

22-10-2020



# Universidad Autónoma de Querétaro

Facultad de Informática

Jessica Karina Del Angel Calva

DIPLOMADO DE DESARROLLO DE APLICACIONES WEB 2020-1

NOMBRE DEL PROYECTO: NOVAMEDIK

ENTREGA DEL PROYECTO: 23-10-2020

# ÍNDICE

1. INTRODUCCIÓN .....	4
<b>1.1 Presentación y objetivos</b> .....	4
<b>1.2 Contexto</b> .....	4
<b>1.3 Planteamiento del problema</b> .....	4
<b>1.4 Estructura del documento</b> .....	5
2. ESPECIFICACIÓN DE REQUISITOS .....	5
<b>2.1 Introducción</b> .....	5
2.1.1 Propósito .....	5
2.1.2 Ámbito .....	5
2.1.3 Definiciones, acrónimos y abreviaturas .....	6
2.1.4 Visión global .....	6
<b>2.2 Descripción general</b> .....	6
2.2.1 Perspectiva del producto .....	6
2.2.2 Características del usuario .....	6
2.2.3 Restricciones generales .....	8
2.2.4 Supuestos y dependencias .....	8
<b>2.3 Requisitos específicos</b> .....	8
2.3.1 Requerimientos de interfaces externos .....	8
2.3.2 Interfaces de usuarios .....	8
2.3.2.1 Interfaces hardware .....	9
2.3.2.2 Interfaces software .....	9
2.3.2.3 Interfaces de comunicaciones .....	9
Descripción .....	10
Instalación .....	10
Funcionalidades .....	<b>¡Error! Marcador no definido.</b>
Métodos del servicio web .....	12
Usuarios .....	12

Entrar.....	12
<b>Crear usuario</b> .....	13
Obtener usuarios .....	14
Actualizar contraseña de usuario .....	16
Crear un paciente .....	17
2.3.3 Obligaciones del diseño .....	20
2.3.3.1 Estándares cumplidos.....	20
2.3.3.2 Limitaciones hardware .....	20
2.3.4 Atributos .....	20
2.3.4.1 Seguridad.....	20
2.3.4.2 Portabilidad.....	20
2.3.4.3 Otros requerimientos.....	20
3. ANÁLISIS.....	22
3.1 Modelo del Proceso del Negocio .....	22
4 DISEÑO .....	22
4.1 Capa de persistencia o datos .....	22
5 IMPLEMENTACIÓN .....	23
5.1 Tecnologías utilizadas en el desarrollo del proyecto .....	24
5.1.1 HTML.....	24
5.1.2 CSS .....	24
5.1.3 JAVASCRIPT .....	25
5.1.4 SQL.....	25
5.1.5 MYSQL.....	25
5.1.6 NODE.JS.....	26
5.1.7 JQUERY.....	27
5.1.8 BOOTSTRAP .....	27
5.1.9 BCRIPT.....	28
5.1.10 HEROKU .....	29
5.1.11 JSON WEB TOKEN .....	29
5.1.12 EXPRESS.JS .....	29
5.1.13 PHPMYADMIN .....	30
5.2 Descripción del proyecto.....	30



*Universidad Autónoma de Querétaro*  
*Diplomado de Desarrollo de Aplicaciones Web 2020-1*

5.2.3	Capa de presentación .....	30
5.2.2	Capa de persistencia o de datos .....	33
<b>5.3</b>	<b>Reglas de Negocio .....</b>	<b>34</b>
6	CONCLUSIÓN .....	36
<b>6.1</b>	<b>Valoración personal del trabajo realizado .....</b>	<b>36</b>
7	BIBLIOGRAFÍA .....	36

# 1. INTRODUCCIÓN

## 1.1 Presentación y objetivos

En este proyecto de ampliación tecnológica describe el trabajo realizado en el proyecto final del Diplomado de Desarrollo de aplicaciones Web. El proyecto consiste en el desarrollo del sitio web para un Hospital (NOVAMEDIK).

El objetivo de dicha web, busca implementar distintos tipos de soluciones tecnológicas que pretenden mejorar la calidad del servicio y a su vez la calidad de la atención médica a los pacientes. Los tres ejes que se buscan mejorar son: atención médica, logística, y organización.

La web es accesible desde cualquier navegador por el internet, en cuanto a los usuarios, la página está hecha para cuatro tipos de usuarios, que son: el administrador, el doctor, el enfermero y el laboratorista). Posteriormente cada usuario cuenta con sus respectivos módulos en donde llevaran a cabo cada función que les corresponden.

## 1.2 Contexto

El proyecto ha sido realizado para el hospital Novamedik, para el desarrollo de la aplicación se realizó diversas reuniones con el personal del hospital, quienes serán los responsables de usar el sistema. Tomando en cuenta la mayoría de las decisiones de estética y de las funciones de la aplicación fueron tomadas junto con el administrador de dicho Hospital, que además proporciono el flujo que hace cada usuario.

## 1.3 Planteamiento del problema

Novamedik necesita un sistema en donde sea más eficiente, este automatizado y sobre todo mejorar la interacción del personal médico con los pacientes, de los cuales existen dos tipos: los internados por alguna situación médica que requiere observación constante y mientras los transitorios, que son los pacientes que solamente se presentan a citas en las clínicas de salud.

## 1.4 Estructura del documento

El presente documento está dividido en una serie de capítulos que corresponden, básicamente, a las distintas etapas que conforman el proceso de desarrollo del proyecto. Estas etapas han sido:

- **Especificación de requisitos:** Se redactó de una manera global una visión del proyecto donde se señala los requisitos a cumplir. La finalidad de este proyecto es plasmar el acuerdo entre el desarrollador y el cliente acerca de las funcionalidades del proyecto.
- **Análisis:** Se realizó el modelado del proceso del negocio, por lo tanto, el modelo ayuda a visualizar cómo será el proceso del sistema. En esta etapa se especifica qué debe hacer la aplicación, pero no cómo debe hacerlo.
- **Diseño:** Se utilizó el modelo obtenido durante el análisis para darle continuidad y utilizarlo para el entorno web. Se diseñaron los ejes que se buscan para mejorar son: atención médica y organización.
- **Implementación:** Se utilizaron los elementos obtenidos en el diseño para permitir la elaboración del prototipo funcional, es decir, que puede ser puesto en marcha y sometido a pruebas. Para ello se consideraron las diversas tecnologías que han intervenido en la elaboración de dicho producto.

# 2.ESPECIFICACIÓN DE REQUISITOS

## 2.1 Introducción

### 2.1.1 Propósito

El propósito de la especificación de requisitos es llevar a cabo los requerimientos que debe tener el sistema que se va a desarrollar y a mejorar la funcionalidad de cada usuario que lo ocupará.

### 2.1.2 Ámbito

El desarrollo del sitio web está orientado a ofrecer diferentes tipos de soluciones tecnológicos, se pretende mejorar la calidad del servicio y a su vez la calidad de la atención médica a los pacientes que se tiene día con día. No obstante, a mejorar el flujo que maneja el hospital.

### 2.1.3 Definiciones, acrónimos y abreviaturas

- ❖ **Sitio web:** Conjunto de archivos electrónicos y páginas web referentes a un tema en particular que incluye una página inicial de bienvenida, con un nombre de dominio y dirección en Internet específicos.
- ❖ **Interfaz:** Parte del programa informático que permite el flujo de información entre varias aplicaciones o entre el propio programa y el usuario.
- ❖ **Navegador:** Permite al usuario recuperar y visualizar páginas web a través de Internet.
- ❖ **Servidor web:** Se trata de un programa que implementa el protocolo HTTP (HyperText Transfer Protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.
- ❖ **REST:** es un estilo de arquitectura de software que se utiliza para describir cualquier interfaz entre diferentes sistemas que utilice HTTP para comunicarse. Este término significa REpresentational State Transfer (transferencia de estado representacional), lo que quiere decir que, entre dos llamadas cualquiera, el servicio no guarda los datos.

### 2.1.4 Visión global

A continuación, se realizará la descripción general del sistema desarrollado con sus funciones, características del usuario, restricciones y supuestos.

## 2.2 Descripción general

### 2.2.1 Perspectiva del producto

La aplicación desarrollada pretende mejorar la calidad en tanto en interactuar con el paciente, dar más agilidad a la hora de agendar citas para los dos tipos de pacientes, con respecto si es paciente interno, se le dará la atención que requiera.

### 2.2.2 Características del usuario

A continuación, se muestran las características según el tipo de usuario que lo requiera.

ADMINISTRADOR:

- ✚ *Internar pacientes:* el administrador podrá internar a los pacientes que son internos, ya que los transitorios solamente se presentan a citas en las clínicas de salud.
- ✚ *Crear paciente:* podrá crear un paciente desde cero, ya sea paciente interno o paciente transitorio, tomando en cuenta los datos personales de cada paciente.
- ✚ *Asignar personal a habitación:* podrá asignar personal a habitación, siempre y cuando sea un paciente internado, ya que recibe una habitación numerada fija, en el cual permanecerá solo hasta ser dado de alta o transferido a alguna otra unidad o hospital.
- ✚ *Asignar personal a consultorio:* podrá asignar personal a consultorio, siempre y cuando sea para el paciente transitorio, ya que se le asigna en una fecha y hora programada por su cita.
- ✚ *Transferir paciente:* podrá transferir un paciente interno a otra unidad u hospital.
- ✚ *Dar de alta personal:* solo podrá dar de alta a los pacientes internos.

#### LABORATORISTA:

- ✚ *Generar exámenes médicos:* el laboratorista solo podrá generar los exámenes médicos de los pacientes internos.
- ✚ *Consultar resultados médicos:* solamente podrá consultar los resultados médicos que el mismo laboratorista ha expedido.

#### ENFERMERO:

- ✚ *Tener pacientes asignados:* el enfermero podrá atender a los pacientes que se le ha asignado.
- ✚ *Visualizar historial del paciente:* el enfermero solo podrá visualizar el historial de aquellos pacientes que tiene a su cargo.
- ✚ *Tener habitaciones asignadas:* el enfermero podrá presentarse a las habitaciones asignadas, donde tiene que cumplir con sus responsabilidades.
- ✚ *Visualizar exámenes médicos:* así como visualizar el historial, es prácticamente lo mismo de visualizar los exámenes médicos de sus pacientes asignados.
- ✚ *Asignar cita a paciente transitorio:* el será encargado de asignarle las citas a los pacientes transitorios, solamente a sus pacientes.

#### DOCTOR:



- ✚ *Tener pacientes asignados:* podrá atender a los pacientes que se le han asignado.
- ✚ *Visualizar historial del paciente:* podrá visualizar el historial de sus pacientes internos y pacientes transitorios.
- ✚ *Tener habitaciones asignadas:* tendrá a su cargo las habitaciones que se le asigno para atender a los pacientes.
- ✚ *Tener consultorio asignado:* podrá tener un consultorio asignado para atender a los pacientes transitorios.
- ✚ *Solicitar exámenes médicos:* tiene y puede solicitar los exámenes médicos de sus pacientes.
- ✚ *Visualizar exámenes médicos:* puede visualizar los exámenes médicos de sus pacientes.
- ✚ *Consultar resultados médicos:* puede consultar los resultados médicos de los pacientes que se le asignaron.
- ✚ *Asignar un diagnóstico:* puede asignar el diagnostico a sus pacientes internos y transitorios.
- ✚ *Transferir paciente:* el doctor puede transferir de área al paciente.
- ✚ *Asignar cita a paciente transitorio:* podrá asignar una cita al paciente transitorio para otra valoración, o a su vez darle seguimiento a lo que el paciente tiene.

### 2.2.3 Restricciones generales

Al tratarse de una aplicación web, se requiere un ordenador con un navegador convencional y una conexión a Internet básica.

### 2.2.4 Supuestos y dependencias

La aplicación desarrollada, trabaja al margen de cualquier hardware o software. La dependencia importante es que está relacionada con el servidor web (HEROKU), que permite alojar el sistema, este es capaz de soportar html, css, javaScript, sql, node.js, juery, boststrap, bcrypt, jsonwebtoken y express.js.

## 2.3 Requisitos específicos

### 2.3.1 Requerimientos de interfaces externos

### 2.3.2 Interfaces de usuarios

A continuación, se muestra una captura de pantalla (figura 1) en la que se especifican las zonas que conforman la interfaz gráfica con la que interactuará el usuario que haga uso del portal web.

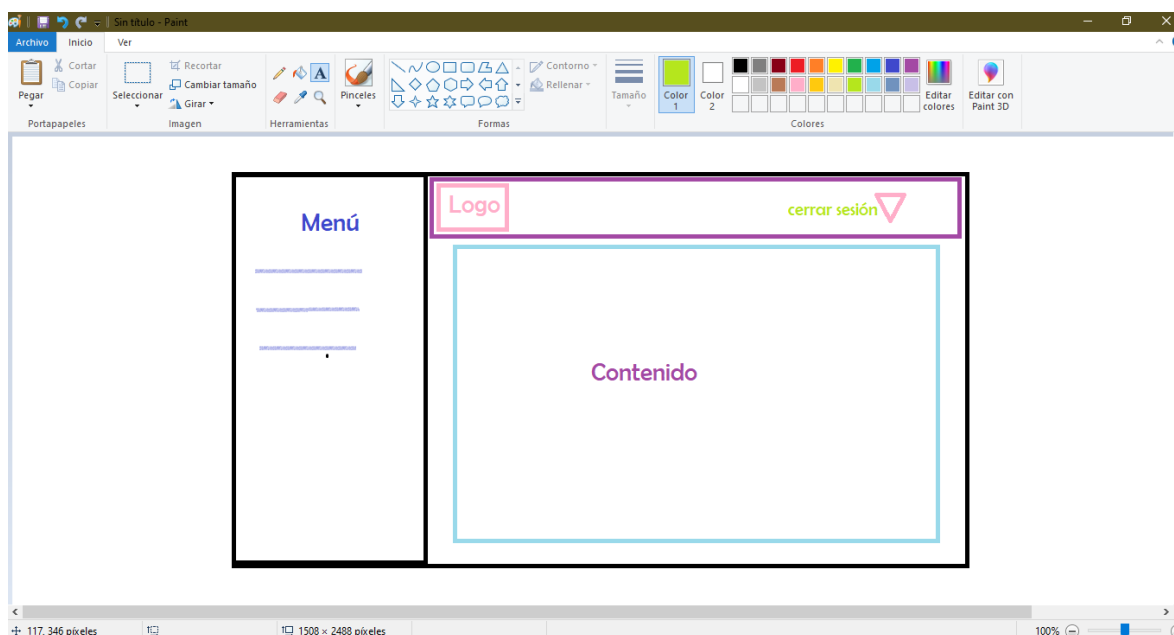


Figura 1.

### 2.3.2.1 Interfaces hardware

Al tratarse de una aplicación web, se podrá visualizar sobre cualquier sistema operativo.

### 2.3.2.2 Interfaces software

La aplicación funcionará en cualquier máquina con un navegador web y conexión a Internet.

### 2.3.2.3 Interfaces de comunicaciones

Las comunicaciones se efectuarán siguiendo el protocolo HTTP.

El siguiente link, en donde se encuentra la ruta del servidor, en donde está alojado el proyecto.

<https://novamedik.herokuapp.com>

## Descripción

Este proyecto es una aplicación web que utiliza Node.js como servidor para alojar tanto la página web como la API Rest que se comunica con la base de datos. No es necesario que la aplicación web y la base de datos estén en el mismo servidor, aunque es requisito que se modifiquen las variables de entorno para que el servidor pueda saber de dónde llamar la base de datos.

La aplicación web no está limitada a correr en un sistema operativo en específico; puede correr en cualquier sistema operativo donde se pueda instalar Node.js.

## Instalación

Es necesario tener instalado Node.js dentro del equipo que aloje la aplicación web. El instalador de Node.js instala también la aplicación NPM (Node.js Package Manager), que es la herramienta que descargará localmente (en el proyecto) las dependencias de la aplicación, además de ejecutar los comandos para iniciar la aplicación.

Otra herramienta que puede usarse en el proyecto es Yarn, otro gestor de paquetes de Node.js, aunque esta alternativa queda como sugerencia para el desarrollador.

Para hacer la descarga de las dependencias ejecutamos el siguiente comando

```
$ npm install
```

Si se está trabajando en un entorno de desarrollo, es necesario crear un archivo .env que será donde estarán almacenadas las variables de entorno que el servidor usará. Podemos simplemente copiar y usar como base el archivo .env.example dentro del proyecto:

```
$ cp .env.example .env
```

Cambié los valores del archivo para estén acordes a los datos de la base de datos, llave secreta, puerto que el servidor escuchará y contraseña del primer administrador.

Dentro del archivo package.json se especifican dos comandos:

- start: Inicia el servidor, se sugiere para producción
- dev: Es el comando sugerido para los desarrolladores

Sea cual sea el caso se debe ejecutar con alguno de los siguientes comandos:



*Universidad Autónoma de Querétaro*  
*Diplomado de Desarrollo de Aplicaciones Web 2020-1*

- Para desarrolladores

```
$ npm run dev
```

- Para producción

```
$ npm start
```

## Métodos del servicio web

### Usuarios

#### Entrar

**Post** /api/login

Método que genera el token para poder operar en el sistema. Cualquier usuario que desee usar el sistema puede hacer uso de este método

#### Encabezados

Encabezado	Tipo de dato	Contenido
Content-Type	string	application/x-www-form-urlencoded

#### Parámetros del cuerpo

Parámetro	Tipo de dato	Descripción
usuarioid*	string	Id del usuario
contrasena*	string	Contraseña del usuario

\*: requerido

#### Respuestas

##### 200: OK

Se obtuvo correctamente el token

```
{
  "ok": true,
  "datos": {
    "id": "Id del usuario",
    "nombres": "Nombres del usuario",
    "apellidos": "Apellidos del usuario"
  },
  "token": "Token generado por el servicio web"
}
```

##### 400: Mala solicitud

Los datos enviados son incorrectos

```
{
  "message": "Credenciales incorrectas"
}
```

##### 500: Error del servidor

Error interno del servidor o de la base de datos

```
{
```

```
{
  "message": "Error del servidor"
}
```

### Crear usuario

**Post** /api/{tipo de usuario}

Métodos para crear un usuario con cualquiera de los roles que su endpoint indique. Este método solo puede ser llamado por un administrador. **admin** para crear un administrador, **doctor** para crear un doctor, **enfermero** para crear un enfermero, **laboratorista** para crear un laboratorista

#### Encabezados

Encabezado	Tipo de dato	Contenido
Content-Type	string	application/x-www-form-urlencoded
Authorization*	string	Bearer token

\*: requerido

#### Parámetros del cuerpo

Parámetro	Tipo de dato	Descripción
nombres*	string	Nombres del usuario
apellidos*	string	Apellidos del usuario
contrasena*	string	Contraseña del usuario

\*: requerido

#### Respuestas

**200: OK**

Se creo correctamente el usuario doctor

```
{
  "message": "Se creo el usuario"
}
```

**400: Mala solicitud**

Los datos enviados son incorrectos

```
{
  "message": "Ya existe un usuario registrado con esos datos"
}
```

**500: Error del servidor**

Error interno del servidor o de la base de datos

```
{
```

```
"message": "No se pudo crear el usuario"
}
```

## Obtener usuarios

**Get** /api/usuarios

Método para obtener los usuarios registrados en el sistema. Este método puede ser llamado por el administrador y por el enfermero. El enfermero muestra solo los doctores; dentro del cliente (frontend) el enfermero necesita el listado de doctores disponibles para crear una consulta. El administrador muestra en una lista todos los usuarios existentes en el sistema.

### Encabezados

Encabezado	Tipo de dato	Contenido
Content-Type	string	application/x-www-form-urlencoded
Authorization*	string	Bearer token

\*: requerido

### Respuestas

#### 200: OK

Se hizo la solicitud correctamente. El servidor enviará un arreglo de objetos con cada tipo de usuario, y cada objeto tendrá tres campos: id, nombres y apellidos.

La respuesta al enfermero:

```
{
  "doctor": [{
    "id": "Id del doctor",
    "nombres": "Nombres del doctor",
    "apellidos": "Apellidos del doctor"
  }]
}
```

La respuesta al administrador:

```
{
  "doctor": [{
    "id": "Id del doctor",
    "nombres": "Nombres del doctor",
    "apellidos": "Apellidos del doctor"
  }],
  "enfermero": [{
```

```
{
  "id": "Id del enfermero",
  "nombres": "Nombres del enfermero",
  "apellidos": "Apellidos del enfermero"
},
{laboratorista": [{
  "id": "Id del laboratorista",
  "nombres": "Nombres del laboratorista",
  "apellidos": "Apellidos del laboratorista"
}],
administrador": [{
  "id": "Id del administrador",
  "nombres": "Nombres del administrador",
  "apellidos": "Apellidos del administrador"
}],
}
```

#### 403: Acceso no autorizado

Las credenciales que se enviaron no concuerdan con los requisitos del método

```
{
  "message": "Acceso no autorizado"
}
```

#### 500: Error del servidor

Error interno del servidor o de la base de datos

```
{
  "message": "Error del servidor"
}
```



### Actualizar contraseña de usuario

**Put** /api/usuario/:usuariold

Método para cambiar la contraseña de un usuario. El administrador puede cambiar la contraseña de cualquier usuario. Este método también puede ser llamado por el mismo usuario para cambiar su contraseña

#### Parámetros de la solicitud

Parámetro	Tipo de dato	Descripción
usuariold*	string	Id del usuario

\*requerido

#### Encabezados

Encabezado	Tipo de dato	Contenido
Content-Type	string	application/x-www-form-urlencoded
Authorization*	string	Bearer token

\*: requerido

#### Parámetros del cuerpo

Parámetro	Tipo de dato	Descripción
contrasena*	string	Contraseña nueva

\*: requerido

#### Respuestas

200: OK

Se hizo el cambio correctamente.

```
{
  "message": "Datos cambiados correctamente"
}
```

400: Mala solicitud

Los datos enviados son incorrectos

```
{
  "message": "Tipo de usuario incorrecto"
}
```

500: Error del servidor

Error interno del servidor o de la base de datos

```
{
```

```
"message": "Error del servidor"
}
```

## Pacientes

### Crear un paciente

**Post** /api/paciente

Método para crear un paciente nuevo. Este método solo puede ser llamado por un administrador.

#### Encabezados

Encabezado	Tipo de dato	Contenido
Content-Type	string	application/x-www-form-urlencoded
Authorization*	string	Bearer token

\*: requerido

#### Parámetros del cuerpo

Parámetro	Tipo de dato	Descripción
nombres*	string	Nombres del paciente
apellidos*	string	Apellidos del paciente
fechaNacimiento*	string	Fecha de nacimiento del paciente
lugarNacimiento*	string	Lugar de nacimiento del paciente
sexo*	string	Sexo del paciente
curp*	string	CURP del paciente
grupoSanguineo*	string	Tipo de sangre del paciente
email	string	Correo electrónico para contactarse del paciente
telefono*	string	Número telefónico para contactarse con el paciente
enfermedadesPreexistentes	string	Antecedentes personales patológicos del paciente
alergias	string	Alergias del paciente
direccion*	string	Domicilio (calle, número, colonia y municipio) donde reside el paciente
contactoNombre*	string	Nombre completo del contacto de emergencias del paciente
contactoDireccion*	string	Domicilio del contacto de emergencias del paciente
contactoTelefono*	string	Número de teléfono del contacto de emergencias del paciente
contactoEmail	string	Correo electrónico del contacto de emergencias del paciente

\*: requerido

### *Respuestas*

200: OK

Se creo correctamente el paciente

```
{  
  "message": "Paciente creado"  
}
```

400: Mala solicitud

Los datos enviados son incorrectos

```
{  
  "message": "Ya existe un paciente con esos datos"  
}
```

500: Error del servidor

Error interno del servidor o de la base de datos

```
{  
  "message": "No se pudo crear el paciente"  
}
```

**Post** /api/turno

Método para asignar turnos a los doctores o enfermeros, y al mismo tiempo generar asignar habitaciones de guardia.

**Post** /api/internar

Método para internar un paciente ya registrado en la base de datos.

**Put** /api/paciente

Método para dar de alta un paciente internado.

**Post** /api/habitacion

Método para crear una habitación en un área específica

**Get** /api/mis-habitaciones

Método para que un doctor o un enfermero pueda ver sus habitaciones asignadas.

**Get** /api/habitacion

Método para que el administrador pueda ver qué habitaciones hay disponibles.

**Get** /api/habitacion/:area

Método para que el administrador pueda ver qué habitaciones hay disponibles en un área. La URL puede ser escrita de cualquiera de las siguientes dos formas:

```
https://novamedik.herokuapp.com/api/habitacion/medicina interna  
https://novamedik.herokuapp.com/api/habitacion/medicina-interna
```

**Post** /api/consulta

Método para que el doctor o enfermero pueda crear una consulta nueva.

**Get** /api/consultas

Método para que el doctor vea sus consultas anteriores y próximas.

**Post** /api/consultorio

Método para crear un consultorio nuevo.

**Get** /api/consultorio

Método para que el doctor vea su consultorio asignado.

**Put** /api/consultorio/:consultoriold

Método para asignar un doctor a un consultorio.

**Get** /api/pacientes

Método para que los doctores y enfermeros vean sus pacientes asignados.

**Get** /api/paciente/:patientId

Método para obtener la información de un paciente en específico. Se debe escribir en la URL el ID del paciente.

**Get** /api/examenes

Método para obtener las solicitudes pendientes de los exámenes.

**Get** /api/examenes/:patientId

Método para obtener los exámenes de un paciente en específico.

**Get** /api/historial/:patientId

Método para obtener el historial clínico de un paciente

### 2.3.3 Obligaciones del diseño

#### 2.3.3.1 Estándares cumplidos

Se han intentado cumplir los estándares de cualquier web con acceso seguro, creando un sistema de autenticación para que nadie pueda acceder al sistema, se utilizara las contraseñas que sea han agregado para los tipos de usuarios, quienes son los que utilizaran el sistema. El idioma elegido para la presentación de las páginas ha sido en latino.

#### 2.3.3.2 Limitaciones hardware

Al tratarse de una aplicación web no se requiere un hardware específico. El servidor que albergará la base de datos del sistema deberá permanecer conectado a Internet las 24 horas, puesto que este host será quien atienda las peticiones de lectura y escritura de los usuarios que accedan.

### 2.3.4 Atributos

#### 2.3.4.1 Seguridad

La seguridad es un componente fundamental en el portal. La administración del sitio web está sujeta a la identificación satisfactoria de cada uno de los roles registrados en la aplicación, de forma que ningún usuario pueda acceder. Para asegurar la identidad del administrador junto con los demás roles se requerirá un nombre de usuario y contraseña que autenticará a estos en la aplicación. La información acerca de la cuenta se guardará en la base de datos. El proceso de login o autenticación llevará al administrador y a los otros roles donde podrán acceder al sitio web.

#### 2.3.4.2 Portabilidad

La aplicación ha sido diseñada con tecnología libre, luego podrá ser soportada por cualquier plataforma y sistema operativo. Por lo mismo, se podrá acceder a ella desde cualquier navegador.

#### 2.3.4.3 Otros requerimientos

El portal usa una base de datos MySQL donde se almacena toda la información referida a los pacientes, doctores, los exámenes médicos, etc. Las consultas a la base de datos se realizan por parte del servidor web mediante su API de acceso a bases de datos MySQL.

Por lo tanto, la aplicación web es una API Rest, se ha comentado que, para poder introducir datos en la base de datos:

- ✓ Internar paciente - DOCTOR y ADMINISTRADOR
- ✓ Dar de alta al paciente - ADMINISTRADOR
- ✓ Crear paciente – ADMINISTRADOR
- ✓ Crear consultorio - ADMINISTRADOR
- ✓ Crear habitación - ADMINISTRADOR
- ✓ Crear usuario (en las rutas se define qué tipo de usuario se va a crear) - ADMINISTRADOR
- ✓ Crear turno de usuario (aquí se definen las habitaciones que cubrirá el personal médico) - ADMINISTRADOR
- ✓ Asignar personal a consultorio - ADMINISTRADOR
- ✓ Asignar consulta a paciente transitorio - DOCTOR y ENFERMERO
- ✓ Comentar consulta - DOCTOR
- ✓ Crear exámenes médicos (es la solicitud de exámenes médicos) - DOCTOR
- ✓ Crear resultados de exámenes médicos - LABORATORISTA
- ✓ Crear comentario de resultado médico - LABORATORISTA y DOCTOR
- ✓ Transferir paciente ejecuta dos queries, un UPDATE y un INSERT
- ✓ Transferir paciente - ADMINISTRADOR y DOCTOR

Cada vez que se manda a llamar poolQuery, es decir, es una solicitud a la base de datos y una función personalizada, poolQuery está definida en utils/index.js, va a regresar un ok puede ser falso o verdadero, un mensaje de que se hizo la solicitud, y el objeto de respuesta de la ejecución.

```
{  
  ok: true,  
  message: 'Se hizo la solicitud',  
  response: // Lo que mande MySQL  
}
```

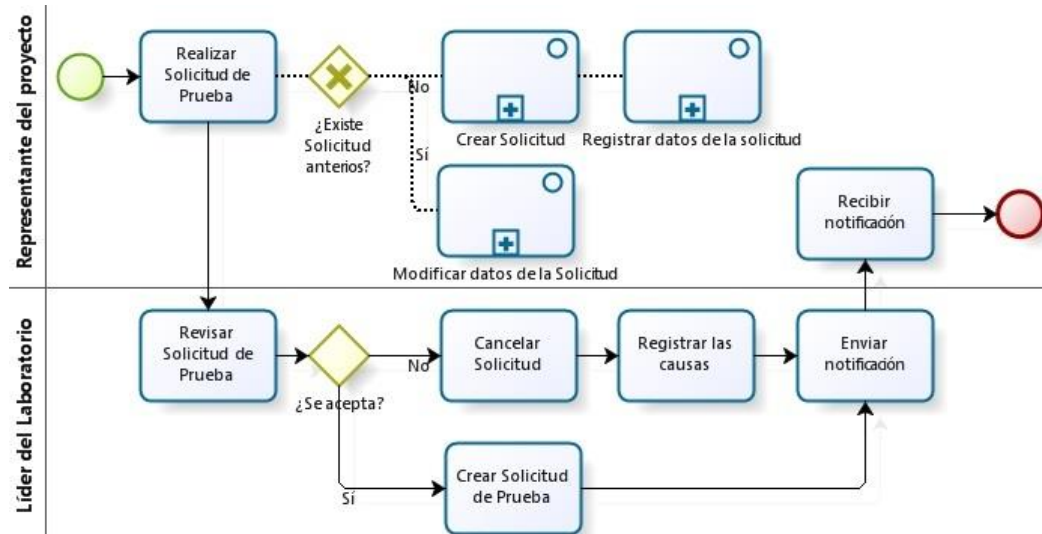
En las solicitudes HTTP (sea cual sea la ruta), se están enviando cabeceras, que en este caso solo envía el token, si no se envía el token, la aplicación web responde con un error.

Mentira, cuando no hay token se manda el 403 con el mensaje de "Acceso no autorizado".

Del lado del cliente JQuery hace la solicitud mediante su método ajax y siempre se envía el token.

## 3. ANÁLISIS

### 3.1 Modelo del Proceso del Negocio



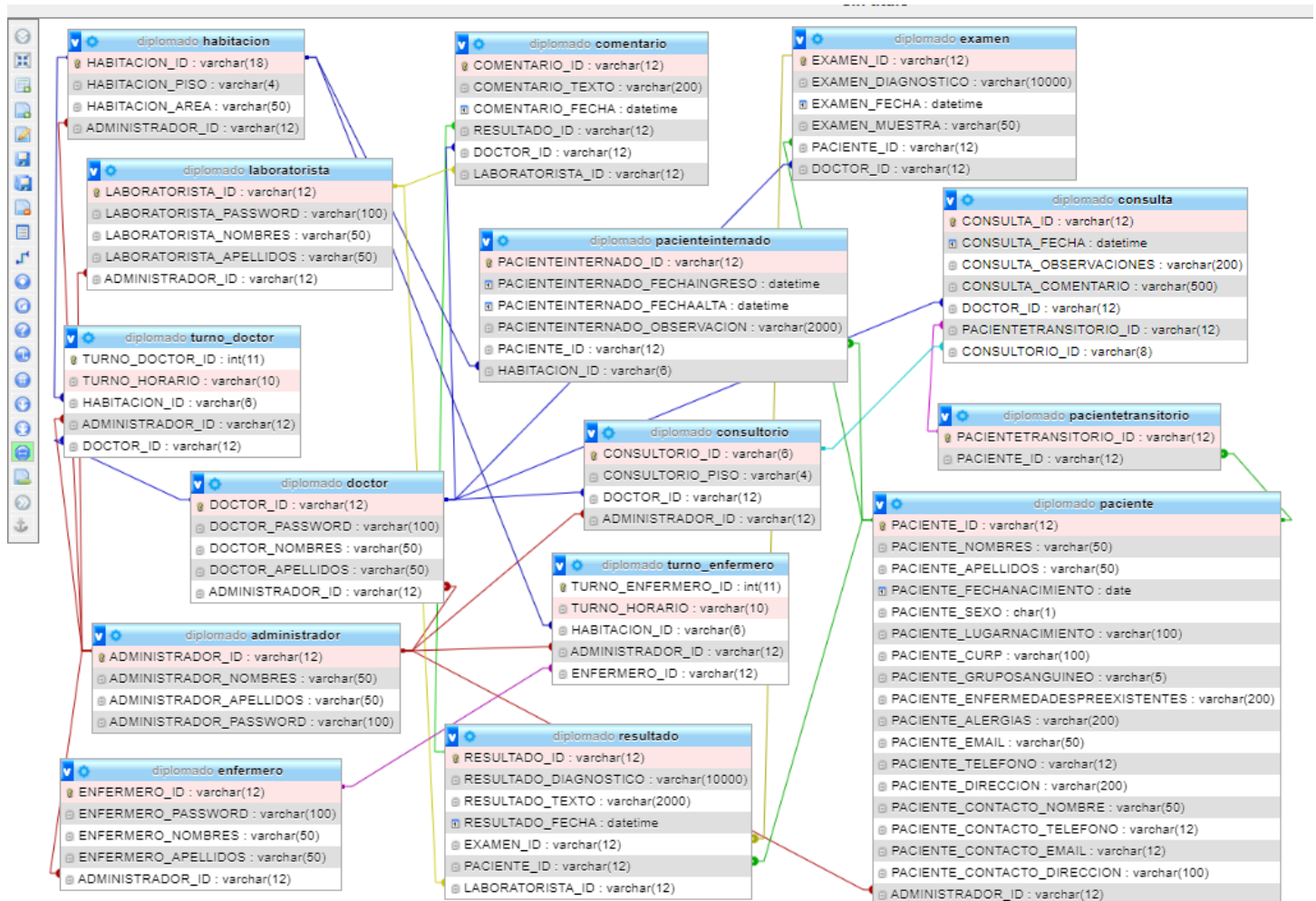
## 4 DISEÑO

### 4.1 Capa de persistencia o datos

En el proyecto la capa de persistencia se corresponde con la base de datos de la aplicación y las distintas tablas que la conforman. Estas son: administrador, laboratorista, doctor, paciente.

Tomando en cuenta que la base de datos en heroku está alojada en el siguiente link:

<https://remotemysql.com/#about>



## 5 IMPLEMENTACIÓN



## 5.1 Tecnologías utilizadas en el desarrollo del proyecto

### 5.1.1 HTML

HTML, siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>).

El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium, más conocido como W3C.

Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo. El propio W3C define el lenguaje HTML como "un lenguaje reconocido universalmente y que permite publicar información de forma global".

Por convención, los archivos de formato HTML usan la extensión .htm o .html.

### 5.1.2 CSS

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. En este último podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style".

Las ventajas de utilizar CSS (u otro lenguaje de estilo) son:

- Control centralizado de la presentación de un sitio web completo, con lo que se agiliza de forma considerable la actualización del mismo.

- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web remoto, con lo que aumenta considerablemente la accesibilidad.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

### 5.1.3 JAVASCRIPT

JavaScript es un lenguaje interpretado utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java. Sin embargo, al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia. Es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM (Modelo de Objetos del Documento). JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

### 5.1.4 SQL

El lenguaje de consulta estructurado (SQL Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre la misma.

### 5.1.5 MYSQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado, se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

MySQL es muy utilizado en aplicaciones web como MediaWiki, Amazon, Yahoo, Flickr o Drupal; en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla.

### 5.1.6 NODE.JS

#### **Node.js**

Es un entorno de ejecución para JavaScript que corre del lado del servidor. Node.js se basa en el motor JavaScript de Google Chrome V8 y se utiliza principalmente para crear servidores web, pero no se limita a eso.

Node.js es código abierto, multiplataforma, y desde su aparición en 2009, se ha vuelto sumamente popular y ahora juega un rol relevante en la escena del desarrollo web.

#### **Características de Node.js**

##### **Rápido**

Uno de los puntos fuertes de Node.js es su velocidad. El código JavaScript que corre en Node.js puede ser el doble de rápido que lenguajes compilados como C o Java, y, según la magnitud, puede ser más rápido que lenguajes interpretados como Python o Ruby, por su paradigma no bloqueante.

##### **JavaScript**

Node.js corre código JavaScript. Esto significa que millones de desarrolladores frontend que ya usan JavaScript en el navegador son capaces de correr código del lado del servidor y del lado del cliente usando el mismo lenguaje de programación sin la necesidad de aprender del todo una herramienta nueva.

##### **V8**

Al ejecutarse en el motor JavaScript de Google V8, que es de código abierto, Node.js puede aprovechar el trabajo de miles de ingenieros que hicieron (y seguirán haciendo) el tiempo de ejecución de JavaScript de Chrome increíblemente rápido.

##### **Plataforma asíncrona**

En los lenguajes de programación tradicionales (C, Java, Python, PHP) todas las instrucciones son bloqueantes por default a menos que explícitamente le digas que ejecute operaciones asíncronas. Si realizas una solicitud de red para leer un JSON, la ejecución de ese hilo en particular se bloquea hasta que la respuesta esté lista.

**HTTP** es un elemento destacado en Node.js, diseñado teniendo en cuenta la transmisión de operaciones con streaming y baja latencia. Esto hace que Node.js sea muy adecuado para la base de una librería o un framework web.

Que Node.js esté diseñado para trabajar sin hilos no significa que no pueda aprovechar múltiples núcleos en su entorno. Se pueden generar subprocesos o procesos hijos utilizando nuestra API `child_process.fork()`, la cual está diseñada para que la comunicación entre ellos sea fácil mediante su proceso principal. Desarrollada sobre esa misma interfaz está el módulo `cluster`, que le permite compartir sockets entre procesos para permitir el balanceo de carga entre sus múltiples núcleos.

### 5.1.7 JQUERY

jQuery es una biblioteca o framework de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

### 5.1.8 BOOTSTRAP

Bootstrap es un framework CSS desarrollado por Twitter en 2010, para estandarizar las herramientas de la compañía.

Inicialmente, se llamó Twitter Blueprint y, un poco más tarde, en 2011, se transformó en código abierto y su nombre cambió para Bootstrap. Desde entonces fue actualizado varias veces y ya se encuentra en la versión 4.4.

El framework combina CSS y JavaScript para estilizar los elementos de una página HTML. Permite mucho más que, simplemente, cambiar el color de los botones y los enlaces.

Esta es una herramienta que proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso y más.

Además de todas las características que ofrece el framework, su principal objetivo es permitir la construcción de sitios web responsive para dispositivos móviles.

Esto significa que las páginas están diseñadas para funcionar en desktop, tablets y smartphones, de una manera muy simple y organizada.

### 5.1.9 BCRYPT

Bcrypt es una función de hashing de passwords diseñado por Niels Provos y David Maxieres, basado en el cifrado de Blowfish. Se usa por defecto en sistemas OpenBSD y algunas distribuciones Linux y SUSE. Lleva incorporado un valor llamado salt, que es un fragmento aleatorio que se usará para generar el hash asociado a la password, y se guardará junto con ella en la base de datos. Así se evita que dos passwords iguales generen el mismo hash y los problemas que ello conlleva, por ejemplo, ataque por fuerza bruta a todas las passwords del sistema a la vez. Otro ataque relacionado es el de Rainbow table, que son tablas de asociaciones entre textos y su hash asociado, para evitar su cálculo y acelerar la búsqueda de la password. Con el salt, se añade un grado de complejidad que evita que el hash asociado a una password sea único.

NodeJS, MongoDB y bcrypt.- Finalmente decidimos utilizar la librería de encriptación: bcrypt. Con esta librería podemos generar el hash de cualquier campo. Nos permite elegir el valor de saltRounds, que nos da el control sobre el coste de procesamiento de los datos. Cuanto más alto es este número, más tiempo requiere la máquina para calcular el hash asociado a la password. Es importante a la hora de elegir este valor, seleccionar un número lo suficientemente alto como para que alguien que intente encontrar la password para un usuario por fuerza bruta, requiera tanto tiempo para generar todo el hash de las contraseñas posibles que no le compense. Y, por otra parte, debe ser lo suficientemente pequeño para no acabar con la paciencia del usuario a la hora de registrarse y de acceder (dicha paciencia no suele ser muy alta). Por defecto, el valor saltRounds es 10.

### 5.1.10 HEROKU

Heroku es una plataforma en la nube (plataforma como servicio) que vino a solventar ese problema a través de un conjunto de herramientas para subir, monitorear y escalar nuestra aplicación sin pensar en la infraestructura.

Soporta múltiples lenguajes como Ruby, Java, PHP, Python, Go entre muchos otros. Se indica en qué lenguaje está el proyecto y Heroku se encarga de configurar todo el entorno en un contenedor inteligente manejando la base de datos, seguridad, orquestación, balanceadores de cargas, cache y los registros.

Un concepto que utiliza Heroku es dynos, son unidades de cómputo encargados de procesar y alojar nuestro sitio web. Cada dyno es independiente del otro, permitiendo dividir nuestro proyecto en diferentes contenedores.

### 5.1.11 JSON WEB TOKEN

JWT (JSON Web Token) es un estándar que está dentro del documento RFC 7519.

En el mismo se define un mecanismo para poder propagar entre dos partes, y de forma segura, la identidad de un determinado usuario, además con una serie de claims o privilegios.

Estos privilegios están codificados en objetos de tipo JSON, que se incrustan dentro del payload o cuerpo de un mensaje que va firmado digitalmente.

Los Tokens en tres partes:

- Header: encabezado donde se indica, al menos, el algoritmo y el tipo de token, que en el caso del ejemplo anterior era el algoritmo HS256 y un token JWT.
- Payload: donde aparecen los datos de usuario y privilegios, así como toda la información que queramos añadir, todos los datos que creamos convenientes.
- Signature: una firma que nos permite verificar si el token es válido, y aquí es donde radica el quid de la cuestión, ya que si estamos tratando de hacer una comunicación segura entre partes y hemos visto que podemos coger cualquier token y ver su contenido con una herramienta sencilla.

### 5.1.12 EXPRESS.JS

Express.js es un framework para Node.js que sirve para ayudarnos a crear aplicaciones web en menos tiempo ya que nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, y un largo etc.

Express.js está basado en Connect, que a su vez es un framework basado en http para Node.js. Podemos decir que Connect tiene todas las opciones del módulo http que viene por defecto con Node y le suma funcionalidades. A su vez, Express hace lo mismo con Connect, con lo que tenemos un framework ligero, rápido y muy útil.

### 5.1.13 PHPMYADMIN

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 50 idiomas. Se encuentra disponible bajo la licencia GPL.

## 5.2 Descripción del proyecto

Las páginas Web que componen la aplicación están implementadas siguiendo una estrategia basada en contenidos, es decir, las páginas Web se estructuran en zonas (divs) donde cada una es responsable de proporcionar cierta información sobre un contenido en concreto.

A continuación, va a procederse a hacer una descripción del proyecto siguiendo el esquema por capas que se ha seguido para implementarlo. Para cada capa, se ha elegido un fichero de los que la conforman para explicar su estructura, ya que el número de zonas o divs es siempre el mismo.

### 5.2.3 Capa de presentación

A continuación, se muestra el contenido de uno de los ficheros que conforman la capa de presentación. Para poder explicarlo mejor, he dividido el fichero y he añadido un breve texto explicativo antes de cada una de las partes en las que ha sido dividido.

```
<!DOCTYPE html>
```

```
<html lang="es-MX">
```



```
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
<title>iDoctor</title>
<link rel="stylesheet" href="/assets/bootstrap/css/bootstrap.min.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700">
<link rel="stylesheet" href="/assets/fonts/fontawesome-all.min.css">
<link rel="stylesheet" href="/assets/fonts/font-awesome.min.css">
<link rel="stylesheet" href="/assets/css/Login-Form-Clean.css">
<link rel="stylesheet" href="/assets/css/Navigation-Clean.css">
<link rel="stylesheet" href="/assets/css/styles.css">
</head>
```

La primera zona del cuerpo del documento es el contenedor, dentro de la cual se añadirán todas las zonas que conforman la aplicación.

```
<body>
  <!-- Start: Navigation Clean -->
  <nav class="navbar navbar-light navbar-expand-md navigation-clean d-none"
    style="background-color: #3598DB;">
    <div class="container">
      <a class="navbar-brand" href="#" style="color: #fafafa;">iDoctor</a>
      <button class="navbar-toggler" data-toggle="collapse" data-target="#navcol-1">
        <span class="sr-only">Toggle navigation</span>
        <span class="navbar-toggler-icon text-white-50"></span>
      </button>
      <div class="collapse navbar-collapse" id="navcol-1">
```



```
<ul class="nav navbar-nav ml-auto">
  <li class="dropdown nav-item">
    <a class="dropdown-toggle nav-link" data-toggle="dropdown"
      aria-expanded="false" href="#">
      
    </a>
    <div class="dropdown-menu dropdown-menu-right" role="menu">
      <a class="dropdown-item" href="/mis-datos.html"
        role="presentation">Mis datos</a>
      <a class="dropdown-item" role="presentation" id="btn-cerrar-
        sesion">Cerrar sesión</a>
    </div>
  </li>
</ul>
</div>
</div>
</nav>
```

Representa una sección de una página cuyo propósito es proporcionar enlaces de navegación, ya sea dentro del documento actual o a otros documentos. Ejemplos comunes de secciones de navegación son menús, tablas de contenido e índices.

```
<main class="d-none">
  <div class="container mt-5">
    <div class="row my-0 mx-1 p-2 row-content">
      <div class="col p-2 col-content text-center">
        <h2 class="my-4">iDoctor</h2>
        <span style="float: left; padding-right: 20%; padding-left: 20%; padding-
          bottom: 24px;">
          //aquí se escribe un párrafo, del cual aparecerá en la pantalla.
```

```
</span>
</div>
</div>
</div>
</main>
```

La segunda zona que se añade al contenedor de zonas es la correspondiente al menú, compuesta únicamente por una lista no numerada de enlaces a las distintas páginas que conforman la parte pública de la aplicación.

```
<div id="menu-sidebar" class="sidebar d-none"></div>
<script src="/assets/js/jquery.min.js"></script>
<script src="/assets/bootstrap/js/bootstrap.min.js"></script>
<script src="/assets/js/bootstrap-4-autocomplete.min.js"></script>
<script src="/assets/js/main.js"></script>
<script src="/assets/js/switch.js"></script>
<script src="/assets/js/sidebar.js"></script>
</body>
</html>
```

Un script en el lado del cliente es un programa que puede acompañar a un documento HTML o que puede estar incluido en él. El programa se ejecuta en la máquina del cliente cuando se carga el documento, o en algún otro instante, como por ejemplo cuando se activa un vínculo. El soporte de scripts de HTML es independiente del lenguaje de scripts.

### 5.2.2 Capa de persistencia o de datos

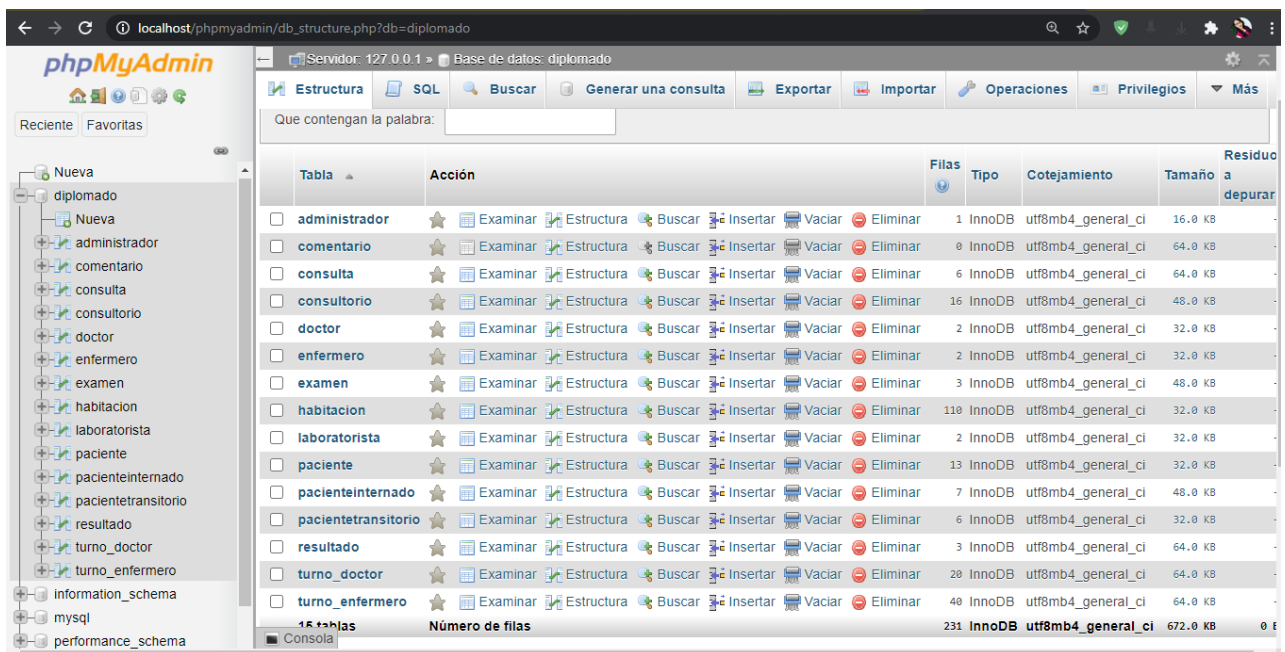


Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> administrador	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_general_ci	16.0 KB	
<input type="checkbox"/> comentario	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	64.0 KB	
<input type="checkbox"/> consulta	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	utf8mb4_general_ci	64.0 KB	
<input type="checkbox"/> consultorio	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	16	InnoDB	utf8mb4_general_ci	48.0 KB	
<input type="checkbox"/> doctor	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	32.0 KB	
<input type="checkbox"/> enfermero	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	32.0 KB	
<input type="checkbox"/> examen	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	48.0 KB	
<input type="checkbox"/> habitacion	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	110	InnoDB	utf8mb4_general_ci	32.0 KB	
<input type="checkbox"/> laboratorista	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_general_ci	32.0 KB	
<input type="checkbox"/> paciente	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	13	InnoDB	utf8mb4_general_ci	32.0 KB	
<input type="checkbox"/> pacienteinternado	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	7	InnoDB	utf8mb4_general_ci	48.0 KB	
<input type="checkbox"/> pacientetransitorio	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	utf8mb4_general_ci	32.0 KB	
<input type="checkbox"/> resultado	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	64.0 KB	
<input type="checkbox"/> turno_doctor	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	20	InnoDB	utf8mb4_general_ci	64.0 KB	
<input type="checkbox"/> turno_enfermero	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	40	InnoDB	utf8mb4_general_ci	64.0 KB	
<b>15 tablas</b>	<b>Número de filas</b>	<b>231</b>	<b>InnoDB</b>	<b>utf8mb4_general_ci</b>	<b>672.0 KB</b>	<b>0 B</b>

Figura 2. Tabla de la base de datos

Como se aprecia en la figura, la base de datos está compuesta por quince tablas de las cuales son: administrador, comentario, consulta, consultorio, doctor, enfermero, examen, habitación, laboratorista, paciente, pacienteinterno, pacientetransitorio, resultado, turno\_doctor y turno\_enfermero.

### 5.3 Reglas de Negocio

- ✚ Los administradores crean pacientes, personal médico (enfermeros, doctores y laboratoristas), consultorios y habitaciones.
- ✚ Los administradores internan pacientes.
- ✚ Los administradores dan de alta pacientes internados.
- ✚ Los administradores asignan personal médico a habitaciones.
- ✚ Los administradores asignan doctores a consultorios.
- ✚ Los pacientes tienen diagnósticos, exámenes médicos e historial clínico.
- ✚ Los PACIENTES INTERNADOS tienen asignados uno o varios enfermeros en TURNO.
- ✚ Los PACIENTES INTERNADOS tienen asignados un doctor en TURNO.
- ✚ Los PACIENTES INTERNADOS tienen asignados una HABITACIÓN.
- ✚ Los PACIENTES TRANSITORIOS tienen asignado un CONSULTORIO MÉDICO.

- ✚ Los doctores tienen asignado un CONSULTORIO MÉDICO.
- ✚ Los doctores solicitan EXÁMENES MÉDICOS.
- ✚ Los doctores tienen asignado un TURNO de 8 horas, ya sea mañana, tarde o noche.
  
- ✚ Los doctores y administradores asignan un diagnóstico.
- ✚ Los doctores y administradores transfieren un paciente.
  
- ✚ Los enfermeros y doctores tienen asignados varios PACIENTES INTERNADOS.
- ✚ Los enfermeros y doctores tienen asignados varias HABITACIONES
- ✚ Los enfermeros y doctores ven el HISTORIAL del PACIENTE ASIGNADO.
- ✚ Los enfermeros y doctores ven los EXÁMENES MÉDICOS del PACIENTE ASIGNADO.
- ✚ Los enfermeros y doctores asignan CITA a un PACIENTE TRANSITORIO.
- ✚ Los enfermeros, doctores y laboratoristas consultan los RESULTADOS médicos del PACIENTE ASIGNADO.
  
- ✚ Los laboratoristas generan los RESULTADOS de los EXÁMENES MÉDICOS.

#### # Comentarios

- ✚ Los doctores pueden ser asignados a una o muchas habitaciones durante su rotación, así como cumplir funciones de consulta en el área de pacientes transitorios.
- ✚ Los enfermeros solamente tienen funciones dentro del área de pacientes internados, y un enfermero podrá cubrir más de una HABITACIÓN durante su TURNO.
- ✚ Los EXÁMENES MÉDICOS se solicitan desde la plataforma y los enfermeros podrán visualizarlos en caso de que ellos tengan que cubrir alguna función en el examen médico (como la toma de muestras).
- ✚ Los laboratoristas reciben la solicitud y el material para trabajar y generan un EXAMEN MÉDICO de resultado (en PDF), así como un breve DIAGNÓSTICO (si es posible) como detalle extra en la solicitud.
- ✚ Todo el personal médico, así como los pacientes, son dados de alta directamente por el perfil de administrador, y solo pueden ser modificados por él.

- ✚ En caso de que un PACIENTE sea transferido a otra unidad (que puede ser cuidados intensivos, quirófano o piso), se mantendrá su HISTORIAL MÉDICO.

## 6 CONCLUSIÓN

### 6.1 Valoración personal del trabajo realizado

Este proyecto ha sido para mí un gran reto, ya que no estaba familiarizada con el desarrollo web, con el paso del tiempo en este Diplomado, me ha enseñado desde ceros a realizar uno, sin embargo, como soy primeriza, si me costó trabajo entender el mecanismo y sobre todo implementar ese conocimiento al Proyecto de Novamedik, ya que no tenía idea por dónde empezar, con el paso del tiempo estuve realizando muchas investigaciones acerca de lo que se requeriría para llevarlo a cabo, hasta que encontré plataformas de servidor, frameworks, entornos para utilizar javascript, que desde luego fue una parte que aún se me dificulta realizar y todo lo que tiene que ver con el front-end.

Por otra parte para solucionar la parte del back-end, utilice una API ( Application Programming Interface) REST (Representational State Transfer), que es un conjunto de restricciones, del cual se utiliza para crear aplicaciones web y respetando la parte de HTTP.

## 7 BIBLIOGRAFÍA

<https://nodejs.org/es/>

<https://flaviocopes.com/>

<https://www.heroku.com/nodejs>

<https://git-scm.com/book/es/v2/Git-en-el-Servidor-GitLab>

<https://expressjs.com/es/>

[https://www.w3schools.com/nodejs/nodejs\\_mysql.asp](https://www.w3schools.com/nodejs/nodejs_mysql.asp)

<https://www.npmjs.com/package/mysql#pooling-connections>



<https://www.npmjs.com/package/mysql#custom-format>

<https://stackoverflow.com/questions/54730641/node-js-how-to-apply-util-promiseify-to-mysql-pool-in-its-simplest-way>

<https://www.npmjs.com/package/bcrypt#usage>

[https://www.w3schools.com/nodejs/nodejs\\_mysql\\_insert.asp](https://www.w3schools.com/nodejs/nodejs_mysql_insert.asp)

<https://getbootstrap.com>

<https://fullcalendar.io/docs/bootstrap-theme-demo>

<https://www.yukei.net/2016/01/getting-response-headers-data-from-an-ajax-request-with-jquery>

<https://stackoverflow.com/questions/4811807/hidden-sidebar-that-shows-up-on-hover>

[https://www.w3schools.com/howto/howto\\_js\\_sidenav.asp](https://www.w3schools.com/howto/howto_js_sidenav.asp)

<https://html-online.com/articles/get-url-parameters-javascript/>

<https://stackoverflow.com/questions/8668174/indexof-method-in-an-object-array>

<https://www.npmjs.com/package/swagger-ui-express#usage>

<https://stackoverflow.com/questions/47089230/how-to-convert-buffer-to-stream-in-nodejs/54136803>

<https://www.npmjs.com/package/morgan#skip>

<https://www.geekytheory.com/que-es-una-api-rest-y-para-que-se-utiliza>

Libro:

HTML & CSS, Design and build websites; Jon Duckett; Wiley; 2014; Indianapolis, Indiana, EUA

Javascript & JQuery, Interactive front-end web development; Jon Duckett; Wiley; 2014; Indianapolis, Indiana, EUA.

- Título: The Node.js Handbook  
Autor: Flavio Copes  
Fecha de creación: 24 de septiembre de 2018

RESTful Web API Design with Node.js; Valentin Bojinov; marzo 2015; Packt Publishing Ltd.; Birmingham, Reino Unido