Manual De Configuración Base de Datos

<Solution IPS>

Versión: <1.1>

**HISTORIAL DE REVISIÓN**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **VERSIÓN** | **ELABORACIÓN** | | **REVISIÓN** | | **APROBACIÓN** | |
| **Fecha** | **Responsable** | **Fecha** | **Responsable** | **Fecha** | **Responsable** |
| 1.0.0 | 15/09/21 | Julian Franco  Alejo Giraldo  Cris Solano  Guillermo Domínguez  Juan Hernández  Juan Agudelo  Johnier Rodríguez  Jorge Camelo | 17/09/21 | Alejo Giraldo  Cris Solano  Guillermo Domínguez  Juan Hernández  Juan Agudelo  Johnier Rodríguez  Jorge Camelo | 21/09/2021 | Willington Rincón |
| 1.1 | 05/12/2021 | Julian Franco  Alejandro Giraldo  Juan Agudelo  Juan Hernández  Jorge Camelo | 08/12/2021 | Julian Franco  Alejandro Giraldo  Juan Agudelo  Juan Hernández  Jorge Camelo | 10/12/2021 | Willington Rincón |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**CAMBIOS RESPECTO A LA VERSIÓN ANTERIOR**

|  |  |
| --- | --- |
| **VERSIÓN** | **MODIFICACIÓN RESPECTO VERSIÓN ANTERIOR** |
| 1.0 | Documento inicial para la configuración de BD |
| 1.1 | Complementación de la configuración de BD, se anexan los sub dominio de valores de ciertas tablas |
| 1.2 | Se define el diccionario de datos |
|  |  |
|  |  |
|  |  |

**Tabla de contenido**

[1. Introducción 4](#_30j0zll)

[2. Alcance 4](#_1fob9te)

[4. Responsables e involucrados 4](#_3znysh7)

[5. Modelo Entidad Relación (MER) 4](#_2et92p0)

[6. Diccionario de Datos. 4](#_tyjcwt)

[7. Modelo Relacional. 4](#_3dy6vkm)

[8. Justificación Motor Seleccionado 4](#_1t3h5sf)

[9. Requisitos de Configuración 4](#_4d34og8)

[10. Scripts 4](#_2s8eyo1)

[11. Configuración y Ejecución de la Base de Datos 4](#_17dp8vu)

[12. Otras Consideraciones 5](#_3rdcrjn)

# 1. Introducción

Las entidades de salud necesitan herramientas eficientes que permitan una administración inteligente de los diferentes recursos con los que cuentan para hacer frente a los retos diarios de su misión.

El rápido crecimiento del mercado de empresas prestadoras de salud en Colombia brinda la oportunidad de aportar a la solución de los retos que enfrentan estas organizaciones, mediante el desarrollo de aplicaciones informáticas y otras tecnologías que faciliten a los administradores el cumplimiento de las metas planteadas mediante la adecuada gestión de las diferentes actividades.

# 2. Alcance

## Con este documento se pretende identificar y plasmar los diferentes escenarios en cuanto a BD se refiere, tablas, atributos, conexiones de un sistema modular informático que permita la administración eficiente de una IPS enfocada en hacer exámenes ocupacionales, haciendo una revisión del alcance y justificación de la misma.

# 4. Responsables e involucrados

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo(Responsable / Involucrado)** | **Rol** |
| Juan José Agudelo | Responsable | Documentación |
| Juan Camilo Hernández | Responsable | Desarrollo Back end |
| Julian Franco | Responsable | Documentación |
| Alejo Giraldo | Responsable | Desarrollo Back end |
| Jorge Camelo | Responsable | Desarrollador BD |
|  |  |  |

# 

# 5. Modelo Entidad Relación (MER)

Aunque se utiliza un motor de BD relacional, no existen relaciones entre las tablas ya que se diseña en un modelo por microservicios el cual contiene una BD individual para cada uno.

Además, se opta por el uso de ORM un ORM (de sus siglas en inglés, Object Relational Mapper), no es más que una utilidad o librería que permite manipular las tablas de una base de datos como si fueran objetos de nuestro programa.

Lo más habitual es que una tabla se corresponda con una clase, cada fila de una tabla con un objeto (o instancia de una clase), las columnas de una tabla con los atributos de una clase y las claves ajenas (o Foreign Keys) con relaciones entre clases (definidas también a partir de atributos).

Ventajas de usar un ORM

* Acceder a las tablas y filas de una base de datos como clases y objetos.
* En la mayoría de ocasiones no es necesario usar el lenguaje SQL.
* El ORM se encarga de hacer las traducciones oportunas.
* Independencia de la base de datos.
* Es posible cambiar de motor de base de datos modificando muy poco código en la aplicación.
* Incrementa la productividad del desarrollador.

SQLAlchemy es una librería para Python que facilita el acceso a una base de datos relacional, así como las operaciones a realizar sobre la misma.

Es independiente del motor de base de datos a utilizar, es decir, en principio, es compatible con la mayoría de bases de datos relacionales conocidas: PostgreSQL, MySQL, Oracle, Microsoft SQL Server, Sqlite, …

Aunque se puede usar SQLAlchemy utilizando consultas en lenguaje SQL nativo, la principal ventaja de trabajar con esta librería se consigue haciendo uso de su ORM. El ORM de SQLAlchemy mapea tablas a clases Python y convierte automáticamente llamadas a funciones dentro de estas clases a sentencias SQL.

Además, SQLAlchemy implementa múltiples patrones de diseño que te permiten desarrollar aplicaciones rápidamente y te abstrae de ciertas tareas, como manejar el pool de conexiones a la base de datos.

SQLAlchemy proporciona una interfaz única para comunicarte con los diferentes drivers de bases de datos Python que implementan el estándar Python DBAPI.

Este estándar, especifica cómo las librerías Python que se integran con las bases de datos deben exponer sus interfaces. Por tanto, al usar SQLAlchemy no interactuarás directamente con dicho API, sino con la interfaz que precisamente proporciona SQLAlchemy. Esto es lo que permite cambiar el motor de base de datos de una aplicación sin modificar apenas el código que interactúa con los datos.

# 6. Diccionario de Datos.

# Se clasifican por microservicios. A continuación las Tablas y campos en MySQL

Microservicio de autenticación







Microservicio Medicina







Microservicio de Fonoaudiología



Microservicio de Optometría



Microservicio de Psicología



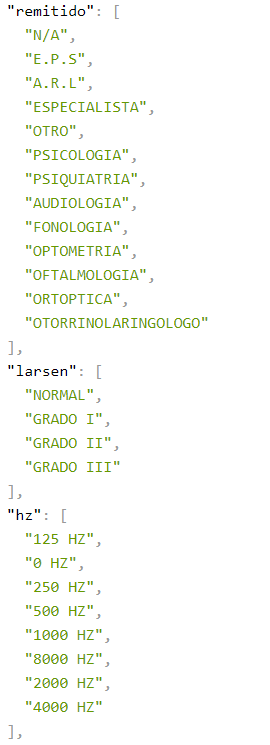
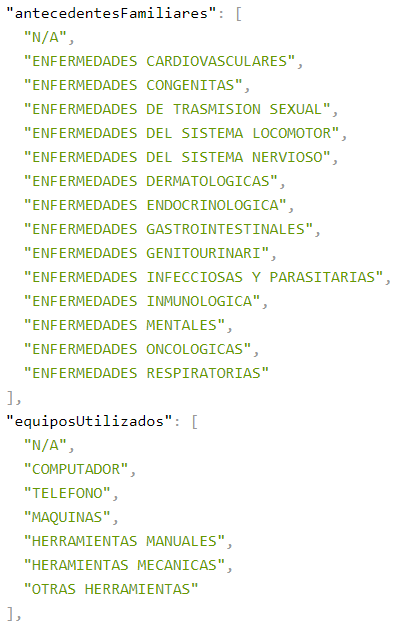
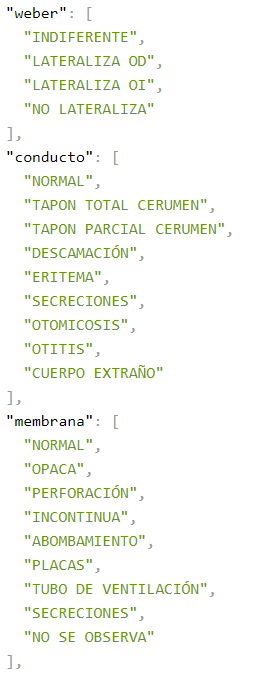
Microservicio de Certificación

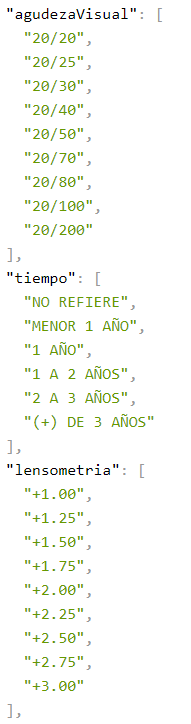
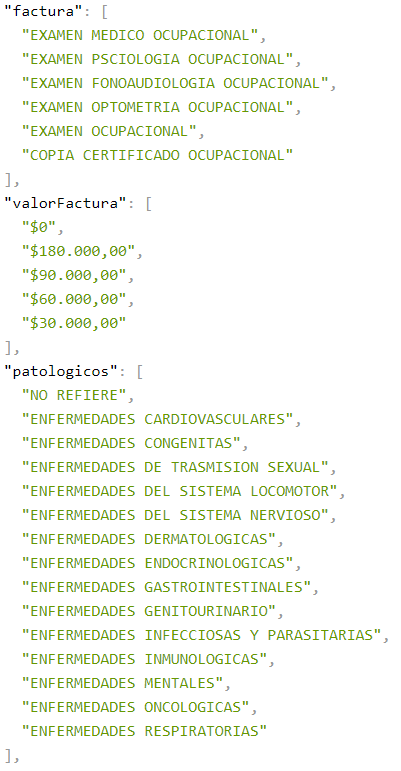
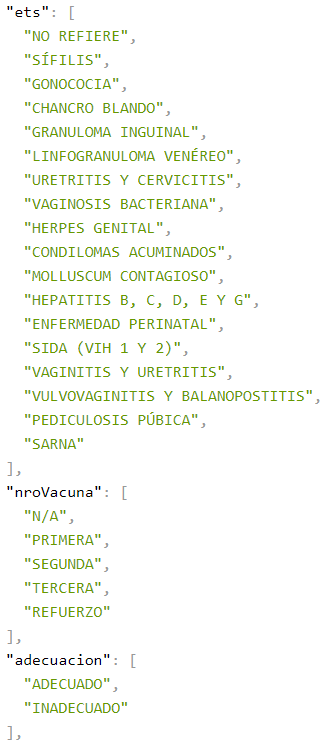
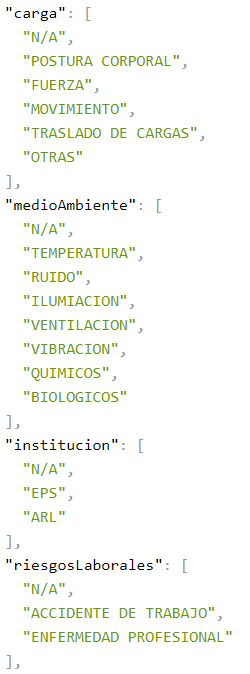


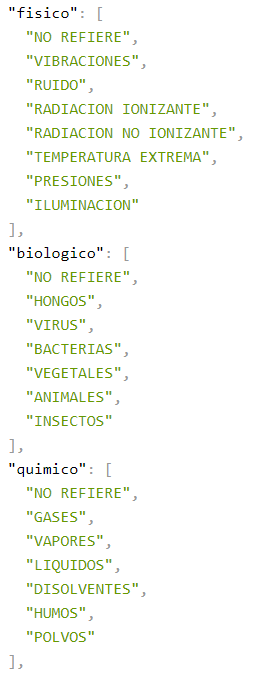
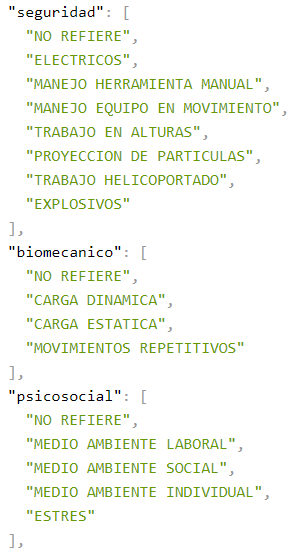
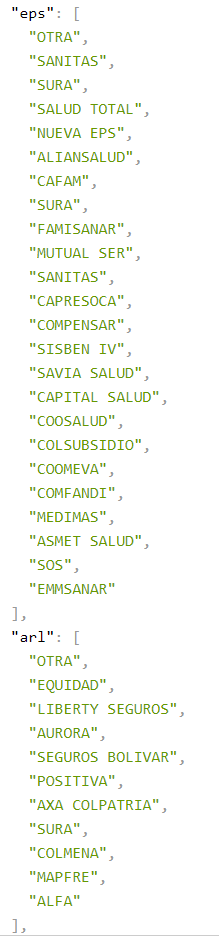
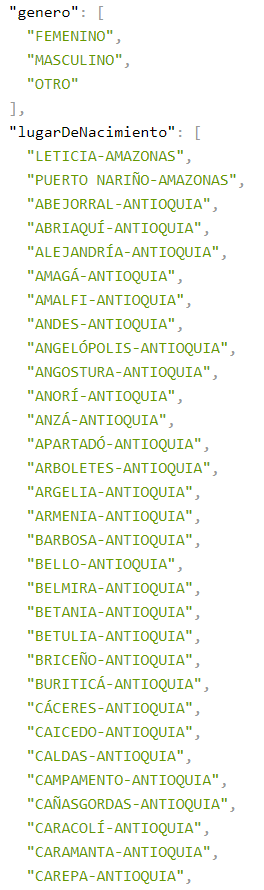
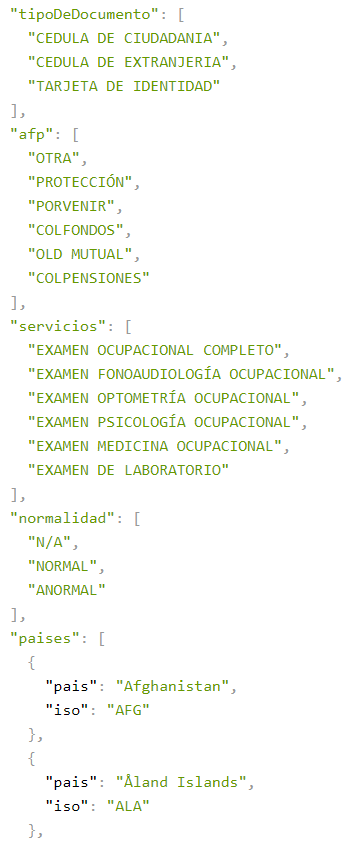
Microservicio de Facturación

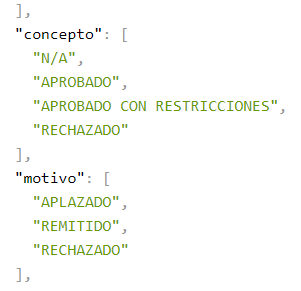
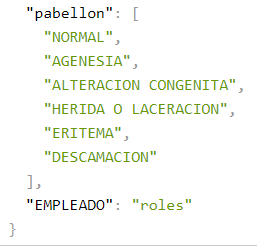


A continuación, las colecciones y documentos en MongoDB para el uso de las API´s

Api con valores



# 7. Modelo Relacional

N/A

*.*

# 8. Justificación Motor Seleccionado

1. Para la administración de las historias del área clínica se opta por utilizar MySQL.

MySQL es un sistema de base de datos relacionales muy popular, es de los más utilizados para los sistemas de gestión de contenidos WordPress Además, tiene muchas ventajas que lo hacen una excelente elección para trabajar como:

Es una BD gratis y al ser de código abierto no tiene costo

Es muy fácil de usar, y muy prácticos para personas que no tienen mucho conocimiento en manejo de bases de datos, es ideal para empezar a trabajar en este motor porque se trabaja con comandos muy sencillos.

Es una base de datos muy rápido, por lo que da un buen rendimiento, necesita pocos recursos de RAM y no maneja muchos requerimientos también es muy importante decir que es compatible con Linux y Windows. Basado en <https://fp.uoc.fje.edu/blog/por-que-elegir-el-gestor-de-base-de-datos-mysql/>

1. Para la administración de las Api se utiliza MongoDB

MongoDB es un sistema para la gestión de datos NoSQL o no relacional. Se trata de un modelo orientado a documentos que se almacenan en BSON, una representación binaria de JSON, y que no usa tablas como los sistemas SQL ni necesita seguir un esquema.

MongoDB se centra en el uso de colecciones, las cuales son un conjunto de documentos que, a su vez, contienen una estructura JSON y, dentro, unas claves que funcionan como campos.

MongoDB puede emplearse en la mayoría de los proyectos, pero no soporta las transacciones, solo garantiza operaciones atómicas en el ámbito de documentos, por lo cual solo nos sirve para usarlo como conjunto de opciones a elegir.

1. Mongo Atlas para el despliegue

Mediante su servicio en la nube de Mongo, se utiliza para el despliegue en su capa gratuita, por medio de una llave de conexión define los controladores para sus métodos, posteriormente la api se conecta a Heroku para hacerla una url publica para el acceso a ella por parte del equipo de desarrollo.

MongoDB Atlas es un servicio de Cloud Database (o Base de Datos en la Nube), que te permite crear y administrar tu BBDD Mongo desde cualquier lugar del mundo, a través de su plataforma.

Además, MongoDB Atlas no solo está orientado a ser accesible desde el navegador, sino que, fue desarrollado con el objetivo de aliviar el trabajo de los desarrolladores, al quitarles la necesidad de instalar y administrar entornos de BBDD, los que a veces pueden ser lentos y engorrosos.

# 9. Requisitos de Configuración

* Requerimientos básicos para **MySQL:**
  + 512 Mb de memoria Ram
  + 1024 Mb máquina virtual
  + 1 GB de espacio de disco duro
  + Sistema operativo: Windows, Linux y Unix
  + Arquitectura del sistema 32/64 bit
  + Protocolo de red TCP/IP Tomado de: http://lopez-garcia-victor.blogspot.com/2012/08/11-requerimientos-e-instalacion-del-sgbd.html
* Requerimientos básicos para MongoDB
  + Funciona con cualquier sistema operativo de Windows
  + No se debe usar un sistema operativo de 32 bits tiene el problema de estar limitada a 2Gb de datos
  + Mínimo de 512 de memoria RAM

# 10. Scripts

Se crea desde el back-end por medio de un ORM con SQLAlchemy

*class HistoriaMedicaService():*

*def buscar\_num\_historia(self, doc):*

*stmt = select(*

*HistoriaMedica.numero\_historia*

*).where(*

*HistoriaMedica.documento\_paciente == doc*

*).order\_by(*

*desc(HistoriaMedica.numero\_historia)*

*)*

*historias = db.session.execute(stmt).fetchall()*

*if historias:*

*return historias[0][0]*

*return 0*

*def create(self, content):*

*doc = content.get("documento\_paciente")*

*data = dict(*

*documento\_paciente=doc,*

*estado = content.get("estado"),*

*ant\_padre\_card = content.get("ant\_padre\_card"),*

*ant\_madre\_card = content.get("ant\_madre\_card"),*

*ant\_padre\_cong = content.get("ant\_padre\_cong"),*

*ant\_madre\_cong = content.get("ant\_madre\_cong"),*

*ant\_per\_pato = content.get("ant\_per\_pato"),*

*ant\_per\_qui = content.get("ant\_per\_qui"),*

*ant\_per\_trau = content.get("ant\_per\_trau"),*

*ant\_per\_toxi = content.get("ant\_per\_toxi"),*

*ant\_per\_alergi = content.get("ant\_per\_alergi"),*

*ant\_per\_ets = content.get("ant\_per\_ets"),*

*ant\_per\_obs1 = content.get("ant\_per\_obs1"),*

*ant\_per\_gin\_fup = content.get("ant\_per\_gin\_fup"),*

*ant\_per\_gin\_fum = content.get("ant\_per\_gin\_fum"),*

*ant\_per\_gin\_plan = content.get("ant\_per\_gin\_plan"),*

*ant\_per\_gin\_dism = content.get("ant\_per\_gin\_dism"),*

*ant\_per\_gin\_disp = content.get("ant\_per\_gin\_disp"),*

*ant\_per\_gin\_mam = content.get("ant\_per\_gin\_mam"),*

*ant\_per\_gin\_mens = content.get("ant\_per\_gin\_mens"),*

*ant\_per\_obs2 = content.get("ant\_per\_obs2"),*

*inm\_hep\_a = content.get("inm\_hep\_a"),*

*inm\_hep\_b = content.get("inm\_hep\_b"),*

*inm\_trip = content.get("inm\_trip"),*

*inm\_teta = content.get("inm\_teta"),*

*inm\_mala = content.get("inm\_mala"),*

*inm\_amar = content.get("inm\_amar"),*

*inm\_tifo = content.get("inm\_tifo"),*

*inm\_cov = content.get("inm\_cov"),*

*mani\_ali = content.get("mani\_ali"),*

*inm\_obs = content.get("inm\_obs"),*

*hab\_ciga = content.get("hab\_ciga"),*

*hab\_alco = content.get("hab\_alco"),*

*hab\_drog = content.get("hab\_drog"),*

*hab\_dep = content.get("hab\_dep"),*

*hab\_les = content.get("hab\_les"),*

*hab\_obs = content.get("hab\_obs"),*

*sis\_derma = content.get("sis\_derma"),*

*sis\_ost\_musc = content.get("sis\_ost\_musc"),*

*sis\_ost\_arti = content.get("sis\_ost\_arti"),*

*sis\_geni = content.get("sis\_geni"),*

*sis\_meta = content.get("sis\_meta"),*

*sis\_neur = content.get("sis\_neur"),*

*sis\_carf = content.get("sis\_carf"),*

*sis\_endo = content.get("sis\_endo"),*

*sis\_uro = content.get("sis\_uro"),*

*sis\_gatro = content.get("sis\_gatro"),*

*sis\_orl = content.get("sis\_orl"),*

*rie\_exp\_fis\_1 = content.get("rie\_exp\_fis\_1"),*

*rie\_exp\_fis\_2 = content.get("rie\_exp\_fis\_2"),*

*rie\_exp\_fis\_3 = content.get("rie\_exp\_fis\_3"),*

*rie\_exp\_fis\_4 = content.get("rie\_exp\_fis\_4"),*

*rie\_exp\_bio\_1 = content.get("rie\_exp\_bio\_1"),*

*rie\_exp\_bio\_2 = content.get("rie\_exp\_bio\_2"),*

*rie\_exp\_bio\_3 = content.get("rie\_exp\_bio\_3"),*

*rie\_exp\_bio\_4 = content.get("rie\_exp\_bio\_4"),*

*rie\_exp\_quim\_1 = content.get("rie\_exp\_quim\_1"),*

*rie\_exp\_quim\_2 = content.get("rie\_exp\_quim\_2"),*

*rie\_exp\_quim\_3 = content.get("rie\_exp\_quim\_3"),*

*rie\_exp\_quim\_4 = content.get("rie\_exp\_quim\_4"),*

*rie\_exp\_seg\_1 = content.get("rie\_exp\_seg\_1"),*

*rie\_exp\_seg\_2 = content.get("rie\_exp\_seg\_2"),*

*rie\_exp\_seg\_3 = content.get("rie\_exp\_seg\_3"),*

*rie\_exp\_seg\_4 = content.get("rie\_exp\_seg\_4"),*

*rie\_exp\_biom\_1 = content.get("rie\_exp\_biom\_1"),*

*rie\_exp\_biom\_2 = content.get("rie\_exp\_biom\_2"),*

*rie\_exp\_biom\_3 = content.get("rie\_exp\_biom\_3"),*

*rie\_exp\_biom\_4 = content.get("rie\_exp\_biom\_4"),*

*rie\_exp\_psico\_1 = content.get("rie\_exp\_psico\_1"),*

*rie\_exp\_psico\_2 = content.get("rie\_exp\_psico\_2"),*

*rie\_exp\_psico\_3 = content.get("rie\_exp\_psico\_3"),*

*rie\_exp\_psico\_4 = content.get("rie\_exp\_psico\_4"),*

*rie\_exp\_obs = content.get("rie\_exp\_obs"),*

*rie\_ant\_fis\_1 = content.get("rie\_ant\_fis\_1"),*

*rie\_ant\_fis\_2 = content.get("rie\_ant\_fis\_2"),*

*rie\_ant\_fis\_3 = content.get("rie\_ant\_fis\_3"),*

*rie\_ant\_fis\_4 = content.get("rie\_ant\_fis\_4"),*

*rie\_ant\_bio\_1 = content.get("rie\_ant\_bio\_1"),*

*rie\_ant\_bio\_2 = content.get("rie\_ant\_bio\_2"),*

*rie\_ant\_bio\_3 = content.get("rie\_ant\_bio\_3"),*

*rie\_ant\_bio\_4 = content.get("rie\_ant\_bio\_4"),*

*rie\_ant\_quim\_1 = content.get("rie\_ant\_quim\_1"),*

*rie\_ant\_quim\_2 = content.get("rie\_ant\_quim\_2"),*

*rie\_ant\_quim\_3 = content.get("rie\_ant\_quim\_3"),*

*rie\_ant\_quim\_4 = content.get("rie\_ant\_quim\_4"),*

*rie\_ant\_seg\_1 = content.get("rie\_ant\_seg\_1"),*

*rie\_ant\_seg\_2 = content.get("rie\_ant\_seg\_2"),*

*rie\_ant\_seg\_3 = content.get("rie\_ant\_seg\_3"),*

*rie\_ant\_seg\_4 = content.get("rie\_ant\_seg\_4"),*

*rie\_ant\_biom\_1 = content.get("rie\_ant\_biom\_1"),*

*rie\_ant\_biom\_2 = content.get("rie\_ant\_biom\_2"),*

*rie\_ant\_biom\_3 = content.get("rie\_ant\_biom\_3"),*

*rie\_ant\_biom\_4 = content.get("rie\_ant\_biom\_4"),*

*rie\_ant\_psico\_1 = content.get("rie\_ant\_psico\_1"),*

*rie\_ant\_psico\_2 = content.get("rie\_ant\_psico\_2"),*

*rie\_ant\_psico\_3 = content.get("rie\_ant\_psico\_3"),*

*rie\_ant\_psico\_4 = content.get("rie\_ant\_psico\_4"),*

*rie\_ant\_obs = content.get("rie\_ant\_obs"),*

*ocu\_equi = content.get("ocu\_equi"),*

*ocu\_acti = content.get("ocu\_acti"),*

*ocu\_acc\_emp1 = content.get("ocu\_acc\_emp1"),*

*ocu\_acc\_diag1 = content.get("ocu\_acc\_diag1"),*

*ocu\_acc\_emp2 = content.get("ocu\_acc\_emp2"),*

*ocu\_acc\_diag2 = content.get("ocu\_acc\_diag2"),*

*ocu\_obs = content.get("ocu\_obs"),*

*cie\_concep\_desc = content.get("cie\_concep\_desc"),*

*cie\_concep\_reco = content.get("cie\_concep\_reco"),*

*cie\_concep\_aplaz = content.get("cie\_concep\_aplaz"),*

*cie\_concep\_aplaza = content.get("cie\_concep\_aplaza"),*

*cie\_concep\_reco\_mot = content.get("cie\_concep\_reco\_mot"),*

*cie\_obs = content.get("cie\_obs"),*

*cie\_concep\_fin = content.get("cie\_concep\_fin")*

*)*

*num\_historia = self.buscar\_num\_historia(doc)*

*abierta = HistoriaMedica.query.filter\_by(numero\_historia=num\_historia).first()*

*if abierta:*

*if abierta.estado:*

*return self.update(content)*

*try:*

*data["numero\_historia"] = num\_historia + 1*

*historia = HistoriaMedica(\*\*data)*

*db.session.add(historia)*

*db.session.commit()*

*return ({*

*"response": "Historia médica creada correctamente."*

*}, 201)*

*except Exception as e:*

*db.session.rollback()*

*return ({*

*"response": str(e),*

*}, 400)*

*def get(self, doc):*

*historias = HistoriaMedica.query.filter\_by(documento\_paciente=doc).all()*

*data = [{x.to\_dict().get("documento\_paciente") + "-" + str(x.to\_dict().get("numero\_historia")):x.to\_dict()} for x in historias]*

*return ({*

*"response": {*

*"historias\_clinicas": data*

*}*

*}, 200)*

*def update(self, content):*

*doc = content.get("documento\_paciente")*

*num\_historia = self.buscar\_num\_historia(doc)*

*historia = HistoriaMedica.query.filter\_by(documento\_paciente=doc, numero\_historia=num\_historia).first()*

*if not historia:*

*return ({*

*"response": "No existe una historia médica con ese documento."*

*}, 406)*

*if not historia.estado:*

*return ({*

*"response": "La historia médica ha sido cerrada y no se puede modificar"*

*}, 401)*

*historia.estado = content.get("estado")*

*historia.ant\_padre\_card = content.get("ant\_padre\_card")*

*historia.ant\_madre\_card = content.get("ant\_madre\_card")*

*historia.ant\_padre\_cong = content.get("ant\_padre\_cong")*

*historia.ant\_madre\_cong = content.get("ant\_madre\_cong")*

*historia.ant\_per\_pato = content.get("ant\_per\_pato")*

*historia.ant\_per\_qui = content.get("ant\_per\_qui")*

*historia.ant\_per\_trau = content.get("ant\_per\_trau")*

*historia.ant\_per\_toxi = content.get("ant\_per\_toxi")*

*historia.ant\_per\_alergi = content.get("ant\_per\_alergi")*

*historia.ant\_per\_ets = content.get("ant\_per\_ets")*

*historia.ant\_per\_obs1 = content.get("ant\_per\_obs1")*

*historia.ant\_per\_gin\_fup = content.get("ant\_per\_gin\_fup")*

*historia.ant\_per\_gin\_fum = content.get("ant\_per\_gin\_fum")*

*historia.ant\_per\_gin\_plan = content.get("ant\_per\_gin\_plan")*

*historia.ant\_per\_gin\_dism = content.get("ant\_per\_gin\_dism")*

*historia.ant\_per\_gin\_disp = content.get("ant\_per\_gin\_disp")*

*historia.ant\_per\_gin\_mam = content.get("ant\_per\_gin\_mam")*

*historia.ant\_per\_gin\_mens = content.get("ant\_per\_gin\_mens")*

*historia.ant\_per\_obs2 = content.get("ant\_per\_obs2")*

*historia.inm\_hep\_a = content.get("inm\_hep\_a")*

*historia.inm\_hep\_b = content.get("inm\_hep\_b")*

*historia.inm\_trip = content.get("inm\_trip")*

*historia.inm\_teta = content.get("inm\_teta")*

*historia.inm\_mala = content.get("inm\_mala")*

*historia.inm\_amar = content.get("inm\_amar")*

*historia.inm\_tifo = content.get("inm\_tifo")*

*historia.inm\_cov = content.get("inm\_cov")*

*historia.mani\_ali = content.get("mani\_ali")*

*historia.inm\_obs = content.get("inm\_obs")*

*historia.hab\_ciga = content.get("hab\_ciga")*

*historia.hab\_alco = content.get("hab\_alco")*

*historia.hab\_drog = content.get("hab\_drog")*

*historia.hab\_dep = content.get("hab\_dep")*

*historia.hab\_les = content.get("hab\_les")*

*historia.hab\_obs = content.get("hab\_obs")*

*historia.sis\_derma = content.get("sis\_derma")*

*historia.sis\_ost\_musc = content.get("sis\_ost\_musc")*

*historia.sis\_ost\_arti = content.get("sis\_ost\_arti")*

*historia.sis\_geni = content.get("sis\_geni")*

*historia.sis\_meta = content.get("sis\_meta")*

*historia.sis\_neur = content.get("sis\_neur")*

*historia.sis\_carf = content.get("sis\_carf")*

*historia.sis\_endo = content.get("sis\_endo")*

*historia.sis\_uro = content.get("sis\_uro")*

*historia.sis\_gatro = content.get("sis\_gatro")*

*historia.sis\_orl = content.get("sis\_orl")*

*historia.rie\_exp\_fis\_1 = content.get("rie\_exp\_fis\_1")*

*historia.rie\_exp\_fis\_2 = content.get("rie\_exp\_fis\_2")*

*historia.rie\_exp\_fis\_3 = content.get("rie\_exp\_fis\_3")*

*historia.rie\_exp\_fis\_4 = content.get("rie\_exp\_fis\_4")*

*historia.rie\_exp\_bio\_1 = content.get("rie\_exp\_bio\_1")*

*historia.rie\_exp\_bio\_2 = content.get("rie\_exp\_bio\_2")*

*historia.rie\_exp\_bio\_3 = content.get("rie\_exp\_bio\_3")*

*historia.rie\_exp\_bio\_4 = content.get("rie\_exp\_bio\_4")*

*historia.rie\_exp\_quim\_1 = content.get("rie\_exp\_quim\_1")*

*historia.rie\_exp\_quim\_2 = content.get("rie\_exp\_quim\_2")*

*historia.rie\_exp\_quim\_3 = content.get("rie\_exp\_quim\_3")*

*historia.rie\_exp\_quim\_4 = content.get("rie\_exp\_quim\_4")*

*historia.rie\_exp\_seg\_1 = content.get("rie\_exp\_seg\_1")*

*historia.rie\_exp\_seg\_2 = content.get("rie\_exp\_seg\_2")*

*historia.rie\_exp\_seg\_3 = content.get("rie\_exp\_seg\_3")*

*historia.rie\_exp\_seg\_4 = content.get("rie\_exp\_seg\_4")*

*historia.rie\_exp\_biom\_1 = content.get("rie\_exp\_biom\_1")*

*historia.rie\_exp\_biom\_2 = content.get("rie\_exp\_biom\_2")*

*historia.rie\_exp\_biom\_3 = content.get("rie\_exp\_biom\_3")*

*historia.rie\_exp\_biom\_4 = content.get("rie\_exp\_biom\_4")*

*historia.rie\_exp\_psico\_1 = content.get("rie\_exp\_psico\_1")*

*historia.rie\_exp\_psico\_2 = content.get("rie\_exp\_psico\_2")*

*historia.rie\_exp\_psico\_3 = content.get("rie\_exp\_psico\_3")*

*historia.rie\_exp\_psico\_4 = content.get("rie\_exp\_psico\_4")*

*historia.rie\_exp\_obs = content.get("rie\_exp\_obs")*

*historia.rie\_ant\_fis\_1 = content.get("rie\_ant\_fis\_1")*

*historia.rie\_ant\_fis\_2 = content.get("rie\_ant\_fis\_2")*

*historia.rie\_ant\_fis\_3 = content.get("rie\_ant\_fis\_3")*

*historia.rie\_ant\_fis\_4 = content.get("rie\_ant\_fis\_4")*

*historia.rie\_ant\_bio\_1 = content.get("rie\_ant\_bio\_1")*

*historia.rie\_ant\_bio\_2 = content.get("rie\_ant\_bio\_2")*

*historia.rie\_ant\_bio\_3 = content.get("rie\_ant\_bio\_3")*

*historia.rie\_ant\_bio\_4 = content.get("rie\_ant\_bio\_4")*

*historia.rie\_ant\_quim\_1 = content.get("rie\_ant\_quim\_1")*

*historia.rie\_ant\_quim\_2 = content.get("rie\_ant\_quim\_2")*

*historia.rie\_ant\_quim\_3 = content.get("rie\_ant\_quim\_3")*

*historia.rie\_ant\_quim\_4 = content.get("rie\_ant\_quim\_4")*

*historia.rie\_ant\_seg\_1 = content.get("rie\_ant\_seg\_1")*

*historia.rie\_ant\_seg\_2 = content.get("rie\_ant\_seg\_2")*

*historia.rie\_ant\_seg\_3 = content.get("rie\_ant\_seg\_3")*

*historia.rie\_ant\_seg\_4 = content.get("rie\_ant\_seg\_4")*

*historia.rie\_ant\_biom\_1 = content.get("rie\_ant\_biom\_1")*

*historia.rie\_ant\_biom\_2 = content.get("rie\_ant\_biom\_2")*

*historia.rie\_ant\_biom\_3 = content.get("rie\_ant\_biom\_3")*

*historia.rie\_ant\_biom\_4 = content.get("rie\_ant\_biom\_4")*

*historia.rie\_ant\_psico\_1 = content.get("rie\_ant\_psico\_1")*

*historia.rie\_ant\_psico\_2 = content.get("rie\_ant\_psico\_2")*

*historia.rie\_ant\_psico\_3 = content.get("rie\_ant\_psico\_3")*

*historia.rie\_ant\_psico\_4 = content.get("rie\_ant\_psico\_4")*

*historia.rie\_ant\_obs = content.get("rie\_ant\_obs")*

*historia.ocu\_equi = content.get("ocu\_equi")*

*historia.ocu\_acti = content.get("ocu\_acti")*

*historia.ocu\_acc\_emp1 = content.get("ocu\_acc\_emp1")*

*historia.ocu\_acc\_diag1 = content.get("ocu\_acc\_diag1")*

*historia.ocu\_acc\_emp2 = content.get("ocu\_acc\_emp2")*

*historia.ocu\_acc\_diag2 = content.get("ocu\_acc\_diag2")*

*historia.ocu\_obs = content.get("ocu\_obs")*

*historia.cie\_concep\_desc = content.get("cie\_concep\_desc")*

*historia.cie\_concep\_reco = content.get("cie\_concep\_reco")*

*historia.cie\_concep\_aplaz = content.get("cie\_concep\_aplaz")*

*historia.cie\_concep\_aplaza = content.get("cie\_concep\_aplaza")*

*historia.cie\_concep\_reco\_mot = content.get("cie\_concep\_reco\_mot")*

*historia.cie\_obs = content.get("cie\_obs")*

*historia.cie\_concep\_fin = content.get("cie\_concep\_fin")*

*historia.fecha\_cierre = get\_datetime()*

*db.session.commit()*

*return ({*

*"response": "La historia médica se ha actualizado"*

*}, 200)*

# 11. Configuración y Ejecución de la Base de Datos

*En esta sección se describe paso a paso el proceso de configuración de la base de datos, si es necesario especificar el proceso de instalación del motor, el cliente de BD o del server, en caso contrario hacer referencia a manuales externos que deben estar incluidos en los anexos, se debe mostrar el proceso de ejecución del Script y evidenciar la creación de la BD*

# 12. Otras Consideraciones

Ciertas tablas tendrán subconjunto de valores.