Trabajo Fin de Máster

Máster en Ingeniería de las Tecnologías de Telecomunicación

Integración de FHIR en la Plataforma Web de prescripción de ejercicio físico “MyTraining”

Autor: Pablo Carmona Rebollo

Tutor: María Teresa Ariza Gómez

Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Máster
Máster en Ingeniería de las Tecnologías de Telecomunicación

Integración de FHIR en la Plataforma Web de prescripción de ejercicio físico “MyTraining”

Autor:
Pablo Carmona Rebollo

Tutor:
María Teresa Ariza Gómez
Profesor titular

Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Máster: Integración de FHIR en la Plataforma Web de prescripción de ejercicio físico “MyTraining”

Autor: Pablo Carmona Rebollo

Tutor: María Teresa Ariza Gómez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis maestros

A mis amigos

Resumen

A lo largo de este documento, se explicará de manera detallada todos aquellos aspectos relevantes que forman parte del proyecto y procesos que han sido llevados a cabo para la realización de este, así como, influencias que han motivado su desarrollo.

El proyecto consiste en la integración de un estándar para el intercambio de datos de atención médica llamado FHIR (**Fast Healthcare Interoperability Resources)** en la Plataforma "MyTraining" basada en la e-Salud, una aplicación Web de prescripción de rutinas de ejercicios que permite a las personas con necesidades especiales obtener entrenamiento personalizado, monitorizado y guiado de manera remota.

La arquitectura de este proyecto se basa, de manera general, en un servidor FHIR, ejecutado en un contenedor, sobre el que se realizan las acciones y peticiones desde la Plataforma "MyTraining". Esta aplicación da la oportunidad a un usuario externo de seguir una rutina de ejercicios dada por un especialista, con su correspondiente monitorización, sin tener que desplazarse a un centro especializado en ello con las molestias que eso conlleva. Es tanto, una forma de favorecer un estilo de vida saludable de una forma práctica y sencilla sin tener que moverse de casa para aquellas personas con características especiales, como un avance hacia la normalización y globalización de los datos médicos de las personas en el sistema sanitario.

Índice

Contenido

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

"Cualquier tecnología suficientemente avanzada
es equivalente a la magia."

Arthur C. Clarke

Actualmente, cada vez se digitalizan más datos médicos, datos que deben estar disponibles y ser comprendidos por entidades médicas de diferentes regiones geográficas y diferentes campos de la medicina. Esto hace necesaria la implementación de una normalización de estos datos, de manera que, sean legíbles y puedan ser explotados por los diferentes sistemas de gestión existentes en los hospitales, etc. FHIR (**Fast Healthcare Interoperability Resources)** es una norma médica creada por la necesidad de disponer de un conjunto de especificaciones para el intercambio riguroso de datos entre aplicaciones médicas.

1.1. Motivación y Objetivos

Existe en la actualidad una fuerte demanda de sistemas que sean interoperables entre sí. Sin embargo, todos conocemos que dicha interoperabilidad no es sencilla con los protocolos y sistemas actuales, muy dependientes de negociaciones, configuraciones o adaptaciones hasta el punto de que dos sistemas que pueden parecer a primera vista casi idénticos son incapaces de comunicarse.

FHIR ha sido diseñado desde el punto de vista de las necesidades de las implantaciones y con la idea de que sea sencillo, dentro de lo que cabe, de aprender y utilizar.

El principal objetivo de este proyecto es la migración de los mecanismos de gestión de datos médicos o historia clínica de los pacientes de la aplicación Web REST llamada MyTraining a FHIR, así como, la implementación de nuevas funcionalidades a dicha plataforma de prescripción de ejercicio físico.
Dicha aplicación pretende favorecer un estilo de vida saludable desde casa mediante rutinas de ejercicios de entrenamiento personalizadas dadas por especialistas a todas aquellas personas con características especiales que necesitan realizar ejercicio físico.
MyTraining permitirá tener dos tipos de usuarios, el usuario 'paciente' y el usuario 'especialista'. Los 'especialistas' serán los encargados de la gestión de rutinas, ejercicios y 'pacientes', mientras que el resto serán atendidos por ellos recibiendo rutinas de entrenamiento personalizadas con videos explicativos y asesoramiento si lo requieren a través de un correo electrónico. Destacar que, también, se podrán registrar observaciones, tanto por parte del especialista para realizar un mejor tratamiento, dejando constancia de valoraciones respecto al paciente, como por parte del paciente para dejar constancia de cómo se ha sentido al realizar cierta rutina.

Mi principal motivación a la hora de realizar este proyecto fue la oportunidad de aprender acerca de los estándares HL7, entre ellos FHIR, que actualmente se encuentra en continuo crecimiento y comienza a asentar las bases de la normalización de la historia clínica de las personas. Además, de la oportunidad de facilitar el tratamiento y cuidado de la salud de pacientes vía telemática.

1.2. Antecedentes

Como antecedente principal a nuestro proyecto, debemos destacar principalmente la siguiente línea de trabajo:

Plataforma web de prescripción de ejercicios usando Spring

Este es un trabajo final de grado que desarrollé en la Escuela Técnica Superior de Ingeniería de Sevilla en el año 2018.

En este proyecto se desarrolló una aplicación Web REST que permite gestionar rutinas de entrenamiento para usuarios que quieren realizar ejercicio físico y llevar un estilo de vida saludable de una manera sencilla y sin moverse de casa. La finalidad de este proyecto es la creación de dichas rutinas y su personalización por los especialistas que las crean, para su posterior uso por los usuarios.

Dicho trabajo final de grado y las tecnologías utilizadas en él han sido utilizados como base para alcanzar el objetivo deseado.

1.3. Contexto

"La práctica de actividad física constituye uno de los pilares fundamentales de un estilo de vida saludable y de una verdadera protección y promoción de la salud. … Aunque existe una gran cantidad de aplicaciones móviles para realizar ejercicio físico …. estas aplicaciones no están adaptadas para colectivos con características especiales que necesitan realizar ejercicio físico, ya sea como prevención o como mejora de determinados problemas en relación con el aparato locomotor (artrosis, dolores musculares…), cardiovascular (hipertensión) o metabólicos (diabetes, colesterol, obesidad...). " – Artículo: ***Arquitectura software para la prescripción de ejercicio físico personalizado / Software Architecture for Customized Physical Exercise Prescription (rediris.es)*** [1][14]

Sobre el artículo anteriormente nombrado se establecen las bases de funcionalidad principal de este proyecto.

Son muchos los artículos existentes sobre un tema candente hoy dia como es la eSalud. La integración de las tecnologías TIC en el ámbito sanitario es abordada cada vez más asiduamente.

A continuación, se mencionan algunos de ellos con una breve descripción:

En el artículo *"Use of an eHealth tool for exercise training and online contact in people with severe chronic obstructive pulmonary disease on long-term oxygen treatment: A feasibility study"* [1][23], se expresa como el uso de eHealth tiene efectos positivos en personas con enfermedad pulmonar obstructiva crónica. Por lo tanto, se evalúa la viabilidad de una herramienta de eSalud utilizada para el entrenamiento de ejercicios y los contactos en línea para personas con enfermedad pulmonar obstructiva crónica grave.

En el artículo *"Interoperable and discrete eHealth Data Exchange between Hospital and Patient"* [1][22], se propone una arquitectura que permita el intercambio seguro de datos entre la aplicación móvil del paciente y la infraestructura del hospital basado en HL7 FHIR.

En el artículo *"Healthcare in the Age of Interoperability: The Promise of Fast Healthcare Interoperability Resources"* [1][2], desde un punto de vista general, se habla acerca de algunas de las características de HL7 FHIR, tales como su interoperabilidad y sus recursos.

A continuación, se explican algunos de los conceptos más importantes relacionados para una mejor comprensión del ámbito del proyecto.

1.3.1. HL7

HL7 (**Health Level Seven**) es una organización internacional nacida en EEUU de desarrollo de estándares globales para facilitar el intercambio electrónico de información sanitaria. En definitiva, su misión es lograr una interoperabilidad clínica real entre los diferentes sistemas de información presentes en las organizaciones de salud.

1.3.2. Estándares HL7

HL7 ha desarrollado un conjunto de estándares que se utilizan en la mayoría de los actuales sistemas.

Los estándares HL7 o protocolos HL7 indican cómo se organiza y comunica la información entre dos partes. Estos estándares definen el idioma, la estructura y los tipos de datos requeridos para una integración fluida entre sistemas de salud.

Miles de hospitales en todo el mundo utilizan a diario estos estándares para intercambiar información entre sus sistemas. Si hablamos de estándares HL7, hablamos de interoperabilidad en salud.

Además de los estándares HL7, existen muchos otros que aparecen a menudo junto a ellos.

Los estándares HL7 más importantes son **HL7 V2, HL7 V3, CDA, HL7 FHIR y CCOW**. Estos cinco son los que se conocen como estándares primarios (***HL7 primary standards***) y son los más usados para la integración de sistemas y la interoperabilidad. Se muestran a continuación en la figura 1.1.

Figura 1.1: Estándares HL7 primarios.

A continuación, se procede a mencionar de manera breve las características de cada uno de ellos.

1.3.2.1. HL7 V2

El estándar HL7 V2 es el más usado pero tiene algunos inconvenientes. HL7 V2 tiene una gran flexibilidad y es adaptable a casi todos los casos, pero esta flexibilidad tiene un coste. Este coste es que las diferencias entre distintas implementaciones requieren profundos análisis y mucha negociación para conseguir la integración y, además, hace que sea complicado testear y llevar a cabo pruebas de conformidad. Esta situación impulsó el desarrollo del estándar HL7 V3.

1.3.2.2. HL7 V3

El enfoque de este nuevo estándar es mucho más formal y pretende solucionar algunos de los inconvenientes de su versión anterior. Una de las principales características del estándar HL7 V3 es que está basado en RIM (Reference Information Model), un amplio modelo de objetos de referencia de los datos clínicos.

RIM es un modelo de toda la información de los servicios sanitarios, que identifica el ciclo de vida de la mensajería dentro de la actividad clínica. Este modelo es la referencia que se usa para el desarrollo de todo el estándar.

Sin embargo, debido a su complejidad y extensión hace que no sea sencillo de implementar. Por ello, la versión 2 está más extendida.

1.3.2.3. HL7 CDA

El estándar HL7 CDA (***Clinical Document Architecture)*** es un estándar de documento clínico basado en el modelo de datos RIM y en la metodología de trabajo de HL7 V3. Esta especificación tiene el objetivo de facilitar el intercambio de información en forma de documentos entre proveedores de salud y pacientes. CDA puede contener cualquier tipo de información clínica, por ejemplo: informes de alta, informes de radiología, etc.

1.3.2.4. CCOW

CCOW (**Clinical Context Object Workgroup**) es un estándar de interoperabilidad que pretende facilitar la integración de aplicaciones a nivel de uso mediante una técnica denominada Context Management.

Esta técnica permite sincronizar y unificar a nivel de interfaz de usuario la información de distintos sistemas que contienen información referida al mismo paciente, procedimiento o usuario.

1.3.2.5. FHIR

Por último, y objeto de este proyecto, HL7 FHIR, un estándar de interoperabilidad que combina lo mejor de HL7 V2, HL7 V3 y CDA y se enfoca en facilitar su implementación. Además, usa los estándares web más frecuentes, como XML, JSON y HTTP.

FHIR es la abreviatura de Fast Healthcare Interoperability Resources (Recursos de Interoperabilidad Sanitaria Rápida). Los resources o recursos son las piezas clave de FHIR. Más adelante, en el apartado 2 de este documento, nos dedicaremos a comentar FHIR detenidamente.

1.3.3. E-Salud

La **eSalud** (*eHealth* en su terminología en inglés) es el término con el que se define al conjunto de Tecnologías de la Información y la Comunicación (TICs) que, a modo de herramientas, se emplean en el entorno sanitario en materia de prevención, diagnóstico, tratamiento, seguimiento, así como en la gestión de la salud, ahorrando costes al sistema sanitario y mejorando la eficacia de este.

Engloba diferentes productos y servicios para la salud, como aplicaciones móviles, la telemedicina, los dispositivos *wearables* (para la monitorización que se integran en ropa y accesorios), el *Big Data* (grandes cantidades de datos), los sistemas de apoyo a la decisión clínica, el Internet de las cosas o los videojuegos de salud, entre otros.

1.3.4. Historia clínica y HCE (Historia Clínica Electrónica)

La historia clínica es el conjunto de documentos que contienen los datos, valoraciones e informaciones de cualquier índole, sobre la situación y la evolución clínica de un paciente a lo largo del proceso asistencial.

La Historia Clínica Electrónica (HCE) o en inglés Electronic Health Record (EHR) supone incorporar las Tecnologías de la Información y la Comunicación (TIC) en el núcleo de la actividad sanitaria. La HCE está diseñada para ser usada como parte de un sistema y da seguimiento a todo el ciclo del paciente y hace que cualquier persona que sea responsable de dicho paciente tenga la capacidad de trabajar coordinadamente.

Algunas de las ventajas de implementar HCE son:

- Cualquier persona responsable del cuidado del paciente puede agregar información en tiempo real, analizar información o colaborar.
- La toma de decisiones médicas o de negocio son más eficientes y rápidas.
- Dar una vista completa de la historia médica del paciente; desde alergias hasta radiología pasando por resultados de laboratorio, facturación y caja.

1.4. Descripción de la Solución

1.4.1. Objetivos específicos

- Instalación y puesta en marcha del servidor HAPI FHIR contenido en un docker y su Base de Datos normalizada.
- Integración de FHIR en la Plataforma MyTraining de manera que, toda acción sobre los datos quede reflejada en la Base de Datos normalizada del servidor HAPI FHIR. Dichas acciones son:
  - Creación de Especialistas y pacientes con sus respectivos datos personales.
  - Creación de rutinas con nombre, descripción, pacientes ligados, autor, responsable y ejercicios asociados.
  - Creación de ejercicios con descripción, creador, periodicidad, vídeo asociado y estado de forma recomendado.
  - Creación de observaciones por especialista con código identificativo según tipo de observación, descripción, paciente asociado, rutina asociada y opcionalmente valor numérico y unidad de medida.
  - Creación de observaciones por paciente con código, rutina referenciada, descripción, creador y valor numérico asociado a como se ha sentido el paciente realizando la rutina.
- Correcta interacción en la comunicación entre el servidor HAPI FHIR y el servicio web REST.
- Interfaz de usuario amigable e intuitiva.
- Diferenciar 3 tipos de usuarios: Administrador, especialistas y pacientes.
- El administrador será el encargado de dar de alta a los especialistas incluyendo sus datos personales y cualificación.
- Los Especialistas serán los encargados de crear, modificar, asociar y eliminar las rutinas, ejercicios y videos relacionados con el entrenamiento individual de cada "paciente".
- Se podrán registrar, modificar y eliminar observaciones, tanto por parte de los especialistas para complementar el seguimiento del tratamiento realizado a cierto paciente, como por los pacientes para expresar sus sensaciones respecto a una rutina.
- Cada rutina tendrá asociada un responsable (podrá no ser el propio creador de esta) cuya cualificación variará según la índole (médico, fisioterapeuta, …)
- Cada "paciente" tendrá visibles solo aquellas rutinas y ejercicios que el Especialista le haya asignado, así como, sus propias observaciones.
- Los Especialistas tendrán dos vistas diferenciadas respecto a rutinas y ejercicios:
  - Las propias

- Las del resto de Especialistas

De la misma manera, existirán dos vistas respecto a las observaciones:

- Las propias
- Las registradas tanto por otros especialistas como por los propios pacientes.

1.4.2. Arquitectura

La arquitectura del sistema objeto de este proyecto es la siguiente:

- **Aplicación Web**: Esta parte de la aplicación es la denominada como *Front-end*, que es la parte del software que interactúa con los usuarios. Consta de:
  - **Modelo Web**: Interfaz de la aplicación con la que interactúan los usuarios. Está diseñada para ser intuitiva y amigable. Las tecnologías utilizadas para su implementación han sido *HTML5* y *CSS3*.
  - **Controlador Web**: Controla las interacciones de los usuarios con la interfaz a través de Javascript/Jquery.
- **Servicio Web REST**: Esta es la parte del servidor de la aplicación denominada como *Back-end*, que procesa la entrada desde el *Front-end*. Dicho servidor está implementado mediante el framework *Spring*, empleando como lenguaje principal *Java*. En el se establece el contexto comunicativo necesario para realizar las peticiones oportunas al servidor externo HAPI FHIR.
- **Base de Datos**: Encargada de la permanencia de la información referente a la aplicación web
- **Servidor HAPI FHIR**: Realiza las transacciones derivadas de las acciones del usuario en el Navegador Web. Dicho servidor se ejecuta en un Docker (Contenedor) y almacena y gestiona los datos de la Base de Datos según estipula el estándar HL7 FHIR

El componente 'Aplicación Web' se comunica con el servidor a través de llamadas a las URIs en las que se encuentran alojados los recursos de este. Dichas llamadas las realiza a través de *JQUERY* mediante *AJAX*.

El componente 'Servicio Web REST' se encarga de capturar las llamadas de la 'Aplicación Web' mediante J*ava* y de realizar las peticiones correspondientes a través de *HTTP al servidor HAPI FHIR*, empleando los métodos CRUD (**create, read, update, and delete**), a la Base de Datos. El servidor HAPI FHIR realiza las acciones sobre los datos mediante transacciones y siguiendo el estándar HL7 FHIR.

El intercambio de información entre el 'Navegador Web' y el 'Servicio Web REST' se realiza en formato JSON.

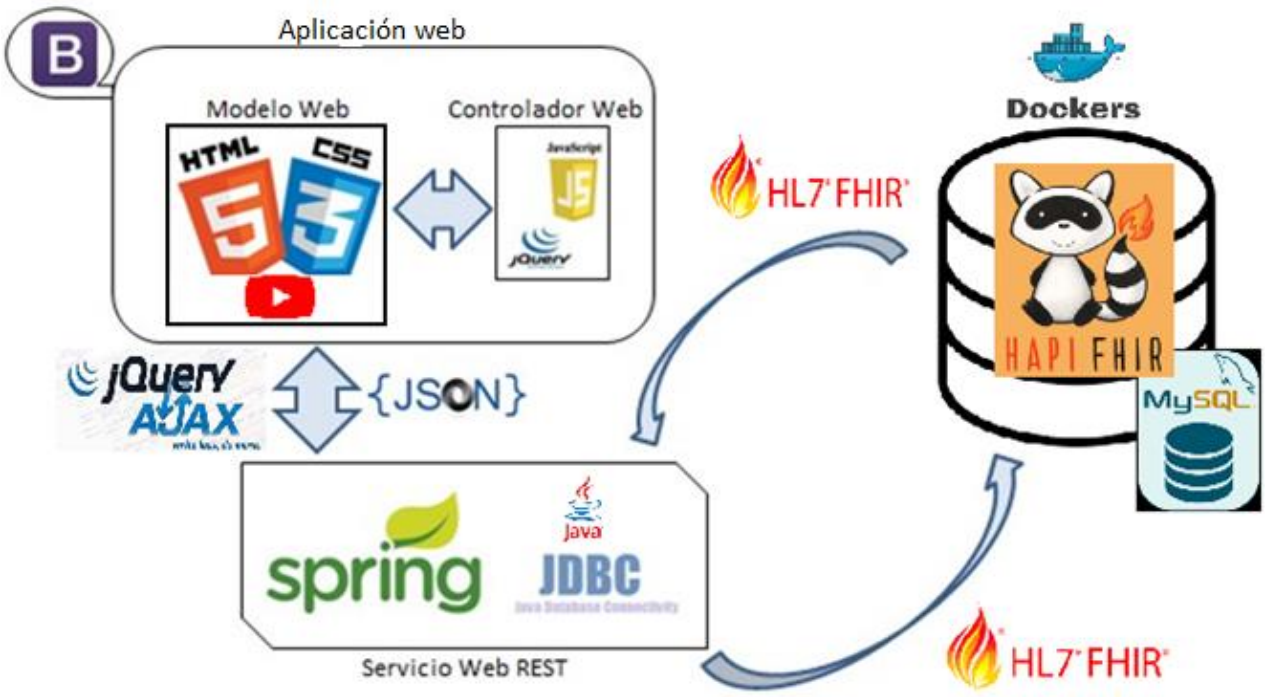


Figura 1.6: Arquitectura de la aplicación Web.

1.5. Estructura de la memoria

En primer lugar, dedicaremos un capítulo a hablar acerca de FHIR. Tanto, su origen, fundamentos y recursos utilizados de este estándar en auge.
Después, se mostrarán las demás tecnologías empleadas a lo largo del proyecto, así como, las herramientas utilizadas.
Acto seguido, se hablará en detalle sobre el diseño del servicio, interfaz de usuario y funcionalidad.
Y, por último, unas conclusiones y líneas futuras.

2. FHIR

2.1 Fundamentos de FHIR

FHIR parte del concepto fundamental de Recursos, donde un recurso es la unidad básica de interoperabilidad, la unidad más pequeña que tiene sentido intercambiar. Los recursos son representaciones de conceptos del mundo sanitario: paciente, médico, problema de salud, observación, …
Los recursos tienen una serie de características comunes:

- Un pequeño conjunto de propiedades principales que la gran mayoría de los sistemas soportan actualmente.
- Un mecanismo de extensión que permite a los implementadores añadir nuevas propiedades de manera sencilla.
- Una *identificación* a través de la cual puede ser registrado, localizado y recuperado.
- Un componente (elementos narrativos) que permite una visión legible de los datos almacenados en el recurso.

Los recursos pueden utilizarse en su forma más simple o agruparse en forma de mensajes, al estilo de las versiones 2 y 3 de HL7 (asemejando los recursos a los segmentos de los mensajes), documentos, de forma similar a los documentos CDA (como una colección de recursos agrupados) o incluso en forma de servicios (empleando uno o más recursos).

FHIR está diseñado específicamente para la web. Los recursos se basan en estructuras XML o JSON que utilizan un protocolo REST basado en http (en contraposición a los servicios basados en SOAP que se pueden encontrar en la mayoría de perfiles IHE (Integrating the Healthcare Enterrpise)).

De forma general, la especificación de FHIR se puede dividir en 3 partes:

- Documentación general: mecanismos de definición de los recursos y material de referencia que incluye los tipos de datos, vocabularios controlados y formatos XML y JSON.
- Implementación: describe la forma de utilizar los recursos definidos utilizando tecnología REST, mensajes, documentos clínicos o empleando arquitecturas basadas en servicios.
- Lista de recursos: relación de todos los recursos definidos en el estándar.

Por último, mencionar que FHIR separa la información en 5 categorías como se muestra a continuación en la figura 2.1 :

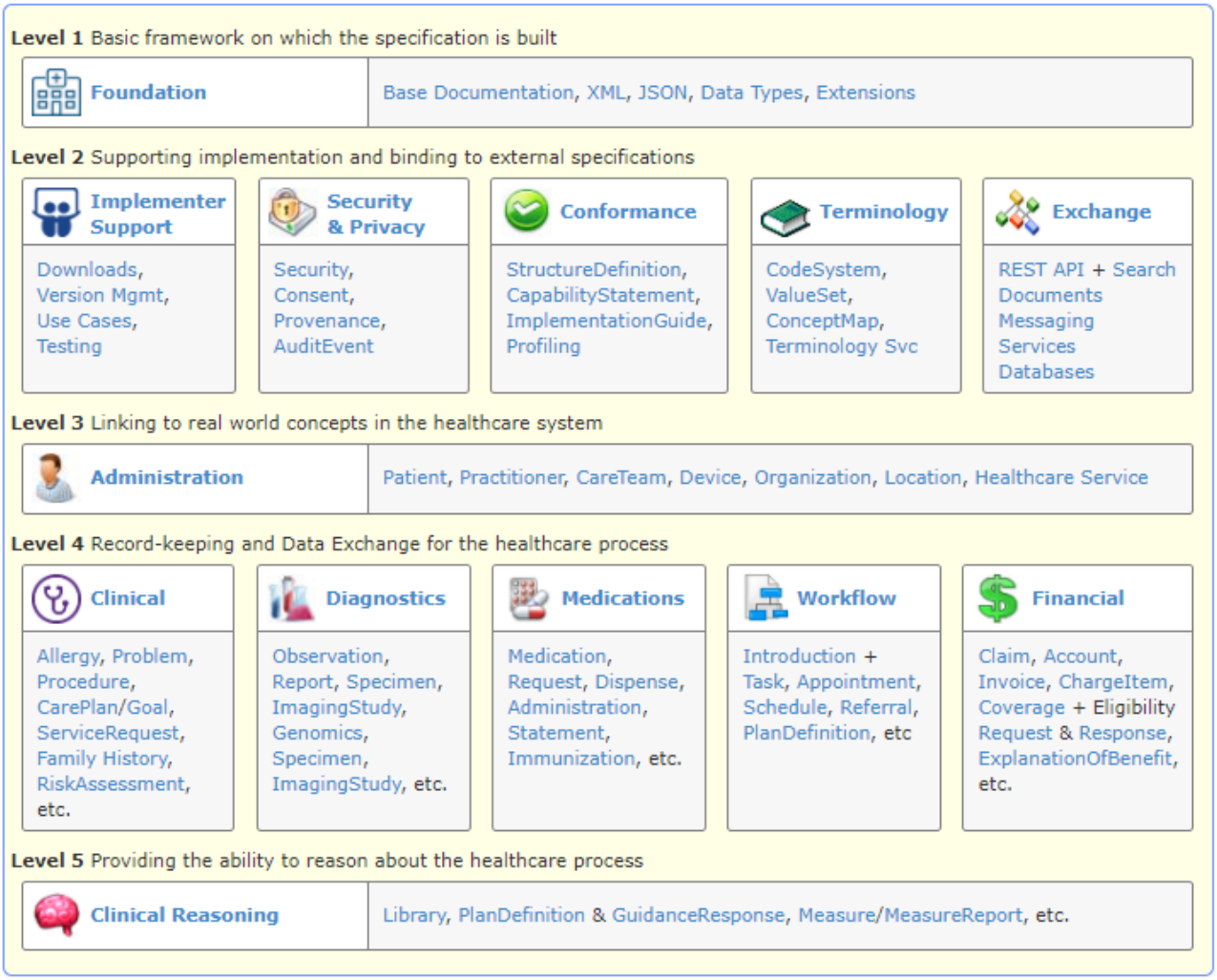


Figura 2.1: Categorías de información FHIR

2.2 Puesta en valor

FHIR en la actualidad, es el candidato más prometedor en el ámbito de estándares de interoperabilidad sanitaria. Sigue los principios de REST y representa a todas las entidades y procedimientos de la asistencia sanitaria como recursos. La integración en el sistema se realiza con interfaces REST a través de métodos HTTP. Estos tienen un diseño más simple que el de SOAP, facilitando mucho su adopción.

La idea básica de FHIR es construir un conjunto de recursos básicos que, singularmente o combinado, abastece de un montón de casos de uso comunes. Los recursos de FHIR definen la estructura y la información que se transmite entre sistemas. Para el resto de información que no está cubierta por FHIR, hay extensiones que pueden ser usadas para construir datos personalizados por país, cultura y dominio específico.

El modelado de los recursos de FHIR se hace por composición, de modo que un recurso puede contener referencias a otros recursos del sistema. Todos los recursos comparten una serie de características:

- Una forma común de definir y representar, a partir de tipos de datos primitivos como números enteros, cadenas, booleanos, etc., tipos de datos complejos como datos personales de un paciente, medidas, citas, periodos, etc.

- Un conjunto común de metadatos referente a la version, última actualización, etc. de los datos.
- Una parte legible llamada Narrativa.

2.3 Otros estándares

Existen actualmente otros estándares que compiten con HL7 FHIR:

Uno de ellos es, el estándar EN/ISO-13606, que se deriva del ENV13606 y describe una arquitectura de información para la interoperabilidad entre los sistemas y componentes necesarios para comunicar datos EHR a través de mensajes electrónicos o como objetos distribuidos. No fue construido para especificar la arquitectura interna o el diseño de la base de datos de los sistemas o componentes EHR. El objetivo es producir tipos de información rigurosos y sostenibles para la comunicación EHR:
- Apoyar la interoperabilidad entre los sistemas y componentes que necesitan interactuar con los servicios EHR;
- Proteger la privacidad y la confidencialidad de los datos de los pacientes

Aunque, sin lugar a dudas, el estándar más destacado es OpenEHR. OpenEHR es un estándar internacional, abierto y de uso libre, enfocado en la estandarización de la arquitectura, funcionalidades, servicios e información para sistemas de información en salud, sobre todo en lo que refiere al registro, acceso, procesamiento, análisis, comunicación e integración de información clínica.

FHIR como openEHR son estándares abierto, FHIR es mantenido por HL7 y openEHR por la fundación openEHR respectivamente. Ambos parecen estar diseñados para resolver problemas ligeramente diferentes.

FHIR está optimizado para el intercambio de datos, a través de la provisión de API REST, simples y fáciles de usar. El componente básico de FHIR son los recursos. Hay alrededor de 100 recursos y, aunque se pueden usar para diversos fines, generalmente se utilizan para intercambiar contenido clínico, como encuentros, planes de cuidado y diagnóstico, etc.

OpenEHR está optimizado para proporcionar una plataforma de datos con un enfoque más fuerte en la persistencia de los datos, con las APIs y el intercambio de datos en segundo plano. A diferencia de los recursos FHIR, openEHR utiliza más de 300 arquetipos. Esta amplitud y profundidad inevitablemente trae un nivel de complejidad elevado, especialmente cuando se compara con FHIR que está optimizado con la opción de extenderse donde sea necesario.

2.4 Recursos

FHIR tiene por objeto definir como recursos las entidades clave que participan en el intercambio de información sanitaria. Cada recurso es una entidad identificable distinta. La especificación de FHIR describe los siguientes atributos de los recursos:

- Los recursos deben tener un límite claro, que coincida con uno o más alcances de transacción lógica.
- Los recursos deben diferir entre sí en cuanto a su significado, no sólo en cuanto a su uso (por ejemplo, las distintas formas de utilizar un informe de laboratorio no deben dar lugar a recursos diferentes).
- Los recursos deben tener una identidad natural.

- Los recursos deben ser muy comunes y utilizados en diferentes transacciones de negocio.
- Los recursos no deben ser lo suficientemente específicos o detallados como para impedir el apoyo a una amplia gama de transacciones de negocio.
- Los recursos deben ser mutuamente excluyentes.
- Los recursos deben utilizar otros recursos, pero deben ser más que simples composiciones de otros recursos; cada recurso debe introducir un contenido novedoso.
- Los recursos deben organizarse en un marco lógico basado en los elementos comunes del recurso y en lo que éste vincula.
- Los recursos deben ser lo suficientemente grandes como para proporcionar contexto; los recursos que contienen sólo unos pocos atributos son probablemente demasiado pequeño para proporcionar un valor significativo.

2.4.1 Recursos empleados

A continuación, se proceden a mostrar los recursos que se han empleado para dar forma a este proyecto.

2.4.1.1 Patient

Los datos del recurso "Patient" cubren la información sobre el paciente: sus atributos se centran en la información demográfica necesaria para apoyar los procedimientos de cuidado de salud.

2.4.1.1.1 Estructura

A continuación, en la figura 2.2, se muestra la estructura de atributos disponibles en el recurso "Patient".

Name	Flags	Card.	Type	Description & Constraints
Patient	N		DomainResource	Information about an individual or animal receiving health care services Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier	Σ	0..*	Identifier	An identifier for this patient
active	?! Σ	0..1	boolean	Whether this patient's record is in active use
name	Σ	0..*	HumanName	A name associated with the patient
telecom	Σ	0..*	ContactPoint	A contact detail for the individual
gender	Σ	0..1	code	male | female | other | unknown AdministrativeGender (Required)
birthDate	Σ	0..1	date	The date of birth for the individual
deceased[x]	?! Σ	0..1		Indicates if the individual is deceased or not
deceasedBoolean			boolean	
deceasedDateTime			dateTime	
address	Σ	0..*	Address	An address for the individual
maritalStatus		0..1	CodeableConcept	Marital (civil) status of a patient MaritalStatus (Extensible)
multipleBirth[x]		0..1		Whether patient is part of a multiple birth
multipleBirthBoolean			boolean	
multipleBirthInteger			integer	
photo		0..*	Attachment	Image of the patient
contact	I	0..*	BackboneElement	A contact party (e.g. guardian, partner, friend) for the patient *+ Rule: SHALL at least contain a contact's details or a reference to an organization*
relationship		0..*	CodeableConcept	The kind of relationship Patient Contact Relationship (Extensible)
name		0..1	HumanName	A name associated with the contact person
telecom		0..*	ContactPoint	A contact detail for the person
address		0..1	Address	Address for the contact person
gender		0..1	code	male | female | other | unknown AdministrativeGender (Required)
organization	I	0..1	Reference(Organization)	Organization that is associated with the contact
period		0..1	Period	The period during which this contact person or organization is valid to be contacted relating to this patient
communication		0..*	BackboneElement	A language which may be used to communicate with the patient about his or her health
language		1..1	CodeableConcept	The language which can be used to communicate with the patient about his or her health Common Languages (Preferred but limited to AllLanguages)
preferred		0..1	boolean	Language preference indicator
generalPractitioner		0..*	Reference(Organization | Practitioner | PractitionerRole)	Patient's nominated primary care provider
managingOrganization	Σ	0..1	Reference(Organization)	Organization that is the custodian of the patient record
link	?! Σ	0..*	BackboneElement	Link to another patient resource that concerns the same actual person
other	Σ	1..1	Reference(Patient | RelatedPerson)	The other patient or related person resource that the link refers to
type	Σ	1..1	code	replaced-by | replaces | refer | seealso

Figura 2.2: Estructura recurso Patient

2.4.1.1.2 Uso

Este recurso se genera en la Base de Datos FHIR cada vez que un especialista da de alta a un nuevo paciente. Contiene la siguiente información del paciente:

- Identificador
- Nombre
- Teléfono móvil
- Email
- Correo electrónico
- Fecha de nacimiento
- Si está dado de baja o no (activo)

2.4.1.1.3 UML

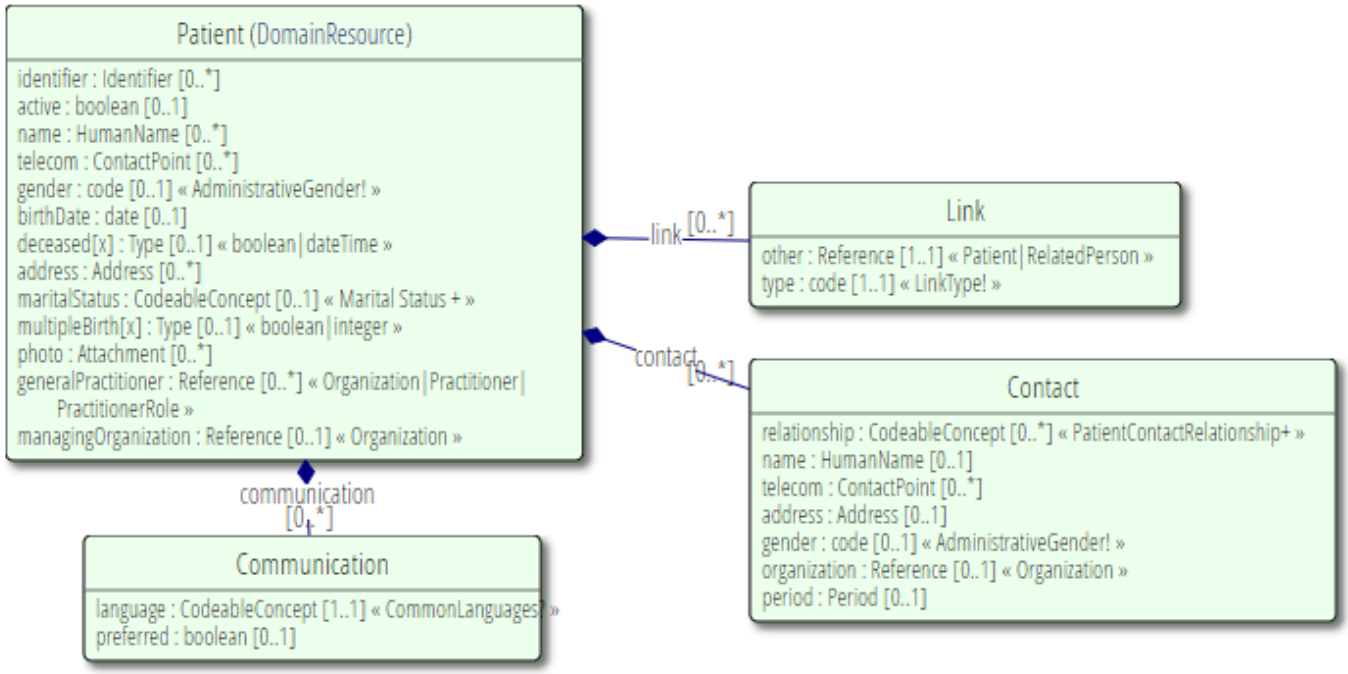


Figura 2.3: UML recurso Patient

2.4.1.1.4 JSON general

```
{
  "resourceType" : "Patient",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier" : [{ Identifier }], // An identifier for this patient
  "active" : <boolean>, // Whether this patient's record is in active use
  "name" : [{ HumanName }], // A name associated with the patient
  "telecom" : [{ ContactPoint }], // A contact detail for the individual
  "gender" : "<code>", // male | female | other | unknown
  "birthDate" : "<date>", // The date of birth for the individual
  // deceased[x]: Indicates if the individual is deceased or not. One of these 2:
  "deceasedBoolean" : <boolean>,
  "deceasedDateTime" : "<dateTime>",
  "address" : [{ Address }], // An address for the individual
  "maritalStatus" : { CodeableConcept }, // Marital (civil) status of a patient
  // multipleBirth[x]: Whether patient is part of a multiple birth. One of these 2:
  "multipleBirthBoolean" : <boolean>,
  "multipleBirthInteger" : <integer>,
  "photo" : [{ Attachment }], // Image of the patient
  "contact" : [{ // A contact party (e.g. guardian, partner, friend) for the patient
    "relationship" : [{ CodeableConcept }], // The kind of relationship
    "name" : { HumanName }, // A name associated with the contact person
    "telecom" : [{ ContactPoint }], // A contact detail for the person
    "address" : { Address }, // Address for the contact person
    "gender" : "<code>", // male | female | other | unknown
    "organization" : { Reference(Organization) }, // C? Organization that is associated with the contact
    "period" : { Period } // The period during which this contact person or organization is valid to be c
ontacted relating to this patient
  }],
  "communication" : [{ // A language which may be used to communicate with the patient about his or her h
ealth
    "language" : { CodeableConcept }, // R!  The language which can be used to communicate with the patie
nt about his or her health
    "preferred" : <boolean> // Language preference indicator
  }],
  "generalPractitioner" : [{ Reference(Organization|Practitioner|
   PractitionerRole) }], // Patient's nominated primary care provider
  "managingOrganization" : { Reference(Organization) }, // Organization that is the custodian of the pati
ent record
  "link" : [{ // Link to another patient resource that concerns the same actual person
    "other" : { Reference(Patient|RelatedPerson) }, // R!  The other patient or related person resource t
hat the link refers to
    "type" : "<code>" // R!  replaced-by | replaces | refer | seealso
  }]
```

Figura 2.4: JSON general recurso Patient

2.4.1.1.5 Ejemplo práctico


```
{
  "resourceType": "Patient",
  "id": "1252",
  "meta": {
    "versionId": "2",
    "lastUpdated": "2020-12-22T19:31:02.000+00:00",
    "source": "#wKGJ7KS0tAwmEddu"
  },
  "text": {
    "status": "generated",
    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\"><div class=\"hapiHeaderText\">Pepi Perez
</div><table class=\"hapiPropertyTable\"><tbody><tr><td>Identifier</td><td>pepi@gmail.com</td>
</tr><tr><td>Date of birth</td><td><span>30 November 2020</span></td></tr></tbody></table></div>"
  },
  "identifier": [
    {
      "value": "pepi@gmail.com"
    }
  ],
  "active": true,
  "name": [
    {
      "given": [
        "Pepi Perez"
      ]
    }
  ],
  "telecom": [
    {
      "system": "phone",
      "value": "123412444",
      "use": "mobile"
    },
    {
      "system": "email",
      "value": "pepi@gmail.com"
    }
  ],
  "birthDate": "2020-11-30"
}
```


Figura 2.5: Ejemplo recurso Patient

2.4.1.2 Practitioner

El recurso “Practitioner” cubre a todas las personas que participan en el proceso de atención médica.

2.4.1.2.1 Estructura

A continuación, en la figura 2.2, se muestra la estructura de atributos disponibles en el recurso “Patient”.

Structure

Name	Flags	Card.	Type	Description & Constraints
Practitioner	TU		DomainResource	A person with a formal responsibility in the provisioning of healthcare or related services Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier	Σ	0..*	Identifier	An identifier for the person as this agent
active	Σ	0..1	boolean	Whether this practitioner's record is in active use
name	Σ	0..*	HumanName	The name(s) associated with the practitioner
telecom	Σ	0..*	ContactPoint	A contact detail for the practitioner (that apply to all roles)
address	Σ	0..*	Address	Address(es) of the practitioner that are not role specific (typically home address)
gender	Σ	0..1	code	male | female | other | unknown AdministrativeGender (Required)
birthDate	Σ	0..1	date	The date on which the practitioner was born
photo		0..*	Attachment	Image of the person
qualification		0..*	BackboneElement	Certification, licenses, or training pertaining to the provision of care
identifier		0..*	Identifier	An identifier for this qualification for the practitioner
code		1..1	CodeableConcept	Coded representation of the qualification v2 table 0360, Version 2.7 (Example)
period		0..1	Period	Period during which the qualification is valid
issuer		0..1	Reference(Organization)	Organization that regulates and issues the qualification
communication		0..*	CodeableConcept	A language the practitioner can use in patient communication Common Languages (Preferred but limited to AllLanguages)

Documentation for this format

Figura 2.6: Estructura recurso Practitioner

2.4.1.2.2 Uso

Este recurso se genera en la Base de Datos FHIR cada vez que un administrador da de alta a un nuevo especialista. Contiene la siguiente información del especialista:

- Identificador
- Nombre
- Teléfono móvil
- Email
- Correo electrónico
- Fecha de nacimiento
- Cualificación

2.4.1.2.3 UML

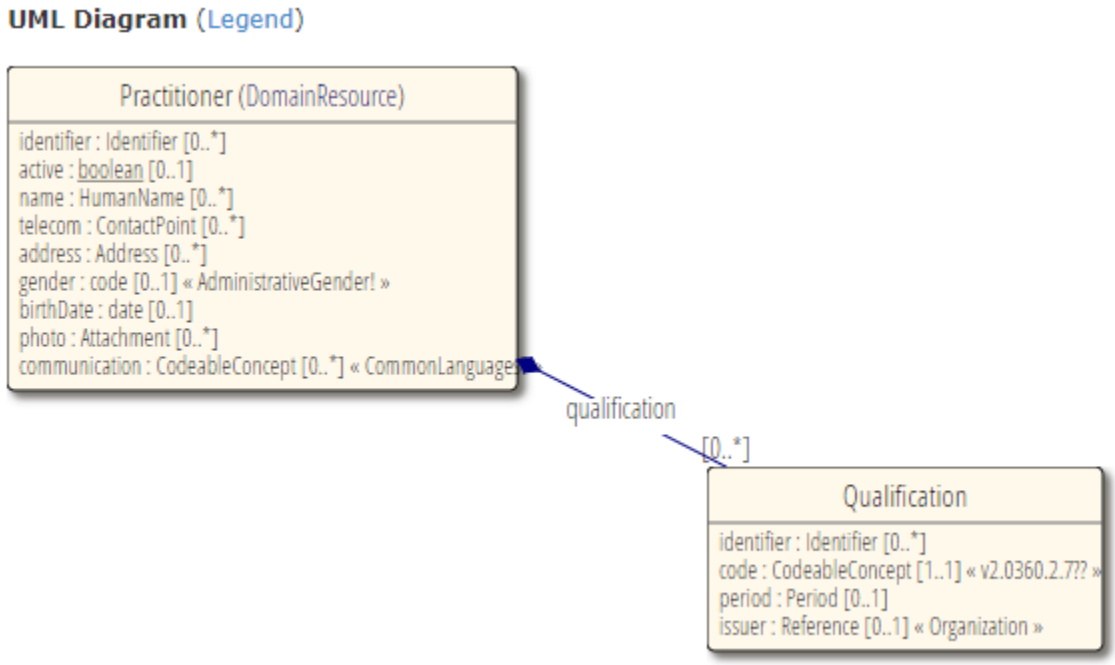


Figura 2.7: UML recurso Practitioner

2.4.1.2.4 JSON general

```
{
  "resourceType" : "Practitioner",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier" : [{ Identifier }], // An identifier for the person as this agent
  "active" : <boolean>, // Whether this practitioner's record is in active use
  "name" : [{ HumanName }], // The name(s) associated with the practitioner
  "telecom" : [{ ContactPoint }], // A contact detail for the practitioner (that apply to all roles)
  "address" : [{ Address }], // Address(es) of the practitioner that are not role specific (typically hom
e address)
  "gender" : "<code>", // male | female | other | unknown
  "birthDate" : "<date>", // The date  on which the practitioner was born
  "photo" : [{ Attachment }], // Image of the person
  "qualification" : [{ // Certification, licenses, or training pertaining to the provision of care
    "identifier" : [{ Identifier }], // An identifier for this qualification for the practitioner
    "code" : { CodeableConcept }, // R!  Coded representation of the qualification
    "period" : { Period }, // Period during which the qualification is valid
    "issuer" : { Reference(Organization) } // Organization that regulates and issues the qualification
  }],
  "communication" : [{ CodeableConcept }] // A language the practitioner can use in patient communication
}
```

Figura 2.8: JSON general recurso Patient

2.4.1.2.5 Ejemplo práctico


```
{
  "resourceType": "Practitioner",
  "id": "902",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2020-12-11T18:02:24.000+00:00",
    "source": "#zHwhuEYaXpn9W3Wf"
  },
  "text": {
    "status": "generated",
    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">This data was generated for test purposes.</div>"
  },
  "identifier": [
    {
      "system": "http://acme.org/mrn",
      "value": "maximo@gmail.com"
    }
  ],
  "name": [
    {
      "given": [
        "Máximo López"
      ]
    }
  ],
  "telecom": [
    {
      "system": "phone",
      "value": "098765432",
      "use": "mobile"
    },
    {
      "system": "email",
      "value": "maximo@gmail.com"
    }
  ],
  "birthDate": "1960-09-09",
  "qualification": [
    {
      "identifier": [
        {
          "value": "Médico"
        }
      ],
      "code": {
        "text": "Especialista"
      }
    }
  ]
}
```


Figura 2.9: Ejemplo recurso Practitioner

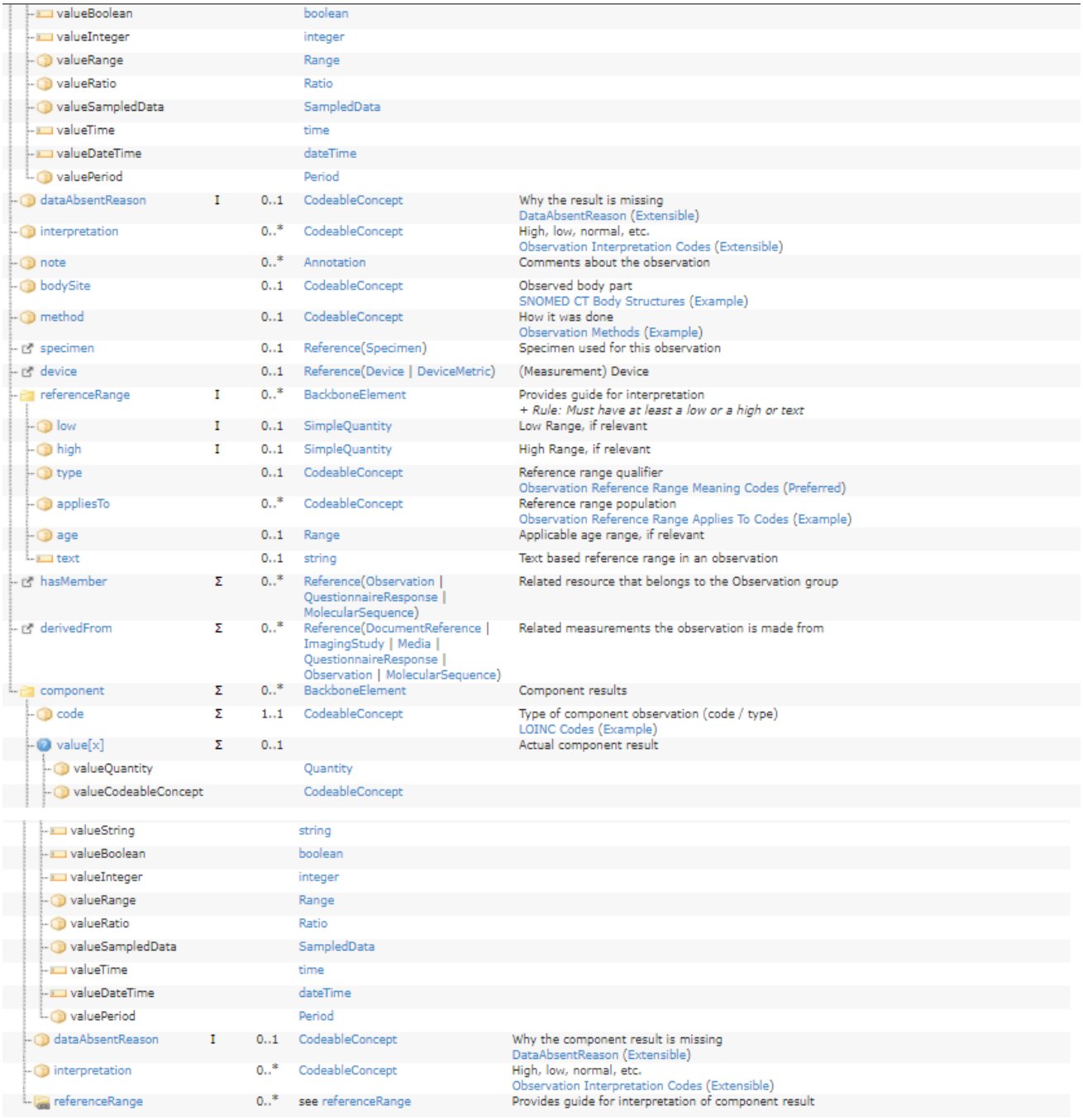
2.4.1.3 Observation

Este recurso recoge mediciones y afirmaciones simples sobre un paciente.

2.4.1.3.1 Estructura

A continuación, en la figura 2.10, se muestra la estructura de atributos del recurso “Observation”.

Name	Flags	Card.	Type	Description & Constraints
Observation	I N		DomainResource	Measurements and simple assertions + *Rule: dataAbsentReason SHALL only be present if Observation.value[x] is not present* + *Rule: If Observation.code is the same as an Observation.component.code then the value element associated with the code SHALL NOT be present* Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier	Σ	0..*	Identifier	Business Identifier for observation
basedOn	Σ	0..*	Reference(CarePlan | DeviceRequest | ImmunizationRecommendation | MedicationRequest | NutritionOrder | ServiceRequest)	Fulfills plan, proposal or order
partOf	Σ	0..*	Reference(MedicationAdministration | MedicationDispense | MedicationStatement | Procedure | Immunization | ImagingStudy)	Part of referenced event
status	?! Σ	1..1	code	registered | preliminary | final | amended + ObservationStatus (Required)
category		0..*	CodeableConcept	Classification of type of observation Observation Category Codes (Preferred)
code	Σ	1..1	CodeableConcept	Type of observation (code / type) LOINC Codes (Example)
subject	Σ	0..1	Reference(Patient | Group | Device | Location)	Who and/or what the observation is about
focus	Σ TU	0..*	Reference(Any)	What the observation is about, when it is not about the subject of record
encounter	Σ	0..1	Reference(Encounter)	Healthcare event during which this observation is made
effective[x]	Σ	0..1		Clinically relevant time/time-period for observation
effectiveDateTime			dateTime	
effectivePeriod			Period	
effectiveTiming			Timing	
effectiveInstant			instant	
issued	Σ	0..1	instant	Date/Time this version was made available
performer	Σ	0..*	Reference(Practitioner | PractitionerRole | Organization | CareTeam | Patient | RelatedPerson)	Who is responsible for the observation
value[x]	Σ I	0..1		Actual result
valueQuantity			Quantity	
valueCodeableConcept			CodeableConcept	
valueString			string	



valueBoolean			boolean	
valueInteger			integer	
valueRange			Range	
valueRatio			Ratio	
valueSampledData			SampledData	
valueTime			time	
valueDateTime			dateTime	
valuePeriod			Period	
dataAbsentReason	I	0..1	CodeableConcept	Why the result is missing DataAbsentReason (Extensible)
interpretation		0..*	CodeableConcept	High, low, normal, etc. Observation Interpretation Codes (Extensible)
note		0..*	Annotation	Comments about the observation
bodySite		0..1	CodeableConcept	Observed body part SNOMED CT Body Structures (Example)
method		0..1	CodeableConcept	How it was done Observation Methods (Example)
specimen		0..1	Reference(Specimen)	Specimen used for this observation
device		0..1	Reference(Device | DeviceMetric)	(Measurement) Device
referenceRange	I	0..*	BackboneElement	Provides guide for interpretation *+ Rule: Must have at least a low or a high or text*
low	I	0..1	SimpleQuantity	Low Range, if relevant
high	I	0..1	SimpleQuantity	High Range, if relevant
type		0..1	CodeableConcept	Reference range qualifier Observation Reference Range Meaning Codes (Preferred)
appliesTo		0..*	CodeableConcept	Reference range population Observation Reference Range Applies To Codes (Example)
age		0..1	Range	Applicable age range, if relevant
text		0..1	string	Text based reference range in an observation
hasMember	Σ	0..*	Reference(Observation | QuestionnaireResponse | MolecularSequence)	Related resource that belongs to the Observation group
derivedFrom	Σ	0..*	Reference(DocumentReference | ImagingStudy | Media | QuestionnaireResponse | Observation | MolecularSequence)	Related measurements the observation is made from
component	Σ	0..*	BackboneElement	Component results
code	Σ	1..1	CodeableConcept	Type of component observation (code / type) LOINC Codes (Example)
value[x]	Σ	0..1		Actual component result
valueQuantity			Quantity	
valueCodeableConcept			CodeableConcept	
valueString			string	
valueBoolean			boolean	
valueInteger			integer	
valueRange			Range	
valueRatio			Ratio	
valueSampledData			SampledData	
valueTime			time	
valueDateTime			dateTime	
valuePeriod			Period	
dataAbsentReason	I	0..1	CodeableConcept	Why the component result is missing DataAbsentReason (Extensible)
interpretation		0..*	CodeableConcept	High, low, normal, etc. Observation Interpretation Codes (Extensible)
referenceRange		0..*	see referenceRange	Provides guide for interpretation of component result

Figura 2.10: Estructura recurso Observation

2.4.1.3.2 Uso

Este recurso se genera en la Base de Datos FHIR cada vez que un especialista o un paciente crea una nueva observación de un paciente o de la realización de una rutina respectivamente.

En el caso del especialista, la observación contiene la siguiente información:

- Código que referencia el tipo de observación (Dato biomédico, historia clínica, …)
- Descripción
- Paciente asociado

- Especialista creador de la observación

Parámetros opcionales:

- Valor númerico y unidad de medida (En caso de ser un dato biomédico como por ejemplo la glucosa en sangre)

En el caso del paciente, la observación contiene la siguiente información:

- Código
- Rutina en la que se basa
- Descripción
- Paciente que la ha creado
- Valor numérico referenciado a cómo se ha sentido el paciente realizando dicha rutina.

2.4.1.3.3 UML

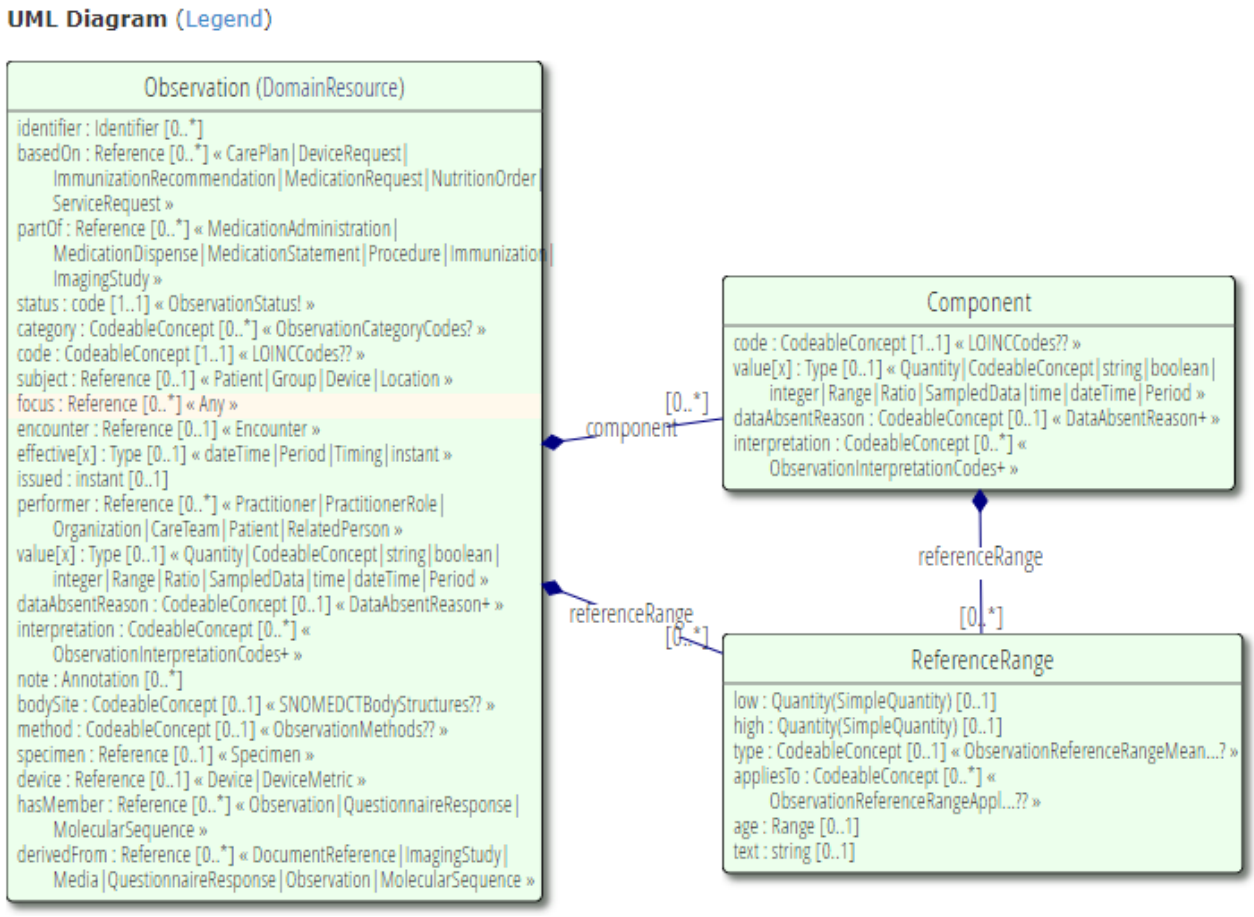


Figura 2.11: UML recurso Observation

2.4.1.3.4 JSON general

```
{
  "resourceType" : "Observation",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier" : [{ Identifier }], // Business Identifier for observation
  "basedOn" : [{ Reference(CarePlan|DeviceRequest|ImmunizationRecommendation|
   MedicationRequest|NutritionOrder|ServiceRequest) }], // Fulfills plan, proposal or order
  "partOf" : [{ Reference(MedicationAdministration|MedicationDispense|
   MedicationStatement|Procedure|Immunization|ImagingStudy) }], // Part of referenced event
  "status" : "<code>", // R!  registered | preliminary | final | amended +
  "category" : [{ CodeableConcept }], // Classification of  type of observation
  "code" : { CodeableConcept }, // R!  Type of observation (code / type)
  "subject" : { Reference(Patient|Group|Device|Location) }, // Who and/or what the observation is about
  "focus" : [{ Reference(Any) }], // What the observation is about, when it is not about the subject of r
ecord
  "encounter" : { Reference(Encounter) }, // Healthcare event during which this observation is made
  // effective[x]: Clinically relevant time/time-period for observation. One of these 4:
  "effectiveDateTime" : "<dateTime>",
  "effectivePeriod" : { Period },
  "effectiveTiming" : { Timing },
  "effectiveInstant" : "<instant>",
  "issued" : "<instant>", // Date/Time this version was made available
  "performer" : [{ Reference(Practitioner|PractitionerRole|Organization|
   CareTeam|Patient|RelatedPerson) }], // Who is responsible for the observation
  // value[x]: Actual result. One of these 11:
  "valueQuantity" : { Quantity },
  "valueCodeableConcept" : { CodeableConcept },
  "valueString" : "<string>",
  "valueBoolean" : <boolean>,
  "valueInteger" : <integer>,
  "valueRange" : { Range },
  "valueRatio" : { Ratio },
  "valueSampledData" : { SampledData },
  "valueTime" : "<time>",
  "valueDateTime" : "<dateTime>",
  "valuePeriod" : { Period },
  "dataAbsentReason" : { CodeableConcept }, // C? Why the result is missing
  "interpretation" : [{ CodeableConcept }], // High, low, normal, etc.
  "note" : [{ Annotation }], // Comments about the observation
  "bodySite" : { CodeableConcept }, // Observed body part
  "method" : { CodeableConcept }, // How it was done
  "specimen" : { Reference(Specimen) }, // Specimen used for this observation
  "device" : { Reference(Device|DeviceMetric) }, // (Measurement) Device
  "referenceRange" : [{ // Provides guide for interpretation
```

```
    "low" : { Quantity(SimpleQuantity) }, // C? Low Range, if relevant
    "high" : { Quantity(SimpleQuantity) }, // C? High Range, if relevant
    "type" : { CodeableConcept }, // Reference range qualifier
    "appliesTo" : [{ CodeableConcept }], // Reference range population
    "age" : { Range }, // Applicable age range, if relevant
    "text" : "<string>" // Text based reference range in an observation
  }],
  "hasMember" : [{ Reference(Observation|QuestionnaireResponse|
   MolecularSequence) }], // Related resource that belongs to the Observation group
  "derivedFrom" : [{ Reference(DocumentReference|ImagingStudy|Media|
   QuestionnaireResponse|Observation|MolecularSequence) }], // Related measurements the observation is ma
de from
  "component" : [{ // Component results
    "code" : { CodeableConcept }, // R!  Type of component observation (code / type)
    // value[x]: Actual component result. One of these 11:
    "valueQuantity" : { Quantity },
    "valueCodeableConcept" : { CodeableConcept },
    "valueString" : "<string>",
    "valueBoolean" : <boolean>,
    "valueInteger" : <integer>,
    "valueRange" : { Range },
    "valueRatio" : { Ratio },
    "valueSampledData" : { SampledData },
    "valueTime" : "<time>",
    "valueDateTime" : "<dateTime>",
    "valuePeriod" : { Period },
    "dataAbsentReason" : { CodeableConcept }, // C? Why the component result is missing
    "interpretation" : [{ CodeableConcept }], // High, low, normal, etc.
    "referenceRange" : [{ Content as for Observation.referenceRange }] // Provides guide for interpretati
on of component result
  }]
}
```

Figura 2.12: JSON general recurso Observation

2.4.1.3.5 Ejemplo práctico


```
{
  "resourceType": "Observation",
  "id": "1352",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2020-12-31T18:04:05.000+00:00",
    "source": "#Boi5RCzuUhDOpSVW"
  },
  "status": "final",
  "code": {
    "coding": [
      {
        "system": "http://loinc.org",
        "code": "12345-00",
        "display": "El paciente \"Juan Alonso Pérez\" muestra mejoría mediante la realización de la rutina titulada \"Levantamiento de peso III\""
      }
    ]
  },
  "subject": {
    "reference": "Patient/52"
  },
  "performer": [
    {
      "reference": "Practitioner/202"
    }
  ],
```

```
{
  "resourceType": "Observation",
  "id": "1353",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2020-12-31T18:06:37.000+00:00",
    "source": "#RDFVCRWGvpAXKFP5"
  },
  "status": "final",
  "code": {
    "coding": [
      {
        "system": "http://loinc.org",
        "code": "12345-01",
        "display": "Glucosa en sangre"
      }
    ]
  },
  "subject": {
    "reference": "Patient/52"
  },
  "performer": [
    {
      "reference": "Practitioner/202"
    }
  ],
  "valueQuantity": {
    "value": 100.0,
    "unit": "mg / dL",
    "system": "http://unitsofmeasure.org"
  }
}
```

```
{
  "resourceType": "Observation",
  "id": "1302",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2020-12-24T14:18:37.000+00:00",
    "source": "#tjBTpscYhUPXfalQ"
  },
  "basedOn": [
    {
      "reference": "CarePlan/1202"
    }
  ],
  "status": "final",
  "code": {
    "coding": [
      {
        "system": "http://loinc.org",
        "code": "11111-00",
        "display": "EJEMPLOOOO"
      }
    ]
  },
  "subject": {
    "reference": "Patient/52"
  },
  "valueQuantity": {
    "value": 1.0,
    "system": "http://unitsofmeasure.org"
  }
}
```


Figura 2.13: Ejemplo recurso Observation

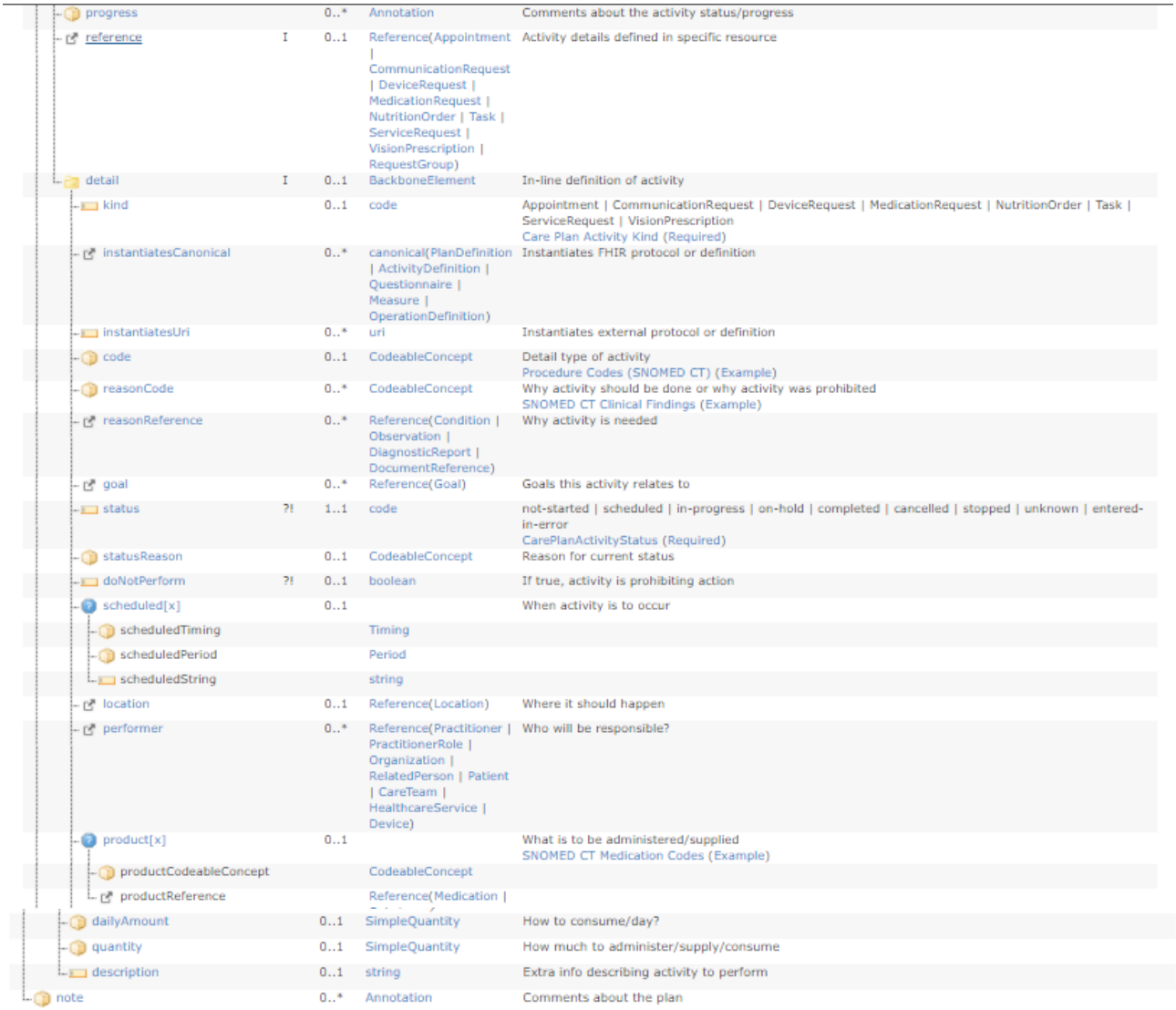
2.4.1.4 CarePlan

Describe cómo uno o más especialistas tienen la intención de atender a un paciente o grupo mediante un plan de atención médica (en nuestro caso, rutinas de ejercicios).

2.4.1.4.1 Estructura

A continuación, en la figura 2.14, se muestra la estructura de atributos del recurso "Observation".

Name	Flags	Card.	Type	Description & Constraints
CarePlan	TU		DomainResource	Healthcare plan for patient or group Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier	Σ	0..*	Identifier	External Ids for this plan
instantiatesCanonical	Σ	0..*	canonical(PlanDefinition | Questionnaire | Measure | ActivityDefinition | OperationDefinition)	Instantiates FHIR protocol or definition
instantiatesUri	Σ	0..*	uri	Instantiates external protocol or definition
basedOn	Σ	0..*	Reference(CarePlan)	Fulfills CarePlan
replaces	Σ	0..*	Reference(CarePlan)	CarePlan replaced by this CarePlan
partOf	Σ	0..*	Reference(CarePlan)	Part of referenced CarePlan
status	?! Σ	1..1	code	draft | active | on-hold | revoked | completed | entered-in-error | unknown RequestStatus (Required)
intent	?! Σ	1..1	code	proposal | plan | order | option Care Plan Intent (Required)
category	Σ	0..*	CodeableConcept	Type of plan Care Plan Category (Example)
title	Σ	0..1	string	Human-friendly name for the care plan
description	Σ	0..1	string	Summary of nature of plan
subject	Σ	1..1	Reference(Patient | Group)	Who the care plan is for
encounter	Σ	0..1	Reference(Encounter)	Encounter created as part of
period	Σ	0..1	Period	Time period plan covers
created	Σ	0..1	dateTime	Date record was first recorded
author	Σ	0..1	Reference(Patient | Practitioner | PractitionerRole | Device | RelatedPerson | Organization | CareTeam)	Who is the designated responsible party
contributor		0..*	Reference(Patient | Practitioner | PractitionerRole | Device | RelatedPerson | Organization | CareTeam)	Who provided the content of the care plan
careTeam		0..*	Reference(CareTeam)	Who's involved in plan?
addresses	Σ	0..*	Reference(Condition)	Health issues this plan addresses
supportingInfo		0..*	Reference(Any)	Information considered as part of plan
goal		0..*	Reference(Goal)	Desired outcome of plan
activity	I	0..*	BackboneElement	Action to occur as part of plan *+ Rule: Provide a reference or detail, not both*
outcomeCodeableConcept		0..*	CodeableConcept	Results of the activity Care Plan Activity Outcome (Example)
outcomeReference		0..*	Reference(Any)	Appointment, Encounter, Procedure, etc.



progress		0..*	Annotation	Comments about the activity status/progress
reference	I	0..1	Reference(Appointment | CommunicationRequest | DeviceRequest | MedicationRequest | NutritionOrder | Task | ServiceRequest | VisionPrescription | RequestGroup)	Activity details defined in specific resource
detail	I	0..1	BackboneElement	In-line definition of activity
kind		0..1	code	Appointment | CommunicationRequest | DeviceRequest | MedicationRequest | NutritionOrder | Task | ServiceRequest | VisionPrescription Care Plan Activity Kind (Required)
instantiatesCanonical		0..*	canonical(PlanDefinition | ActivityDefinition | Questionnaire | Measure | OperationDefinition)	Instantiates FHIR protocol or definition
instantiatesUri		0..*	uri	Instantiates external protocol or definition
code		0..1	CodeableConcept	Detail type of activity Procedure Codes (SNOMED CT) (Example)
reasonCode		0..*	CodeableConcept	Why activity should be done or why activity was prohibited SNOMED CT Clinical Findings (Example)
reasonReference		0..*	Reference(Condition | Observation | DiagnosticReport | DocumentReference)	Why activity is needed
goal		0..*	Reference(Goal)	Goals this activity relates to
status	?!	1..1	code	not-started | scheduled | in-progress | on-hold | completed | cancelled | stopped | unknown | entered-in-error CarePlanActivityStatus (Required)
statusReason		0..1	CodeableConcept	Reason for current status
doNotPerform	?!	0..1	boolean	If true, activity is prohibiting action
scheduled[x]		0..1		When activity is to occur
scheduledTiming			Timing	
scheduledPeriod			Period	
scheduledString			string	
location		0..1	Reference(Location)	Where it should happen
performer		0..*	Reference(Practitioner | PractitionerRole | Organization | RelatedPerson | Patient | CareTeam | HealthcareService | Device)	Who will be responsible?
product[x]		0..1		What is to be administered/supplied SNOMED CT Medication Codes (Example)
productCodeableConcept			CodeableConcept	
productReference			Reference(Medication |	
dailyAmount		0..1	SimpleQuantity	How to consume/day?
quantity		0..1	SimpleQuantity	How much to administer/supply/consume
description		0..1	string	Extra info describing activity to perform
note		0..*	Annotation	Comments about the plan

Figura 2.14: Estructura recurso CarePlan

2.4.1.4.2 Uso

Este recurso se genera en la Base de Datos FHIR cada vez que un especialista crea una nueva rutina de ejercicios. Contiene la siguiente información de la rutina:

- Identificador
- Título
- Descripción
- Paciente o grupo de pacientes ligados a ella
- Autor/creador de la rutina
- Responsable de la rutina
- Ejercicios físicos asociados

2.4.1.4.3 UML

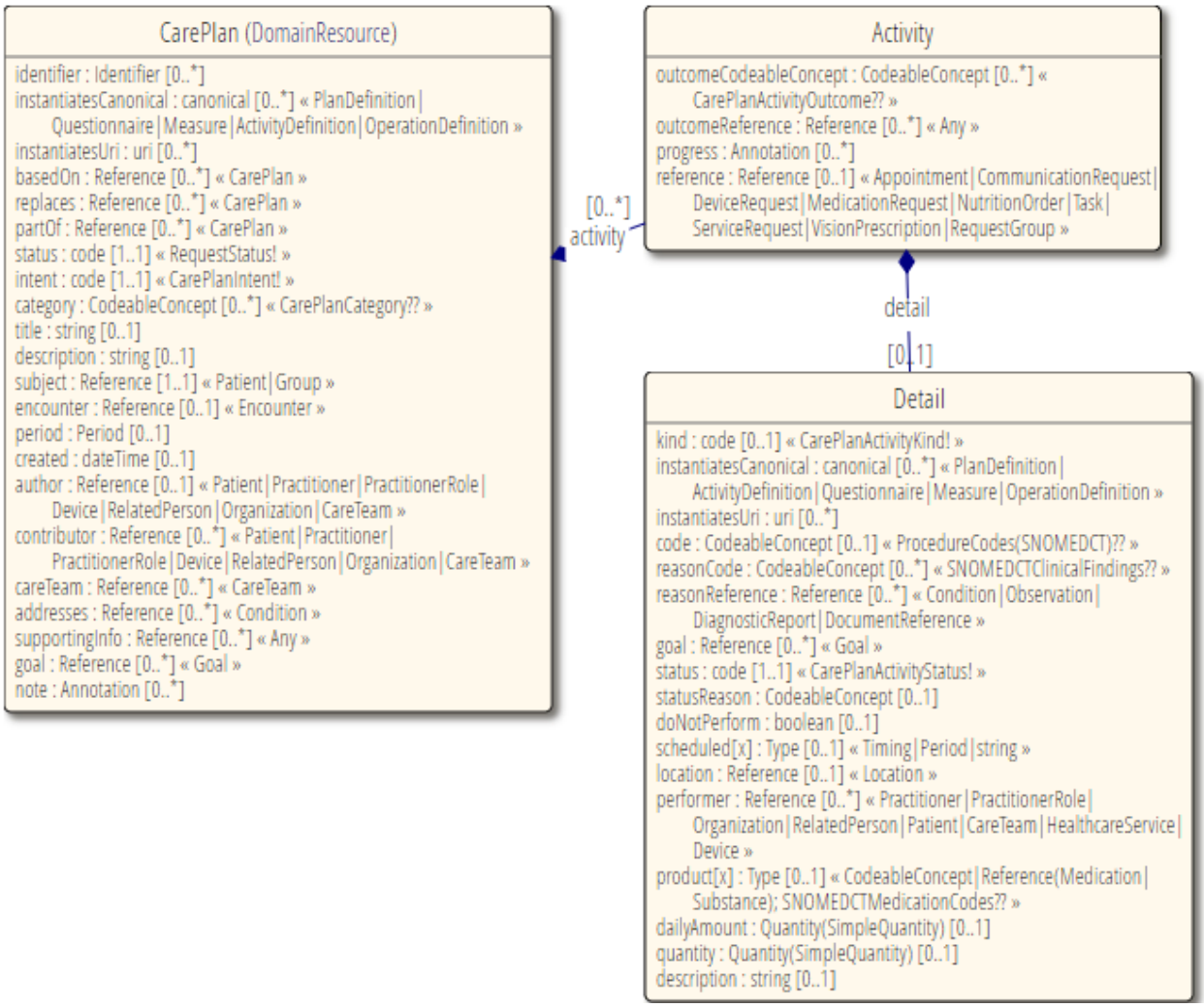


Figura 2.15: UML recurso CarePlan

2.4.1.4.4 JSON general

```
{
  "resourceType" : "CarePlan",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier" : [{ Identifier }], // External Ids for this plan
  "instantiatesCanonical" : [{ canonical(PlanDefinition|Questionnaire|Measure|
   ActivityDefinition|OperationDefinition) }], // Instantiates FHIR protocol or definition
  "instantiatesUri" : ["<uri>"], // Instantiates external protocol or definition
  "basedOn" : [{ Reference(CarePlan) }], // Fulfills CarePlan
  "replaces" : [{ Reference(CarePlan) }], // CarePlan replaced by this CarePlan
  "partOf" : [{ Reference(CarePlan) }], // Part of referenced CarePlan
  "status" : "<code>", // R!  draft | active | on-hold | revoked | completed | entered-in-error | unknown
  "intent" : "<code>", // R!  proposal | plan | order | option
  "category" : [{ CodeableConcept }], // Type of plan
  "title" : "<string>", // Human-friendly name for the care plan
  "description" : "<string>", // Summary of nature of plan
  "subject" : { Reference(Patient|Group) }, // R!  Who the care plan is for
  "encounter" : { Reference(Encounter) }, // Encounter created as part of
  "period" : { Period }, // Time period plan covers
  "created" : "<dateTime>", // Date record was first recorded
  "author" : { Reference(Patient|Practitioner|PractitionerRole|Device|
   RelatedPerson|Organization|CareTeam) }, // Who is the designated responsible party
  "contributor" : [{ Reference(Patient|Practitioner|PractitionerRole|Device|
   RelatedPerson|Organization|CareTeam) }], // Who provided the content of the care plan
  "careTeam" : [{ Reference(CareTeam) }], // Who's involved in plan?
  "addresses" : [{ Reference(Condition) }], // Health issues this plan addresses
  "supportingInfo" : [{ Reference(Any) }], // Information considered as part of plan
  "goal" : [{ Reference(Goal) }], // Desired outcome of plan
  "activity" : [{ // Action to occur as part of plan
    "outcomeCodeableConcept" : [{ CodeableConcept }], // Results of the activity
    "outcomeReference" : [{ Reference(Any) }], // Appointment, Encounter, Procedure, etc.
    "progress" : [{ Annotation }], // Comments about the activity status/progress
    "reference" : { Reference(Appointment|CommunicationRequest|DeviceRequest|
    MedicationRequest|NutritionOrder|Task|ServiceRequest|VisionPrescription|
    RequestGroup) }, // C? Activity details defined in specific resource
    "detail" : { // C? In-line definition of activity
      "kind" : "<code>", // Appointment | CommunicationRequest | DeviceRequest | MedicationRequest | Nutr
itionOrder | Task | ServiceRequest | VisionPrescription
      "instantiatesCanonical" : [{ canonical(PlanDefinition|ActivityDefinition|
     Questionnaire|Measure|OperationDefinition) }], // Instantiates FHIR protocol or definition
      "instantiatesUri" : ["<uri>"], // Instantiates external protocol or definition
      "code" : { CodeableConcept }, // Detail type of activity
      "reasonCode" : [{ CodeableConcept }], // Why activity should be done or why activity was prohibited
      "reasonReference" : [{ Reference(Condition|Observation|DiagnosticReport|
```

```
    DocumentReference) }], // Why activity is needed
     "goal" : [{ Reference(Goal) }], // Goals this activity relates to
     "status" : "<code>", // R!  not-started | scheduled | in-progress | on-hold | completed | cancelled
| stopped | unknown | entered-in-error
     "statusReason" : { CodeableConcept }, // Reason for current status
     "doNotPerform" : <boolean>, // If true, activity is prohibiting action
     // scheduled[x]: When activity is to occur. One of these 3:
     "scheduledTiming" : { Timing },
     "scheduledPeriod" : { Period },
     "scheduledString" : "<string>",
     "location" : { Reference(Location) }, // Where it should happen
     "performer" : [{ Reference(Practitioner|PractitionerRole|Organization|
    RelatedPerson|Patient|CareTeam|HealthcareService|Device) }], // Who will be responsible?
     // product[x]: What is to be administered/supplied. One of these 2:
     "productCodeableConcept" : { CodeableConcept },
     "productReference" : { Reference(Medication|Substance) },
     "dailyAmount" : { Quantity(SimpleQuantity) }, // How to consume/day?
     "quantity" : { Quantity(SimpleQuantity) }, // How much to administer/supply/consume
     "description" : "<string>" // Extra info describing activity to perform
   }
  }],
  "note" : [{ Annotation }] // Comments about the plan
}
```

Figura 2.16: JSON general recurso CarePlan

2.4.1.4.5 Ejemplo práctico


```
{
  "resourceType": "CarePlan",
  "id": "1360",
  "meta": {
    "versionId": "3",
    "lastUpdated": "2020-12-31T19:07:49.000+00:00",
    "source": "#INvTr3BlkjqUdNEa"
  },
  "text": {
    "status": "generated",
    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">This data was generated for test purposes.</div>"
  },
  "identifier": [
    {
      "value": "69"
    }
  ],
  "status": "unknown",
  "intent": "plan",
  "title": "Levantamiento de peso III",
  "description": "Levantamiento de peso para corrección de postura",
  "subject": {
    "reference": "Group/1361"
  },
  "author": {
    "reference": "Practitioner/202"
  },
  "activity": [
    {
      "detail": {
        "status": "unknown",
        "performer": [
          {
            "reference": "Practitioner/620"
          }
        ]
      }
    },
    {
      "reference": {
        "reference": "Task/1377"
      }
    }
  ]
}
```


Figura 2.17: Ejemplo recurso CarePlan

2.4.1.5 Task

Este recurso describe un ejercicio o tarea a realizar.

2.4.1.5.1 Estructura

A continuación, en la figura 2.18, se muestra la estructura de atributos del recurso "Task".

Name	Flags	Card.	Type	Description & Constraints
Task	I TU		DomainResource	A task to be performed *+ Rule: Last modified date must be greater than or equal to authored-on date.* Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier		0..*	Identifier	Task Instance Identifier
instantiatesCanonical	Σ	0..1	canonical(ActivityDefinition)	Formal definition of task
instantiatesUri	Σ	0..1	uri	Formal definition of task
basedOn	Σ	0..*	Reference(Any)	Request fulfilled by this task
groupIdentifier	Σ	0..1	Identifier	Requisition or grouper id
partOf	Σ	0..*	Reference(Task)	Composite task
status	?! Σ	1..1	code	draft | requested | received | accepted | + TaskStatus (Required)
statusReason	Σ	0..1	CodeableConcept	Reason for current status
businessStatus	Σ	0..1	CodeableConcept	E.g. "Specimen collected", "IV prepped"
intent	Σ	1..1	code	unknown | proposal | plan | order | original-order | reflex-order | filler-order | instance-order | option TaskIntent (Required)
priority		0..1	code	routine | urgent | asap | stat Request priority (Required)
code	Σ	0..1	CodeableConcept	Task Type Task Codes (Example)
description	Σ	0..1	string	Human-readable explanation of task
focus	Σ	0..1	Reference(Any)	What task is acting on
for	Σ	0..1	Reference(Any)	Beneficiary of the Task
encounter	Σ	0..1	Reference(Encounter)	Healthcare event during which this task originated
executionPeriod	Σ	0..1	Period	Start and end time of execution
authoredOn	I	0..1	dateTime	Task Creation Date
lastModified	Σ I	0..1	dateTime	Task Last Modified Date
requester	Σ	0..1	Reference(Device | Organization | Patient | Practitioner | PractitionerRole | RelatedPerson)	Who is asking for task to be done
performerType		0..*	CodeableConcept	Requested performer Procedure Performer Role Codes (Preferred)
owner	Σ	0..1	Reference(Practitioner | PractitionerRole | Organization | CareTeam | HealthcareService | Patient | Device | RelatedPerson)	Responsible individual
location	Σ	0..1	Reference(Location)	Where task occurs
reasonCode		0..1	CodeableConcept	Why task is needed
reasonReference		0..1	Reference(Any)	Why task is needed
insurance		0..*	Reference(Coverage | ClaimResponse)	Associated insurance coverage
note		0..*	Annotation	Comments made about the task
relevantHistory		0..*	Reference(Provenance)	Key events in history of the Task
restriction		0..1	BackboneElement	Constraints on fulfillment tasks
repetitions		0..1	positiveInt	How many times to repeat
period		0..1	Period	When fulfillment sought
recipient		0..*	Reference(Patient | Practitioner | PractitionerRole | RelatedPerson | Group | Organization)	For whom is fulfillment sought?
input		0..*	BackboneElement	Information used to perform task
type		1..1	CodeableConcept	Label for the input
value[x]		1..1	*	Content to use in performing the task
output		0..*	BackboneElement	Information produced as part of task
type		1..1	CodeableConcept	Label for output
value[x]		1..1	*	Result of output

Figura 2.18: Estructura recurso Task

2.4.1.5.2 Uso

Este recurso se genera en la Base de Datos FHIR cada vez que un especialista crea un nuevo ejercicio físico. Contiene la siguiente información del ejercicio:

- Identificador
- Descripción
- Creador/Propietario del ejercicio
- Repeticiones o periodicidad del ejercicio
- URL de vídeo asociado al ejercicio
- Estado de forma recomendado para su realización

Parámetros opcionales

- Paciente o grupo de pacientes asociados (si el ejercicio está destinado para ser asociado a una rutina, este campo no es necesario)

2.4.1.5.3 UML

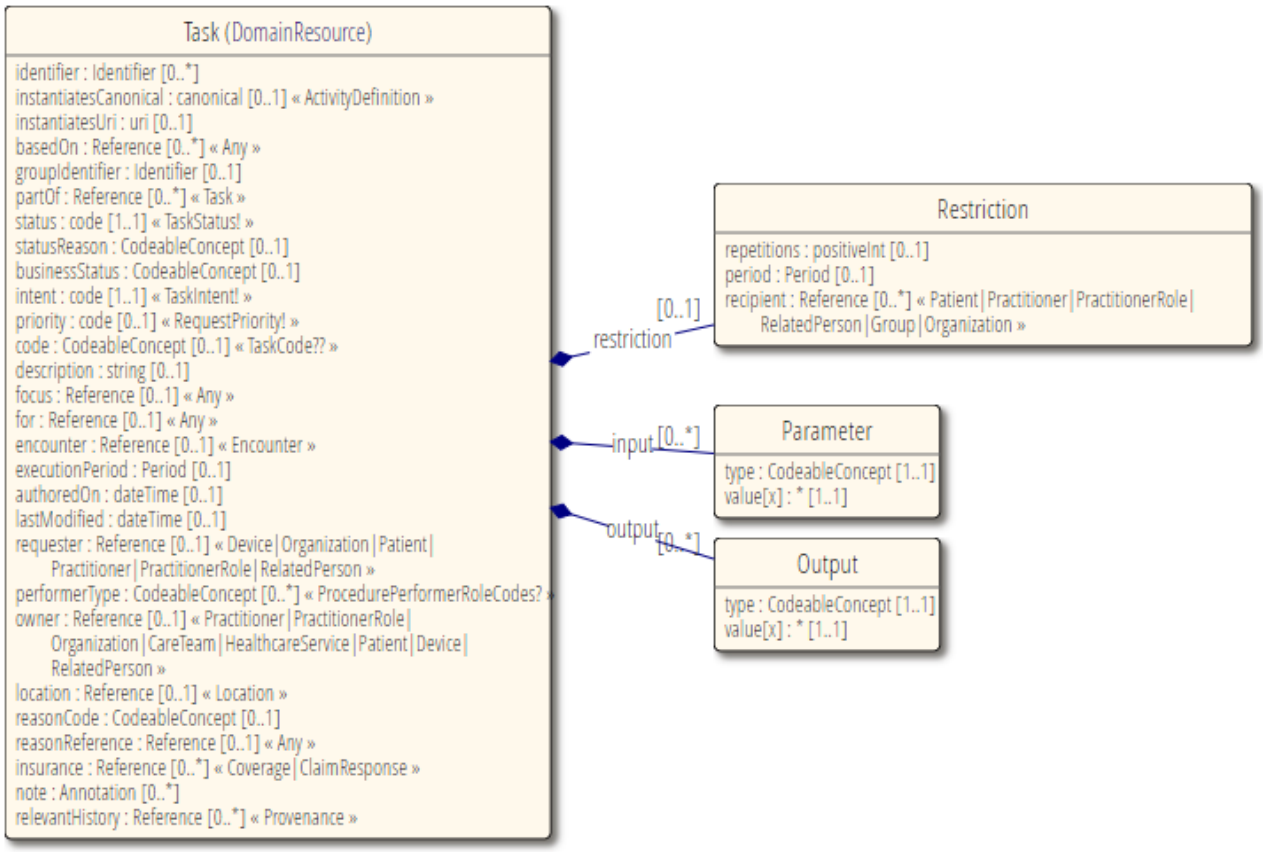


Figura 2.19: Estructura recurso Task

2.4.1.5.4 JSON general

```
{
  "resourceType" : "Task",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier" : [{ Identifier }], // Task Instance Identifier
  "instantiatesCanonical" : { canonical(ActivityDefinition) }, // Formal definition of task
  "instantiatesUri" : "<uri>", // Formal definition of task
  "basedOn" : [{ Reference(Any) }], // Request fulfilled by this task
  "groupIdentifier" : { Identifier }, // Requisition or grouper id
  "partOf" : [{ Reference(Task) }], // Composite task
  "status" : "<code>", // R!  draft | requested | received | accepted | +
  "statusReason" : { CodeableConcept }, // Reason for current status
  "businessStatus" : { CodeableConcept }, // E.g. "Specimen collected", "IV prepped"
  "intent" : "<code>", // R!  unknown | proposal | plan | order | original-order | reflex-order | filler-order | instance-order | option
  "priority" : "<code>", // routine | urgent | asap | stat
  "code" : { CodeableConcept }, // Task Type
  "description" : "<string>", // Human-readable explanation of task
  "focus" : { Reference(Any) }, // What task is acting on
  "for" : { Reference(Any) }, // Beneficiary of the Task
  "encounter" : { Reference(Encounter) }, // Healthcare event during which this task originated
  "executionPeriod" : { Period }, // Start and end time of execution
  "authoredOn" : "<dateTime>", // C? Task Creation Date
  "lastModified" : "<dateTime>", // C? Task Last Modified Date
  "requester" : { Reference(Device|Organization|Patient|Practitioner|
   PractitionerRole|RelatedPerson) }, // Who is asking for task to be done
  "performerType" : [{ CodeableConcept }], // Requested performer
  "owner" : { Reference(Practitioner|PractitionerRole|Organization|CareTeam|
   HealthcareService|Patient|Device|RelatedPerson) }, // Responsible individual
  "location" : { Reference(Location) }, // Where task occurs
  "reasonCode" : { CodeableConcept }, // Why task is needed
  "reasonReference" : { Reference(Any) }, // Why task is needed
  "insurance" : [{ Reference(Coverage|ClaimResponse) }], // Associated insurance coverage
  "note" : [{ Annotation }], // Comments made about the task
  "relevantHistory" : [{ Reference(Provenance) }], // Key events in history of the Task
  "restriction" : { // Constraints on fulfillment tasks
    "repetitions" : "<positiveInt>", // How many times to repeat
    "period" : { Period }, // When fulfillment sought
    "recipient" : [{ Reference(Patient|Practitioner|PractitionerRole|
    RelatedPerson|Group|Organization) }] // For whom is fulfillment sought?
  },
  "input" : [{ // Information used to perform task
    "type" : { CodeableConcept }, // R!  Label for the input
    // value[x]: Content to use in performing the task. One of these 50:
    "valueBase64Binary" : "<base64Binary>"
    "valueBoolean" : <boolean>
    "valueCanonical" : "<canonical>"
    "valueCode" : "<code>"
    "valueDate" : "<date>"
    "valueDateTime" : "<dateTime>"
    "valueDecimal" : <decimal>
    "valueId" : "<id>"
    "valueInstant" : "<instant>"
    "valueInteger" : <integer>
    "valueMarkdown" : "<markdown>"
    "valueOid" : "<oid>"
```

```
    "valuePositiveInt" : "<positiveInt>"
    "valueString" : "<string>"
    "valueTime" : "<time>"
    "valueUnsignedInt" : "<unsignedInt>"
    "valueUri" : "<uri>"
    "valueUrl" : "<url>"
    "valueUuid" : "<uuid>"
    "valueAddress" : { Address }
    "valueAge" : { Age }
    "valueAnnotation" : { Annotation }
    "valueAttachment" : { Attachment }
    "valueCodeableConcept" : { CodeableConcept }
    "valueCoding" : { Coding }
    "valueContactPoint" : { ContactPoint }
    "valueCount" : { Count }
    "valueDistance" : { Distance }
    "valueDuration" : { Duration }
    "valueHumanName" : { HumanName }
    "valueIdentifier" : { Identifier }
    "valueMoney" : { Money }
    "valuePeriod" : { Period }
    "valueQuantity" : { Quantity }
    "valueRange" : { Range }
    "valueRatio" : { Ratio }
    "valueReference" : { Reference }
    "valueSampledData" : { SampledData }
    "valueSignature" : { Signature }
    "valueTiming" : { Timing }
    "valueContactDetail" : { ContactDetail }
    "valueContributor" : { Contributor }
    "valueDataRequirement" : { DataRequirement }
    "valueExpression" : { Expression }
    "valueParameterDefinition" : { ParameterDefinition }
    "valueRelatedArtifact" : { RelatedArtifact }
    "valueTriggerDefinition" : { TriggerDefinition }
    "valueUsageContext" : { UsageContext }
    "valueDosage" : { Dosage }
    "valueMeta" : { Meta }
  }],
  "output" : [{ // Information produced as part of task
    "type" : { CodeableConcept }, // R!  Label for output
    // value[x]: Result of output. One of these 50:
    "valueBase64Binary" : "<base64Binary>"
    "valueBoolean" : <boolean>
    "valueCanonical" : "<canonical>"
    "valueCode" : "<code>"
    "valueDate" : "<date>"
    "valueDateTime" : "<dateTime>"
    "valueDecimal" : <decimal>
    "valueId" : "<id>"
    "valueInstant" : "<instant>"
    "valueInteger" : <integer>
    "valueMarkdown" : "<markdown>"
    "valueOid" : "<oid>"
    "valuePositiveInt" : "<positiveInt>"
    "valueString" : "<string>"
    "valueTime" : "<time>"
```

```
    "valueUnsignedInt" : "<unsignedInt>"
    "valueUri" : "<uri>"
    "valueUrl" : "<url>"
    "valueUuid" : "<uuid>"
    "valueAddress" : { Address }
    "valueAge" : { Age }
    "valueAnnotation" : { Annotation }
    "valueAttachment" : { Attachment }
    "valueCodeableConcept" : { CodeableConcept }
    "valueCoding" : { Coding }
    "valueContactPoint" : { ContactPoint }
    "valueCount" : { Count }
    "valueDistance" : { Distance }
    "valueDuration" : { Duration }
    "valueHumanName" : { HumanName }
    "valueIdentifier" : { Identifier }
    "valueMoney" : { Money }
    "valuePeriod" : { Period }
    "valueQuantity" : { Quantity }
    "valueRange" : { Range }
    "valueRatio" : { Ratio }
    "valueReference" : { Reference }
    "valueSampledData" : { SampledData }
    "valueSignature" : { Signature }
    "valueTiming" : { Timing }
    "valueContactDetail" : { ContactDetail }
    "valueContributor" : { Contributor }
    "valueDataRequirement" : { DataRequirement }
    "valueExpression" : { Expression }
    "valueParameterDefinition" : { ParameterDefinition }
    "valueRelatedArtifact" : { RelatedArtifact }
    "valueTriggerDefinition" : { TriggerDefinition }
    "valueUsageContext" : { UsageContext }
    "valueDosage" : { Dosage }
    "valueMeta" : { Meta }
  }]
}
```

Figura 2.20: JSON general recurso Task

2.4.1.5.5 Ejemplo práctico


```
{
  "resourceType": "Task",
  "id": "1377",
  "meta": {
    "versionId": "3",
    "lastUpdated": "2020-12-31T19:12:58.000+00:00",
    "source": "#RGxAjRuQFrwtIRlp"
  },
  "text": {
    "status": "additional",
    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">This is the narrative text<br/>this is line 2</div>"
  },
  "identifier": [
    {
      "value": "48"
    }
  ],
  "instantiatesUri": "Nombre->Pesas,URL->https://www.youtube.com/watch?v=7KL8SgCP4KQ&ab_channel=Fiteligente",
  "status": "on-hold",
  "intent": "plan",
  "description": "Colócate con las rodillas ligeramente flexionadas y la espalda derecha. Lleva la cadera hacia atrás,
como si la levantaras al cielo. Sujeta la barra con un agarre ligeramente más abierto que tus piernas. Sube sin arquear
la espalda y lleva el peso a la parte media del cuerpo. Regresa con un movimiento controlado, este lo lograrás al
apretar el abdomen y los glúteos.",
  "owner": {
    "reference": "Practitioner/202"
  },
  "note": [
    {
      "text": "Estado de forma del paciente: Bajo"
    }
  ],
  "restriction": {
    "repetitions": 4
  }
}
```


Figura 2.21: Ejemplo recurso Task

2.4.1.6 Group

Este recurso representa un conjunto de entidades definidas. En este caso, dichas entidades son los pacientes.

2.4.1.6.1 Estructura

A continuación, en la figura 2.22, se muestra la estructura de atributos del recurso "Group".

Name	Flags	Card.	Type	Description & Constraints
Group	I TU		DomainResource	Group of multiple entities *+ Rule: Can only have members if group is "actual"* Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier	Σ	0..*	Identifier	Unique id
active	Σ	0..1	boolean	Whether this group's record is in active use
type	Σ	1..1	code	person | animal | practitioner | device | medication | substance GroupType (Required)
actual	Σ I	1..1	boolean	Descriptive or actual
code	Σ	0..1	CodeableConcept	Kind of Group members
name	Σ	0..1	string	Label for Group
quantity	Σ	0..1	unsignedInt	Number of members
managingEntity	Σ	0..1	Reference(Organization | RelatedPerson | Practitioner | PractitionerRole)	Entity that is the custodian of the Group's definition
characteristic		0..*	BackboneElement	Include / Exclude group members by Trait
code		1..1	CodeableConcept	Kind of characteristic
value[x]		1..1		Value held by characteristic
valueCodeableConcept			CodeableConcept	
valueBoolean			boolean	
valueQuantity			Quantity	
valueRange			Range	
valueReference			Reference()	
exclude		1..1	boolean	Group includes or excludes
period		0..1	Period	Period over which characteristic is tested
member	I	0..*	BackboneElement	Who or what is in group
entity		1..1	Reference(Patient | Practitioner | PractitionerRole | Device | Medication | Substance | Group)	Reference to the group member
period		0..1	Period	Period member belonged to the group
inactive		0..1	boolean	If member is no longer in group

Figura 2.22: Estructura recurso Group

2.4.1.6.2 Uso

Este recurso se genera en la Base de Datos FHIR cada vez que se asocia una rutina o ejercicio a un conjunto de pacientes. Contiene la siguiente información:

- Los pacientes que forman parte de ese conjunto o grupo.

2.4.1.6.3 UML

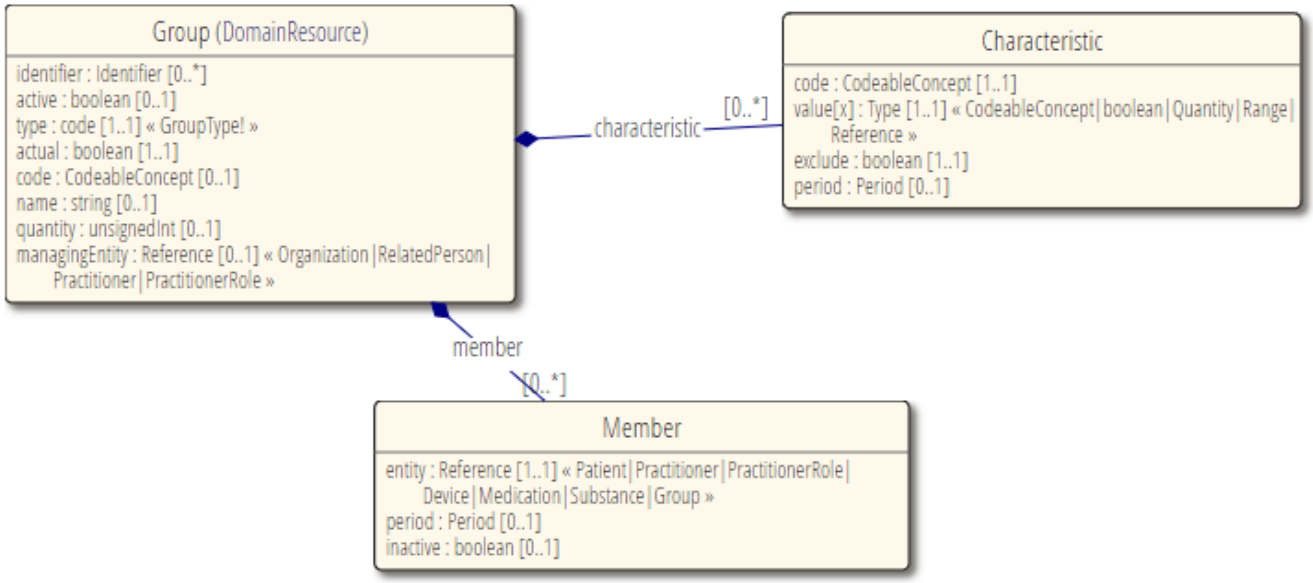


Figura 2.23: UML recurso Group

2.4.1.6.4 JSON general

```
{
  "resourceType" : "Group",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier" : [{ Identifier }], // Unique id
  "active" : <boolean>, // Whether this group's record is in active use
  "type" : "<code>", // R!  person | animal | practitioner | device | medication | substance
  "actual" : <boolean>, // C? R!  Descriptive or actual
  "code" : { CodeableConcept }, // Kind of Group members
  "name" : "<string>", // Label for Group
  "quantity" : "<unsignedInt>", // Number of members
  "managingEntity" : { Reference(Organization|RelatedPerson|Practitioner|
   PractitionerRole) }, // Entity that is the custodian of the Group's definition
  "characteristic" : [{ // Include / Exclude group members by Trait
    "code" : { CodeableConcept }, // R!  Kind of characteristic
    // value[x]: Value held by characteristic. One of these 5:
    "valueCodeableConcept" : { CodeableConcept },
    "valueBoolean" : <boolean>,
    "valueQuantity" : { Quantity },
    "valueRange" : { Range },
    "valueReference" : { Reference },
    "exclude" : <boolean>, // R!  Group includes or excludes
    "period" : { Period } // Period over which characteristic is tested
  }],
  "member" : [{ // C? Who or what is in group
    "entity" : { Reference(Patient|Practitioner|PractitionerRole|Device|
    Medication|Substance|Group) }, // R!  Reference to the group member
    "period" : { Period }, // Period member belonged to the group
    "inactive" : <boolean> // If member is no longer in group
  }]
}
```

Figura 2.24: Estructura recurso Group

2.4.1.6.5 Ejemplo práctico


```
{
  "resourceType": "Group",
  "id": "1207",
  "meta": {
    "versionId": "3",
    "lastUpdated": "2020-12-22T17:59:05.000+00:00",
    "source": "#54XTtaWVvjWK8UmK"
  },
  "identifier": [
    {
      "value": "64"
    }
  ],
  "type": "person",
  "actual": true,
  "member": [
    {
      "entity": {
        "reference": "Patient/1"
      }
    },
    {
      "entity": {
        "reference": "Patient/52"
      }
    }
  ]
}
```


Figura 2.25: Ejemplo recurso Group

2.4.2 Diagrama relacional de los recursos

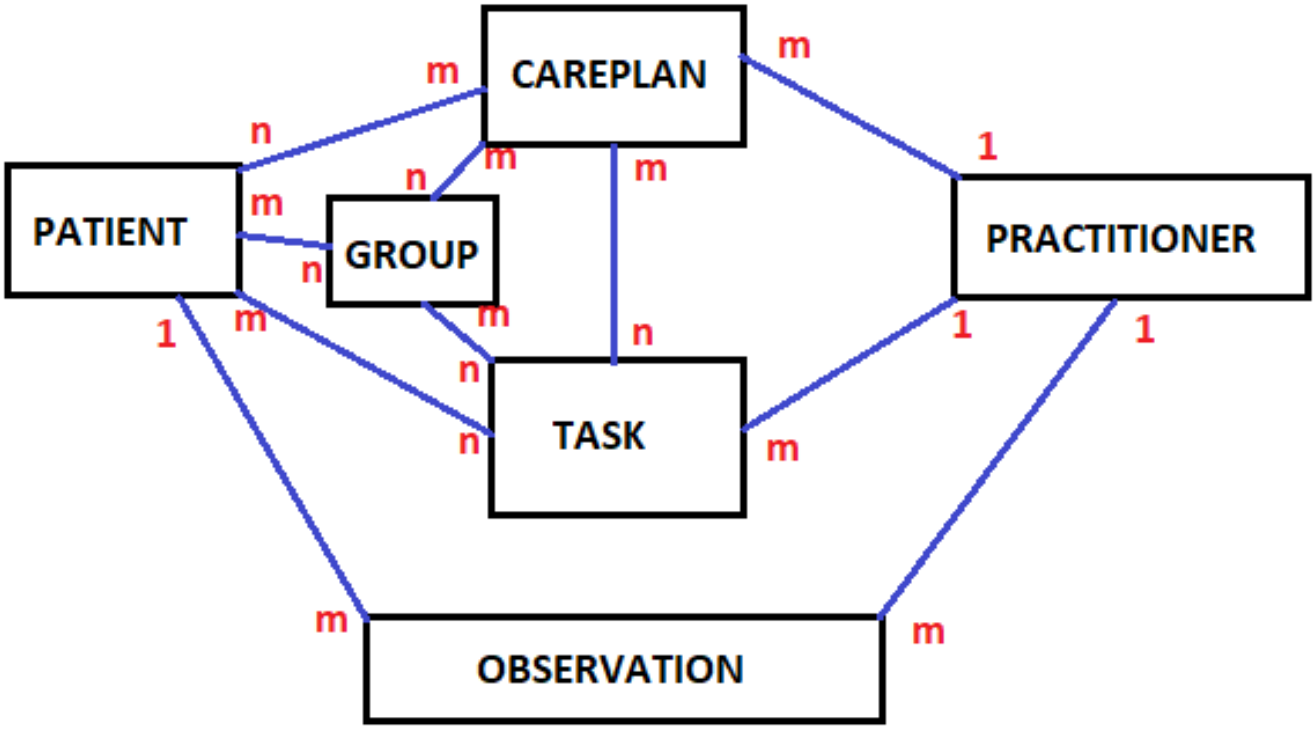


Figura 2.26: Diagrama ER

3 OTRAS TECNOLOGÍAS UTILIZADAS

3.1 Servicio Web REST

3.1.1 Arquitectura REST

La arquitectura REST o Transferencia de Estado Representacional es un estilo de arquitectura introducido y definido en el año 2000 por Roy Fielding en su tesis doctoral. Fielding es uno de los principales autores del protocolo de transferencia de hipertexto (HTTP).

Esta arquitectura hace énfasis en que las interacciones entre los clientes y los servicios se mejoran al tener un número limitado de operaciones (verbos). La flexibilidad se obtiene asignando recursos a sus propios identificadores de recursos universales únicos (URI). Debido a que cada verbo tiene un significado específico (GET, POST, PUT y DELETE), REST evita la ambigüedad:

- ✓ GET: Obtener un recurso.
- ✓ POST: Crear un recurso en el servidor.
- ✓ PUT: Cambiar el estado de un recurso o actualizarlo.
- ✓ DETELE: Eliminar un recurso.

REST es preferible a la arquitectura SOAP (Simple Object Access Protocol), debido a que REST es más ligero, lo que hace que sea una mejor opción para su uso a través de internet.

Figura 3.1: Logo HTTP REST.

Las URIs en las que se encuentran alojados los recursos que obtenemos a través de peticiones HTTP REST pueden verse en el Anexo A.

3.2 Formato para el intercambio de datos (JSON)

3.2.1 Introducción

Cuando intercambiamos datos entre el navegador Web del cliente y el servidor necesitamos gestionar de forma sencilla y eficaz mucha información. JSON es un estándar para el intercambio de datos siguiendo una sintaxis específica y definida de forma que podemos intercambiar información entre diferentes

aplicaciones independientemente de la plataforma o lenguaje en el que hayan sido desarrolladas. JSON nació como una alternativa a XML, debido a su gran facilidad de uso con JavaScript.

Se destaca principalmente porque este formato puede ser leído por cualquier lenguaje de programación, lo cual nos proporciona mayor integridad y flexibilidad.

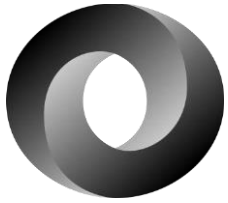


Figura 3.2: Logo de JSON.

3.2.2 Otros formatos (XML)

XML ,usado para algunos ficheros de configuración de Spring, es un lenguaje de marcado que define un conjunto de reglas para codificar información de forma que sea legible por un ser humano o por un ordenador. Gracias a la flexibilidad que proporciona, carece de las limitaciones que tiene HTML y puede ser usada para representar cualquier estructura de datos. Como ventaja principal de este lenguaje, cabe destacar que proporciona soporte Unicode, lo cual permite escribir la información en cualquier idioma del mundo.

Figura 3.3: Logo de XML

3.3 Tecnologías en el Servidor: Java Spring Framework

3.3.1 Introducción: Spring

Este es un framework que se usa para el desarrollo de Servicios y aplicaciones, y como contenedor de inversión de control de código abierto para la plataforma Java.

Spring fue escrito para la plataforma J2EE de Java, plataforma orientada al desarrollo de aplicaciones web, y ha ido evolucionando rápidamente hasta el día de hoy, donde podemos encontrar diferentes ramas de desarrollo de la mano de SpringSource y todo su equipo de desarrolladores.

Figura 3.4: Logo de Spring Framework

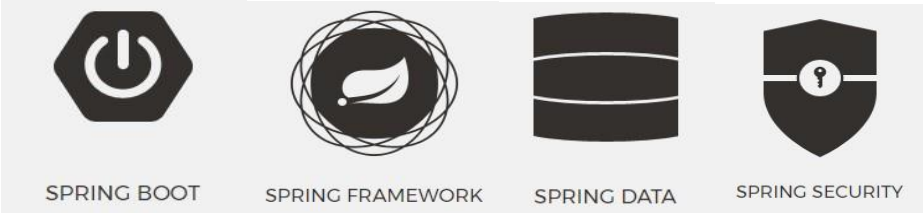


Figura 3.5: Módulos de Spring Usados en este proyecto.

3.3.2 Spring Boot

Spring Boot es un módulo de Spring que nos permite crear diferentes tipos de aplicaciones autónomas de una manera rápida y sencilla, donde simplemente hay que ejecutarlas, simplificando la creación de dependencias y el despliegue en un servidor de aplicaciones.

Principalmente se trata de un mecanismo centrado en una clase principal, la cual tiene una configuración por defecto y que podemos personalizar. Este es compatible con Maven y Gradle, y su formato por defecto para el despliegue es ".JAR", aunque también soporta ".WAR".

Figura 3.6: Logo de Spring Boot.

3.3.3 Spring Security

Spring Security es un módulo que permitirá gestionar todo lo relativo a la seguridad de nuestra aplicación web, desde el protocolo de seguridad, hasta los roles que necesitan los usuarios para acceder a los diferentes recursos de la aplicación. Desde el punto de vista de la autenticación, nos permite comprobar que un usuario exista y permitir el acceso a los recursos de la aplicación, dando autorización a unos recursos u otros según sus credenciales y sus privilegios.

Este módulo consta de una serie de elementos, como clases y parámetros en Java, los cuales hay que personalizar para que se adapte de la mejor manera posible al comportamiento y finalidad que queremos llegar a conseguir en nuestra aplicación Web.

Figura 3.7: Logo de Spring Security.

3.3.4 Spring Data

Spring Framework ya proporcionaba soporte para el acceso a bases de datos como JDBC, Hibernate o JPA, simplificando la implementación de la capa de acceso a datos, unificando la configuración y creando una jerarquía de excepciones común para todas ellas.

Spring Data permite cubrir el soporte necesario para todas las tecnologías de persistencia y , además, integra las tecnologías de acceso tradicionales, simplificando el trabajo a la hora de crear las implementaciones concretas.

Figura 3.8: Logo de Spring Data

3.3.4.1 Patrón DAO (Data Access Object)

Este patrón de diseño software pertenece al catálogo de Core J2EE Patterns de Java, por lo que no es un patrón exclusivo de Spring, aunque es el utilizado para realizar aplicaciones con este framework que hagan uso de la persistencia.

El patrón DAO muestra una forma de envolver el conocimiento sobre el acceso a datos, mediante el uso de objetos de acceso a datos para conseguir abstraer y encapsular el acceso a dichos datos.

Un objeto DAO permite obtener y guardar datos, manejando la conexión con el sistema de persistencia que se haya implementado. Todo ello lo realiza mediante operaciones atómicas (Creación, actualización, borrado, obtención de registros) en la base de datos, por lo que nunca son necesarias las transacciones. Normalmente, se crea un objeto DAO por cada Objeto que tengamos en nuestra aplicación.

3.3.4.2 JDBC (Java Database Connectivity)

JDBC es una API que describe o define una librería estándar para el acceso a Fuentes de datos desde el lenguaje de programación Java, independientemente del Sistema operativo donde se ejecute o de la base de datos a la que se acceda, principalmente orientado a bases de datos relacionales.

Figura 3.9: Logo de JDBC

Esta API consiste en un conjunto de interfaces y clases escritas en Java. Con estas interfaces y clases estándar, los programadores pueden escribir aplicaciones que conecten con la base de datos, enviar consultas escritas en el lenguaje de consulta estructurada "SQL" y posteriormente procesar los resultados obtenidos.

3.3.5 Spring Beans

Un bean se define como un componente de software que tiene la particularidad de ser reutilizable. Estos deben cumplir ciertos criterios en java:

- Implementación serializable.
- Poseer todos sus atributos privados (private).
- Poseer métodos "set" y "get" públicos de los atributos privados que nos interese.
- Poseer un constructor público por defecto.

Figura 3.10: Logo de Java Beans.

A diferencia de los beans convencionales que representan una clase, la particularidad de los beans en Spring es que son objetos creados y manejados por el contenedor de Spring.

3.3.6 Construcción y Gestión de Software (Maven)

Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores. Maven provee soporte no solo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios.

Figura 3.11: Logo de Maven.

Todo ello se hará de forma automática sin que el usuario tenga que hacer nada más que definir las dependencias en el formato que mostramos a continuación:

```
<dependencies>
<!-- This dependency includes the core HAPI-FHIR classes -->
    <dependency>
        <groupId>ca.uhn.hapi.fhir</groupId>
        <artifactId>hapi-fhir-base</artifactId>
        <version>${hapifhir_version}</version>
    </dependency>

  <!-- Include the client -->
    <dependency>
        <groupId>ca.uhn.hapi.fhir</groupId>
        <artifactId>hapi-fhir-client</artifactId>
        <version>${hapifhir_version}</version>
  </dependency>

    <!-- At least one "structures" JAR must also be included -->
    <dependency>
        <groupId>ca.uhn.hapi.fhir</groupId>
        <artifactId>hapi-fhir-structures-r4</artifactId>
        <version>${hapifhir_version}</version>
    </dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web-services</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-logging</artifactId>
    </dependency>
    <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
```

Figura 3.11.1: pom.xml

3.4 Persistencia: Base de datos

3.4.1 Introducción: Modelo Relacional

El modelo relacional es una alternativa para la organización y representación de la información que se pretende almacenar en una base de datos. Se trata de un modelo teórico matemático que, además de proporcionarnos los elementos básicos de modelado (las relaciones), incluye un conjunto de operadores (definidos en forma de un álgebra relacional) para su manipulación, sin ambigüedad posible.

Este modelo, hace relativamente sencilla su representación y gestión por medio de herramientas informáticas, siendo elegido como referencia para la gran mayoría de los sistemas de Gestión de Bases de Datos disponibles en el mercado. Además, también es seleccionado como referencia para la elaboración del esquema lógico de una base de datos en su diseño, junto con herramientas como el álgebra relacional.

3.4.2 Gestor MySQL

Es un sistema de gestión de bases de datos relacional desarrollado por Oracle Corporation, considerado como el más popular del mundo, además de ser desarrollado y distribuido libremente (Open Source).

MySQL es multiplataforma, lo cual da soporte a una amplia lista de sistemas operativos y trae soporte para aproximadamente diez motores de almacenamiento, en los cuales se encuentran InnoDB que es el motor predeterminado, que soporta transacciones y bloqueo de registros. Esto hace de este gestor que posea un alto rendimiento comparado con sistemas similares.

Figura 3.12: Logo de MySQL.

Por último, este sistema de gestión de bases de datos provee de varias herramientas de gestión y de diseño de bases de datos de forma interactiva y gráfica, como son XAMPP y MySQLWorkbench. Estas herramientas nos han facilitado el diseño y montaje de la base de datos, además de su gestión.

3.5 Tecnologías para la interfaz web de usuario

3.5.1 Patrón Modelo-Vista-Controlador (MVC)

El objetivo principal de este patrón es el de realizar una separación de los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello, este patrón define tres componentes o módulos básicos:

- **Modelo:** el modelo representa la parte de la aplicación que implementa la lógica de negocio. Esto significa que es responsable de la recuperación de datos convirtiéndolos en conceptos significativos para la aplicación, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos.
- **Vista:** es una presentación de los datos del modelo al usuario, estando separada de los objetos del modelo. Es responsable del uso de la información de la cual se dispone para producir cualquier interfaz de presentación de cualquier petición que se presente.
- **Controlador:** gestiona las peticiones de los usuarios. Es responsable de responder a la información solicitada con la ayuda tanto del modelo como de la vista.

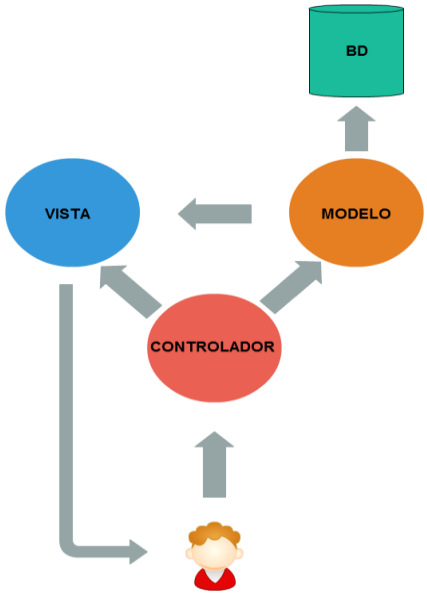


Figura 3.13: Esquema del Patrón MVC.

3.5.2 Bootstrap (HTML5 + CSS3 + JavaScript)

Este es un framework o conjunto de herramientas de código abierto para el diseño de sitios y aplicaciones Web. Contiene plantillas de diseño de tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basados en HTML y CSS, así como extensiones de JavaScript adicionales. Además es compatible con la mayoría de los navegadores web.

Figura 3.14: Logo de BootStrap.

3.5.2.1 HTML5

HTML es un lenguaje de marcado utilizado para definir la estructura y contenido de una página o documento Web. La idea es utilizar un lenguaje para hacer referencia a otros documentos, como archivos, imágenes, videos, audio, etc.

Es por tanto que HTML5 se trata de una nueva versión de HTML en la cual se especifican nuevos elementos, atributos y comportamientos. Además, contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance.

Figura 3.15: Logo de HTML5.

A través de HTML5 hemos conseguido alojar videos de Youtube en nuestra aplicación web sin necesidad de subirlos y alojarlos en el servidor físicamente, lo que hubiera supuesto una gran necesidad de recursos para ello. El propio Youtube te proporciona el código para ello.

Figura 3.16: Logo de Youtube

El código se obtiene pulsando en la opción de COMPARTIR en cualquier vídeo de Youtube que quieras y escogiendo INSERTAR .

Figura 3.16.1: Código de Youtube

3.5.2.2 CSS3

CSS es un lenguaje usado para definir el estilo o apariencia de las páginas Web, escritas con HTML o de los documentos XML. Este lenguaje se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas.

Figura 3.17: Logo de CSS3.

3.5.2.3 JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Este estándar es también un lenguaje de programación interpretado, definido como orientado a objetos y basado en prototipos, imperativo y dinámico. Se utiliza principalmente en su forma del lado cliente, implementado como una parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

Figura 3.18: Logo de JavaScript.

Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

El modelo DOM es esencialmente una interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.

Figura 3.19: Logo de DOM.

Tradicionalmente, JavaScript se venía utilizando en páginas Web HTML para realizar operaciones y únicamente en el marco de la aplicación del cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con la ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

3.5.3 Controlador de cliente: JQuery/Ajax

3.5.3.1 JQuery

JQuery es la biblioteca ECMAScript más extendida y usada en el mundo. Esta biblioteca de software libre y de código abierto permite simplificar la forma de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción mediante AJAX a páginas Web.

Figura 3.20: Logo de JQuery.

3.5.3.2 AJAX, Asynchronous JavaScript And XML

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas completamente, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Figura 3.21: Logo de AJAX.

3.5.3.3 JQuery AJAX

JQuery facilita el trabajo con AJAX mediante métodos que simplifican todo el proceso anterior. La más importante es ".ajax()", que realiza una petición asíncrona y tiene multitud de opciones.

Figura 3.22: Logo de JQuery AJAX.

En nuestro caso, usaremos AJAX a través del método $.ajax() que es la implementación de dicha tecnología a través de JavaScript, concretamente de la biblioteca JQuery, que es configurado a través de un objeto, el cual contiene todas las instrucciones que necesita JQuery para completar una petición de este tipo. Dicho método es muy útil debido a que ofrece la posibilidad de especificar acciones en caso de que la petición haya fallado o no. Además, al estar configurado a través de un objeto, es posible definir sus propiedades de forma separada, haciendo que sea más fácil la reutilización del código.

```
$.ajax({
  url: '/ruta/hasta/pagina.php',
  type: 'POST',
  async: true,  // Valor por defecto, no necesario
  data: 'parametro1=valor1&parametro2=valor2',
              // Datos que se  envían al servidor
  success: procesaRespuesta,
 // función que procesa los datos devueltos por el servidor
  error: muestraError // función invocada si la petición falla
});
```

Figura 3.23: Ejemplo de sintaxis de una consulta simple en AJAX.

3.5.3.4 JQuery Validation Plugin

Este es un Plugin de JQuery cuya finalidad es realizar la validación de formularios HTML en el lado del cliente, ofreciendo una gran cantidad de opciones personalizables.

Figura 3.24: Logo de JQuery Plugin Validation.

3.5.4 HAPI FHIR

HAPI FHIR es una implementación completa del estándar HL7 FHIR para la interoperabilidad de la asistencia sanitaria en Java. HAPI FHIR es un producto de Smile CDR. HAPI define modelos de clases para cada tipo de recurso y tipo de datos definidos por la especificación FHIR. La especificación FHIR está diseñada para ser legible e implementable.

HAPI está diseñado con una intención principal: proporcionar una forma flexible de agregar FHIR a las aplicaciones. Este proyecto fue desarrollado originalmente en University Health Network para permitir a UHN construir un sistema de servicios FHIR unificados para exponer datos respaldados por una serie de sistemas y repositorios.

Figura 3.25: Logo HAPI FHIR.

3.5.5 Contenedores

3.5.5.1 Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.

En este proyecto se ha empleado para contener al servidor HAPI FHIR.

Figura 3.26: Logo Docker.

3.5.5.2 YAML

YAML es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl. Sus siglas significan *YAML Ain't Markup Language* (YAML no es otro lenguaje de marcado). Este lenguaje es muy legible para las personas, más legible que un **JSON** y sobretodo que **XML**. Se utiliza normalmente, por ejemplo, para archivos de configuración.

En este caso, ha sido utilizado para definir uno de los ficheros de configuración del Docker (contenedor) en el que se ejecuta el servidor HAPI FHIR.

Figura 3.27: YAML.

4 HERRAMIENTAS UTILIZADAS

4.1 Spring Tool Suite (STS)

Spring Tool Suite proporciona un entorno *ready-to-use* para implementar, depurar, ejecutar y desplegar las aplicaciones Spring, incluyendo integraciones para Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven, AspectJ, y viene encima de las últimas versiones de Eclipse.

Este entorno de desarrollo incluye la edición para desarrolladores de Pivotal tc Server, una versión de Apache Tomcat optimizado para Spring. Con su consola Spring Insight, tc Server Developer Edition ofrece una visión en tiempo real gráfica de los parámetros de rendimiento de aplicaciones que permite a los desarrolladores identificar y diagnosticar los problemas desde sus escritorios.

Además, soporta el despliegue de aplicaciones tanto en servidores locales, virtuales y en la nube. Es de libre acceso para el desarrollo y uso en operaciones internas sin límite de tiempo, completamente de código abierto y licenciada bajo los términos de la Licencia Pública Eclipse.

Figura 4.1: Logo de Spring Tool Suite.

Algunas de las características que podemos destacar son las siguientes:

- ✓ **Validaciones para la configuración de Spring:** ofrece un amplio conjunto de validaciones que se están aplicando de forma automática. Esas validaciones indican errores en las configuraciones directamente en el IDE, mucho antes de que sea ejecutada la aplicación. Encontrar problemas y errores de configuración es mucho más fácil.
- ✓ **Soporte Refactoring para su aplicación Spring:** el término refactorizacón en ingeniería del Software es usado para describir la modificación del código fuente sin alterar el comportamiento del mismo. Es una de las partes más importantes de la ingeniería de software de hoy. El IDE agrega nuevas refactorizaciones para los elementos de Spring (como el cambio de nombre de los beans de Spring, por ejemplo).
- ✓ **Code Assistant:** proporciona contenido de ayuda significativo en todos lados, junto con soluciones rápidas para los errores y problemas comunes.
- ✓ **Soporte AOP (Programación Orientada a Aspectos):** la Spring Tool Suite se integra con las herramientas de Eclipse y proporciona el soporte más completo para AOP disponible hoy en día.

- ✓ **Integrado con Cloud Foundry y Pivotal tc Server:** permite el despliegue de las aplicaciones directamente en Cloud Foundry o una instancia tc Server (incluyendo el soporte para la depuración, creación de la instancia, Spring Insight, servicios y más).

En nuestro proyecto hemos utilizado este framework para construir nuestra aplicación web.

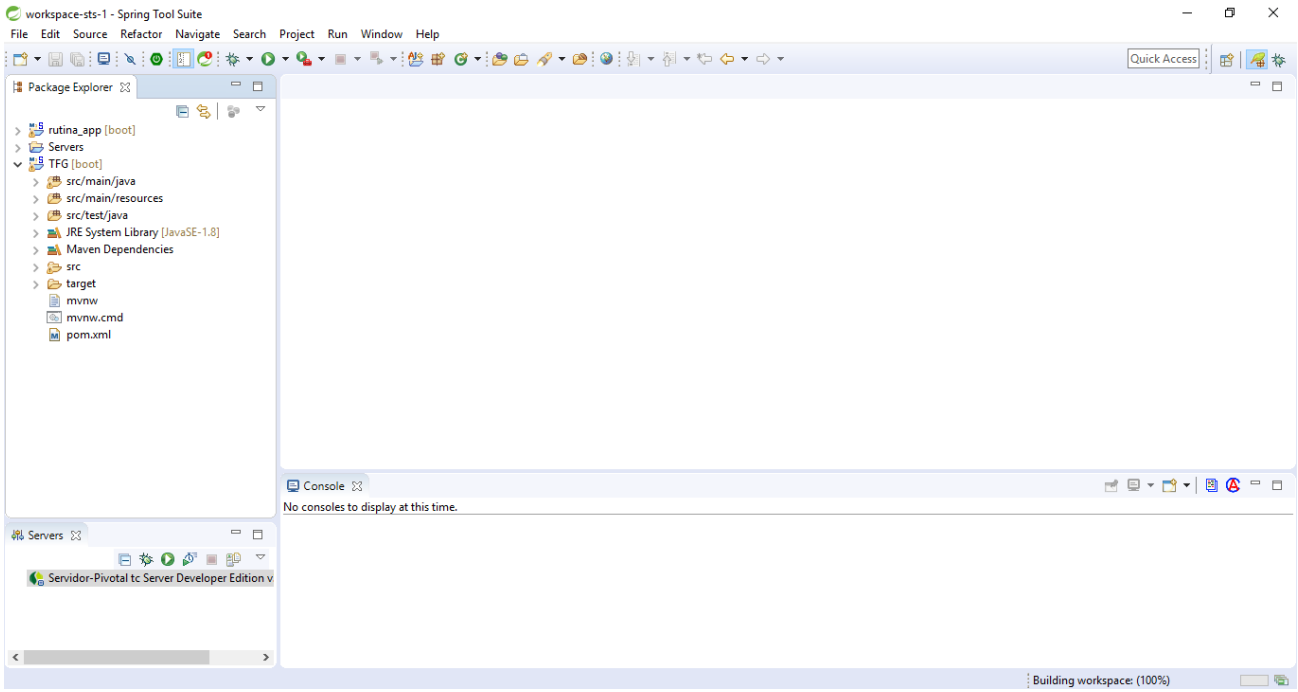


Figura 4.2: Interfaz Gráfica de STS.

4.2 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOs. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft

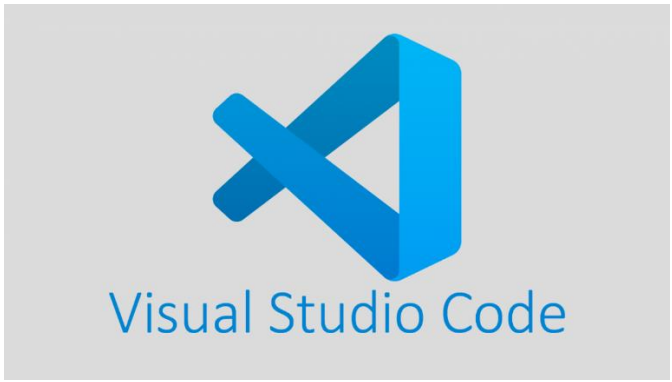


Figura 4.3: Visual Studio Code

Visual Studio Code se basa en Electron, un framework que se utiliza para implementar Chromium y Node.js como aplicaciones para escritorio, que se ejecuta en el motor de diseño Blink. Aunque utiliza el framework Electron, el software no usa Atom y en su lugar emplea el mismo componente editor (Monaco) utilizado en Visual Studio Team Services (anteriormente llamado Visual Studio Online).

En este proyecto, esta herramienta se ha utilizado para configurar y desplegar tanto el servidor HAPI FHIR, como el Docker que lo contiene, ya que facilita con plugins su implementación.

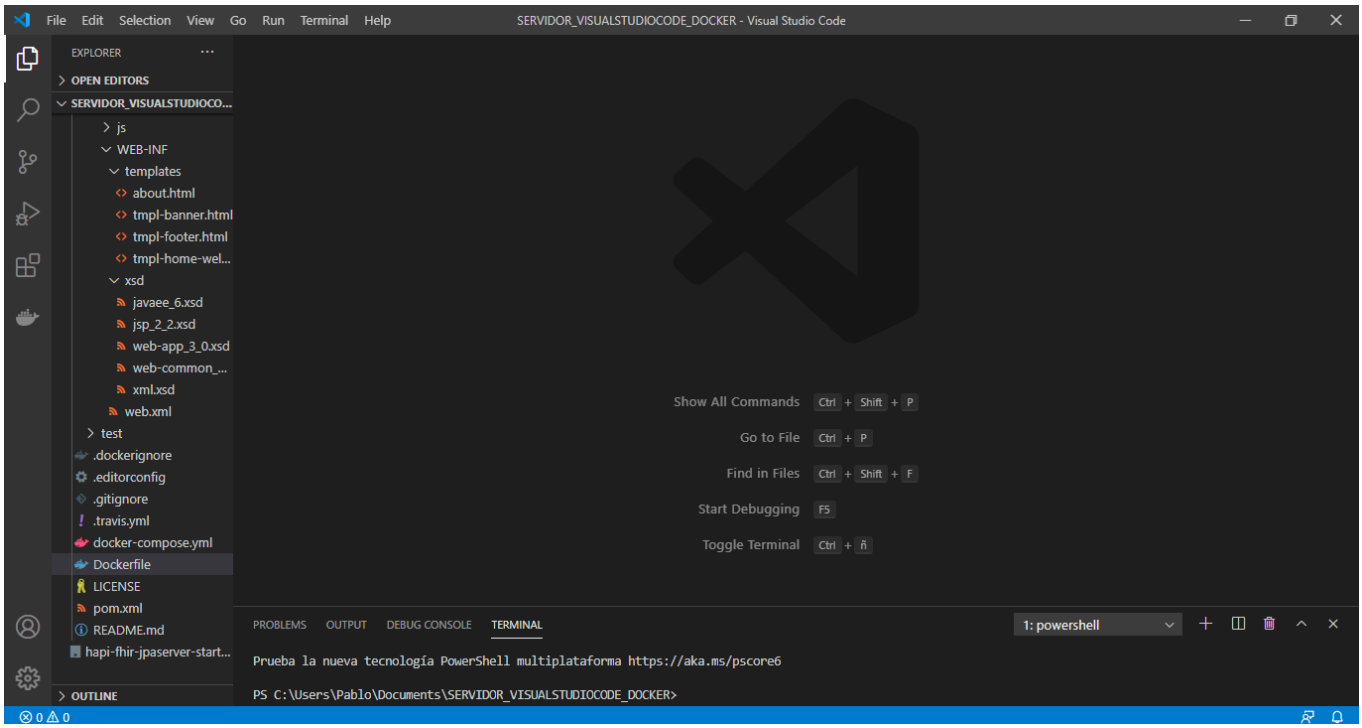


Figura 4.4: Interfaz Visual Studio Code

4.3 Insomnia

Insomnia REST Client permite gestionar las peticiones REST de una manera sencilla proporcionando las siguientes características básicas:

- Validación JSON

- Resaltado de sintaxis (JSON, JavaScript, XML, CSS...)

- Ayudante de autenticación básica HTTP

- Conmutación rápida de solicitudes

- Codificación y descodificación URL

- Codificación y descodificación base 64

Permite realizar peticiones GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS y métodos personalizados contra un servidor o servicio externo o interno.

En este proyecto, se ha empleado para realizar peticiones al servidor HAPI FHIR a modo de pruebas para verificar la correcta comunicación con este y para comprobar que las acciones realizadas en Front-end, tenían su correspondiente efecto en la Base de Datos.

Figura 4.5: Insomnia

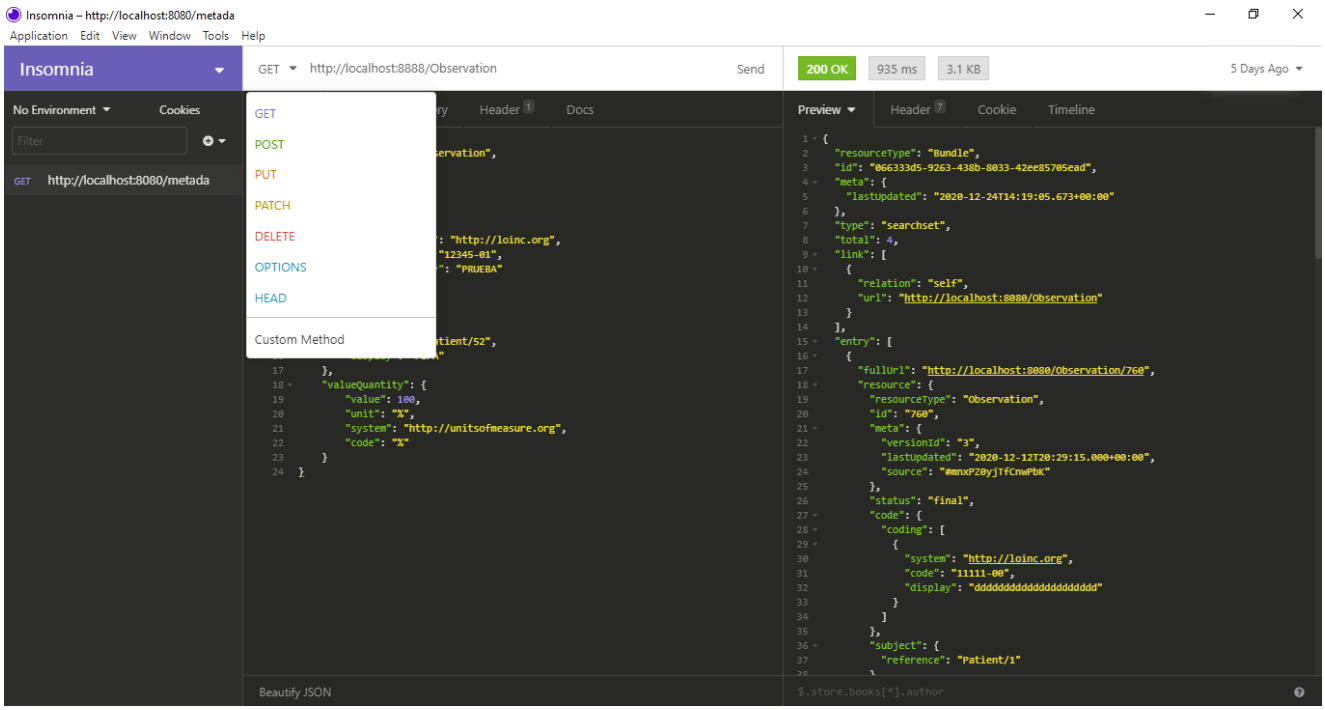


Figura 4.6: Interfaz Insomnia

4.4 Docker Desktop

Docker Desktop es una aplicación para Windows y MacOs para la creación y uso compartido de aplicaciones en contenedores.

Figura 4.7: Docker

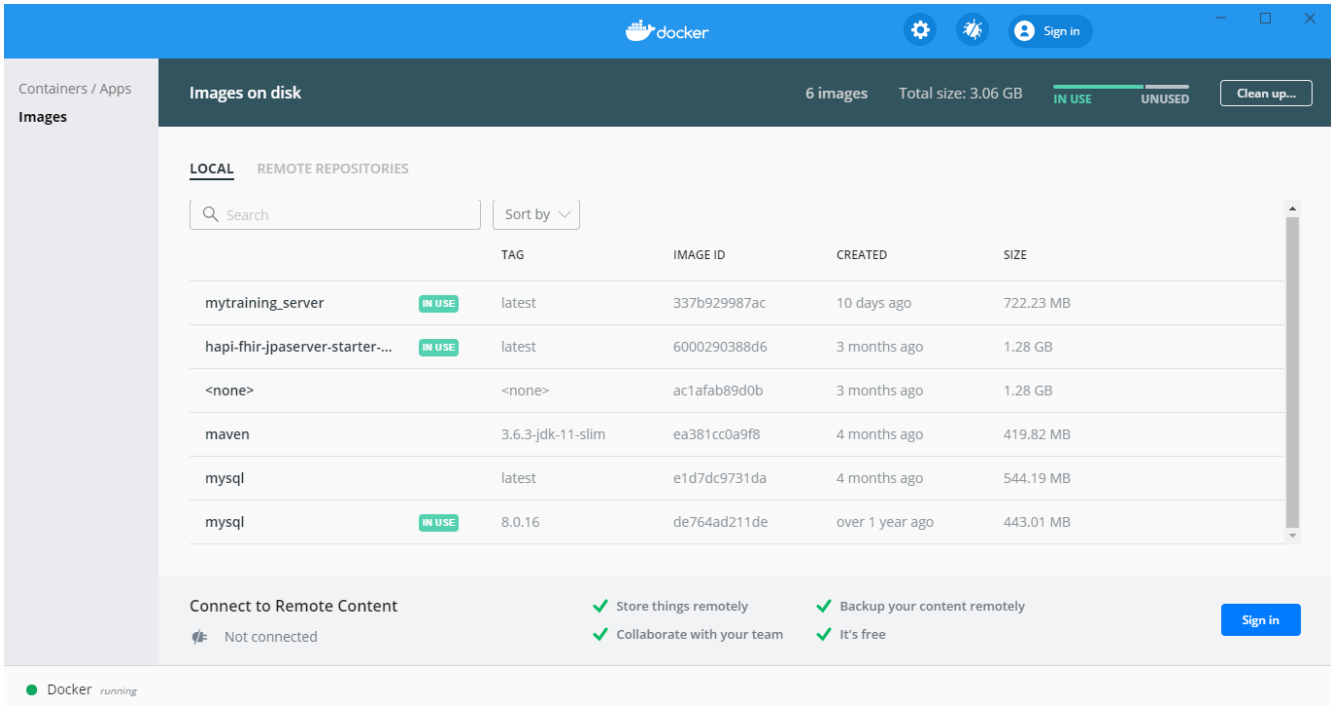


Figura 4.8: Docker Desktop

4.5 XAMPP

XAMPP es un paquete de instalación independiente de plataforma, software libre, que consiste en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl.

El nombre proviene del acrónimo de X (Multiplataforma), Apache, MariaDB, PHP, Perl. Desde la versión "5.6.15", XAMPP cambió la base de datos de MySQL a MaríaDB. El cual es un fork de MySQL con licencia GPL.

El programa se distribuye bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y Mac OS X.

XAMPP se diseñó para su uso como una herramienta de desarrollo, para permitir a los diseñadores de sitios webs y programadores testear su trabajo en sus propios ordenadores cuando no tienen ningún acceso a Internet. En la práctica, sin embargo, XAMPP se utiliza actualmente como servidor de sitios web, ya que, con algunas modificaciones, es generalmente lo suficientemente seguro para serlo. Con el paquete se incluye una herramienta especial para proteger fácilmente las partes más importantes en una página.

Figura 4.9: Logo de XAMPP.

phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 72 idiomas. Se encuentra disponible bajo la licencia GPL Versión 2.

Figura 4.10: Logo de phpMyAdmin.

4.6 Wireshark

Wireshark, antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones.

Añade una interfaz gráfica y muchas opciones de organización y filtrado de información. Así, permite ver todo el tráfico que pasa a través de una red (usualmente una red Ethernet, aunque es compatible con algunas otras) estableciendo la configuración en modo promiscuo.

Permite examinar datos de una red viva o de un archivo de captura salvado en disco. Se puede analizar la información capturada, a través de los detalles y sumarios por cada paquete. Wireshark incluye un completo lenguaje para filtrar lo que queremos ver y la habilidad de mostrar el flujo reconstruido de una sesión de TCP.

Wireshark es software libre, y se ejecuta sobre la mayoría de sistemas operativos Unix y compatibles, incluyendo Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Android, y Mac OS X, así como en Microsoft Windows.

En este proyecto se ha empleado para capturar el tráfico de la interfaz 'loopback' para observar que las peticiones al servidor HAPI FHIR se realizaban correctamente.

Figura 4.11: Wireshark

5. Diseño del servicio web

5.1 Clases

A continuación, se explican brevemente las distintas clases java que hacen que la aplicación web funcione:

- **UriConstants:** En esta clase definimos como variables estáticas las URIs que se emplean para solicitar al servidor a través de AJAX los recursos necesarios.

- **SQLConstants:** En esta clase definimos como variables estáticas las sentencias SQL que ejecuta el servidor en la Base de Datos a través de la implementación del modelo DAO.

- **UsuarioController, FHIRController, RutinaController, EjercicioController y VideoController:** Estas son las clases "Controlador". Dichas clases poseen una serie de funciones que son llamadas a través de las peticiones realizadas a las URIs y del método utilizado (GET, POST, PUT, DELETE).

- **UsuarioDaoImpl, ObservacionDaoImpl, RutinaDaoImpl, EjercicioDaoImpl y VideoDaoImpl:** Estas clases poseen funciones que son llamadas por la clase "Controlador" correspondiente. Dichas funciones se encargan de ejecutar el código asociado para obtener los datos deseados y mapearlos en un objeto java el cual es devuelto al controlador. Estas clases son implementaciones de las siguientes interfaces: **UsuarioDao, ObservacionDao, RutinaDao, EjercicioDao y VideoDao.**

- **Usuario, Observacion, UsuarioLogin, UsuarioRol, Rutina, Ejercicio y Video:** Estas clases definen los objetos que maneja el servicio Web, y en ellas se define el constructor, los atributos y métodos correspondientes a cada objeto.

- **UsuarioRowMapper,UsuarioRowLoginMapper,UsuarioRolRowMapper, RutinaRowMapper, EjercicioRowMapper y VideoRowMapper:** Estas clases permiten la realización de un mapeo entre los datos obtenidos de la BBDD a través de un objeto DAO a un objeto Java determinado.

- **SecurityConfiguration,RESTAuthenticattionSuccessHandler, RESTAuthenticationFailureHandler, RESTAuthenticationEmtryPoint:** Estas clases son las que configuran el módulo de Spring Security.

- **MyTrainingApplication:** Clase principal del servicio web.

5.2 Diagrama de clases

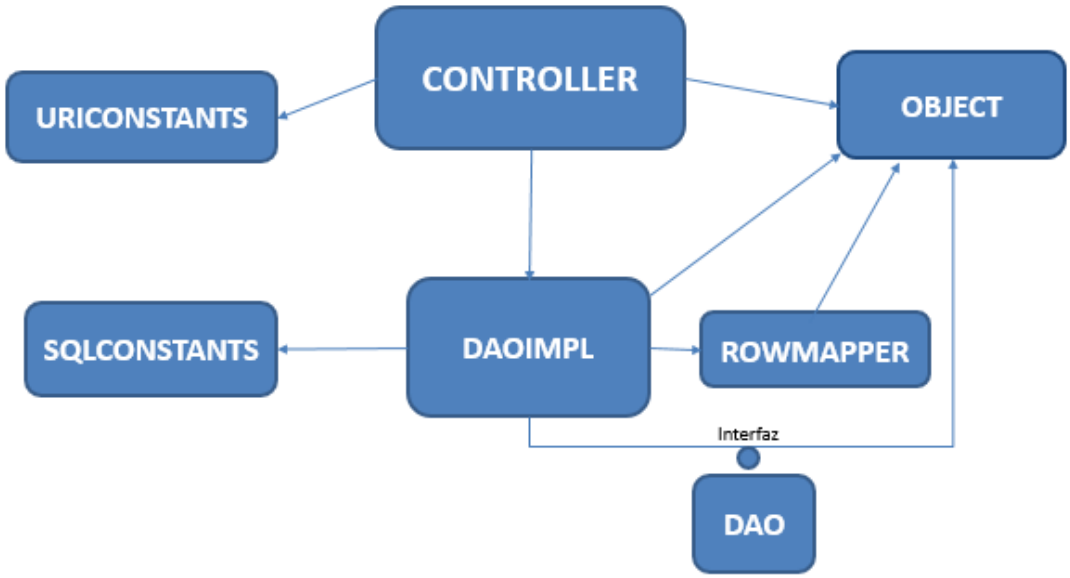


Figura 5.1: Diagrama de clases

5.3 Diagramas de secuencia

Creación nuevo recurso

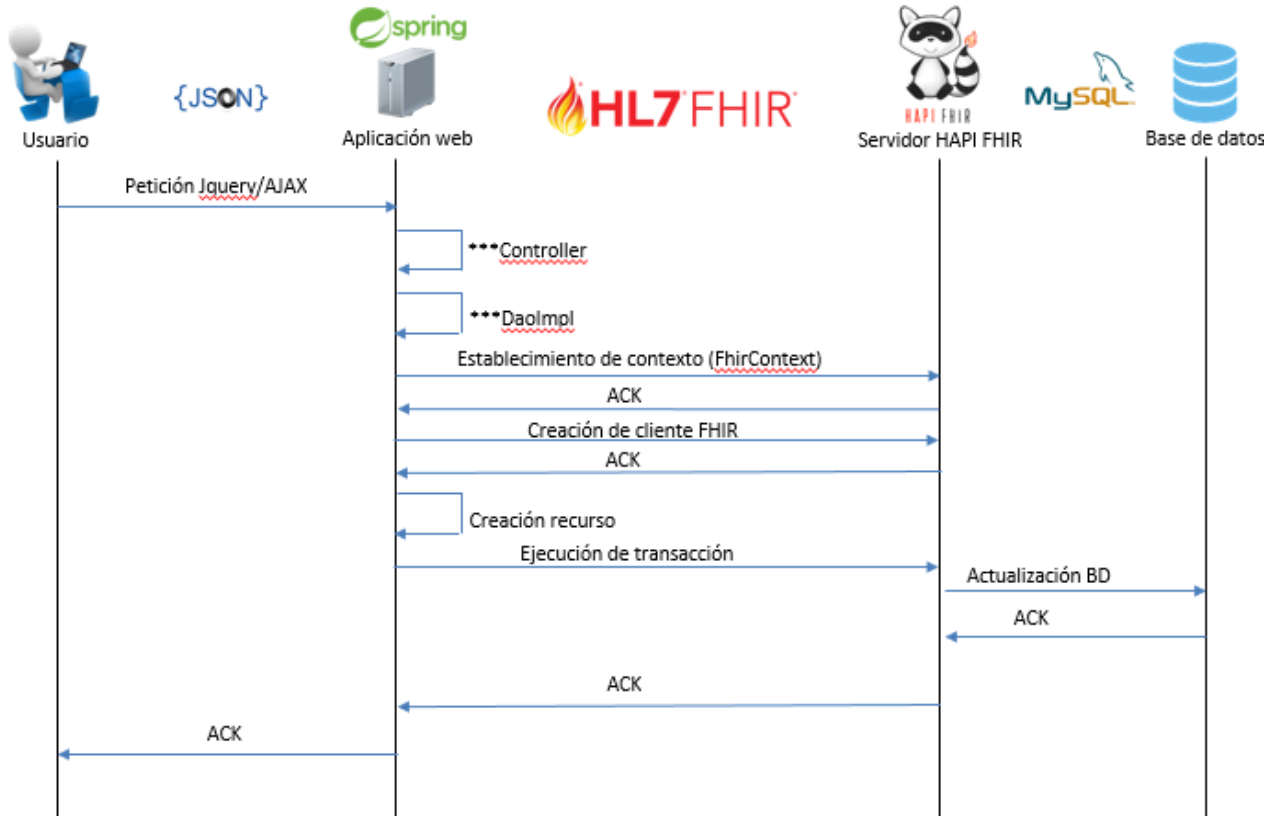


Figura 5.2: Diagrama crear recurso

Modificación recurso

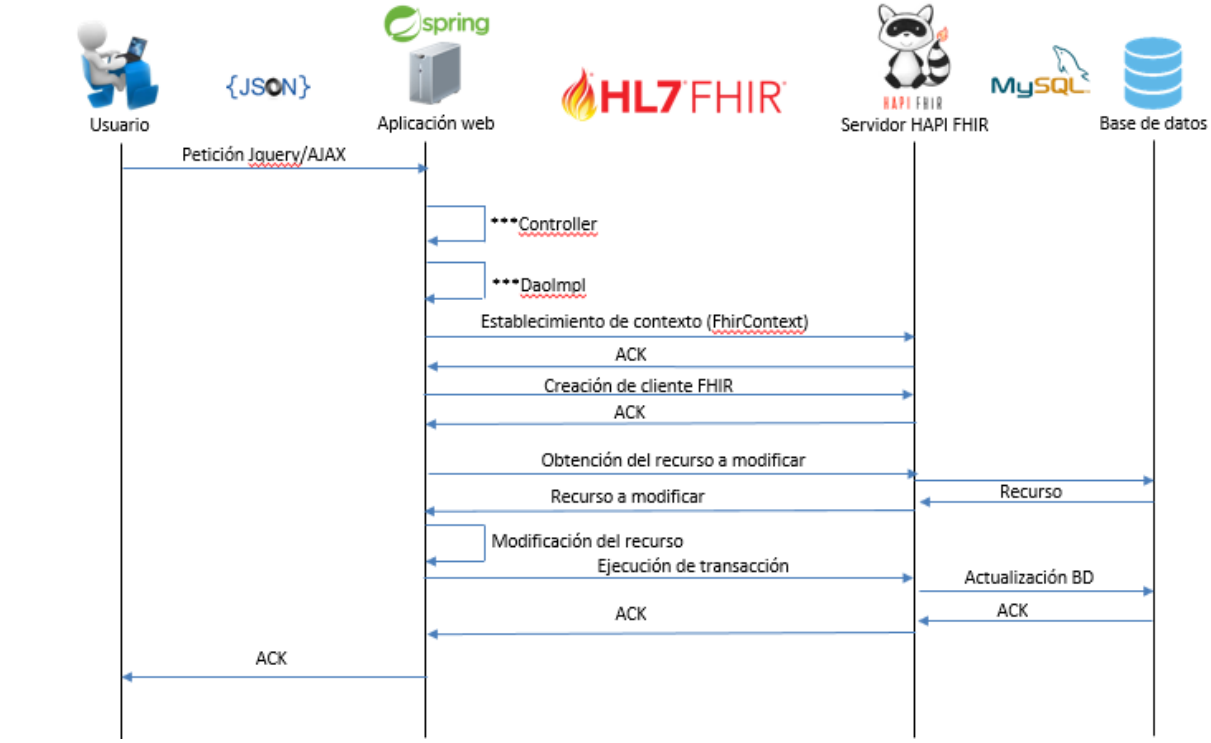


Figura 5.3: Diagrama modificar recurso

Eliminación recurso

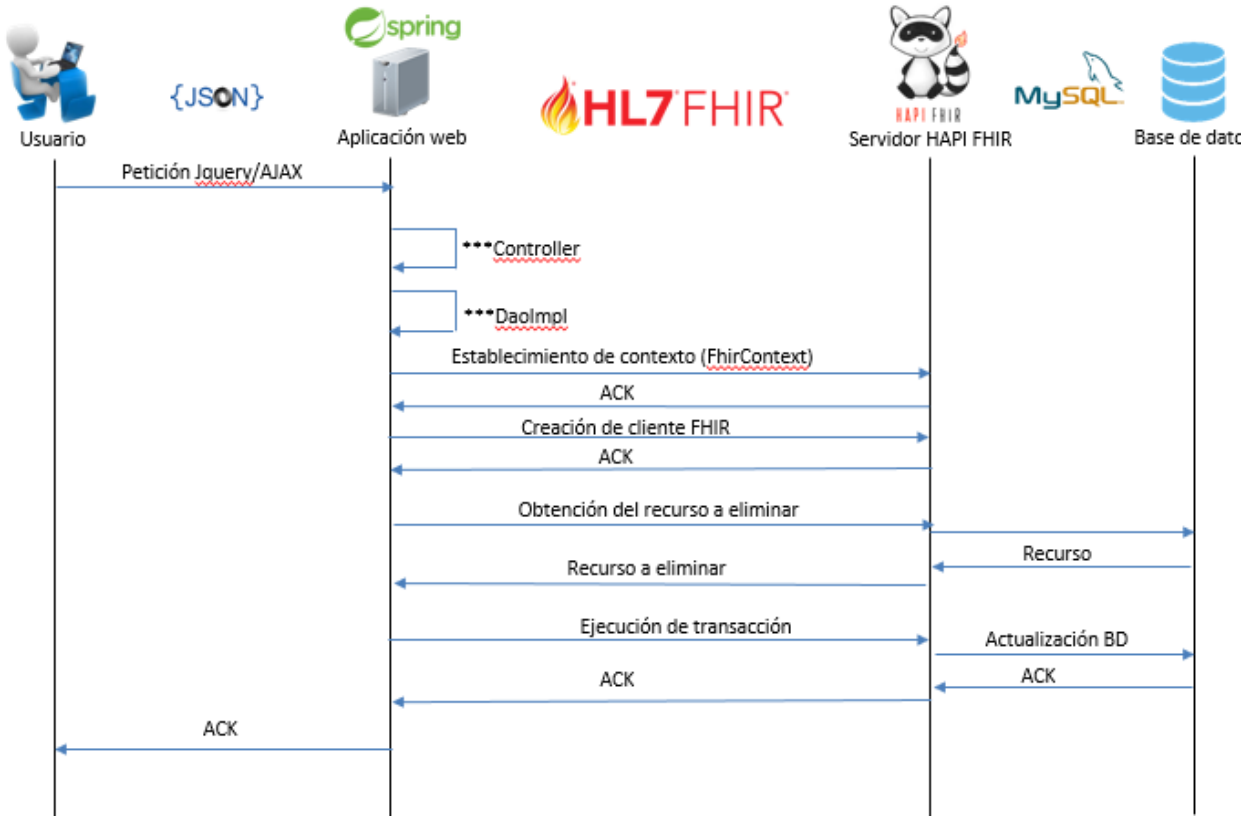


Figura 5.4: Diagrama eliminar recurso

5.4 Códigos ejemplo de implementación de FHIR en la aplicación web

En este apartado, se mostrarán breves extractos del código que ha sido necesario implementar en la aplicación web para establecer la comunicación con el servidor HAPI FHIR y reflejar las acciones realizadas sobre la app web en la base de datos FHIR.

Creación de contexto y cliente para la comunicación con el servidor HAPI FHIR

Dicho servidor se encuentra en la dirección "http://localhost:8888". Esta dirección es configurable, como veremos en el apartado "5.6 Configuración del docker".

```
//Create context
FhirContext ctx = FhirContext.forR4();
// Create a client
IGenericClient client = ctx.newRestfulGenericClient("http://localhost:8888");
```

Creación de un nuevo recurso y asignación de atributos

```
 // Create a Observation
Observation newObservation = new Observation();

newObservation.setStatus(Observation.ObservationStatus.FINAL);
newObservation.setValue(new StringType("This is a value"));

newObservation
    .getCode()
        .addCoding()
            .setSystem("http://loinc.org")
            .setCode(obsCode)
            .setDisplay(obsDescripcion);
newObservation.setValue(
    new Quantity()
        .setValue(obsMedida)
        .setUnit(obsUnidad)
        .setSystem("http://unitsofmeasure.org"));

newObservation.addBasedOn(new Reference("CarePlan/"+IDRUTINA));

// The observation refers to the patient using the ID
newObservation.setSubject(new Reference("Patient/" + patID));
// The observation refers to the especialista using the ID
newObservation.addPerformer(new Reference("Practitioner/" + creadorID));
```

Transacción para la creación de un recurso en el servidor HAPI FHIR

```
 // Create the resource on the server
 MethodOutcome outcome = client
      .create()
      .resource(newObservation)
      .execute();
```

Obtención de un recurso existente en base a un criterio o condición

```
Bundle rut = client.search()
        .forResource(CarePlan.class)
        .where(CarePlan.IDENTIFIER.exactly().identifier(rutinaRef))
        .returnBundle(Bundle.class)
        .execute();
```

Obtención de un recurso existente en base a su identificador en el servidor HAPI FHIR

```
Patient patient = client.read().resource(Patient.class).withId(paciente).execute();
```

Obtención de atributos de un recurso obtenido

```
Observation obs = client.read().resource(Observation.class).withId(obsId).execute();
String code = obs.getCode().getCoding().get(0).getCode();
String descripcion = obs.getCode().getCoding().get(0).getDisplay();
String unidad = obs.getValueQuantity().getUnit();
String medida = obs.getValueQuantity().getValue().toString();
String paciente = obs.getSubject().getReference();
```

Actualización de los atributos de un recurso obtenido

```
Observation observacion = client.read().resource(Observation.class).withId(obs_id).execute();

Coding code = new Coding();
code.setCode(obsCode);
code.setDisplay(obsDescripcion);

observacion.getValueQuantity().setUnit(obsUnidad);
observacion.getValueQuantity().setValue(obsMedida);
observacion.getCode().getCoding().set(0, code);
```

Transacción para la actualización de un recurso en el servidor HAPI FHIR

```
MethodOutcome outcome = client
        .update()
        .resource(observacion)
        .execute();
```

Eliminación de un recurso del servidor HAPI FHIR

```
IBaseOperationOutcome outcome = client
    .delete()
    .resourceById(new IdType("Observation", obsId))
    .execute();
```

5.5 Configuración del servidor HAPI FHIR

En este apartado, hablaremos del fichero "hapi.properties", que contiene la configuración del servidor HAPI FHIR. Este fichero es un poco extenso, por lo que se mencionarán las líneas más destacables. Pero primero, en la figura 5.5, se muestra la estructura de ficheros del servidor HAPI FHIR que mencionaremos, tanto en este apartado, como en el 5.6.

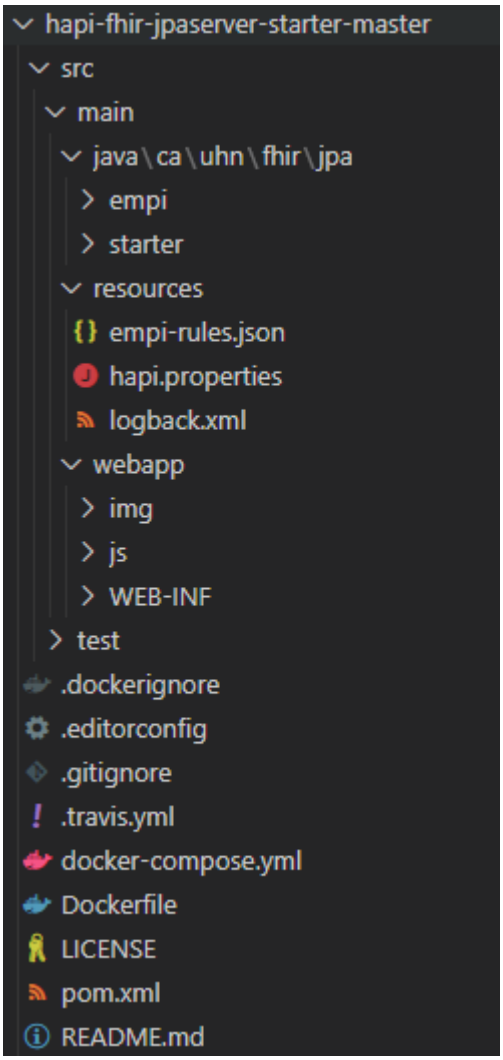


Figura 5.5: Estructura archivos HAPI FHIR

Hapi.properties

- Atributo "fhir_version": Permite especificar la versión de FHIR soportada por el servdiro HAPI FHIR. La versión R4 es la última existente.
- Atributo "server address": Permite especificar la dirección IP del servidor.

hapi-fhir-jpaserver-starter-master > src > main > resources > hapi.properties

```
# Adjust this to set the version of FHIR supported by this server. See
# FhirVersionEnum for a list of available constants. Example values include
# DSTU2, DSTU3, R4.
fhir_version=R4

# This is the address that the FHIR server will report as its own address.
# If this server will be deployed (for example) to an internet accessible
# server, put the DNS name of that server here.
#
# Note that this is also the address that the hapi-fhir-testpage-overlay
# (the web UI similar to the one at http://hapi.fhir.org) will use to
# connect internally to the FHIR server, so this also needs to be a name
# accessible from the server itself.
server_address=http://localhost:8080/
```

- Atributo "default_encoding": Permite especificar la codificación por defecto de los datos.

```
default_encoding=JSON
```

- Atributo "datasource.driver" y "datasource.url": Permiten especificar el tipo de driver para la base de datos FHIR (en este caso Mysql) y la URL a la que comunicarse para acceder a esta.

```
datasource.driver=com.mysql.jdbc.Driver
```

```
datasource.url=jdbc:mysql://host.docker.internal:3306/rutina_app
?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
```

- Atributo "datasource.username" y "datasource.password": Permite especificar la dupla usuario/contaseña de acceso a la base de datos.

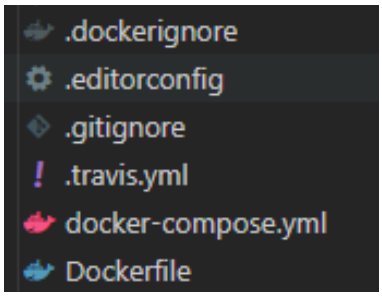
```
datasource.username=root
datasource.password=
```

- Atributo hibernate.dialect: Permite especificar el dialecto para la comunicación entre el servidor HAPI FHIR y la base de datos.

```
hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
```

5.6 Configuración del docker

A continuación, hablaremos de los archivos de configuración necesarios para la puesta en marcha del Docker.



Dockerfile

Docker puede crear imágenes automáticamente leyendo las instrucciones de este archivo. Es un documento de texto que contiene los comandos a llamar en la línea de comandos para ensamblar una imagen.

```
hapi-fhir-jpaserver-starter-master > Dockerfile > ...
1   FROM maven:3.6.3-jdk-11-slim
2   WORKDIR /tmp/hapi-fhir-jpaserver-starter
3
4   COPY pom.xml .
5   RUN mvn dependency:go-offline
6
7   COPY src/ /tmp/hapi-fhir-jpaserver-starter/src/
8   RUN mvn clean install -DskipTests
9
10  EXPOSE 8080
11
12  CMD ["mvn", "jetty:run"]
13
```

- La instrucción FROM inicializa una nueva etapa de compilación y establece la *imagen base* para instrucciones posteriores.
- La instrucción WORKDIR establece el directorio de trabajo
- La instrucción EXPOSE informa a Docker de que el contenedor escucha en los puertos de red especificados en tiempo de ejecución.
- La instrucción RUN ejecutará cualquier comando en una nueva capa encima de la imagen actual y confirmará los resultados.
- La instrucción COPY copia nuevos archivos o directorios al sistema de archivos del contenedor en la ruta de especificada.
- El propósito principal del comando CMD es proporcionar valores predeterminados para un contenedor en ejecución.

Docker-compose

Compose es una herramienta para definir y ejecutar aplicaciones Docker de varios contenedores. Con Compose, se utiliza un archivo YAML para configurar los servicios de la aplicación.

Destacamos la entrada "ports", que mapea el puerto definido en el apartado "Configuración del servidor HAPI FHIR", el 8080, al 8888.

hapi-fhir-jpaserver-starter-master > docker-compose.yml

```yaml
version: "3"
services:
  server:
    build: .
    ports:
      - "8888:8080"
    volumes:
      - ./src:/tmp/hapi-fhir-jpaserver-starter/src/
```

.dockerignore

Si este archivo existe, la CLI modifica el contexto para excluir archivos y directorios que coinciden con los patrones en él. Esto ayuda a evitar el envío innecesario de archivos y directorios grandes o confidenciales al "demonio" Docker.

5.7 Spring Security

En nuestro proyecto hemos utilizado este módulo de Spring como encargado del ámbito de la seguridad en torno a la autenticación de usuario, el control de acceso a las diversas páginas de la aplicación basándose en el rol del usuario y la obtención de recursos de la aplicación.

En la figura 5.1, se explica de manera visual cómo actúa Spring Security a la hora de enfrentarse al proceso de autenticación de credenciales antes de otorgar los recursos que la petición realizada por el usuario pide.

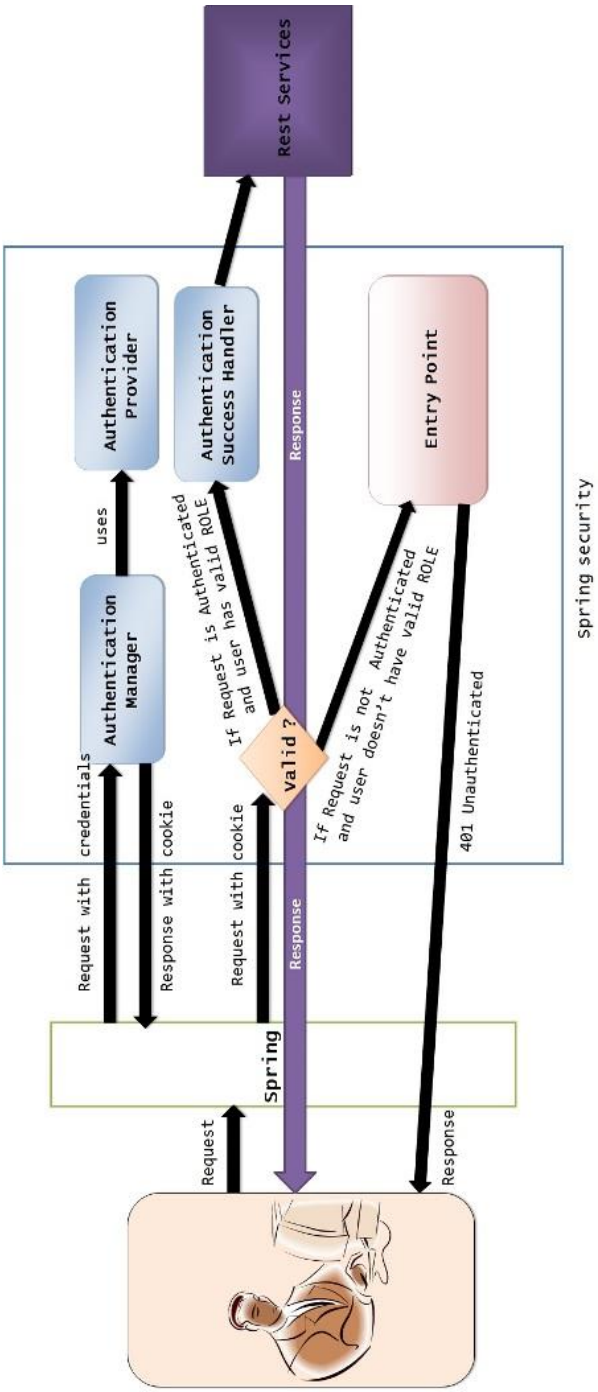


Figura 5.6: Diagrama de acción de Spring Security.

En el diagrama puede verse como a la hora de comprobar credenciales sucede lo siguiente:

1. El usuario envía una petición con sus credenciales como dato y Spring la captura.
2. La petición es manejada por el "Authentication Manager" o "Gestor de autenticación" que decide cuando un usuario es válido. Este gestor está relacionado con el "Authentication Provider" que es quién define la forma en la que un usuario se ha de validar. Puede ser contra una base de datos, puede ser contra un Ldap, puede ser contra un fichero o puede personalizarse.
3. El "Authentication Manager" devuelve una cookie respuesta a Spring.

4. Si la Cookie es válida, significa que la autenticación de las credenciales ha resultado con éxito por lo que el "Authentication Success Handler" maneja la petición y, finalmente, el servicio REST otorga el recurso. Si la Cookie no es válida devolverá un error de autenticación.

Spring Security tiene muchas funcionalidades y configuraciones posibles. A continuación, se procede a mostrar los métodos de la clase principal de configuración de Spring Security:

SecurityConfiguration.java

En este método se obtiene, mediante las sentencias SQL "GET_USER_AUTHENTICATION" y "GET_USER_AUTHORITY", las credenciales del usuario de la base de datos para que Spring Security las compare con las que el usuario ha introducido y su rol para establecer los permisos de acceso.

```
/* AUTHENTICATION */
public static final String GET_USER_AUTHENTICATION =
    "SELECT Email,Password,Enabled FROM USUARIOS WHERE Email=?";

public static final String GET_USER_AUTHORITY =
    "SELECT USUARIOS_Email,Role FROM USUARIOS_ROLES WHERE USUARIOS_email=?";
```

Figura 5.7: SQL Authentication

```
@Override
protected void configure(AuthenticationManagerBuilder builder) throws Exception {

    builder.jdbcAuthentication()
    .dataSource(dataSource)
    .usersByUsernameQuery(SqlConstants.GET_USER_AUTHENTICATION)
    .authoritiesByUsernameQuery(SqlConstants.GET_USER_AUTHORITY);
```

Figura 5.8: Authentication Method

En este método se configuran los filtros a las distintas páginas de la aplicación y las acciones en caso de acceso permitido o no.

Los dos tipos de roles para usuario son "ROLE_ADMIN" o "ADMIN", "ROLE_USER" o "USER" y "ROLE_ESPECIALISTA" o "ESPECIALISTA".

Puede establecer los filtros para que solo accedan a 'x' URL un tipo específico de rol, o todos los autenticados, o todos en general, etc.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
        // Las reglas van de más específicas a menos específicas
        // antMatchers() indica el patrón sobre el que se aplicará la regla
        http.authorizeRequests().antMatchers("/").permitAll();
        http.authorizeRequests().antMatchers("/RutinaRegister/**").authenticated();
        http.authorizeRequests().antMatchers("/login.html","/SMSLogin.html").permitAll();
        http.authorizeRequests().antMatchers("/WelcomeRutinaApp.html").authenticated();
        http.authorizeRequests().antMatchers("/usuario/*").hasRole("USER");
        http.authorizeRequests().antMatchers("/especialista/*").hasRole("ESPECIALISTA");
        http.authorizeRequests().antMatchers("/administrador/*").hasRole("ADMIN");
        http.authorizeRequests().antMatchers("/*.html").authenticated();

        http.exceptionHandling().accessDeniedPage("/error.html");

        http.authorizeRequests().antMatchers("/Rutina_app/**").authenticated();

        http.exceptionHandling().authenticationEntryPoint(authenticationEntryPoint);


        http.formLogin().successHandler(authenticationSuccessHandler);
        http.formLogin().failureHandler(authenticationFailureHandler);
        http.logout().logoutSuccessUrl("/login.html");

        http.csrf().disable();
    // CSRF tokens handling
    http.addFilterAfter(new CsrfTokenResponseHeaderBindingFilter(), CsrfFilter.class);
}
```

Figura 5.9: Filter Method

Y finalmente, mostramos los métodos de las clases RESTAuthenticattionSuccessHandler, RESTAuthenticationFailureHandler y RESTAuthenticationEntryPoint, encargados de manejar el éxito, fallo y avisos de autenticación respectivamente.

```
@Component
public class RESTAuthenticationSuccessHandler extends SimpleUrlAuthenticationSuccessHandler {

    @Override
    public void onAuthenticationSuccess(HttpServletRequest request, HttpServletResponse response,
                                        Authentication authentication) throws IOException, ServletException {

        clearAuthenticationAttributes(request);
    }
}
```

Figura 5.10: RESTAuthenticattionSuccessHandler

```java
@Component
public class RESTAuthenticationFailureHandler extends
        SimpleUrlAuthenticationFailureHandler {

    @Override
    public void onAuthenticationFailure(HttpServletRequest request,
        HttpServletResponse response,
        AuthenticationException exception) throws IOException,
    ServletException {
        super.onAuthenticationFailure(request, response, exception);
    }
}
```

Figura 5.11: RESTAuthenticattionFailureHandler

```java
@Component
public class RESTAuthenticationEntryPoint implements AuthenticationEntryPoint {

    @Override
    public void commence(HttpServletRequest request,
        HttpServletResponse response,
        AuthenticationException authException)
            throws IOException, ServletException {
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED,
            "No tiene autorización para acceder a este contenido");
    }
}
```

Figura 5.12: RESTAuthenticattionEntryPoint

5.8 Base de datos

El sistema, a nivel conceptual, consta de dos Bases de Datos, una local, en la que se encuentra y almacena la información referente a rutinas, ejercicios, pacientes, especialistas y administradores

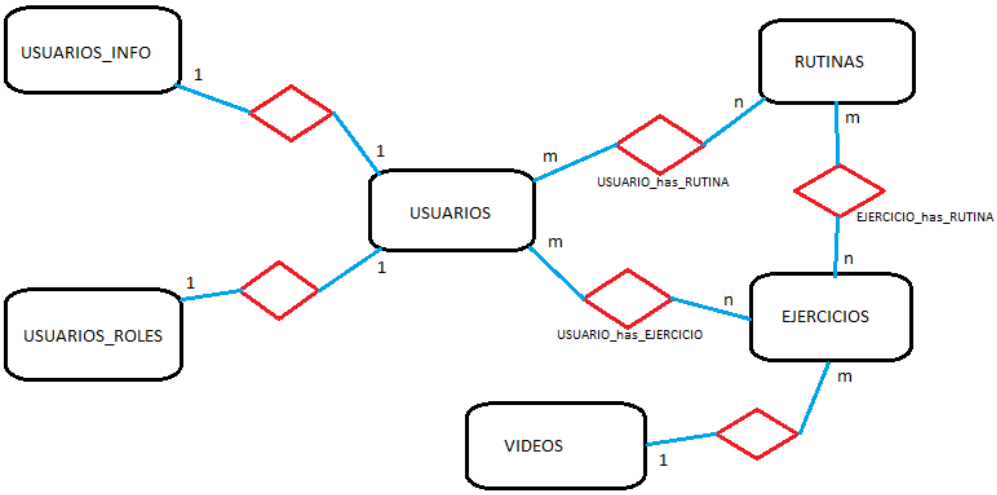


Figura 5.13: Diagrama ER Base Datos local

y una externa, que sería la asociada al servidor HAPI FHIR. En la Base de Datos FHIR, se encuentran y almacenan los mismos datos que en la local y los datos referentes a las observaciones de pacientes y especialistas con el formato propio del estándar FHIR.

La Base de Datos local se emplea para la autenticación de usuario a la hora de acceder a la app, y para la lectura de datos y la impresión de estos en la aplicación web, exceptuando las observaciones, que se leen directamente de la Base de Datos FHIR.

6 INTERFAZ DE USUARIO Y FUNCIONALIDAD

6.1 Introducción

En este apartado se va a proceder a mostrar dos de los aspectos más importantes para el usuario como son la interfaz gráfica y la funcionalidad de la aplicación. La interfaz gráfica debe ser amigable al usuario, intuitiva y fácil de comprender. Las funciones que puede realizar cada usuario difieren según el rol que posean dentro de la aplicación.

6.2 Registro de especialista

Del registro de especialistas se encarga el administrador. El administrador, tras acceder a la aplicación, dispondrá de la siguiente página para introducir los datos personales y cualificación del especialista a dar de alta.

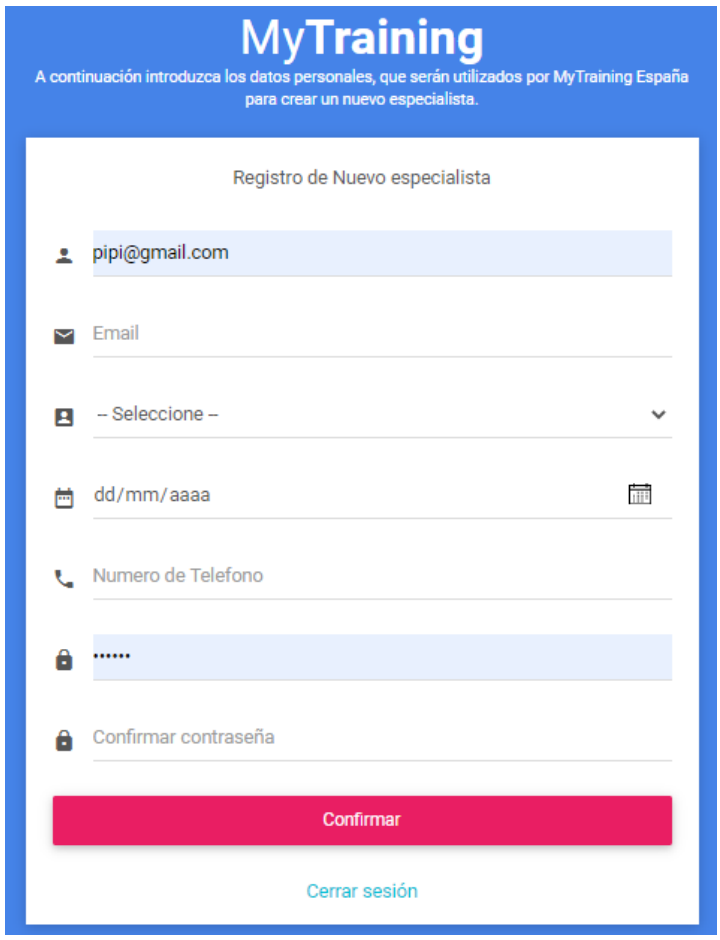


Figura 6.1: Dar de alta especialista.

6.2 Inicio de Sesión

El usuario tras acceder a la página de inicio de sesión deberá acreditarse con sus credenciales, que en este caso son su correo y su contraseña. También, desde aquí, podrá descargarse la app para móvil de MyTraining.

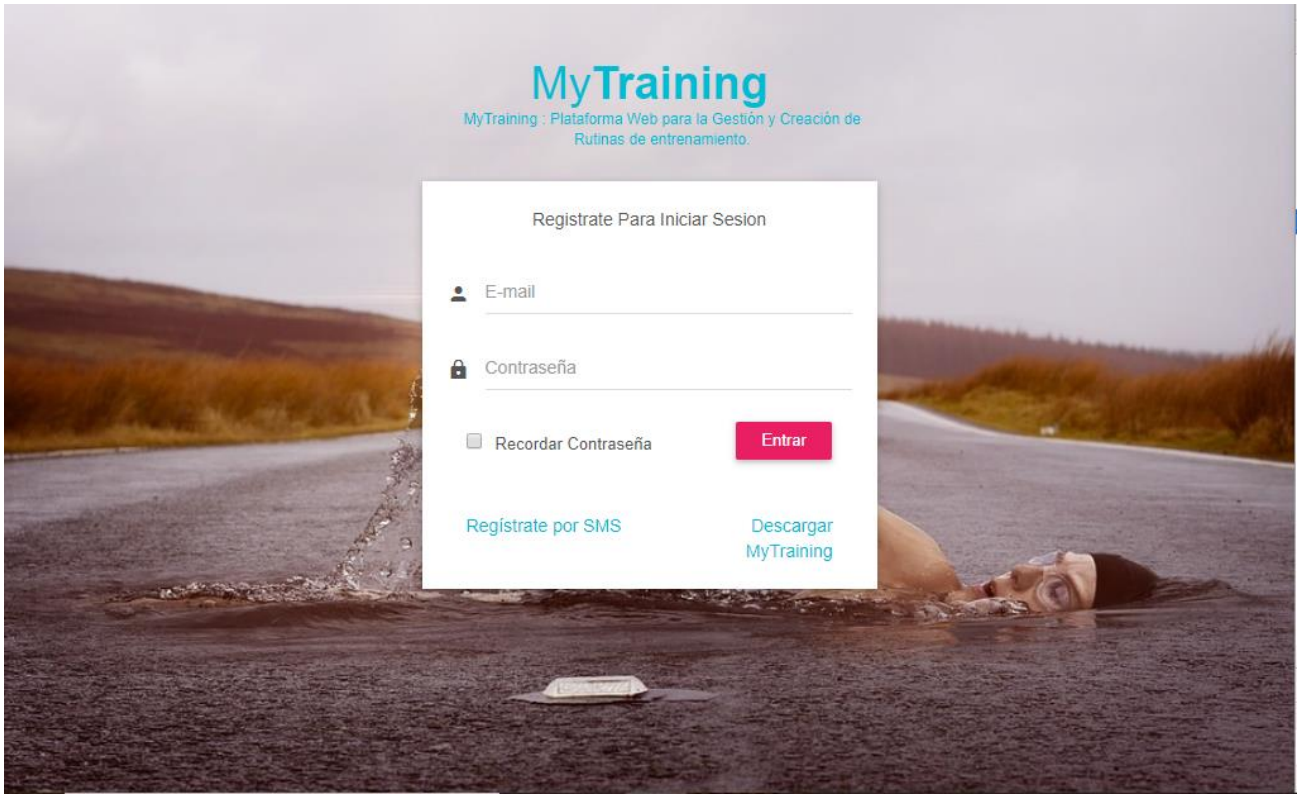


Figura 6.1: Página de Inicio de Sesión.

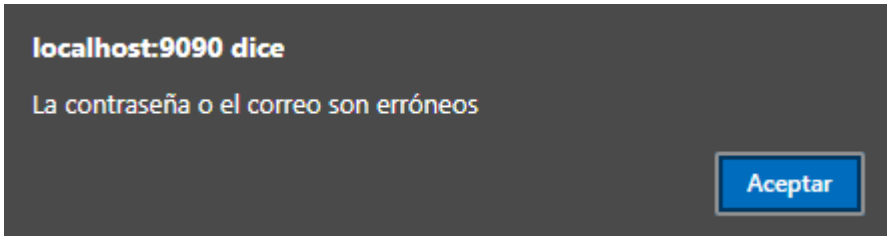


Figura 6.2: Mensaje de Aviso de Credenciales Introducidas Erróneas.

5.3 Vistas y funciones de la web

A continuación, procederemos a mostrar las diferentes vistas de la aplicación:

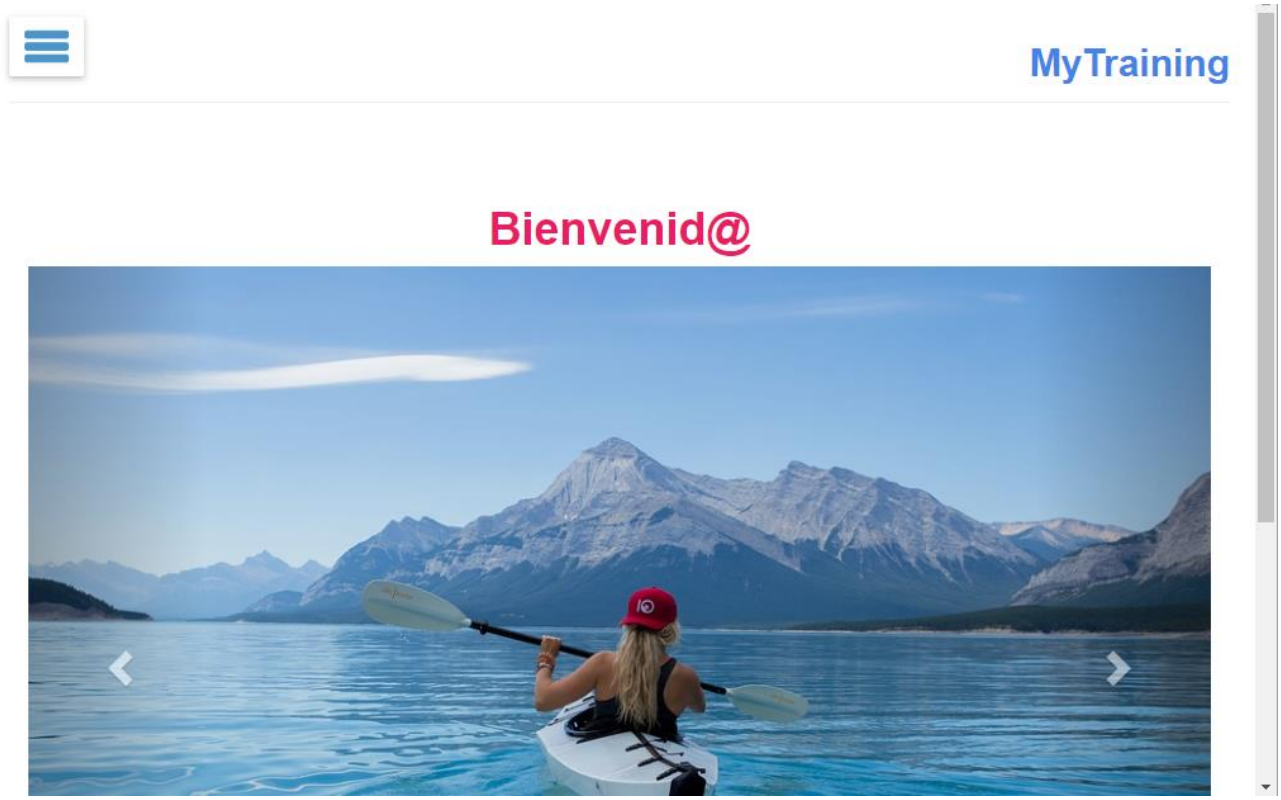


Figura 6.3: Página principal.

Este es el aspecto de la página principal para ambos usuarios. En la esquina superior izquierda hay un botón cuya función es desplegar un menú lateral que difiere según el rol del usuario. Dicho menú solo se encontrará oculto inicialmente en la página principal, en el resto de páginas podremos encontrarlo ya desplegado con oportunidad de ocultarlo.

.

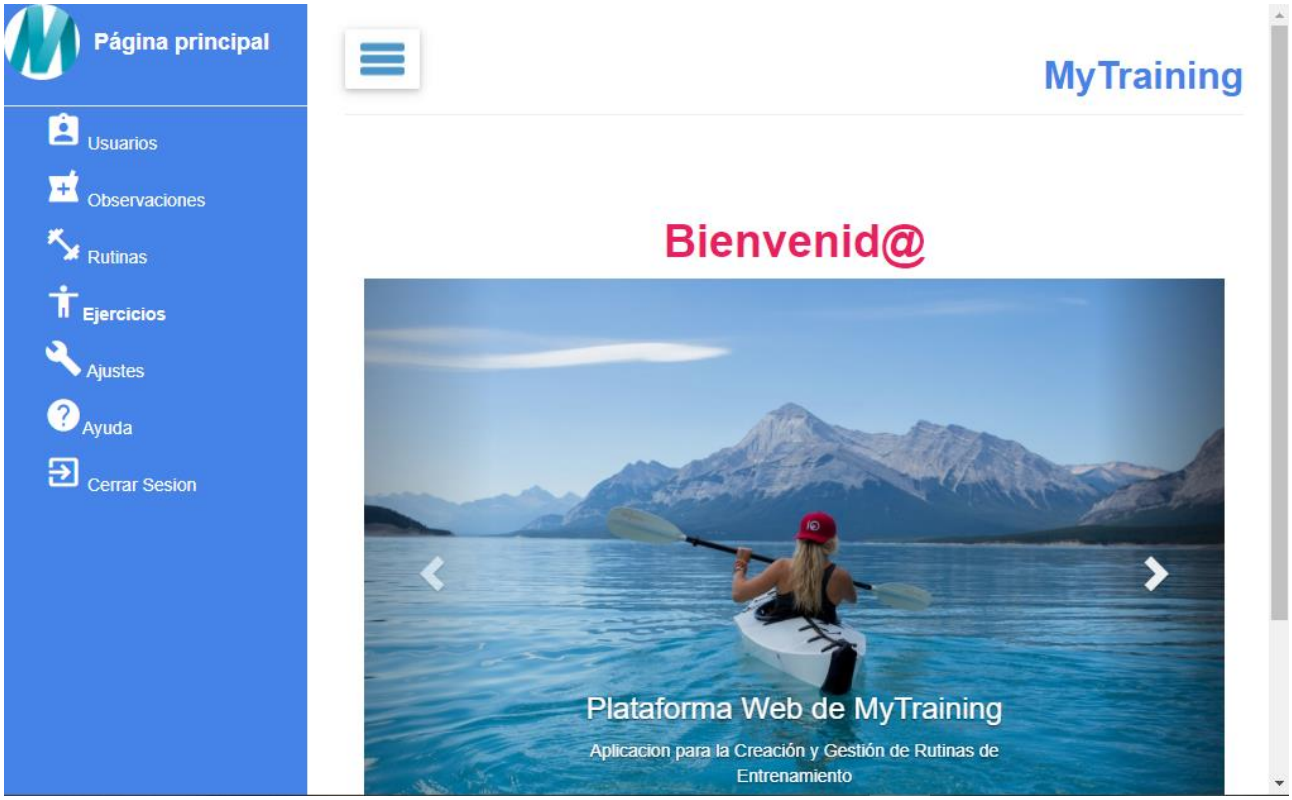


Figura 6.4: Página principal con menú lateral desplegado de Usuario Especialista.

En la imagen anterior se muestran las opciones del menú lateral de un usuario Especialista. Si pinchamos sobre "Página principal" desde cualquier pestaña seremos redirigidos a la página de Bienvenida que se muestra.

Procedemos a ver cada una de las pestañas.

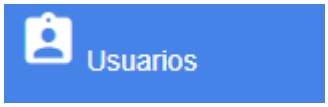


Figura 6.5: Pestaña 'Usuarios'.

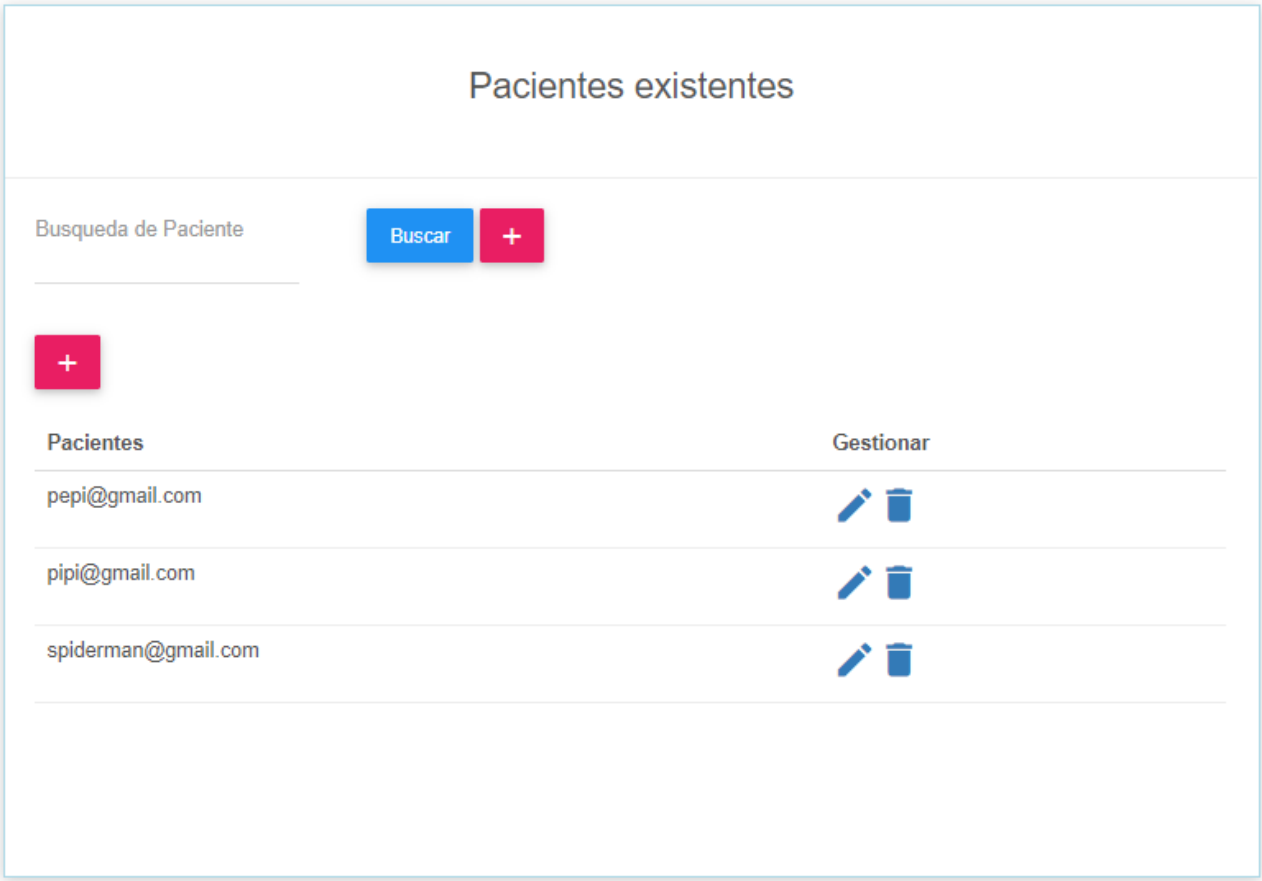


Figura 6.6: Página 'Usuarios'

En esta página el usuario Especialista puede:

- Ver un listado de los pacientes existentes y filtrarlos mediante búsqueda.
- Gestionar dichos pacientes (Modificar sus datos o darlos de baja)
- Dar de alta a un nuevo paciente.

Para dar de alta a un paciente, solo hay que pulsar el botón 

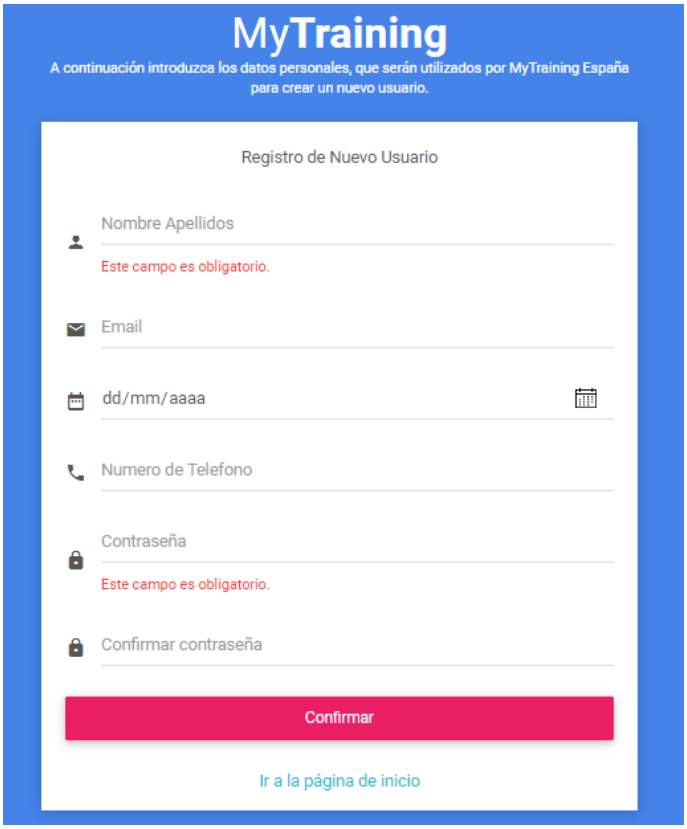


Figura 6.7: Dar de alta paciente

Para modificar los datos de algún paciente, debe clicarse sobre 

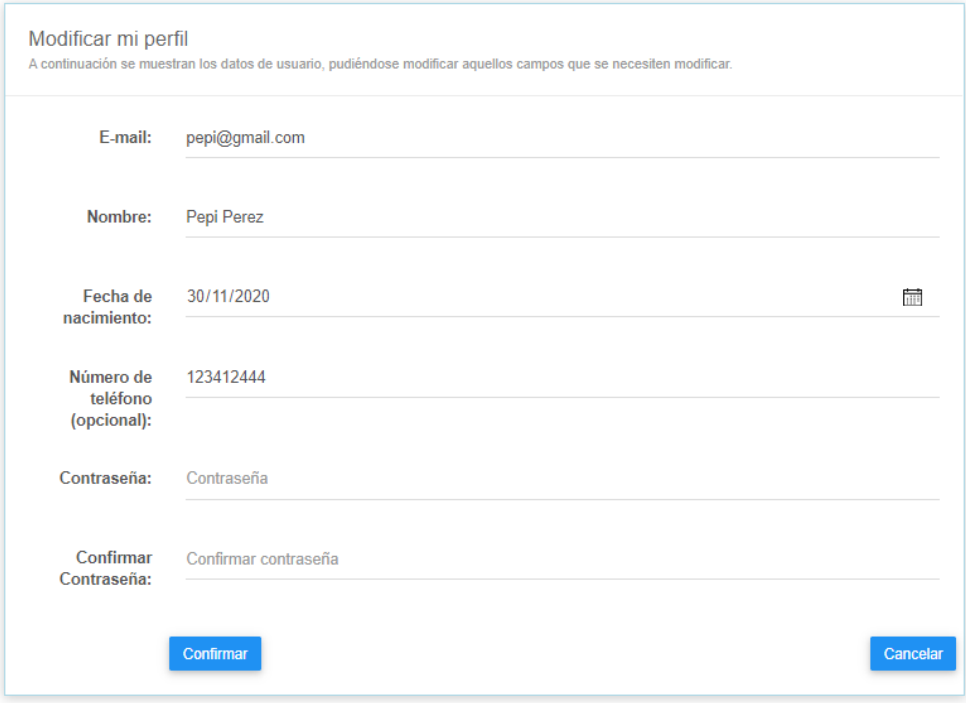


Figura 6.8: Modificar datos de paciente

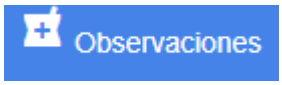
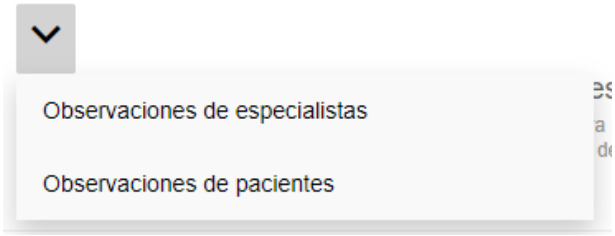


Figura 6.9: Pestaña Observaciones

En esta pestaña, el especialista podrá listar todas las observaciones del resto de especialistas y las propias, pudiendo filtrar estas solo por paciente o por paciente y por código de clasificación. Cada especialista podrá modificar o eliminar solo sus propias observaciones.

El especialista también podrá listar las observaciones que hayan generado los pacientes, seleccionándolo en el desplegable de la esquina superior izquierda.



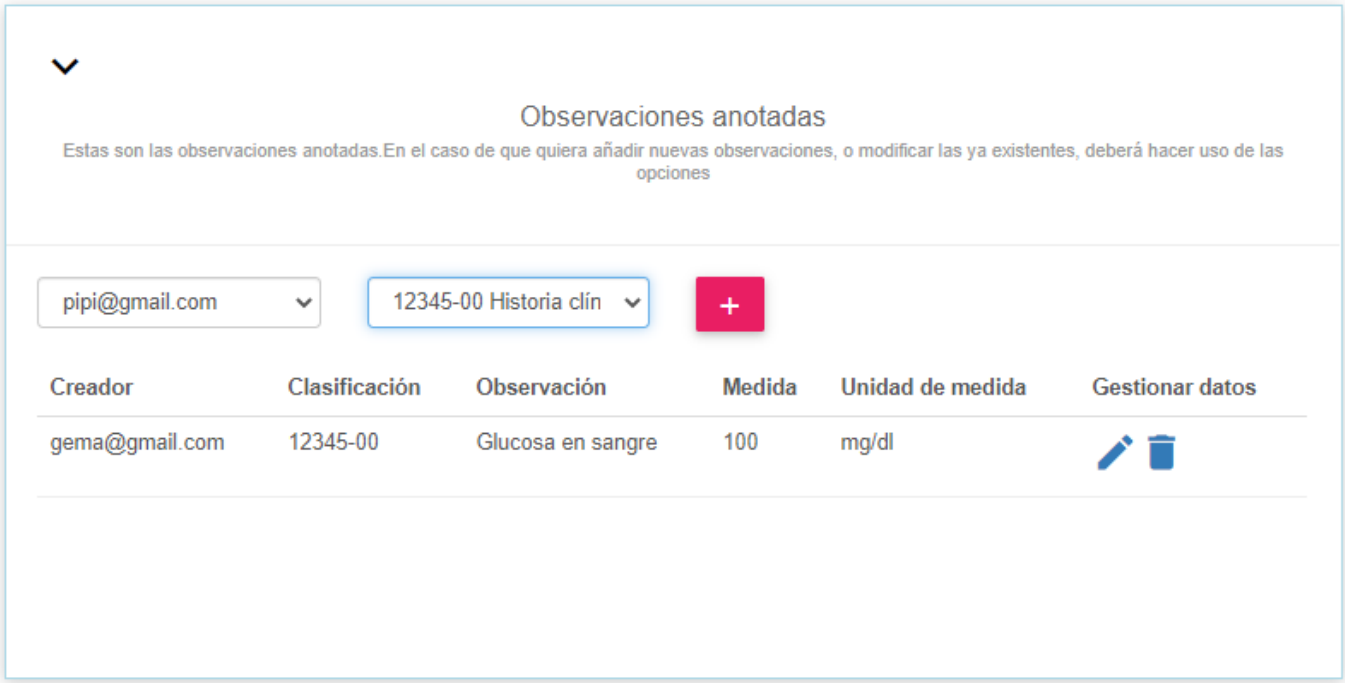


Figura 6.10: Observaciones especialistas

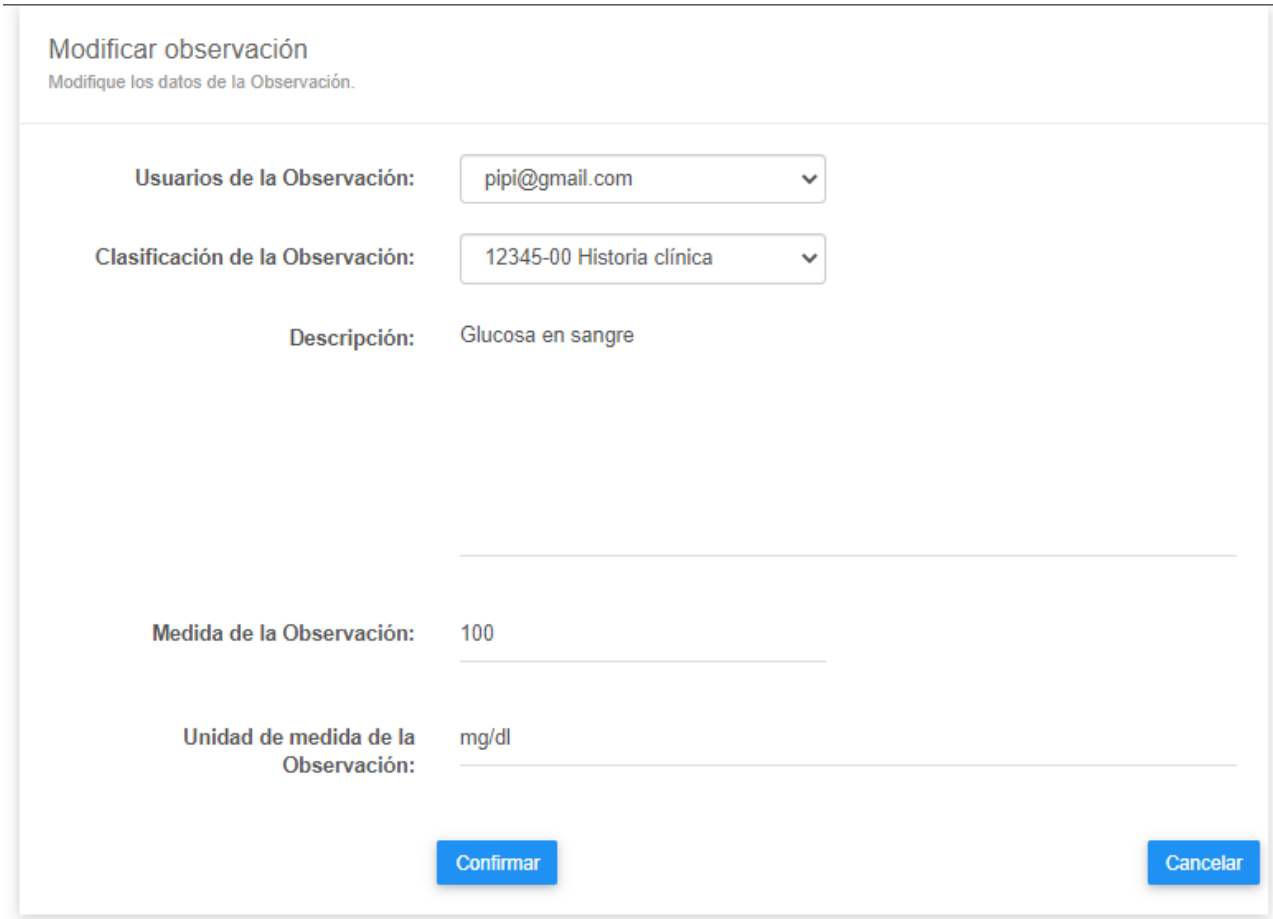


Figura 6.11: Modificar observación

Figura 6.12: Observaciones pacientes

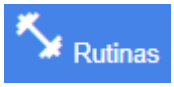


Figura 6.13: Pestaña 'Rutinas'

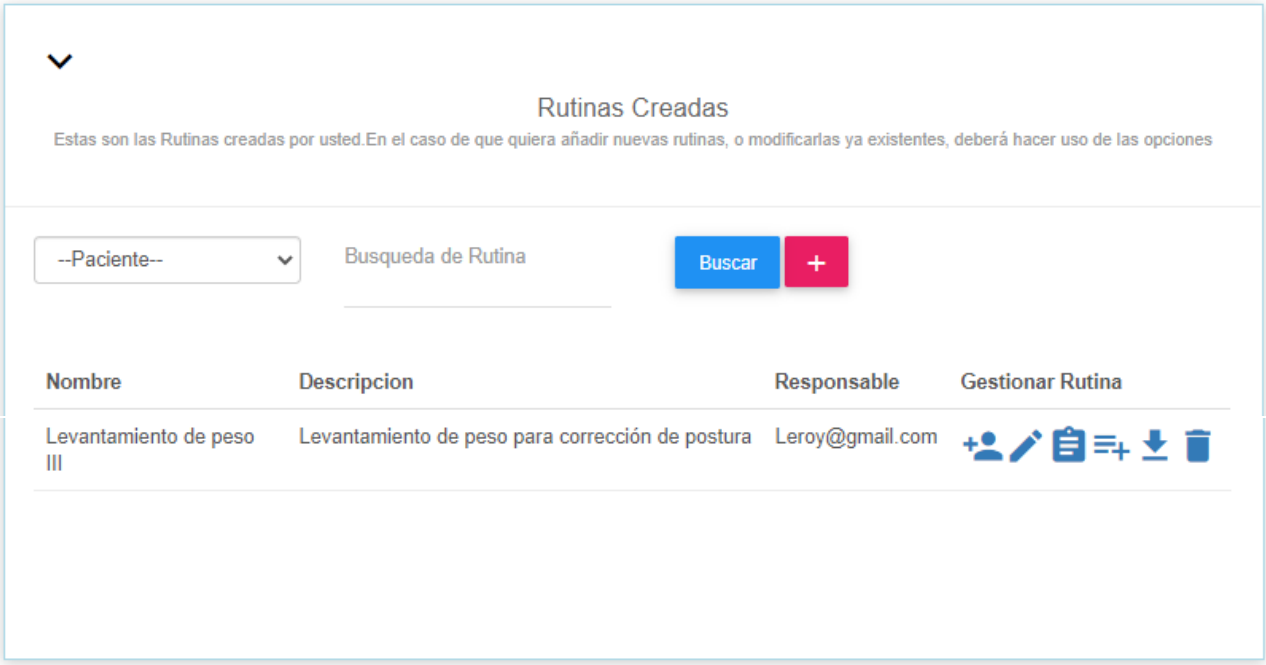
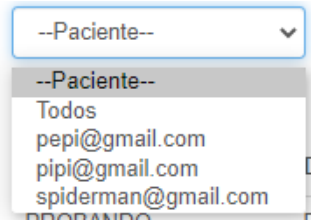


Figura 6.14: Página 'Rutinas'

En esta página, el usuario Especialista puede ver las rutinas que él mismo ha creado y las que han creado el resto de Especialistas mediante . Dichas rutinas están compuestas de un nombre y una descripción.

El usuario Especialista puede asociar dicha rutina a un usuario , modificar la rutina y los usuarios asociados a esta , añadir ejercicios a dicha rutina , ver los ejercicios que posee la rutina , descargar un json con la información de la rutina y sus ejercicios y eliminar la rutina . Con el filtro "paciente" pueden obtenerse las rutinas relativas a un paciente determinado.



Las rutinas pueden buscarse a través de su nombre mediante el cuadro de búsqueda.

Y pueden añadirse nuevas mediante el botón

Figura 6.15: Página 'Añadir rutina'

Campos de la página 'Añadir rutina'

- Usuarios de la rutina: Lista desplegable de pacientes a elegir a los que se les quiere asignar dicha rutina (Los Especialistas no aparecen en la lista).

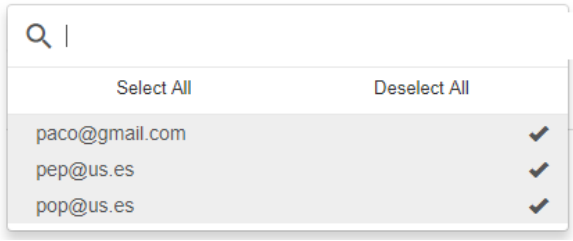


Figura 6.15.1: Lista usuarios

- Responsable de la rutina (Especialista con cierta cualificación)
- Nombre de la rutina
- Descripción de la rutina

Para acceder a las rutinas del resto de Especialistas solo hay que utilizar el enlace correspondiente en la lista desplegable.

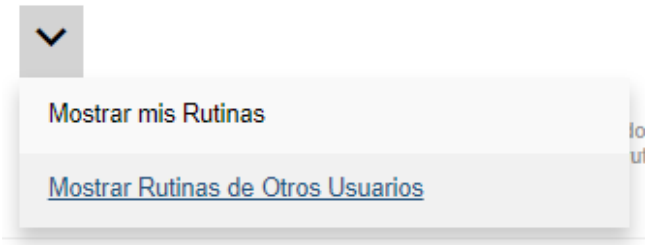


Figura 6.16: Rutinas de otros usuarios

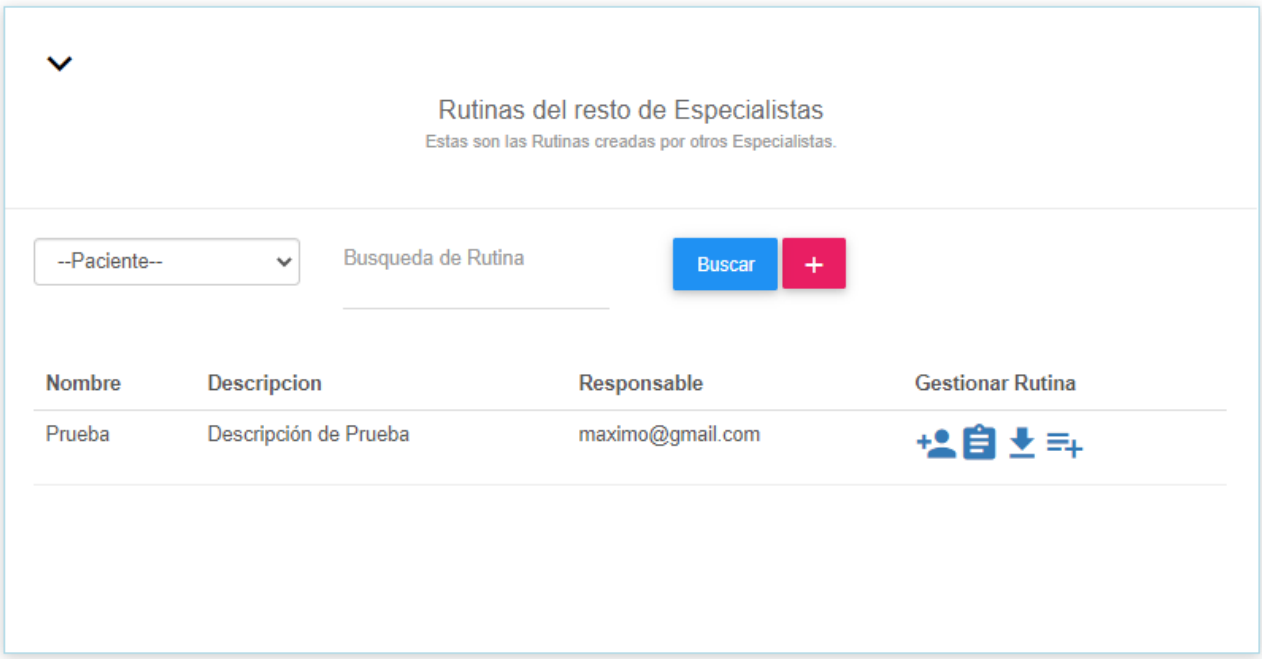


Figura 6.17: Página Rutinas de otros usuarios

La vista de las rutinas de otros usuarios Especialistas es similar a la de las propias, exceptuando que, en este apartado las rutinas no pueden modificarse ni eliminarse, ya que eso solo concierne al creador de estas. Sin embargo, cualquier usuario Especialista puede añadir ejercicios a cualquier rutina.

Ahora se procederá a mostrar el resto de acciones sobre rutinas mencionadas anteriormente.

Asociar rutina

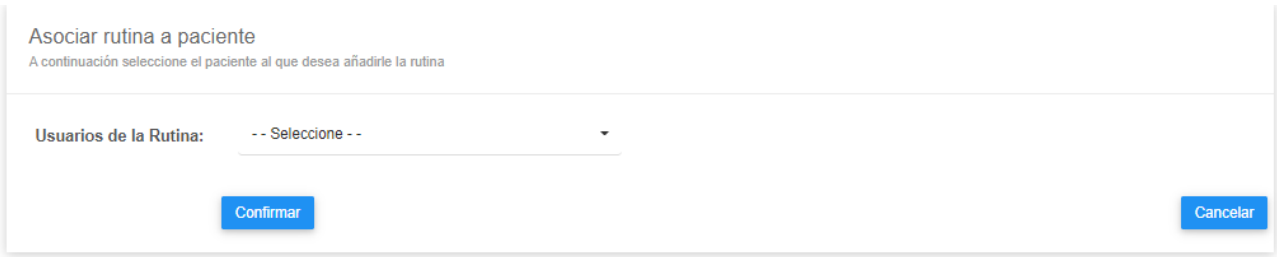


Figura 6.18: Página 'Rutinas' asociar rutina a paciente

Modificar rutina

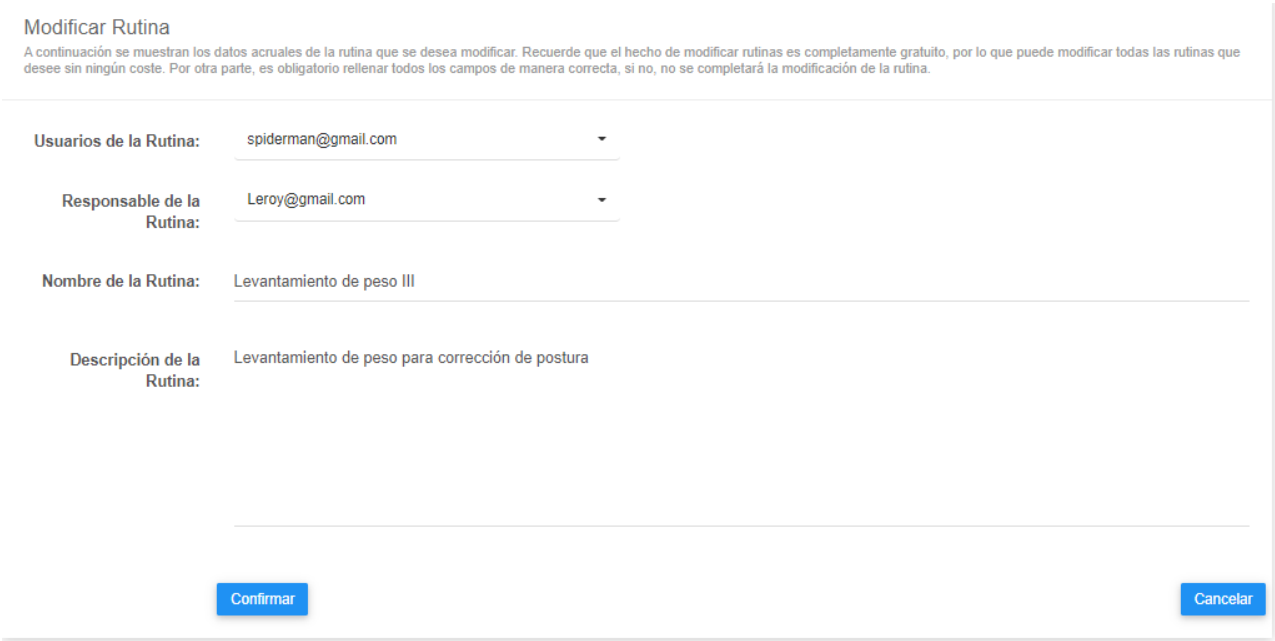


Figura 6.19: Pestaña Modificar Rutina

Añadir ejercicio a rutina y Ejercicios asociados a rutina

Para entender mejor este apartado, primero explicaremos la pestaña relacionada con los ejercicios.

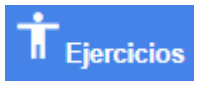


Figura 6.20: Pestaña Ejercicios

Figura 6.21: Página Ejercicios

En esta página, el usuario Especialista puede ver los ejercicios que él mismo ha creado y los que han creado el resto de Especialistas mediante . Dichos ejercicios están compuestos de un nombre, los pacientes asociados al ejercicio o si el ejercicio ha sido creado específicamente para añadirlo a una rutina (**Para Rutina), una descripción, el estado de forma del usuario al que se le ha asignado el ejercicio y las repeticiones que debe realizar del ejercicio.

El usuario Especialista puede modificar el ejercicio , añadir un video al ejercicio (en el caso de que no posea ya uno) y eliminar el ejercicio . En el caso de que el ejercicio posea un vídeo, este vídeo puede ser visualizado y/o eliminado .

Al igual que en las rutinas, pueden añadirse nuevos mediante el botón

Y buscarlos

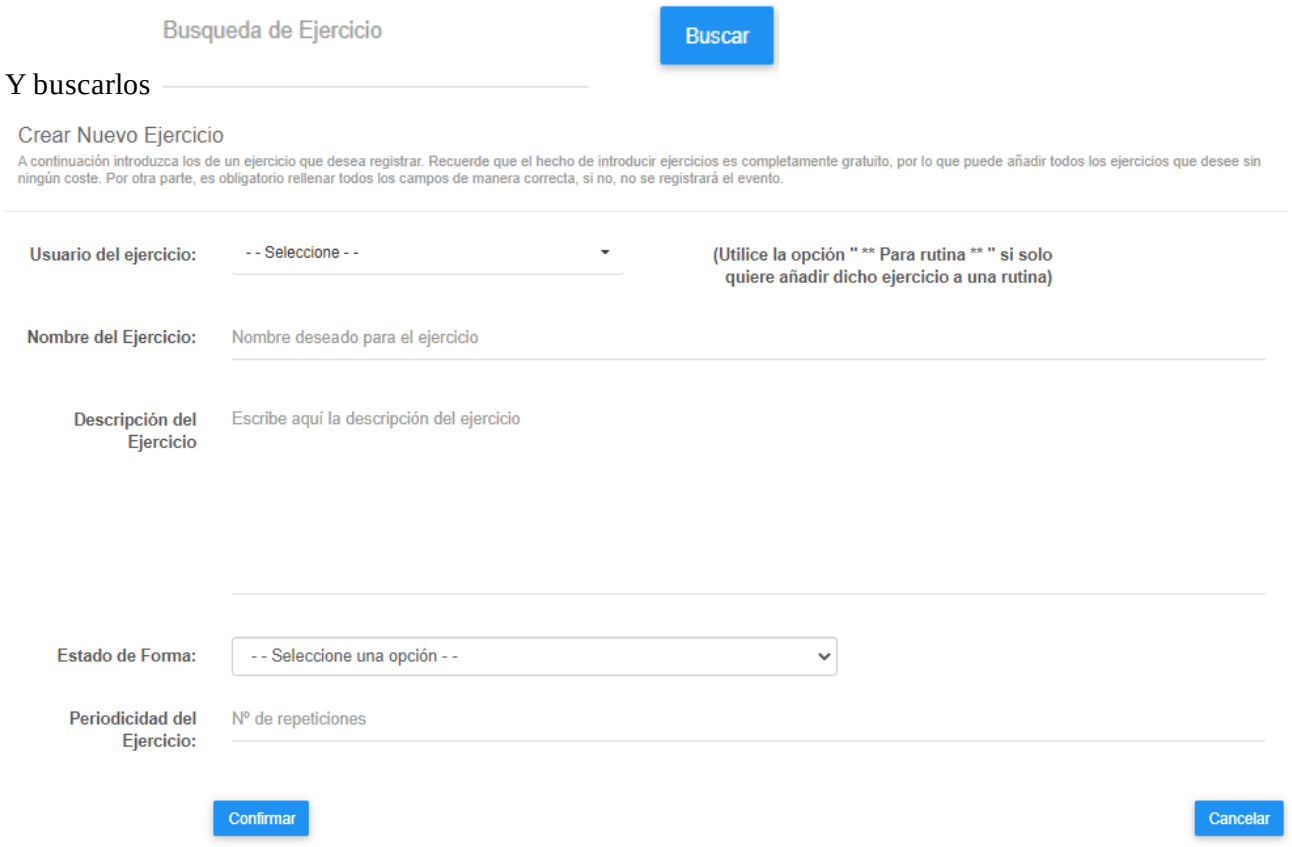


Figura 6.22: Página Añadir Ejercicio

Campos de la página 'Añadir ejercicio'

- Usuarios del ejercicio: Lista desplegable de usuarios a elegir a los que se les quiere asignar dicho ejercicio (Los Especialistas no aparecen en la lista).
- Nombre del ejercicio
- Descripción del ejercicio
- Estado de forma: Estado de forma del usuario del ejercicio

- Repeticiones del ejercicio

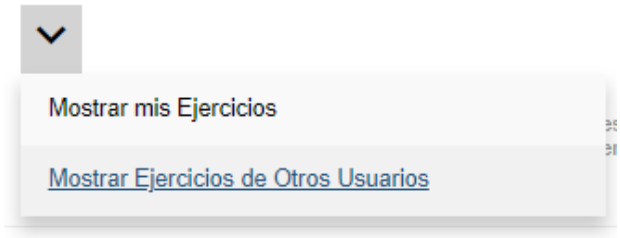


Figura 6.23: Ejercicios de otros usuarios

Figura 6.24: Página Ejercicios de otros usuarios

La vista de los ejercicios de otros usuarios Especialistas es similar a la de los propios, exceptuando que, en este caso, existe una columna "Creador" que indica quién es el creador del ejercicio, así como que en este apartado no pueden añadirse videos a los ejercicios, y que estos no pueden modificarse ni eliminarse, ya que eso solo concierne al creador de estos. Si el ejercicio posee un vídeo, se ofrecerá la opción de visualizarlo .

Ahora procederemos a explicar el resto de acciones a realizar sobre un ejercicio.

Modificar Ejercicio

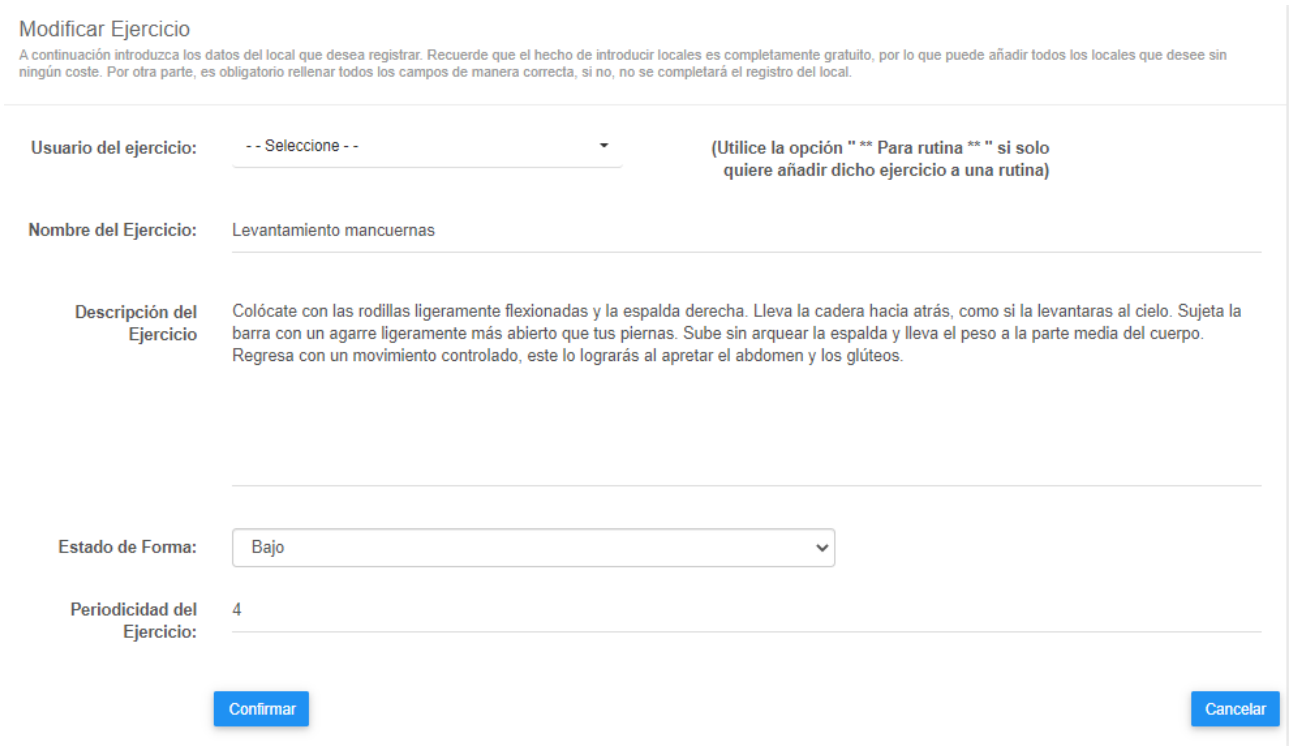


Figura 6.25: Página Modificar Ejercicio

Eliminar ejercicio

Figura 6.26: Eliminar Ejercicio

Añadir vídeo

Para añadir el vídeo, el usuario Especialista deberá rellenar el campo URL con la url perteneciente al vídeo del ejercicio, el cuál debe estar alojado en Youtube.

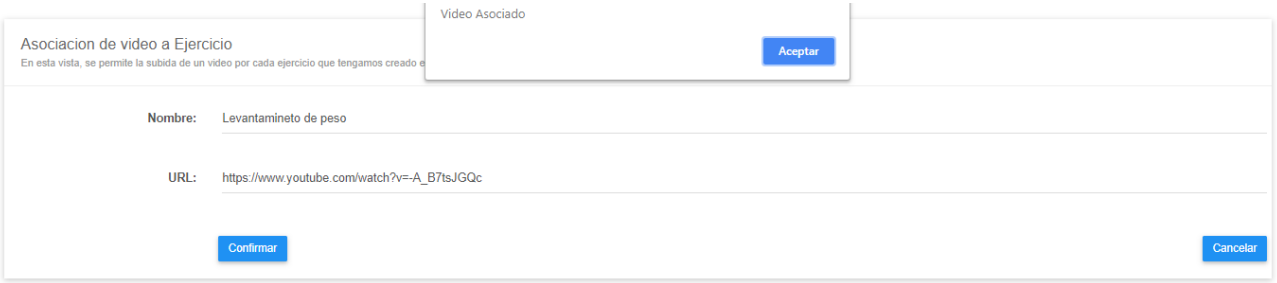


Figura 6.27: Página Añadir vídeo

Ver vídeo

En esta página podrá visualizarse el vídeo de Youtube asociado al ejercicio. No es necesario salir de MyTraining a Youtube para ello ya que el vídeo se encuentra embebido en la aplicación web.

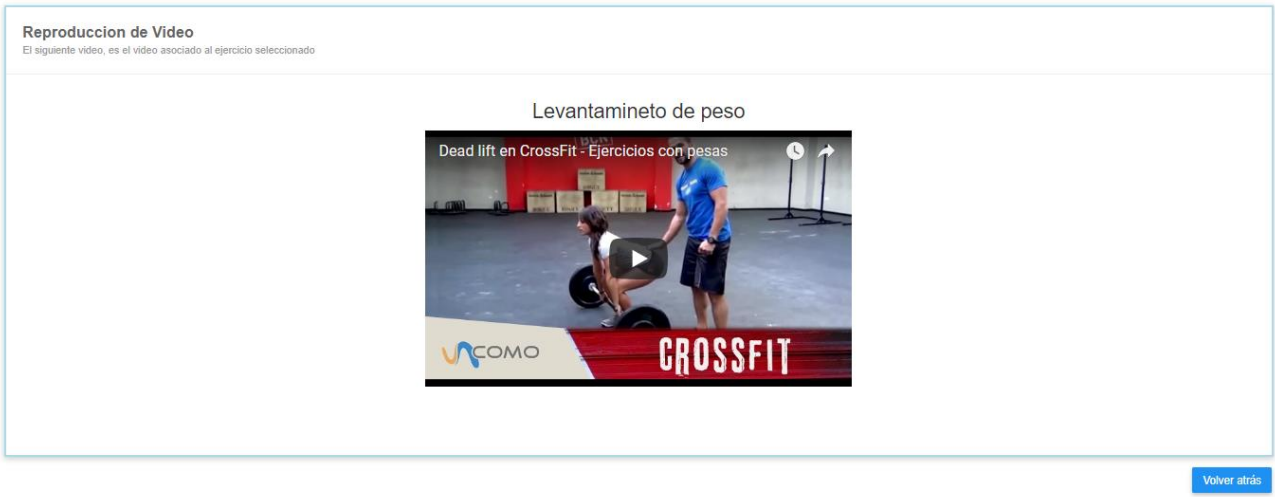


Figura 6.28: Página Ver vídeo

Eliminar vídeo

Figura 6.29: Eliminar vídeo

Una vez explicados los ejercicios y las acciones que pueden realizarse sobre ellos, retomamos las rutinas.

Añadir ejercicio a rutina

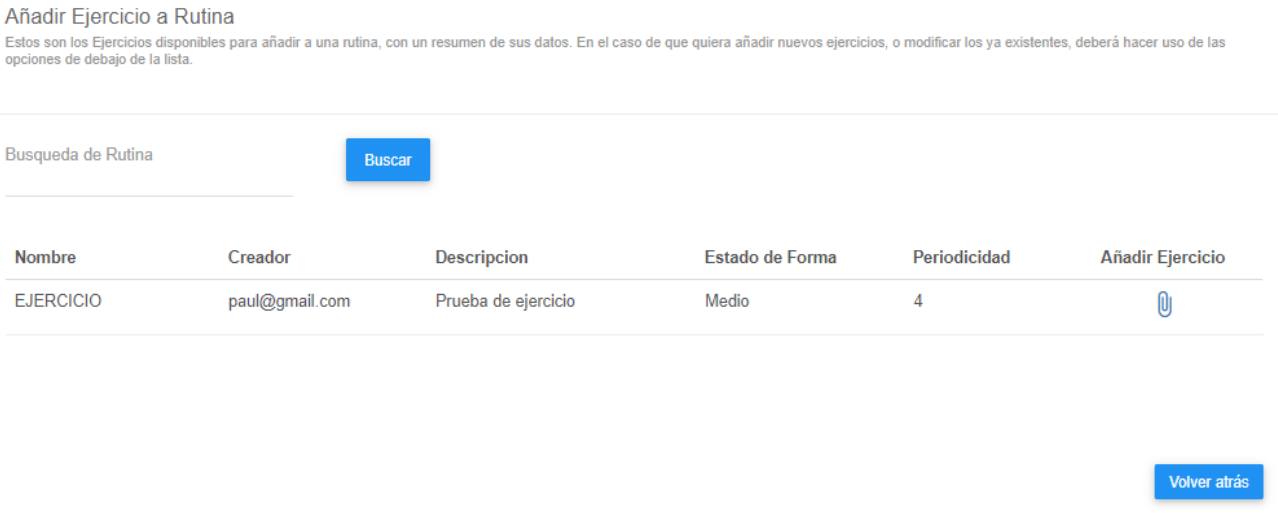


Figura 6.30: Página Añadir Ejercicio a Rutina

Se muestran todos los ejercicios disponibles de todos los usuarios Especialistas para añadir. Aquellos ya añadidos a esa rutina no aparecerán en la lista.

Los ejercicios se añaden pulsando  .

Figura 6.31: Ejercicio asociado

Si el ejercicio que se va a añadir a la rutina no tiene vídeo aparecerá el siguiente mensaje

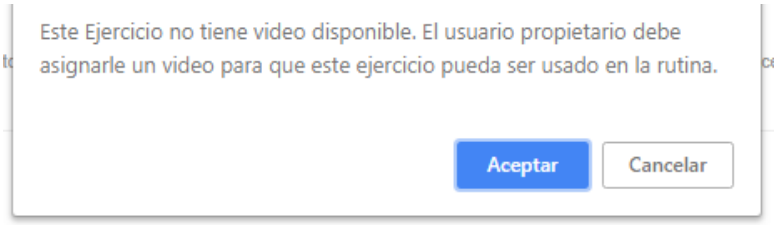
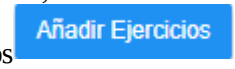


Figura 6.32: Ejercicio sin vídeo

 Ver ejercicios de rutina

Figura 6.33: Página Ver Ejercicios de Rutina

Desde esta página, aparte de ver los ejercicios que tiene una rutina, así como sus datos y acceder a su vídeo asociado, se pueden quitar dichos ejercicios  y añadir nuevos 

.

Eliminar rutina

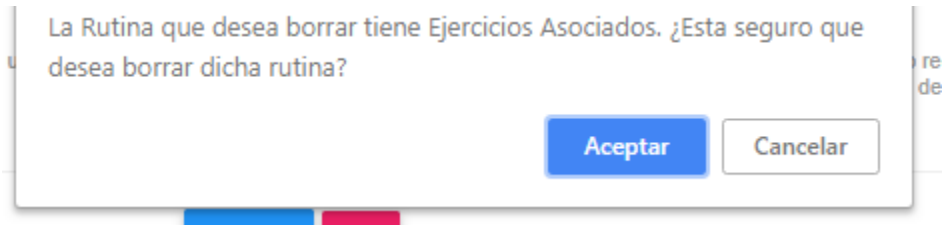


Figura 6.34: Eliminar rutina 1

Figura 6.35: Eliminar rutina 2

Descargar rutina

La descarga de la rutina en un json tiene como finalidad su uso en la aplicación móvil homóloga a esta aplicación Web.

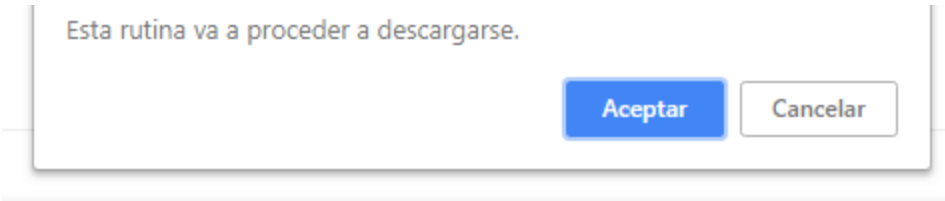


Figura 6.36: Descargar rutina


```
[{"Nombre Rutina":"PECHO","Creador":"GEMA","Descripcion
Rutina":"RecuÃ©state de espalda sobre un banco y sujeta 2
mancuernas al nivel del pecho, a los lados del cuerpo, con las
palmas apuntando hacia tus pies.\n\nEleva las mancuernas en
forma recta hacia arriba hasta que tus codos se encuentren
cerca de trabarse y bÃ¡jalas lentamente luego de una breve
pausa.\n\nExhala al levantar las mancuernas e inhala al
bajarlas."},{"Informacion
Ejercicios":[{"Nombre":"BICEPS","Subtitulo":"PAUL","Descripcio
n":"hhhh","Estado de forma":"Medio","Repeticiones":9}]}]
```


Figura 6.37: JSON rutina

Figura 6.38: Pestaña Ajustes

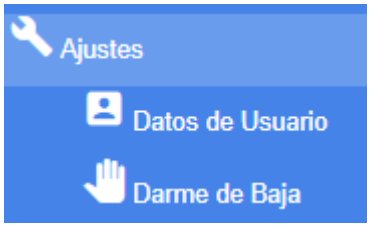


Figura 6.39: Pestaña Ajustes 2

Datos de Usuario

En esta página el usuario puede modificar sus datos personales

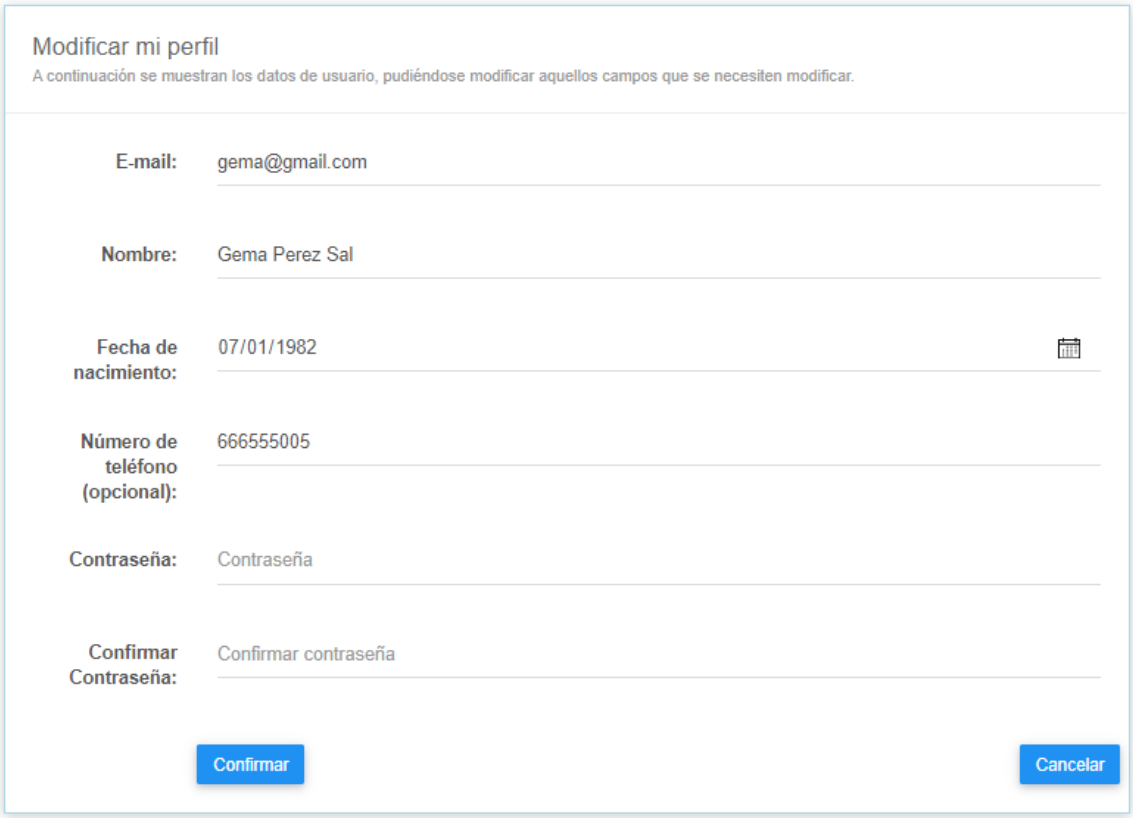


Figura 6.40: Página Datos de Especialista

Darme de Baja

En esta página el usuario podrá darse de baja borrándose de esta manera todos los datos relacionados con este de la aplicación.

Figura 6.41: Página Darse de Baja

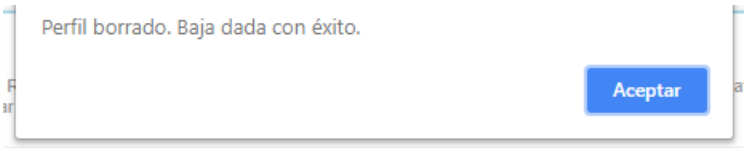
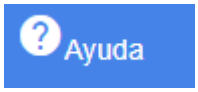
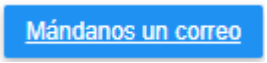


Figura 6.42: Perfil borrado



En esta página se puede contactar mediante un mensaje de correo con personal que resolverá las dudas de los usuarios. 

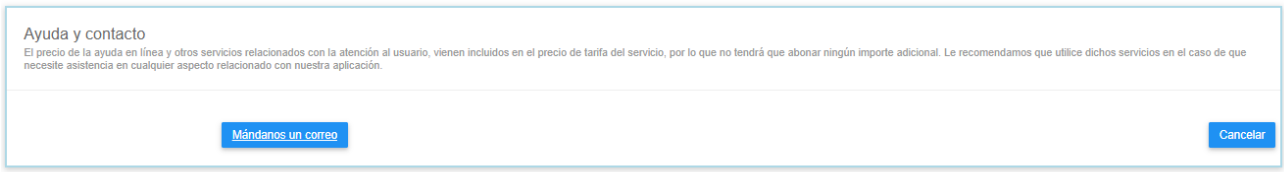


Figura 6.43: Página Ayuda

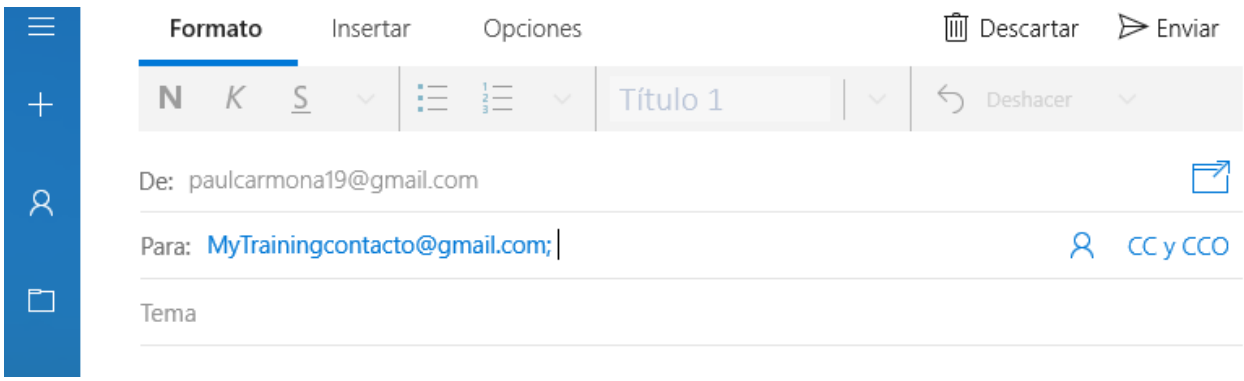


Figura 6.44: Correo

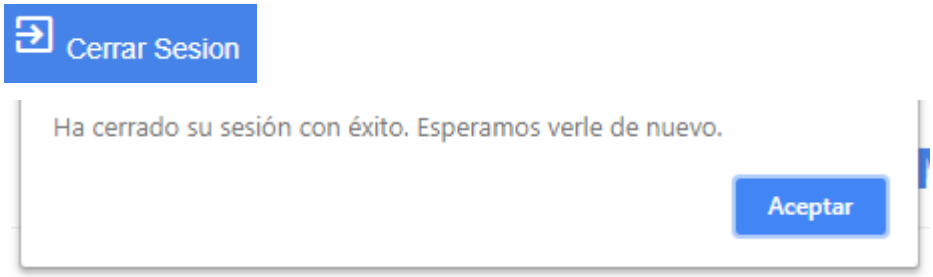
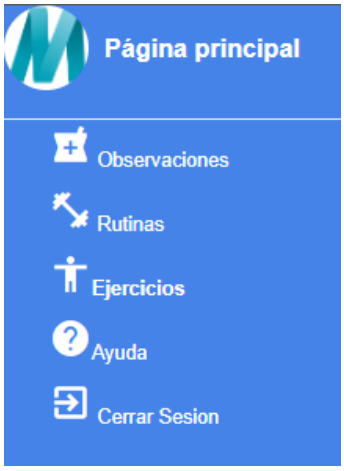


Figura 6.45: Cerrar sesión

Ahora procedemos a revisar la vista del paciente.

La vista del paciente es similar a la del Especialista, exceptuando algunas funciones que vamos a detallar:



Este es el menú lateral de un usuario. Como puede observarse, la pestaña "Usuarios" y "Ajustes" no se encuentra ya que esas acciones conciernen al Especialista.

Un paciente solo podrá ver, modificar y eliminar las observaciones que él mismo genere acerca de las rutinas que tenga asociadas. Así como, crear nuevas.

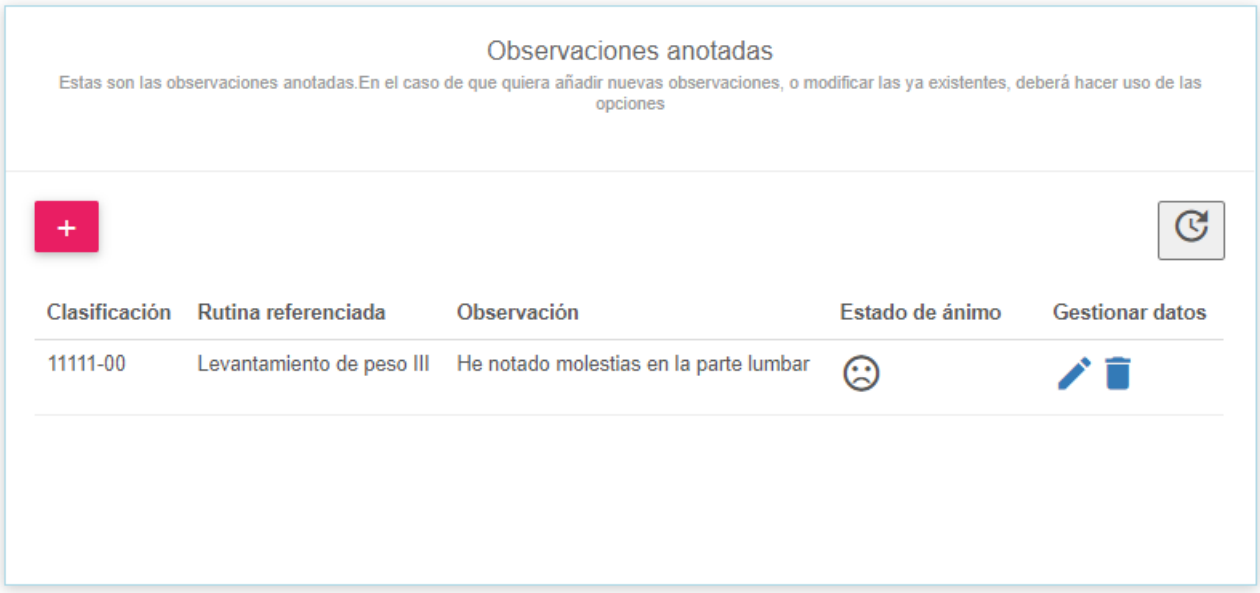


Figura 6.46.1: Observación paciente

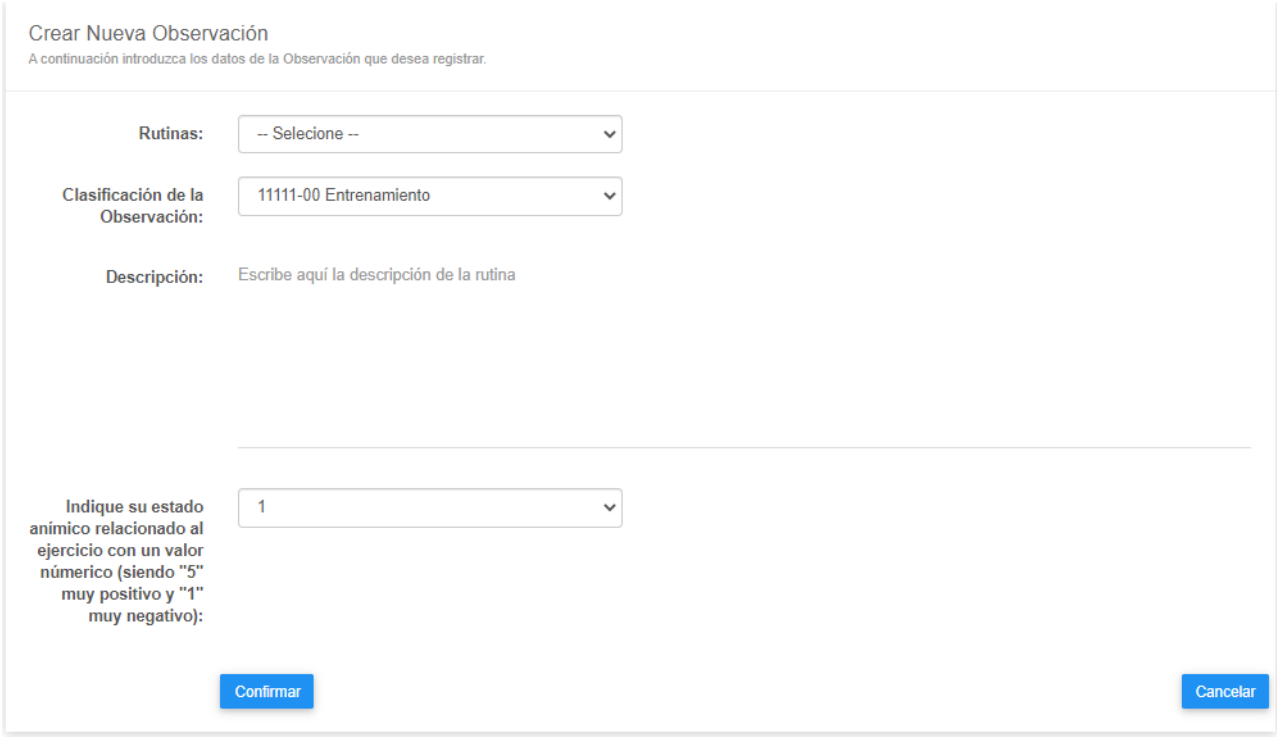


Figura 6.46.2: Crear observación paciente

Campos de la página ‘Añadir Observación de paciente”

- Rutina a la que hace referencia dicha observación
- Clasificación de la observación (fija)
- Descripción
- Estado anímico del paciente relacionado con la realización de la rutina (basado en un valor numérico del 1 al 5 que se verá representado gráficamente con caritas)

De igual manera, solo podrá ver las rutinas y ejercicios que le asignen los Especialistas, así como descargar las rutinas, ver los ejercicios que posea cada una de ellas y los videos de los respectivos ejercicios.

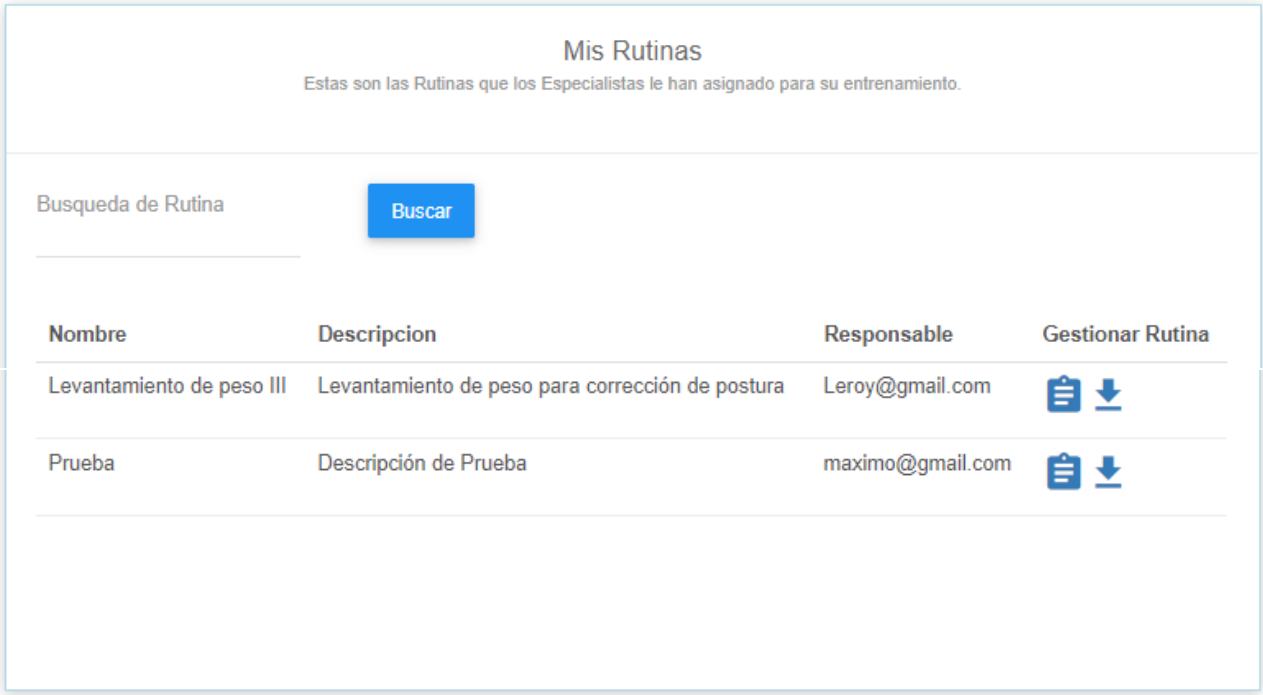


Figura 6.47: Rutinas de paciente

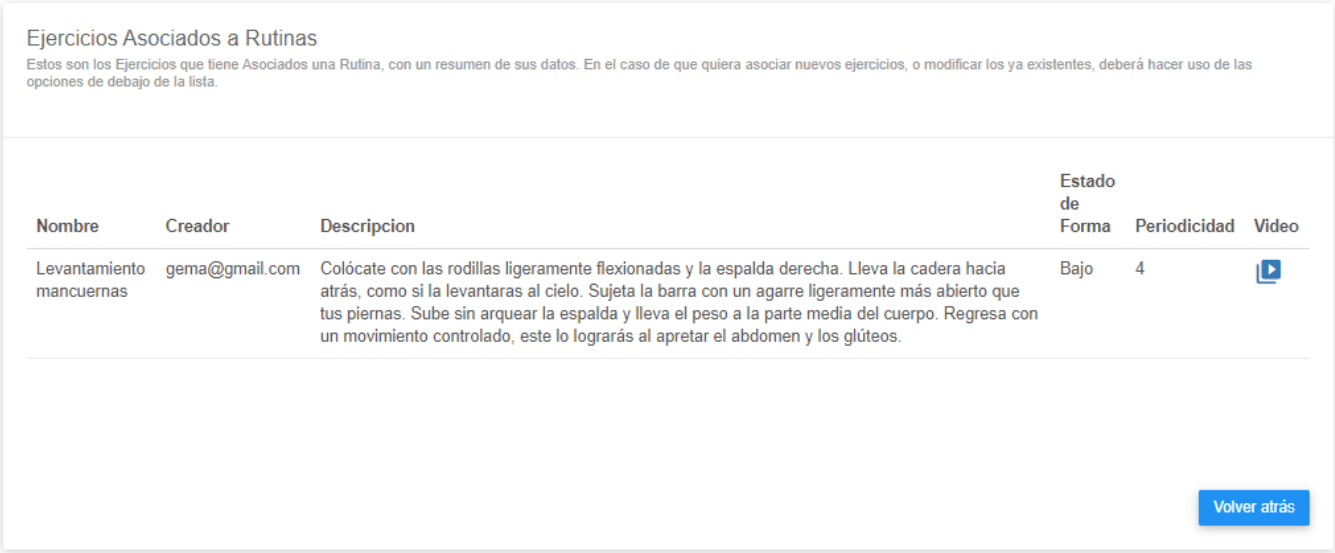


Figura 6.48: Ejercicios de rutinas de paciente

Figura 6.49: Ejercicios de paciente

7. Conclusiones y líneas futuras

Finalmente, tras tiempo de dedicación a entender y aprender a utilizar el estándar HL7 FHIR, se ha conseguido que la plataforma interactúe correctamente con el servidor HAPI FHIR y que los datos generados se almacenen según estipula dicho estándar.

Se ha logrado la migración de las acciones generadoras de datos de la plataforma "MyTraining" de manera que, se produce una comunicación establecida mediante creación de un contexto con el servidor HAPI FHIR a través de java y se almacenan y mapean los datos en la Base de Datos FHIR con transacciones y objetos en formato JSON.

Este proyecto ofrece la posibilidad de obtener un entrenamiento o tratamiento personalizado, asistido por especialistas de una forma cómoda y virtual, para aquellas personas con características y necesidades especiales, facilitándoles mejorar su salud.

Durante el proceso de realización de este proyecto he aprendido mucho acerca de los estándares existentes que buscan la interoperabilidad en el ámbito sanitario. La creciente necesidad de un estándar que permita la globalización de los datos médicos en los diferentes sistemas hospitalarios ha impulsado el desarrollo y mi interés por este proyecto.

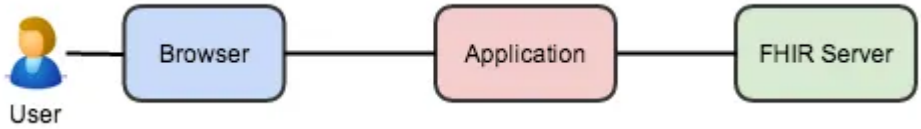
En definitiva, HL7 FHIR, es un fuerte candidato para solventar dicha necesidad. Parece dispuesto a quedarse y a seguir creciendo al ser un estándar optimizado y relativamente fácil de aprender e implementar.

7.1 Líneas futuras

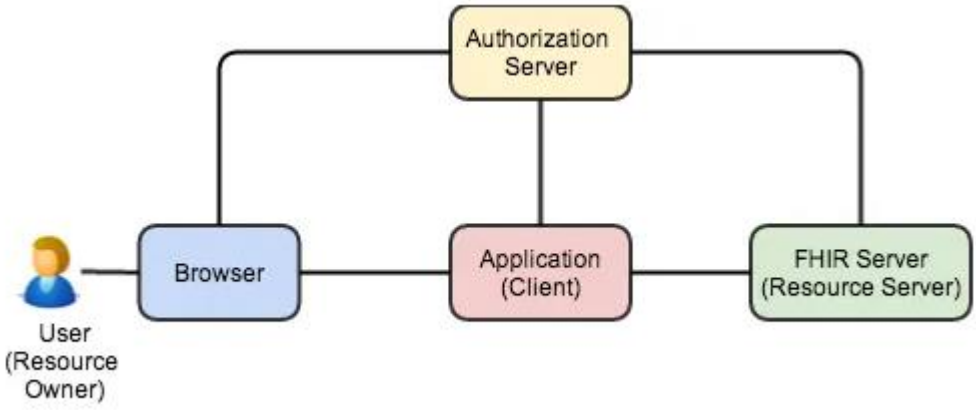
7.1.1 OAuth 2.0

Una interesante línea futura para este proyecto sería la de implementar junto a FHIR el framework OAuth2, que separa el rol de autenticación y autorización. OAuth2 permite a una aplicación tener acceso a los datos de un usuario sin que esta necesite tener una copia de las credenciales de los usuarios en su sistema.

La arquitectura de nuestra aplicación se vería de la siguiente manera en un esquema simple:



Si se implementara OAuth2, la arquitectura quedaría de la siguiente manera:



Como puede verse, con OAuth2 se genera un nuevo componente que es el Servidor de Autorización, único componente encargado de las credenciales de usuario. El resto de componentes confían en dicho servidor para identificar a los usuarios e indicar si los datos sobre ellos se pueden entregar a la aplicación que realiza la llamada.

Normalmente, el servidor de autorización será un sistema de confianza donde el usuario ya tiene una cuenta, como Google o Amazon, pero puede ser un componente independiente, o incluso la funcionalidad del servidor FHIR.

7.1.2 Dispositivos wearables

Otra posible línea futura puede ser la implementación de dispositivos wearables, de manera que recoja datos del paciente que puedan almacenarse en la base de datos FHIR para realizar un seguimiento de este aún más completo y en tiempo real.

Los dispositivos más populares son las pulseras inteligentes, que pueden recoger medidas tales como, el nivel de glucosa en sangre, ritmo cardíaco, temperatura, etc. Dichas pulseras también podrían disponer de un sistema de aviso paciente-especialista o encargado en caso de que al paciente le ocurriese alguna incidencia.

Los sensores van un paso más allá, están adheridos en el cuerpo. Están perfectamente diseñados para que no sean molestos o produzcan inflamaciones. Un proyecto activo es el de la Universidad de Illinois, actualmente desarrollan dispositivos que miden el esfuerzo de energía, el ritmo y cómo varía la frecuencia cardíaca.

REFERENCIAS

[1] En relación a FHIR:

[1][1] Healthcare in the Age of Interoperability: The Promise of Fast Healthcare Interoperability Resources - IEEE Journals & Magazine

[1][2] Health Care in the Age of Interoperability: The Potential and Challenges - IEEE Journals & Magazine

[1][3] Healthcare in the Age of Interoperability: Part 3 - IEEE Journals & Magazine

[1][4] Health Care in the Age of Interoperability: Part 4 - IEEE Journals & Magazine

[1][5] Health care in the age of interoperability part 5: the personal health record - IEEE Journals & Magazine

[1][6] Health Care in the Age of Interoperability Part 6: The Future of FHIR - IEEE Journals & Magazine

[1][7] Local EHR management based on FHIR - IEEE Conference Publication

[1][8] HL7 FHIR: An Agile and RESTful approach to healthcare information exchange - IEEE Conference Publication

[1][9] Health Level Seven International - Homepage | HL7 International

[1][10] Index - FHIR v4.0.1 (hl7.org)

[1][11] Table of Contents - HAPI FHIR Documentation

[1][12] Los 5 estándares HL7 fundamentales | Caduceus Connecting eHealth

[1][13] Interoperabilidad clínica: ¿Has oído hablar de FHIR? - Dedalus España (dedalusgs.com)

[1][14] Arquitectura software para la prescripción de ejercicio físico personalizado / Software Architecture for Customized Physical Exercise Prescription (rediris.es)

[1][15] GitHub - hapifhir/hapi-fhir: 🔥 HAPI FHIR - Java API for HL7 FHIR Clients and Servers

[1][16] ¿Qué es la eSalud? Definición y concepto | LaeSalud.com

[1][17] ¿Qué son y cuál es la diferencia entre EMR y EHR? | by Jose Miguel Sainz | Ecaresoft's Blog

[1][18]Leaming HL7 FHIR Using the HAPI FHIR Server and Its Use in Medical Imaging with the SIIM Dataset | SpringerLink

[1][19] Sidharth Ramesh - YouTube

[1][20] Welcome to Codota!

[1][21] ¿Qué es FHIR y por qué debería preocuparme? | Hablando de eSalud (wordpress.com)

[1][22] Interoperable and discrete eHealth Data Exchange between Hospital and Patient - IEEE Conference Publication

[1]][23] Use of an eHealth tool for exercise training and online contact in people with severe chronic obstructive pulmonary disease on long-term oxygen treatment: A feasibility study - Pernilla Sönnerfors, Karin Wadell, Ing-Mari Dohrn, André Nyberg, Michael Runold, Alexandra Halvarsson, 2020 (sagepub.com)

[2] Receta deportiva: Receta Deportiva (consejo-colef.es)

[3] En relación a contenedores:

[3][1]Docker Hub

[3][2]Docker Desktop for Windows - Docker Hub

[3][3] Sidharth Ramesh - YouTube

[4] En relación a MySQL:

[4][1] Web Oficial de MySQL: https://www.mysql.com/

[4][2] Documentación de XAMPP: https://www.apachefriends.org/community.html

[5] En relación a Spring Framework y Módulos Relacionados:

[5][1] Documentación de Spring: https://spring.io/docs

http://forum.spring.io/forum/spring-projects/security

[5][2] Guías de Spring: https://spring.io/guides

[5][3] API REST: https://openehr.atlassian.net/wiki/display/spec/openEHR+REST+APIs
https://codezone4.wordpress.com/2012/11/08/restful-web-services-java-mysql-and-json/

[5][4] Tutoriales de Propósito General:
http://www.mkyong.com/tutorials/ [3][5] Tutoriales Específicos en Spring: http://www.baeldung.com/

[6] En relación a Java:

[6][1] General: https://javapointers.com/

[6][2] Creación de JSON en Java: https://www.java2novice.com/java-json/javax.json/create-json-array/

[7] En relación a BootStrap(HTML5, CSS3 y JS):

[7][1] Referencia del W3C de HTML5: https://dev.w3.org/html5/html-author/

[7][2] Guía de HTML5 en MDN: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5

[7][3] Tutoriales de HTML5 de W3C: http://www.w3schools.com/html/

[7][4] Tutoriales de CSS3 de W3C: http://www.w3schools.com/css/css3_intro.asp

[7][5] Tutoriales y Documentación de BootStrap: https://getbootstrap.com/docs/4.0/getting-started

[8] En relación a JavaScript y librerías relacionadas:

[8][1] Tutoriales de JavaScript de W3C: http://www.w3schools.com/js/

[8][2] Web Oficial de JQuery: https://jquery.com/

[8][3] Web Oficial de JQuery Validation Plugin: https://jqueryvalidation.org/

[8][4] Perfil de Github de Klaus Hartl (librerías de cookies): https://github.com/carhartl/

[8][5] Ajax: https://www.webucator.com/tutorial/learn-ajax/jquery/ajax-with-jquery-reading.cfm

[9] Cuestiones varias: https://stackoverflow.com/

[10] OAuth2:

[10][1] Verifiable Delegated Authorization for User-Centric Architectures and an OAuth2 Implementation - IEEE Conference Publication

[10][2] FHIR and OAuth2 | Hay on FHIR (fhirblog.com)

[10][3] Una introducción a OAuth 2 | DigitalOcean

ANEXO A: API REST DOCUMENTATION

En este anexo, mostraremos las URIs que se emplean para realizar las llamadas a las diferentes funciones que realizarán las peticiones oportunas a la Base de Datos y los parámetros que necesitan estos.

A.1 Recurso Usuario

Método	URI	Parámetros	Acción
POST	"/RutinaRegister//"	Atributos de un paciente: Nombre, Email, Fecha de nacimiento	Crear un nuevo paciente con sus datos personales
POST	"/RutinaRegisterAdmin//"	Atributos de un especialista: Nombre, Email, Fecha de nacimiento, cualificación	Crear un nuevo especialista con sus datos personales
POST	"/Rutina_app/{user_id:.+}"	*user_id:* Identificador único del usuario. Atributos de un Usuario.	Modificar los datos personales de un usuario
GET	"/Rutina_app/{user_id:.+}"	*user_id:* Identificador único de un usuario.	Obtener datos de un usuario
GET	"/Rutina_app//"	(Sin parámetros)	Obtener datos de todos los usuarios
GET	"/Rutina_app/rol/{user_id:.+}"	*user_id:* Identificador único de un usuario.	Obtener el rol de un usuario
GET	"/Rutina_app/rol/"	(Sin parámetros)	Obtener los roles de todos los usuarios
DELETE	"/Rutina_app/{user_id:.+}"	*user_id:* Identificador único de un usuario.	Eliminar un usuario junto con sus datos

A.2 Recurso Rutina

Método	URI	Parámetros	Descripción
POST	"/Rutina_app/rutinas/{reset_group:.+}/{rut_id:.+}"	*Reset_group:* Bandera para modificar el grupo de pacientes asociados a la rutina. Atributos de una Rutina.: Nombre, descripción, responsable *rut_id:*Identificador único de una rutina.	Modificar una rutina.
POST	"/Rutina_app/rutinas/{user_id:.+}//"	*user_id:* Identificador único del usuario. Atributos de una Rutina	Crear una nueva rutina con sus datos en la BD local
POST	"/Rutina_app/rutinas/FHIR/{user_id:.+}//"	*user_id:* Identificador único del usuario. Atributos de una Rutina	Crear una nueva rutina con sus datos en BD FHIR
GET	"/Rutina_app/rutinas/{rut_id:.+}"	*rut_id:*Identificador único de una rutina.	Obtener los datos de una rutina.
GET	"/Rutina_app/rut/{user_id:.+}/"	*user_id:* Identificador único de un usuario. Atributos de una Rutina.	Obtener el identificador de una rutina.
POST	"/Rutina_app/rut/add/{user_id:.+}/"	*user_id:* Identificador único de un usuario.	Asociar una rutina a un paciente.

GET	"/Rutina_app/rutinas_user/{user_id:.+}//"	*user_id:* Identificador único de un usuario. Elemento de búsqueda (opcional)	Obtener las rutinas asociadas a un paciente que accede.
GET	"/Rutina_app/rutinas_user_filter/{esp_id:.+}//"	*esp_id:* Identificador único de un usuario.	Obtener las rutinas asociadas a un paciente por especialista que accede.
GET	"/Rutina_app/rutinas/{user_id:.+}//"	*user_id:* Identificador único de un usuario. Elemento de búsqueda (opcional) Bandera de rutinas: propias/Del resto	Obtener las rutinas relacionadas.
DELETE	"/Rutina_app/rutinas/{user_id:.+}/{rut_id:.+}"	*user_id:* Identificador único de un usuario.	Eliminar una rutina junto a sus datos.
POST	"/Rutina_app/downloads/{rut_id:.+}"	*rut_id:*Identificador único de una rutina.	Descargar la rutina en un json

A.3 Recurso Ejercicio

Método	URI	Parámetros	Descripción
POST	"/Rutina_app/ejercicios/{user_id:.+}//"	*user_id:* Identificador único de un usuario. Atributos de un Ejercicio: Nombre, descripción, periodicidad, estado de forma	Crear un ejercicio

		recomendado, pacientes asociados	
GET	"/Rutina_app/ejercicios/{user_id:.+}/{ej_id:.+}"	*user_id:* Identificador único de un usuario. *ej_id:*Identificador único de un ejercicio.	Obtener los datos de un ejercicio
GET	"/Rutina_app/ejercicios/{user_id:.+}//"	*user_id:* Identificador único de un usuario. Elemento de búsqueda (opcional) Bandera de ejercicios: propias/Del resto	Obtener los ejercicios relacionados
GET	"/Rutina_app/ej/{user_id:.+}/"	Atributos de un Ejercicio *user_id:* Identificador único de un usuario.	Obtener el identificador del ejercicio
POST	"/Rutina_app/ej/add/{user_id:.+}/"	*user_id:* Identificador único de un usuario. Atributos de un Ejercicio	Asociar un ejercicio a un usuario
GET	"/Rutina_app/ejercicios_user/{user_id:.+}//"	*user_id:* Identificador único de un usuario. Elemento de búsqueda (opcional)	Obtiene los ejericicios asociados a un usuario
DELETE	"/Rutina_app/ejercicios/{user_id:.+}/{ej_id:.+}"	*user_id:* Identificador único de un usuario.	Eliminar un ejercicio

		*ej_id:*Identificador único de un ejercicio	
DELETE	"/Rutina_app/rutinas/asociaciones/{rut_id:.+}/{ej_id:.+}"	*ej_id:*Identificador único de un ejercicio *rut_id:*Identificador único de una rutina	Eliminar asociación de 'y' ejercicio asociado 'x' a rutina
POST	"/Rutina_app/ejercicios/{user_id:.+}/{ej_id:.+}"	*user_id:* Identificador único de un usuario. *ej_id:*Identificador único de un ejercicio Atributos de un Ejercicio	Modificar un ejercicio
DELETE	"/Rutina_app/rutinas/asociaciones/{rut_id:.+}//"	*rut_id:*Identificador único de una rutina	Eliminar las asociaciones de todos los ejercicios asociados a una rutina 'x'
GET	"/Rutina_app/rutinas/noasociaciones/{rut_id:.+}//"	*rut_id:*Identificador único de una rutina Elemento de búsqueda (opcional)	Obtener los ejercicios que no están asociados a 'x' rutina
POST	"/Rutina_app/rutinas/asociaciones/{rut_id:.+}/{ej_id:.+}"	*user_id:* Identificador único de un usuario. *ej_id:*Identificador único de un ejercicio	Asociar ejercicios a rutinas
GET	"/Rutina_app/rutinas/asociaciones/{rut_id:.+}//"	*rut_id:*Identificador único de una rutina	Obtener los ejercicios asociados a una rutina

A.4 Recurso Vídeo

Método	URI	Parámetros	Descripción
GET	"/Rutina_app/videos/{user_id:.+}/{ej_id:.+}"	*user_id:* Identificador único de un usuario. *ej_id:*Identificador único de un ejercicio	Obtener video
POST	"/Rutina_app/videos/{ej_id:.+}"	*ej_id:*Identificador único de un ejercicio Atributos de un Video: Nombre, URL	Añadir video
DELETE	"/Rutina_app/videos/{user_id:.+}/{ej_id:.+}"	*user_id:* Identificador único de un usuario. *ej_id:*Identificador único de un ejercicio	Eliminar video

A.5 Recurso Observación

Método	URI	Parámetros	Descripción
POST	"/Rutina_app/FHIR/create"	Atributos de la observación: creador, clasificación, descripción, unidad, medida	Crear una observación
POST	"/Rutina_app/update/FHIR/{obs_id:.+}"	*obs_id:* Identificador único de la rutina. Atributos de una Observación	Modificar observación de especialista
POST	"/Rutina_app/update/FHIR/user/{obs_id:.+}"	*obs_id:* Identificador único de la rutina.	Modificar observación de paciente

		Atributos de una Observación	
GET	"/Rutina_app/modify/FHIR/{obs_id:.+}"	*obs_id:* Identificador único de la rutina.	Obtener datos de la observación a modificar por especialista
GET	"/Rutina_app/modify/FHIR/user/{obs_id:.+}"	*obs_id:* Identificador único de la rutina.	Obtener datos de la observación a modificar por paciente
POST	"/Rutina_app/update/FHIR/user/{obs_id:.+}"	*obs_id:* Identificador único de la Observación. Atributos de una Observación	Modificar observación de paciente
GET	"/Rutina_app/FHIR/observation/{patientId:.+}"	*patient_id*:Identificador único de un paciente.	Obtener los datos de observaciones de un paciente que accede
GET	"/Rutina_app/FHIR/all_observation/{patientId:.+}/{obs_id:.+}"	*patient_id*:Identificador único de un paciente. *practitionerId_id:* Identificador único de la Observación	Obtener los datos de las observaciones disponibles por especialista
DELETE	"/Rutina_app/FHIR/delete/{obsId:.+}"	*obs_id:* Identificador único de la Observación.	Eliminar una observación

ANEXO B: MANUAL DE INSTALACIÓN Y DESPLIEGUE DE LA APLICACIÓN

En este anexo se explicará el proceso de instalación y despliegue de la aplicación sobre el sistema operativo sobre el que se ha desarrollado el proyecto, Windows 10 de 64 bits.

B.1 Instalación de Java JDK 8

1. Descargar JDK para Windows a través de la página http://www.oracle.com/technetwork/java/javase/downloads/jdk10-downloads-4416644.html
2. Ejecutar el .exe para instalarlo y escoger una ruta para la instalación. Debemos recordar dicha ruta para un paso posterior.

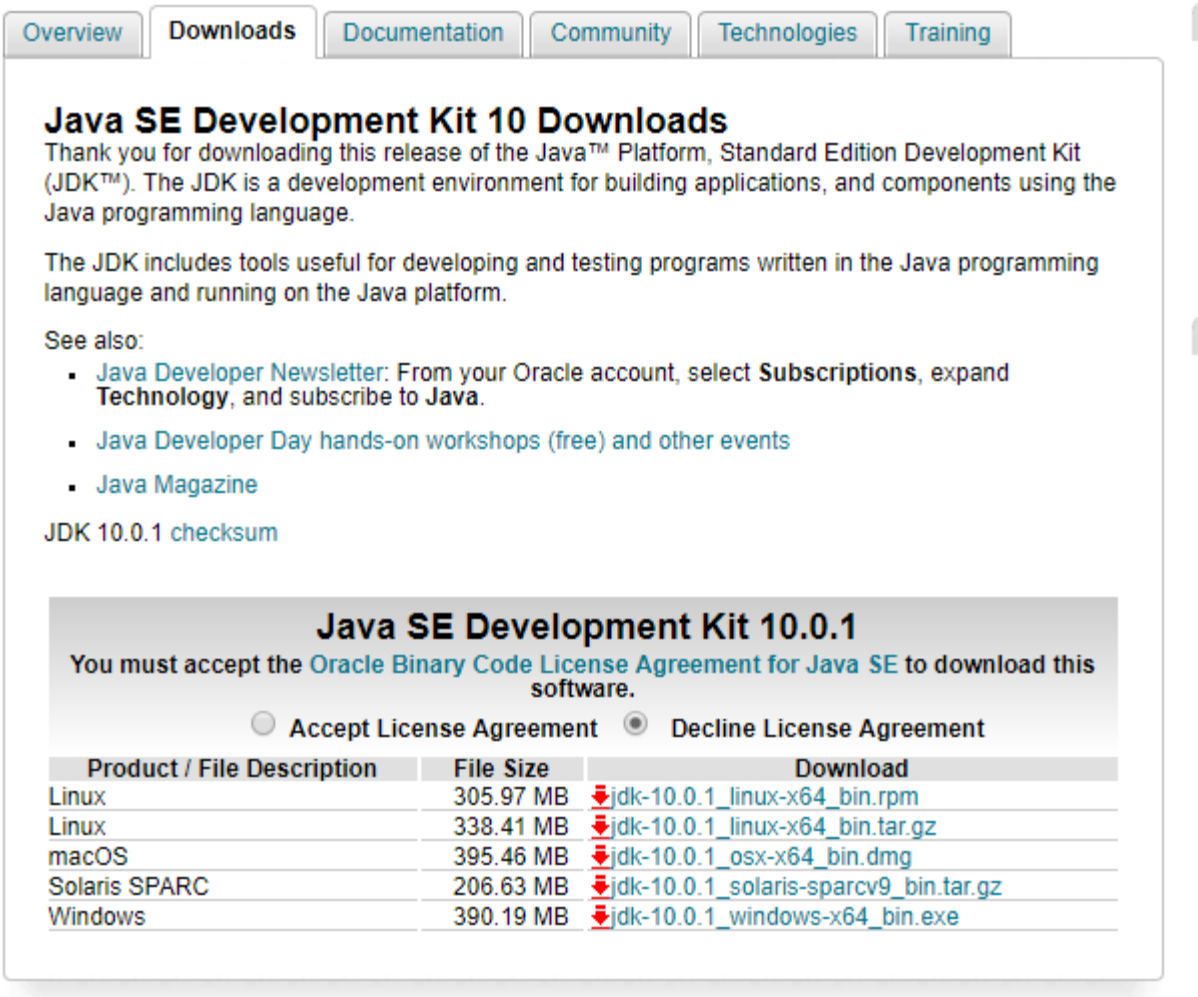


Figura B.1: JDK

B.2 MAVEN

1. Descargamos el archivo .zip de Maven de la página

   https://maven.apache.org/download.cgi

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.5.3-bin.tar.gz	apache-maven-3.5.3-bin.tar.gz.sha1	apache-maven-3.5.3-bin.tar.gz.asc
Binary zip archive	apache-maven-3.5.3-bin.zip	apache-maven-3.5.3-bin.zip.sha1	apache-maven-3.5.3-bin.zip.asc
Source tar.gz archive	apache-maven-3.5.3-src.tar.gz	apache-maven-3.5.3-src.tar.gz.sha1	apache-maven-3.5.3-src.tar.gz.asc
Source zip archive	apache-maven-3.5.3-src.zip	apache-maven-3.5.3-src.zip.sha1	apache-maven-3.5.3-src.zip.asc

Figura B.2: MAVEN

2. Lo descomprimimos con WinRar o similar y lo ubicamos en la ruta que escojamos, a ser preferible una cercana a la raíz C:\, que tendremos que recordar más adelante para su uso.

B.3 Creación de variables de entorno

En caso de no tener configuradas las variables de entorno necesarias para que Maven funcione correctamente, habrá que seguir estos pasos:

1. Acceder a "Este equipo->Configuración avanzada del sistema->variables de entorno

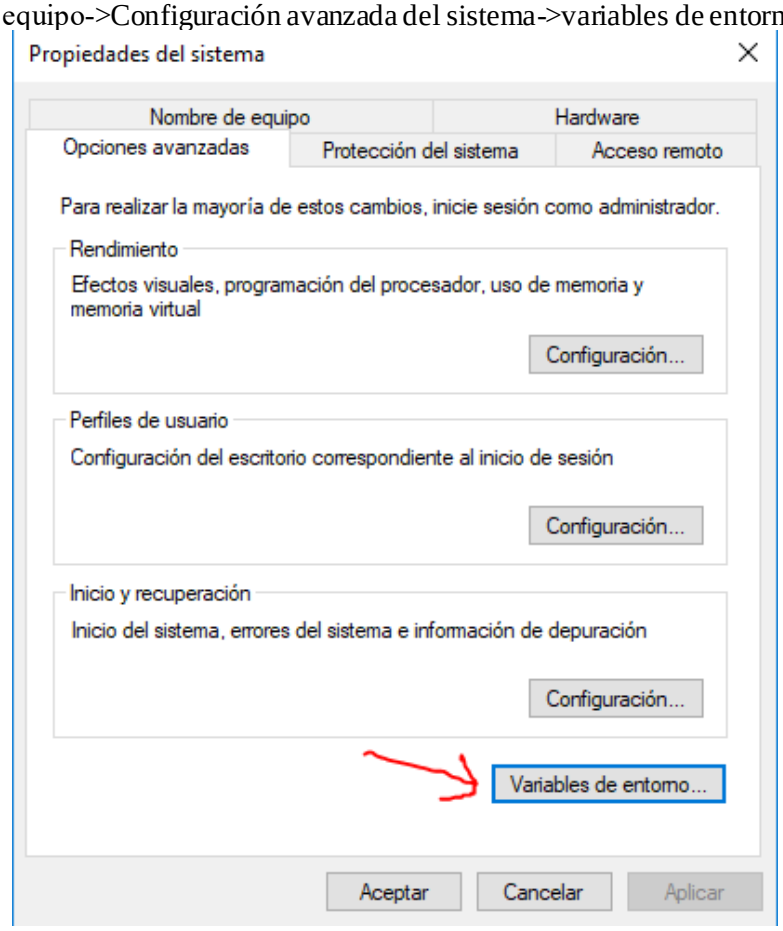


Figura B.3: Variables de entorno

2. Ahora necesitaremos crear la variable de entorno 'JAVA_HOME' que contendrá la ruta que escogimos al instalar el JDK.
3. Incluimos la ruta de 'JAVA_HOME' y la ruta de Maven a la variable de entorno 'Path'. A la ruta de Maven deberá añadirse \bin, es decir, 'tu_ruta_de_maven\bin'.

B.4 Comprobación en cmd e instalación

Abrimos la consola de Windows y ejecutamos el comando "mvn -version". Si la salida es parecida a la siguiente todo está correcto.

```
C:\WINDOWS\system32>mvn -version
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T09:58:13+02:00)
Maven home: C:\Program Files\apache-maven-3.5.2\bin\..
Java version: 1.8.0_161, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_161\jre
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Figura B.4: versión Maven/java

Acto seguido ejecutamos "mvn install" para que se se instale y configure nuestro repositorio local.

B.5 Instalación de Spring Tools Suite (STS)

Descargamos el zip correspondiente de la página de Spring https://spring.io/tools/sts/all

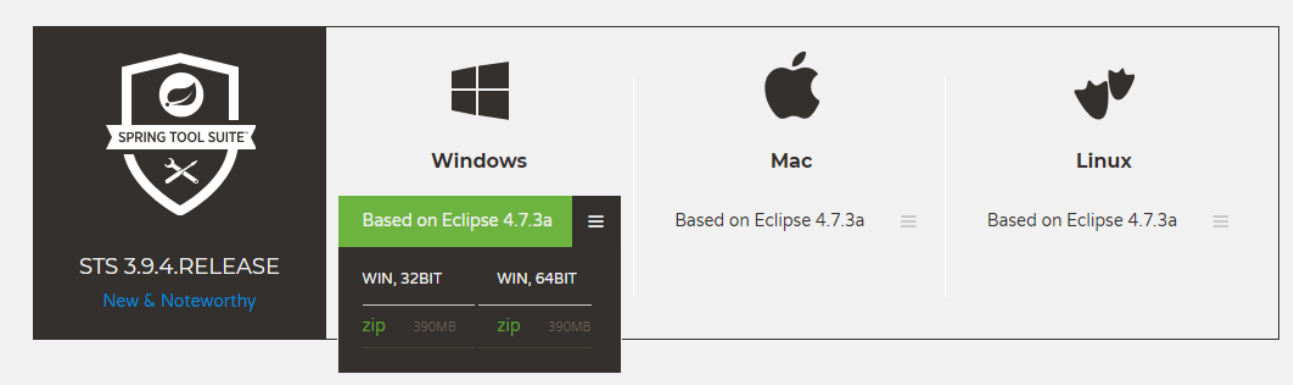


Figura B.5: Descarga STS

Una vez ejecutado e instalado, iniciamos la aplicación.

Figura B.6: Inicio STS

Y por último, escogemos un directorio para el workspace de la aplicación

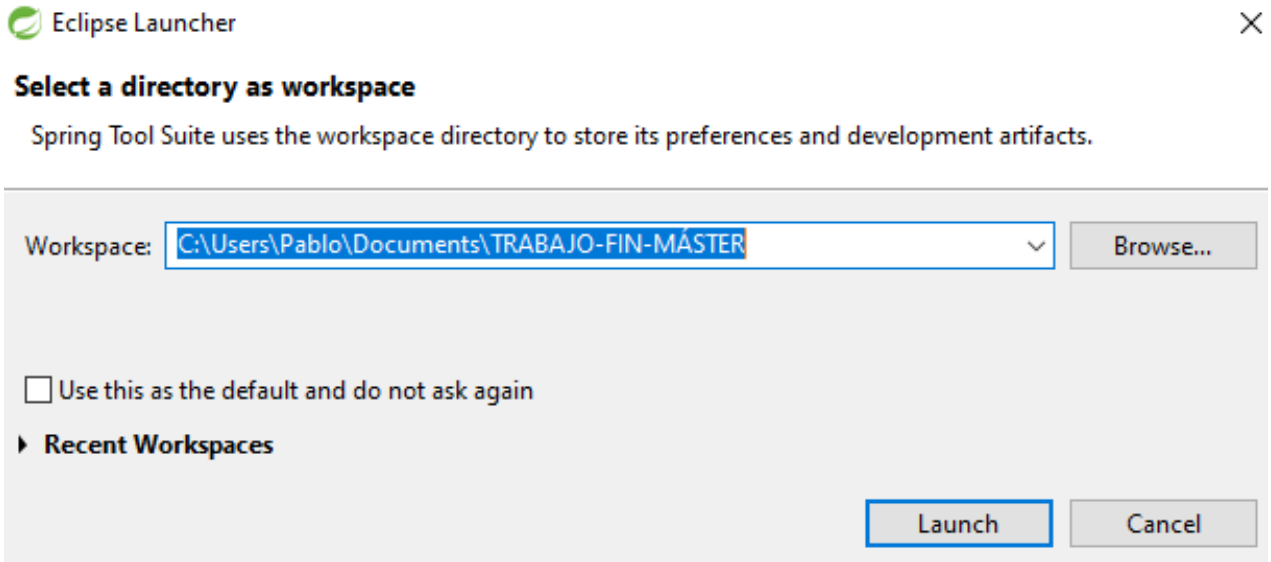


Figura B.7: Workspace

B.6 Instalación de XAMPP

Descargamos XAMPP de la página https://www.apachefriends.org/es/index.html y lo instalamos.

Figura B.8: Descarga de XAMPP

Una vez instalado iniciamos la aplicación y activamos el servidor Mysql.

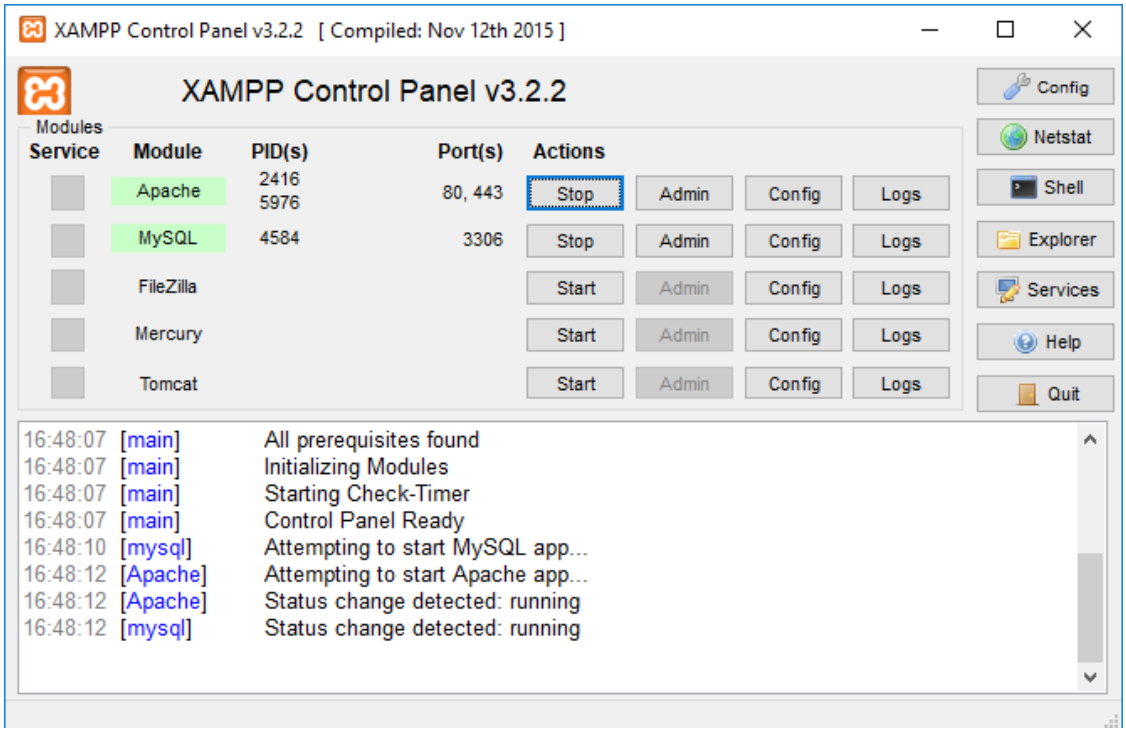


Figura B.9: Interfaz XAMPP

Acto seguido pulsamos sobre “Admin” en la fila de MySQL para dirigirnos a la interfaz gráfica de la base de datos “phpMyAdmin” donde podremos gestionar nuestras bases de datos

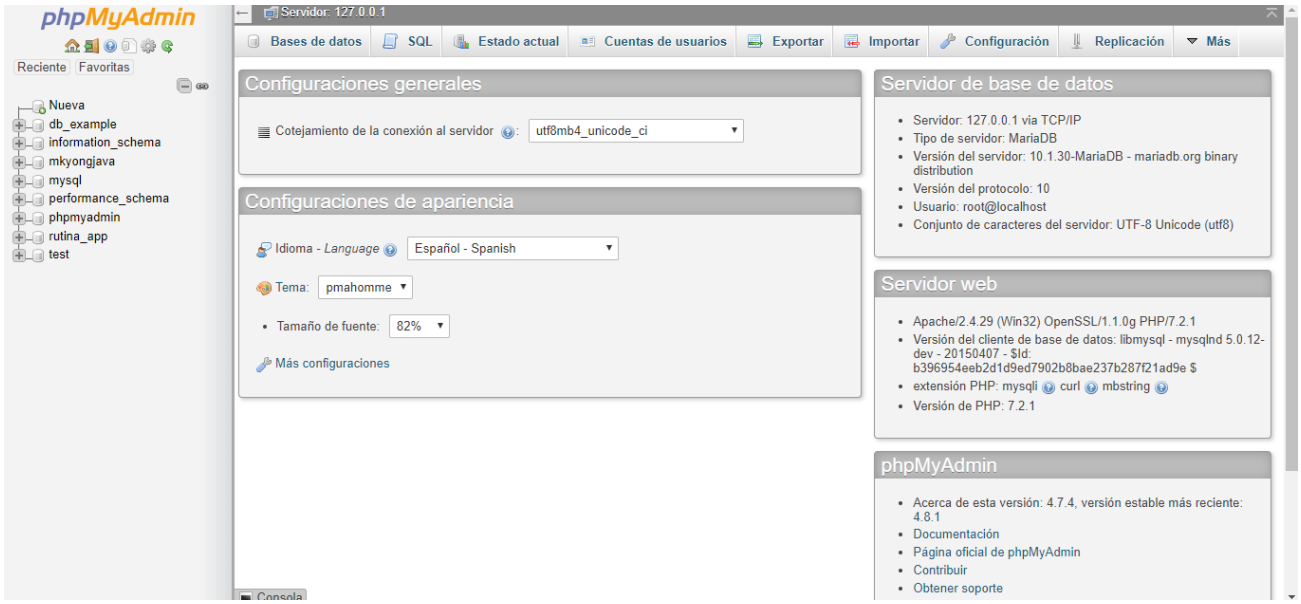


Figura B.10: phpMyAdmin XAMPP

B.7 Instalación Docker Desktop

Figura B.11: Docker Desktop

Es importante destacar que, a la hora de la instalación de Docker Desktop, debemos desmarcar la opción de instalar componentes relativos a WSL2 en la primera página del nstalador.

Además, es necesario tener instalado tanto Ubuntu TLS y el paquete de actualización de paquete del kernel de Linux.

Figura B.12: Ubuntu TLS

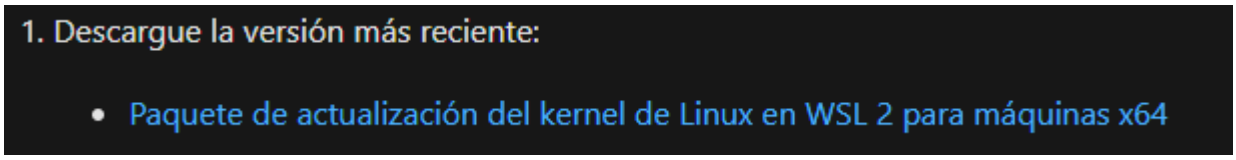


Figura B.13: Ubuntu TLS

B.8 Instalación Visual Studio Code

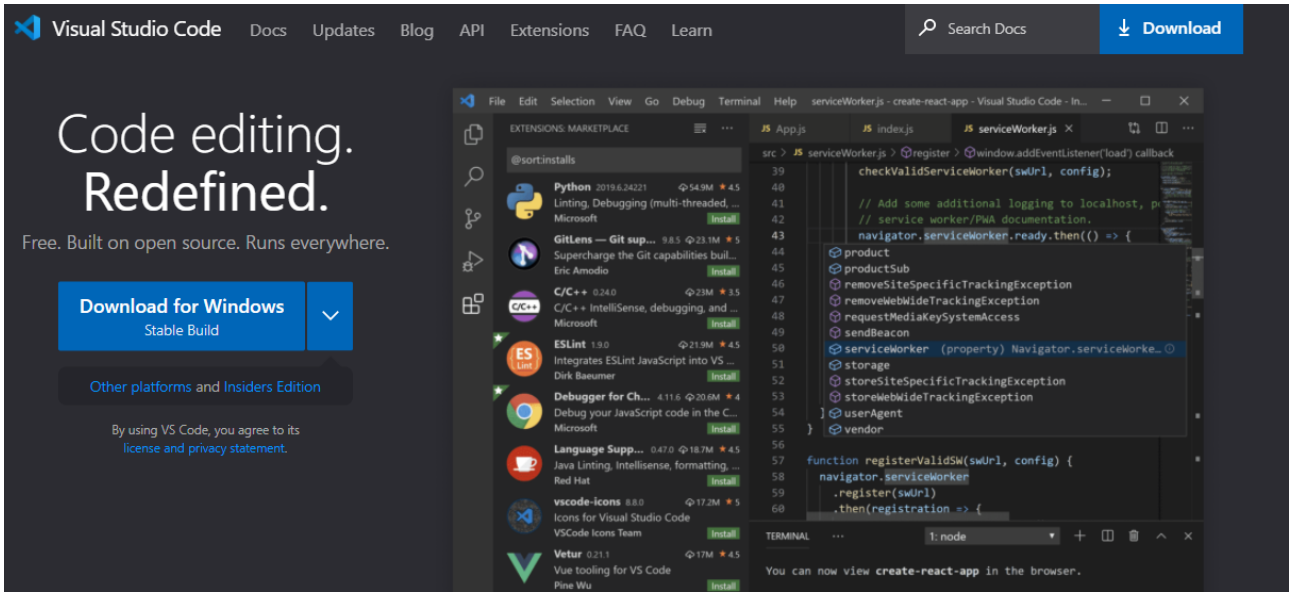


Figura B.14: Visual Studio Code

B.9 Despliegue de la aplicación

I. Para crear la base de datos que utilizará la aplicación web, en la interfaz de phpMyAdmin, elegimos la opción “Importar” y escogemos el archivo ‘rutina_app.sql’.

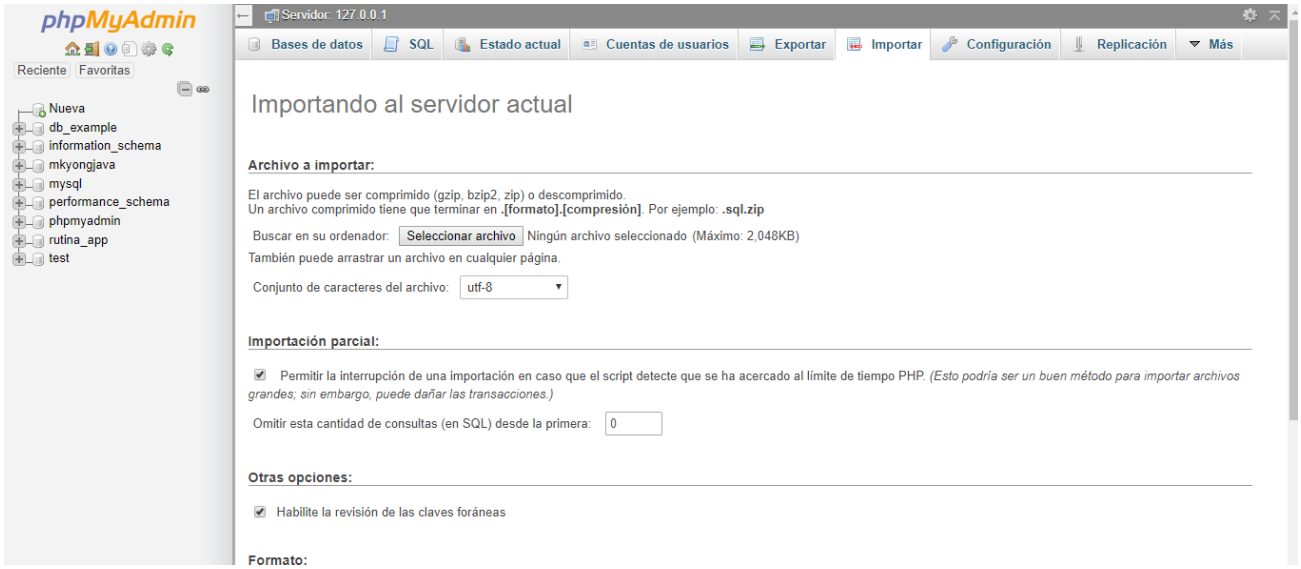


Figura B.15: Importar BD

II. Descomprimimos el archivo 'MyTraining.zip' y colocamos la carpeta 'MyTraining' en el workspace de la aplicación definido en el apartado B.5.

III. Iniciamos STS y seleccionamos la opción 'File->Import->Existing Maven projects'

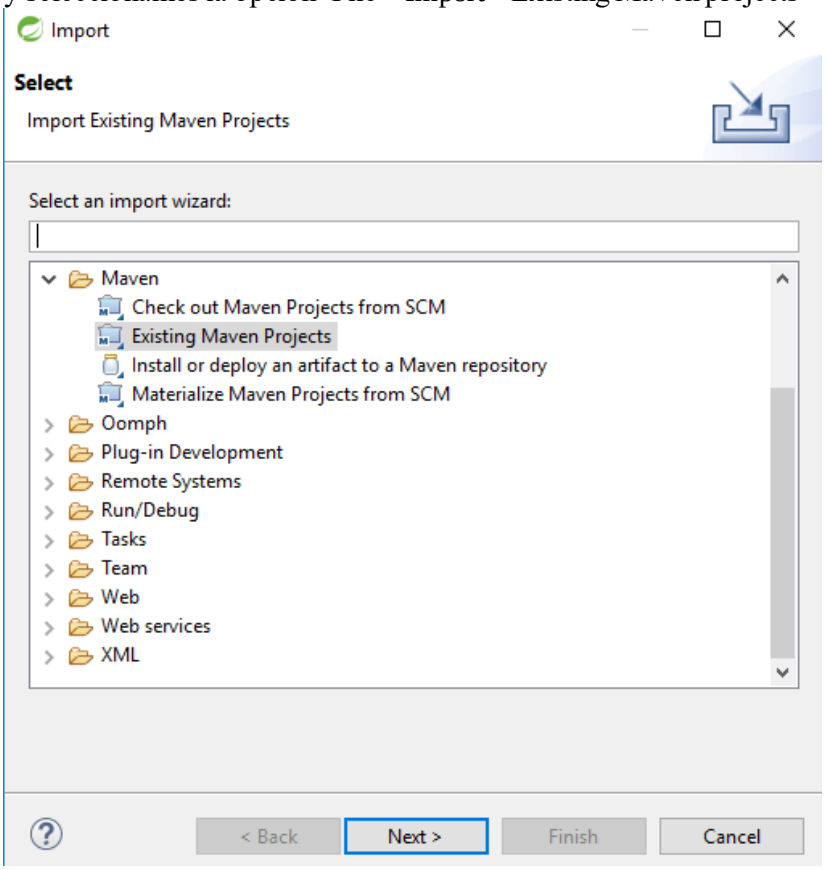


Figura B.16: Importar proyecto

IV. Escogemos como Root Directory la carpeta del proyecto anteriormente colocada en el workspace.

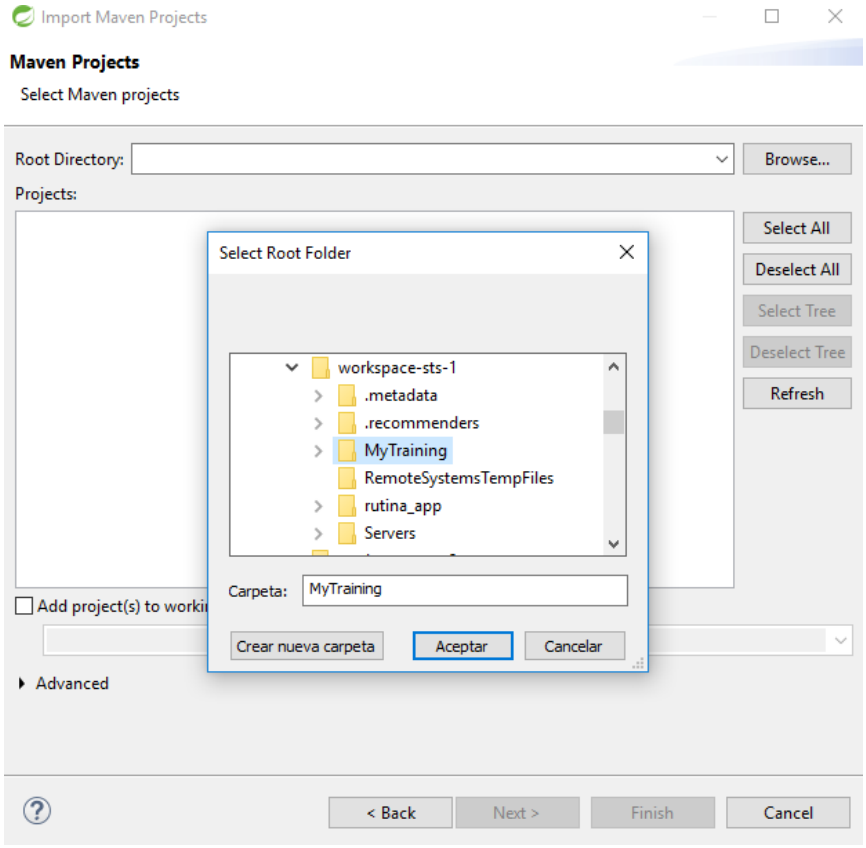


Figura B.17: Root Directoy

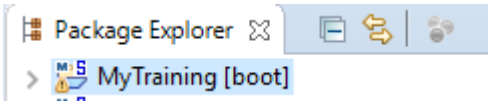
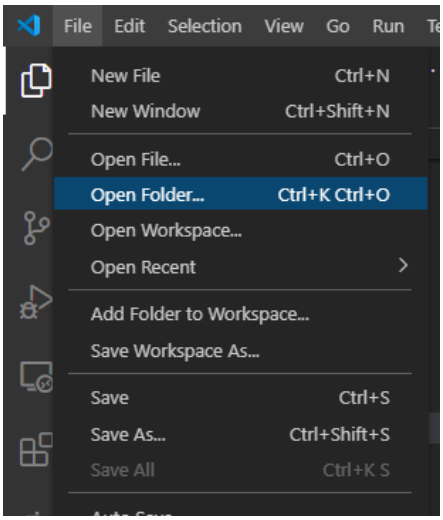
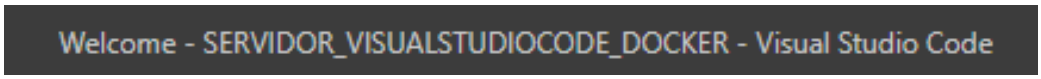


Figura B.18: Proyecto importado

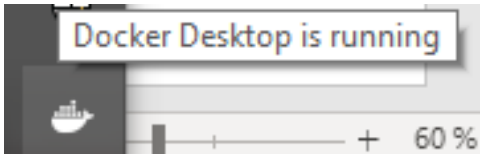
V. Importar el servidor HAPI FHIR en Visual Studio Code



Y escogemos el directorio “SERVIDOR_VISUASTUDIOCODE_DOCKER”



VI. Activar Docker Desktop iniciando la aplicación



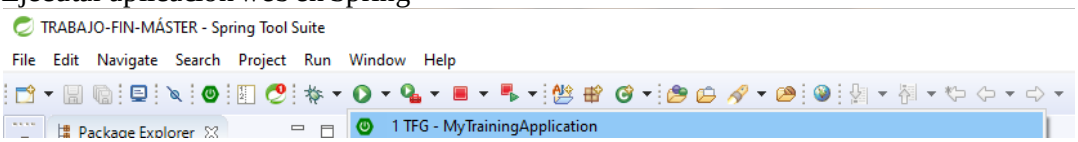
VII. Ejecutar el servidor HAPI FHIR ejecutando en Visual Studio Code lo siguiente

```
PS C:\Users\Pablo\Documents\SERVIDOR_VISUALSTUDIOCODE_DOCKER> CD .\hapi-fhir-jpaserver-starter-master\
PS C:\Users\Pablo\Documents\SERVIDOR_VISUALSTUDIOCODE_DOCKER\hapi-fhir-jpaserver-starter-master> docker-compose up
```

El resultado final debe ser similar al siguiente

```
server_1  | [INFO] Started o.e.j.m.p.JettyWebAppContext@763b0996{/,[file:///tmp/hapi-fhir-jpaserver-starter/src/main/webapp/, file:///tmp/hapi-fhir-jpaserver-starter/target/jetty_overlays/hapi-fhir-testpage-overlay-5_1_0_war/, jar:file:///root/.m2/repository/org/webjars/Eonasdan-bootstrap-datetimepicker/4.17.43/Eonasdan-bootstrap-datetimepicker-4.17.43.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/bower/awesome-bootstrap-checkbox/1.0.1/awesome-bootstrap-checkbox-1.0.1.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/font-awesome/5.8.2/font-awesome-5.8.2.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/jstimezonedetect/1.0.6/jstimezonedetect-1.0.6.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/bower/jquery/3.3.1/jquery-3.3.1.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/jquery/1.11.1/jquery-1.11.1.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/bower/moment/2.15.1/moment-2.15.1.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/bootstrap/3.3.7/bootstrap-3.3.7.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/momentjs/2.10.3/momentjs-2.10.3.jar!/META-INF/resources, jar:file:///root/.m2/repository/org/webjars/select2/4.0.3/select2-4.0.3.jar!/META-INF/resources],AVAILABLE}{file:///tmp/hapi-fhir-jpaserver-starter/src/main/webapp/}
server_1  | [INFO] Started ServerConnector@6281472{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
server_1  | [INFO] Started @40806ms
server_1  | [INFO] Started Jetty Server
```

VIII. Ejecutar aplicación web en Spring




```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.10.RELEASE)

2021-01-12 20:38:15.051  INFO 13560 --- [           main] s.rutina.app.Main.MyTrainingApplication  : Starting MyTrainingApplication on DESKTOP-ITI8DSH
2021-01-12 20:38:15.055  INFO 13560 --- [           main] s.rutina.app.Main.MyTrainingApplication  : No active profile set, falling back to default pro
2021-01-12 20:38:15.106  INFO 13560 --- [           main] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedd
2021-01-12 20:38:15.523  INFO 13560 --- [           main] o.s.b.f.xml.XmlBeanDefinitionReader      : Loading XML bean definitions from class path resou
2021-01-12 20:38:15.598  INFO 13560 --- [           main] o.s.b.f.xml.XmlBeanDefinitionReader      : Loading XML bean definitions from class path resou
2021-01-12 20:38:15.622  INFO 13560 --- [           main] o.s.b.f.xml.XmlBeanDefinitionReader      : Loading XML bean definitions from class path resou
2021-01-12 20:38:15.637  INFO 13560 --- [           main] o.s.b.f.xml.XmlBeanDefinitionReader      : Loading XML bean definitions from class path resou
2021-01-12 20:38:15.651  INFO 13560 --- [           main] o.s.b.f.xml.XmlBeanDefinitionReader      : Loading XML bean definitions from class path resou
2021-01-12 20:38:15.666  INFO 13560 --- [           main] o.s.b.f.xml.XmlBeanDefinitionReader      : Loading XML bean definitions from class path resou
2021-01-12 20:38:15.679  INFO 13560 --- [           main] o.s.b.f.xml.XmlBeanDefinitionReader      : Loading XML bean definitions from class path resou
2021-01-12 20:38:15.690  INFO 13560 --- [           main] o.s.b.f.xml.XmlBeanDefinitionReader      : Loading XML bean definitions from class path resou
```