



**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**  
**Facultad de Ingeniería en Electricidad y Computación**

INTEGRACIÓN DE LOS MÓDULOS DE HISTORIAS MÉDICAS Y  
REGISTRO DE BENEFICIARIOS DEL SISTEMA INTEGRADO DEL  
PATRONATO PROVINCIAL DEL SERVICIO SOCIAL DE PASTAZA

**INFORME DE PROYECTO INTEGRADOR**

Previo a la obtención del Título de:

**INGENIERO EN COMPUTACIÓN**

Por:

César Augusto Ramírez Ávila

Wellington Andrés Martínez Flores

**GUAYAQUIL – ECUADOR**

**AÑO: 2019**

## **AGRADECIMIENTOS**

Agradezco a Dios, por todo lo que me ha brindado, a mis Padres por todo la paciencia, compresión y apoyo que me han dado, a mis Hermanos y Hermanas por siempre confiar en mí, a mi familia, a mi mujer y mis hijos por estar conmigo; también a mi profesor de materia integradora PHD. Boris Vintimilla y mi tutor PHD. Luis Mendoza.

Cesar Ramírez Avila.

## **DEDICATORIA**

Dedico el presente proyecto a Dios por siempre estar ahí, a mi abuela y mi padre que están en el cielo, a mi querida madre por su amor eterno, a mis hermanos y hermanas por su cariño, a mi familia paterna y materna por su presencia, a mi mujer y mis hijos por estar conmigo.

Cesar Ramírez Avila.

## **DECLARACIÓN EXPRESA**

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

César Augusto  
Ramírez Ávila

Wellington Andrés  
Martínez Flores

## RESUMEN

En proyectos integradores anteriores fueron creados los módulos Historias Médicas y Registro de Beneficiarios, con el objeto de automatizar la información de las actividades diarias de atención médica que brinda el Patronato Provincial de Servicio Social de Pastaza, PPSSPz. Sin embargo, estos módulos funcionan independientemente y no permiten el flujo de información necesaria entre ellos para evitar el registro de información duplicada. Por ello, el presente proyecto integrador tuvo por objetivo la integración de estos módulos para que funcionen como un solo sistema, y que pueda ser utilizado en el menor tiempo posible. Como solución al problema descrito, se creó una Interfaz de Programación de Aplicaciones (API, del inglés Application Programming Interface) integradora que permite que los módulos Historias Médicas y Registro de Beneficiarios funcionen conjuntamente, interactuando y compartiendo información implícitamente, y permitiendo la navegación de los usuarios entre los distintos módulos como un solo sistema llamado Sistema Integrado del Patronato Provincial de Servicio Social de Pastaza, SIPPSSPz. La API integradora fue implementada usando NodeJS, y usa SQLite como base de datos; también usa Sequelize como ORM y JSON, para intercambio de información. Además, se analizaron y revisaron las interfaces de cada módulo para garantizar su correcto funcionamiento. Como resultado de la integración, se cuenta con un proceso integrado que diferencia entre los roles de los usuarios finales e intercambiar información de forma concisa y coherente, el cual ya puede ser utilizado por los usuarios finales en las actividades diarias que el PPSSPz brindan a la ciudadanía.

Palabras Clave: Servicio social, Integración de sistemas, API, Historias médicas, Registro de beneficiarios.

## **ABSTRACT**

*In previous integrating projects, the modules for Medical Stories and Beneficiary Records were created, in order to automatize the information of the daily health care activities provided by the Pastaza Provincial Social Service Board PPSSPz. However, these modules work independently and do not allow the flow of necessary information between them to avoid duplicating records or information. Therefore, this integrating project aimed at integrating these modules so that all functions are in one system, and they can be used in the shortest possible time. As a solution to the problem described, an integrating Application Programming Interface (API) was created. It allows the Medical History and Beneficiary Records modules to work together, interacting and sharing information implicitly, and allowing the navigation of users among the different modules as a single system called the Integrated System of the Provincial Board of Social Service of Pastaza, SIPPSSPz. The integrating API was implemented using NodeJS and uses SQLite as a database. It also uses Sequelize as ORM and JSON, for information exchange. In addition, the interfaces of each module were analyzed and reviewed to ensure proper operation. As a result of the integration, there is an integrated process that differentiates between the roles of the end users and exchange information in a concise and coherent way, which can already be used by the end users in the daily activities that the PPSSPz provide to the citizens.*

*Keywords: PPSSPz, MHMRB, SIPPSSPz, API integrator, end users*

## ÍNDICE GENERAL

DEDICATORIA.....	3
RESUMEN .....	I
<i>ABSTRACT</i> .....	II
ÍNDICE GENERAL.....	III
ABREVIATURAS.....	VI
ÍNDICE DE FIGURAS .....	VII
ÍNDICE DE TABLAS .....	IX
CAPÍTULO 1 .....	11
1.    Introducción .....	11
1.1    Antecedentes.....	11
1.2    Descripción del problema .....	11
1.3    Justificación .....	12
1.4    Objetivos.....	12
1.4.1    Objetivo General.....	12
1.4.2    Objetivos Específicos .....	12
1.5    Marco Teórico.....	13
1.5.1    Integración de sistemas.....	13
1.5.2    Interfaz de Programación de Aplicaciones API.....	15
1.5.3    Arquitectura de una aplicación web .....	16
1.5.4    JavaScript .....	17
1.5.5    JavaScript Como Backend y Frontend .....	18
1.5.6    ReactJS y NodeJS.....	18
CAPÍTULO 2 .....	20
2.    Metodología .....	20

2.1	Plan de recolección de datos .....	20
2.2	Fiabilidad de los datos .....	20
2.3	Análisis de los datos .....	21
2.4	Propuesta de solución – Arquitectura de la API.....	23
2.4.1	Vista de Escenarios .....	23
2.4.2	Vista Lógica .....	25
2.4.3	Vista de Desarrollo .....	28
2.4.4	Vista de Procesos.....	29
2.4.5	Vista de Física .....	29
2.5	Plan de Implementación .....	30
CAPÍTULO 3 .....		32
3.	IMPLEMENTACIÓN DE LA SOLUCION.....	32
3.1	API integrador.....	32
3.1.1	Entorno físico del SIPPSSPz.....	32
3.1.2	Implementación API.....	34
3.2	Actualizaciones adicionales a los módulos historias médicas y registro de beneficiarios .....	38
3.2.1	Acceso al sistema PPSSPZ y módulo registro de beneficiarios ..	38
3.2.2	Módulo Historias Médicas.....	44
3.3	Análisis de resultados .....	47
3.3.1	Funcionalidad .....	47
3.3.2	Datos antes de la integración .....	49
3.3.3	Resultados después de la integración .....	49
Conclusiones Y RECOMENDACIONES .....		50
Conclusiones.....		50
Recomendaciones .....		50

Bibliografía .....	54
ANEXOS .....	56

## ABREVIATURAS

ESPOL	Escuela Superior Politécnica del Litoral
FIEC	Facultad de Ingeniería en Electricidad y Computación
PPSSPz	Patronato Provincial de Servicio Social de Pastaza
API	Application Programming Interface
W3C	Consortium World Wide Web
MHMRB	Módulos de Historias Médicas y Registro de Beneficiarios
SIPPSSPz	Sistema Integrado del PPSSPz

## ÍNDICE DE FIGURAS

Figura 1.1 Actividades de Integración [2]. .....	14
Figura 1.2 Interación del API.....	15
Figura 1.3 Arquitectura Aplicaciones Web [4].....	16
Figura 2.1 Esquema general Sistema Integrado del PPSSPZ. .....	21
Figura 2.2 Proceso derivación de cita médica a atención médica [11].....	22
Figura 2.3 Modelo de “4+1” vistas [10].....	23
Figura 2.4 Diagrama de Clases del API. ....	25
Figura 2.5 Diagrama de Secuencia para historia de usuario HU002.....	26
Figura 2.6 Diagrama de Secuencia para historia de usuario HU004.....	27
Figura 2.7 Modelo Conceptual de la base de datos del API.....	27
Figura 2.8 Diagrama de Despliegue.....	30
Figura 3.1 Estructura física del sistema PPSSPz.....	33
Figura 3.2 Estructura archivos del API. ....	35
Figura 3.3 Carpeta de DB.....	35
Figura 3.4 Carpeta routers. ....	36
Figura 3.5 Archivo principal del API. ....	36
Figura 3.6 Tablas de la Base de Datos del API.....	37
Figura 3.7 Código para acceder a la tabla Api_recurso. ....	37
Figura 3.8 Interfaz de inicio de sesión del sistema PPSSPz. ....	38
Figura 3.9 Interfaz de Unidad. ....	39
Figura 3.10 Menú principal del sistema PPSSPz. ....	39
Figura 3.11 Interfaz de Ingreso y modificación de Pacientes. ....	40
Figura 3.12 Interfaz de Consulta de Paciente. ....	41
Figura 3.13 Interfaz de Ingreso de Citas Médicas. ....	42
Figura 3.14 Interfaz de Citas programadas. ....	42
Figura 3.15 Menú principal del Módulo Historias Médicas. ....	44
Figura 3.16 Interfaz de Historia Médica Medicina General.....	45
Figura 3.17 Usuario con Rol atendiendo una cita médica. ....	48
Figura 3.18 Usuario con Rol Médico registrando la historia médica.....	48
Figura R.1 Procedimiento onClickIrAsistencia.....	51

Figura R.2 Procedimiento obtenerUbi_fun. ....	51
Figura R.3 Método componentDidMount.....	52
Anexo A2.1 Interfaz Terapia Física .....	57
Anexo A2.2 Interfaz Terapia de Lenguaje .....	58
Anexo A2.3 Interfaz Psicología .....	58
Anexo A2.4 Interfaz Odontología .....	59

## ÍNDICE DE TABLAS

Tabla 1.1 Elementos de Integración para Sistemas de Productos [2] .....	15
Tabla 2.1 Personal entrevistado para recolectar datos. .....	20
Tabla 2.2 Rol indirecto. .....	24
Tabla 2.3 Roles principales del API.....	24
Tabla 2.4 Historias de Usuario. ....	24
Tabla 2.5 Descripción detallada historia de usuario Conseguir uri destino. ....	25
Tabla 2.6 Planificación de Actividades. ....	31



# CAPÍTULO 1

## 1. INTRODUCCIÓN

En este capítulo exponemos la descripción del problema, justificación, así como los objetivos que se quiere llegar a solucionar; también definimos el marco teórico, donde mostramos los conceptos necesarios que vamos a utilizar en el desarrollo de nuestro proyecto.

### 1.1 Antecedentes

En el segundo semestre del año 2018, el Patronato Provincial de Servicio Social de Pastaza (PPSSPz) [1] inicia un proceso de automatización de sus procesos medulares. Para ello, como parte de un convenio entre el PPSSPz y la Escuela Superior Politécnica del Litoral (ESPOL), estudiantes de la materia integradora de Ingeniería de la Computación de la ESPOL procedieron a crear los módulos de Historias Médicas y de Registro de Beneficiarios (MHMRB) para el PPSSPz. El objetivo general de estos módulos fue sistematizar los registros manuales de atención médica a la ciudadanía en general, que se ejecutan en los diferentes proyectos de servicio social que ofrece el PPSSPz.

### 1.2 Descripción del problema

A pesar de que los MHMRB funcionan correctamente, todavía no pueden ser utilizados por los usuarios finales de una forma integrada. Cada módulo funciona de forma independiente, con acceso y registros de información a distintas bases de datos, hay diferencias entre las interfaces de usuario, no hay un proceso común que relacione la interacción entre los dos módulos y no se cuenta con algún reporte que muestre información resultante de la integración de los datos que cada módulo procesa. Otro aspecto que hay que considerar es que en el futuro se van a crear nuevos módulos y que, en conjunto, todos los módulos conformarán lo que se ha denominado el ***Sistema Integrado del PPSSPz (SIPPSSPz)***.

### **1.3 Justificación**

En la actualidad, el personal del PPSSPz que interactúa con los beneficiarios y pacientes del patronato, no utilizan los MHMRB para registrar la información. Por consiguiente, siguen manejando la información de forma manual. Por otro lado, dado que a la fecha todavía los servidores del PPSSPz no están a tono para alojar los módulos desarrollados, se ha visto como una oportunidad lograr la integración de los MHMRB antes de que éstos pasen a producción.

### **1.4 Objetivos**

#### **1.4.1 Objetivo General**

Integrar los módulos de Historias médicas y de Registro de Beneficiarios (MHMRB) del Sistema Integrado del Patronato Provincial de Servicio Social de Pastaza (SIPPSSPz), automatizando el flujo de información entre los módulos, para soportar la atención, seguimiento y control de los pacientes y beneficiarios de los servicios que el PPSSPz ofrece a la ciudadanía.

#### **1.4.2 Objetivos Específicos**

- 1) Analizar el diseño e implementación de los MHMRB.
- 2) Diseñar la arquitectura de Interfaz de Programación de Aplicaciones (API, del inglés Interfaz de Programación de Aplicaciones) requerida para la integración.
- 3) Implementar el API para automatizar el flujo de información.
- 4) Efectuar las pruebas de sistema para verificar el correcto funcionamiento del API.
- 5) Efectuar las pruebas de los MHMRB junto con el API.

## 1.5 Marco Teórico

### 1.5.1 Integración de sistemas

La Integración de sistemas [2] es un proceso muy útil cuando una organización o institución tiene sistemas de productos de software funcionando paralelamente e independiente, haciendo que los productos puedan interactuar conjuntamente y que el flujo de información sea coherente y consistente a la vista del usuario final.

Hoy en día la información es muy importante para cada empresa; pero, a veces, el acceso a la información puede ser complicada cuando en la misma empresa utilizan varios sistemas que trabajan independientemente, dando como resultado datos separados [2]. Por esta razón, la integración de sistemas es una gran solución para poder mejorar el acceso a la información, el flujo de información entre cada sistema independiente y que permita a la empresa mejorar el desempeño de su giro de negocio [2].

El integrar sistemas independientes o subsistemas supone obtener los componentes discretos del sistema; es decir, sistemas de productos que son elementos de software o hardware, que se van a unir a un único sistema con el objetivo de que trabajen en conjunto, sin alterar las características o propiedades con que fueron diseñados [2]. Además, se debe seguir un flujo de actividades de integración (ver Figura 1.1) que ayude a ensamblar de forma concisa los componentes discretos del sistema.



**Figura 1.1 Actividades de Integración [2].**

La Figura 1.1 muestra los pasos o actividades de integración y verificación, para poder ensamblar los componentes discretos del sistema. El primer paso es recibir los componentes discretos del sistema [2], analizarlos, verificar su funcionamiento, arquitectura y tecnologías con que fueron creados. El segundo paso es ensamblar los componentes discretos, usando técnicas de integración como: integración global, flujo, incremental, subconjuntos, etc. El tercer paso es la revisión del funcionamiento y errores de configuración del componente mientras se está integrando al sistema. El último paso es la validación del sistema final, junto con las pruebas necesarias de funcionamiento, rendimiento, flujo de información y posibles errores de configuración no considerados en el paso de revisión [2].

También hay que considerar los tipos de elementos de integración [2] para los sistemas de productos, dado que no es igual un sistema de producto que sistemas de servicios y sistemas empresariales. En la Tabla 1 se muestran los elementos que se deben considerar cuando se van a integrar sistemas de productos a un sistema final.

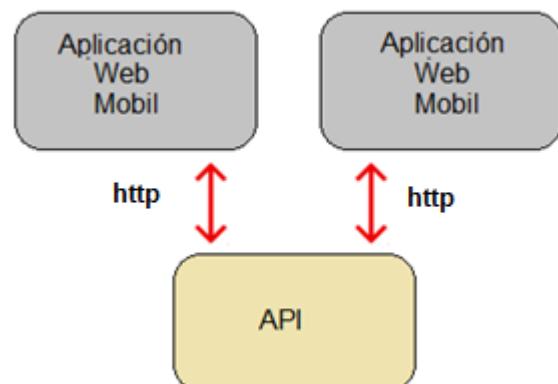
Elemento	Sistema de Producto
Elementos del Sistema	Partes de Hardware Partes de Software Usuarios
Interfaces Físicas	Procedimientos Protocolos APIs
Herramientas de Ensamblaje	Tecnologías de desarrollo
Herramientas de Verificación	Simuladores
Herramientas de Validación	Ambiente Operacional
Técnicas de Integración recomendadas	Técnicas de Integración de arriba hacia abajo Técnicas de Integración de abajo hacia arriba

**Tabla 1.1 Elementos de Integración para Sistemas de Productos [2].**

Además, cuando se integren los subsistemas se deberán hacer las respectivas verificaciones y validaciones del funcionamiento (pruebas), para detectar defectos o fallas del sistema total [2].

### 1.5.2 Interfaz de Programación de Aplicaciones API

API es un conjunto de conceptos y protocolos que sirven, de una forma simplificada, para poder comunicar o integrar aplicaciones sin necesidad de conocer o entender como fueron implementadas; así también, permite exponer y acceder a ciertas funcionalidades entre las aplicaciones que está comunicando [11].



**Figura 1.2 Interacción del API**

### 1.5.3 Arquitectura de una aplicación web

Una aplicación web [4] está basada en la arquitectura cliente – servidor, en donde el cliente es el navegador y el servidor es el servidor web, donde están alojadas las páginas web que son mostradas en el navegador y que interactúan con el usuario, tal como se muestra en la Figura 1.2.

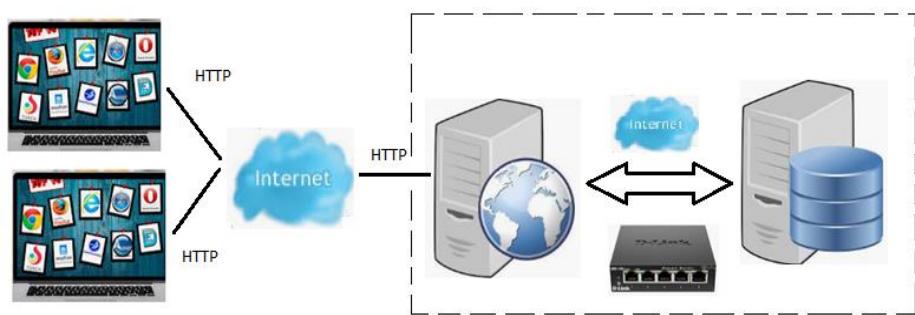


Figura 1.3 Arquitectura Aplicaciones Web [4].

Las aplicaciones web se pueden ejecutar en un solo servidor o en varios servidores [4]. La comunicación entre el navegador y el servidor puede ser mediante una intranet o Internet, y la base de datos puede estar alojada en la misma máquina donde está instalado el servidor web o en otro servidor de forma local o mediante Internet [4]. Hay varias formas en que una aplicación web puede ser diseñada e implementada, utilizando desde una máquina de escritorio hasta grandes servidores, sin olvidar la posible implementación en la nube [6].

Además, para crear aplicaciones web hay que seguir ciertos estándares de desarrollo web que se encuentran en el Consorcio World Wide Web (W3C, por las siglas en inglés de World Wide Web Consortium) [5], que es una organización donde los miembros desarrollan estándares para el desarrollo web, como son el Lenguaje de Marcas de Hipertexto (HTML, por las siglas en inglés de HyperText

Markup Language), las Hojas de Estilo en Cascada (CSS, por las siglas en inglés de Cascading Style Sheets), el lenguaje JavaScript web api, internacionalización (Para crear aplicaciones web que funcionen en diferentes culturas o idiomas), accesibilidad, gráficos, los protocolos de comunicación, los formatos de texto para paso de información, entre otros [5].

#### 1.5.4 JavaScript

En la década de los 90, cuando Internet estaba comenzando a hacer protagonista en el mundo tecnológico, comienza a surgir el lenguaje de programación JavaScript [3], con el objetivo de ayudar al navegador para que las páginas sean más interactivas con el usuario, y aumentar el rendimiento de acceso a las aplicaciones web.

Algunas características de JavaScript son [3]: es un lenguaje de programación interpretado, no necesita compilador; puede ejecutar diferentes acciones, tanto en el lado del cliente como en el lado del servidor; es orientado a objetos, aunque la declaración de clases y objetos es muy diferente al estándar de ciertos lenguajes programación como Java [3]; y es de tipado blando; es decir, que no se necesita que se declaren el tipo de datos de las variables antes de ser utilizarlas. Con el pasar de los años, JavaScript ha evolucionado inmensamente, ya que en la actualidad hay muchos frameworks o librerías que ayudan a los programadores a crear páginas web fuertes y robustas con mejor rendimiento y acceso de solicitudes y respuestas entre navegador y servidor web [3].

En la actualidad existen librerías derivadas de JavaScript que se utilizan para crear aplicaciones web, robustas, eficientes, presentables y de gran rendimiento; como ReactJS [7] enfocada a la visualización o interfaz de usuario, usando componentes que pueden ser reutilizables, interactivos y renderizados en páginas web, y que, junto con NodeJS

[8], que es un entorno de ejecución de código abierto en el lado del servidor; permiten crear y ejecutar aplicaciones web de una forma extremadamente rápida.

#### **1.5.5 JavaScript Como Backend y Frontend**

Muchos roles intervienen en el desarrollo de aplicaciones web modernas. Dos de ellos son [9]: El frontend y el backend. En conjunto, éstos cubren el desarrollo de cómo funciona y cómo se ve una página web [5]. Un desarrollador que conozca JavaScript puede programar a plenitud una aplicación web, pudiendo ejercer el rol tanto del frontend como del backend. Es por esto que JavaScript ha ido evolucionando, incorporando nuevas tecnologías para solucionar los problemas propios de estos dos roles, volviéndose así compatible con todos los navegadores web modernos y la mayoría de los sitios web.

En el frontend, JavaScript cumple el papel de hacer interactivas las aplicaciones web, especialmente mejorando las capacidades de las aplicaciones web de una sola página [9].

En el backend, JavaScript se encarga de otorgar servicios de API RESTful, que generalmente se relaciona con la modificación de bases de datos, procesos de autenticación, almacenamiento de imágenes y otras funciones no relacionadas con la interfaz de usuario [9].

#### **1.5.6 ReactJS y NodeJS**

ReactJS [7] es una biblioteca escrita en JavaScript, cuyo objetivo es crear interfaces de usuario interactivas de manera fácil y sencilla, crea y utiliza componentes haciendo que el código sea declarativo, predecible y reutilizable, para poder ser renderizados sin complicaciones en una aplicación web [5]. Por otra parte, NodeJS [8], un entorno de ejecución orientado a eventos asíncronicos, construido

netamente con JavaScript, que gestiona conexiones concurrentes y libres de bloqueo de procesos, en el cual se puede crear una aplicación muy rápida y escalable. Estas dos herramientas juntas, permiten crear aplicaciones web robustas y escalables, de una manera facil y sencilla.

## CAPÍTULO 2

### 2. METODOLOGÍA

En este capítulo se expone el diseño de la API que integra los MHMRB, en base al procesamiento de la información recolectada de los distintos involucrados en el desarrollo del proyecto.

#### 2.1 Plan de recolección de datos

Para desarrollar la API destinada a integrar los MHMRB, se tuvo que analizar la documentación de ambos módulos por separado, y luego analizar toda esta información de forma conjunta; es decir, que se analizó el sistema que se forma con los dos módulos puestos en producción. En paralelo a dicho análisis, se entrevistó a los desarrolladores de ambos módulos por separado, con el fin de entender el funcionamiento completo de estos para así llevar a cabo la tarea de integrarlos. En la Tabla 2.1, se detallan las personas que fueron entrevistadas.

Módulo de historias médicas		Módulo de Registro de beneficiarios	
Desarrolladores	Área de desarrollo	Desarrolladores	Área de desarrollo
Branny Chito	Back-end	Erick Pérez	Back-end
Israel Zurita	Front-end	Julián Adams	Front-end

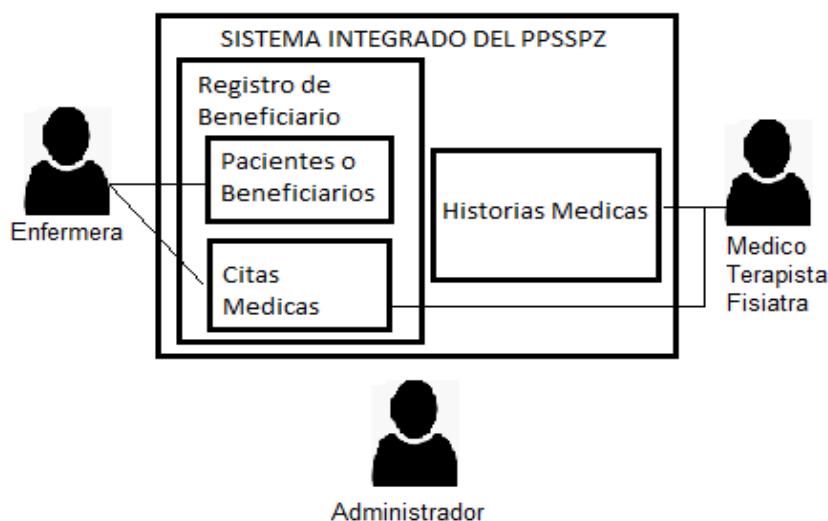
**Tabla 2.1 Personal entrevistado para recolectar datos.**

#### 2.2 Fiabilidad de los datos

Se puede asegurar que los datos obtenidos son fiables porque fueron recopilados a través de constantes entrevistas con los desarrolladores de ambos módulos, los mismos que presentaban diversas documentaciones las cuales fueron estudiadas cuidadosamente previo a las entrevistas mencionadas, para corroborar el conocimiento necesario para desarrollar la solución.

## 2.3 Análisis de los datos

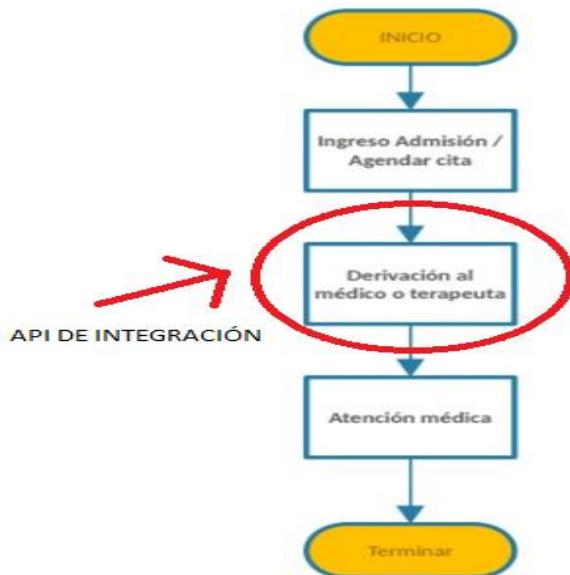
Una vez de haber analizado y resumido la información investigada, se procedió a crear un esquema global de cómo funcionan los MHMRB, para clarificar nuestras dudas y poder diseñar la mejor solución de interacción entre estos dos módulos.



**Figura 2.1 Esquema general Sistema Integrado del PPSSPZ.**

La Figura 2.1 muestra que el módulo de Registro de Beneficiarios tiene dos submódulos: (1) Pacientes o Beneficiarios, que es donde la enfermera registra a los pacientes que se van a hacer atender en el PPSSPz, y (2) Citas Médicas, que es donde la enfermera agenda las citas médicas que van a hacer atendidas por un Médico. Por su parte, según la Figura 2.1, el módulo de Historias Médicas no tiene submódulos y es donde el Médico registra la historia médica del paciente o beneficiario en el momento que lo está atendiendo. Podemos ver en la Figura 2.1 que no hay un enlace o integración entre el submódulo Citas médicas, del módulo Registro de Beneficiarios, con el módulo de Historias Médicas.

Siguiendo con nuestro análisis de datos pudimos determinar que ambos módulos intervienen en la sistematización de un proceso de negocio, el cual se muestra en la Figura 2.2.



**Figura 2.2 Proceso derivación de cita médica a atención médica [11].**

En el proceso de la Figura 2.2 se observa que el flujo de actividades establece que la actividad *Derivación al médico o terapeuta* se hace después de la actividad *Ingreso Admisión / Agendar cita* y antes de la *Atención médica*. La actividad del negocio *Ingreso Admisión / Agendar cita* está soportada por el Módulo de Registro de Beneficiarios, mientras que la actividad de negocio *Atención médica* está soportada por el Módulo de Historias Médicas, quedando por fuera de la automatización actual para el PPSSPz, la actividad de negocio *Derivación al médico o terapeuta*. En este sentido, tal como se señala en la Figura 2.2, el API a desarrollar a través de este proyecto integrador tiene por objetivo integrar los módulos antes mencionados para lograr la sistematización completa del proceso de la Figura 2.2.

## 2.4 Propuesta de solución – Arquitectura de la API

La propuesta de solución para realizar la integración de los módulos de Historias Médicas y Registro de Beneficiarios se documenta a través del Modelo de Arquitectura 4+1 Vistas de Kruchten [10], en el cual se manifiesta que la arquitectura de la solución de un proyecto de software se divide en cinco vistas, las cuales se muestran en la Figura 2.3.

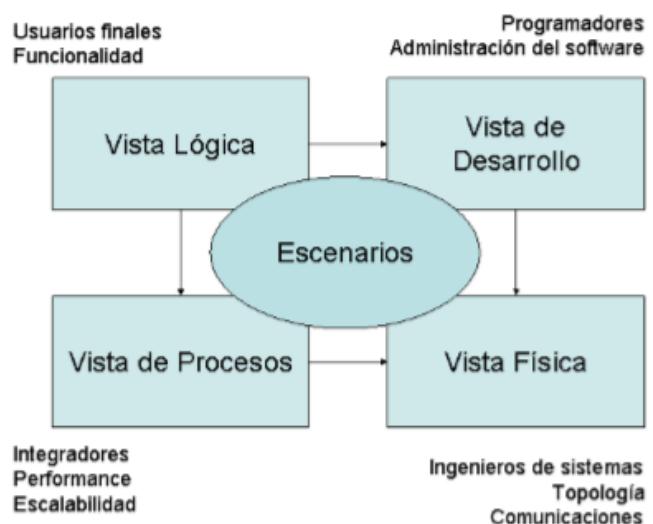


Figura 2.3 Modelo de “4+1” vistas [10].

### 2.4.1 Vista de Escenarios

Durante el tiempo que se recolectaron y analizaron los datos sobre los MHMRB del SIPPSSPz, se identificaron los requerimientos funcionales que sirven para integrar dichos módulos. Además, se detectó que existe un rol indirecto llamado Médico, mostrado en la Tabla 2.2, que interactúa directamente con el submódulo Citas Médicas y el módulo Historias Médicas, y que dicha interacción ejecuta los roles que se muestran en la tabla 2.3 siendo estos últimos los principales roles para el diseño del API integrador.

Actor	Descripción
-------	-------------

Médico	Es la persona que brinda la atención médica en el submódulo Citas Médicas del Módulo Registro de Beneficiarios y registra la historia médica del paciente o beneficiario en el módulo de Historias Médicas
--------	--

**Tabla 2.2 Rol indirecto.**

Actor	Descripción
Submódulo Citas Médicas	Es el responsable de manejar la información general de los pacientes o beneficiarios del PPSSPz y de gestionar las citas médicas de estos con los distintos servicios del PPSSPz.
Módulo Historias Médicas	Es el responsable de gestionar toda la información relacionada con las historias médicas que los beneficiarios pueden tener como usuarios de los distintos servicios que ofrece el PPSSPz.

**Tabla 2.3 Roles principales del API.**

Una vez definidos los roles principales se definieron las historias de usuarios como se muestra en la Tabla 2.4.

Código	Descripción
HU001	Conseguir uri destino
HU002	Dirigirme a uri destino
HU003	Conseguir uri origen
HU004	Dirigirme a uri origen
HU005	Conseguir Cita Médica
HU006	Conseguir Usuario que inicio sesión
HU007	Establecer cita médica a atendida

**Tabla 2.4 Historias de Usuario.**

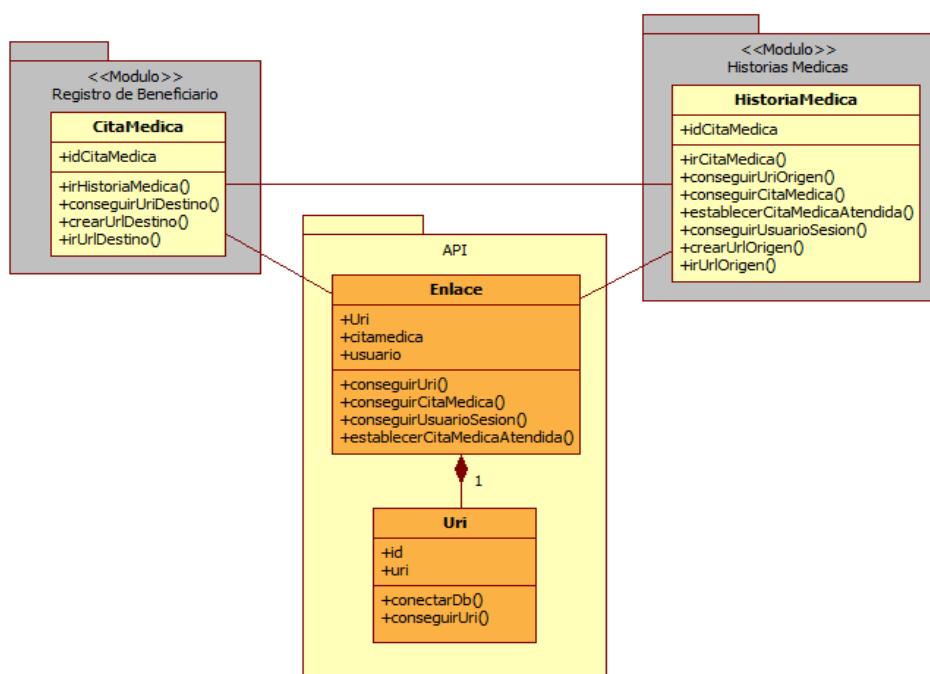
En la Tabla 2.5 se muestra una historia de usuario detallada, donde se especifican el rol, la funcionalidad de la necesidad y los posibles escenarios que se deben considerar cuando se vaya a codificar el API. El resto de las historias de usuario detalladas se muestran en el Anexo A1.

<b>Código</b>	HU001
<b>Rol</b>	Submódulo Citas Médicas
<b>Funcionalidad</b>	Conseguir uri destino
<b>Razón</b>	Para poder dirigirme al módulo Historias Médicas
<b>Escenarios o criterios de validación</b>	<p>Si existe uri destino entonces crear la url destino para dirigirme al módulo de Historias Médicas</p> <p>Si no existe uri destino entonces mostrar un mensaje donde se explique que no se puede continuar</p>

**Tabla 2.5 Descripción detallada historia de usuario Conseguir uri destino.**

#### 2.4.2 Vista Lógica

Esta vista muestra la lógica de funcionamiento del API, la cual es presentada gráficamente en el Diagrama de Clases de la Figura 2.4.

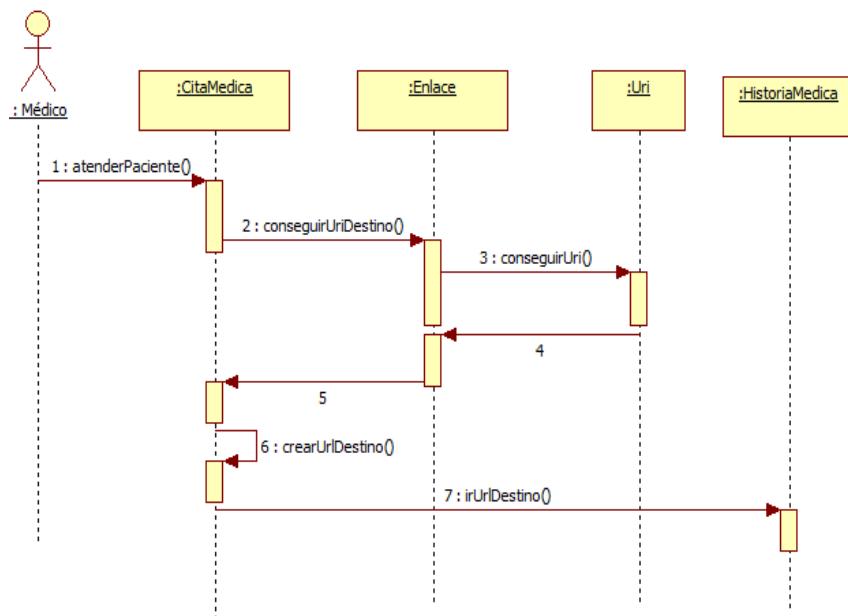


**Figura 2.4 Diagrama de Clases del API.**

El diagrama de clases que nos muestra la Figura 2.4, podemos observar que la clase Uri es la encargada de conectarse a la base de datos y conseguir los recursos uri que se van a necesitar en la clase Enlace. La clase Enlace es la encargada de interactuar con el submódulo Citas Médicas, del módulo Registro de Beneficiarios, y el módulo Historias Médicas, mediante los métodos conseguirUri, conseguirCitaMedica, conseguirUsuarioSesion, establecerCitamedicaAtendida, por lo tanto, esta clase es la que provee los recursos uri a los MHMRB para que exista el enlace de

integración entre los MHMRB como se muestra arriba del API en la Figura 2.4.

La Figura 2.5 nos muestra el diagrama de secuencia para la historia de usuario HU002, dirigirme a uri destino, cuando el rol indirecto Médico va a atender un paciente y escoge una cita médica del submódulo Citas Médicas en el módulo de Registro de Beneficiarios, y este módulo, a su vez pide al API el uri destino; entonces el API mediante la clases Enlace y Uri devuelve el recurso uri al submódulo Citas Médicas del módulo Registro de Beneficiarios, y así poder dirigirse al módulo Historias Médicas.



**Figura 2.5 Diagrama de Secuencia para historia de usuario HU002.**

La Figura 2.6 nos muestra el diagrama de secuencia para la historia de usuario HU004, dirigirme a uri origen, cuando el rol indirecto Médico ha finalizado la atención médica y procede a registrar el historial médico del paciente o beneficiario en el módulo de Historias Médicas, y este módulo, a su vez, pide al API el uri origen; entonces el API mediante la

clases `Enlace` y `Uri` devuelve el recurso `uri` al módulo Historias Médicas, y así poder dirigirse submódulo Citas Médicas en el módulo de Registro de Beneficiarios.

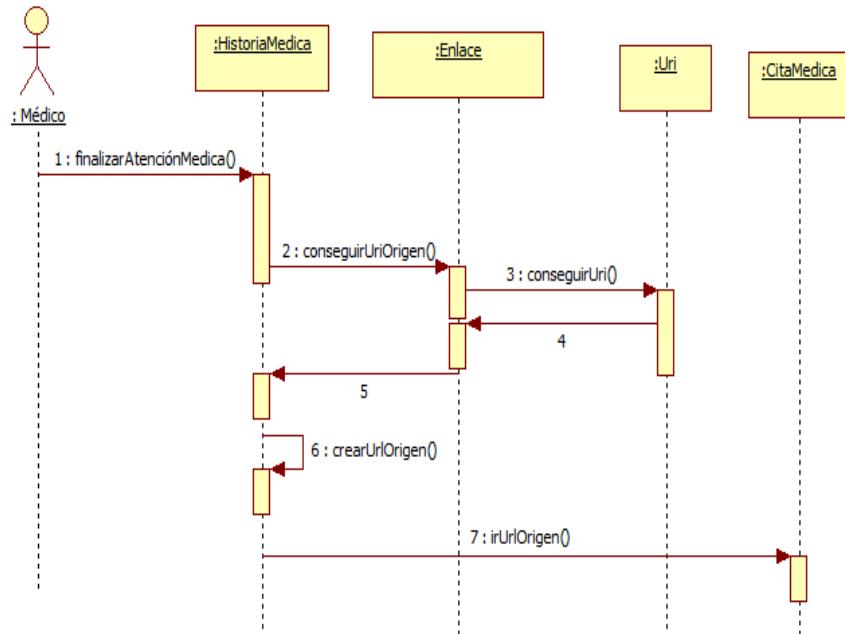


Figura 2.6 Diagrama de Secuencia para historia de usuario HU004.

El modelo conceptual de la base de datos de la Figura 2.7, nos muestra la tabla `api_recurso`, que almacena información de los recursos `uri`, que son básicamente las direcciones `url` del grupo de páginas que se encuentran definidas en el módulo de Historias Médicas y la `url` de la página cita médica del submódulo Citas Médicas del módulo Registro de Beneficiarios; y que por lo tanto estos recursos `uri` van hacer fundamentales en el API, para que haya la integración entre los MHMRB.

api_recurso	
<code>id</code>	INTEGER
<code>nombre</code>	VARCHAR
<code>uri</code>	VARCHAR

Figura 2.7 Modelo Conceptual de la base de datos del API.

### 2.4.3 Vista de Desarrollo

Para el desarrollo del API de integración nos basamos en el patrón de arquitectura Modelo Vista Controlador (MVC). En la Figura 2.8 se presenta el Diagrama de Componentes del API de integración.

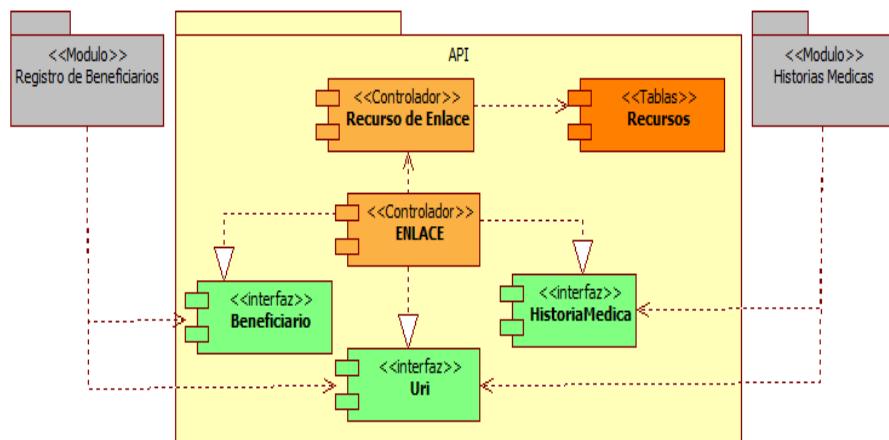


Figura 2.8 Diagrama de Componentes.

La capa Vista, representada en la Figura 2.8 mediante los componentes <<interfaz>> de color verde, está conformada por Beneficiario, Uri, HistoriaMedica, y son las interfaces del API que van a interactuar con los MHMRB.

La capa Controlador, representada en la Figura 2.8 mediante los componentes <<Controlador>> de color naranja, está conformada por enlace y recurso de enlace, son los encargados de recuperar los recursos de la base de datos y enviarlos a las vistas.

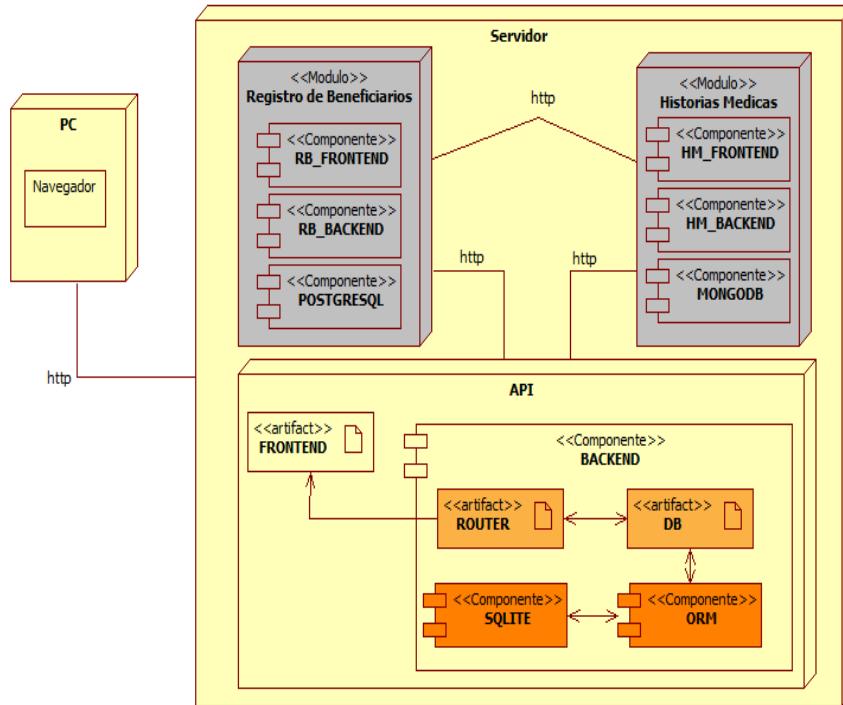
La capa Modelo, representada en la Figura 2.8 mediante el componente <<Tablas>> de color naranja oscuro, que solo contiene el componente Recursos, es el encargado de interactuar con la base de datos que contiene la tabla donde están almacenados los recursos uri.

#### **2.4.4 Vista de Procesos**

En esta vista nos encargamos de garantizar que los requerimientos no funcionales de rendimiento y concurrencia sean satisfechos. Para el diseño del API se cuenta con las herramientas tecnológicas NodeJS y PostgreSQL, la cuales nos garantizan una solución a los aspectos de rendimiento y concurrencia. Por lo anterior, para esta vista no se especifica algún tipo de diagrama.

#### **2.4.5 Vista de Física**

El API de integración está desplegado en el servidor, específicamente en el lado del Backend; pero, en base a los conceptos de integración de sistemas [2] necesitamos considerar el funcionamiento del sistema como un todo, entre API de integración, módulo de Historias Médicas y el módulo de Registro de Beneficiarios. Por eso, en la Figura 2.9 se muestra el diagrama de despliegue para el SIPPSSPZ, donde podemos apreciar que están los programas y/o tecnologías utilizadas por los usuarios finales y los programas, librerías y/o tecnologías que se van a utilizar en el lado del servidor.



**Figura 2.8 Diagrama de Despliegue.**

## 2.5 Plan de Implementación

Para gestionar y poner en funcionamiento nuestro proyecto, vamos a utilizar el marco de trabajo SCRUM por ser una metodología ágil y que puede ser adaptable a cualquier proyecto. La planificación de las actividades o tareas que tendremos que desarrollar semanalmente se muestran en la Tabla 2.6.

Actividades	Inicio	Fin
Levantar y probar funcionamiento Módulo Registro de Beneficiario y Módulo Historias Médicas.	24/06/2019	30/06/2019
Implementar Historia de Usuario HU001, HU002.	01/07/2019	03/07/2019
Implementar Historia de Usuario HU003, HU004.	04/07/2019	07/07/2019
Implementar Historia de Usuario HU005, HU006.	08/07/2019	10/07/2019
Implementar Historia de Usuario HU007 y pruebas funcionales de todas las historias de usuarios implantadas.	11/07/2019	14/07/2019
Corrección de errores encontrados en las pruebas funcionales de todas las historias de usuarios implantadas.	15/07/2019	18/07/2019
Pruebas funcionales entre los módulos Historias Médicas y Registro de Beneficiarios y API.	19/07/2019	21/07/2019
Corrección de inconvenientes presentados en las pruebas funcionales.	22/07/2019	25/07/2019
Revisión y corrección de novedades presentadas en el funcionamiento del módulo Registro de Beneficiarios.	26/07/2019	30/07/2019

Revisión y corrección de novedades presentadas en el funcionamiento del módulo Historias Médicas.	31/07/2019	04/08/2019
Pruebas funcionales entre Registro de Beneficiarios, Historias Médicas y API.	05/08/2019	08/08/2019
Corrección de inconvenientes presentados en las pruebas funcionales.	09/08/2019	11/08/2019

**Tabla 2.6 Planificación de Actividades.**

# CAPÍTULO 3

## 3. IMPLEMENTACIÓN DE LA SOLUCION

En este capítulo explicaremos la solución que implantamos para integrar los MHMRB. Comenzaremos revisando la implementación de nuestro API integrador, para luego revisar las modificaciones adicionales a las interfaces de los MHMRB y, por último, el análisis de resultados de todo el SIPPSSPz.

### 3.1 API integrador

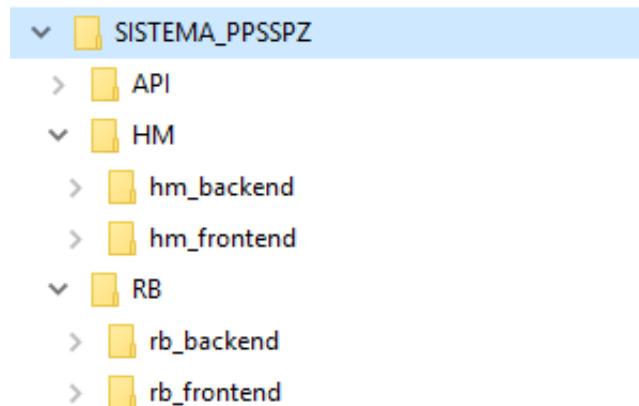
Para implementar el API integrador se tuvo que analizar los MHMRB, como se definió en el Capítulo 2 de este documento, para especificar y definir los requerimientos funcionales necesarios del API integrador; es por ello, que lo que primero se hizo fue definir el entorno del SIPPSSPz, para luego proceder a crear el API integrador. El API integrador es un API REST intermedio que interactúa con los MHMRB, enviando y recibiendo información, para que dichos módulos puedan procesar la información que reciben del API y poder navegar entre ellos.

#### 3.1.1 Entorno físico del SIPPSSPz

Para definir el entorno del SIPPSSPz, se tuvo que analizar los MHMRB para conocer cómo estaban creados y la forma en que se ejecutan; es por esto que se detectó que cada módulo estaba compuesto por un *backend* y un *frontend* que se ejecutan por separado; además, se detectó que los MHMRB no se podían ejecutar al mismo tiempo, porque había conflicto con los puertos de salida entre los *backend* y *frontend* de cada módulo.

Después de detectar los inconvenientes, se procedió a definir el entorno, estructura física del SIPPSSPz, como se muestra en la Figura 3.1, donde se aprecia una carpeta global `SISTEMA_PPSSPZ`, que

contiene al API y los backend y frontend de los MHMRB por separado, como se describe a continuación:



**Figura 3.1 Estructura física del sistema PPSSPz.**

- Carpeta `SISTEMA_PPSSPZ`, es la carpeta principal que contiene todas carpetas y archivo de codificación que definen al SIPPSSPz.
- Carpeta `API`, que contiene los archivos de codificación del API.
- Carpeta `HM`, para el módulo Historias Médicas, que a su vez está compuesta por las subcarpetas `hm_backend` y `hm_frontend`, donde están los archivos de codificación de cada uno respectivamente.
- Carpeta `RB`, para el módulo Registro de Beneficiarios, que a su vez está compuesta por las subcarpetas `rb_backend` y `rb_frontend`, donde están todos los archivos de codificación de cada uno respectivamente.

Además, se tuvo que reasignar los puertos de los backend y frontend de cada módulo para que funcionen conjuntamente y sin conflictos, tal como se muestra a continuación:

- Puerto 3003 para el API.
- Módulo Registro de Beneficiarios:
  - Puerto 3001 para el frontend.

- Puerto 5000 para el backend.
- Módulo Historias Médicas:
  - Puerto 3002 para frontend.
  - Puerto 3000 para backend.

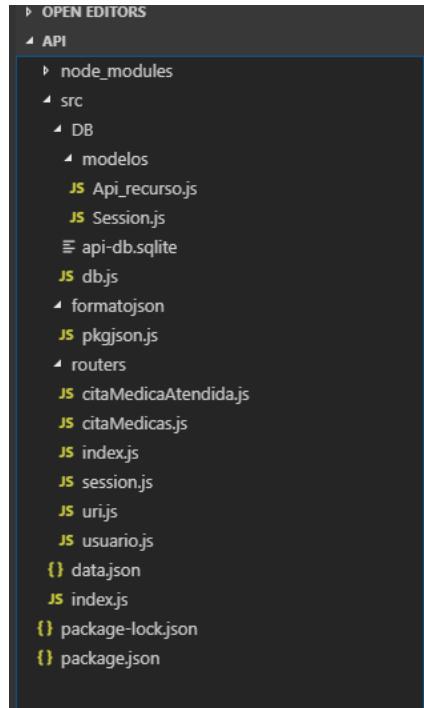
Por último, se configuró el API y los backends y frontends de cada módulo para que se ejecuten con el comando `npm run start`.

### 3.1.2 Implementación API

Nuestro API integrador es un API REST que interactúa con los frontend y backend de los MHMRB y que fue implementado utilizando las herramientas tecnológicas que se describen a continuación:

- NodeJS, un entorno de ejecución en el lado del servidor para crear el API REST de nuestra API, y que interactúa enviando y recibiendo datos de los MHMRB.
- SQLite, como base de datos liviana para almacenar los recursos URI que el API necesita para interactuar con los MHMRB.
- Sequelize como ORM, mapeo relacional Objeto, para poder interactuar entre la base de datos SQLite y NodeJS.
- JSON como formato de intercambio de datos, para poder intercambiar datos entre el API integrador y los MHMRB.

En la Figura 3.2 mostramos la estructura física del API integrador, que a su vez consta de dos carpetas principales `DB` y `routers`. Estas carpetas corresponden a los artefactos `ROUTER` y `BD` del componente `BACKEND` del API que se muestra en el diagrama de despliegue de la Figura 2.9 del Capítulo 2.



**Figura 3.2 Estructura archivos del API.**

La Figura 3.3 muestra la carpeta DB que contiene el archivo de base de datos, api-db.sqlite, así como los archivos de definición de las tablas y el código para poder autenticarse y tener acceso a la base de datos.

```

> OPEN EDITORS
  API
    > node_modules
    > src
      > DB
        > modelos
          > Api_recurso.js
          > Session.js
          > api-db.sqlite
        > db.js
        > formatojson
        > pkgjson.js
      > routers
        > citaMedicaAtendida.js
        > citaMedicas.js
        > index.js
        > session.js
        > uri.js
        > usuario.js
      > data.json
      > index.json
    > package-lock.json
    > package.json

```

```

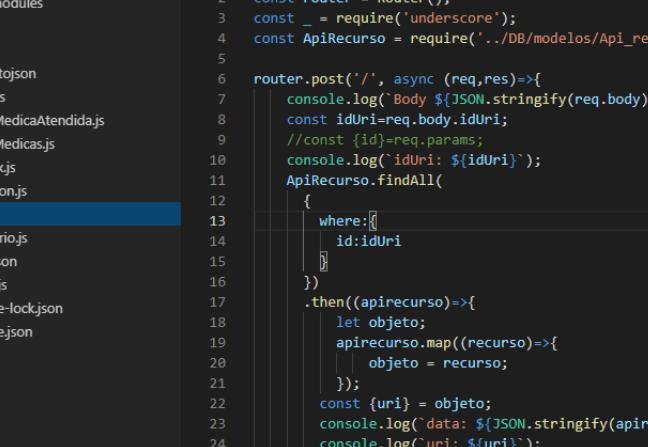
src > DB > db.js > ...
1
2  const path = require('path');
3  const Sequelize = require('sequelize');
4  const db = new Sequelize('api-db','','',{
5    dialect: 'sqlite',
6    storage: path.join(__dirname,'api-db.sqlite'),
7    define: {
8      freezeTableName: true
9    }
10 });
11 //Probar la conexion
12 /*db.authenticate().then(()=>{
13   | console.log('Connectado');
14 })
15 .catch(err => {
16   | console.error('Error ',err);
17 });*/
18
19 module.exports = db;
20

```

**Figura 3.3 Carpeta de DB.**

La Figura 3.4 muestra la carpeta routers, la cual contiene los enlaces del API integrador que van a hacer la comunicación entre el API y los

MHMRB. Además, se puede apreciar el código del archivo `uri.js` que el API define y que es utilizado por estos módulos para poder navegar entre ellos.



The image shows a code editor interface with a sidebar on the left and a main editor area on the right.

**Left Sidebar (Open Editors):**

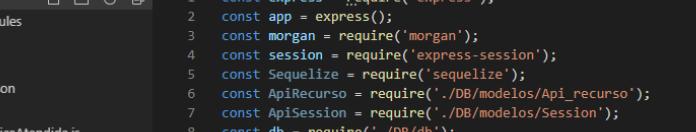
- > OPEN EDITORS
- ✓ API
  - > node\_modules
  - < src
    - > DB
    - > formatojson
    - < routers
      - JS `citaMedicaAtendida.js`
      - JS `citaMedicas.js`
      - JS `index.js`
      - JS `session.js`
      - JS `uri.js`** (highlighted in blue)
      - JS `usuario.js`
    - { } `data.json`
    - { } `package-lock.json`
    - { } `package.json`

**Main Editor Area:**

```
src > routers > JS uri.js > router.post('/') callback
  1  const { Router } = require('express');
  2  const router = Router();
  3  const _ = require(' underscore');
  4  const ApiRecurso = require('../DB/modelos/Api_recurso');
  5
  6  router.post('/', async (req,res)=>{
  7    console.log(`Body ${JSON.stringify(req.body)}`);
  8    const idUri=req.body.idUri;
  9    //const [id]=req.params;
 10   console.log(`idUri: ${idUri}`);
 11   ApiRecurso.findAll(
 12     {
 13       where:[
 14         {
 15           id:idUri
 16         }
 17     }
 18   ).then((apirecurso)=>{
 19     let objeto;
 20     apirecurso.map((recurso)=>{
 21       objeto = recurso;
 22     });
 23     const {uri} = objeto;
 24     console.log(`data: ${JSON.stringify(apirecurso)}`);
 25     console.log(`uri: ${uri}`);
 26     return res.send(uri);
 27   })
 28   .catch((err) => {
 29     res.send(err);
 30   }
 31 )
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50
 51
 52
 53
 54
 55
 56
 57
 58
 59
 60
 61
 62
 63
 64
 65
 66
 67
 68
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105
 106
 107
 108
 109
 110
 111
 112
 113
 114
 115
 116
 117
 118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130
 131
 132
 133
 134
 135
 136
 137
 138
 139
 140
 141
 142
 143
 144
 145
 146
 147
 148
 149
 150
 151
 152
 153
 154
 155
 156
 157
 158
 159
 160
 161
 162
 163
 164
 165
 166
 167
 168
 169
 170
 171
 172
 173
 174
 175
 176
 177
 178
 179
 180
 181
 182
 183
 184
 185
 186
 187
 188
 189
 190
 191
 192
 193
 194
 195
 196
 197
 198
 199
 200
 201
 202
 203
 204
 205
 206
 207
 208
 209
 210
 211
 212
 213
 214
 215
 216
 217
 218
 219
 220
 221
 222
 223
 224
 225
 226
 227
 228
 229
 230
 231
 232
 233
 234
 235
 236
 237
 238
 239
 240
 241
 242
 243
 244
 245
 246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295
 296
 297
 298
 299
 300
 301
 302
 303
 304
 305
 306
 307
 308
 309
 310
 311
 312
 313
 314
 315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433
 434
 435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491
 492
 493
 494
 495
 496
 497
 498
 499
 500
 501
 502
 503
 504
 505
 506
 507
 508
 509
 510
 511
 512
 513
 514
 515
 516
 517
 518
 519
 520
 521
 522
 523
 524
 525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550
 551
 552
 553
 554
 555
 556
 557
 558
 559
 559
 560
 561
 562
 563
 564
 565
 566
 567
 568
 569
 569
 570
 571
 572
 573
 574
 575
 576
 577
 578
 579
 579
 580
 581
 582
 583
 584
 585
 586
 587
 587
 588
 589
 589
 590
 591
 592
 593
 594
 595
 596
 597
 598
 599
 599
 600
 601
 602
 603
 604
 605
 606
 607
 608
 609
 609
 610
 611
 612
 613
 614
 615
 616
 617
 618
 619
 619
 620
 621
 622
 623
 624
 625
 626
 627
 628
 629
 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 639
 640
 641
 642
 643
 644
 645
 646
 647
 648
 649
 649
 650
 651
 652
 653
 654
 655
 656
 657
 658
 659
 659
 660
 661
 662
 663
 664
 665
 666
 667
 668
 669
 669
 670
 671
 672
 673
 674
 675
 676
 677
 678
 679
 679
 680
 681
 682
 683
 684
 685
 686
 687
 687
 688
 689
 689
 690
 691
 692
 693
 694
 695
 696
 697
 698
 699
 699
 700
 701
 702
 703
 704
 705
 706
 707
 708
 709
 709
 710
 711
 712
 713
 714
 715
 716
 717
 718
 719
 719
 720
 721
 722
 723
 724
 725
 726
 727
 728
 729
 729
 730
 731
 732
 733
 734
 735
 736
 737
 738
 739
 739
 740
 741
 742
 743
 744
 745
 746
 747
 748
 749
 749
 750
 751
 752
 753
 754
 755
 756
 757
 758
 759
 759
 760
 761
 762
 763
 764
 765
 766
 767
 768
 769
 769
 770
 771
 772
 773
 774
 775
 776
 777
 778
 779
 779
 780
 781
 782
 783
 784
 785
 786
 787
 787
 788
 789
 789
 790
 791
 792
 793
 794
 795
 796
 797
 798
 799
 799
 800
 801
 802
 803
 804
 805
 806
 807
 808
 809
 809
 810
 811
 812
 813
 814
 815
 816
 817
 818
 819
 819
 820
 821
 822
 823
 824
 825
 826
 827
 828
 829
 829
 830
 831
 832
 833
 834
 835
 836
 837
 838
 839
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 849
 850
 851
 852
 853
 854
 855
 856
 857
 858
 859
 859
 860
 861
 862
 863
 864
 865
 866
 867
 868
 869
 869
 870
 871
 872
 873
 874
 875
 876
 877
 878
 879
 879
 880
 881
 882
 883
 884
 885
 886
 887
 887
 888
 889
 889
 890
 891
 892
 893
 894
 895
 896
 897
 898
 899
 899
 900
 901
 902
 903
 904
 905
 906
 907
 908
 909
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929
 929
 930
 931
 932
 933
 934
 935
 936
 937
 938
 939
 939
 940
 941
 942
 943
 944
 945
 946
 947
 948
 949
 949
 950
 951
 952
 953
 954
 955
 956
 957
 958
 959
 959
 960
 961
 962
 963
 964
 965
 966
 967
 968
 969
 969
 970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 979
 980
 981
 982
 983
 984
 985
 986
 987
 987
 988
 989
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026
 1027
 1028
 1029
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1078
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1088
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1098
 1099
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1188
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1198
 1199
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1248
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1258
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1269
 1270
 1271
 1272
 1273
 1274
 1275
 1276
 1277
 1278
 1278
 1279
 1280
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1288
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1298
 1298
 1299
 1299
 1300
 1301
 1302
 1303
 1304
 1305
 1306
 1307
 1308
 1308
 1309
 1310
 1311
 1312
 1313
 1314
 1315
 1316
 1317
 1318
 1318
 1319
 1320
 1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328
 1328
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1338
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1348
 1349
 1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1358
 1359
 1360
 1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1368
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1398
 1399
 1399
 1400
 1401
 1402
 1403
 1404
 1405
 1406
 1407
 1408
 1408
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1418
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1428
 1429
 1430
 1431
 1432
 1433
 1434
 1435
 1436
 1437
 1438
 1438
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1448
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457
 1458
 1458
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1488
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1498
 1499
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1508
 1509
 1510
 1511
 1512
 1513
 1514
 1515
 1516
 1517
 1518
 1518
 1519
 1520
 1521
 1522
 1523
 1524
 1525
 1526
 1527
 1528
 1528
 1529
 1530
 1531
 1532
 1533
 1534
 1535
 1536
 1537
 1538
 1538
 1539
 1540
 1541
 1542
 1543
 1544
 1545
 1546
 1547
 1548
 1548
 1549
 1550
 1551
 1552
 1553
 1554
 1555
 1556
 1557
 1558
 1558
 1559
 1560
 1561
 1562
 1563
 1564
 1565
 1566
 1567
 1568
 1568
 1569
 1570
 1571
 1572
 1573
 1574
 1575
 1576
 1577
 1578
 1578
 1579
 1580
 1581
 1582
 1583
 1584
 1585
 1586
 1587
 1588
 1588
 1589
 1590
 1591
 1592
 1593
 1594
 1595
 1596
 1597
 1598
 1598
 1599
 1599
 1600
 1601
 1602
 1603
 1604
 1605
 1606
 1607
 1608
 1608
 1609
 1610
 1611
 1612
 1613
 1614
 1615
 1616
 1617
 1618
 1618
 1619
 1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1648
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1658
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1668
 1669
 1670
 1671
 1672
 1673
 1674
 1675
 1676
 1677
 1678
 1678
 1679
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1688
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1698
 1699
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1708
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1718
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1758
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1778
 1779
 1780
 1781
 1782
 1783
 1784
 1785
 1786
 1787
 1788
 1788
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1798
 1799
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1808
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1818
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1828
 1829
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1848
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1858
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1868
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1888
 1889
 1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1898
 1899
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907
 1908
 1908
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1918
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1928
 1929
 1930
 1931
 1932
 1933
 1934
 1935
 1936
 1937
 1938
 1938
 1939
 1940
 1941
 1942
 1943
 1944
 1945
 1946
 1947
 1948
 1948
 1949
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957
 1958
 1958
 1959
 1960
 1961
 1962
 1963
 1964
 1965
 1966
 1967
 1968
 1968
 1969
 1970
 1971
 1972
 1973
 1974
 1975
 1976
 1977
 1978
 1978
 1979
 1980
 1981
 1982
 1983
 1984
 1985
 1986
 1987
 1988
 1988
 1989
 1990
 1991
 1992
 1993
 1994
 1995
 1996
 1997
 1998
 199
```

Figura 3.4 Carpeta routers.

La Figura 3.5 muestra la cabecera del archivo principal `index.js` que está en la carpeta `src`; este archivo es el que se ejecuta cuando el API comienza a funcionar.



The image shows a code editor interface with the following details:

- Open Editors:** OPEN EDITORS
- API:** node\_modules, src, DB, formatojson, routers, **citaMedicaAtendida.js**, **citaMedicas.js**, **index.js** (highlighted in blue), **session.js**, **uri.js**, **usuario.js**, **datajson**, **index.js** (highlighted in blue), package-lock.json, package.json.
- src > index.js:** The code is as follows:

```
1  const express = require('express');
2  const app = express();
3  const morgan = require('morgan');
4  const session = require('express-session');
5  const Sequelize = require('sequelize');
6  const ApiRecurso = require('../DB/modelos/Api_recurso');
7  const ApiSession = require('../DB/modelos/Session');
8  const db = require('../DB/db');
9
10 //-----
11
12 //propiedades de configuracion
13 //SE LE PONE 3001.
14 app.set('puerto',process.env.PORT || 3003);
15 //mostrar en consola lo que el servidor recibe
16 // Configurar cabeceras y cors
17 //para no tener inconveniente con CORS
18 app.use((req, res, next) => {
19   res.header('Access-Control-Allow-Origin', '*');
```

### Figura 3.5 Archivo principal del API.

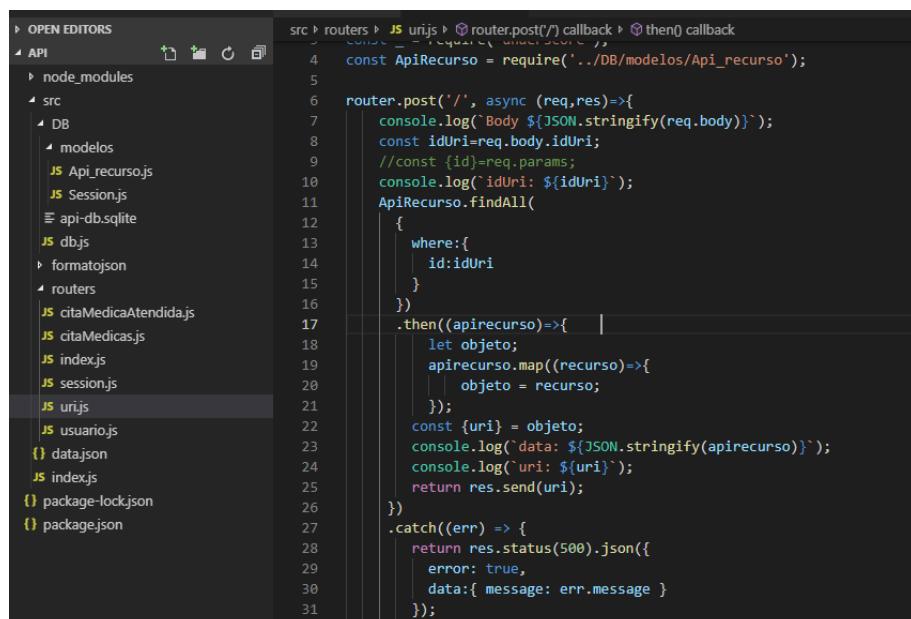
La Figura 3.6 muestra las tablas de la base de datos SQLite que el API utiliza para integrar los MHMRB. La tabla `Api_recurso` almacena los

recursos URI, que son la URL específicas de las páginas web que ya están creadas en los MHMRB. La tabla Session almacena las sesiones que se crean cuando un usuario inicia sesión al sistema PPSSPz.

Session		api_recurso	
sid	VARCHAR	id	INTEGER
userId	VARCHAR	nombre	VARCHAR
expires	DATETIME	uri	VARCHAR
data	VARCHAR	createdAt	DATETIME
createdAt	DATETIME	updatedAt	DATETIME
updatedAt	DATETIME		

Figura 3.6 Tablas de la Base de Datos del API.

La Figura 3.7 muestra el código del API; donde el API recibe el código de la URI, `idUri`, que es enviada de los frontends de los MHMRB; para luego acceder a la base de datos y retornar la URI que se está solicitando.



```

OPEN EDITORS
API
  node_modules
    src
      DB
        modelos
          Api_recurso.js
          Session.js
          api-db.sqlite
        db.js
      formatojson
    routers
      citaMedicaAtendida.js
      citaMedicas.js
      index.js
      session.js
      uri.js
      usuario.js
    data.json
    index.js
    package-lock.json
    package.json

src > routers > JS uri.js > router.post('') callback > then() callback
  1  const _ = require('underscore');
  2  const ApiRecurso = require('../DB/modelos/Api_recurso');
  3
  4  router.post('', async (req,res)=>{
  5    console.log(`Body ${JSON.stringify(req.body)}`);
  6    const idUri=req.body.idUri;
  7    //const {id}=req.params;
  8    //console.log(`idUri: ${idUri}`);
  9    ApiRecurso.findAll(
 10      {
 11        where:{
 12          id:idUri
 13        }
 14      }
 15    )
 16    .then((apiRecurso)=>{
 17      let objeto;
 18      apiRecurso.map((recurso)=>{
 19        objeto = recurso;
 20      });
 21      const {uri} = objeto;
 22      console.log(`data: ${JSON.stringify(apiRecurso)}`);
 23      console.log(`uri: ${uri}`);
 24      return res.send(uri);
 25    })
 26    .catch((err) => {
 27      return res.status(500).json({
 28        error: true,
 29        data:{ message: err.message }
 30      });
 31    });

```

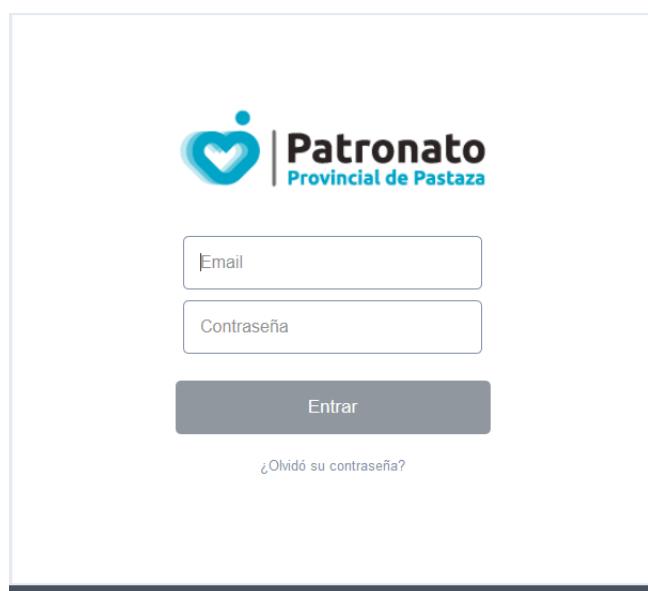
Figura 3.7 Código para acceder a la tabla Api\_recurso.

### 3.2 Actualizaciones adicionales a los módulos historias médicas y registro de beneficiarios

Una de las actividades fundamentales en la integración de sistemas es analizar y hacer las pruebas de funcionamiento de los componentes discretos, como se definió en el Capítulo 1, sección 1.5.1 del marco teórico. Es por ello que se tuvo que hacer actualizaciones adicionales a las interfaces de los MHMRB que presentaban inconvenientes en el funcionamiento y almacenamiento de la información en la base de datos.

#### 3.2.1 Acceso al sistema PPSSPZ y módulo registro de beneficiarios

En el módulo de registro de beneficiario está la interfaz de iniciar sesión, donde los usuarios van a poder ingresar sus credenciales y poder acceder a las opciones del sistema del PPSSPz, como se muestra en la Figura 3.8.



**Figura 3.8 Interfaz de inicio de sesión del sistema PPSSPz.**

Una vez que el usuario se ha autenticado con éxito, se presenta la interfaz de Seleccionar Unidad, como se muestra en la Figura 3.9. Esta interfaz fue modificada para que presente las unidades físicas,

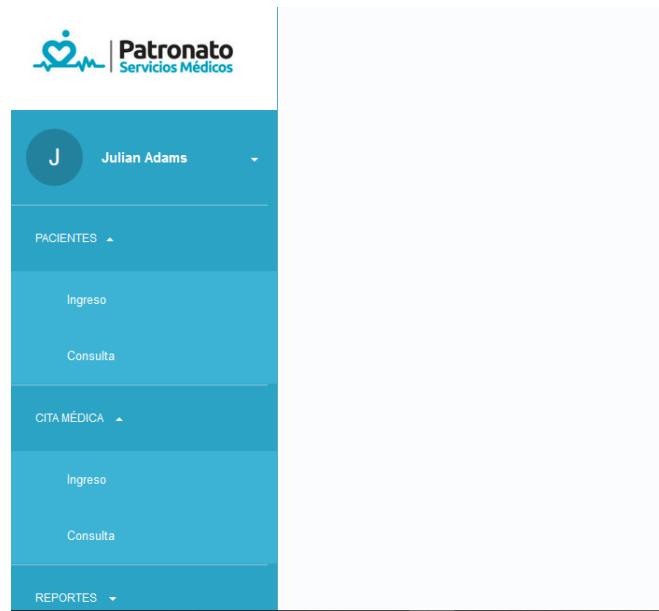
consultorio tipo A o CITET, donde se prestan los servicios médicos, y no los roles de los usuarios que acceden al sistema.

### Seleccionar Unidad



**Figura 3.9 Interfaz de Unidad.**

Una vez que el usuario ha escogido la unidad, el sistema muestra al usuario el menú principal (ver Figura 3.10) del sistema PPSSPz, así como también el nombre del usuario que esta autenticado.



**Figura 3.10 Menú principal del sistema PPSSPz.**

Se puede observar que hay dos submódulos que son Pacientes y Cita Médica.

La Figura 3.11 muestra la interfaz de registro de pacientes. Esta interfaz fue analizada y no presentaba ninguna novedad por lo que no se hizo ninguna modificación.

### Ingreso de paciente

1. Datos Generales    2. Procedencia    3. Ocupación    4. Datos de referencia    5. Final

**Nombres:**

**Apellidos:**

**Identificación:**   No tiene

**Lugar de nacimiento:**

**Fecha de nacimiento:**  **Estado civil:**

**Nacionalidad:**  **Grupo**    
cultural:

**Sexo:**  **Teléfono:**

**Siguiente**

**Figura 3.11 Interfaz de Ingreso y modificación de Pacientes.**

La Figura 3.12 muestra la interfaz de consulta de pacientes registrados en el sistema PPSSPz; al pulsar el botón editar, el sistema carga los datos en la interfaz de registro de pacientes para poder modificarlos. Al analizar esta interfaz, no se encontró ninguna novedad por lo que no se hizo ninguna modificación.

## Consulta de Paciente

- Seleccione la fila para ver los datos del paciente
- Posicione el cursor sobre la columna por la que desea filtrar

# Histori...	Nombres	Apellidos	Identificación	Teléfono	Acciones
2	Lusmila	Tapia	2020302020	0930102030	<a href="#">Editar</a>
3	Evelyn	Tapia	4520342020	0930145630	<a href="#">Editar</a>
4	Jose	Lara	4530342050	0920145630	<a href="#">Editar</a>
5	Cesar	Ramirez Avila	0918233792	2617545	<a href="#">Editar</a>
1	Juan	Fernandez	1010102020	0910102030	<a href="#">Editar</a>
6	AXAXaxAX	axAXaxAX	0923454657		<a href="#">Editar</a>

1

**Figura 3.12 Interfaz de Consulta de Paciente.**

La Figura 3.13 muestra la interfaz de Ingreso de Citas Médicas. A esta interfaz se le hicieron varios cambios, tal como se detalla a continuación:

- Se agregó el campo unidad proyecto, por lo que una cita médica puede ser registrada en cualquiera de las unidades existentes del PPSSPz, donde un paciente se puede hacer atender.
- Se quitó el bloque de consulta detallada de los pacientes y se agregó el campo paciente.
- Se agregó el campo observación para ingresar alguna descripción de la cita médica.
- Se reescribió el código del campo Valor, para poder consultarlos en la lista, ya que no se consultaban.
- Se actualizó el código para poder grabar los datos de la cita médica, ya que al momento de registrar datos de la Cita Médica no se almacenaban en la base de datos.

## Ingreso de citas



Formulario para el ingreso de citas médicas. Los campos son:

- Unidad Proyecto: Seleccionar Unidad Proyecto
- Paciente: Seleccionar Paciente
- Servicio: Seleccionar servicio
- Medico: Seleccionar Medico
- Fecha: 06-08-2019
- Hora: 04:55
- Valor: Seleccionar valor
- Observación: Texto de observación

**Figura 3.13 Interfaz de Ingreso de Citas Médicas.**

La Figura 3.14 muestra la interfaz de Citas Programadas. A esta interfaz se le hicieron las siguientes actualizaciones:

## Citas programadas



Formulario para la consulta de citas programadas. Los campos son:

- Tipo: Medicina General
- Fecha: 06-08-2019

Tabla de citas programadas:

# Histori...	Nombres	Apellidos	Identific...	Estado	Hora	Acciones
Ingrese	Ingrese nombre	Ingrese apellidos	Ingrese			 
2	Lusmila	Tapia	2020302...	Pendiente	05:07	

**Figura 3.14 Interfaz de Citas programadas.**

- Se actualizó el código del campo tipo, ya que inicialmente consultaba todos los tipos de atención como son: medicina general, odontología, psicología, terapia física y terapia de lenguaje; y no había coherencia con el tipo de atención del

Médico que había iniciado sesión. Por ejemplo, si el tipo de atención del Médico es medicina general, entonces cuando él inicia sesión en el sistema y vaya a atender las citas médicas programadas, sólo se deberían mostrar las que sean medicina general y no todas; por esto, se reescribió el código para que sólo se presenten los tipos de atención que están relacionados con el tipo de atención del Médico que ha iniciado sesión.

- Se actualizó el código del campo fecha, por lo que cuando se escogía una fecha no se mostraban las citas programadas.
- Se quitó el botón `marcar cita` en la columna de `Acciones`, por lo que no tenía sentido cambiarle el estado de la cita médica a atendido, si el sistema lo debe hacer automáticamente cuando el Médico registra el historial médico del paciente.
- Al botón `eliminar`, forma de x de color rojo de la columna `Acciones`, se le reescribió el código para que no elimine el registro de la base de datos, sino que sólo cambie el estado y quede el registro como un histórico.
- Al botón `ficha médica`, forma de persona color azul de la columna `Acciones`, se le agregó el código para que interactúe con el API integrador y poder integrar las citas programadas con el registro de la ficha médica del paciente del módulo de Historias Médicas. Aquí podemos mencionar el siguiente proceso al pulsar este botón:
  - Citas Programadas pide al API, según el tipo de atención, el URI de la interfaz de Historia médica donde se va a registrar el historial médico del paciente.
  - El API toma el código del URI, que le envía la interfaz de Citas Programadas, y consulta en la base de datos; si existe el código, el API envía la URI; en caso contrario, envía un error.
  - La interfaz de Citas Programas recibe la URI del API; si no hubo error, entonces forma una URL agregando como

parámetros el código de la cita programa y la sesión del usuario que esta autenticado, y hace la redirección a la URL que ha formado del módulo de historia médicas.

### 3.2.2 Módulo Historias Médicas

El módulo de Historias Médicas tiene las interfaces para registrar el historial médico de los pacientes que se hacen atender en el PPSSPZ. En la Figura 3.15 podemos observar los tipos de atención que existen en el sistema PPSSPz, como son: Medicina General, Terapia Física, Terapia de Lenguaje, Psicología y Odontología. A todas estas interfaces se las revisó y se procedió a agregar el código para poder integrarlas con la interfaz de Citas Programadas del módulo de Registro de Beneficiarios.

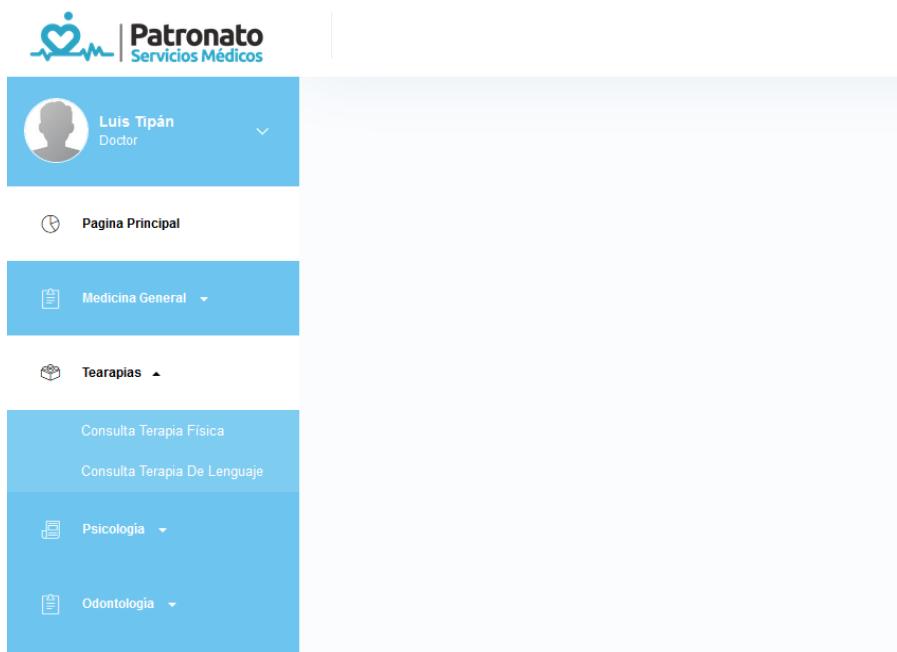
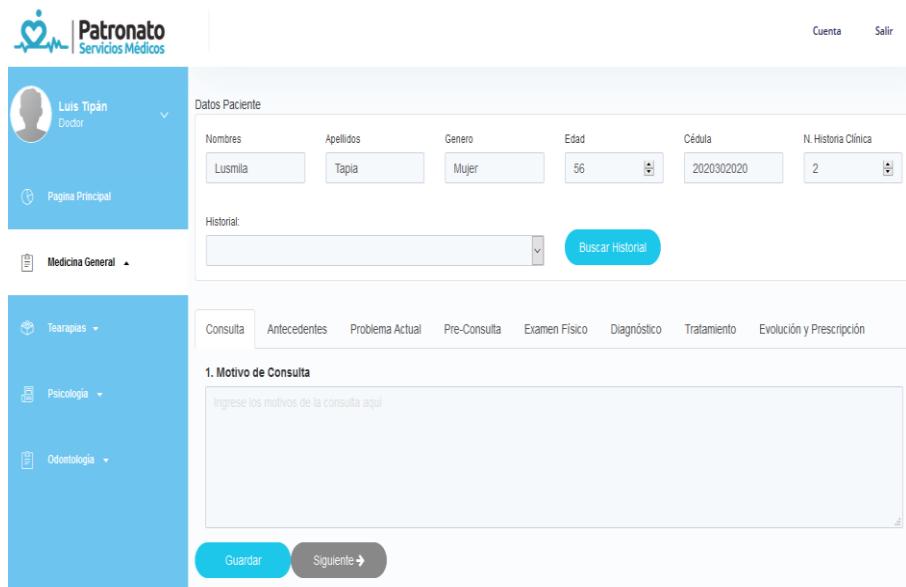


Figura 3.15 Menú principal del Módulo Historias Médicas.

La Figura 3.16 muestra la interfaz de registro de historia médica del tipo de atención Medicina General. A esta interfaz se le modificó para

que pueda recibir los datos del paciente que viene de la interfaz Cita Programada del módulo Registro de Beneficiario.



The screenshot shows the Patronato Servicios Médicos web interface. The top navigation bar includes the logo 'Patronato Servicios Médicos', 'Cuenta', and 'Salir'. On the left, a sidebar menu is visible with options: 'Pagina Principal', 'Medicina General' (selected), 'Terapias', 'Psicología', and 'Odontología'. The main content area is titled 'Datos Paciente' and contains fields for 'Nombres' (Lusmila), 'Apellidos' (Tapia), 'Genero' (Mujer), 'Edad' (56), 'Cédula' (2020302020), and 'N. Historia Clínica' (2). Below these fields is a search bar labeled 'Historial' with a 'Buscar Historial' button. A tab navigation bar at the top of the main form includes 'Consulta', 'Antecedentes', 'Problema Actual', 'Pre-Consulta', 'Examen Físico', 'Diagnóstico', 'Tratamiento', and 'Evolución y Prescripción'. A large text area is labeled '1. Motivo de Consulta' with the placeholder 'Ingrese los motivos de la consulta aquí'. At the bottom of the form are two buttons: 'Guardar' and 'Siguiente'.

**Figura 3.16 Interfaz de Historia Médica Medicina General.**

Cuando la interfaz Medicina General es llamada por la interfaz Cita Programada del módulo de Registro de beneficiario, ésta procede hacer lo siguiente:

- Interfaz Medicina General recibe los parámetros de la URL que son el código de la cita programa y la sesión del usuario conectado.
- Interfaz Medicina General pide al API los datos del paciente o beneficiario pasándole como parámetro el código de la cita programada.
- El API recibe como parámetro el código de la cita programada y le pide al backend del módulo de Registro de Beneficiarios los datos del paciente; una vez que el API recibe los datos del paciente, éstos son enviados a la interfaz de Medicina General.

- Cuando la interfaz Medicina General recibe los datos del paciente, entonces los carga en los campos de la interfaz.

Una vez cargados los datos del paciente en el navegador, el usuario Médico procede a ingresar la historia médica del paciente. Al finalizar el ingreso de la historia médica del paciente, el usuario o Médico pulsa el botón guardar y se procesa lo siguiente:

- Interfaz Medicina General guarda la información de la historia médica del paciente en la base de datos.
- Interfaz Medicina General llama al API, enviándole como parámetro el código de la cita programada, para que actualice el estado de la cita programa a Atendida.
- API recibe como parámetro el código de la cita programada y le pide al backend del módulo de Registro de Beneficiarios que ponga el estado “A” a la cita programada.
- API notifica la actualización del estado de la cita programada a la interfaz Medicina General.
- Interfaz Medicina General pide al API la URI de la interfaz Cita Programada del módulo de Registro de Beneficiarios, enviándole como parámetro el código de la URI.
- El API recibe como parámetro el código de la URI, consulta en la base de datos y procede a enviar la URI.
- Interfaz Medicina General recibe la URI del API y crea la URL para regresar a la interfaz de Citas Programadas del módulo de Registro de Beneficiarios.

Podemos concluir que las demás interfaces como Terapia física, Terapia de Lenguaje, Psicología y Odontología, tienen el mismo procedimiento que la interfaz de Medicina General, por lo que se mostraran las interfaces en el Anexo A2.

### 3.3 Análisis de resultados

Aquí vamos a revisar la funcionalidad del sistema del PPSSPz, compuesto por los MHMRB y el API integrador, siendo implícito para el usuario que inicie sesión, la interacción y navegación entre los componentes del sistema PPSSPz.

#### 3.3.1 Funcionalidad

Cuando el usuario inicia sesión al sistema PPSSPz, el sistema diferencia entre los roles Médico y Asistente. En las interfaces de beneficiarios, tanto el usuario con rol Asistente como con rol Médico, pueden ingresar y consultar pacientes, beneficiarios; pero en las interfaces de citas médicas van a tener diferentes privilegios, de tal forma que un usuario con rol asistente puede crear una cita médica, pero no la puede atender, porque el sistema no le va a permitir. En la interfaz de citas programadas, Figura 3.14, sólo va a mostrar las citas médicas a los usuarios que tengan rol de Médico y que estén relacionadas con el tipo de atención que el Médico brinda; como, por ejemplo, si un Médico brinda el tipo de atención medicina general, entonces cuando él inicie sesión al sistema PPSSPz y vaya a la interfaz de citas programadas, esta interfaz le mostrará sólo las citas médicas que pertenezcan a medicina general. Esto mismo se aplicará con el resto de los tipos de atención.

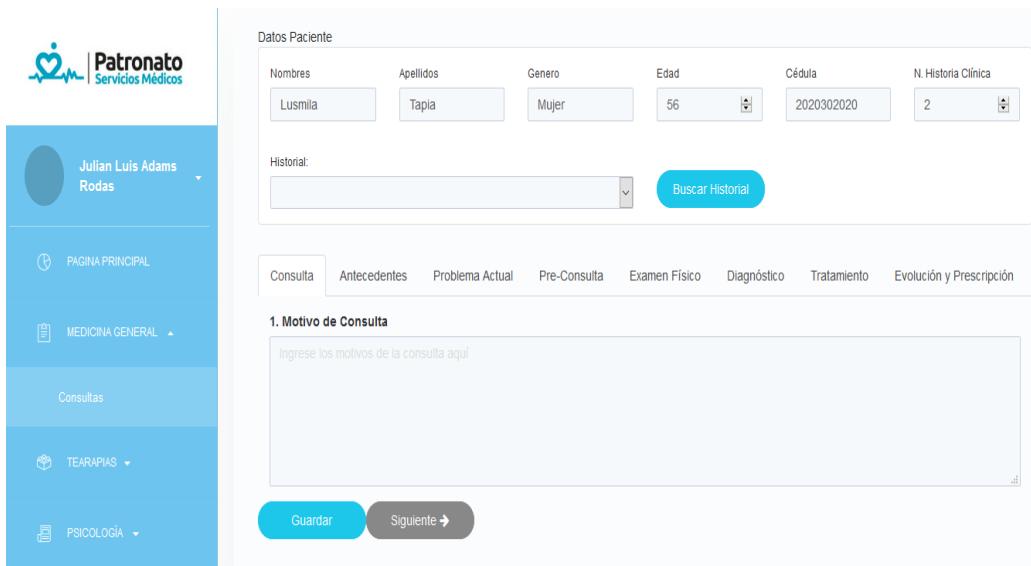
La Figura 3.17 muestra al usuario que ha iniciado sesión, su nombre en la esquina superior izquierda, y que va a atender una cita médica. Usando la interfaz citas programadas, cuando da clic en el botón para atender la cita, el sistema lo redirige a la interfaz de historia médica dependiendo del tipo de atención que, para la Figura 3.17, es medicina general.



The screenshot shows the Patronato Servicios Médicos application. On the left, a sidebar menu includes 'Cerrar sesión', 'PACIENTES', 'CITA MÉDICA', 'Ingreso', and 'Consulta'. The main content area is titled 'Citas programadas' and displays a table with one row of data. The table columns are: # Historial, Nombres, Apellidos, Identificación, Estado, Hora, and Acciones. The data row shows: 2, Lusmila, Tapia, 2020302..., Pendiente, 08:21, and a row of icons for edit and delete. A blue circular badge with the number 1 is in the bottom right corner of the main content area.

**Figura 3.17 Usuario con Rol atendiendo una cita médica.**

La Figura 3.18 muestra al usuario Médico, nombre se presenta en la esquina superior izquierda, que está atendiendo la cita médica y que ahora va a registrar el historial médico del paciente en la interfaz de historia médica tipo de atención medicina general.



The screenshot shows the Patronato Servicios Médicos application. On the left, a sidebar menu includes 'PAGINA PRINCIPAL', 'MEDICINA GENERAL', 'Consultas', 'TEARAPIAS', and 'PSICOLOGÍA'. The main content area is titled 'Datos Paciente' and shows a form with fields: Nombres (Lusmila), Apellidos (Tapia), Genero (Mujer), Edad (56), Cédula (20203020), and N. Historia Clínica (2). Below this is a 'Historial' section with a search bar and a 'Buscar Historial' button. At the bottom, there are tabs for 'Consulta', 'Antecedentes', 'Problema Actual', 'Pre-Consulta', 'Examen Físico', 'Diagnóstico', 'Tratamiento', and 'Evolución y Prescripción'. A section titled '1. Motivo de Consulta' with a text area for 'Ingresar los motivos de la consulta aquí' is shown. At the bottom are 'Guardar' and 'Siguiente →' buttons.

**Figura 3.18 Usuario con Rol Médico registrando la historia médica.**

Por lo expuesto en los casos anteriores, se quiere demostrar que el usuario que inicie sesión para usar el sistema PPSSPz, en su interacción, validación, navegación y flujo de información, solo percibe

la usabilidad como un solo sistema y no por módulos separados, tal como estaba antes de este proyecto integrador.

### **3.3.2 Datos antes de la integración**

- Los módulos se ejecutaban independientemente.
- Los MHMRB no funcionaban conjuntamente, porque el frontend y backend de cada módulo utilizaban el mismo puerto de salida.
- Cuando un usuario iniciaba sesión, se presentaban los roles, mas no el sitio físico donde se iban a realizar las citas médicas.
- Las interfaces del módulo Citas Médicas no funcionaban completamente.
- El nombre del usuario sólo aparecía en el módulo de registro de Beneficiario.
- Las interfaces del módulo de Historia Médicas almacenaban en la base de datos información de un solo beneficiario.

### **3.3.3 Resultados después de la integración**

- Los MHMRB funcionan conjuntamente, ya que se volvieron asignar los puertos como se muestra en la sesión 3.1.1 de este capítulo.
- Todas las interfaces de los dos módulos funcionan correctamente.
- Se mejoró la amigabilidad del sistema con el usuario, debido a que anteriormente existían campos en las diversas interfaces de los distintos módulos que manipulaban información de forma inconsistente, especialmente al momento de actuar en conjunto con los demás módulos del sistema.

## CONCLUSIONES Y RECOMENDACIONES

### Conclusiones

Con lo expuesto en este documento podemos concluir que se logró integrar los módulos de Historias Médicas y de Registro de Beneficiarios (MHMRB) del Sistema Integrado del Patronato Provincial de Servicio Social de Pastaza (SIPPSSPz).

Ahora los MHMRB funcionan como un único sistema que automatiza el flujo de información entre los dos módulos; además, el SIPPSSPz está listo para ser utilizado por el personal que labora en el PPSSPz.

Los MHMRB fueron integrados exitosamente como resultado de haber seguido todas las actividades del proceso de Integración de Sistemas establecido por el SEBoK [2], el cual fue una gran ayuda desde el punto de vista metodológico.

Finalmente, a través del diseño de la arquitectura el API integrador, se sentaron las bases para que los futuros módulos que se desarrollen para ir completando el SIPPSSPz, se puedan integrar a este sistema de una forma efectiva y consistente.

### Recomendaciones

Para los nuevos módulos que se vayan a desarrollar se recomienda que utilicen el código de integración que se definió en los MHMRB como son:

- Para el módulo Registro de Beneficiarios, actualizar el método `onClickIrAsistencia` que se muestra en la Figura R.1, para que agreguen el código uri de las nuevas interfaces de historias médicas.

```

 291
 292 >   onChangeFechaConsulta(fecha){ ... }
 293
 294
 295   async onClickIrAsistencia(id_cita){
 296     let idUri=0;
 297     const id_servicio = this.state.id_servicio;
 298     //alert(id_servicio);
 299     if(id_servicio==1)//Medicina General
 300     {
 301       idUri = 2;
 302     }
 303     else if(id_servicio==2)//Odontología...
 304     {
 305       idUri = 3;
 306     }
 307     else if(id_servicio==3)//Terapia de Lenguaje...
 308     {
 309       idUri = 4;
 310     }
 311     else if(id_servicio==4)//Terapia Física...
 312     {
 313       idUri = 5;
 314     }
 315     else if(id_servicio==5)//Psicología...
 316     {
 317       idUri = 6;
 318     }
 319     else if(id_servicio==6)//Equinoterapia...
 320     {
 321       idUri = 7;
 322     }
 323     else if(id_servicio==7)//Estimulación Temprana...
 324     {
 325       idUri = 8;
 326     }
 327     else...
 328     {
 329       idUri = 9;
 330     }
 331     else...
 332     {
 333       idUri = 10;
 334     }
 335     else...
 336     {
 337       idUri = 11;
 338     }
 339     /*const { body } = await getUriHm({ idUri});*/

```

**Figura R.1 Procedimiento onClickIrAsistencia.**

- En las nuevas interfaces de historias médicas, deben utilizar el procedimiento `obtenerUbi_fun`, para obtener los datos del beneficiario y el usuario que ha iniciado sesión; como se muestra en la Figura R.2.

```

 88   }
 89
 90   obtieneUbi_fun()
 91   {
 92     //obtengo la url que actual
 93     const url_string = window.location.href;
 94     //alert(url_string);
 95     //Lo convierto objeto URL para obtener el query
 96     const Url = require('url-parse');
 97     const url = new Url(url_string);
 98     const parametro = url.query;
 99     const cadena = (parametro || parametro.length>1)?parametro.substring(4):'undefined';
100
101   if ([cadena]=='undefined')
102   {
103     window.location.assign('http://localhost:3000/');
104   }
105   else
106   {
107     const id_usuario = Number(cadena.substring(0,5));
108     const id_cita = Number(cadena.substring(5,15));
109     console.log(`Usuario: ${id_usuario} CitaMedica: ${id_cita}`);
110
111     const body1 = {id_usuario:id_usuario};
112     fetch('http://localhost:3003/api/usuario/',
113       { method: 'POST',
114         body:JSON.stringify(body1),
115       }
116     );
117   }
118 }
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144

```

**Figura R.2 Procedimiento obtenerUbi\_fun.**

- El procedimiento `obtenerUbi_fun`, debe ser definido en el método `componentDidMount` que ReactJS define cuando la interfaz se está inicializando, como se muestra en la Figura R.3

```

src > pages > MedicinaGeneral > NuevaConsulta > JS index.js > NuevaConsulta > obtieneUbi
154     identificacion: bene.identificacion,
155     edad: edad,
156     genero: bene.sexo,
157     cedula: bene.identificacion,
158     hcclinica: (bene.id_beneficiario).toString()
159   );
160   );
161   );
162   );
163   );
164   );
165   );
166   );
167   );
168   );
169   );
170   );
171   );
172   );
173   );
174   );
175   );
176   );
177   );
178   );
179   );
180   );

```

**Figura R.3 Método componentDidMount.**

También se debe crear el módulo de acceso y permisos, donde se definen las interfaces de usuarios, roles, opciones, usuarios por roles y roles por opciones; ya que estas interfaces van a ser útil para que el administrador del SIPPSSPZ pueda crear nuevos accesos a las nuevas personas que vayan a utilizar el SIPPSSPZ

También se debe crear una interfaz que se llame consulta de historias médicas detallada, donde se muestre un histórico detallado de las historias médicas de un paciente o beneficiario. Esta interfaz puede ser accedida de la siguiente manera:

- Cuando un Médico quiere revisar el historial médico de un paciente o beneficiario antes de atender una cita médica en el módulo de Registro de Beneficiarios; entonces la interfaz debe permitir al Médico buscar un paciente o beneficiario y una vez buscado, mostrar el histórico detallado de las historias médicas de ese paciente o beneficiario
- Cuando el Médico este registrando el historial médico de un paciente o beneficiario en el módulo de Historias Médicas y quiera consultar las historias médicas de ese paciente, entonces la interfaz se debe abrir en una nueva pestaña del navegador y

mostrar los datos del paciente que se está haciendo atender junto con su histórico detallado.

- Después que la interfaz le muestre al Médico el histórico y, a su vez, él escoja y quiera revisar una historia médica específica, entonces la interfaz debe enviar esa historia médica del paciente al módulo de Historias Médicas para que se presente en modo lectura.

Consideramos que todas estas recomendaciones son necesarias porque van a ayudar a que los desarrolladores de las nuevas interfaces de historias médicas las puedan integrar sin mucha dificultad; y también, para complementar o extender el funcionamiento del SIPPSSPz.

## BIBLIOGRAFÍA

- [1] Patronato Provincial de Pastaza, «Patronato Provincial de Pastaza,» [En línea]. Available: <https://www.patronatopastaza.gob.ec/>.
- [2] IEEE Computer Society, I. C. o. S. E. INCOSE y Systems Engineering Research Center, «SEBoK, GUIDE TO THE SYSTEMS ENGINEERING BODY OF KNOWLEDGE,» SEBoK, 12 Octubre 2018. [En línea]. Available: [https://www.sebokwiki.org/wiki/System\\_Integration#Methods\\_and\\_Techniques](https://www.sebokwiki.org/wiki/System_Integration#Methods_and_Techniques). [Último acceso: 01 Junio 2019].
- [3] MDN web docs, «Recurso para desarrolladores, creados para desarrolladores,» developer.mozilla.org, [En línea]. Available: <https://developer.mozilla.org/es/>.
- [4] Instituto Tecnológico de Matehuala, «Programación Web,» 24 Febrero 2015. [En línea]. Available: <https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web/>. [Último acceso: 01 Junio 2019].
- [5] Tim Berners-Lee, «World Wide Web Consortium,» [En línea]. Available: <https://www.w3.org/>. [Último acceso: 01 Junio 2019].
- [6] IBMCloud y IBMCloud, «Infraestructura para la computación en la nube,» IBMCloud, [En línea]. Available: <https://www.ibm.com/mxe/cloud/learn/what-is-a-cloud-server>. [Último acceso: 31 05 2019].
- [7] F. O. Source, «React,» Facebook Open Source, 2019. [En línea]. Available: <https://es.reactjs.org/>. [Último acceso: 31 Mayo 2019].
- [8] L. F. C. Projects, «Node JS,» Linux Foundation Collaborative Projects, [En línea]. Available: <https://nodejs.org/es/about/>. [Último acceso: 30 Mayo 2019].
- [9] C. d. Biblios, «<https://cosasdebiblios.blogspot.com/2018/03/el-front-end-y-el-back-end-entendiendo.html>,» 6 Marzo 2018. [En línea].

- Available: <https://cosasdebiblios.blogspot.com/2018/03/el-front-end-y-el-back-end-entendiendo.html>. [Último acceso: 30 Mayo 2019].
- [10] P. Kruchten, «*Planos Arquitectónicos: El Modelo de “4+1” Vistas de la Arquitectura del Software,*» 1995. [En línea]. Available: [http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4\\_1.pdf](http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf). [Último acceso: 20 Junio 2019].
- [11] J. E. ADAMS ESCOBAR y E. A. PÉREZ ARGUELLO, «*Proyecto Integrador "DESARROLLO DEL MÓDULO DE REGISTRO DE BENEFICIARIOS PARA EL SISTEMA INTEGRADO DEL SERVICIO SOCIAL PROVINCIAL DE PASTAZA,*» Octubre 2018. [En línea]. Available: [https://espolec-my.sharepoint.com/:w/r/personal/lemendoza\\_espol\\_edu\\_ec/\\_layouts/15/Doc.aspx?sourcedoc=%7BB2EFE85E-8D94-436D-9575-F8E99A10445A%7D&file=Reg\\_Benef\\_Version5\\_LEMM.doc&action=default&mobileredirect=true](https://espolec-my.sharepoint.com/:w/r/personal/lemendoza_espol_edu_ec/_layouts/15/Doc.aspx?sourcedoc=%7BB2EFE85E-8D94-436D-9575-F8E99A10445A%7D&file=Reg_Benef_Version5_LEMM.doc&action=default&mobileredirect=true). [Último acceso: Junio 2019].
- [12] W. Recommendation, «*Glosario W3C,*» [En línea]. Available: <https://www.w3.org/2005/03/DOM3Core-es/glosario.html>.

## ANEXOS

### Anexo A1: Historias de Usuarios

<b>Código</b>	HU002
<b>Rol</b>	Submódulo Citas médicas
<b>Funcionalidad</b>	Dirigirme a uri destino
<b>Razón</b>	Para poder registrar el historial médico del paciente
<b>Escenarios o criterios de validación</b>	<p>Si existe el código de cita médica a atender y la sesión de usuario entonces ir al módulo de Historias Médicas</p> <p>Si no existe el código de la cita médica a atender o la sesión de usuario entonces mostrar un mensaje donde se muestre que no se puede ir al módulo de Historias Médicas</p>

<b>Código</b>	HU003
<b>Rol</b>	Módulo Historias Médicas
<b>Funcionalidad</b>	Conseguir uri origen
<b>Razón</b>	Para poder dirigirme al submódulo de Citas Médicas
<b>Escenarios o criterios de validación</b>	<p>Si existe uri origen entonces crear la url origen para dirigirme al submódulo Citas Médicas</p> <p>Si no existe uri origen entonces mostrar un mensaje donde se explique que no se puede continuar</p>

<b>Código</b>	HU004
<b>Rol</b>	Módulo Historias Médicas
<b>Funcionalidad</b>	Dirigirme a uri origen
<b>Razón</b>	Para continuar revisando las citas pendientes
<b>Escenarios o criterios de validación</b>	<p>Si se registró exitosamente el historial médico del paciente entonces dirigirme al submódulo citas médicas</p> <p>Si no se registró exitosamente el historial médico del paciente entonces mostrar un mensaje del error</p>

<b>Código</b>	HU005
<b>Rol</b>	Módulo Historias Médicas
<b>Funcionalidad</b>	Conseguir Cita Médica
<b>Razón</b>	Para conocer cuál es el código del paciente o beneficiario que se está atendiendo
<b>Escenarios o criterios de validación</b>	<p>Si existe código de paciente o beneficiario entonces continuar con el registro del historial médico</p> <p>Si no existe código de paciente o beneficiario entonces mostrar un mensaje que no existe código de paciente o beneficiario y dirigirme al submódulo Citas Médicas</p>

<b>Código</b>	HU006
<b>Rol</b>	Módulo Historias Médicas
<b>Funcionalidad</b>	Conseguir Usuario que inicio sesión
<b>Razón</b>	Para conocer que usuario va a registrar el historial médico del paciente o beneficiario
<b>Escenarios o criterios de validación</b>	<p>Si existe usuario entonces continuar con el registro del historial médico</p> <p>Si no existe usuario entonces mostrar un mensaje que no existe usuario que inicio sesión y dirigirme al submódulo Citas Médicas</p>

<b>Código</b>	HU007
<b>Rol</b>	Módulo Historias Médicas
<b>Funcionalidad</b>	Establecer cita médica a atendida
<b>Razón</b>	Para finalizar el registro del historial médico
<b>Escenarios o criterios de validación</b>	<p>Si se estableció el estado de la cita médica a atendida entonces continuar con la finalización del registro del historial médico del paciente o beneficiario</p> <p>Si no se estableció el estado de la cita médica atendida entonces mostrar un mensaje del error</p>

## ANEXO A2: Interfaces del módulo Historias Médicas

### Anexo A2.1 Interfaz Terapia Física

## Anexo A2.2 Interfaz Terapia de Lenguaje

## Anexo A2.3 Interfaz Psicología


**Julian Luis Adams Rodas**

-  [PAGINA PRINCIPAL](#)
-  [MEDICINA GENERAL](#)
-  [TERAPIAS](#)
-  [PSICOLOGÍA](#)
-  [ODONTOLOGÍA](#)

Consulta Odontológica

**Datos Paciente**

Nombres	Apellidos	Genero	Edad
Juan	Fernandez	Hombre	37
		Cédula	
		1010102020	
		N. Historia Clínica	
		1	

Historial:

**Buscar Paciente** (1)

Antecedentes	Medicamentos	H. Problema Funcional	Anamnesis del Dolor	Exploración Física	Tratamiento	Seguimiento
<b>Antecedentes Personales:</b>	<input type="checkbox"/> Hipertension Arterial <input type="checkbox"/> Diabetes <input type="checkbox"/> Colesterol Alto <input type="checkbox"/> Osteoartritis <input type="checkbox"/> ACV	<input type="checkbox"/> Infarto <input type="checkbox"/> Arritmias <input type="checkbox"/> Cancer <input type="checkbox"/> hepatitis <input type="checkbox"/> Tuberculosis	<input type="checkbox"/> Transfuciones <input type="checkbox"/> Accidentes <input type="checkbox"/> Otros			
<b>Antecedentes Familiares:</b>	<input type="checkbox"/> Hipertension Arterial <input type="checkbox"/> Diabetes	<input type="checkbox"/> Infarto del Miocardio <input type="checkbox"/> Demencia	<input type="checkbox"/> Cancer <input type="checkbox"/> Otros			
<b>Descripción de Antecedentes y Otros</b>						
<p>Ingrese la descripción aquí</p>						

## Anexo A2.4 Interfaz Odontología