



Instituto Tecnológico Nacional de México en Celaya



Ing. Sistemas computacionales

Departamento de sistemas y computación

Manual Técnico Consultorio Online

17030658 Ortiz González Naomi

17030622 Otero Martínez Paulina

Tópicos Avanzados de Programación Web

26 de enero de 2021

Contenido

Introducción	3
Objetivo	4
Herramientas	5
Librerías utilizadas	7
Paquetes	9
Frameworks	11
Configuración de la base de datos	12
Diccionario de datos	14
Modelo entidad relación	22
Modelo relacional	22
Descripción de las clases	23
Consultas	38
Descripción de componentes vue	42
Paciente	43
Doctor	46
Usuarios del sistema	49
Pantallas del sistema	51
Descripción de módulos	69

Introducción

El presente manual técnico ha sido desarrollado con la finalidad de presentar el sistema desde un punto de vista técnico, familiarizando al personal encargado en las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

El sistema posee una base de datos cuya estructura se muestra en el Diagrama de Entidad-Relación y el Diagrama Relacional. Adicionalmente la base de datos se encuentra descrita en el Diccionario de Datos donde se hace descripción de todas las tablas y sus atributos correspondientes.

El manual técnico facilita las tareas de instalación del sistema, así como el servicio asociado a este, dándole al responsable del sistema las herramientas necesarias que le permitan cumplir de manera eficiente las tareas de configuración del sistema. Se muestra un marco conceptual para el desarrollo del sistema, familiarizado a los analistas y programadores a la estructura del sistema.

Objetivo

General

La finalidad del manual técnico es la de capacitar al personal con acceso al sistema para la aplicación correcta del sistema de información

Específicos

- Representar la funcionalidad técnica de la estructura, diseño y definición del sistema.
- Definir claramente el procedimiento de instalación del aplicativo.
- Detallar la especificación de los requerimientos de Hardware y Software necesarios para la instalación de la aplicación.
- Describir las herramientas utilizadas para el diseño y desarrollo del prototipo

Herramientas

MySQL

Es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales, utiliza múltiples tablas para almacenar y organizar la información. Está basado en lenguaje de consulta estructurado (SQL).

Comenzó como una iniciativa de Software Libre y continúa ofreciéndose como tal, para usuarios particulares. Si se desea utilizarlo para promover datos en una empresa, puede comprarse una licencia, como un software propietario, autoría de Oracle Corporation.

MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción lenguajes de programación como PHP, Perl y Java y su integración en distintos sistemas operativos.

Para el desarrollo del proyecto utilizamos MySQL Community Server en su versión 5.7.31.

Requerimientos

Windows

- Soporte para protocolo TCP/IP.
- Una copia de la distribución binaria de MySQL para Windows, que se puede descargar de <http://dev.mysql.com/downloads/>.
- Una herramienta capaz de leer ficheros .zip, para descomprimir el fichero de distribución.
- Suficiente espacio en disco rígido para descomprimir, instalar, y crear las bases de datos de acuerdo a sus requisitos. Generalmente se recomienda un mínimo de 200 megabytes.

Linux

La manera recomendada de instalar MySQL en Linux es utilizando paquetes RPM, originalmente pensado para Red Hat Linux, sin embargo, en la actualidad muchas

distribuciones GNU/Linux lo usan, dentro de las cuales las más destacadas son Fedora, Mandriva, Mageia, PCLinuxOS, openSUSE, SuSE Linux.

También, puede usar el repositorio APT de MySQL, que proporciona paquetes deb para instalar y administrar MySQL, en Debian y Ubuntu.

En la mayoría de los casos, sólo será necesario instalar los paquetes MySQL-server y MySQL-client para conseguir una instalación de MySQL en funcionamiento.

Symfony

Symfony es un framework para construir aplicaciones web con PHP, basado en el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas partes que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos. La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista.

Además proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. También automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Su licencia es de tipo software libre; ha sido desarrollado por una empresa francesa llamada Sensio Labs, completamente en PHP 5.3. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. Para el desarrollo de nuestro sitio se trabajó con la versión 5.2.1.

Requerimientos

- Servidor web (Apache, por ejemplo),

- Motor de base de datos (MySQL, PostgreSQL, SQLite, o cualquier motor de base de datos compatible con PDO)
- PHP 7.2.5 o superior
- Interfaz de línea de comandos.
- Instalar Composer , utilizado para instalar paquetes PHP.

Vue

Vue es un framework de open source de JavaScript, el cual permite construir interfaces de usuarios de una forma muy sencilla, fue creado por Evan You ex trabajador de Google. Una de las características más importantes de Vue es el trabajo con componentes. Un componente Vue, es un elemento el cual se encapsula código reutilizable. Dentro de un componente, pueden encontrarse etiquetas HTML, estilos de CSS y código JavaScript. Esto permite desarrollar proyectos modularizados y fáciles de escalar. Utilizamos Vue.js en su versión 2.6.11.

NPM es el método de instalación recomendado para construir aplicaciones con Vue.

Librerías utilizadas

jsPDF

Solución de Javascript para la generación de archivos PDF. Utilizado para emitir los reportes de recetas, facturas y productos comprados.

Instalación

```
npm install jspdf --save
```



woocommerce/woocommerce-rest-api

Librería de JavaScript para la API REST de WooCommerce, compatible con CommonJS (CJS) y Embedded System Module (ESM).

Instalación

```
npm install --save @woocommerce/woocommerce-rest-api
```



Aplicación móvil

GSON

Librería de código abierto creada por Google que permite serializar objetos Java para convertirlos en un String. Su uso más frecuente es para convertir un objeto en su representación JSON y a la inversa.

La gran ventaja de esta librería es que puede ser usada sobre objetos de cualquier tipo de clases, incluso clases preexistentes que no se hayan creado. Esto es posible al no ser necesario introducir código en las clases para que sean serializadas.

Instalación

Agregar la siguiente dependencia al archivo build.gradle del proyecto:

```
implementation 'com.google.code.gson:gson:2.8.6'
```



Volley

Volley es una librería HTTP que facilita y agiliza el uso de redes en apps para Android, disponible en GitHub. Volley ofrece los siguientes beneficios:

- Programación automática de solicitudes de red
- Varias conexiones de red simultáneas
- Almacenamiento de respuestas en caché y en disco transparentes con coherencia de caché en HTTP estándar
- Compatibilidad con la priorización de solicitudes
- API de cancelación de solicitudes (permite cancelar una única solicitud, o bien establecer bloques o grupos de solicitudes para cancelar)
- Personalización sencilla, por ejemplo, de reintentos o retiradas

- Ordenamiento sólido que permite completar correctamente la IU con datos recuperados de forma asíncrona de la red
- Herramientas de depuración y rastreo

Instalación

Agregar la siguiente dependencia al archivo build.gradle del proyecto:

```
implementation 'com.android.volley:volley:1.1.1'
```



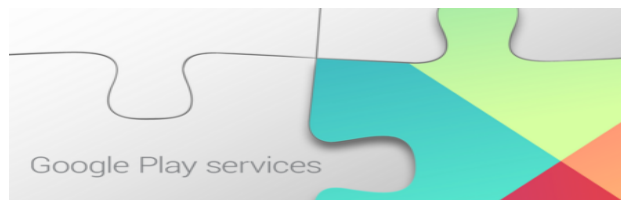
play-services-maps

Utilizada para mostrar el mapa del mundo que muestra la ubicación de los casos COVID-19 confirmados alrededor del mundo.

Instalación

Agregar la siguiente dependencia al archivo build.gradle del proyecto:

```
implementation 'com.google.android.gms:play-services-maps:17.0.0'
```



Paquetes

vue2-google-maps

Utilizado para mostrar la información correspondiente a los casos de coronavirus en el mundo. Permite la utilización de marcadores para señalar una ubicación con base en sus coordenadas latitud y longitud.

Installation

```
npm install vue2-google-maps
```

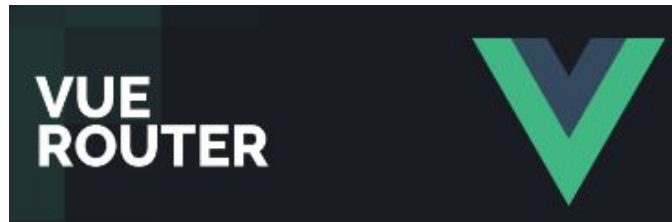
vue-router

Router oficial Vue.js . Se integra con el núcleo de Vue.js para facilitar la creación de aplicaciones de una sola página con Vue. Las características incluyen:

- Mapeo de rutas / vistas anidadas
- Configuración de enrutador modular basada en componentes
- Parámetros de ruta, consulta, comodines
- Control de navegación
- Enlaces con clases CSS activas automáticas

Instalación

```
npm install vue-router
```



svg

Crea elementos svg a partir de una cadena. Empleado para la visualización de los casos de coronavirus en la República Mexicana. Consume una cadena JSON conformada por el Id del estado, nombre, y un sistema de coordenadas para la generación de la ilustración.

Instalación

```
npm install svg
```

js-cookie

Permite almacenar información en cookies, es decir en el archivo creado por un sitio web que contiene pequeñas cantidades de datos y que se envían entre un emisor y un receptor. Usado con el propósito principal de identificar al usuario almacenando su historial de actividad en un sitio web específico.

Instalación

```
npm i js-cookie
```



AXIOS

Cliente HTTP basado en promesas para el navegador, preparado para consumir APIs REST. Nos permite realizar solicitudes contra un servidor y recibir la respuesta de una manera sencilla de procesar.

Instalación

```
npm install axios
```



Bootstrap

Kit de herramientas de código abierto de front-end utilizado para el diseño y personalización del sitio.

Instalación

```
npm install bootstrap@next
```



Frameworks

Symfony

Framework de PHP diseñado para desarrollar aplicaciones web. Está basado en el patrón Modelo Vista Controlador. Es ampliamente utilizado en la creación de APIs. Para poder trabajar con esta herramienta es necesario disponer una consola de comandos del sistema operativo y cualquier versión de PHP5 o PHP7.

Creación nuevo proyecto Symfony

```
symfony new my_project_name
```



Vue.js

Vue.js es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página. Se basa en la creación de componentes, los cuales son elementos en los que se encapsula código reutilizable. Dentro de un componente podemos encontrar etiquetas HTML, estilos de CSS y código JavaScript.

Creación de nuevo proyecto Vue

```
npm install -g @vue/cli  
vue create new_project  
npm run serve
```



Vuetify

Framework de Material Design para crear interfaces de usuario de Vue. Proporciona gran cantidad de componentes listos para usar, configurables con distintas opciones para cambiar su estética y comportamiento.

Una vez que se tiene un proyecto de vue instanciado, es posible agregar el paquete Vuetify mediante el siguiente comando:

```
vue add vuetify
```



Configuración de la base de datos

Symfony proporciona las herramientas que necesitas para usar bases de datos mediante Doctrine , un conjunto de librerías PHP enfocadas principalmente en el almacenamiento de bases de datos y mapeo de objetos. Estas herramientas admiten bases de datos relacionales como MySQL, con la que estaremos trabajando.

La información de conexión de la base de datos se almacena como una variable de entorno llamada DATABASE_URL, la cual puede ser configurada desde dentro del archivo .env

Se deberá especificar el nombre de usuario creado previamente para la base de datos, la contraseña, nombre de host y nombre de la base de datos.

```
# DATABASE_URL="sqlite:///kernel.project_dir%/var/data.db"
DATABASE_URL="mysql://admin:123456@127.0.0.1:3306/consultas_online?serverVersion=5.7"
#DATABASE_URL="postgresql://db_user:db_password@127.0.0.1:5432/db_name?serverVersion=13&charset=utf8"
###< doctrine/doctrine-bundle ###
```

Una vez configurados los parámetros de conexión, Doctrine puede crea la base de datos por nosotros a través de comando:

```
php bin/console doctrine:database:create
```

Para la creación de las tablas usamos el comando `make:entity` que genera la clase de la entidad y cualquier campo que necesite. El comando le hará algunas preguntas: nombre de la entidad, campos y su tipo de dato. Además podemos hacer uso de los tipos `ManyToOne`, `OneToMany` y `ManyToMany` que permiten mapear una entidad en la base de datos con una columna de clave externa, es decir permiten asociar las tablas a través de llaves foráneas.

Una vez que están completamente configuradas las clases y listas para guardar en su tabla correspondiente. Debemos ejecutar el siguiente comando para añadirlas a nuestra base de datos de MySQL.

```
php bin/console make:migration
```

Se generará un archivo de migración, si lo abrimos visualizamos que contiene el SQL necesario para actualizar nuestra base de datos. Para ejecutar ese SQL, debemos ejecutar las migraciones usando:

```
php bin/console doctrine:migrations:migrate
```

Este comando ejecuta todos los archivos de migración que aún no se han ejecutado en la base de datos.

```
Tables_in_consultas_online
+-----+
| allergy
| bill
| chronic_disease
| doctor
| doctor_service
| doctrine_migration_versions
| media
| medical_consultation
| medicine
| patient
| patient_allergy
| patient_disease
| patient_surgery
| payment
| prescription
| service
| speciality
| surgery
| tax_data
| ticket
| userdata
+-----+
```

Diccionario de datos

Nombre de la tabla		allergy		
Descripción	Tabla que contiene la descripción de las alergias			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la alergia
description	VARCHAR		100	Descripción de la alergia

Nombre de la tabla		bill		
Descripción	Tabla que mantiene la relación entre una consulta médica y su correspondiente factura			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la alergia
id_consultation	INT	FK		Identificador de la

				consulta
filename	VARCHAR		30	Nombre con el que se almacena el archivo de factura

Nombre de la tabla		chronic_disease		
Descripción	Tabla que contiene la descripción de las enfermedades crónicas			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la enfermedad
description	VARCHAR		100	Descripción de la enfermedad

Nombre de la tabla		consultation_service		
Descripción	Tabla que mantiene la relación entre una consulta médica, los servicios de dicha consulta y el doctor que la atendió			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la alergia
id_consultation	INT	FK		Identificador de la consulta
id_service	INT	FK		Identificador del servicio brindado
id_doctor	INT	FK		Identificador del doctor que atendió.

Nombre de la tabla		media		
Descripción	Tabla que almacena los recursos visuales que se han subido de una consulta			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción

id	INT	PK		Identificador del recurso
id_consultation	INT	FK		Identificador de la consulta
filename	VARCHAR		30	Nombre con el que se almacena el recurso

Nombre de la tabla		medical_consultation		
Descripción	Tabla que almacena los datos de una consulta médica			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la consulta médica
id_patient	INT	FK		Identificador del paciente que solicita la consulta
symptom	VARCHAR		200	Descripción de los síntomas
attention_status	INT			Indica estatus de atención. Atendida =1, no atendida=2
consultation_date	DATE			Fecha de la consulta
id_doctor	INT	FK		Identificador del doctor asignado a la consulta.

Nombre de la tabla		medicine		
Descripción	Tabla que almacena la información de los medicamentos			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador del medicamento
name	VARCHAR		50	Nombre del medicamento

type	VARCHAR		50	Tipo de medicamento
substance	VARCHAR		50	Sustancia activa
laboratory	VARCHAR		50	Nombre del laboratorio
cost	DECIMAL		(10,2)	Costo del medicamento

Nombre de la tabla		patient		
Descripción	Tabla que almacena la información de los pacientes			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador del paciente
name	VARCHAR		50	Nombre del paciente
lastname	VARCHAR		50	Apellidos del paciente
address	VARCHAR		100	Dirección del paciente
city	VARCHAR		100	Ciudad
state	VARCHAR		100	Estado
country	VARCHAR		100	País
birthdate	DATE			Fecha de nacimiento
phone	VARCHAR		10	Teléfono del paciente
email	VARCHAR		100	Correo electrónico del paciente
id_user	INT	FK		Identificador de usuario
status_covid	VARCHAR		10	“Negativo”, “Sospechoso”, “Confirmado”
latitud	FLOAT			Latitud de la ubicación

				del paciente
longitud	FLOAT			Longitud de la ubicación del paciente

Nombre de la tabla		patient_allergy		
Descripción	Tabla que mantiene la relación entre un paciente y sus alergias.			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la alergia
id_patient	INT	FK		Identificador del paciente
id_allergy	INT	FK		Identificador de la alergia

Nombre de la tabla		patient_disease		
Descripción	Tabla que mantiene la relación entre un paciente y sus enfermedades crónicas			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la alergia
id_patient	INT	FK		Identificador del paciente
id_disease	INT	FK		Identificador de la enfermedad

Nombre de la tabla		patient_surgery		
Descripción	Tabla que mantiene la relación entre un paciente y sus cirugías			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la

				alergia
id_patient	INT	FK		Identificador del paciente
id_surgery	INT	FK		Identificador de la cirugía

Nombre de la tabla		payment		
Descripción	Tabla que almacena los tipos de pago disponibles			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador del tipo de pago
description	VARCHAR			Descripción del tipo de pago

Nombre de la tabla		prescription		
Descripción	Tabla que almacena los medicamentos e instrucciones que conformarán una receta			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador del tipo de pago
id_consultation	INT	FK		Identificador de la consulta médica
id_medicine	INT	FK		Identificador del medicamento
instructions	LONGTEXT			Instrucciones para ese medicamento

Nombre de la tabla		service		
Descripción	Tabla que almacena los servicios disponibles			
Nombre del	Tipo de dato	Integridad	Longitud	Descripción

campo				
id	INT	PK		Identificador del servicio
description	VARCHAR		100	Descripción del servicio

Nombre de la tabla		speciality		
Descripción	Tabla que almacena las especialidades médicas			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la especialidad
description	VARCHAR		100	Descripción de la especialidad

Nombre de la tabla		surgery		
Descripción	Tabla que almacena las cirugías			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la cirugía
description	VARCHAR		100	Descripción de la cirugía

Nombre de la tabla		tax_data		
Descripción	Tabla que almacena los datos de facturación de un paciente			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador de la cirugía
id_patient	INT	FK		Descripción de la

				cirugía
billing_address	VARCHAR		100	Domicilio de facturación
shipping_address	DATE			Dirección de envío
id_payment	INT	FK		Identificador tipo de pago

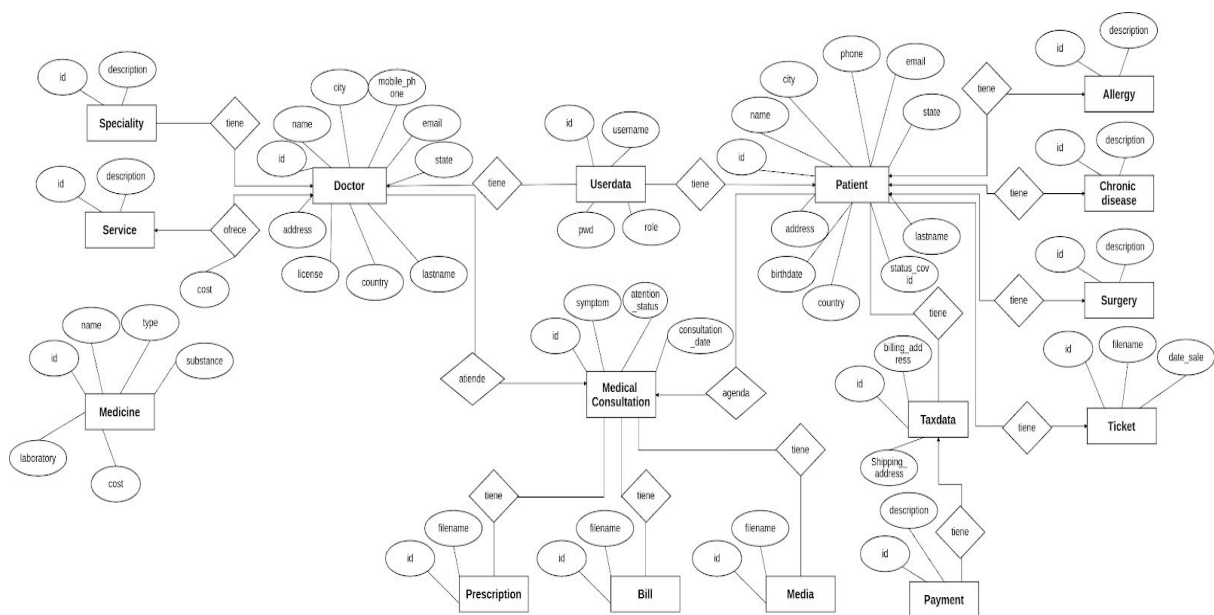
Nombre de la tabla		ticket		
Descripción	Tabla que almacena los tickets generados por las compras de medicamentos			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador del ticket
id_patient_id	INT	FK	100	Identificador del paciente que generó la compra
date_sale	DATE			Fecha de la venta
filename	VARCHAR		30	Nombre del archivo del ticket

Nombre de la tabla		userdata		
Descripción	Tabla que almacena los tickets generados por las compras de medicamentos			
Nombre del campo	Tipo de dato	Integridad	Longitud	Descripción
id	INT	PK		Identificador del usuario
username	VARCHAR		50	Nombre de usuario
pwd	VARCHAR		32	Contraseña

role	INT			Rol del usuario: 1 = Doctor 2 = Paciente
------	-----	--	--	------------------------------------------------

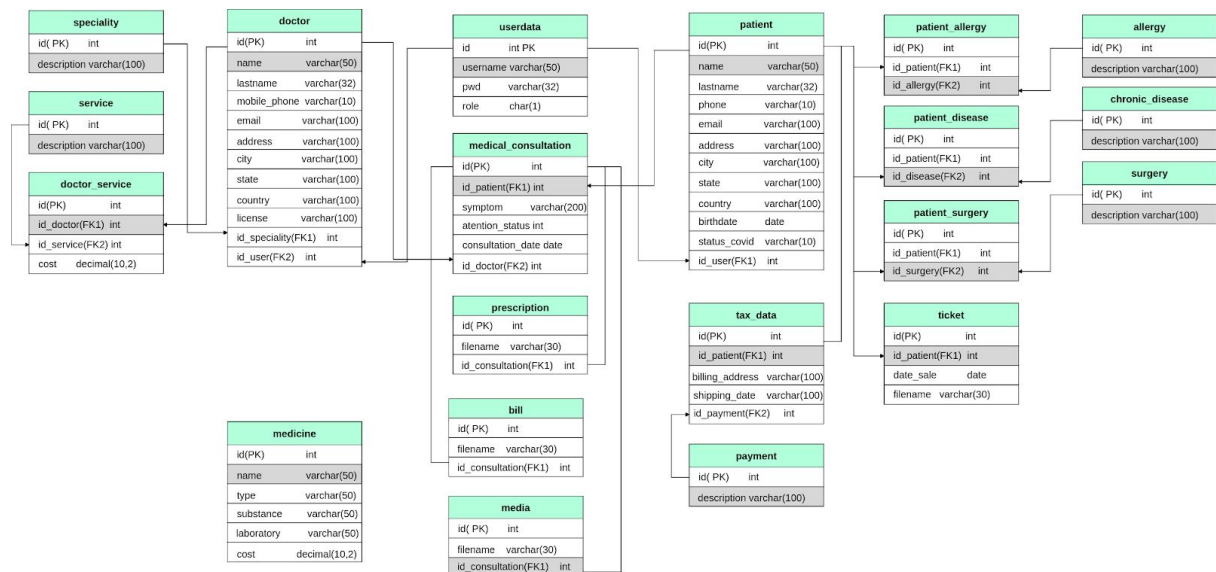
Modelo entidad relación

El modelo entidad relación ilustra cómo una colección de objetos básicos llamados entidades se relacionan entre sí dentro de un sistema. En nuestro sistema la estructura lógica de la base de datos se expresa de la siguiente manera:



Modelo relacional

Se trata de un esquema de organización y gestión de bases de datos, el cual consiste en el almacenamiento de datos en tablas compuestas por filas y columnas o campos, las tuplas se identifican de manera única mediante claves. Se debe garantizar que los datos almacenados en una estructura del modelo relacional sean correctos, para lo cual existen las reglas de integridad.



Descripción de las clases

Doctrine en vez de trabajar con filas y tablas, permite guardar y obtener objetos enteros a partir de la información de la base de datos. Para ello se encarga de una clase PHP a una tabla de la base de datos y después, mapear las propiedades de la clase PHP a las columnas de esa tabla. Dichas clase se identifican como un Entity.

Por otro lado, un controller es un callable PHP creado que recibe la información del HTTP request y crea y devuelve una HTTP response (devuelto como un objeto Response). La respuesta podría ser una página HTML, un documento XML, un array serializado JSON, una imagen, una redirección, un error 404 o cualquier otra cosa. El controller contiene cualquier lógica arbitraria que la aplicación necesita para renderizar el contenido de una página. El objetivo de un controller es siempre el mismo: crear y devolver un objeto Response. En el proceso se puede leer información del request, cargar una base de datos, enviar un email, o establecer información de la sesión de usuario. Pero en todos los casos el controller eventualmente devolverá un objeto Response, que se enviará al cliente.

Allergy

- Clase Entity: Allergy

En esta clase se definen los atributos y relaciones correspondientes a las diferentes alergias, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: AllergyController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Obtener el listado de todas las alergias

```
* @Route("/allergy", name="getAllAllergies", methods={"GET"})
```

→ Obtener una alergia específica con base en su id.

```
*@Route("/allergy/{id}", name="getOneAllergy", methods={"GET"})
```

Bill

- Clase Entity: Bill

En esta clase se definen los atributos y relaciones correspondientes a las facturas, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: BillController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla bill.

```
* @Route("/bill", name="bill", methods={"POST"})
```

→ Obtener el listado de todas las facturas.

```
* @Route("/bill", name="getAllBill", methods={"GET"})
```

→ Obtener una factura específica con base en su id.

```
* @Route("/bill/{id}", name="getOneBill", methods={"GET"})
```

→ Obtener todas las facturas de un paciente determinado con base en el id del paciente.

```
*@Route("/patient/{id}/bills",name="getBillPatient",  
methods={"GET"})
```

→ Obtener todas las facturas emitidas por un doctor determinado con base en el id del doctor.

```
*@Route("/doctor/{id}/bills",name="getBillByDoctor",  
methods={"GET"})
```


- Obtener el nombre del archivo de la factura de una consulta específica con base en el id de la consulta.

```
*@Route("/bill/consultation/{id}", name="getOneBillByIdConsultation", methods={"GET"})
```

Chronic Disease

- Clase Entity: ChronicDisease

En esta clase se definen los atributos y relaciones correspondientes a las diferentes enfermedades crónicas, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: ChronicDiseaseController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

- Obtener el listado de todas las enfermedades crónicas.

```
* @Route("/chronicdisease", name="getAllChronicDiseases", methods={"GET"})
```

- Obtener una enfermedad específica con base en su id.

```
* @Route("/chronicdisease/{id}", name="getOneChronicDisease", methods={"GET"})
```

Doctor

- Clase Entity: Doctor

En esta clase se definen los atributos y relaciones correspondientes a los doctores, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: DoctorController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

- Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla doctor.

```
* @Route("/doctor", name="doctor", methods={"POST"})
```

- Obtener el listado de todos los médicos.

```
* @Route("/doctor", name="getAllDoctors", methods={"GET"})
```

- Obtener un doctor específico con base en su id.

```
* @Route("/doctor/{id}", name="getOneDoctor", methods={"GET"})
```

- Obtener todos los doctores que posean una especialidad determinada con base en el id de la especialidad.

```
* @Route("/doctor/speciality/{id}", name="getDoctorSpeciality",  
methods={"GET"})
```

- Obtener los datos de un doctor determinado con base en su id de usuario.

```
*@Route("/doctor/user/{id}", name="getDoctorById",  
methods={"GET"})
```

Doctor Service

- Clase Entity: DoctorService

En esta clase se definen los atributos y relaciones correspondientes a los servicios brindados por cada doctor, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: DoctorServiceController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

- Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla doctor_service.

```
*@Route("/doctorService", name="doctorService",  
methods={"POST"})
```

- Obtener el listado de todos los registros de la tabla doctor_service.

```
*@Route("/doctorService", name="getAllDoctorServices",  
methods={"GET"})
```

- Obtener un registro de la tablas doctor_service específico con base en su id.

```
@Route("/doctorService/{id}", name="getOneDS", methods={"GET"})
```

- Recuperar la información enviada por el cliente para modificar el costo de un servicio brindado por un doctor en específico.

```
*@Route("/doctorService/edit/{id}", name="updateDoctorService",  
methods={"PUT"})
```

- Eliminar un servicio brindado por un doctor

```
*@Route("/doctorService/{id}", name="deleteDoctorService",  
methods={"DELETE"})
```

→ Obtener los servicios brindados por un doctor

```
* @Route("/services/doctor/{id}", name="getServicesByDoctor",  
methods={"GET"})
```

→ Obtener un servicio específico brindado por un determinado doctor

```
*@Route("/service/{id_service}/doctor/{id_doctor}",  
name="getDoctorServiceByIds", methods={"GET"})
```

Email

- Clase Controller: EmailController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para enviar un correo electrónico al paciente con el archivo correspondiente a su ticket de alguna compra.

```
*@Route("/email/ticket", name="sendTicketEmail",  
methods={"POST"})
```

→ Recuperar la información enviada por el cliente para enviar un correo electrónico al paciente con el archivo correspondiente a la receta de alguna de sus consultas.

```
* @Route("/email/prescription", name="sendPrescriptionEmail",  
methods={"POST"})
```

→ Recuperar la información enviada por el cliente para añadir enviar un correo electrónico al paciente con el archivo correspondiente a la factura de alguna de sus consultas.

```
* @Route("/email/bill", name="sendBillEmail", methods={"POST"})
```

Media

- Clase Entity: Media

En esta clase se definen los atributos y relaciones correspondientes a las imágenes, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: MediaController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla media.

```
* @Route("/media", name="media", methods={"POST"})
```

→ Obtener el listado de todos los registros de la tabla media.

```
* @Route("/media", name="getAllMedias", methods={"GET"})
```

→ Obtener un registro de la tabla media con base en su id.

```
* @Route("/media/{id}", name="getOneMedia", methods={"GET"})
```

→ Obtener las imágenes asociadas a una consulta médica específica con base en el id de consulta.

```
*@Route("/media/consultation/{id}",name="getMediaByConsultation", methods={"GET"})
```

Medical Consultation

- Clase Entity: MedicalConsultation

En esta clase se definen los atributos y relaciones correspondientes a las consultas médicas, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: MedicalConsultationController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla medical_consultation.

```
*@Route("/medicalconsultation", name="medicalconsultation", methods={"POST"})
```

→ Obtener el listado de todas las consultas médicas registradas.

```
*@Route("/medicalconsultation",name="getAllMedicalConsultations", methods={"GET"})
```

→ Obtener la información de una consulta médica con base en su id.

```
*@Route("/medicalconsultation/{id}",name="getOneMedicalConsultation", methods={"GET"})
```

→ Obtener el id de la última consulta médica registrada en la tabla.

```
*@Route("/consultation/last",name="getLastMedicalConsultation", methods={"GET"})
```

→ Obtener el listado de las consultas de un paciente específico.

```
*@Route("/medicalconsultation/patient/{id}",name="getMedicalConsultationPatient", methods={"GET"})
```

→ Obtener el listado de las consultas de un paciente específico.

```
*@Route("/medicalconsultation/patient/{id}",name="getMedicalConsultationPatient", methods={"GET"})
```

→ Obtener el listado de las consultas atendidas por un doctor específico.

```
*@Route("/medicalconsultation/doctor/{id}",name="getMedicalConsultationByDoctor", methods={"GET"})
```

→ Obtener el listado de las consultas registradas pertenecientes a una especialidad determinada.

```
*@Route("/consultation/speciality/{id}",name="getConsultationsBySpeciality", methods={"GET"})
```

→ Recuperar la información recibida del cliente para modificar el id del doctor de una consulta específica.

```
*@Route("/medicalconsultation/update/{id}/{id_doctor}",name="updateMedicalConsultation", methods={"POST"})
```

→ Recuperar la información recibida del cliente para modificar el estatus de atención de una consulta específica.

```
*@Route("/medicalconsultation/updatestatus/{id}/{status}",name="updateStatus", methods={"PUT"})
```

Medicine

- Clase Entity: Medicine

En esta clase se definen los atributos y relaciones correspondientes a los diferentes medicamentos, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: MedicineController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Obtener el listado de todos los medicamentos.

```
* @Route("/medicine", name="getAllMedicine", methods={"GET"})
```

→ Obtener un medicamento específico con base en su id.

```
*@Route("/medicine/{id}",name="getOneMedicine",methods={"GET"})
```

Patient Allergy

- Clase Entity: PatientAllergy

En esta clase se definen los atributos y relaciones correspondientes a las alergias de cada paciente, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: PatientAllergyController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla patient_allergy.

```
*@Route("/patientallergy",name="patientAllergy",methods={"POST"},
```

→ Obtener el listado de todos los registros de la tabla patient_allergy.

```
*@Route("/patientallergy",name="patientAllergy",methods={"POST"},
```

→ Obtener un registro de la tabla patient_allergy con base en su id.

```
*@Route("/patientallergy",name="getAllPatientAllergy",methods={"GET"},
```

→ Obtener todas las alergias de un paciente determinado con base en el id del paciente.

```
* @Route("/patient/allergy/{id}", name="getPatientAllergyById", methods={"GET"})
```

→ Eliminar un registro de la tabla patient_allergy.

```
*@Route("/patient/{id_patient}/allergy/{id_allergy}",name="deletePatientAllergy", methods={"DELETE"})
```

→ Obtener una alergia de un paciente específico.

```
*@Route("/patient/allergy/{id_patient}/{id_allergy}",name="getOnePatientAllergyByIds", methods={"GET"})
```

Patient

- Clase Entity: Patient

En esta clase se definen los atributos y relaciones correspondientes a los doctores, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: PatientController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla paciente.

```
* @Route("/patient", name="patient", methods={"POST"})
```

→ Obtener el listado de todos los pacientes.

```
* @Route("/patient", name="getAllPatients", methods={"GET"})
```

→ Obtener un paciente específico con base en su id.

```
*@Route("/patient/{id}", name="getOnePatient", methods={"GET"})
```

→ Modificar el status covid de un paciente

```
*@Route("/patient/updatestatus/{id}/{status}",  
name="updateStatusCovid", methods={"PUT"})
```

→ Obtener el status covid de los pacientes, filtrados por estado y país.

```
*@Route("/patient/covid/{state}",name="getPatientCovid",  
methods={"GET"})
```

→ Obtener los datos de un paciente determinado con base en su id de usuario.

```
*@Route("/patient/user/{id}",name="getPatientByUserId",methods=  
{ "GET" })
```

→ Obtener la ubicación de los pacientes catalogados como casos COVID-19 confirmados.

```
* @Route("/covid/world", name="getLocationCovid",  
methods={"GET"})
```

Patient Disease

- Clase Entity: PatientDisease

En esta clase se definen los atributos y relaciones correspondientes a las enfermedades crónicas de cada paciente, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: PatientDiseaseController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla patient_disease.

```
*@Route("/patientdisease",name="patientdisease",methods={"POST"  
})
```

→ Obtener el listado de todos los registros de la tabla patient_disease.

```
*@Route("/patientdisease",name="getAllPatientDisease",methods={  
"GET"})
```

→ Obtener un registro de la tabla patient_disease con base en su id.

```
* @Route("/patientdisease/{id}", name="getOnePatientDisease",  
methods={"GET"})
```

→ Obtener todas las enfermedades de un paciente determinado con base en el id del paciente.

```
*@Route("/patient/disease/{id}",name="getOnePatientDiseaseById",  
, methods={"GET"})
```

→ Eliminar un registro de la tabla patient_disease.

```
*@Route("/patient/{id_patient}/disease/{id_disease}",name="deletePatientDisease", methods={"DELETE"})
```

→ Obtener una enfermedad de un paciente específico.

```
*@Route("/patient/disease/{id_patient}/{id_disease}",name="getOnePatientDiseaseByIds", methods={"GET"})
```

Patient Surgery

- Clase Entity: PatientSurgery

En esta clase se definen los atributos y relaciones correspondientes a las cirugías de cada paciente, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: PatientSurgery

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla patient_surgery.

```
* @Route("/patientsurgery", name="patientSurgery",  
methods={"POST"})
```

→ Obtener el listado de todos los registros de la tabla patient_surgery.

```
* @Route("/patientsurgery", name="getAllPatientSurgery",  
methods={"GET"})
```

→ Obtener un registro de la tabla patient_surgery con base en su id.

```
* @Route("/patientsurgery/{id}", name="getOnePatientSurgery",  
methods={"GET"})
```


- Obtener todas las cirugías de un paciente determinado con base en el id del paciente.

```
* @Route("/patient/surgery/{id}", name="getPatientSurgeryById",  
methods={"GET"})
```

- Eliminar un registro de la tabla patient_surgery.

```
*@Route("/patient/{id_patient}/surgery/{id_surgery}",name="deletePatientSurgery", methods={"DELETE"})
```

- Obtener una cirugía de un paciente específico.

```
*@Route("/patient/surgery/{id_patient}/{id_surgery}",name="getOnePatientSurgeryByIds", methods={"GET"})
```

Payment

- Clase Entity: Payment

En esta clase se definen los atributos y relaciones correspondientes a los diferentes tipos de pago, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: PaymentController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

- Obtener el listado de todos los tipos de pago.

```
* @Route("/payment", name="getAllPayment", methods={"GET"})
```

- Obtener un tipo de pago específico con base en su id.

```
*@Route("/payment/{id}", name="getOnePayment", methods={"GET"})
```

Prescription

- Clase Entity: Prescription

En esta clase se definen los atributos y relaciones correspondientes a las facturas, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: PrescriptionController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

- Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla prescription.

```
* @Route("/prescription", name="prescription",
methods={"POST"})
```

→ Obtener el listado de todas las recetas.

```
* @Route("/prescription", name="getAllPrescriptions",
methods={"GET"})
```

→ Obtener una receta médica específica con base en su id.

```
* @Route("/prescription/{id}", name="getOneprescription",
methods={"GET"})
```

→ Obtener todas las recetas de un paciente determinado con base en el id del paciente.

```
*@Route("/patient/{id}/prescriptions",name="getPrescriptionPatient", methods={"GET"})
```

→ Obtener todas las recetas prescritas por un doctor determinado con base en el id del doctor.

```
*@Route("/doctor/{id}/prescriptions",name="getPrescriptionByDoctor", methods={"GET"})
```

→ Obtener el nombre del archivo de la receta de una consulta específica con base en el id de la consulta.

```
@Route("/prescription/consultation/{id}",name="getOneprescriptionByIdConsultation", methods={"GET"})
```

Service

- Clase Entity: Service

En esta clase se definen los atributos y relaciones correspondientes a los diferentes servicios que puede ofrecer un médico, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: ServiceController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla service.

```
* @Route("/service", name="service", methods={"POST"})
```

→ Obtener el listado de todos los servicios.

```
* @Route("/service", name="getAllServices", methods={"GET"})
```

→ Obtener un servicio específico con base en su id.

```
*@Route("/service/{id}", name="getOneService", methods={"GET"})
```

→ Obtener el id del último servicio registrado en la tabla.

```
*@Route("/services/last",name="getLastService",methods={"GET"})
```

Speciality

- Clase Entity:Speciality

En esta clase se definen los atributos y relaciones correspondientes a las diferentes especialidades de los médicos, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: SpecialityController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Obtener el listado de todas la especialidades.

```
*@Route("/speciality",name="getAllSpecialities",methods={"GET"})
```

→ Obtener una especialidad específica con base en su id.

```
* @Route("/speciality/{id}", name="getOneSpeciality", methods={"GET"})
```

→ Obtener lista de las especialidades cuyo id se a diferente uno determinado.

```
*@Route("/speciality/canalize/{id}",name="getAllSpecialitiesExceptOne", methods={"GET"})
```

Surgery

- Clase Entity: Surgery

En esta clase se definen los atributos y relaciones correspondientes a las diferentes cirugías, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: SurgeryController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Obtener el listado de todas las cirugías.

```
* @Route("/surgery", name="getAllSurgeries", methods={"GET"})
```

→ Obtener una cirugía específica con base en su id.

```
*@Route("/surgery/{id}", name="getOneSurgery", methods={"GET"})
```

Tax Data

- Clase Entity: TaxData

En esta clase se definen los atributos y relaciones correspondientes a los datos de facturación de los pacientes, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: TaxDataController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla tax_data.

```
* @Route("/taxdata", name="taxdata", methods={"POST"})
```

→ Obtener el listado de todos los registros de la tabla tax_data.

```
* @Route("/taxdata", name="getAllTaxes", methods={"GET"})
```

→ Obtener un registro específico con base en su id.

```
*@Route("/taxdata/{id}", name="getOneTaxdata", methods={"GET"})
```

→ Obtener los datos de facturación de un paciente específico.

```
* @Route("/taxdata/patient/{id}", name="getOneTaxData", methods={"GET"})
```

→ Modificar los datos de facturación de un paciente específico.

```
*@Route("/taxdata/update", name="updateTaxData", methods={"POST"})
```

Ticket

- Clase Entity: Ticket

En esta clase se definen los atributos y relaciones correspondientes a los ticket de compra, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: TicketController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

→ Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla ticket.

```
* @Route("/ticket", name="tickets", methods={"POST"})
```

- Obtener todos los tickets de un paciente determinado con base en el id del paciente.

```
* @Route("/patient/{id}/tickets", name="getTicketPatient",  
methods={"GET"})
```

- Obtener el id del último registro de la tabla ticket.

```
* @Route("/ticket/last", name="getLastTicket", methods={"GET"})
```

User Data

- Clase Entity: UserData

En esta clase se definen los atributos y relaciones correspondientes a los usuarios del sistema, contiene un constructor y los métodos get y set de cada atributo.

- Clase Controller: UserController

Aquí se definen todas las rutas y sus métodos correspondientes, encargados de crear y devolver los objetos Response ante las peticiones del cliente, específicamente contiene las rutas encargadas de:

- Recuperar la información enviada por el cliente para añadir un nuevo registro a la tabla user_data.

```
* @Route("/user", name="user", methods={"POST"})
```

- Obtener todos los usuarios.

```
* @Route("/user", name="getAllUsers", methods={"GET"})
```

- Obtener la información de un usuario con base en su id.

```
* @Route("/user/{id}", name="getOneUser", methods={"GET"})
```

- Obtener la información de un usuario con base en su username.

```
* @Route("/username/{username}", name="getOneUsername",  
methods={"GET"})
```

- Obtener la información de un usuario con base en su username y su password.

```
* @Route("/login", name="login", methods={"POST"})
```

- Obtener la información del último registro de la tabla user_data.

```
* @Route("/userlast", name="getLastId", methods={"GET"})
```

Consultas

Al generar una entidad con `make:entity`, el comando también generó una clase `repository`. Un repositorio es una clase cuyo objetivo es ayudar a buscar entidades de una determinada clase.

Los repositorios vienen provistos por funciones que permiten realizar consultas básicas, tales como:

- `find($id)`: Permite buscar un objeto con base en el id del mismo.
- `findOneBy(array $criteria)`: Permite obtener un objeto que coincida con el arreglo de criterios proporcionado.
- `findAll()`: Devuelve todos los objetos de una determinada entidad.
- `findBy(array $criteria)`: Devuelve un arreglo de objetos que coinciden con el arreglo de criterios proporcionado.

Doctrine también permite escribir consultas más complejas utilizando el Lenguaje de Consulta Doctrine (DQL), el cual cuenta con una sintaxis similar a SQL, que permite buscar objetos de una determinada entidad (`entity`) en lugar de buscar filas de una tabla de base de datos, por lo tanto, intentar usar nombres de tablas y nombres de columnas una consulta con Doctrine arrojará un error. Se debe pensar en DQL como un lenguaje de consulta para objetos de las clases modelo, no para el esquema relacional.

Al realizar consultas en Doctrine, es posible:

1. Generar consultas a partir de cadenas enteras escritas a mano: En este caso se pasará la cadena DQL a la función `createQuery()`.
2. Utilizar el generador de consultas de Doctrine: `QueryBuilder` proporciona una API que está diseñada para construir una consulta DQL en varios pasos.

Ambos métodos de generación de consultas son utilizados en el presente proyecto.

Bill Repository

saveBill

Permite crear y almacenar una nueva instancia de la clase factura.

getBillsByPatient

Obtiene todas las facturas pertenecientes a un paciente, de acuerdo al id de este.

getBillsByDoctor

Obtiene todas las facturas emitidas por un doctor por la prestación de sus servicios en las consultas médicas que ha atendido.

Doctor Repository

saveDoctor

Almacenar la información de un nuevo doctor.

getDoctor

Obtener la información de todos los doctores registrados.

DoctorService Repository

saveDoctorService

Almacenar la relación entre un doctor y los servicios que él brinda

updateDoctorService

Actualizar el precio de un servicio en la relación entre un doctor y los servicios que ofrece.

removeDoctorService

Eliminar la relación de un doctor y algún servicio que ya no desee brindar.

Media Repository

saveMedia

Guardar la información de un nuevo recurso imagen asociado a una determinada consulta médica.

findByMedicalConsultation

Obtener las imágenes de una determinada consulta médica.

MedicalConsultation Repository

saveMedicalConsultation

Guardar la información de una nueva consulta médica.

findLast

Recuperar la información de la última consulta médica de un paciente.

findBySomeField

Obtener la información de consulta y de doctor, de aquellas consultas que ya han sido atendidas.

Patient Repository

savePatient

Almacena la información de un nuevo paciente

getPatient

Obtiene la información de un paciente de acuerdo a su id de usuario

findByState

Devuelve los pacientes que habitan en un estado determinado de la república. Información necesaria para generar el mapa de casos COVID-19 por estado.

countCovid

Cuenta la cantidad de casos positivos, negativos y sospechosos de COVID-19

PatientAllergy Repository

savePateintAllergy

Guardar una nueva relación entre un paciente y una alergia que padezca

findManyBySomeField

Obtener todas las alergias que padece un paciente

findOneBySomeField

Obtener una determinada relación entre paciente y alergia, con base en el id de paciente e id de alergia.

removePatientAllergy

Eliminar una relación entre paciente y alergia.

PatientDisease Repository

savePateintDisease

Guardar una nueva relación entre un paciente y una enfermedad crónica que padece.

findManyBySomeField

Obtener todas las enfermedades crónicas que padece un paciente

findOneBySomeField

Obtener una determinada relación entre paciente y enfermedad crónica, con base en el id de paciente e id de enfermedad.

removePatientAllergy

Eliminar una relación entre paciente y enfermedad crónica.

PatientSurgery Repository**savePateintSurgery**

Guardar una nueva relación entre un paciente y una cirugía que se ha realizado.

findManyBySomeField

Obtener todas las cirugías que se ha realizado un paciente

findOneBySomeField

Obtener una determinada relación entre paciente y cirugía, con base en el id de paciente e id de cirugía.

removePatientAllergy

Eliminar una relación entre paciente y cirugía.

Prescription Repository**savePrescription**

Almacenar la información de una receta médica

getPrescriptionByPatient

Obtener todas las recetas que se han prescrito a un paciente.

getPrescriptionByDoctor

Obtener todas las recetas que un doctor ha prescrito de acuerdo a la consulta.

Service Repository**saveService**

Guardar la información de un nuevo servicio

getLast

Recuperar la información del último servicio creado por un doctor.

Speciality Repository**findByExampleField**

Obtener especialidades que sean distintas a una especialidad dada.

TaxData Repository

saveTaxData

Guardar los datos de facturación de un paciente.

updateTaxData

Actualizar los datos de facturación de un paciente.

findOneBySomeField

Obtener los datos de facturación de un determinado paciente.

Ticket Repository

saveTicket

Guardar la información de un nuevo ticket de un paciente

getLastTicket

Obtener el último ticket de compra de un paciente

UserData Repository

saveUser

Almacenar la información de un nuevo usuario

findBySomeField

Recuperar la información de un usuario con base en su username

login

Recuperar la información de un usuario cuyo username y password coincidan con las credenciales ingresadas.

Descripción de componentes vue

Home.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista HOME.

Se compone de:

- Barra de menú que muestra las opciones disponibles del menú, según los permisos que posea el usuario en sesión
- Componentes que muestran una descripción general del sitio incluyendo texto e imágenes.

Login.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista LOGIN.

Se compone de:

- Formulario. Este a su vez se conforma de varios elementos como labels, inputs, y finalmente button, dichos elementos son necesarios para que el paciente ingrese sus credenciales y posteriormente el sistema las verifique al presionar el botón Login.

App.vue

Define los elementos de la parte gráfica que servirán para mostrar la barra de menú en cada una de las vistas.

Paciente

BillsHistory.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista HISTORIAL DE FACTURAS. Se compone de:

- Componente datatable para mostrar un listado de las facturas correspondientes a consultas previas del paciente.
- Botón asociado a cada consulta médica listada en el datatable, para acceder a la factura que se emitió para esta. Se abrirá en una nueva ventana del navegador al presionar el botón “Ver factura”.

ClinicHistory.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista HISTORIA CLÍNICA. Se compone de:

- Componente v-autocomplete para mostrar las alergias existentes, en el cual el paciente podrá seleccionar o quitar elementos.
- Componente v-autocomplete para mostrar las enfermedades crónicas existentes, en el cual el paciente podrá seleccionar o quitar elementos.
- Componente v-autocomplete para mostrar las cirugías existentes, en el cual el paciente podrá seleccionar o quitar elementos.

- Botón GUARDAR para almacenar o actualizar la información correspondiente a las alergias, enfermedades crónicas y cirugías del paciente en sesión.

DoctorList.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista MÉDICOS. Se compone de:

- Componente datatable para mostrar un listado de los médicos registrados en el sistema.

MedicalConsultationForm.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista CONSULTA MÉDICA. Se compone de:

- Formulario. Este a su vez se conforma de varios elementos como labels, inputs, select, y finalmente button, dichos elementos son necesarios para que el paciente registre información necesaria para agendar una nueva consulta médica y posteriormente se almacene al presionar el botón GUARDAR.

Medicines.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista MEDICAMENTOS correspondiente a la tienda en línea. Se compone de:

- Componente datatable para mostrar un listado de los medicamentos disponibles en la tienda.
- Botón AGREGAR AL CARRITO asociado a cada medicamento listado en el datatable, para añadirlo al carrito de compra.
- Botón VER CARRITO DE COMPRAS que despliega un componente datatable con el listado de los productos que el usuario ha añadido al carrito
- Botón asociado a cada producto que permite quitarlo del carrito.
- Botón VOLVER A PRODUCTOS que regresa al componente datatable que muestra listado de los medicamentos disponibles en la tienda por si el usuario desea añadir más productos al carrito.
- Botón GENERAR que confirma la compra y envía el ticket por correo electrónico.

PatientForm.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista REGISTRO de un paciente. Se compone de:

- Formulario. Este a su vez se conforma de varios elementos como labels, inputs, select, y finalmente button, dichos elementos son necesarios para que el nuevo paciente ingrese su información y posteriormente se almacene al presionar el botón GUARDAR.

PriorConsultation.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista CONSULTAS PREVIAS. Se compone de:

- Componente datatable para mostrar un listado de las consultas previas del paciente.
- Botón asociado a cada consulta médica listada en el datatable, para acceder a la receta que se prescribió para esta. Se abrirá en una nueva ventana del navegador al presionar el botón “Ver receta”.
- Botón asociado a cada consulta médica listada en el datatable, para acceder a la tienda en línea. Se abrirá en una nueva ventana del navegador al presionar el botón “Surtir en línea”.

TaxDataForm.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista DATOS DE FACTURACIÓN. Se compone de:

- Formulario. Este a su vez se conforma de varios elementos como labels, inputs, select, y finalmente button, dichos elementos son necesarios para que el paciente ingrese su información y posteriormente se almacene o actualice al presionar el botón GUARDAR.

TicketHistory.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista HISTORIAL DE COMPRAS. Se compone de:

- Componente datatable para mostrar un listado de los tickets correspondientes a compras previas del paciente.

- Botón asociado a cada compra listada en el datatable, para acceder al ticket que se emitió para esta. Se abrirá en una nueva ventana del navegador al presionar el botón “Ver ticket”.

Doctor

Bill.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista FACTURA. Se compone de:

- Botón marcado con el icono + para que muestre una nueva ventana dialog con la opción de añadir servicios a los conceptos de facturación.
- Componente dialog para mostrar el formulario de AGREGAR CONCEPTO.
- Componente card para la visualización de los servicios que se han agregado como concepto de cobro a la factura.
- Botón GENERAR para crear el PDF a partir de los servicios que se han agregado a la factura y enviar automáticamente por correo el link de descarga de la factura.

ConsultationDetails.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista DETALLES DE LA CONSULTA. Se compone de:

- Varios componentes div para mostrar la información del paciente y las características de la consulta.
- Botón CANALIZAR para que muestre una nueva ventana dialog con la opción de canalizar al paciente a otro especialista.
- Botón STATUS COVID-19 para mostrar una nueva ventana dialog con el fin de actualizar el estatus del paciente como confirmado, sospechoso o negativo.
- Botón RECETAR para mostrar una nueva ventana donde se procederá a generar la receta para esa consulta médica.

ConsultationList.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista CONSULTAS PENDIENTES. Se compone de:

- Componente card para la visualización de la información general de la consulta médica.
- Botón en la esquina superior derecha de cada componente card, para abrir los detalles completos de la consulta médica que se ha seleccionado.

CRUDDoctorService.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista SERVICIOS. Se compone de:

- Componente datatable para mostrar un listado de los servicios ofrecidos por un determinado doctor.
- Botón marcado con el icono de papelera, asociado a cada servicio listado en el datatable, para eliminar ese servicio del catálogo del doctor.
- Botón AGREGAR, para abrir una nueva ventana dialog que contendrá el formulario para agregar un servicio de los existentes al catálogo de servicios personal del doctor.
- Botón NUEVO, para abrir una nueva ventana dialog que contendrá el formulario para crear un nuevo servicio si este no existe, y agregarlo al catálogo de servicios personal del doctor.

DoctorForm.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista REGISTRO de un doctor. Se compone de:

- Formulario. Este a su vez se conforma de varios elementos como labels, inputs, select, y finalmente button, dichos elementos son necesarios para que el nuevo doctor ingrese su información y posteriormente se almacene al presionar el botón GUARDAR.

Prescription.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista RECETA. Se compone de:

- Botón marcado con el icono + para que muestre una nueva ventana dialog con la opción de añadir medicamentos a la receta.

- Componente dialog para mostrar el formulario de AGREGAR MEDICAMENTO.
- Componente card para la visualización de los medicamentos que se han agregado a la receta.
- Botón GENERAR para crear el PDF a partir de los medicamentos que se han agregado a la receta y enviar automáticamente por correo el link de descarga de la receta.

PriorConsultation.vue

Define los elementos de la parte gráfica que servirán para mostrar la vista CONSULTAS PREVIAS. Se compone de:

- Componente datatable para mostrar un listado de las consultas previas que ya ha atendido ese doctor.
- Botón asociado a cada consulta médica listada en el datatable, para acceder a la receta médica que se emitió para esta. Se abrirá en una nueva ventana del navegador al presionar el botón “Ver receta”
- Botón asociado a cada consulta médica listada en el datatable, para acceder a la factura que se emitió para esta. Se abrirá en una nueva ventana del navegador al presionar el botón “Ver factura”.

MexicoProvidencesMap.vue

Define los elementos de la parte gráfica que servirán para mostrar un mapa de la república que será incluido en la vista MAPA CASOS COVID-19. Se compone de:

- Componente div para mostrar un mapa de México a partir de las cadenas de texto correspondientes al SVG, que son recuperadas del archivo mexico-providences.json. En este mapa podrá seleccionarse un estado en particular.

GoogleMap.vue

Define los elementos de la parte gráfica que servirán para mostrar una plantilla de mapa mundial que será renderizada en la vista MAPA CASOS COVID-19. Se compone de:

- Componente gmap-map para mostrar un mapa mundial en el que se mostrarán los casos de covid.

- Componente `gmap-marker`, para definir el estilo y posición en que se mostrarán los diferentes marcadores correspondientes a cada caso confirmado de covid-19.

RenderMap.vue

Define los elementos de la parte gráfica que servirán para la vista MAPA CASOS COVID-19. Se compone de:

- Componente `GoogleMap`, generado anteriormente para mostrar un mapa mundial, así como los diferentes marcadores correspondientes a la posición de cada caso confirmado de covid-19.
- Componente `MéxicoProvincencesMap`, generado anteriormente para mostrar un mapa de México en el cual cada estado tiene asociado un evento `map-click` para devolver el id y título del estado.
- Componente `datatable` para mostrar la información de los casos sospechosos, confirmados y negativos por cada ciudad perteneciente al estado que se haya seleccionado.

Usuarios del sistema

Cada usuario posee un ID propio, un nombre de usuario y una contraseña para acceder al sistema, además de que se encuentra asociado a un rol específico (doctor o paciente), de acuerdo a este se conforma un esquema de permisos para el control de las actividades que puede realizar.

Paciente: El usuario paciente posee los siguientes permisos:

- Registro y modificación de su historia clínica (alergias, enfermedades crónicas y cirugías)
- Acceso al listado de médicos, siendo capaz de filtrar la búsqueda por: nombre, especialidad, ciudad, estado, país y correo electrónico.
- Alta de una nueva consulta médica.
- Consultar su historial de citas médicas, con la posibilidad de descargar el archivo correspondiente a la receta que le fue prescrita en cada una de ellas.
- Registro y modificación de sus datos de facturación.
- Acceso a la tienda en línea para que pueda surtir sus medicamentos.

- Acceso a un mapa del mundo en el que se podrán visualizar los casos de COVID-19 que han sido confirmados, con base a la información que posee el sistema.
- Acceso a un mapa de la República Mexicana en la cual podrá seleccionar cualquier estado y se le mostrará una tabla con los casos confirmados, sospechosos y negativos de COVID-19 indicando la ciudad.
- Consultar su historial de compras, con la posibilidad de descargar el archivo correspondiente al ticket de cualquiera de sus compras realizadas en la tienda.
- Acceso a la aplicación móvil en la podrá visualizar su historial de citas y los detalles de cada una, así como agendar una nueva cita y ver el mapa de casos COVID-19 confirmados en el mundo.



Vista de la pantalla principal del sistema disponible para usuarios con el rol de paciente.

Doctor: Será capaz de llevar a cabo las acciones descritas a continuación:

- Alta, bajas y cambios de los servicios que ofrece.
- Acceso a listado de consultas que se encuentran en estatus de atención pendiente y que corresponden a su especialidad.
- Acceso a los detalles de una determinada consulta.
- Prescripción de recetas médicas y envío de las mismas por correo electrónico.
- Generación de facturas por los servicios brindados y envío de las mismas por correo electrónico.
- Canalizar pacientes a médicos de otra especialidad.
- Actualizar la información de un paciente para indicar si es un caso confirmado, sospechoso o negativo de COVID-19.
- Consultar el historial de las citas que ha atendido, con la posibilidad de descargar los archivos correspondientes a la receta prescrita y la factura generada.



Vista de la pantalla principal del sistema disponible para usuarios con el rol de doctor.

Pantallas del sistema

Login

Se solicitará al usuario ingresar nombre de usuario y contraseña para su ingreso al sistema.

Consultorio Online Home Login

Consultorio Online

Username

Username

Password

Password

Login

¿No tienes cuenta? [Registrate](#)

Registro

Para el registro, se solicitará al usuario ingresar un nombre de usuario, contraseña, la confirmación de la contraseña y deberá indicar si es un paciente o un doctor.

Consultorio Online Home Login

Sign Up

Username

Username

Password

Password

Confirmar password

Password

Role

Doctor

Registrarme

Registrarme

Formulario de paciente

Continuando con el registro de un usuario en el sistema, en caso de que seleccione el rol de paciente, deberá llenar el siguiente formulario. La información solicitada es: nombre, apellidos, dirección, latitud y longitud, teléfono, fecha de nacimiento, email, ciudad, estado y país.

Nuevo Paciente

Nombre

Apellidos

Dirección

Latitud

Longitud

Teléfono

Fecha de nacimiento



Email

Ciudad

Estado

País

Guardar

Formulario de doctor

Continuando con el registro de un usuario en el sistema, en caso de que seleccione el rol de doctor, deberá llenar el siguiente formulario. La información solicitada es: nombre, apellidos, dirección, teléfono móvil, cédula profesional, email, su especialidad, ciudad, estado y país.

Nuevo Médico

Nombre

Apellidos

Dirección

Teléfono móvil

Cédula profesional

Email

Especialidad

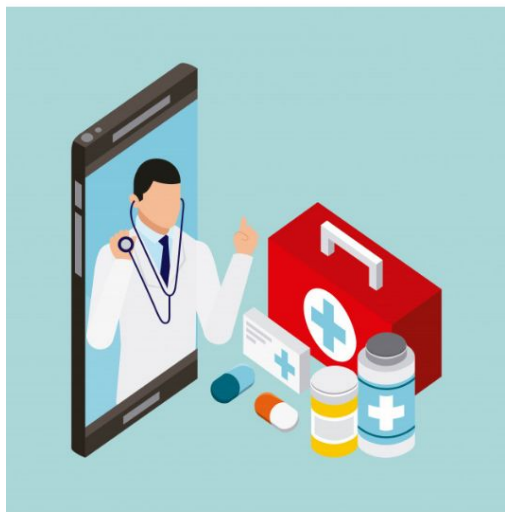
Ciudad

Estado

País

Menú Principal.

Se muestra una descripción sobre el sitio, así como las opciones disponibles del menú, según los permisos que posea el usuario en sesión.



¿QUIÉNES SOMOS?

Consultorio Online es una plataforma que busca ayudar a médicos y pacientes

Podrás seleccionar y comunicarte de forma ágil y segura con el profesional de la salud que más se adecue a tus necesidades a través de un amplio directorio de especialidades, además serás capaz de registrar tu historia clínica, ver tu historial de citas y acceder fácilmente a tus recetas y facturas.

¡Descarga nuestra app móvil!

Revisa tu historial de citas, y agenda consultas desde cualquier lugar.

Nuestros Servicios



Médicos

Podrás registrar tus servicios y sus costos, tendrás acceso a la historia clínica de tus pacientes y podrás generar recetas de forma digital.



Pacientes

Podrás seleccionar a los especialistas que se adapten a tus necesidades, registra tus síntomas y adjunta fotos como evidencia de forma segura.



Tienda en línea

Contamos con una tienda en línea para que surtas tus medicamentos, elige los productos que se adapten mejor a tus necesidades con base en su precio y sus características.

Copyrights © [2021] CONSULTORIO ONLINE Design by Paulina Otero y Naomi Ortiz

Paciente

Historia clínica

Se podrán registrar, modificar y eliminar las alergias y enfermedades crónicas que un paciente padezca, así como las cirugías a las que se ha sometido.

Consultorio Online Home Historia Clínica Médicos Consultas Médicas Facturas Compras Casos COVID 19 Tienda Logout

Historia Clínica

Seleccione las alergias que tenga

Alergia al latex

Alergia al sol

Seleccione las enfermedades crónicas que padezca

Diabetes

Fibrosis quística

Seleccione las cirugías a las que se ha sometido

Reparación de hernia

Guardar

Médicos

Se desplegará una tabla con la información de todos los especialistas registrados en el sistema, los datos pueden filtrarse por nombre,apellidos, especialidad, ciudad, estado, país o email.

Consultorio Online Home Historia Clínica Médicos Consultas Médicas Facturas Compras Casos COVID 19 Tienda Logout						
Médicos						
<input type="text"/>						
Name	Lastname	Speciality	City	State	Country	Email
Pedro	Sánchez Pérez	Pediatría	Celaya	Guanajuato	México	pedrosm@gmail.com
Julieta	Ramírez Trejo	Geriatría	Orizaba	Veracruz	México	julieta@gmail.com
Saul Fabian	Cruces Anaya	Médico general	Candelaria	Campeche	México	saul@gmail.com
Pamela	Ramírez Trejo	Traumatología	Santiago de Querétaro	Querétaro	México	pamela@gmail.com
Rows per page: 10 1-4 of 4 < >						

Consulta médica

El paciente podrá registrar una nueva consulta, registrando sus síntomas, la fecha en que desea ser atendido, y el especialista, además se permite adjuntar una imagen en caso de ser necesario.

Consultorio Online Home Historia Clínica Médicos Consultas Médicas Facturas Compras Casos COVID 19 Tienda Logout	
Consulta Médica	
Síntoma	<input type="text"/>
Fecha de consulta	<input type="text"/>
Especialidad	<input type="text"/>
Doctor	<input type="text"/>
File	<div>Seleccionar archivo No se eligió archivo</div>
Guardar	

Consultas previas

Se desplegará el historial de citas del paciente en sesión, la información podrá filtrarse por fecha, especialidad, doctor y estatus de atención. Además, en el caso de las consultas que ya han sido atendidas, se muestran dos botones, uno para ver la receta prescrita por el médico y otro para surtir los medicamentos en la tienda en línea.

Consultas Previas

<div> <div>Buscar</div> <div></div> </div>					
Fecha	Especialidad	Doctor	Estatus de atención	Ver receta	Surtir en línea
2021-01-13	Médico general	Saul Fabian Cruces Anaya	Atendido		
2021-01-22	Médico general	Saul Fabian Cruces Anaya	Atendido		
2021-01-13	Geriatría	Julieta Ramírez Trejo	Pendiente		
2021-01-13	Pediatría	Pedro Sánchez Pérez	Pendiente		
<div> <div>Rows per page:</div> <div>10</div> <div>1-4 of 4</div> <div><</div> <div>></div> </div>					

Datos de facturación

Se podrán visualizar, registrar y modificar los datos de facturación de usuario en sesión. La información a ingresar es: domicilio de facturación, domicilio de envío y tipo de pago.

Datos de facturación

Nombre

Jorge Torres Pérez

Domicilio de Facturación

Domicilio de facturación

Domicilio de envío

Domicilio de envío

Tipo de pago

Selecciona un tipo de pago

Guardar

Historial de facturas

Se mostrará el historial de facturación del paciente en sesión, con la posibilidad de ver el archivo correspondiente a la factura de una consulta determinada. Los datos se podrán filtrar por fecha de consulta, especialidad y doctor.

Consultorio Online Home Historia Clínica Médicos Consultas Médicas Facturas Compras Casos COVID 19 Tienda Logout			
Historial de facturas			
<div> <div>Buscar</div> <div></div> </div>			
Fecha	Especialidad	Doctor	Ver factura
2021-01-13	Médico general	Saul Fabian Cruces Anaya	
2021-01-22	Médico general	Saul Fabian Cruces Anaya	
<div> <div>Rows per page:</div> <div>10</div> <div>1-2 of 2</div> <div><</div> <div>></div> </div>			

Historial de compras

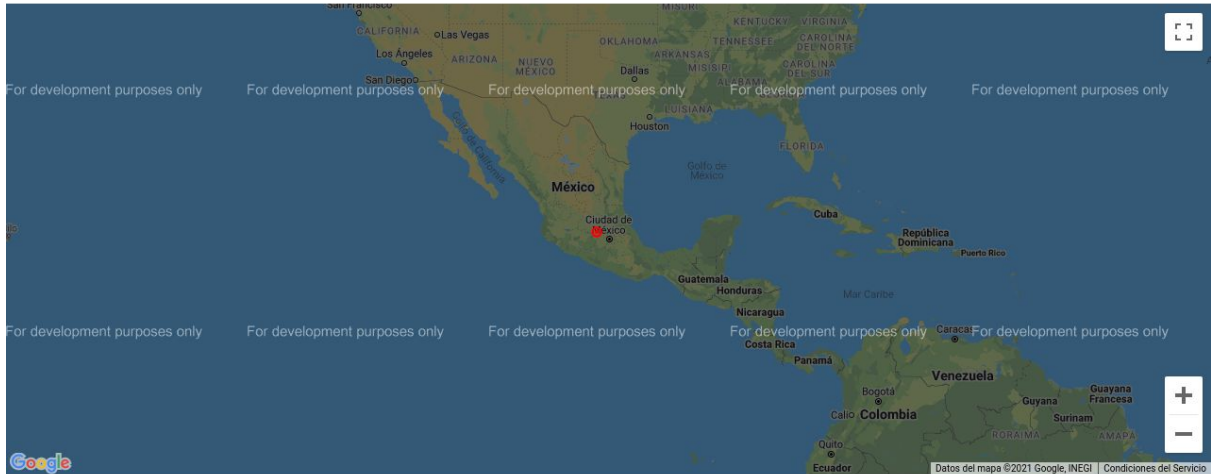
Se mostrará el historial del paciente en sesión, de las compras realizadas en la tienda en línea, con la posibilidad de ver el archivo correspondiente al ticket. Los datos se podrán filtrar por fecha de compra.

Consultorio Online Home Historia Clínica Médicos Consultas Médicas Facturas Compras Casos COVID 19 Tienda Logout	
Historial de compras	
<div> <div>Buscar</div> <div></div> </div>	
Fecha de compra	Ver ticket
2021-01-23	
2021-01-23	
2021-01-23	
2021-01-23	
2021-01-23	
<div> <div>Rows per page:</div> <div>10</div> <div>1-5 of 5</div> <div><</div> <div>></div> </div>	

Mapa de casos COVID-19

Se desplegará un mapa del mundo en el que se muestran los casos de personas contagiadas de COVID-19 a través de un punto rojo colocado en su ubicación. Además se visualizará un mapa de México, en el cual se podrá dar clic a un estado específico y se listaran los casos confirmados, sospechosos y negativos de las ciudades.

Mapa casos COVID-19



República Mexicana

Guanajuato

Ciudad	Confirmados	Sospechosos	Negativos
Celaya	3	0	1

Rows per page: 10 1-1 of 1



Medicamentos

Se mostrarán todos los medicamentos disponibles en la tienda en línea. Podrá realizarse un filtro por: Nombre del medicamento, tipo, sustancia activa, laboratorio y categoría. Cada uno de estos productos cuenta con un botón que permitirá añadirlo al carrito de compras.

[Consultorio Online](#)
[Home](#)
[Historia Clínica](#)
[Médicos](#)
[Consultas Médicas](#)
[Facturas](#)
[Compras](#)
[Casos COVID 19](#)
[Tienda](#)
[Logout](#)

Medicamentos

VER CARRITO DE COMPRAS

Loratadina

Tipo: Tabletas

Sustancia Activa: Loratadina

Laboratorio: ULTRA

Categoría: Antialérgicos

Precio: 12.50

AGREGAR AL CARRITO

Aliren

Tipo: Tabletas

Sustancia Activa: Amantadina/Clorf

Laboratorio: SIEGFRIED RHEIN

Categoría: Antitusivos

Precio: 63.00

AGREGAR AL CARRITO

Gelmicin

Tipo: Crema

Sustancia Activa: Betametasona/Gei

Laboratorio: Collins

Categoría: Antialérgicos

Precio: 39.00

AGREGAR AL CARRITO

Versalver

Tipo: Comprimidos

Sustancia Activa: Valsartán

Laboratorio: MAVER

Categoría: Antiinflamatorios

Precio: 139.00

AGREGAR AL CARRITO

Carrito de compras

Se podrá visualizar, modificar la cantidad y eliminar los elementos del carrito de compras.

[Consultorio Online](#)
[Home](#)
[Historia Clínica](#)
[Médicos](#)
[Consultas Médicas](#)
[Facturas](#)
[Compras](#)
[Casos COVID 19](#)
[Tienda](#)
[Logout](#)

Carrito de compras

VOLVER A PRODUCTOS

	Nombre	Precio	Cantidad	Subtotal
	Gelmicin	39.00	1	39.00
	Aliren	63.00	1	63.00

Rows per page: 10 1-2 of 2

Total: \$ 102






GENERAR

Doctor

Servicios

Se podrán visualizar, modificar y eliminar los servicios que ofrece el doctor en sesión.

60

Consultorio Online Home Servicios Consultas Médicas ▾ Logout		
Servicios	<div>NUEVO</div> <div>AGREGAR</div>	
Descripción	Costo	
Consulta general	400.00	
Servicio de nutrición	300.00	
Control de embarazo	200.00	
Prueba	100.00	
Prueba2	120.00	
<div>Rows per page: 10 ▾ 1-5 of 5 < ></div>		

Agregar servicio

Se podrá añadir un elemento nuevo a la lista de servicios que ofrece el doctor, únicamente deberá seleccionarlo de la lista y registrar el costo.

Agregar a mis servicios

Descripción ▴

Control de embarazo
Servicio de nutrición
Consulta general
Prueba
Prueba2
Prueba 3
Revisión general

Costo

0

CANCELAR

GUARDAR

Nuevo servicio

En caso de que el servicio que se desea brindar no esté disponible en el sistema, podrá añadirse registrando la descripción y el costo.

Nuevo servicio

Descripción

Costo

0

CANCELAR GUARDAR

Consultas pendientes

Se desplegarán las consultas que se encuentren pendientes de atender, correspondientes a la especialidad del doctor en sesión.

Consultorio Online Home Servicios Consultas Médicas Logout

Consultas pendientes

Paciente: Naomi Ortiz González

2021-01-25

Síntomas

Fiebre, pérdida de apetito, dolor de ca...

Paciente: Paulina Otero
Martínez

2021-01-25

Síntomas

Dolor de cabeza, fatiga intensa, manc...

Detalles de la consulta

Se mostrará la información correspondiente a una consulta en específica, visualizando el nombre del paciente, su edad, síntomas, alergias, enfermedades crónicas y cirugías.

Consultorio Online Home Servicios Consultas Médicas Logout

Detalles de la consulta

Nombre: Paulina Otero Martínez

Edad: 22

CANALIZAR

STATUS COVID-19



Síntomas

Dolor de cabeza, fatiga intensa, manchas rojas y planas en la cara, las manos y los antebrazos, y en el tronco.

Alergias

Alergia al latex
Alergia al sol
Alergia a las hormonas

Enfermedades crónicas

Fibrosis quística
Alzheimer

Cirugías

Biopsias de
crecimientos
Cesárea

[Recetar](#)

Canalizar paciente a otro especialista

El doctor podrá canalizar al paciente, seleccionado la especialidad y el doctor.

Canalizar paciente a otro especialista

Especialidad

Doctor

Seleccione un doctor

[ACEPTAR](#) [CANCELAR](#)

Actualizar status COVID-19 del paciente

El doctor podrá seleccionar de la lista las opciones: sospechoso, negativo y confirmado.

Actualizar status COVID-19 del paciente

Status

Negativo

[ACTUALIZAR](#) [CANCELAR](#)

Receta médica

El doctor podrá visualizar la lista de medicamentos que se están prescribiendo, además podrá modificar las instrucciones dadas. Cuando termine de añadir los medicamentos a la receta, podrá generar un pdf y se enviará por correo.

Receta médica



Agrifen, Acetaminofén, Tabletas, Laboratorio PISA	
Instrucciones Tomar 1 cada 6 horas por 3 días	Costo 120.00

GENERAR

Recetar medicamento

Se seleccionará de la lista el medicamento a prescribir y se desplegará automáticamente su información. Deberán registrarse las instrucciones que deberá seguir el paciente.

Recetar medicamento

Nombre	Tipo
Sustancia	Laboratorio
Instrucciones	Costo 0

CANCELAR AGREGAR

Generar factura

El doctor podrá visualizar los servicios que se están brindando en una consulta siendo capaz de modificar el costo. Cuando termine de añadir los servicios a la factura, podrá generar un pdf y se enviará por correo.

Generar Factura



Consulta general

Costo
400.00

GENERAR

Agregar concepto

Se seleccionará de la lista el servicio a facturar y se desplegará automáticamente el costo.

Agregar concepto

Descripcion

Consulta general

Servicio de nutrición

Control de embarazo

Prueba

Prueba2

Costo

0

CANCELAR

AGREGAR

Consultas previas

Se desplegarán las consultas que han sido atendidas por el doctor en sesión. La información podrá filtrarse por: fecha de consulta, paciente y síntomas. Además podrá ver la receta y la factura correspondientes a cada cita.

Consultas Previas

Buscar					
Fecha	Paciente	Síntomas	Estatus de atención	Ver receta	Ver factura
2021-01-13	Naomi Ortiz González	Dolor de cabeza, fatiga intensa, manchas rojas y planas en la cara, las manos y los antebrazos, y en el tronco.	Atendido		
2021-01-14	Paulina Otero Martínez	Fiebre, pérdida de apetito, dolor de cabeza, bultos rojos o rosados elevados (pápulas), que brotan durante varios días.	Atendido		
2021-01-22	Naomi Ortiz González	Tos seca y cuerpo cortado	Atendido		
2021-01-25	Paulina Otero Martínez	Dolor de cabeza, fatiga intensa, manchas rojas y planas en la cara, las manos y los antebrazos, y en el tronco.	Atendido		

Rows per page: 10 1-4 of 4

Aplicación móvil

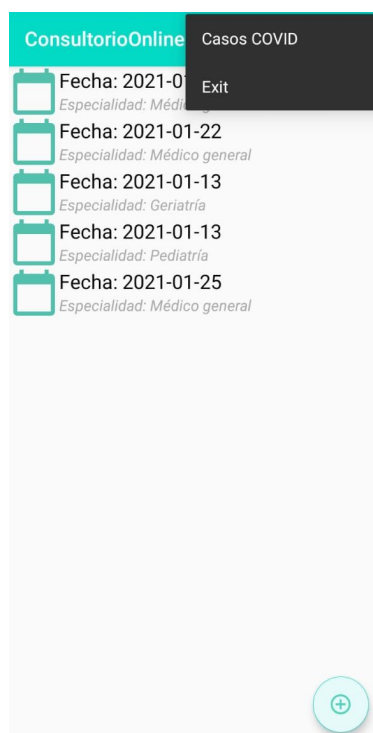
Login

Se solicitará al usuario ingresar nombre de usuario y contraseña para su ingreso al sistema.

The image shows a mobile application login screen with a light blue gradient background. At the top, the text 'Consultorio Online' is displayed in a dark blue font. Below this is a dark blue circular icon containing a white silhouette of a person. Underneath the icon are two input fields: 'Username' and 'Password', each followed by a horizontal line for text entry. At the bottom center, there is a dark blue rectangular button with the word 'LOGIN' in white capital letters.

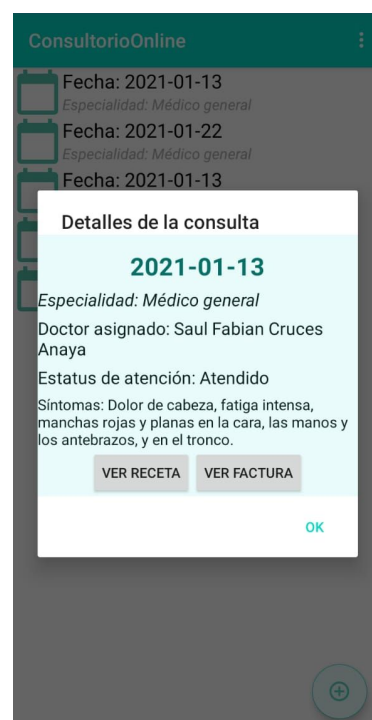
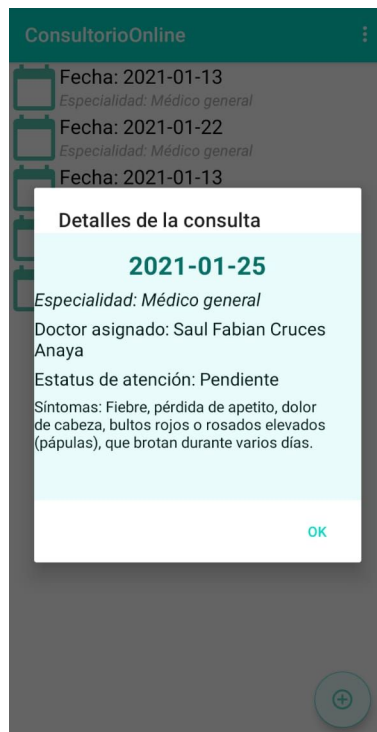
Consultas

Se desplegará una lista con las consultas correspondientes al usuario en sesión. Además contiene un menú con las opciones Casos COVID y exit.



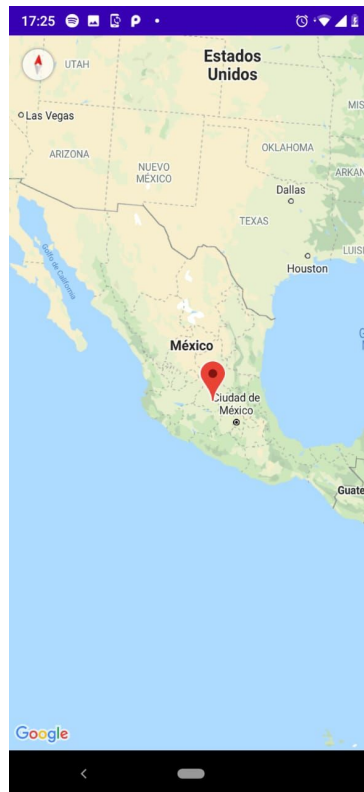
Detalles de consulta

Se mostrarán los detalles de la consulta que se seleccione, mostrando la fecha, la especialidad, el doctor asignado, el estatus de atención y los síntomas registrados. En caso de que la consulta ya haya sido atendida, también se mostrarán los botones ver receta y ver factura para descargar sus respectivos archivos.



Mapa de casos COVID-19

Se desplegará un mapa del mundo en el que se muestran los casos de personas contagiadas de COVID-19 a través de un punto rojo colocado en su ubicación.



Descripción de módulos

Registro nuevo usuario

Nombre : Sing up
Actor(es): Usuario
Descripción: Alta del nuevo usuario en la base de datos del consultorio
Precondiciones: Ninguna
Flujo normal: <ol style="list-style-type: none"> 1. El usuario ingresa a la pantalla de registro que pide nombre de usuario, clave y se solicita que ingrese el sol con el que quiere darse de alta 2. El usuario presiona el botón registrar.
Flujo alternativo: Se genera un POST request al servidor para almacenar la información del nuevo usuario en la base de datos en la tabla userdata.
Pos condición: <ol style="list-style-type: none"> 1. El usuario será redireccionado a un formulario para completar su registro de acuerdo con el rol seleccionado

Formulario nuevo paciente

Nombre : Nuevo paciente
Actor(es): Paciente
Descripción: Alta de los datos de un nuevo paciente
Precondiciones: 1. El paciente debe de haberse dado de alta previamente en el sistema
Flujo normal: 1. El usuario es redirigido al formulario de paciente para llenar los campos nombre, apellidos, dirección, ciudad, estado, país, latitud, longitud, teléfono, fecha de nacimiento y correo electrónico. 2. El usuario presiona el botón guardar para capturar su información
Flujo alternativo: Se genera un POST request al servidor para almacenar la información del nuevo paciente en la base de datos en la tabla patient.
Pos condición: 1. Login del paciente al sistema con sus credenciales de usuario.

Formulario nuevo doctor

Nombre : Nuevo doctor
Actor(es): Doctor
Descripción: Alta de los datos de un nuevo doctor
Precondiciones: 1. El doctor debe de haberse dado de alta previamente en el sistema
Flujo normal: 1. El usuario es redirigido al formulario de doctor para llenar los campos nombre, apellidos, dirección, ciudad, estado, país, teléfono, cédula, especialidad y correo electrónico. 2. El usuario presiona el botón guardar para capturar su información
Flujo alternativo: Se genera un POST request al servidor para almacenar la información del nuevo doctor en la base de datos en la tabla doctor.
Pos condición: 1. Login del doctor al sistema con sus credenciales de usuario.

Ingreso al sistema

Nombre : Login
Actor(es): Paciente o doctor
Descripción: Ingreso a los módulos correspondientes según el rol, a través de usuario y clave
Precondiciones: <ol style="list-style-type: none">1. El usuario deben tener un nombre de usuario y contraseña registrados en la base de datos
Flujo normal: <ol style="list-style-type: none">1. El usuario ingresa a la pantalla de logueo que pide nombre de usuario y clave2. El usuario presiona el botón Login3. El usuario ingresa al menú principal que le corresponde según su rol
Flujo alternativo: Se genera un GET request al servidor para obtener la información del usuario según su rol de doctor o paciente.
Pos condición: <ol style="list-style-type: none">1. Ingreso al sistema de acuerdo a su rol

Formulario historia clínica

Nombre : Historia clínica
Actor(es): Paciente
Descripción: Registro, modificación y eliminación de las alergias y enfermedades crónicas que un paciente padezca, así como las cirugías a las que se ha sometido.
Precondiciones: <ol style="list-style-type: none">1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none">1. El paciente selecciona nuevas alergias, enfermedades crónicas o cirugías mediante el listado disponible en el autocomplete2. El paciente modifica sus alergias, enfermedades crónicas o cirugías eliminandolas al dar clic sobre el icono marcado con una X3. El paciente guarda los cambios presionando el botón Guardar
Flujo alternativo: <ol style="list-style-type: none">1. Se genera un POST request al servidor para almacenar los nuevos padecimientos en la tabla patient_allergy, patient_surgery o patient_disease, según sea el caso.2. Se genera una petición DELETE al servidor para eliminar de la base de datos la

relación patient_allergy, patient_surgery o patient_disease, con base en el id de paciente y id del padecimiento a eliminar.
Pos condición: Historia clínica actualizada.

Listado de médicos

Nombre : Médicos
Actor(es): Paciente
Descripción: Visualización de tabla con la información de todos los especialistas registrados en el sistema, permitiendo búsqueda por nombre,apellidos, especialidad, ciudad, estado, país o email.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none"> 1. El usuario ingresa el criterio de búsqueda en el textfield señalado con el ícono de lupa. 2. La tabla muestra resultados coincidentes con el criterio.
Flujo alternativo: Se genera un GET request al servidor para obtener la información de todos los doctores, necesaria para el llenado inicial de la tabla.
Pos condición:

Formulario nueva consulta médica

Nombre : Consulta médica
Actor(es): Paciente
Descripción: Alta de los datos de una nueva consulta
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none"> 1. El paciente ingresa sus síntomas, una fecha para su consulta, la especialidad que requiere, un doctor de dicha especialidad. 2. Opcionalmente puede agregar imágenes a los datos de su consulta.

Flujo alternativo: Se genera un POST request al servidor para almacenar la información de la nueva consulta en la tabla medical_consultation.
Pos condición: Nueva consulta creada

Listado consultas médicas previas

Nombre : Consultas previas
Actor(es): Paciente
Descripción: Historial de las citas del paciente. Permite búsqueda por fecha, especialidad, doctor y estatus de atención. En el caso de las consultas que ya han sido atendidas puede consultar la receta prescrita y surtir los medicamentos en la tienda en línea.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none"> 1. El usuario ingresa el criterio de búsqueda en el textfield señalado con el ícono de lupa. 2. La tabla muestra resultados coincidentes con el criterio.
Flujo alternativo: <ol style="list-style-type: none"> 1. Se genera un GET request al servidor para obtener la información de todas las consultas, necesaria para el llenado inicial de la tabla. 2. Se genera un GET request al servidor para obtener la receta médica de aquellas consultas médicas que ya han sido atendidas.
Pos condición:

Formulario datos de facturación

Nombre : Datos de facturación
Actor(es): Paciente
Descripción: Alta y modificación de los datos de facturación de un paciente.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente

<p>Flujo normal:</p> <ol style="list-style-type: none"> 1. El paciente ingresa sus datos de facturación en el formulario, o modifica su información si es que esta ya existe. 2. El paciente presiona el botón guardar para almacenar los cambios.
<p>Flujo alternativo:</p> <ol style="list-style-type: none"> 1. Se genera un POST request al servidor para almacenar la información de los datos de facturación de un paciente en la tabla tax_data, si es la primera vez. 2. Se genera un PUT request al servidor para modificar el actual recurso de destino con los parámetros de la petición, en la tabla tax_data, si es que desea cambiar sus datos de facturación.
<p>Pos condición:</p> <p>Datos de facturación creados o modificados.</p>

Listado historial de facturas

Nombre : Historial de facturas
Actor(es): Paciente
Descripción: Historial de facturación del paciente por concepto de consulta y servicios brindados. Permite búsqueda por fecha de consulta, especialidad y doctor, además de poder visualizar el archivo correspondiente.
<p>Precondiciones:</p> <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
<p>Flujo normal:</p> <ol style="list-style-type: none"> 1. El usuario ingresa el criterio de búsqueda en el textfield señalado con el ícono de lupa. 2. La tabla muestra resultados coincidentes con el criterio.
<p>Flujo alternativo:</p> <ol style="list-style-type: none"> 3. Se genera un GET request al servidor para obtener la información, proveniente de la tabla tax_data, que corresponde a todas las facturas del paciente, necesaria para el llenado inicial de la tabla.
Pos condición:

Listado historial de compras

Nombre : Historial de compras
Actor(es): Paciente

Descripción: Historial de compras del paciente en la tienda en línea. Permite búsqueda por fecha de compra, además de poder visualizar el archivo correspondiente.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none"> 1. El usuario ingresa el criterio de búsqueda en el textfield señalado con el ícono de lupa. 2. La tabla muestra resultados coincidentes con el criterio.
Flujo alternativo: <ol style="list-style-type: none"> 1. Se genera un GET request al servidor para obtener la información, proveniente de la tabla ticket, que corresponde a los tickets de compra del paciente, necesarios para el llenado inicial de la tabla.
Pos condición:

Mapa de casos COVID-19

Nombre : Mapa casos COVID-19
Actor(es): Paciente
Descripción: Mapa del mundo en el que se muestran los casos de personas contagiadas de COVID-19 a través de un punto rojo colocado en su ubicación. Además se visualizará un mapa de México, para listar los casos confirmados, sospechosos y negativos de las ciudades de cada estado
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none"> 1. El paciente da clic, arrastra y suelta el cursor sobre el mapa interactivo del mundo para visualizar los casos de COVID en el mundo. 2. El paciente da clic sobre alguno de los estados de la República Mexicana para visualizar la información de casos confirmados, sospechosos y negativos por ciudad, de esa entidad federativa.
Flujo alternativo: <ol style="list-style-type: none"> 1. Se genera un GET request al servidor para obtener la latitud y longitud de la ubicación de los pacientes que han sido confirmados con COVID-19, con fines de ubicación en el mapa mundial. 2. Se genera un GET request al servidor para obtener la cantidad de casos

confirmados, sospechosos y negativos de acuerdo a la entidad federativa que se ha presionado en el mapa de México.
Pos condición:

Catálogo de medicamentos

Nombre : Medicamentos
Actor(es): Paciente
Descripción: Vista de medicamentos disponibles en la tienda en línea. Búsqueda por: nombre del medicamento, tipo, sustancia activa, laboratorio y categoría. Posibilidad de añadirlo al carrito de compras presionando el botón correspondiente.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none"> 1. El paciente ingresa el criterio de búsqueda en el textfield señalado con el ícono de lupa. 2. La tabla muestra resultados coincidentes con el criterio. 3. El paciente presiona el botón agregar al carrito para posteriormente realizar la compra. 4. El paciente presiona el botón ver carrito de compras para visualizar los medicamentos que ha elegido.
Flujo alternativo: <ol style="list-style-type: none"> 1. Se genera un GET request a la API de Woocommerce para obtener la información de los productos, necesaria para el llenado inicial de la vista.
Pos condición:

Carrito de compras

Nombre : Carrito de compras
Actor(es): Paciente
Descripción: Vista de medicamentos agregados al carrito. Permite modificar la cantidad de productos a comprar o eliminar medicamentos de la lista, actualizando el total a pagar.

<p>Precondiciones:</p> <ol style="list-style-type: none"> 1. Para acceder al módulo, el paciente debe haberse logueado en el sistema previamente
<p>Flujo normal:</p> <ol style="list-style-type: none"> 1. El paciente modifica la cantidad de alguno de los medicamentos a comprar. 2. El paciente elimina de su lista de compras uno o varios de los medicamentos a comprar 3. El paciente presiona el botón generar para realizar su compra y recibir su ticket por medio de correo electrónico.
<p>Flujo alternativo:</p> <ol style="list-style-type: none"> 1. Se genera un POST request al servidor para almacenar la información de la compra en la tabla ticket. 2. Se genera un POST request al servidor para enviar el ticket de compra en la tabla ticket.
<p>Pos condición:</p> <p>Creación de nuevo registro en la tabla ticket.</p>

Listado de servicios

<p>Nombre : Servicios</p>
<p>Actor(es): Doctor</p>
<p>Descripción: Alta, baja y modificación de servicios brindados por un doctor</p>
<p>Precondiciones:</p> <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente
<p>Flujo normal:</p> <ol style="list-style-type: none"> 1. El doctor presiona el botón agregar, para añadir a su catálogo personal un servicio de los que se encuentran disponibles. 2. El doctor presiona el botón nuevo para crear un nuevo servicio, si este no se encuentra en la lista de servicios existentes, y posteriormente agregarlo a su catálogo personal. 3. El doctor presiona el botón con ícono de papelera para dar de baja un servicio de su catálogo personal.
<p>Flujo alternativo:</p> <ol style="list-style-type: none"> 1. Se genera un GET request al servidor para recuperar la información de los servicios que ofrece un doctor, proveniente de la tabla doctor_service. 2. Se genera un DELETE request al servidor para eliminar la relación de los servicios que ofrece un doctor, en la tabla doctor_service.

Pos condición:

Formulario agregar a mis servicios

Nombre : Agregar a mis servicios

Actor(es): Doctor

Descripción: Formulario desplegado en un dialog. Permite al doctor seleccionar un servicio de los existentes para agregarlo a su catálogo de servicios personal, configurando el precio del servicio.

Precondiciones:

1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente.
2. Para acceder al módulo, el doctor debe haber presionado el botón “Agregar a mis servicios”, ubicado en la ventana Servicios.

Flujo normal:

1. El doctor selecciona un servicio de los disponibles en la lista.
2. El doctor configura el costo a cobrar por el servicio que desea brindar.
3. El doctor presiona el botón cancelar si desea no guardar los cambios.
4. El doctor presiona el botón guardar para agregar el servicio a su catálogo personal.

Flujo alternativo:

1. Se genera un GET request al servidor para obtener la información de los servicios disponibles, proveniente de la tabla services y mostrarla en el listado de servicios.
2. Se genera un POST request al servidor para almacenar los datos de la nueva relación entre doctor y servicio.
3. Se genera un GET request al servidor para obtener los datos de los servicios de un doctor y con ella recargar la información de su catálogo personal.

Pos condición:

Se actualiza el listado de servicios ofrecidos por un doctor, incluyendo el nuevo servicio recién agregado.

Formulario agregar nuevo servicio

Nombre : Nuevo servicio

Actor(es): Doctor

Descripción: Formulario desplegado en un dialog. Permite al doctor crear un nuevo servicio en caso de que este no exista, y posteriormente agregarlo a su catálogo de servicios personal, configurando el precio del servicio.

<p>Precondiciones:</p> <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente. 2. Para acceder al módulo, el doctor debe haber presionado el botón “Nuevo”, ubicado en la ventana Servicios.
<p>Flujo normal:</p> <ol style="list-style-type: none"> 1. El doctor ingresa la descripción del nuevo servicio que desea brindar. 2. El doctor configura el costo a cobrar por el servicio que desea brindar. 3. El doctor presiona el botón cancelar si desea no guardar los cambios. 4. El doctor presiona el botón guardar para agregar el nuevo servicio a la lista de servicios y a su catálogo personal.
<p>Flujo alternativo:</p> <ol style="list-style-type: none"> 1. Se genera un POST request al servidor para almacenar los datos del nuevo servicio en la tabla services. 2. Se genera un POST request al servidor para almacenar los datos de la nueva relación entre doctor y servicio en la tabla doctor_service. 3. Se genera un GET request al servidor para obtener los datos de los servicios de un doctor y con ella recargar la información de su catálogo personal.
<p>Pos condición:</p> <p>Se actualiza el listado de servicios ofrecidos por un doctor, incluyendo el nuevo servicio recién creado y agregado.</p>

Consultas pendientes

Nombre : Consultas pendientes
Actor(es): Doctor
Descripción: Visualización de las consultas con estatus de atención pendiente, correspondientes a la especialidad del doctor en sesión
<p>Precondiciones:</p> <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente
<p>Flujo normal:</p> <ol style="list-style-type: none"> 1. El doctor dispone de una vista general de las consultas de su especialidad que no han sido atendidas. 2. El doctor presiona el ícono ver, para acceder a los detalles de la consulta.
<p>Flujo alternativo:</p> <ol style="list-style-type: none"> 1. Se genera un GET request al servidor para recuperar la información de las consultas pendientes con base en la especialidad del doctor en sesión, proveniente

de la tabla medical_consultation.
Pos condición:

Detalles de la consulta

Nombre : Detalles de la consulta
Actor(es): Doctor
Descripción: Visualización de la información completa de una consulta en específica.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none"> 1. El doctor dispone de una vista de la información completa de la consulta que ha seleccionado.
Flujo alternativo:
Pos condición:

Canalizar paciente a otro especialista

Nombre : Consultas pendientes
Actor(es): Doctor
Descripción: Canalización del paciente a otro doctor, seleccionando la especialidad de este.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente. 2. El doctor presiona el botón canalizar a otro especialista, en la ventana de detalles de la consulta. 3. El doctor presiona ACEPTAR para confirmar que la acción se llevará a cabo
Flujo normal: <ol style="list-style-type: none"> 1. El doctor elige la nueva especialidad a la cual desea canalizar al paciente. 2. El doctor elige al doctor de esa especialidad a quien desea canalizar el paciente.
Flujo alternativo: <ol style="list-style-type: none"> 1. Se genera un GET request al servidor para recuperar la información de las

<p>especialidades existentes en la tabla speciality.</p> <ol style="list-style-type: none"> 2. Se genera un GET request al servidor para recuperar la información de los doctores que pertenecen a esa especialidad. 3. Se genera un PUT request al servidor para modificar la especialidad de la consulta médica, en la tabla medical_consultation.
<p>Pos condición:</p> <p>Se canaliza al paciente a un nuevo especialista.</p>

Actualizar estatus COVID-19 del paciente

Nombre : Actualizar estatus COVID-19 del paciente
Actor(es): Doctor
Descripción: Marcar el paciente como caso confirmado, sospechoso o negativo de COVID-19
<p>Precondiciones:</p> <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente. 2. El doctor presiona el botón STATUS COVID-19, en la ventana de detalles de la consulta.
<p>Flujo normal:</p> <ol style="list-style-type: none"> 1. El doctor marca el estatus con alguna de las opciones disponibles en la lista: negativo, sospechoso, o confirmado. 2. El doctor presiona ACEPTAR para guardar los cambios.
<p>Flujo alternativo:</p> <ol style="list-style-type: none"> 1. Se genera un PUT request al servidor para modificar el campo status_covid del paciente a quien corresponde la consulta médica.
<p>Pos condición:</p> <p>Se marca al paciente como caso negativo, sospechoso o confirmado.</p>

Generar receta médica

Nombre : Receta médica
Actor(es): Doctor
Descripción: Visualización de la lista de medicamentos para generar la prescripción.
<p>Precondiciones:</p> <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente.

2. El doctor deberá ver los detalles de la consulta y presionar el botón recetar.
Flujo normal: <ol style="list-style-type: none"> 1. La lista de medicamentos ya prescritos se muestra inicialmente vacía. 2. El doctor presiona botón con el icono + para comenzar a agregar los medicamentos que desea prescribir. 3. El doctor podrá recetar cuantos medicamentos sean necesarios agregandolos a la lista de la receta presionando el botón con ícono +. 4. El doctor podrá generar un pdf con la lista de medicamentos, y enviar esta receta al correo electrónico del paciente, presionando el botón GENERAR.
Flujo alternativo: <ol style="list-style-type: none"> 1. Se genera un POST request al servidor para almacenar la información de la receta en la tabla prescription. 2. Se genera un POST request al servidor para enviar el link de descarga de la receta al correo del paciente.
Pos condición: <ol style="list-style-type: none"> 1. Se crea un nuevo registro en prescription con los datos de la receta. 2. Se envía el link de la receta por correo para que el paciente pueda visualizarla y/o descargarla

Recetar medicamento

Nombre : Recetar medicamento
Actor(es): Doctor
Descripción: Formulario para agregar un medicamento, permite seleccionar el nombre del medicamento, despliega automáticamente el tipo, sustancia activa, laboratorio y costo.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente. 2. El doctor deberá ver los detalles de la consulta y presionar el botón recetar. 5. El doctor deberá presionar el botón con icono + en la ventana de Generar receta para comenzar a agregar medicamentos a la receta
Flujo normal: <ol style="list-style-type: none"> 1. El doctor ingresa el nombre del medicamento que desea recetar. 2. Los detalles del medicamento se listan automáticamente. 3. El doctor escribe las instrucciones para el medicamento. 6. El doctor presiona el botón cancelar en caso de no querer agregar el medicamento, o presiona el botón ACEPTAR para agregar el medicamento a la

lista de la receta.
Flujo alternativo: 3. Se genera un GET request al servidor para recuperar la información de los medicamentos.
Pos condición: Se agrega el medicamento al listado de la receta.

Generar factura por los servicios de una consulta

Nombre : Generar factura
Actor(es): Doctor
Descripción: Visualización de la lista de servicios para generar la factura de la consulta.
Precondiciones: 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente. 2. El doctor deberá ver los detalles de la consulta y presionar el botón recetar. 3. El doctor deberá haber atendido la consulta y ya haber asignado prescripción médica a la misma.
Flujo normal: 1. La lista de conceptos a cobrar se muestra inicialmente vacía. 2. El doctor presiona botón con el icono + para comenzar a agregar los servicios por los cuales ha de cobrar en la factura. 3. El doctor podrá agregar cuantos conceptos sean necesarios de acuerdo a los servicios brindados en la consulta, presionando el botón con ícono +. 4. El doctor podrá generar un pdf con la lista de servicios, y enviar esta factura al correo electrónico del paciente, presionando el botón GENERAR.
Flujo alternativo: 4. Se genera un POST request al servidor para almacenar la información de la factura en la tabla bill. 5. Se genera un POST request al servidor para enviar el link de descarga de la factura al correo del paciente.
Pos condición: 3. Se crea un nuevo registro en bill con los datos de la factura. 4. Se envía el link de la factura por correo para que el paciente pueda visualizarla y/o descargarla

Agregar concepto de cobro a la factura

Nombre : Agregar concepto
Actor(es): Doctor
Descripción: Formulario para agregar un concepto de cobro, permite seleccionar el nombre del servicio, despliega automáticamente el costo del mismo.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente. 2. El doctor deberá presionar el botón con icono + en la ventana de Generar factura para comenzar a agregar servicios a la factura
Flujo normal: <ol style="list-style-type: none"> 1. El doctor ingresa el nombre del servicio que desea cobrar. 2. Los detalles del servicio (costo) se listan automáticamente. 3. El doctor presiona el botón cancelar en caso de no querer agregar el servicio, o presiona el botón ACEPTAR para agregar el servicio a la lista de la factura.
Flujo alternativo: <ol style="list-style-type: none"> 1. Se genera un GET request al servidor para recuperar la información de los servicios.
Pos condición: Se agrega el servicio al listado de la factura.

Listado consultas previas

Nombre : Consultas Previas
Actor(es): Doctor
Descripción: Historial de las consultas que han sido atendidas por el doctor en sesión. Permite buscar por: fecha de consulta, paciente y síntomas; ver la receta y la factura correspondientes a cada cita.
Precondiciones: <ol style="list-style-type: none"> 1. Para acceder al módulo, el doctor debe haberse logueado en el sistema previamente
Flujo normal: <ol style="list-style-type: none"> 1. El doctor ingresa el criterio de búsqueda en el textfield señalado con el ícono de lupa. 2. La tabla muestra resultados coincidentes con el criterio. 3. El doctor presiona el botón ver receta para abrir el PDF de la receta médica. 4. El doctor presiona el botón ver factura para visualizar la factura de la consulta médica en cuestión

Flujo alternativo:

1. Se genera un GET request al servidor para obtener la información, proveniente de la tabla medical_consultation, así como de las tablas prescription y bill con base al id de la consulta.

Pos condición:

1. Se abre el link correspondiente a la receta médica, para su visualización o descarga.
2. Se abre el link correspondiente a la factura, para su visualización o descarga.