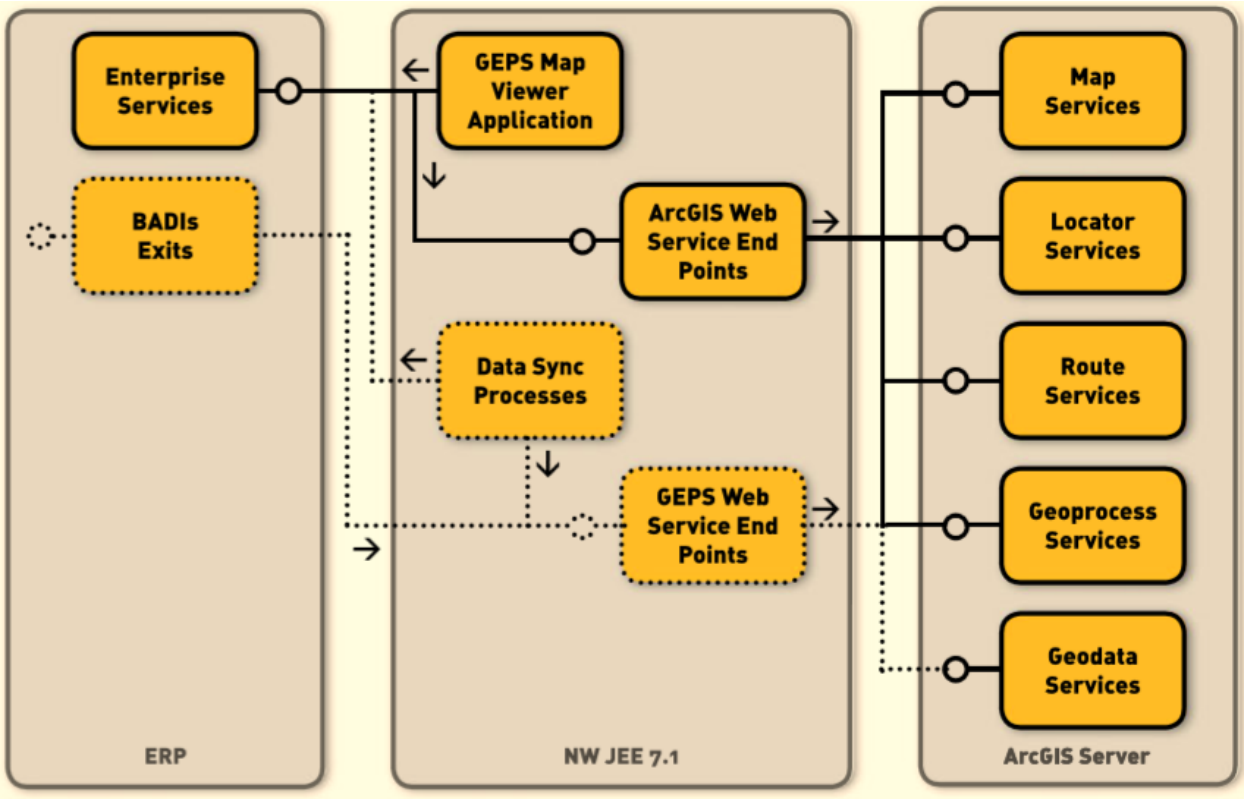


Ilustración 7: Vista estructural de la integración de GIS y un sistema de planeación de recursos empresarial en la arquitectura orientada a servicios.

La solución debe de ser escalable, dado que hay varias instancias de los nodos de procesamiento ERP que se conectan al servidor NW JEE. Además, en sistemas de esta categoría existen muchos tipos de solicitud y es ahí donde se presenta el principal problema al implementar esta arquitectura, dado que un componente sin una implementación adecuada no sabe el tipo de solicitud o de respuesta que llega (no sabe el formato de los datos que llega, ni a qué pertenecen estos datos).

Esta solución no aplica al proyecto dado que buscan integrar sistemas heterogéneos escalables, y cada componente maneja varios tipos de solicitud y respuesta. Por el contrario, cada uno de los módulos asociados a este proyecto solo tienen un tipo de solicitud de datos y un tipo de salida. Además, los módulos no tendrán más de una instancia.

6.5. Comparación entre los elementos del estado del arte

IHCSSC, Integrating Heterogeneous Components in Software Supply Chains
EHISIA, An Evaluation of Hospital Information Systems Integration Approaches
DIMP, Distributed Interoperable Manufacturing Platform
IGWNS, Integrating GI with non-GI services
SSPQ, Sistema de Software para la Planeación de Quirófanos

	IHCSSC	EHISIA	DIMP	IGWNS	SSPQ
Rendimiento					•
Interoperabilidad		•	•	•	•
Operatividad		•			•
Mantenibilidad	•				•
Escalabilidad	•			•	
Portable			•	•	

Ilustración 8: Comparación entre los elementos del estado del arte por atributos de calidad.

7. Desarrollo de la Integración Propuesta

Para integrar los módulos implementados por el profesor Víctor Escallón en su tesis doctoral "Metodologías para la Programación de cirugías electivas en las IPS de alta complejidad", con una interfaz web y una base de datos que pudiera almacenar los resultados de las programaciones de cirugías electivas en los quirófanos de las clínicas colombianas, se propuso una arquitectura y un modelo de datos que estuviera acorde con los requerimientos necesarios para el proyecto.

La arquitectura está construida teniendo en cuenta el tipo de tecnologías que se usa para implementar cada uno de los módulos que buscan la mejor programación de cirugías en un tiempo determinado. Los módulos matemáticos, heurístico, de simulación y de indicadores están construidos en C++, por el motivo de que en este lenguaje existen librerías que permiten la fácil implementación y ejecución de los modelos matemáticos que son la base para hallar un conjunto de programaciones eficientes, según los indicadores analizados durante el desarrollo de la tesis doctoral. Los módulos web, base de datos y los servicios web están implementados en Java y HTML, utilizando un framework llamado Play Framework, que utiliza el patrón de diseño MVC.

Basados en los lenguajes y las tecnologías que se utilizaron en la implementación del sistema en general, se decidió separar la arquitectura en dos servidores y un ejecutable de Windows: un servidor que contiene el motor de base de datos MySql; un servidor que aloja las vistas, controladores y el modelo construido en Java, este es propio del framework que se usa, y un ejecutable de Windows que contiene las implementaciones en C++, como lo son los archivos que contienen los modelos matemáticos y las funciones que permiten hallar un conjunto de programaciones eficientes. Esta arquitectura se ve reflejada en el diagrama de deployment (Anexo 2).

La comunicación entre los servidores de base de datos y el encargado del modelo-vista-control es a través de peticiones al motor de base de datos, que realiza el EntityManager del framework de Java que se está utilizando, esta es la manera con la que se obtienen los datos. Por otro lado, el ejecutable de C++ intercambia datos con el servidor de aplicaciones Java a través de archivos json almacenados dentro este mismo servidor.

El módulo web junto con el módulo de base de datos se encarga de tomar toda la información necesaria para iniciar el proceso de programación de cirugías, una vez el módulo haya definido dicha información, la almacena en un archivo en formato JSON dentro del servidor de aplicaciones de Java e inicia el ejecutable que contiene la implementación del proceso de programación. De la misma forma, el ejecutable crea un archivo en json con la información de los resultados de la programación y lo almacena en un directorio del servidor de aplicaciones de Java, para que así, una vez terminado el proceso, el módulo de base de datos y web pueda leer este archivo y almacenar la información en la base de datos para que pueda ser mostrada por medio del sistema.

Respecto a la base de datos que usa el sistema, se diseñó un esquema de base de datos en segunda forma normal sin redundancias y que admite el acceso rápido a la información solicitada. Este esquema permite almacenar la información más relevante e importante que se involucra en la programación, atención y ejecución de las cirugías, como lo son los datos de los pacientes, cirugías, profesionales de la salud, recursos, disponibilidades, camas, utilería, quirófanos, procedimientos, etc…

A continuación, se puede observar el diagrama de deployment que refleja la arquitectura propuesta del sistema, como anteriormente se había dicho, contará con 2 servidores y un ejecutable, cada uno representado por un nodo de procesamiento, que se comunican entre sí a través de mensajes HTTP y llamados del sistema.

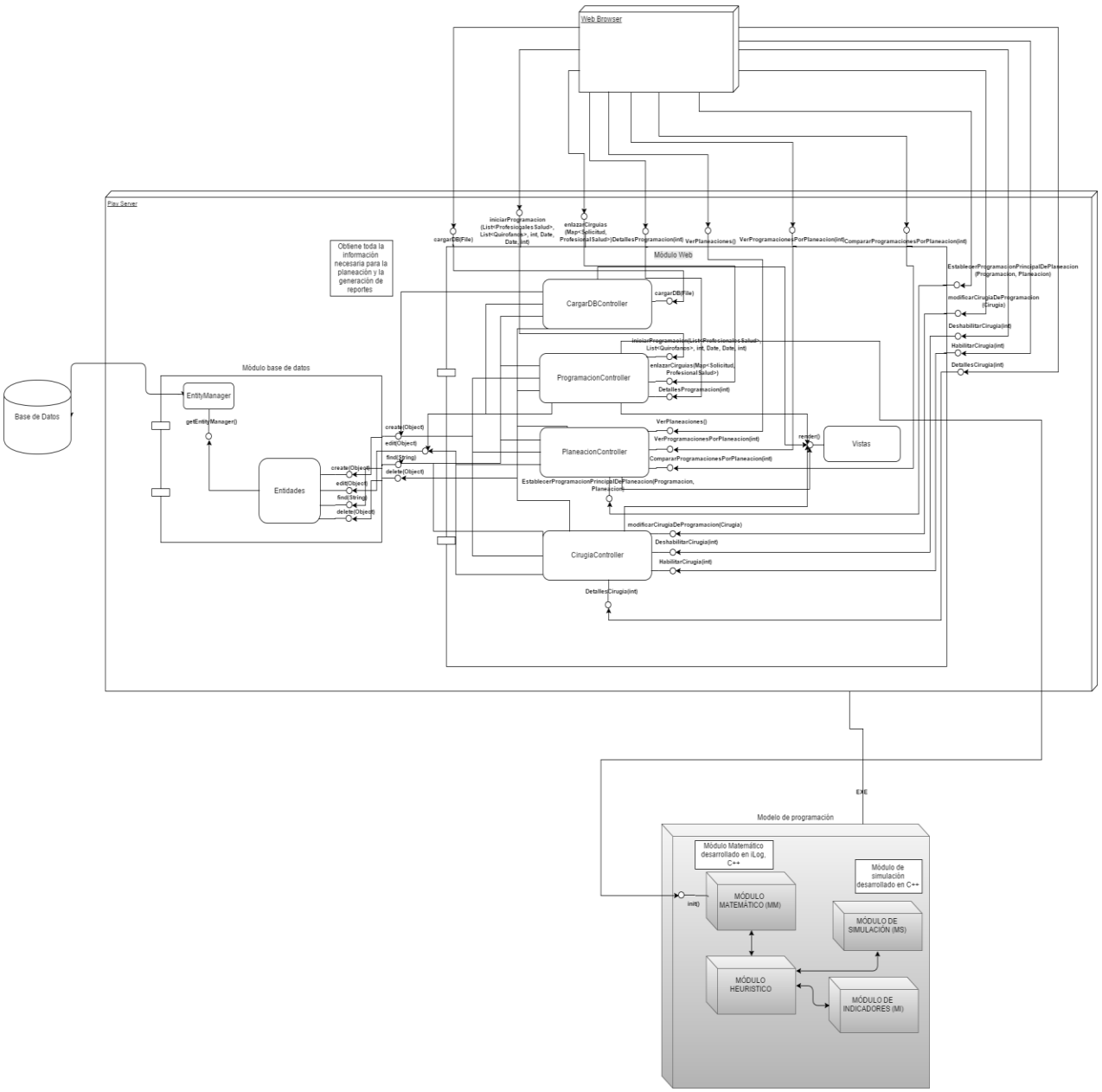


Ilustración 9: Diagrama de deployment de la arquitectura de la solución.

Respecto a la base de datos, primero se planteaba que la aplicación accediera a la base de datos de cada clínica objetivo; esto no era muy práctico, ya que las clínicas puede que no tengan una base de datos o que su modelo de datos cambie de clínica a clínica. Para resolver este problema, se diseñó un esquema de base de datos que contiene los datos necesarios para realizar la planeación utilizando la estrategia propuesta de planeación de cirugías, esto con el fin de manejar un estándar en los datos y que las clínicas tengan la posibilidad de, sin importar que tengan un sistema o no, simplemente carguen la base de datos con su información y estén listos para realizar planeaciones inmediatamente.

Para facilitar la carga de datos al sistema, se ha definido un formato para un archivo de Excel que deberá ser cargado por medio de la interfaz web de la aplicación. Una vez cargado el archivo, la persona encargada de realizar la planeación podrá configurar la información inicial para la ejecución del proceso de planeación, luego de que este proceso haya culminado, se podrá visualizar la información de la planeación a través de una interfaz web amigable y usable que presenta varios tipos de gráficos que diagraman los datos de cada programación, además, esta interfaz permite comparar las programaciones y editar la información de las cirugías que la conforman.

8. Implementación de la solución

El desarrollo de la solución se dividió en cuatro etapas (análisis, diseño, implementación y pruebas). Estas cuatro etapas se llevaron a cabo basados en el modelo incremental del desarrollo de software.

Fase de Análisis

En la fase de análisis se realizaron entrevistas con el tutor con el fin de definir los requerimientos del sistema y recolectar información sobre los componentes de la solución que propone, estas reuniones se llevaron a cabo semanalmente con el profesor Víctor Escallón director del proyecto “Metodologías para la Programación de cirugías electivas en las IPS de alta complejidad”, en las cuales inicialmente se preguntó sobre qué debía y qué no debía hacer la aplicación, se establecieron los formatos con los cuales se van a comunicar los diferentes módulos que componen la solución. Durante este proceso hubo muchos cambios, cada semana se definían diferentes estándares, y se evaluaban diferentes tecnologías para la implementación de los módulos, con el objetivo de cumplir con los requerimientos del sistema.

Como resultado de esta fase se generó el documento de requerimientos del sistema (Anexo 6), el cual contiene los requerimientos funcionales y no funcionales que se definieron durante las entrevistas con el tutor.

Fase de Diseño

En la fase de diseño existen dos resultados significativos, el diseño arquitectónico representado por el diagrama de deployment del sistema, y el esquema de base de datos representado por el modelo entidad - relación. Ambas tareas se realizaron simultáneamente con la ayuda y asesoramiento de los tutores durante las reuniones agendadas.

Para obtener el esquema de base de datos, se tuvo en cuenta el flujo de información que existe entre cada uno de los módulos que conforman la solución propuesta, junto con la información que devuelve como resultados estos módulos, así, se pudo construir un esquema que cumpliera

con los requerimientos y que además permitiera almacenar la información relevante asociada con la programación de cirugías.

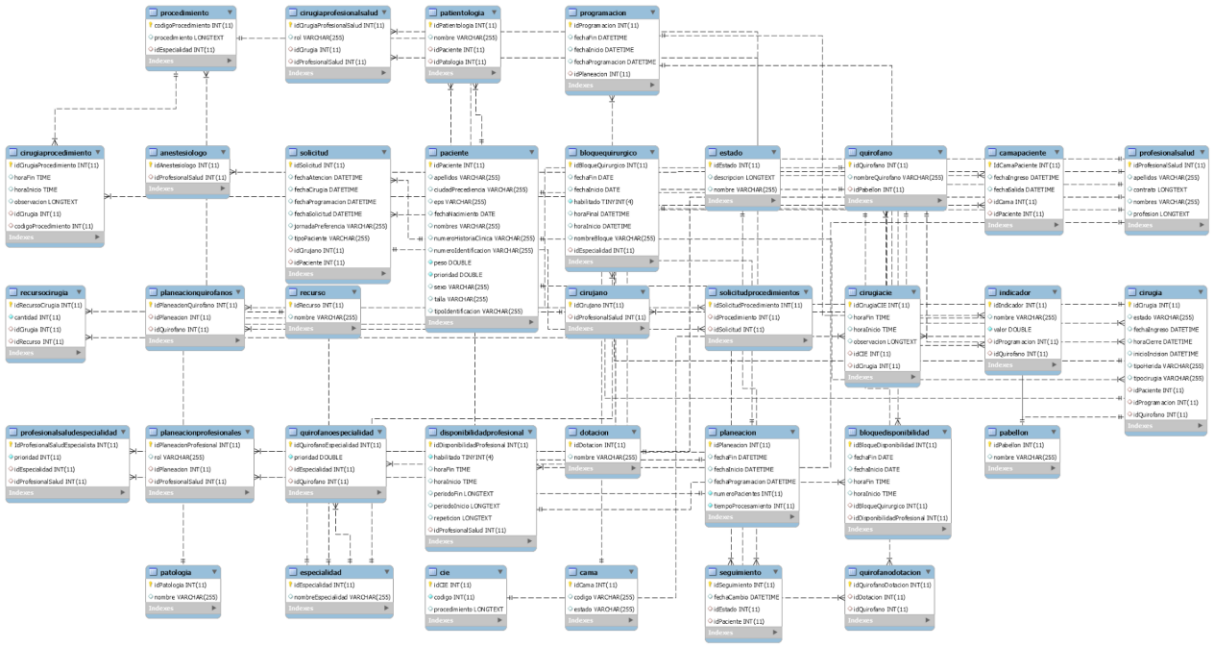


Ilustración 10: Diagrama de la base de datos de la solución.

En cuanto al diseño arquitectónico de todo el sistema, se realizó basado en la información que el tutor entregaba por medio de las reuniones semanales, este diseño fue construido con el objetivo de contribuir al factor de rapidez en la búsqueda de las mejores programaciones en un tiempo definido, y comunicar los módulos implementados en C++ por el tutor con los módulos implementados en Java por el grupo de trabajo.

Hubo varias versiones del diseño arquitectónico que describe el flujo de control y de información del sistema, este diseño variaba mucho debido a que no se había encontrado una tecnología que cumpliera todos los requerimientos que exigía el proyecto en cuanto a los módulos matemáticos, heurístico, de indicadores y de simulación.

La primera versión definía un sistema donde los módulos funcionaban de manera secuencial, donde primero un módulo de datos era el encargado de cargar la información de las cirugías, quirófanos y pacientes; luego un módulo web permitía iniciar el proceso de programación de quirófanos; después un módulo matemático era el encargado de generar todas las programaciones posibles; luego el módulo de simulación era encargado de probar estas programaciones; después el módulo de indicadores era el encargado de calcular los indicadores de cada programación en base a los resultados del módulo anterior; luego el módulo heurístico era el encargado de definir que programaciones cumplían ciertos criterios para ser tenidas en cuenta; por último el módulo web era el encargado de mostrar los resultados que generó el proceso de programación. Esta arquitectura está representada en el siguiente diagrama.

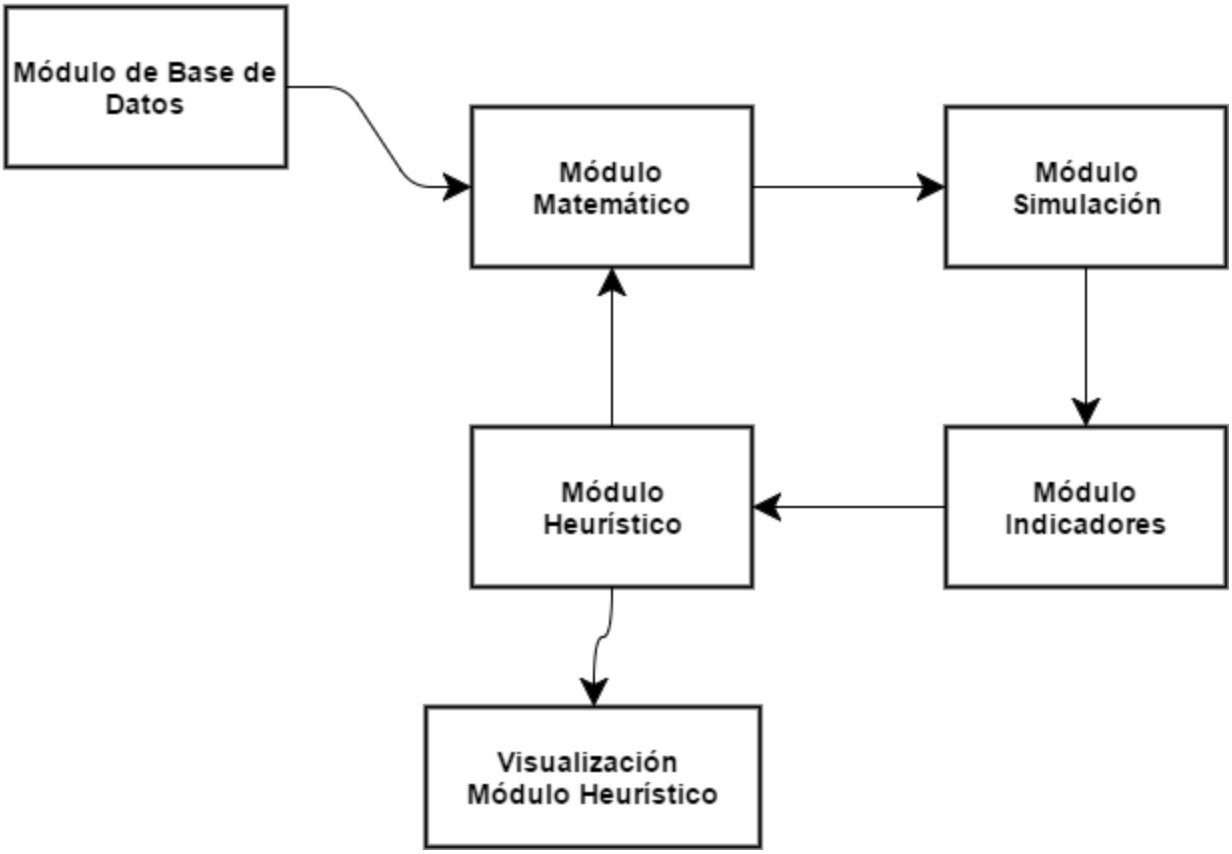


Ilustración 11: Diseño poco detallado de la arquitectura y flujo de la solución.

La segunda versión planteaba un sistema donde los módulos matemáticos, de simulación, de indicadores y heurístico se podían comunicar entre sí, ya que la ejecución de estos módulos era paralela, permitiendo descartar planeaciones durante la ejecución, gracias a reglas definidas en el módulo heurístico. En ese momento, los módulos estaban implementados en diferentes tecnologías; matemático (iLog, C++), simulación (Promodel, Simul8, Flexim, Arena y C++), indicadores (C++) y heurístico (C++ o Java), esto llevó a pensar en la creación de un módulo intermedio que mediante la implementación de las interfaces permitiera comunicar todos los módulos. Esta arquitectura fue descartada debido a dos factores: al alto número de transacciones a través de la red, esto hacía que el sistema se tomará más tiempo en obtener los datos, y el cambio de tecnología que tomo la implementación de estos módulos. A continuación, se muestra la representación gráfica de lo que fue esta arquitectura.

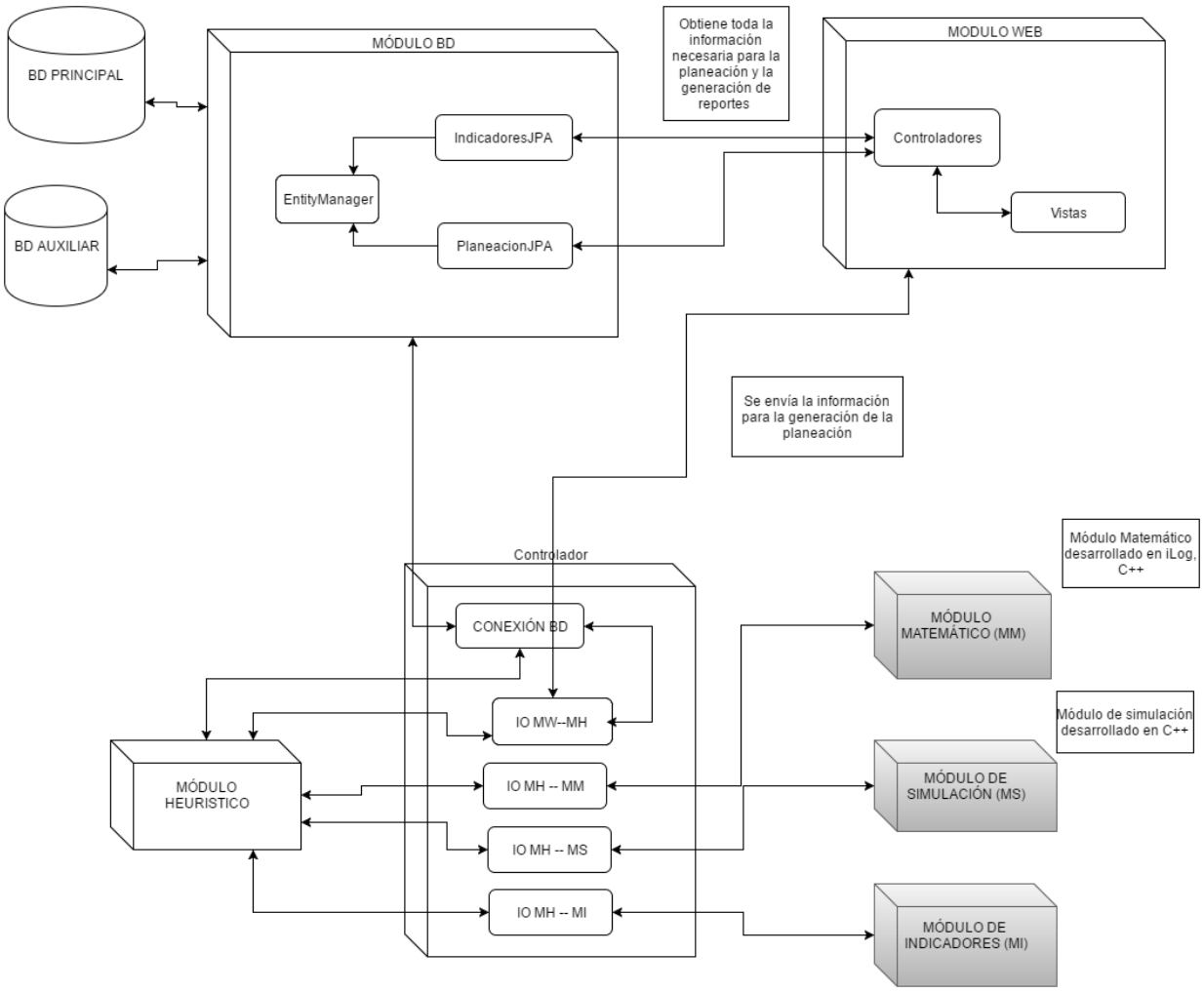


Ilustración 12: Versión 2 de la arquitectura propuesta.

La tercera versión de arquitectura aprovechó que los módulos encargados de la programación se implementarían en un mismo lenguaje de programación (C++), ejecutándose bajo un mismo servidor, el costo y tiempo por cada llamado entre los módulos disminuyó de gran forma. La comunicación con el módulo web y de base de datos se planeó hacerlos a través de servicios web, los cuales son expuestos por el servidor de Java.

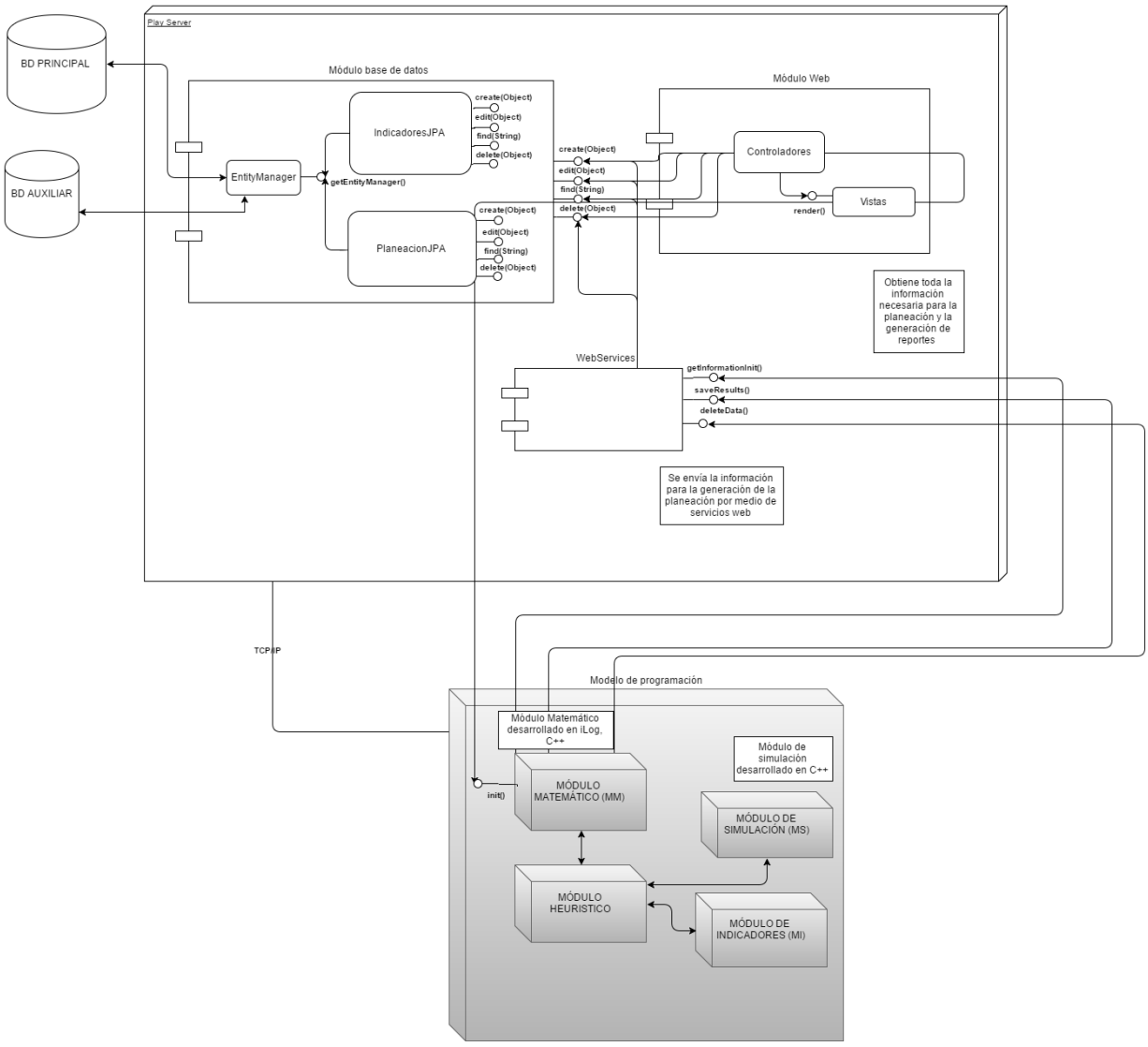


Ilustración 13: Versión 3 de la arquitectura propuesta.

La cuarta versión (*Ilustración 9*) y definitiva plantea una arquitectura muy parecida a la anterior, con la diferencia que la actual aprovecha que la implementación de la lógica de las programaciones se puede iniciar por medio de un ejecutable de Windows, manejando la comunicación de la información por medio de archivos livianos y fáciles de interpretar como lo son los archivos en formato JSON. Con esta solución evitamos el paso de información por medio del protocolo HTTP, ya que el ejecutable puede estar almacenado en el servidor de aplicaciones de Java y ser ejecutado desde uno de los controladores de los módulos escritos en Java. Así, el paso de información entre los módulos se vuelve menos costoso y más rápido.

Las decisiones que se tomaron respecto a la tecnología que se escogió para implementar los módulos fueron tomadas dependiendo de las siguientes comparaciones:

Tabla 1: Comparación entre lenguajes candidatos.

Lenguaje	Conocimiento	Servidores	Frameworks para Web	Licencia	Motores de base de datos
PHP	Muy Bajo	Apache, IIS, Netscape, iPlanet, EasyPHP, etc...	Alta cantidad	Licencia PHP	MySql, SQLite, Postgres, Oracle, etc.…
C# (.NET)	Medio	IIS, WebListener, Kestrel (Linux)	Buena Cantidad	Licencia MIT	SQL server, proveedores OBDC,
Java	Alto	Servidores propios de frameworks, Glassfish, JBoss, Tomcat, etc.…	Alta Cantidad	GNU GPL / Java Community Process	MySql, Postgres, Oracle, SQLite, etc..
C++	Muy Bajo	WebToolKit, Axis C++. etc.…	Poca cantidad	Propietario, EULA	MySql, Postgres, Oracle, SQLite, etc.

Con base en la anterior tabla comparativa entre lenguajes, escogimos Java como el lenguaje utilizado para desarrollar los módulos de base de datos y web. Esta decisión se basa principalmente en la pequeña curva de aprendizaje que existe con esta tecnología, debido a la experiencia del grupo de trabajo, luego otros factores (existencia de servidores que corren en diferentes sistemas operativos, la alta documentación y frameworks que existen para facilitar el desarrollo con esta tecnología, la licencia de software libre y la posibilidad de poder usar varios tipos de motores de base de datos) hacen que sea la opción más conveniente para ser usada.

Tabla 2: Comparación entre frameworks web[1]

	Rápido Desarrollo	Facilidad de Uso	Comunidad y Documentación	Ecosistema del Framework	Rendimiento / Escalabilidad	Mantenimiento del Código / Actualizaciones
Spring MVC	2.5	3	4	4	4	3
Grails	5	4.5	5	4.5	4	4.5
Vaadin	4.5	4.5	5	3	4.5	4
GWT	4	4	4.5	3	4.5	4
Wicket	3.5	3.5	3	3	3	4.5
Play	5	3.5	4	4.5	5	4
Struts	2	3	2.5	3	3	3

[1] The Ultimate Java Web Frameworks Comparison: Spring MVC, Grails, Vaadin, GWT, Wicket, Play, Struts and JSF, comparación de frameworks web de Java dependiendo de sus características.

JSF	3	4	4.5	4	4	4

En la anterior tabla las puntuaciones se hicieron sobre 5, donde los atributos que más destacan al framework que se eligió son: el rápido desarrollo de un prototipo gracias a la buena documentación que existe en el sitio web de Play y a la facilidad para crear nuevos componentes; el ecosistema del framework provee todo lo que se necesita para desarrollar, correr y probar la aplicación, y una excelente escalabilidad y rendimiento trabajando con grandes cantidades de datos, esto lo hace a través de tareas asíncronas. Debido al poco tiempo que se tiene para desarrollar la aplicación, la posibilidad de usar otros frameworks para generar reportes y gráficos, el rendimiento necesario para procesar grandes tamaños de datos y la posibilidad de implementar servicios web de REST se eligió Play Framework.

En cuanto a la tecnología que se usa para la implementación de servicios web, propuesta que estaba activa hasta el tercer diseño arquitectónico, se eligió REST. Esta decisión se tomó teniendo en cuenta los requerimientos del sistema, y los atributos expuestos en la siguiente tabla.

Tabla 3: Comparación entre REST y SOAP

REST		**SOAP**	
Ventajas	**Desventajas**	**Ventajas**	**Desventajas**
Ideal para sistemas distribuidos	Pueden suceder situaciones de desincronización si se programa REST en dos proyectos independientes	Permite la especificación al detalle de la información que se transportará por medio del servicio	Requiere implementaciones especiales para poder ser consumido
Mejora en cuanto a rendimiento, utiliza JSON hace que la lectura de la información sea más rápida	No hay estándares en las respuestas, por lo que no se definen tipo de datos	Aportan interoperabilidad a las aplicaciones independientemente de las tecnologías escogidas	Poca información en la implementación para algunos lenguajes
Uso del protocolo HTTP, no requiere implementaciones especiales para ser consumidos		Respuesta en texto que hacen fácil la lectura de los contenidos	Baja mantenibilidad
Separación de implementación entre el cliente y el servidor		With gSOAP library se pueden consumir servicios SOAP en C++.	No se puede guardar en memoria caché las lecturas de los datos

Se puede llamar el servicio desde C++, con ayuda del sdk casablanca		Es basado en estándares y describe mucha información en sus request and response	
Fiabilidad, escalabilidad, flexibilidad		Es más seguro que REST	
REST requiere menos recursos del servidor			
Lecturas de datos se pueden guardar en la memoria caché			

Fase de Desarrollo

En la fase de desarrollo, se realizaron varias reuniones en las que definimos sobre papel, el funcionamiento de la aplicación web en cuanto a navegabilidad y dividimos la implementación de los diferentes módulos web como se ve en la siguiente tabla:

Tabla 4: División de trabajo.

Jhonatan España	Alejandro Marmolejo
Diseño e implementación de la vista y controlador encargados de la configuración pre-planeación.	Diseño e implementación del layout base de la aplicación web.
Emparejamiento entre solicitudes de cirugías y cirujanos.	Implementación del módulo de carga de datos.
Implementación del módulo que despliega el conjunto de planeaciones realizadas.	Implementación del módulo que permite comparar planeaciones con un diagrama de telaraña usando indicadores.
Implementación de la lectura de los resultados generados por el proceso de planeación.	Implementación de la ejecución del módulo externo encargado del proceso de planeación.
Implementación del módulo de diagrama de gantt que muestra la información de las programaciones realizadas.	

Durante el desarrollo se hizo uso de diferentes librerías y frameworks que permiten ahorrar tiempo y trabajo en la implementación de ciertas funcionalidades que exige el sistema. Se usó la librería Gson[2] para poder manejar los datos en formato JSON como objetos propios de Java.

[2] Librería que permite la conversión de objetos en java a una representación en JSON y viceversa.

En la implementación de los diagramas que muestran la información de las planeaciones, cirugías e indicadores, se usó tres frameworks, el primero es Google Charts[3] que permitió la generación de los diagramas de Gantt y de Barras que representan las cirugías de una programación en una línea de tiempo. El segundo es Chart JS[4] que permitió la implementación de los diagramas que muestran la información de las programaciones con sus respetivos indicadores. Por último, el tercero es DHTMLX Scheduler[5] que permitió la implementación del diagrama de Booking, en el cual se pueden visualizar las cirugías por quirófano desde una vista semanal o mensual.

Respecto a la integración con los módulos encargados de generar las programaciones de las cirugías, no se pudo realizar dicha integración con el módulo desarrollado completamente, debido a que aún está en fase de desarrollo a cargo del profesor Víctor Escallón. Para poder realizar las pruebas con un módulo externo que generara datos, sabiendo que el resultado final de los módulos de programación iba a ser la implementación compilada en un archivo .bat, se creó un archivo .bat que, en base a un archivo generado con resultados de programaciones al llamarse, generara otro archivo con dichos resultados para que el módulo web y de datos pudieran hacer la lectura, inserción y visualización de los resultados.

Fase de Pruebas

En fase de pruebas, principalmente se realizó las actividades de verificación y validación. La verificación se realizó entre el grupo de trabajo, generando datos con ayuda de una herramienta web llamada generatedata[6] y recibiendo un conjunto de datos resultantes de la ejecución de los módulos encargados de la planeación, se pudo obtener la información necesaria para hacer pruebas funcionales y verificar el correcto despliegue de la aplicación web.

La validación se realizó entre el grupo de trabajo encargado de desarrollar el módulo web y los tutores del proyecto, lamentablemente, no hubo la posibilidad de reunirse con una persona perteneciente a la clínica que estuviera relacionado con la programación y realización de cirugías en los quirófanos, pero se consideró que era suficiente con la retroalimentación que brindaba el profesor Víctor Escallón, debido a que él tiene el conocimiento sobre él área médica y sobre él proceso que realizan las personas encargadas de la planeación de las cirugías en la clínicas. Durante el desarrollo de esta actividad se discutía principalmente sobre la funcionalidad del sistema, la manera en que se mostraba la información y la navegabilidad del sistema.

Durante el desarrollo del proyecto se generaron tres versiones del sistema, las cuales eran sometidas a validación por medio de reuniones presenciales, en las cuales asistía el grupo anteriormente descrito, estas reuniones se realizaron cada 20 días, después de que se realizaban las correcciones que se habían propuesto en reuniones anteriores.

[3] Galería de diagramas interactivos, desarrollados por Google.
[4] Framework en Javascript que permite la implementación de diferentes tipos de diagramas.
[5] Framework en Javascript que permite la generación de calendarios y sus eventos.
[6] Herramienta de generación de grandes volúmenes de datos personalizados, http://generatedata.com

En la primera reunión se discutió principalmente sobre la correcta implementación del proceso que debe seguir el inicio de programaciones, el emparejamiento de las solicitudes de cirugía con un cirujano y la visualización de los resultados del proceso de planeación de quirófanos. Además, se discutió sobre la forma y contenido de cada una de las vistas del módulo web. Esta reunión originó las siguientes mejoras y correcciones al sistema:

- El sistema debe de permitir que una cirugía perteneciente a una programación se le pueda modificar el quirófano, la hora de inicio y el tiempo que toma realizar la cirugía.
- Validar la disponibilidad de los quirófanos al cambiar el horario de la cirugía.
- Agregar una columna a la vista de emparejamientos de cirugías que permita visualizar las especialidades de los cirujanos que requiere esa cirugía.

Luego de realizar las correcciones propuestas por los tutores, se realizó la segunda reunión con el propósito de realizar la validación del sistema. El principal objetivo de esta reunión fue revisar que las correcciones propuestas en la anterior reunión fueron implementadas, las cuales se realizaron con éxito y fueron aprobadas, y discutir sobre aspectos de navegabilidad, usabilidad e interacción con el usuario, donde se generaron las siguientes propuestas de mejora para el sistema:

- Implementar la técnica de navegación Miga de Pan[7]
- En la página de inicio añadir un diagrama que explique de forma general el proceso que y las funciones que ofrece el módulo web.
- Añadir una descripción de la funcionalidad que se realiza en cada una de las vistas.
- Añadir información de ayuda, que guíen al usuario en el proceso de inicio de planeación de quirófanos.
- Generar una gran cantidad de datos, que permitan visualizar el despliegue de los diagramas utilizados para mostrar la información de las programaciones.

La tercera reunión se realizó luego de que los cambios propuestos en la anterior reunión se hubieran realizado y verificado por el equipo de desarrollo, esta reunión se enfocó en la revisión de las propuestas de mejora, las cuales fueron aceptadas por los tutores.

Además de las pruebas y validaciones que se realizaron en las reuniones anteriores, el sistema se probó con datos generados por el profesor Víctor Escallón. Un archivo permitió probar la funcionalidad de “Cargar datos” encargada de insertar en la base de datos información relacionada con los pacientes, quirófanos, profesionales de la salud, disponibilidades, etc...

Otro archivo conformado por los resultados de una ejecución de los módulos encargados de la generación de programaciones, permitió probar con un gran volumen de datos la carga y visualización de esta información, este archivo contenía 844 cirugías, cada una con un promedio de 3 procedimientos, repartidas por 4 quirófanos. Según el estudio del profesor Víctor

[7] Línea de texto que indica el recorrido realizado en el sistema y brinda la posibilidad de regresar.

Escallón, se descubrió que, en un hospital como el Hospital Universitario del Valle, sus quirófanos realizan entre 100 a 120 cirugías al mes, por lo tanto, el volumen de datos con el cual se probo es relevante para probar el módulo.

9. Resultados Obtenidos

De acuerdo con los requerimientos planteados para este proyecto, se generaron los resultados posteriormente descritos, que permitieron desarrollar un sistema de software usable y operativo para el usuario final, que, a su vez, integre los diferentes módulos de software necesarios para la generación y simulación de planeaciones de quirófanos que desarrollen cirugías electivas en las clínicas colombianas.

Objetivo 1: Analizar los sistemas matemáticos y de simulación que se requieren integrar para la generación, validación y visualización de las planeaciones de los quirófanos.

Al trabajar sobre el primer objetivo asociado con el análisis de los módulos matemáticos y de simulación, se generaron ciertos documentos y diagramas técnicos que proporcionan una visión más clara del sistema en general, permitiendo describir su funcionamiento, sus interfaces con otros módulos y la información que se transporta por el flujo de control del sistema.

Uno de los diagramas mencionados anteriormente es el diagrama de actividades, el cual permitió describir de una forma más clara el proceso de planeación de quirófanos, esbozando el flujo de trabajo a través de una serie de acciones que son realizadas por los distintos módulos que conforman el sistema.

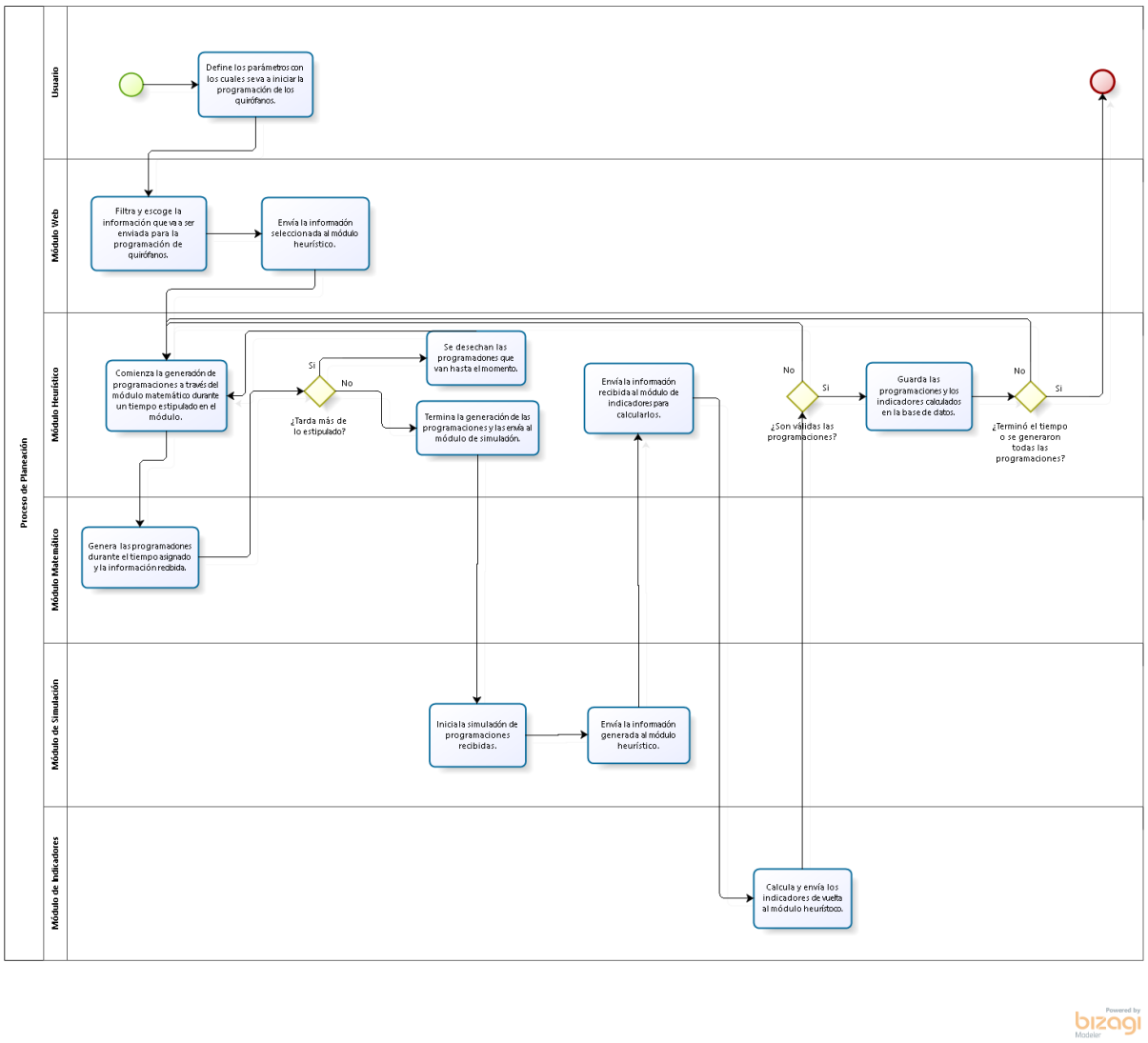


Ilustración 14: Diagrama de actividades del proceso de programación de cirugías

Otro de los diagramas mencionados es el diagrama de casos de usos, el cual permitió especificar el comportamiento del módulo web una vez esté integrado con los otros módulos encargados de la generación de planeaciones de quirófanos, en este diagrama se puede observar que funcionalidades tiene el usuario que va a usar la aplicación web. El siguiente diagrama representa los casos de uso del sistema, donde se puede observar que el usuario tiene interacción directa con la configuración de los datos de entrada de los módulos, la ejecución de ellos, y la visualización y modificación de los datos que genere dicha ejecución. Este diagrama está acompañado con el documento de descripción de casos de uso (Anexo 3), donde se describe detalladamente la funcionalidad, las precondiciones, poscondiciones y un guion de los pasos que el sistema realiza con el fin de cumplir la funcionalidad.

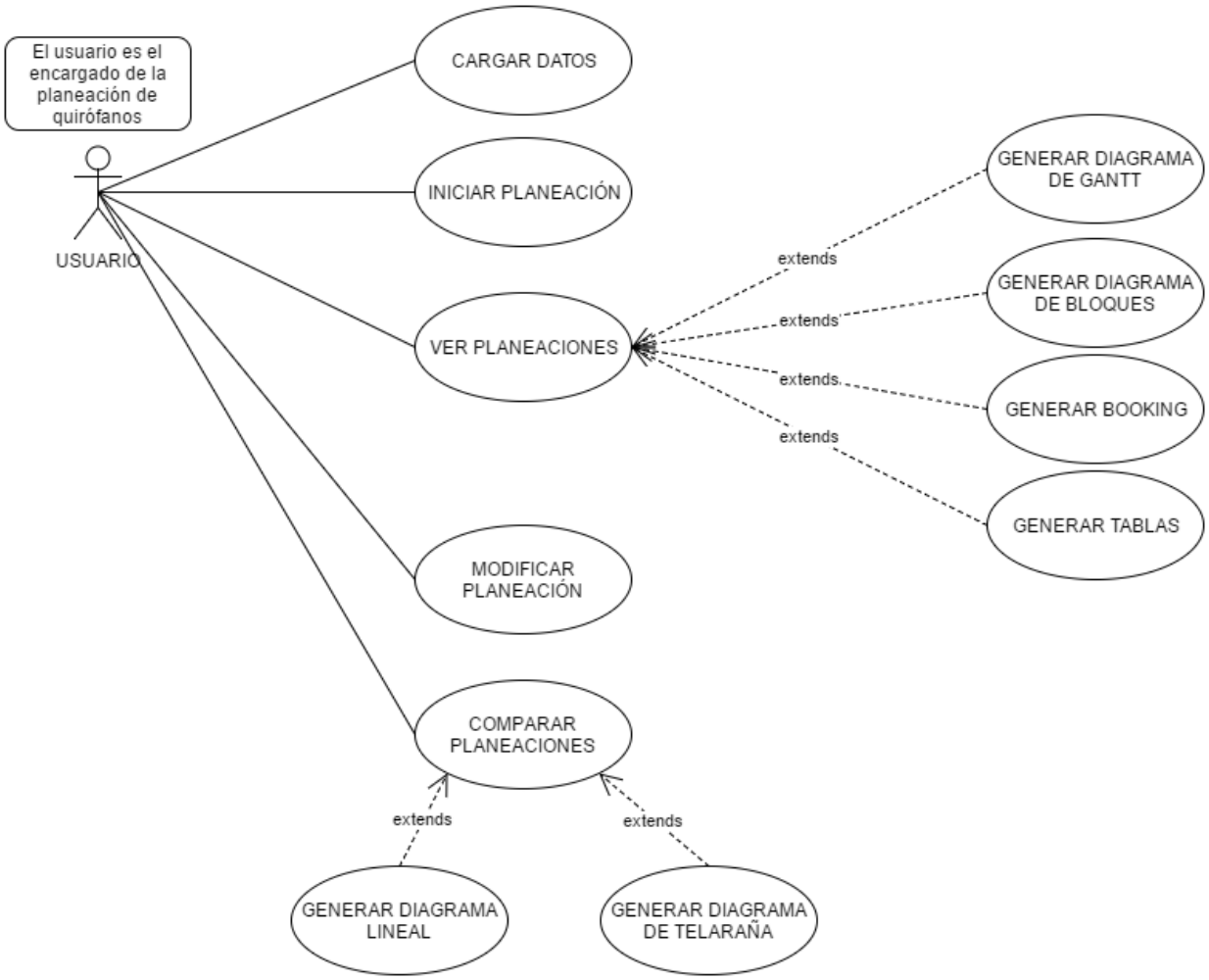


Ilustración 15: Diagrama de casos de uso.

Por último, para poder entender a cabalidad los módulos mencionados, es necesario describir de manera estructurada la información que transita por el flujo de datos del sistema, para ello, se realizó un esquema de base de datos (Anexo 1), el cual permite el almacenamiento de la información requerida para la inicialización, ejecución y estudio de resultados del proceso de generación de planeación de quirófanos.

Objetivo 2: Diseñar e implementar una arquitectura que permita la integración de los diferentes sistemas de software necesarios para la elaboración de la aplicación y **Objetivo 3:** Diseñar e implementar interfaces que permitan la comunicación entre los diferentes módulos de software.

Trabajando sobre la representación arquitectónica que describe los flujos de control y de información que relacionan los diferentes módulos que conforman la solución, se encontró que los principales problemas que se tenían al diseñar dicha arquitectura eran la forma en cómo se iba implementar la comunicación entre los módulos y el grado de incertidumbre que existía al no conocer cuál iba a ser la tecnología que con la que se iba a implementar los módulos.

En un principio, la forma de establecer la comunicación era por medio de servicios web que brindaban pequeñas cantidades de información al módulo matemático, esto en términos de número de transacciones y del tiempo que toma enviar información por medio de la red, era poco viable ya que se quitaba tiempo de procesamiento a los módulos encargados de la planeación mientras consultaban información al módulo de base de datos.

En busca de otra solución, se estableció que se podría enviar solo una vez toda la información que los módulos necesitan para la planeación de quirófanos, y una vez terminado este proceso se enviaba la información resultante del proceso de planeación, esto iba a ser implementado por medio de servicios web utilizando el formato de descripción de datos JSON, aprovechando sus cualidades de ser ligero y fácil de leer, se podría disminuir el tiempo de transporte y lectura de la información. Esta solución no se llevó a cabo, debido a que existía la posibilidad de que los módulos encargados de la planeación pudieran ser ejecutados dentro del mismo servidor donde es ejecutado el módulo web, haciendo llamados por medio del sistema se puede iniciar la ejecución de los módulos anteriormente mencionados, así la información es transportada por medio de archivos de tipo JSON, almacenados en un directorio visible para los módulos encargados de la planeación. Esta arquitectura está representada por un diagrama de deployment (Anexo 2).

Objetivo 4: Diseñar e implementar tres módulos: el primero para la administración de la base de datos, el segundo para el manejo de los indicadores utilizados en la estrategia de planeación y el último que permita la visualización de los resultados obtenidos tras el proceso de planeación de quirófanos.

Con el objetivo de desarrollar los tres módulos descritos en objetivo número 4, lo cuales hacen parte del módulo web, se desarrolló una aplicación web que permite obtener una experiencia amigable al realizar tareas relacionadas con la planeación de quirófanos. Las funcionalidades de la aplicación web se dividen principalmente en tres, las cuales son: configuración de los datos para el proceso de planeación, Visualización y modificación de los resultados del proceso de planeación y Comparación de las programaciones que conforman una planeación de quirófanos de acuerdo a los indicadores asociados a esta. Junto a la implementación de este objetivo, se desarrolló un manual de usuario (Anexo 4) que explica cómo utilizar la aplicación web y un manual de despliegue (Anexo 5) que explica como instalar la aplicación web y sus componentes.

Inicio, la aplicación web hace una breve introducción sobre el funcionamiento general del sistema, como se muestra en la siguiente imagen.

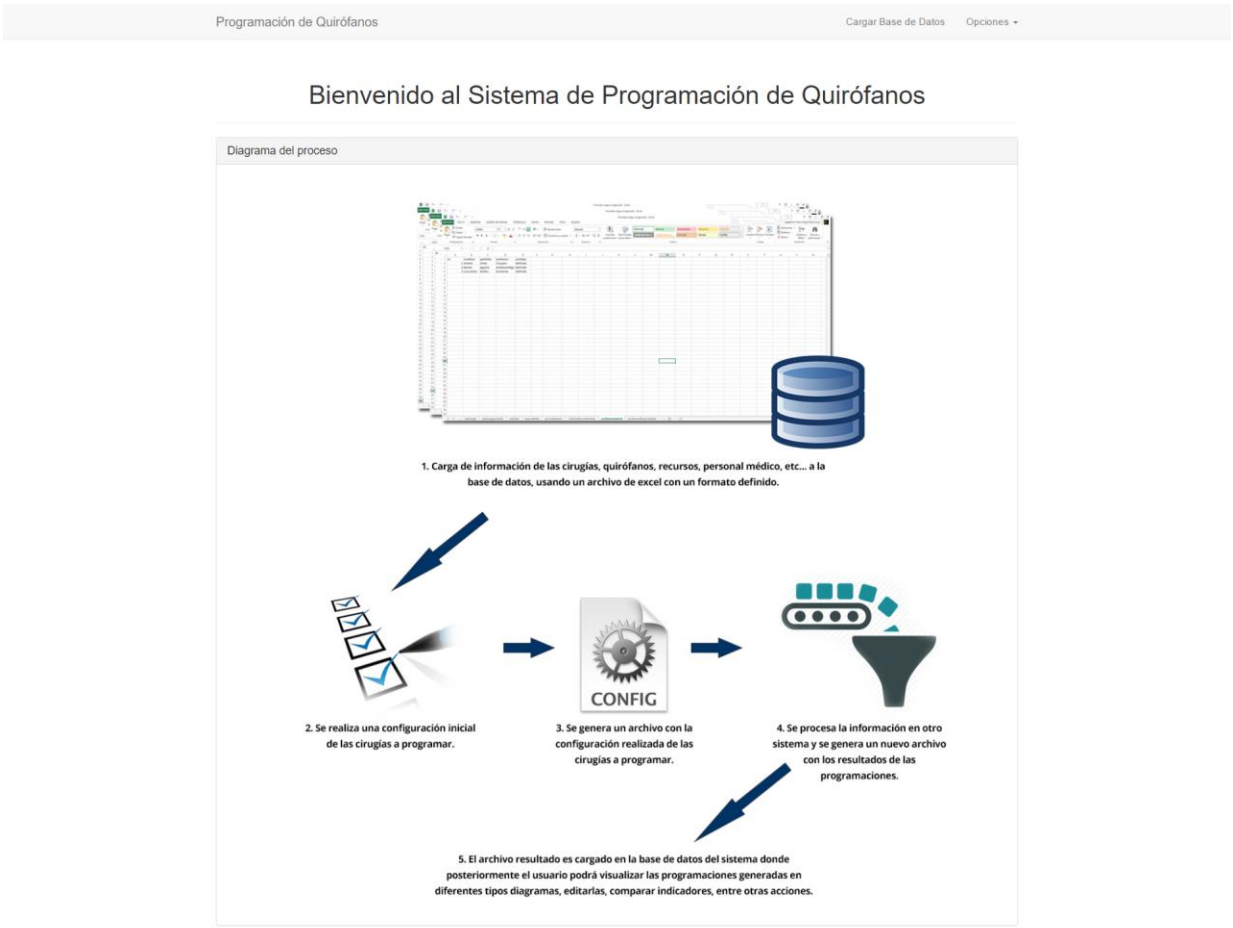


Ilustración 16: Home de la aplicación.

La configuración de los datos para el proceso de planeación está conformada por tres vistas, las cuales son:

Cargue de la base de datos, permite poblar la base de datos por medio de un Excel que es cargado por el usuario a la aplicación.

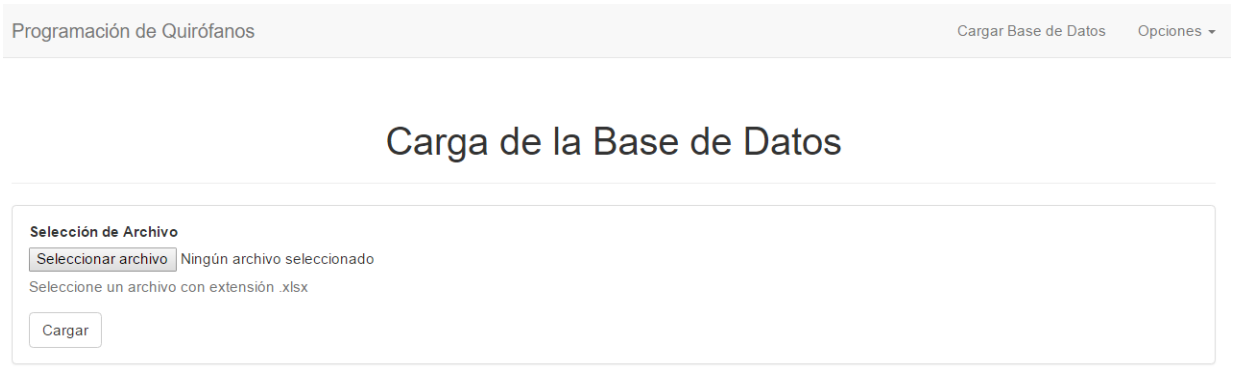


Ilustración 17: Captura del módulo de carga de datos.

Emparejamiento de cirugías, permite relacionar aquellos pacientes quirúrgicos que han presentado una solicitud de cirugía, pero aún no se les han asignado un cirujano que realice la cirugía.

Ilustración 18: Captura emparejamiento de cirugías.

Inicio de programación, permite elegir los parámetros necesarios para poder iniciar el proceso de planeación. Una vez seleccionada la información inicial el sistema crea un archivo en formato JSON con dicha información, luego procede a ejecutar el ejecutable de Windows con la implementación de los módulos encargados de la planeación de quirófanos, una vez este ejecutable haya terminado su ejecución se crea un archivo JSON con los resultados de la

planeación y se procede a almacenar en la base de datos estos resultados para su visualización.

Ilustración 19: Captura inicio de programación.

La visualización y modificación de los datos relacionados con la planeación de quirófanos, consta con un conjunto de vistas encargadas de listar las planeaciones indicando el rango de fechas en que se llevarán a cabo las cirugías, y el número de programaciones relacionadas.

Número de Pacientes	Tiempo Procesamiento	Fecha Inicio	Fecha Fin	Fecha de Programación	No. Programaciones	
23	14	2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:46.0	7	Ver
23	14	2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:07.0	0	Ver
23	14	2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:46:54.0	0	Ver
23	14	2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:44:40.0	0	Ver
12	23	2016-11-20 00:00:00.0	2017-01-19 00:00:00.0	2016-11-20 21:36:11.0	0	Ver
45	30	2016-11-22 00:00:00.0	2016-12-03 00:00:00.0	2016-11-17 08:49:40.0	2	Ver
4565	45	2016-11-11 00:00:00.0	2016-11-18 00:00:00.0	2016-11-11 02:00:15.0	2	Ver

Ilustración 20: Captura listado de generaciones de programaciones.

Para obtener más detalle sobre una planeación, se puede elegir la opción de “Ver” frente a cada planeación, esto permite observar los atributos que tiene una programación perteneciente a la planeación seleccionada, así como se aprecia en la siguiente imagen.

Fecha Inicio	Fecha Fin	Fecha de Programación	Promedio Indicadores	
2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:46.0	3.3333333333333335	Ver
2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:46.0	3.3333333333333335	Ver
2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:46.0	3.3333333333333335	Ver
2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:46.0	3.3333333333333335	Ver
2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:46.0	3.3333333333333335	Ver
2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:46.0	3.3333333333333335	Ver
2016-11-20 00:00:00.0	2017-01-16 00:00:00.0	2016-11-20 21:48:46.0	3.3333333333333335	Ver

Ilustración 21: Listado de programaciones por generación de programaciones.

Al elegir la opción de “Ver” al lado de cada programación se puede ver los datos relacionados con la programación de distintas maneras, las cuales son:

Diagrama de Gantt, donde se puede apreciar las cirugías por quirófanos y paciente, junto con la duración de cada cirugía.

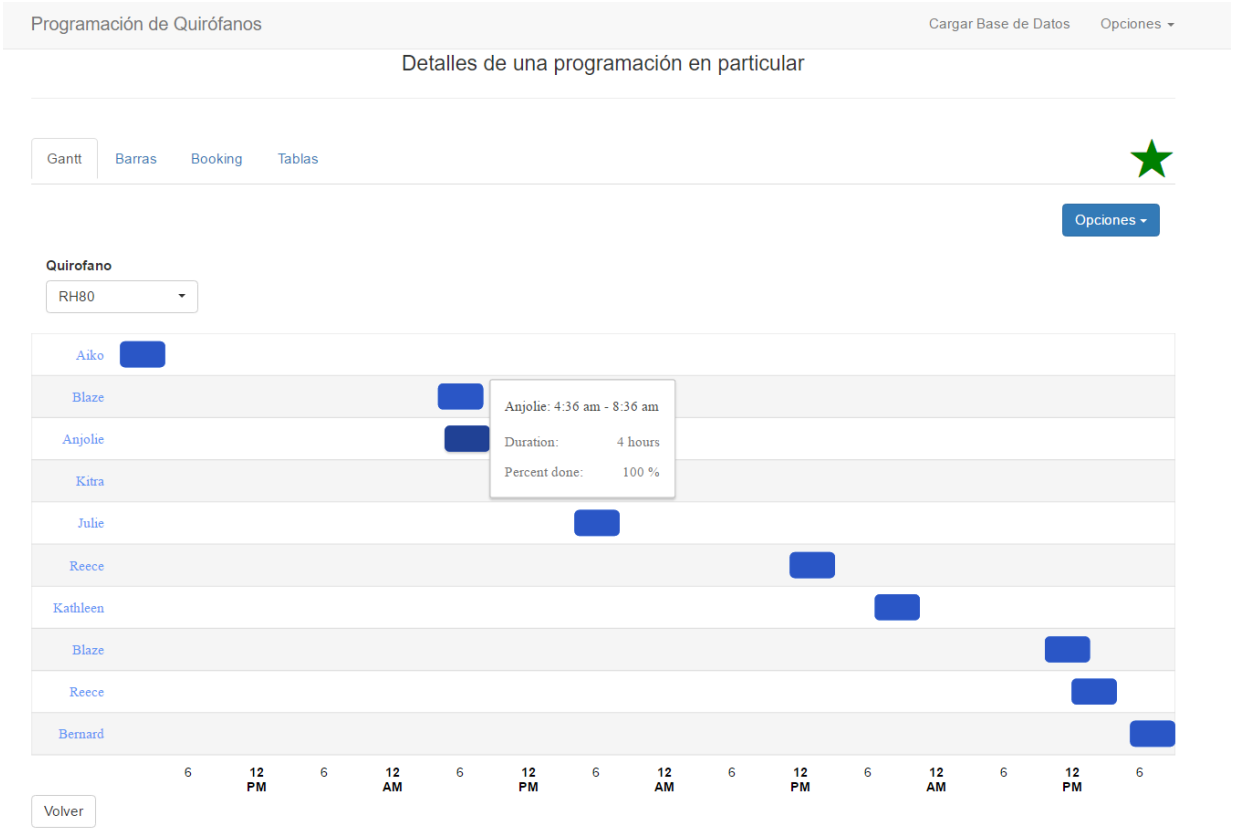


Ilustración 22: Captura diagrama de Gantt

Diagrama de Barras, se puede visualizar todas las cirugías relacionadas con la programación, estas son ordenadas por quirófanos y se puede observar el paciente y la duración asociada a la cirugía.

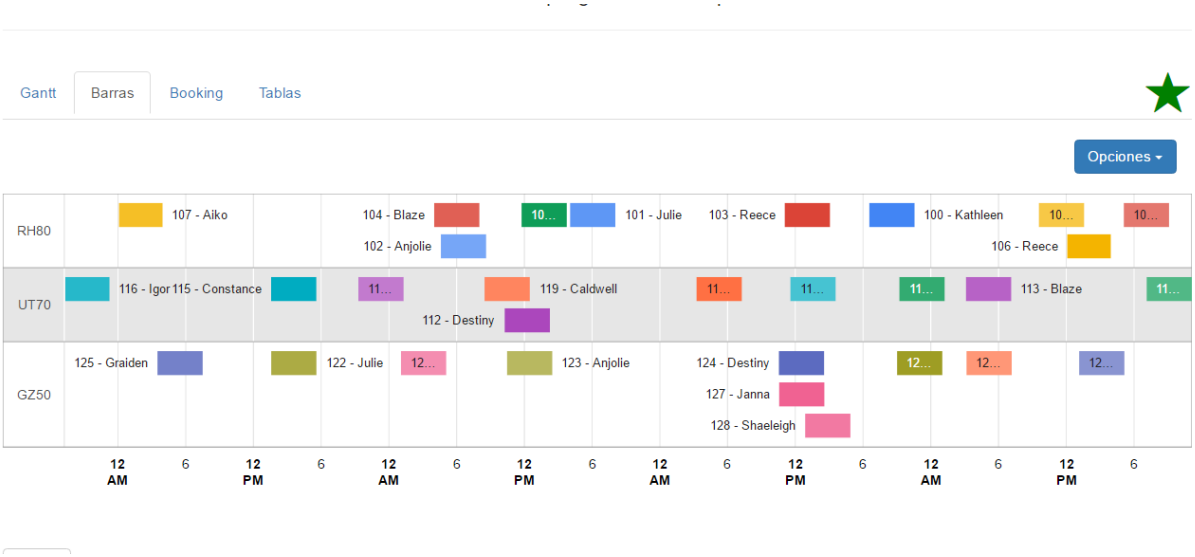


Ilustración 23: Captura diagrama de Barras

Booking, por medio de una agenda se muestran las cirugías por quirófano, junto con el paciente, la fecha y la duración asociada.

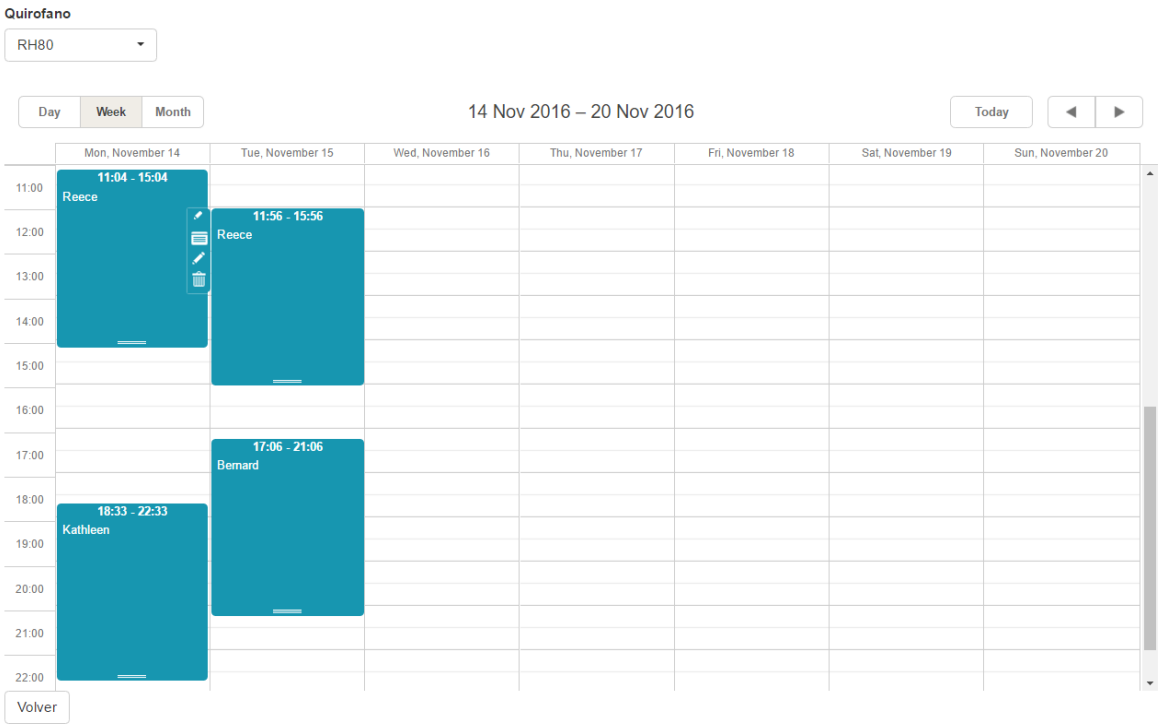
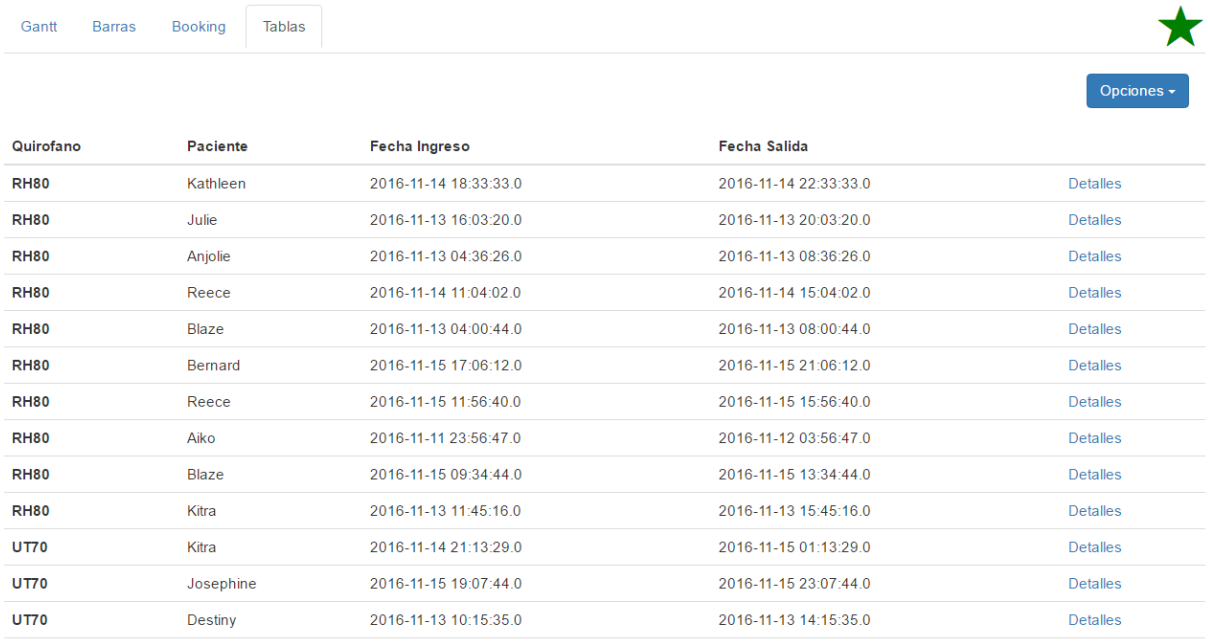


Ilustración 24: Booking

Tabla, la información de cada cirugía es mostrada de forma horizontal por medio de una tabla



Quirofano	Paciente	Fecha Ingreso	Fecha Salida	
RH80	Kathleen	2016-11-14 18:33:33.0	2016-11-14 22:33:33.0	Detalles
RH80	Julie	2016-11-13 16:03:20.0	2016-11-13 20:03:20.0	Detalles
RH80	Anjolie	2016-11-13 04:36:26.0	2016-11-13 08:36:26.0	Detalles
RH80	Reece	2016-11-14 11:04:02.0	2016-11-14 15:04:02.0	Detalles
RH80	Blaze	2016-11-13 04:00:44.0	2016-11-13 08:00:44.0	Detalles
RH80	Bernard	2016-11-15 17:06:12.0	2016-11-15 21:06:12.0	Detalles
RH80	Reece	2016-11-15 11:56:40.0	2016-11-15 15:56:40.0	Detalles
RH80	Aiko	2016-11-11 23:56:47.0	2016-11-12 03:56:47.0	Detalles
RH80	Blaze	2016-11-15 09:34:44.0	2016-11-15 13:34:44.0	Detalles
RH80	Kitra	2016-11-13 11:45:16.0	2016-11-13 15:45:16.0	Detalles
UT70	Kitra	2016-11-14 21:13:29.0	2016-11-15 01:13:29.0	Detalles
UT70	Josephine	2016-11-15 19:07:44.0	2016-11-15 23:07:44.0	Detalles
UT70	Destiny	2016-11-13 10:15:35.0	2016-11-13 14:15:35.0	Detalles

Ilustración 25: Captura visualización de programación en formato tabla.

Además, la aplicación cuenta con la posibilidad de editar los datos de una cirugía por medio de la siguiente vista.

Ilustración 26: Captura edición de cirugías.

La comparación entre programaciones, se hace en base a los indicadores asociados a la programación, esta comparación se puede hacer por medio de diferentes vistas que proporcionan diferentes tipos de diagramas que permiten el estudio de cada una de las programaciones desde diferentes perspectivas. Además, la aplicación web brinda la posibilidad de elegir qué programaciones de la planeación o indicadores incluir en la comparación. En la siguiente imagen se muestra las diferentes vistas utilizadas para comparar las programaciones.

Diagrama de barras, las series corresponde a los indicadores y el eje y corresponde al valor que tiene una programación respecto al indicador.

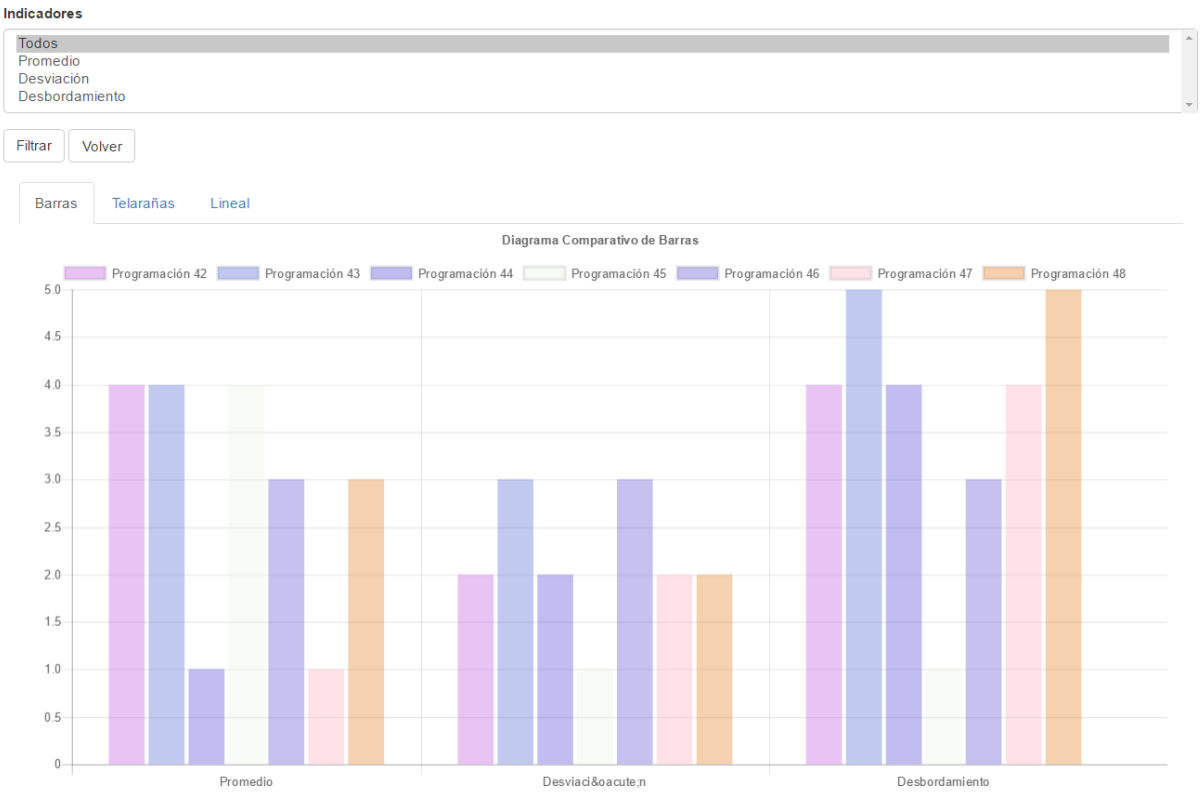


Ilustración 27: Captura comparación de indicadores en diagrama de barras.

Diagrama de Telaraña, cada indicador se ubica en una punta del gráfico y la programación es mapeada de acuerdo al valor del indicador relacionado.

Ilustración 28: Captura comparación de indicadores en diagrama de telaraña.

Diagrama lineal, en el eje x se encuentran ubicados los indicadores y en el eje y están los posibles valores que una programación puede obtener de un indicador.

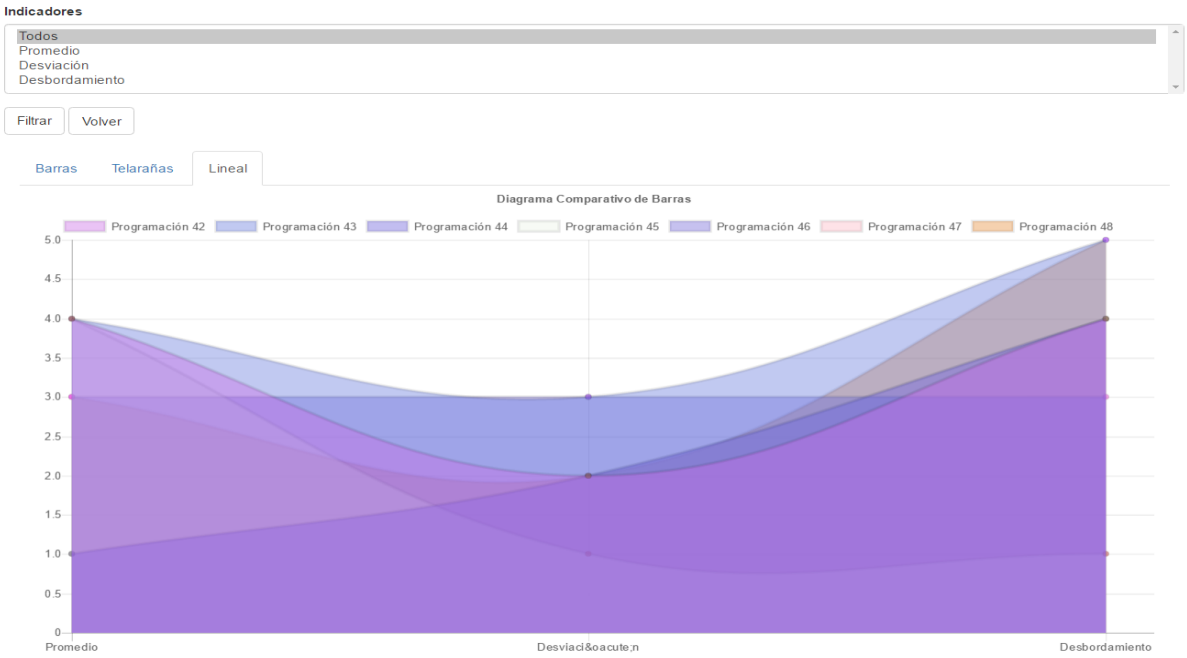


Ilustración 29: Captura comparación de indicadores en diagrama de lineal.

10. Conclusiones

El desarrollo de este proyecto permitió la integración entre un módulo web, capaz de brindar una interfaz al usuario final del sistema; un módulo de datos, capaz de almacenar la información asociada a las programaciones, recursos y entidades de las clínicas, y un conjunto de módulos capaces de generar un conjunto de programaciones. Esta integración en su conjunto brinda una manera en que el usuario final del sistema pueda interactuar con el proceso de programación, visualizar de una forma fácil y amigable los resultados de las programaciones, y editar dichos resultados. Todo esto gracias a las tecnologías usadas en el desarrollo del proyecto, que permiten obtener atributos de calidad como la usabilidad, operatividad, navegabilidad y un buen rendimiento en el procesamiento de datos.

El módulo web permite el estudio de los resultados obtenidos por la ejecución del proceso de programación de quirófanos, a través de diagramas interactivos que permiten ver en detalle la información de cada cirugía o programación. Además, el sistema genera tres tipos de diagramas totalmente interactivos que permiten la comparación entre programaciones distintas, en un mismo horizonte de tiempo, basándose en sus indicadores, con el objetivo de elegir la programación que más se adecue a los criterios de la clínica.

Durante el desarrollo del proyecto se ayudó a afinar y optimizar los procesos de la solución inicial, definiendo estándares de comunicación entre los módulos. Dependiendo de las tecnologías que se iban a utilizar para desarrollarse los módulos, se diseñó un modelo arquitectural que permitió definir la representación general del sistema. Además, el modelo de datos resultante de la fase de diseño provee una definición clara de las entidades y datos involucrados en el proyecto que permite la identificación de la información que se transporta entre los módulos que comprenden la solución.

Con la solución realizada se contribuye al sistema de salud colombiano, ya que en este momento es uno de los sectores con más problemas en el país, haciendo que las clínicas con que tienen una mal administración de recursos o simplemente no cuentan con muchos, tengan la posibilidad de mejorar la productividad que tienen en el momento con los recursos que cuentan.

11. Trabajos Futuros

A partir del trabajo realizado en este proyecto, y en la experiencia obtenida durante el desarrollo del mismo, se consideraron relevantes los siguientes trabajos que le darían continuidad al proyecto, considerando que el trabajo realizado es una parte de un sistema en fase de desarrollo, perteneciente al estudio doctoral del profesor Víctor Escallón.

1. La estructura organizacional de muchas clínicas está conformada por muchos tipos de roles que tienen diferentes funcionalidades y responsabilidades sobre la planeación, ejecución y administración de quirófanos y cirugías, por lo tanto, el sistema está diseñado para que en

trabajos futuros haya la posibilidad de tener un manejo de sesiones y roles, que permita la ejecución de diferentes actividades dependiendo de las funcionalidades relacionadas con el rol que ha iniciado sesión.

2. Para poder sacar provecho económico del sistema desarrollado, se ha planteado como una opción viable la implementación futura de un SaS (Software As a Service), donde el sistema desarrollado se podría exponer como un servicio, vender el acceso a estos servicios y poder darle un uso masivo al sistema, el cual podrá ser usado por cualquier clínica.

3. Conectarse directamente a la base de datos de un sistema donde se estén generando órdenes de cirugías y que el sistema se encargue solo de proponer las planeaciones, sin que un humano esté ordenándole cuando hacerlo.

4. Diseño, desarrollo y pruebas de la visualización y contenido de los indicadores relacionados con las programaciones, dependiendo de los resultados de una investigación dentro del estudio doctoral del profesor Víctor Escallón.

12. Aprendizaje del Proyecto

Durante el desarrollo del proyecto tuvimos la oportunidad toparnos con lenguaje utilizado en el sector clínico, conocer cómo se realizan procesos como la solicitud de cirugías, definición de prioridades de pacientes, programaciones de cirugías electivas y de urgencias, y, además, conocer las condiciones mínimas en cuanto a recursos y personal médico que deben cumplir los quirófanos y las cirugías respecto a las normas colombianas.

También tuvimos la oportunidad de poner en práctica muchas de los conocimientos obtenidos en las clases de ingeniería y arquitectura de software, participando en las fases principales del desarrollo de software como lo son el análisis, diseño, desarrollo y pruebas.

Además, tuvimos la oportunidad de ejercer nuestros conocimientos en el área del desarrollo de aplicaciones web, mientras cumplíamos los atributos de calidad objetivos para el módulo web.

13. Glosario

- **Cirugía electiva:** Cirugía que está programada con antelación porque no implica una emergencia médica.
- **Diagrama de Booking:** En español reserva. Es un diagrama donde se despliega el espacio de tiempo reservado por cada cirugía de una programación como una agenda.
- **Diagrama de Gantt:** Su objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.
- **Framework:** Define un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas similares.
- **Interoperabilidad:** Es la capacidad que tiene un sistema, con interfaces conocidas, de funcionar con otros sistemas sin restricción de acceso o de implementación.
- **IPS:** (Instituto Prestador de Salud) Institutos que prestan los servicios médicos de consulta, hospitalarios, clínicos y de cuidados intensivos.
- **Navegabilidad:** Facilidad con la que un usuario puede desplazarse por las páginas que componen un sitio web.
- **Planeación:** Conjunto de programaciones resultantes del proceso de generación de programaciones.
- **Programación:** Conjunto de cirugías organizadas por quirófano, hora de inicio y finalización.
- **Usabilidad:** Es una cualidad de los sistemas que son sencillos de usar porque cuentan con una interfaz de usuario o funcionamiento que permite a los usuarios interactuar con los sistemas de manera fácil y cómoda.

14. Bibliografía

Aldrich, J., & Garrod, C. (2013). Principles of Software Construction: Objects, Design and Concurrency Java Collections.

Comprender los servicios web, Parte 2: Web Services Description Language (WSDL). (2011, August 8). Retrieved from http://www.ibm.com/developerworks/ssa/webservices/tutorials/ws-understand-web-services2/

Crockford. (2013). JSON. *Journal of Chemical Information and Modeling*, *53*, 1689–1699. http://doi.org/10.1017/CBO9781107415324.004

Fielding, R., Irvine, U. C., & Gettys, J. (1999). Hypertext Transfer Protocol -- HTTP / 1. 1 Status of this Memo, 1–129.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns.

Garlan, D., & Shaw, M. (1993). An Introduction to Software Architecture. *Advances in Software Engineering and Knowledge Engineering*, *1*(January), 1–40. http://doi.org/10.1142/9789812798039_0001

Hartmann, H., Keren, M., Matsinger, A., Rubin, J., Trew, T., & Yatzkar-Haham, T. (2010). Integrating heterogeneous components in software supply chains. In *Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering - PLEASE '10* (pp. 8–15). New York, New York, USA: ACM Press. http://doi.org/10.1145/1808937.1808939

HTML | MDN. (n.d.). Retrieved October 4, 2015, from https://developer.mozilla.org/es/docs/Web/HTML

Monson-haefel, R. (2004). J2EE Web Services. *Analyst, The*, 1–79.

Sabooniha, N., Toohey, D., & Lee, K. (2012). An evaluation of hospital information systems integration approaches. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics - ICACCI '12* (p. 498). New York, New York, USA: ACM Press. http://doi.org/10.1145/2345396.2345479

SWEBOK. (2014). *Igarss 2014*. http://doi.org/10.1007/s13398-014-0173-7.2

Treiblmayr, M., Scheider, S., Krüger, A., & von der Linden, M. (2012). Integrating GI with non-GI services—showcasing interoperability in a heterogeneous service-oriented architecture. *GeoInformatica*, *16*(1), 207–220. http://doi.org/10.1007/s10707-011-0132-9

Wang, X. V., & Xu, X. W. (2012). DIMP: an interoperable solution for software integration and product data exchange. *Enterprise Information Systems*, *6*(3), 291–314. http://doi.org/10.1080/17517575.2011.587544

What is REST? (n.d.). Retrieved October 4, 2015, from http://www.restapitutorial.com/lessons/whatisrest.html

W3schools. (s.f.). *w3schools.* Obtenido de Introduction to XML: http://www.w3schools.com/xml/xml_whatis.asp