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

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

ÍNDICE DE TABLAS

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].

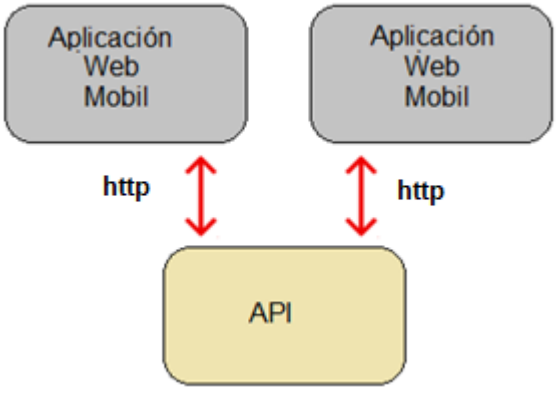


Figura 1.2 Interación del API

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.

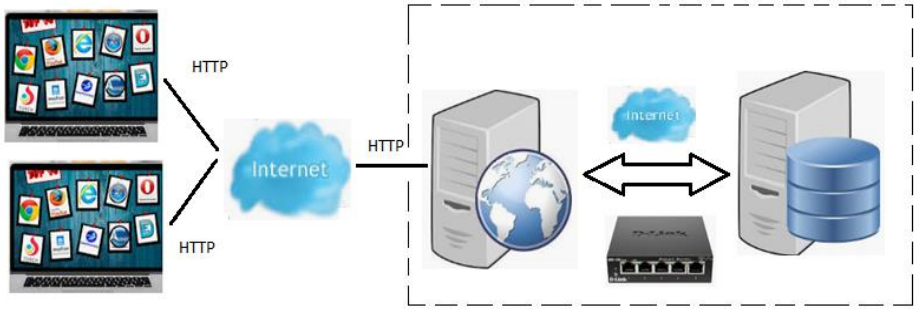


Figura 1.3 Arquitectura Aplicaciones Web [4].

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.

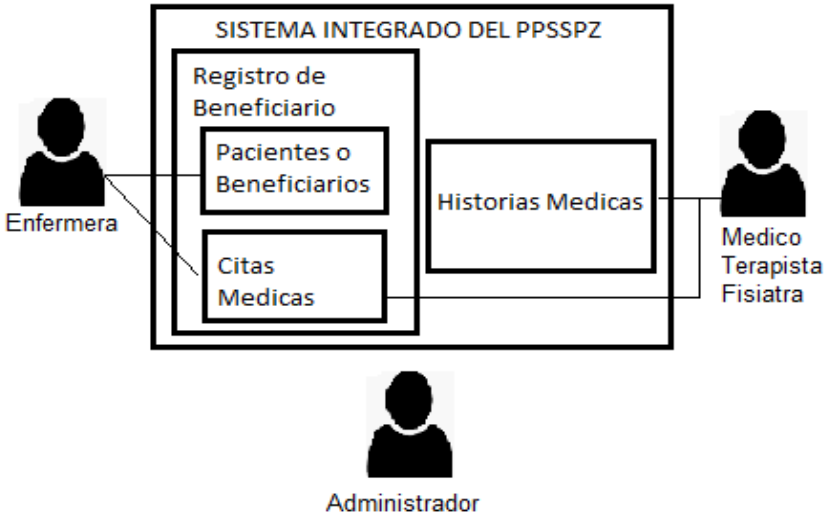


Figura 2.1 Esquema general Sistema Integrado del PPSSPZ.

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.

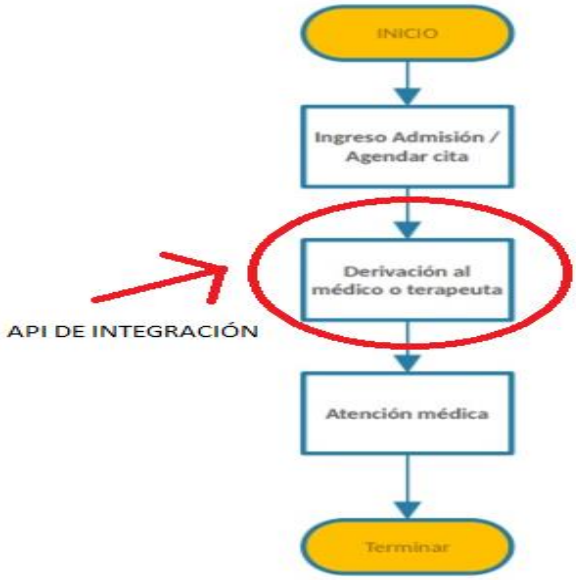


Figura 2.2 Proceso derivación de cita médica a atención médica [11].

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.

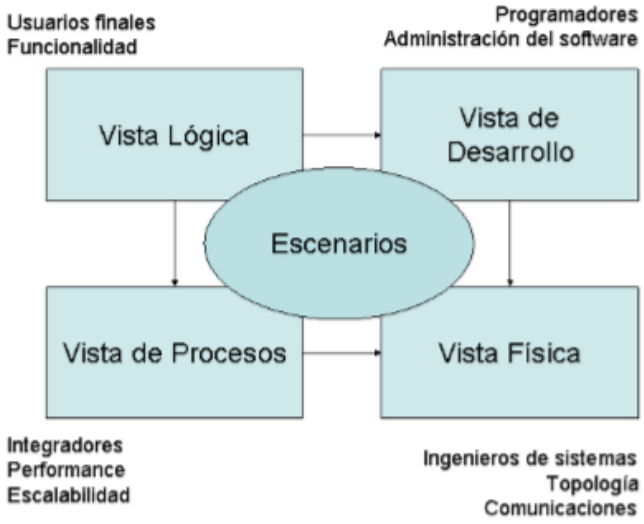


Figura 2.3 Modelo de "4+1" vistas [10].

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.

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

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.

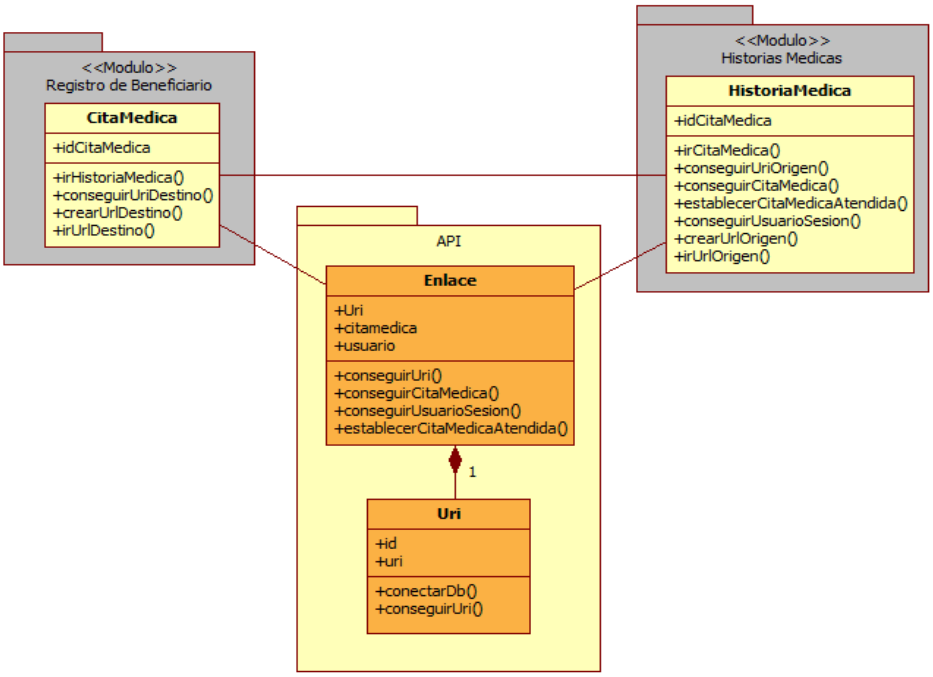


Figura 2.4 Diagrama de Clases del API.

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.

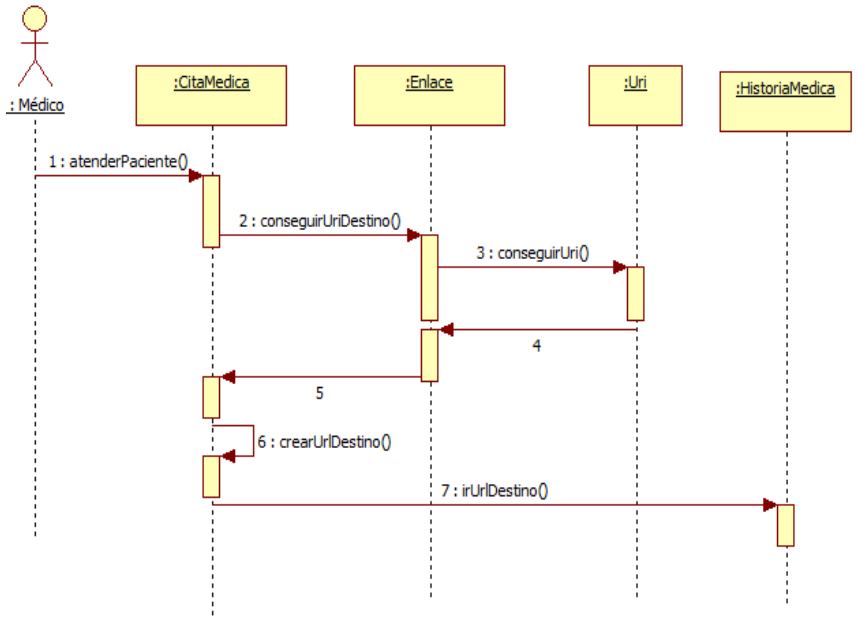


Figura 2.5 Diagrama de Secuencia para historia de usuario HU002.

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.

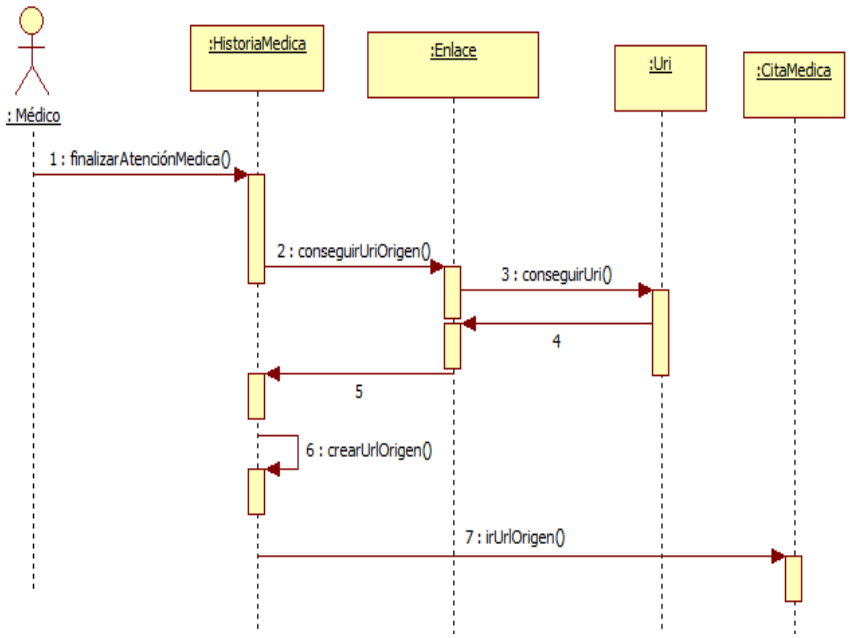


Figura 2.6 Diagrama de Secuencia para historia de usuario HU004.

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.

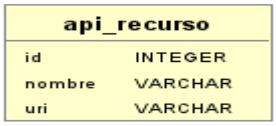


Figura 2.7 Modelo Conceptual de la base de datos del API.

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.

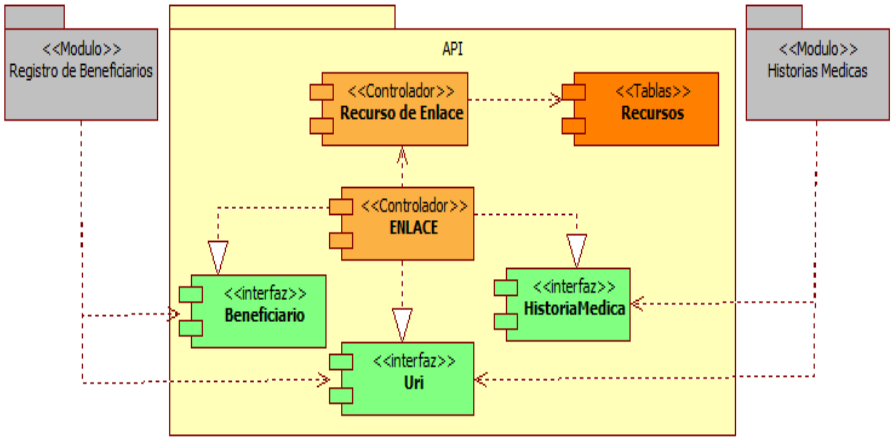


Figura 2.8 Diagrama de Componentes.

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 utilizas por los usuarios finales y los programas, librerías y/o tecnologías que se van a utilizar en el lado del servidor.

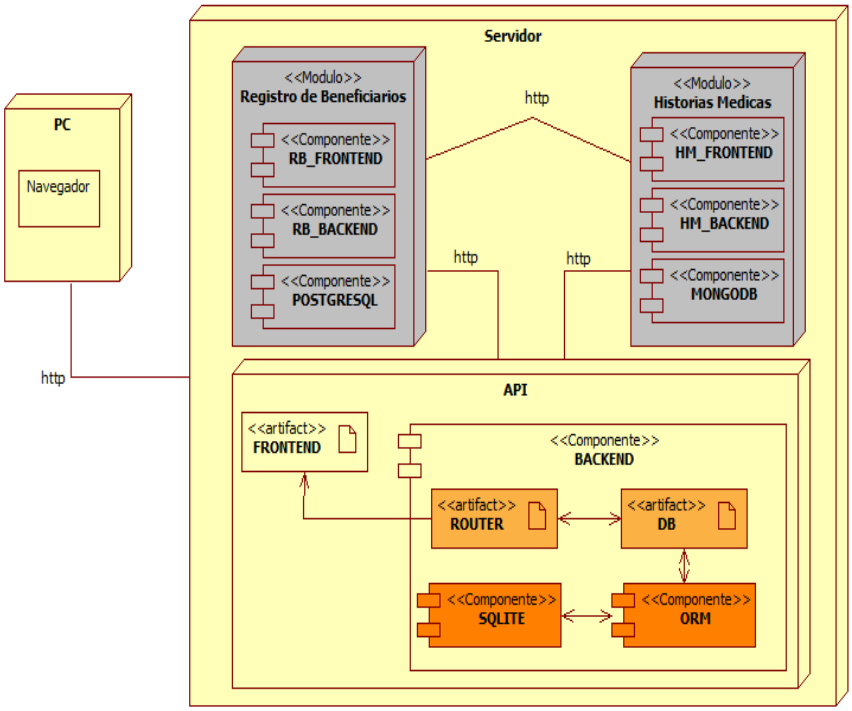


Figura 2.8 Diagrama de Despliegue.

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:

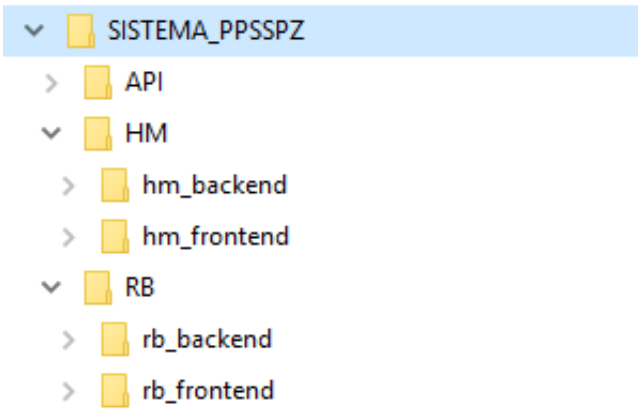


Figura 3.1 Estructura física del sistema PPSSPz.

- 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.

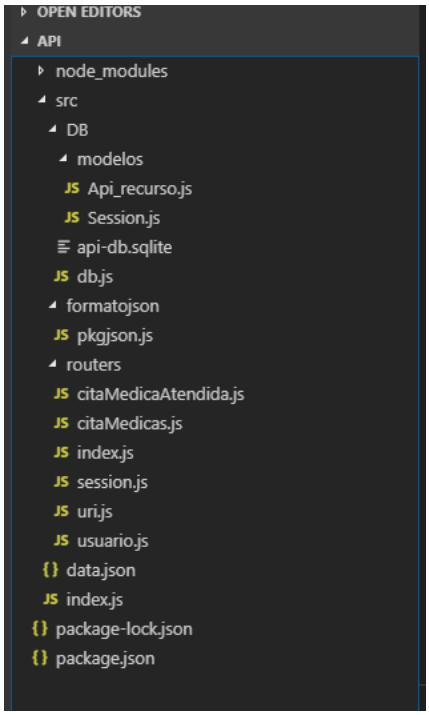


Figura 3.2 Estructura archivos del API.

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.

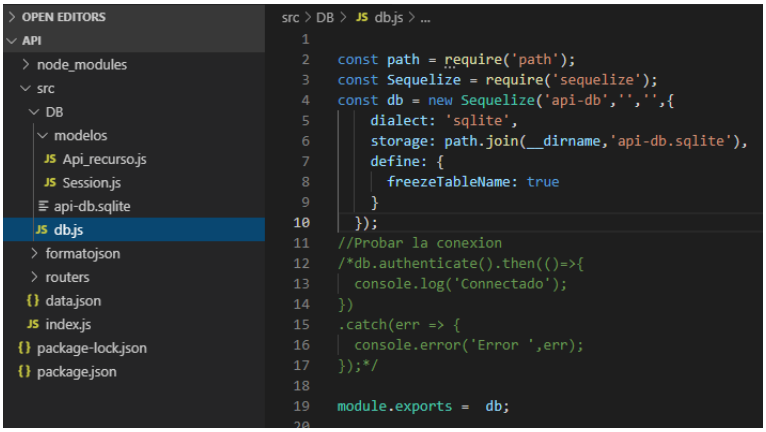


Figura 3.3 Carpeta de DB.

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.

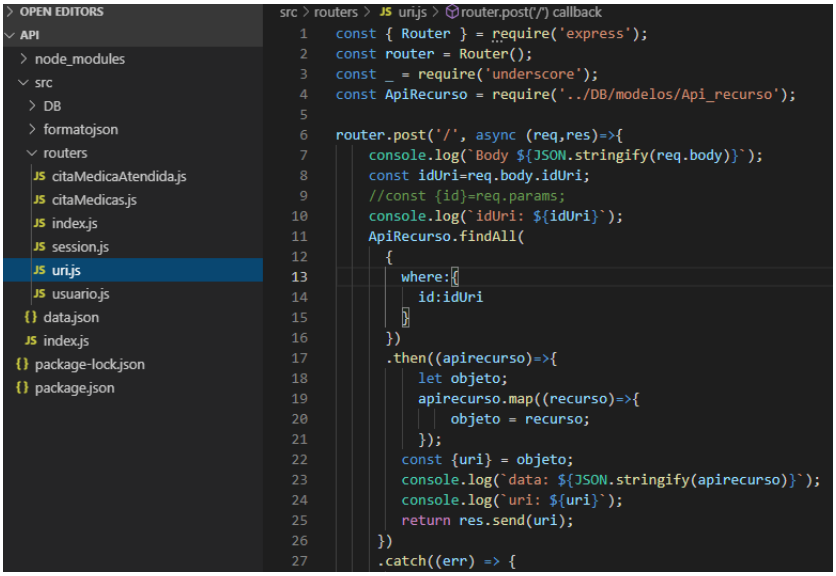


Figura 3.4 Carpeta routers.

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.

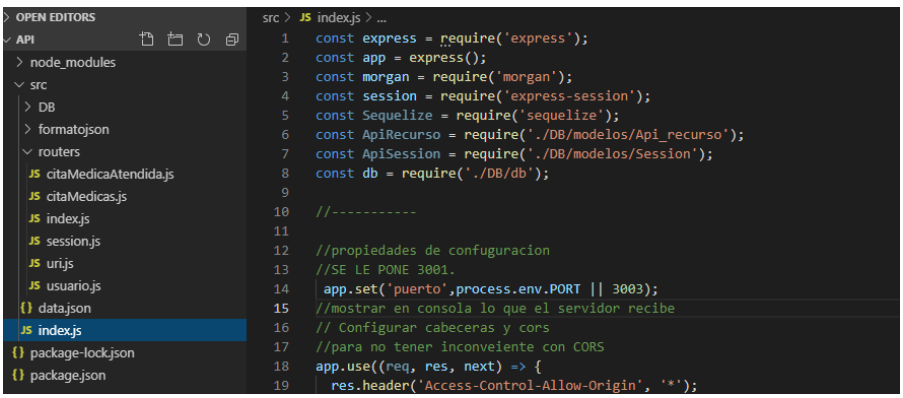


Figura 3.5 Archivo principal del API.

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.

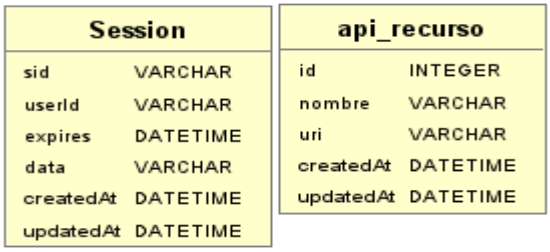


Figura 3.6 Tablas de la Base de Datos del API.

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.

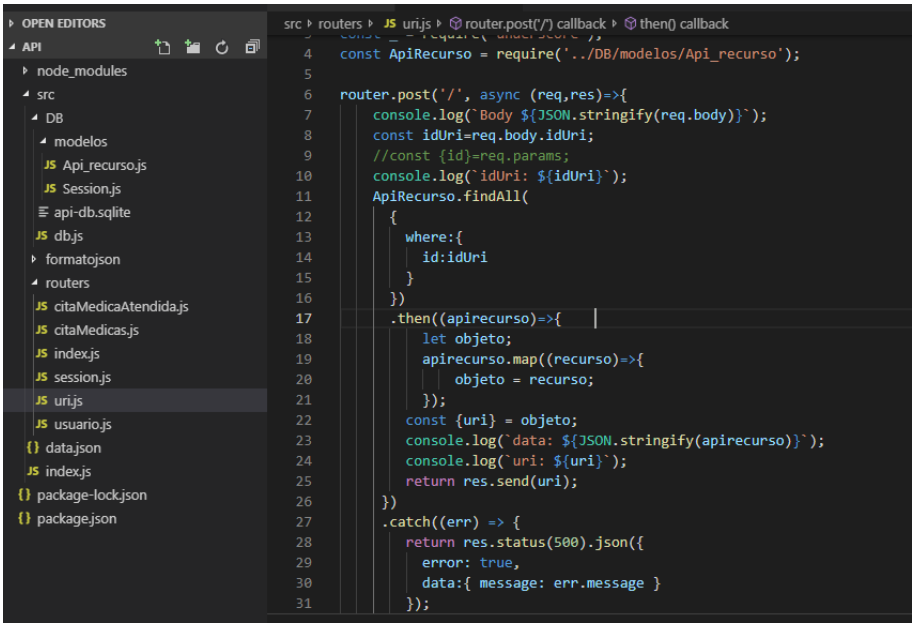


Figura 3.7 Código para acceder a la tabla Api_recurso.

3.2 Actualizaciones adicionales a los módulos historias médicas y registro de beneficiaros

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.

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.

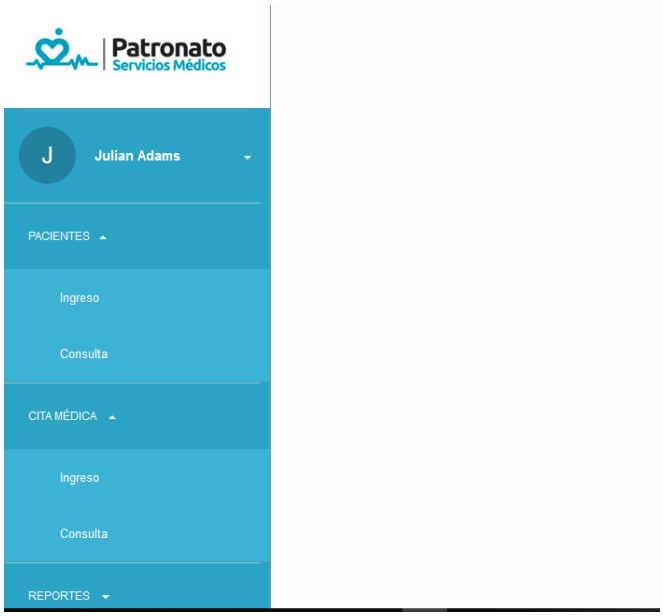


Figura 3.10 Menú principal del sistema PPSSPz.

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.

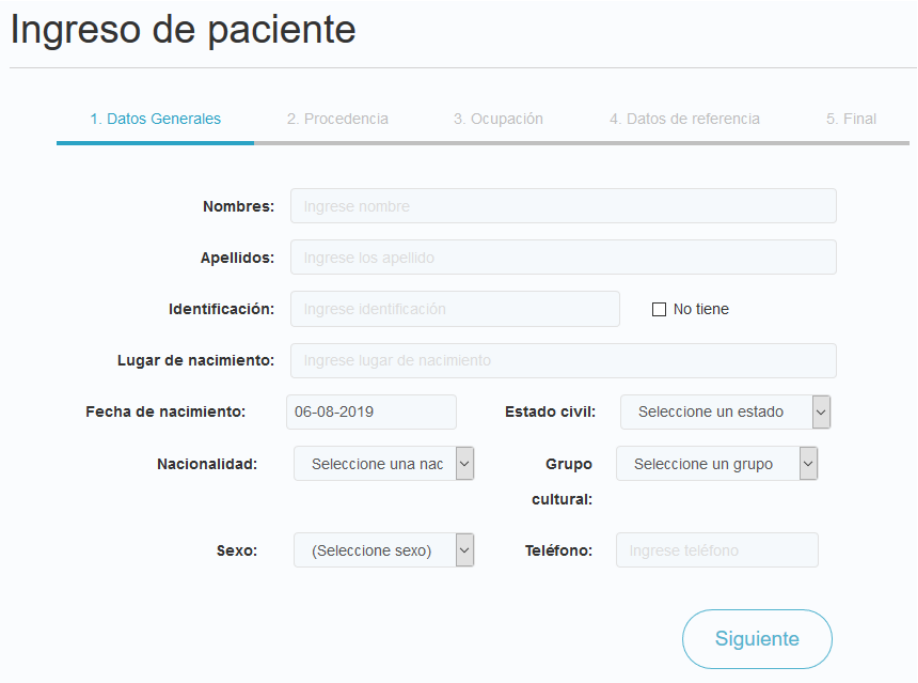


Figura 3.11 Interfaz de Ingreso y modificación de Pacientes.

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.

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.

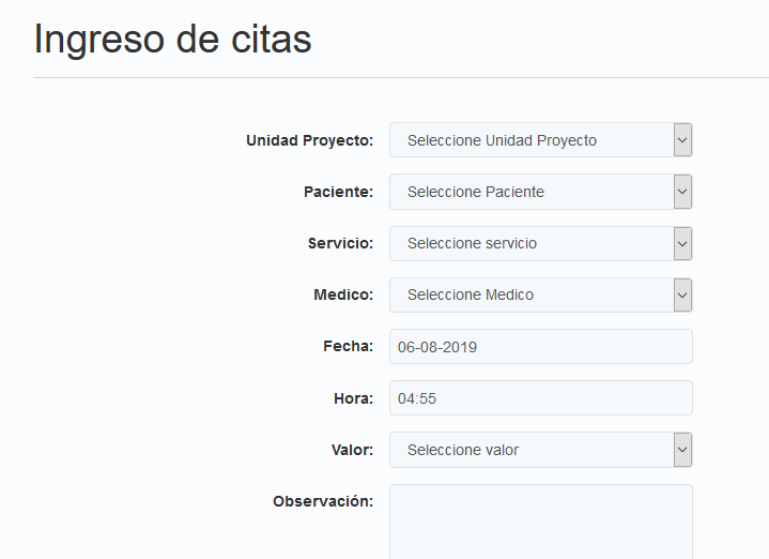


Figura 3.13 Interfaz de Ingreso de Citas Médicas.

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

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.

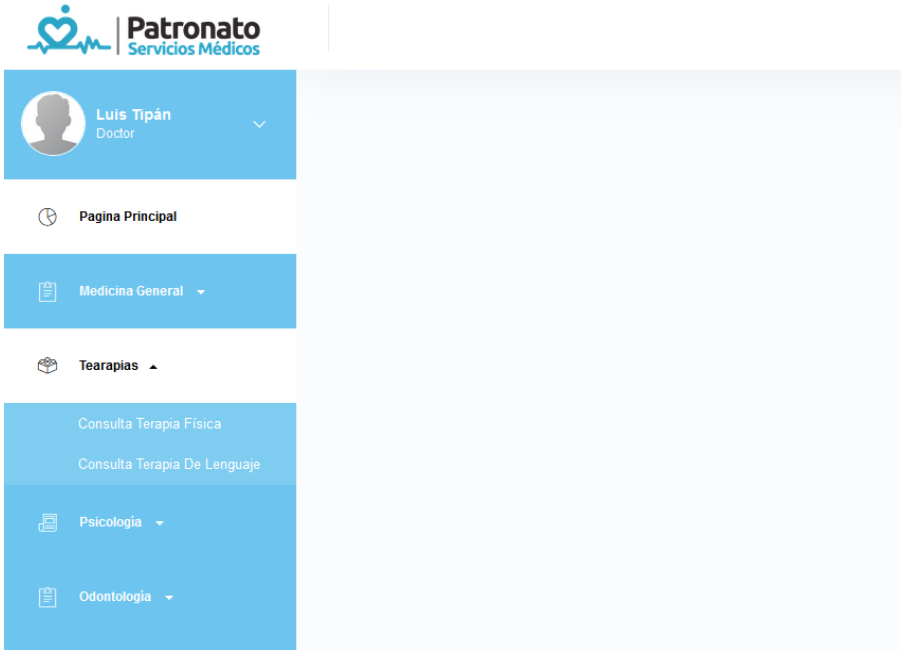


Figura 3.15 Menú principal del Módulo Historias Médicas.

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.

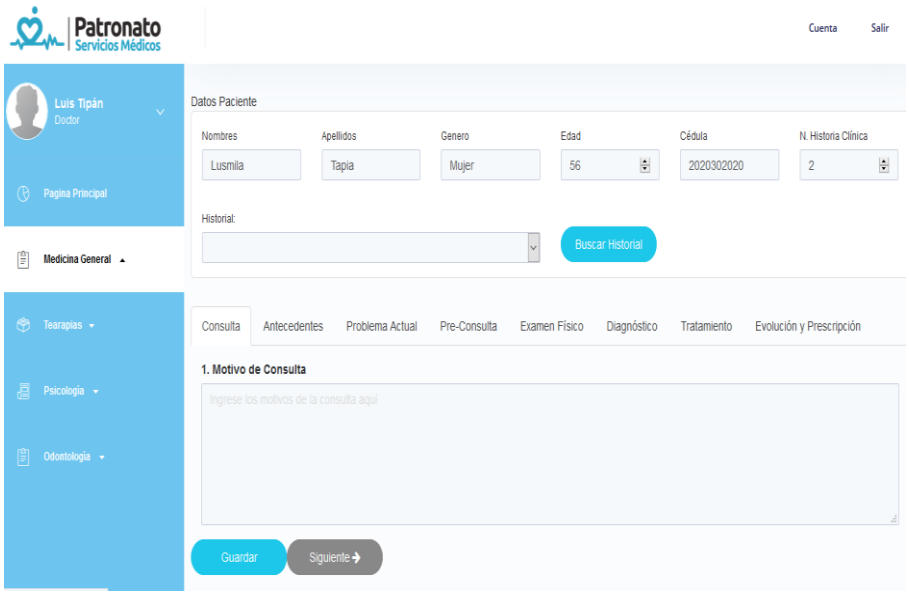


Figura 3.16 Interfaz de Historia Médica Medicina General.

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.

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.

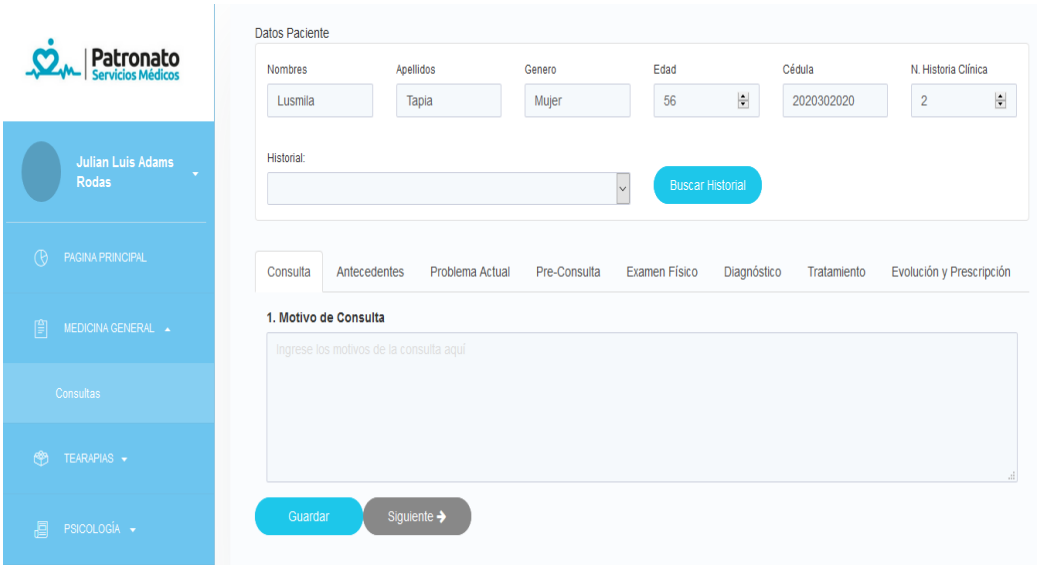


Figura 3.18 Usuario con Rol Médico registrando la historia médica.

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.

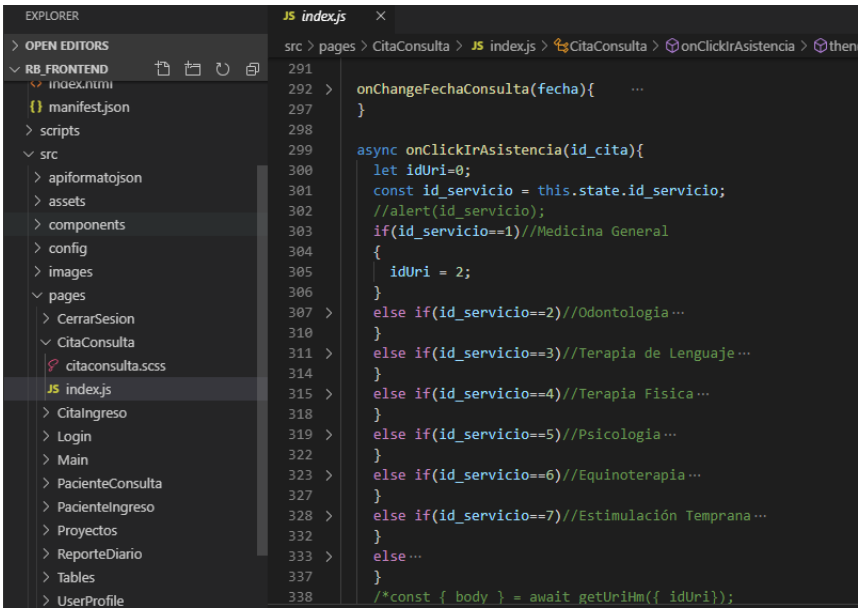


Figura R.1 Procedimiento onClickIrAsistencia.

- 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.

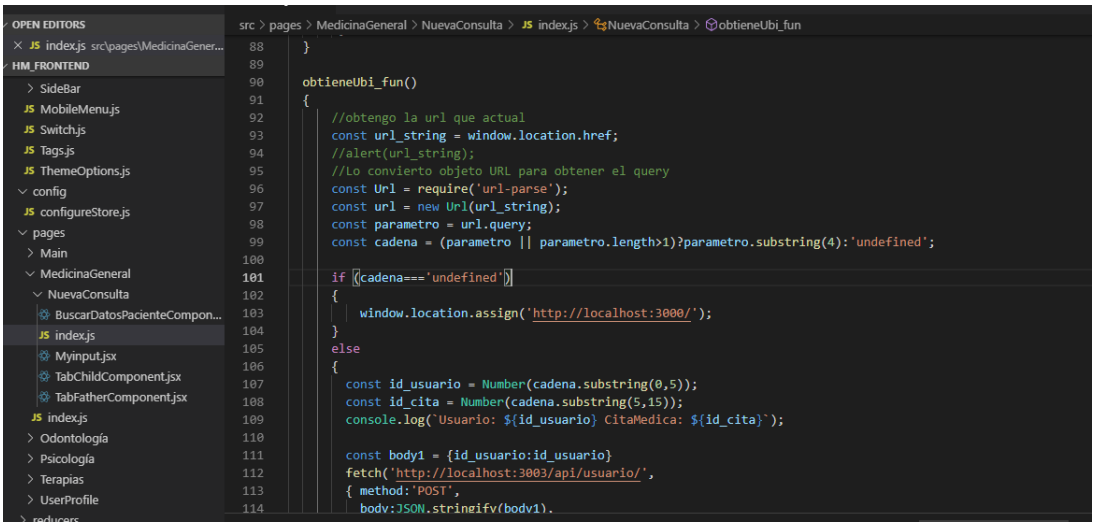


Figura R.2 Procedimiento obtenerUbi_fun.

- 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

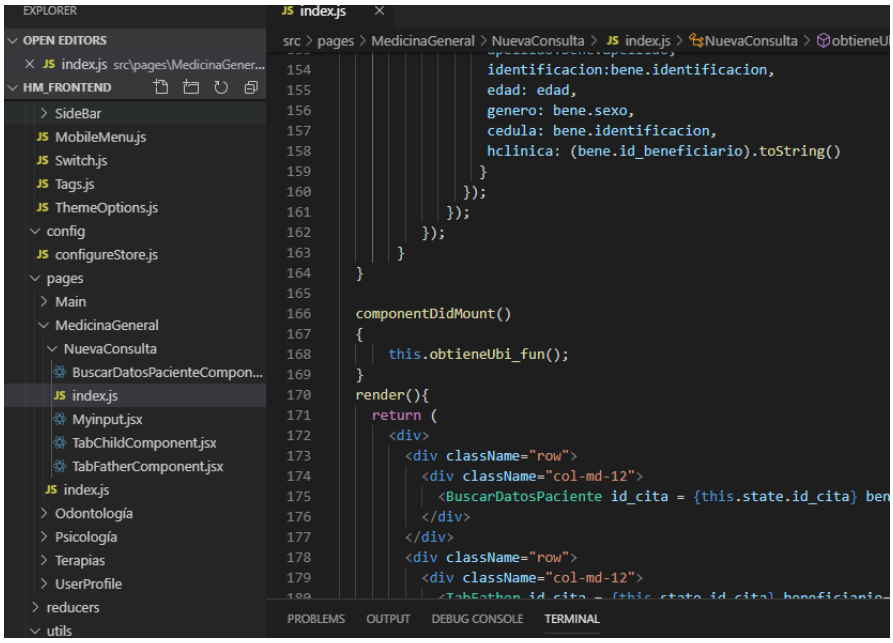


Figura R.3 Método componentDidMount.

También se debe crear el módulo de acceso y permisos, donde se definan las interfaces de usuarios, roles, opciones, usuarios por roles y roles por opciones; ya que están 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 medicas 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. [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 Tecnologico 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/mx-es/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 Arquitect´onicos: 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. [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. [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

Código	HU002
Rol	Submódulo Citas médicas
Funcionalidad	Dirigirme a uri destino
Razón	Para poder registrar el historial médico del paciente
Escenarios o criterios de validación	Si existe el código de cita médica a atender y la sesión de usuario entonces ir al módulo de Historias Médicas
	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

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

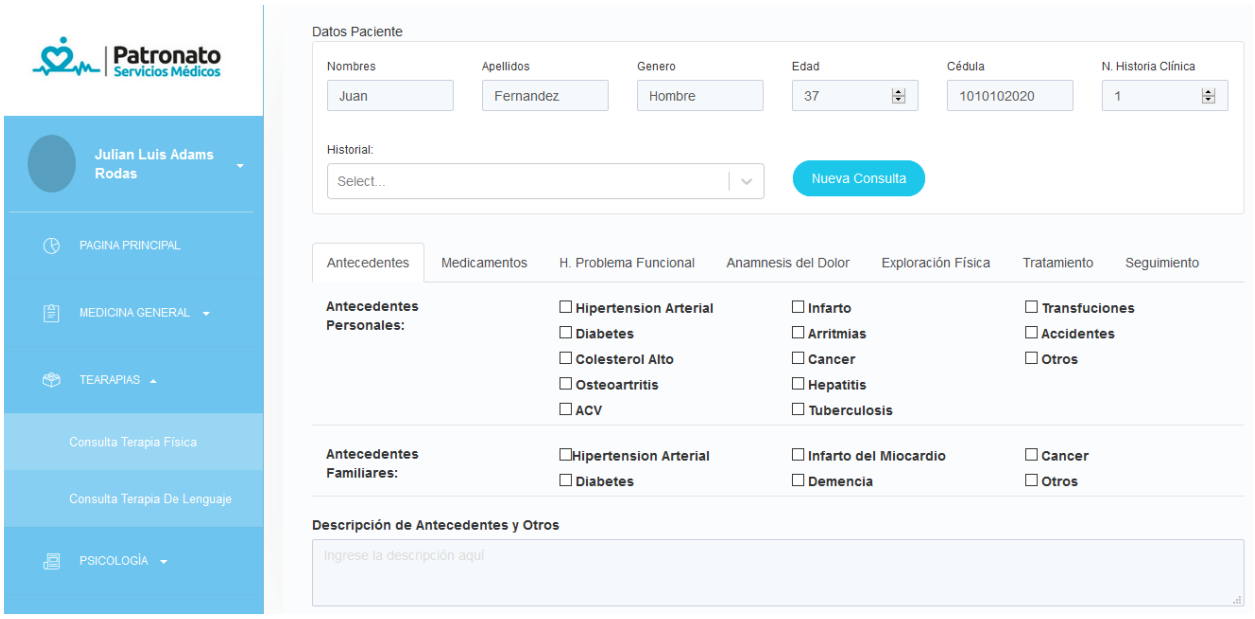
Código	HU004
Rol	Módulo Historias Médicas
Funcionalidad	Dirigirme a uri origen
Razón	Para continuar revisando las citas pendientes
Escenarios o criterios de validación	Si se registró exitosamente el historial médico del paciente entonces dirigirme al submódulo citas médicas
	Si no se registró exitosamente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

Código	HU005
Rol	Módulo Historias Médicas
Funcionalidad	Conseguir Cita Médica
Razón	Para conocer cuál es el código del paciente o beneficiario que se está atendiendo
Escenarios o criterios de validación	Si existe código de paciente o beneficiario entonces continuar con el registro del historial médico
	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

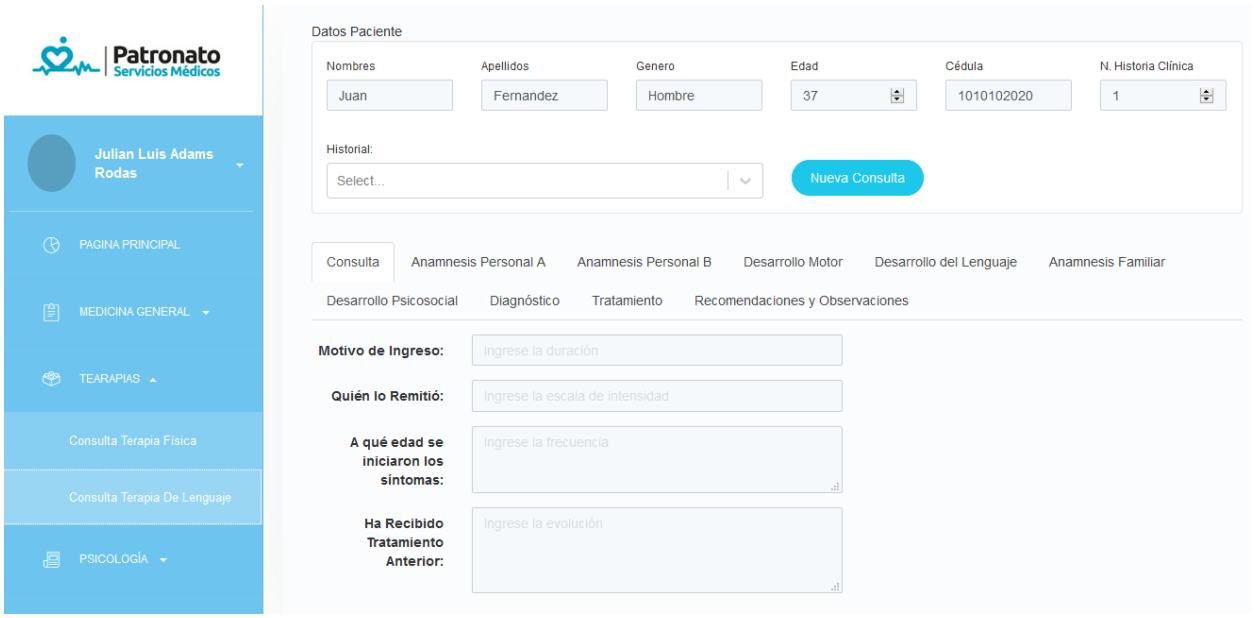
Código	HU006
Rol	Módulo Historias Médicas
Funcionalidad	Conseguir Usuario que inicio sesión
Razón	Para conocer que usuario va a registrar el historial médico del paciente o beneficiario
Escenarios o criterios de validación	Si existe usuario entonces continuar con el registro del historial médico
	Si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

Código	HU007
Rol	Módulo Historias Médicas
Funcionalidad	Establecer cita médica a atendida
Razón	Para finalizar el registro del historial médico
Escenarios o criterios de validación	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
	Si no se estableció el estado de la cita médica atendida entonces mostrar un mensaje del error

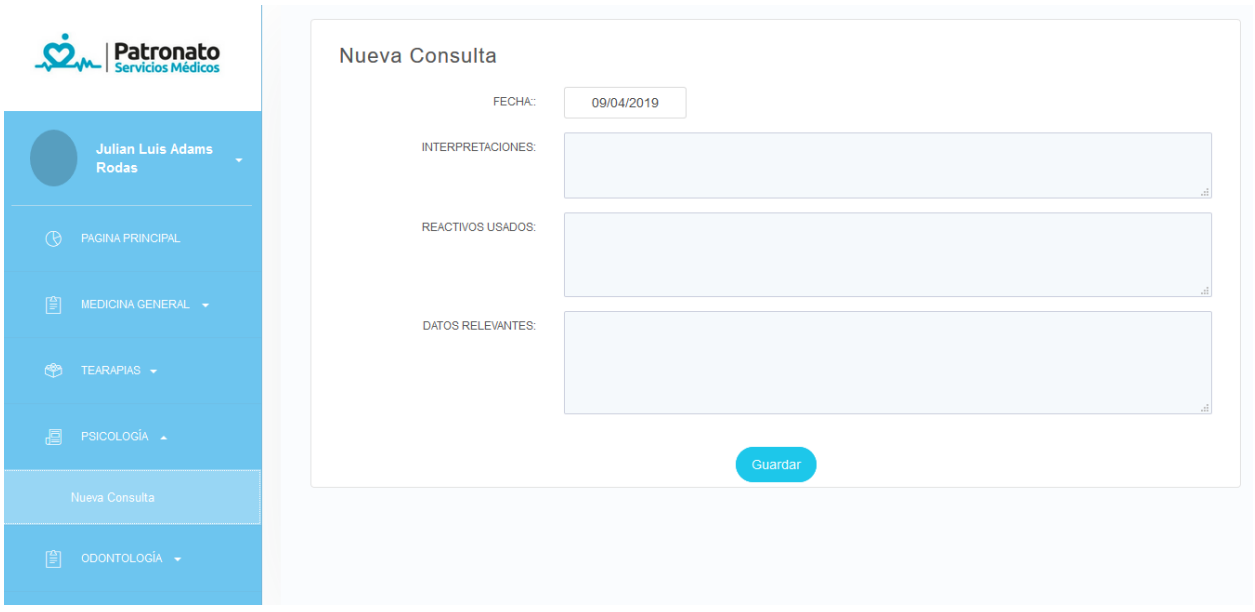
ANEXO A2: Interfaces del módulo Historias Médicas



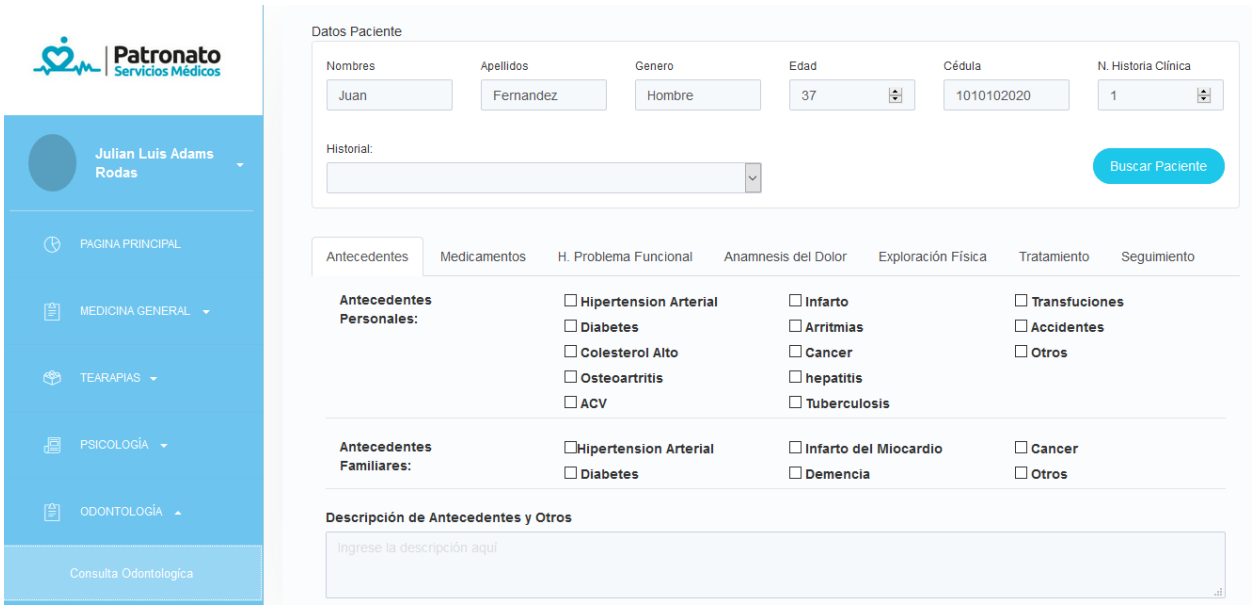
Anexo A2.1 Interfaz Terapia Física



Anexo A2.2 Interfaz Terapia de Lenguaje



Anexo A2.3 Interfaz Psicología



Anexo A2.4 Interfaz Odontología