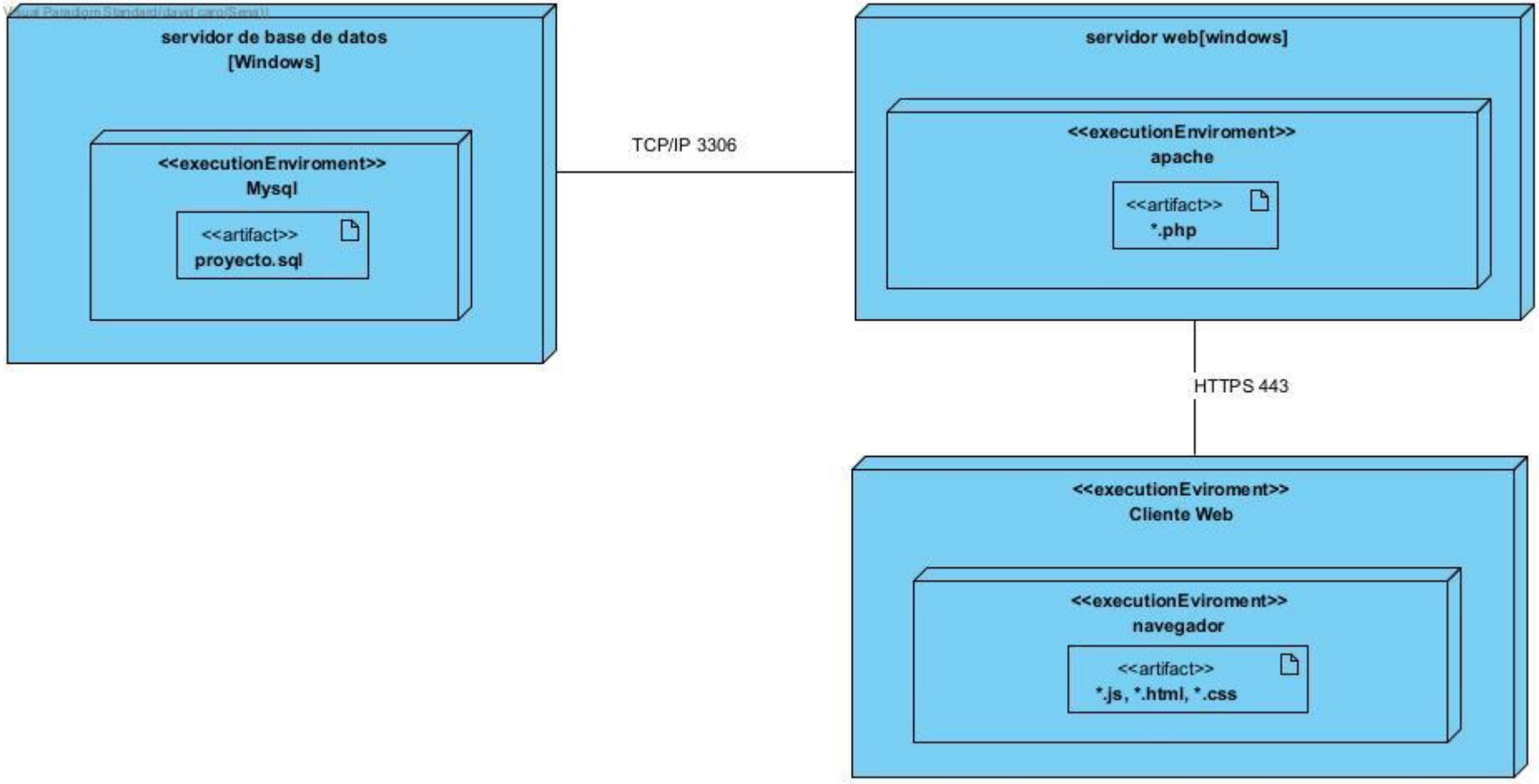
DIAGRAMA DE DESPLIEGUE.



Servicio Nacional de Aprendizaje SENA

Centro Electricidad, Electrónica y Telecomunicaciones

MODELO ENTIDAD RELACIÓN.

SCRIPT BD

```
drop database if exists perezLaraCiaLtda;
create database if not exists perezLaraCiaLtda;
use perezLaraCiaLtda;

create table if not exists tipoDocumento
(
    idTipoDocumento int auto_increment primary key,
    descripcionDocumento varchar(45) not null,
    abreviacion varchar(2)
);

create table if not exists estado
(
    idEstado int auto_increment primary key,
    estado varchar(45)
);

/* Usuarios
   ========== */
create table if not exists estadoUsuario
(
    idEstado int auto_increment primary key,
    estado tinyint(1) not null
);

create table if not exists tipoUsuario
(
    idTipoUsuario int auto_increment primary key,
    tipoUsuario tinyint(1) not null
);

create table if not exists usuario
(
```

```
    documento int primary key,
    primerNombre varchar(45) not null,
    segundoNombre varchar(45),
    primerApellido varchar(45) not null,
    segundoApellido varchar(45),
    celular varchar(15) not null,
    email varchar(45) not null UNIQUE,
    clave varchar(45) not null,
    tipoDocumento int not null,
    tipoUsuario int not null,
    estadoUsuario int not null,
    foreign key(tipoDocumento) references tipoDocumento(idTipoDocumento),
    foreign key (tipoUsuario) references tipoUsuario(idTipoUsuario),
    foreign key (estadoUsuario) references estadoUsuario(idEstado)
);

create table if not exists notificacion
(
    idNotificacion int auto_increment primary key,
    fecha datetime not null,
    titulo varchar(45),
    mensaje varchar(45),
    estado int not null,
    foreign key (estado) references estado(idEstado)

);

create table if not exists notificacionUsuario
(
    documento int not null,
    notificacion int not null,
    foreign key (documento) references usuario(documento),
    foreign key (notificacion) references notificacion(idNotificacion)
);

/* Clientes
```

```
========== */
create table if not exists departamento
(
    idDepartamento int auto_increment primary key,
    departamento varchar(60) not null
);

create table if not exists ciudad
(
    idCiudad int auto_increment primary key,
    ciudad varchar(60) not null,
    departamento int not null,
    foreign key(departamento) references departamento(idDepartamento)
);

create table if not exists cliente
(
    documento int auto_increment primary key,
    primerNombre varchar(45) not null,
    segundoNombre varchar(45),
    primerApellido varchar(45) not null,
    segundoApellido varchar(45),
    fechaNacimiento date not null,
    email varchar(45) not null,
    telefono1 varchar(15) not null,
    telefono2 varchar(15),
    direccion1 varchar(100) not null ,
    direccion2 varchar(100),
    tipoDocumento int not null,
    ciudad int not null,
    estado int not null,
    foreign key (tipoDocumento) references tipoDocumento(idTipoDocumento),
    foreign key (ciudad) references ciudad(idCiudad),
    foreign key (estado) references estado(idEstado)
);
```

```
/* Productos
   ========== */
create table if not exists aseguradora
(
    idAseguradora int auto_increment primary key,
    aseguradora varchar(45) not null,
    logo varchar(45) not null
);

create table if not exists seguro
(
    idSeguro int auto_increment primary key,
    seguro varchar(45) not null,
    imagen varchar(45) not null
);

create table if not exists seguroAseguradora
(
    seguro int not null,
    aseguradora int not null,
    foreign key (seguro) references seguro(idSeguro),
    foreign key (aseguradora) references aseguradora(idAseguradora)
);

/* Trabajo
   ========== */
create table if not exists cotizacion
(
    idCotizacion int auto_increment primary key,
    documentoCliente int not null,
    fecha date not null,
    seguro int not null,
    estado int not null,
    cuadroComparativo varchar(45) not null,
    foreign key (documentoCliente) references cliente(documento),
    foreign key (seguro) references seguro(idSeguro),
```

```
    foreign key (estado) references estado(idEstado)
);

create table if not exists aseguradoraCotizante
(
    cotizacion int not null,
    aseguradora int not null,
    foreign key (cotizacion) references cotizacion(idCotizacion),
    foreign key (aseguradora) references aseguradora(idAseguradora)
);

create table if not exists poliza
(
    codigo int auto_increment primary key,
    documentoCliente int not null,
    fecha date not null,
    seguro int not null,
    aseguradora int not null,
    archivo varchar(45) not null,
    estado int not null,
    foreign key (documentoCliente) references cliente(documento),
    foreign key (seguro) references seguro(idSeguro),
    foreign key (aseguradora) references aseguradora(idAseguradora),
    foreign key (estado) references estado(idEstado)
);

create table if not exists siniestro
(
    idSiniestro int auto_increment primary key,
    fecha date not null,
    descripcion varchar(45) not null,
    monto DOUBLE,
    codigoPoliza int not null,
    anexo varchar(45) not null,
    estado int not null,
    foreign key (codigoPoliza) references poliza(codigo),
```

```
    foreign key (estado) references estado(idEstado)
);

create table if not exists reporte
(
    idReporte int auto_increment primary key,
    fecha date not null,
    titulo varchar(45) not null,
    archivo varchar(45) not null,
    usuario int not null,
    foreign key (usuario) references usuario(documento)
);

/* VIEWS */
create view info_usuario as
    select
        concat(primerNombre,' ',segundoNombre,' ',primerApellido,'
',segundoApellido) as Nombre,
        email as Email,celular as Celular,t2.tipoUsuario as Rol
    from usuario
        inner join tipodocumento t on usuario.tipoDocumento = t.idTipoDocumento
        inner join tipousuario t2 on usuario.tipoUsuario = t2.idTipoUsuario
        inner join estadousuario e on usuario.estadoUsuario = e.idEstado;

create view info_cliente as
    select
        concat(t.abreviacion,' ',documento) as `No. Documento`,
        concat(primerNombre,' ',segundoNombre,' ',primerApellido,'
',segundoApellido) as Cliente,
        d.departamento as Departamento, c.ciudad as Ciudad, concat(direccion1,'
',direccion2) as Direccion,
        fechaNacimiento as `Fecha de Nacimiento`, email as Email, concat(telefono1,'
',telefono2) as Telefono, e.estado as Estado
    from cliente
        inner join tipodocumento t on cliente.tipoDocumento = t.idTipoDocumento
        inner join ciudad c on cliente.ciudad = c.idCiudad
```

```
        inner join departamento d on c.departamento = d.idDepartamento
        inner join estado e on cliente.estado = e.idEstado;

create view info_cotizacion as
    select
        fecha as Fecha, idCotizacion as `No. Cotizacion`, concat(c.primerNombre,'
',c.segundoNombre,' ',c.primerApellido,' ',c.segundoApellido) as Cliente,
        s.seguro as Seguro, a2.aseguradora as Aseguradora, e.estado as Estado
    from cotizacion
        inner join cliente c on cotizacion.documentoCliente = c.documento
        inner join seguro s on cotizacion.seguro = s.idSeguro
        inner join aseguradoracotizante a on cotizacion.idCotizacion = a.cotizacion
        inner join aseguradora a2 on a.aseguradora = a2.idAseguradora
        inner join estado e on cotizacion.estado = e.idEstado;

create view info_poliza as
    select fecha as Fecha,codigo as `Codigo poliza`,
            concat(primerNombre,' ',segundoNombre,' ',primerApellido,'
',segundoApellido) as Cliente,
            s.seguro as Seguro,a.aseguradora as Aseguradora,e.estado as Estado
    from poliza
        inner join cliente c on poliza.documentoCliente = c.documento
        inner join seguro s on poliza.seguro = s.idSeguro
        inner join aseguradora a on poliza.aseguradora = a.idAseguradora
        inner join estado e on poliza.estado = e.idEstado;

create view info_siniestro as
    select idSiniestro as `No. Referencia`,
            concat(primerNombre,' ',segundoNombre,' ',primerApellido,'
',segundoApellido) as Cliente,
            codigo as `Codigo poliza`,p.seguro as Seguro,a.aseguradora as Aseguradora,
siniestro.fecha as Fecha,
            siniestro.estado as Estado
    from siniestro
        inner join poliza p on siniestro.codigoPoliza = p.codigo
        inner join seguro s on p.seguro = s.idSeguro
```

```
inner join aseguradora a on p.aseguradora = a.idAseguradora
inner join cliente c on p.documentoCliente = c.documento
inner join estado e on siniestro.estado = e.idEstado;


/* PROCEDURES */


CREATE PROCEDURE `sp_insert_clientes`(IN `documento` INT(20), IN
`primerNombre` VARCHAR(45), IN `segundoNombre` VARCHAR(45), IN
`primerApellido` VARCHAR(45), IN `segundoApellido` VARCHAR(45),
IN `email` VARCHAR(45), IN `tipoDocumento` INT, IN `fechaNacimiento`
DATE, IN `estado` INT, IN `ciudad` INT, IN `telefono1` VARCHAR(15), IN
`telefono2` VARCHAR(15), IN `direccion1` VARCHAR(100),
IN `direccion2` VARCHAR(100))
INSERT INTO cliente
(documento,primerNombre,segundoNombre,primerApellido,segundoApellido,ema
il,
tipoDocumento, fechaNacimiento,
estado,ciudad,telefono1,telefono2,direccion1,direccion2)
VALUES
(documento,primerNombre,segundoNombre,primerApellido,segundoApellido,ema
il,tipoDocumento,
fechaNacimiento, estado,ciudad,telefono1,telefono2,direccion1,direccion2);

CREATE PROCEDURE `sp_update_clientes`(IN `documento` INT(20), IN
`primerNombre` VARCHAR(45), IN `segundoNombre` VARCHAR(45), IN
`primerApellido` VARCHAR(45),
IN `segundoApellido` VARCHAR(45), IN `email` VARCHAR(45), IN
`tipoDocumento` INT, IN `fechaNacimiento` DATE, IN `estado` INT, IN `ciudad`
INT, IN `telefono1` VARCHAR(15), IN `telefono2` VARCHAR(15),
IN `direccion1` VARCHAR(100), IN `direccion2` VARCHAR(100))
UPDATE cliente SET documento=documento, primerNombre=primerNombre,
segundoNombre=segundoNombre, primerApellido=primerApellido,
segundoApellido=segundoApellido, email=email,
```

```
 tipoDocumento=tipodocumento , fechaNacimiento=fechaNacimiento,
estado=estado, ciudad=ciudad,
telefono1=telefono1,telefono2=telefono2,direccion1=direccion1,direccion2=direcc
ion2
WHERE documento=documento;

CREATE PROCEDURE `sp_filtro_clientes`(IN `departamento` INT, IN `ciudad`
INT, IN `estado` INT)
SELECT concat(t.abreviacion,' ',documento) as 'No. Documento',
concat(primerNombre,' ',segundoNombre,' ',primerApellido,' ',segundoApellido) as
'Cliente',
d.departamento as 'Departamento', c.ciudad as 'Ciudad',
concat(direccion1,' ',direccion2) as 'Direccion', fechaNacimiento as 'Fecha de
Nacimiento', email as 'Email', concat(telefono1,' ',telefono2) as 'Telefono',
 e.estado as 'Estado' from cliente inner join tipodocumento t on
cliente.tipoDocumento = t.idTipoDocumento
inner join ciudad c on cliente.ciudad = c.idCiudad inner join estado e on
cliente.estado = e.idEstado
 inner join departamento d on c.departamento = d.idDepartamento WHERE
ciudad.departamento=departamento and cliente.ciudad=ciudad and
cliente.estado=estado;



CREATE  PROCEDURE `sp_insert_poliza`(IN `codigo` INT(45), IN `documento`
INT(45), IN `fecha` DATE, IN `seguro` INT, IN `aseguradora` INT, IN `archivo`
VARCHAR(500), IN `estado` INT)
INSERT into poliza (codigo, documentoCliente, fecha, seguro, aseguradora,
archivo, estado)
VALUES (codigo, documento, fecha, seguro, aseguradora, archivo, estado);

CREATE PROCEDURE `sp_update_poliza`(IN `codigo` INT(45), IN `documento`
INT(45), IN `fecha` DATE, IN `seguro` INT, IN `aseguradora` INT, IN `archivo`
VARCHAR(500), IN `estado` INT)
UPDATE poliza SET codigo=codigo, documentoCliente=documento, fecha=fecha,
seguro=seguro, aseguradora=aseguradora, archivo=archivo, estado=estado
```

```
WHERE codigo=codigo;


CREATE PROCEDURE `sp_insert_siniestro`(IN `fecha` DATE, IN `descripcion`
VARCHAR(200), IN `monto` DOUBLE, IN `codigoPoliza` INT, IN `anexo`
VARCHAR(500), IN `estado` INT)
INSERT INTO siniestro (fecha, descripcion, monto, codigoPoliza, anexo, estado)
VALUES (fecha, descripcion, monto, codigoPoliza, anexo, estado);


CREATE PROCEDURE `sp_update_siniestro`(IN `fecha` DATE, IN `descripcion`
VARCHAR(200), IN `monto` DOUBLE, IN `codigoPoliza` INT, IN `anexo`
VARCHAR(500), IN `estado` INT)
UPDATE siniestro SET fecha=fecha, descripcion=descripcion, monto=monto,
codigoPoliza=codigoPoliza, anexo=anexo, estado=estado
WHERE codigoPoliza=codigoPoliza;


CREATE PROCEDURE `sp_insert_cotizacion`(IN `documento` INT, IN `fecha`
DATE, IN `seguro` INT, IN `estado` INT, IN `cuadroComparativo`
VARCHAR(500))
INSERT INTO cotizacion (documentoCliente, fecha, seguro, aseguradora, estado,
cuadroComparativo)
VALUES (documentoCliente, fecha, seguro, aseguradora, estado,
cuadroComparativo);


CREATE PROCEDURE `sp_update_cotizacion`(IN `documento` INT, IN `fecha`
DATE, IN `seguro` INT, IN `estado` INT, IN `cuadroComparativo`
VARCHAR(500)
UPDATE cotizacion SET documentoCliente=documentoCliente, fecha=fecha,
seguro=seguro, aseguradora=aseguradora, estado=estado,
cuadroComparativo=cuadroComparativo
WHERE documentoCliente=documentoCliente;
```

```
CREATE PROCEDURE `sp_filtro_cotizaciones`(IN `fechaInicial` DATE, IN
`fechaFinal` DATE, IN `documentoCliente` INT, IN `seguro` INT, IN
`aseguradora` INT, IN `estado` INT)
SELECT fecha as Fecha, idCotizacion as Cotizacion, concat(c.primerNombre,'
',c.segundoNombre,' ',c.primerApellido,' ',c.segundoApellido) as Cliente, s.seguro
as Seguro,
a2.aseguradora as Aseguradora from cotizacion inner join cliente c on
cotizacion.documentoCliente = c.documento inner join seguro s on
cotizacion.seguro = s.idSeguro
inner join aseguradoracotizante a on cotizacion.idCotizacion = a.cotizacion
inner join aseguradora a2 on a.aseguradora = a2.idAseguradora WHERE
fecha>=fechaInicial and fecha<=fechaFinal and
documentoCliente=documentoCliente and
seguro=seguro and aseguradora=aseguradora and estado=estado;
```

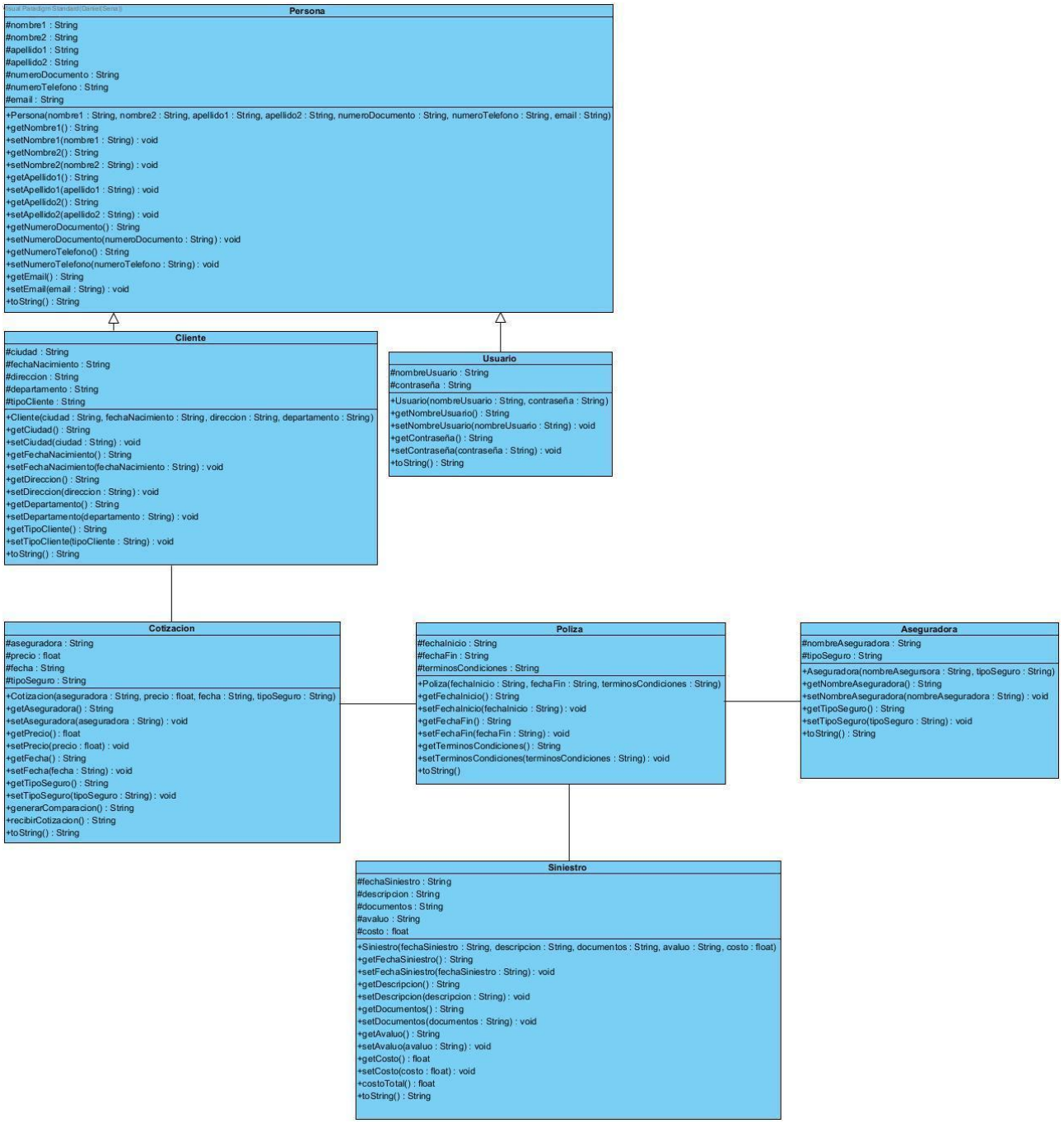
Servicio Nacional de Aprendizaje SENA

Centro Electricidad, Electrónica y Telecomunicaciones

PÉREZ LARA CIA.
Asesores de Seguros
¡Sabemos y podemos!

DICCIONARIO DE DATOS.

DIAGRAMA DE MODELADO DE CLASES.



Servicio Nacional de Aprendizaje SENA

Centro Electricidad, Electrónica y Telecomunicaciones

PLAN DE INSTALACIÓN DEL APLICATIVO

GLOSARIO DE TÉRMINOS.

- **Plataforma :** En informática, plataforma es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de Software.
- **Fork**:Cuando nos referimos a un FORK es un repositorio el cual hace una copia exacta de una BARE del repositorio que es original.
- **Procedure:** procedimiento almacenado en la base de datos que mecaniza los procesos de CRUD de la DB.
- **Script:** Los scripts SQL pueden utilizarse para crear la estructura de la base de datos, realizar operaciones en la base de datos (p. ej. rellenarla con datos) y cambiar o eliminar la estructura de la base de datos
- **Requerimientos de software:** son la descripción de las características y las funcionalidades del sistema 'target'. Los requisitos nos comunican las expectativas de los consumidores de productos software. Los requisitos pueden ser obvios o estar ocultos, conocidos o desconocidos, esperados o inesperados, desde el punto de vista del cliente.
- **Casos de Uso:** Los casos de uso proporcionan una estructura para expresar requisitos funcionales en el contexto de procesos empresariales y de sistema. Los casos de uso pueden representarse como un elemento gráfico en un diagrama y como una especificación de caso de uso en un documento textual.

BIBLIOGRAFÍA.

- https://www.apachefriends.org/es/index.html
- https://es.wikipedia.org/wiki/Visual_Studio_Code
- https://kinsta.com/es/base-de-conocimiento/que-es-github/
- https://definicion.de/plataforma/
- https://aprendegit.com/fork-de-repositorios-para-que-sirve/

CONTROL DE DOCUMENTO.

	Nombre	Cargo	Dependencia	Fecha
Autor(es)	Camila Andrea Torres	Aprendiz/ADSI	Teleinformática	15/05/2022
	David Santiago Caro	Aprendiz/ADSI	Teleinformática	18/05/2022

CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)**.**

	Nombre	Cargo	Dependencia	Fecha
Autor(es)				

MUCHAS GRACIAS