Sistema paqueteria

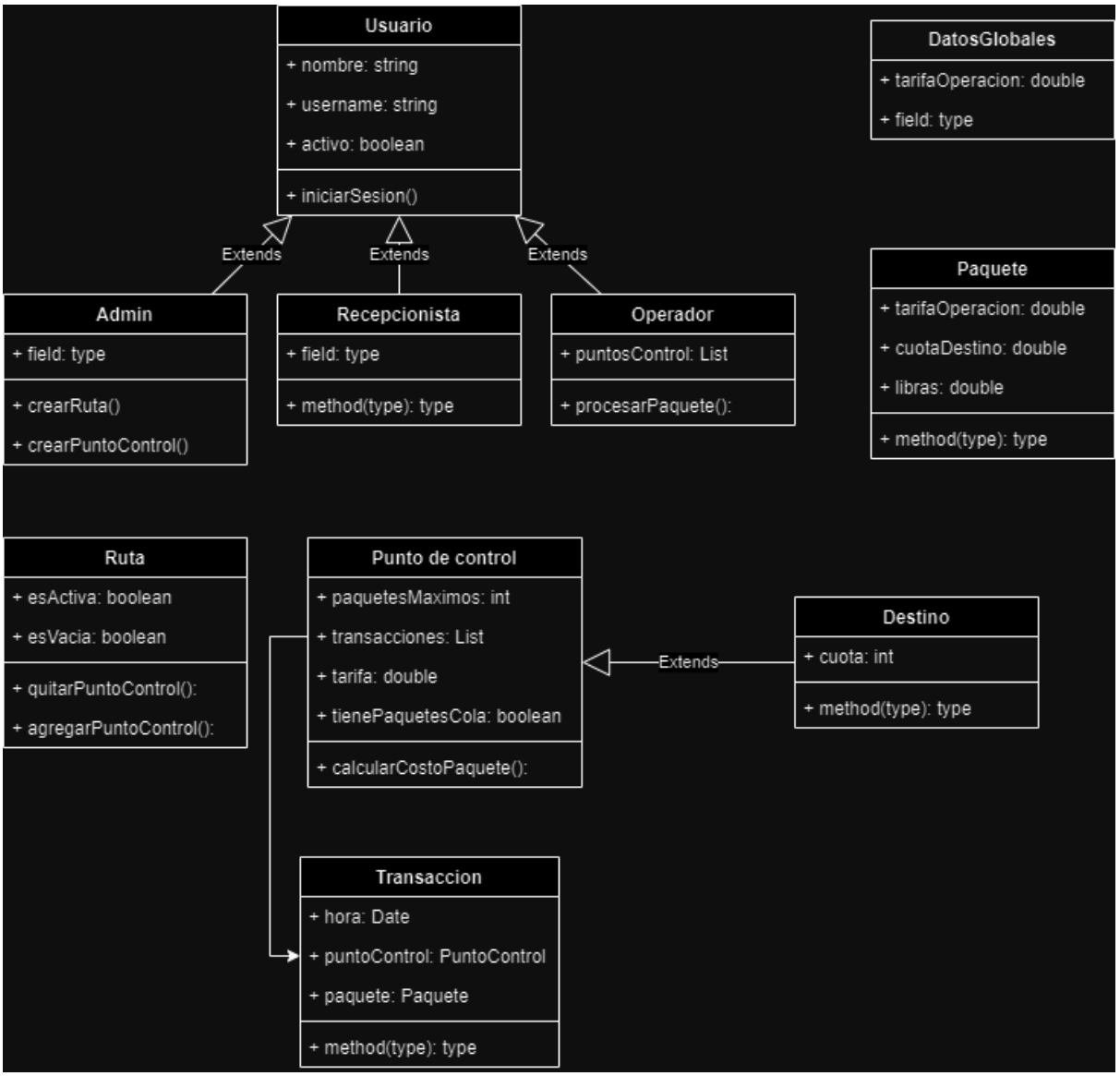
Manual Técnico

Versión: 0100

Fecha:05/04/2023

Desarrollador: Cristian Alejandro Vásquez Escobar, 202131936

Diagrama de clases



Análisis de los requerimiento

Glosario

Cliente

Es la persona que con ayuda del usuario recepcionista realiza un envío o recoge un paquete.

Paquete

Es un objeto con ciertas características que pasará por alguna ruta especificada.

Ruta

Un conjunto de puntos de control por el que pasa el paquete que llevan a un destino para entregarlo.

Ganancia

El resultado del tiempo por el cual el paquete ha pasado por cada uno de los puntos de control dependiendo de una tarifa.

Usuario

La persona final encargada de utilizar el sistema con ciertas características dependiendo de su rol.

Administrador

El usuario con mayor jerarquía encargado de modificar todo lo relacionado con usuarios, rutas y puntos de control; además de modificar ciertos parámetros generales del sistema y la misma creación de más usuarios.

Operador

El usuario asignado a un punto de control, encargado de procesar que el paquete ya ha pasado exitosamente por ahí.

Recepcionista

El usuario encargado de realizar los pedidos y de entregarlos.

Punto de control

Localización por la cual pasan los paquetes dependiendo de las rutas, encargados por solo un operador.

Destino

Localización en la cual el paquete será entregado, será el último punto de una ruta.

Factura

Documento que desglose los cargos dependiendo por los puntos de control por los cuales los paquetes han pasado y demás factores.

Bodega

Un lugar en el cual los pedidos se pondrán en un estado pendiente y cuando el sistema determine que hay una ruta disponible con el destino que se busca para liberarlo ahí.

Casos de uso de alto nivel

Número:	CU001
Caso de uso:	Iniciar sesión
Actores:	Administrador, Recepcionista, Operador
Descripción:	Cualquiera de los usuarios ingresa al sistema con sus credenciales y muestra los módulos disponibles para cada tipo de usuario.
Tipo:	Primario

Número:	CU002
Caso de uso:	Crear ruta
Actores:	Administrador, Operador, Recepcionista

Descripción:	Se crea una ruta con sus respectivos puntos de control, asignando un usuario operador a cada uno de ellos y creando un punto de origen el cual se le asigna a un recepcionista. Se crea un destino con una cuota propia.
Tipo:	Primario

Número:	CU003
Caso de uso:	Ver reportes
Actores:	Administrador
Descripción:	Muestra las opciones de todos los reportes disponibles y al clicar lo lleva al reporte especificado.
Tipo:	Secundario

Número:	CU004
Caso de uso:	Modificar datos globales
Actores:	Administrador
Descripción:	El administrador modifica todo tipo de datos globales que pueden afectar a los pedidos a futuro
Tipo:	Secundario

Número:	CU005
Caso de uso:	Modificar ruta
Actores:	Administrador
Descripción:	El administrador tiene la opción de desactivar o activar una ruta y modificar el punto de destino
Tipo:	Secundario

Número:	CU006

Caso de uso:	Mostrar usuarios
Actores:	Administrador
Descripción:	El administrador puede ver todos los usuarios del sistema y modificarlos, permitiendo la opción de agregar más.
Tipo:	Secundario

Número:	CU007
Caso de uso:	Ver puntos de control
Actores:	Operador
Descripción:	Muestra los puntos de control asignados a él mismo, al seleccionar uno de ellos dará la opción de ver los pedidos en cola y la opción para procesarlos.
Tipo:	Primario

Número:	CU008
Caso de uso:	Consultar pedidos en las rutas
Actores:	Recepcionista, Cliente
Descripción:	El recepcionista consulta el punto de control en el que se encuentra el paquete de un cliente.
Tipo:	Secundario

Número:	CU009
Caso de uso:	Ver pedidos no retirados
Actores:	Recepcionista
Descripción:	Despliega un listado de paquetes que ya llegaron a su destino y que no han sido retirados por los dueños.
Tipo:	Secundario

Número:	CU011
Caso de uso:	Ingreso de paquete
Actores:	Recepcionista, Cliente, Bodega
Descripción:	El cliente realiza un pedido con el recepcionista y se envía a la bodega y se verifica si hay alguna ruta disponible para poner en cola.
Tipo:	Primario

Número:	CU012
Caso de uso:	Registro paquete entregado
Actores:	Recepcionista, Cliente
Descripción:	El recepcionista verifica que el paquete ya haya llegado a ese punto de destino y se le entrega el paquete
Tipo:	Primario

Número:	CU013
Caso de uso:	Generar factura
Actores:	Cliente, facturación
Descripción:	Se verifica si el NIT existe en el sistema, de lo contrario se agrega al cliente, y se genera una factura en base a factores del historial del pedido.
Tipo:	Primario

Diagrama de casos de uso

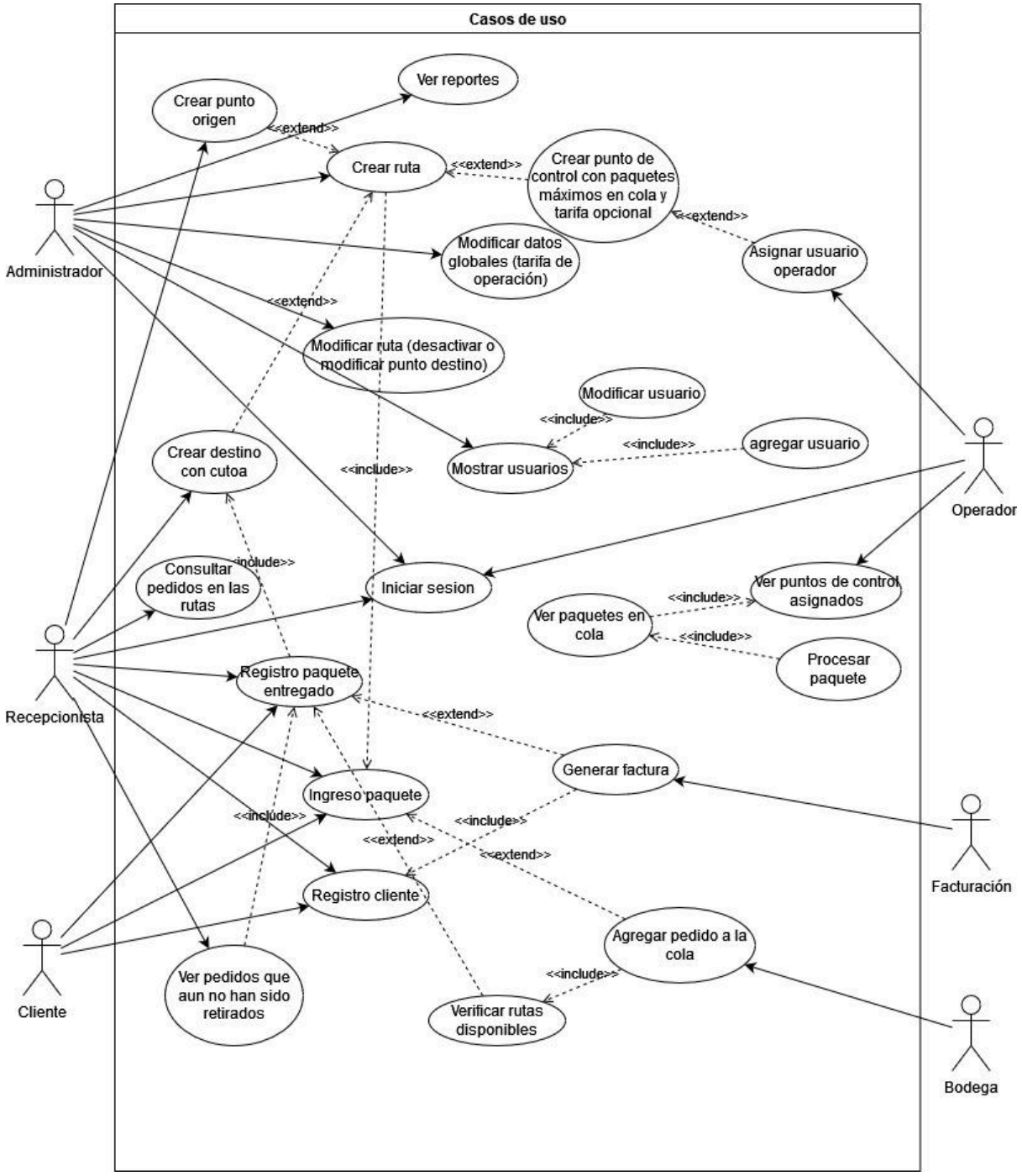


Diagrama E/R

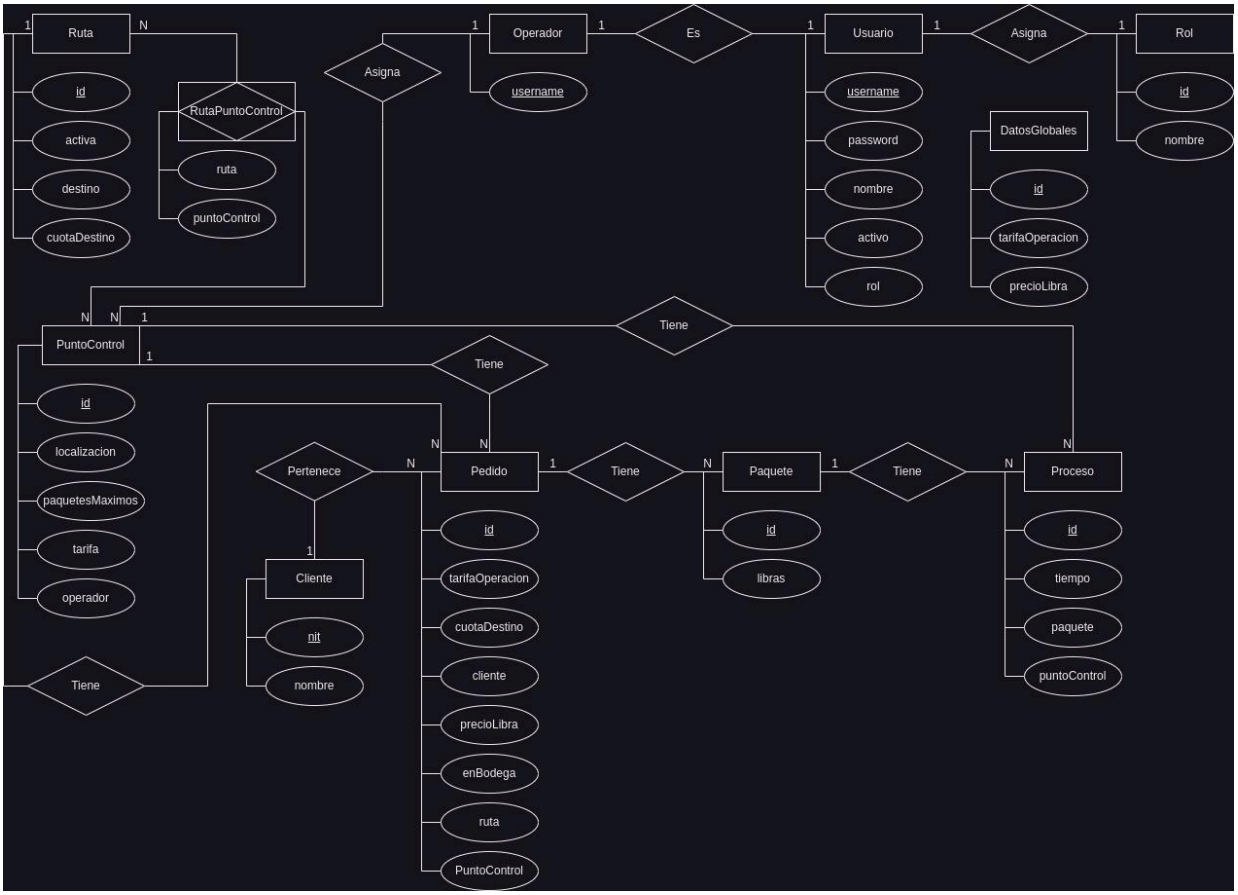
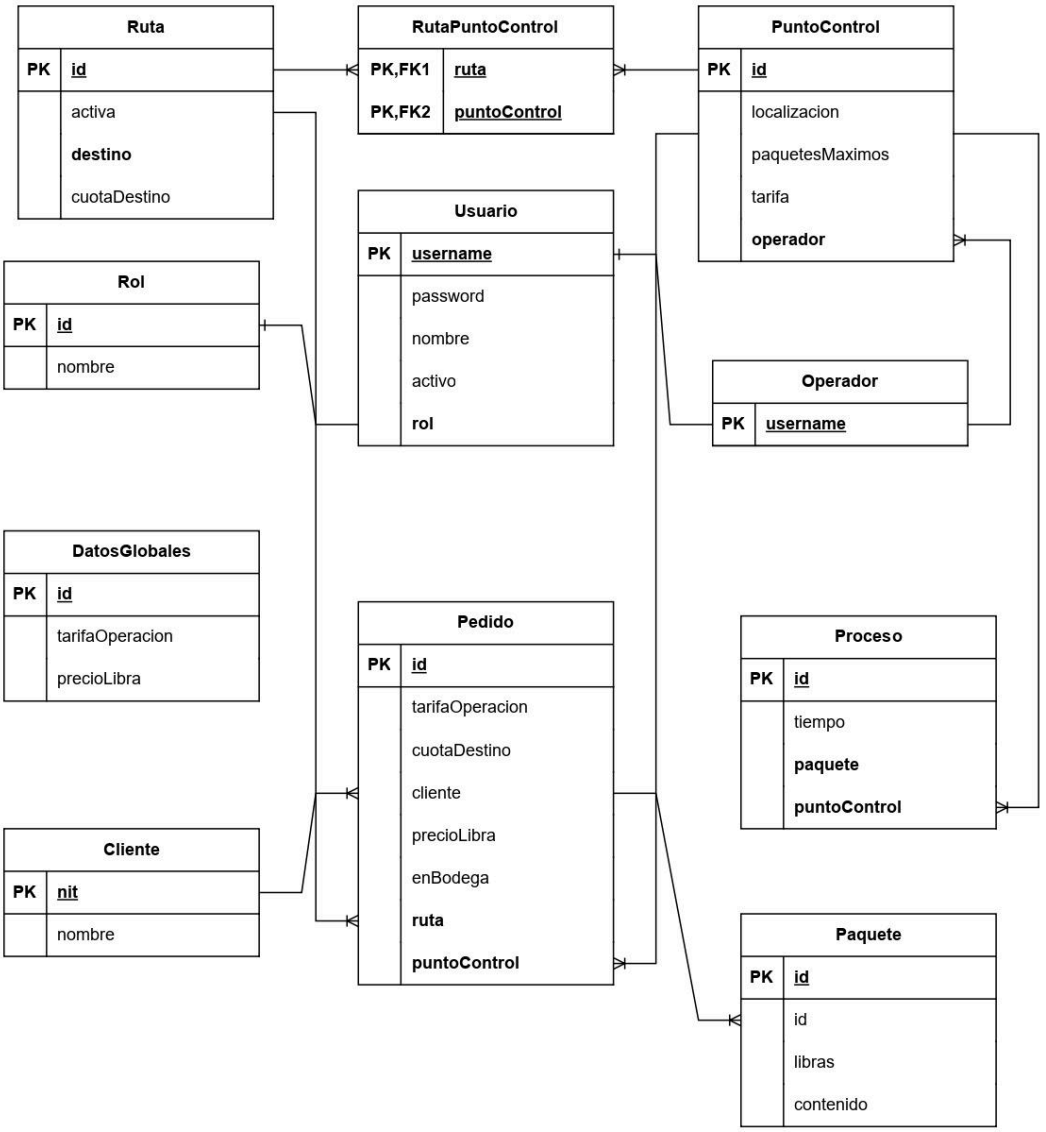
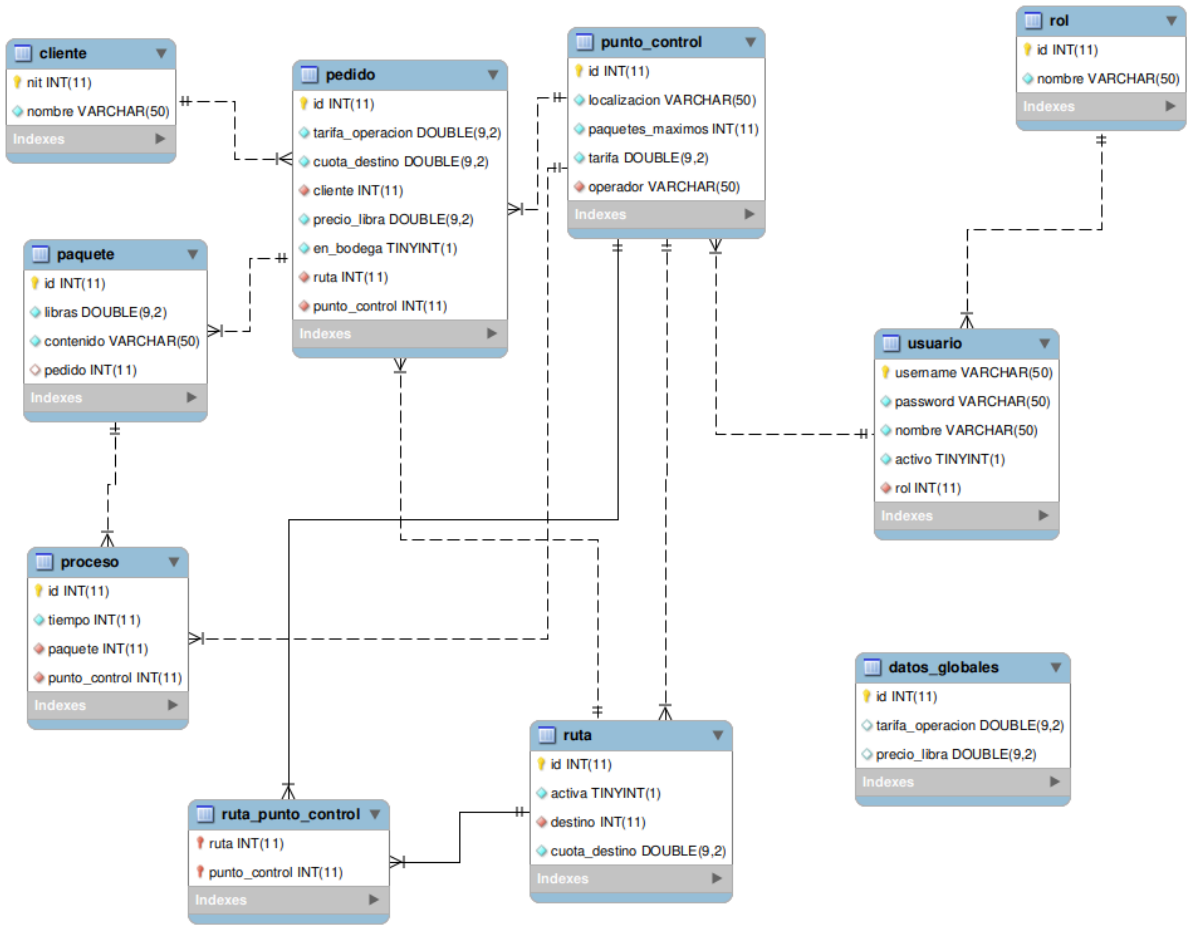

Ruta
Operador
Es
Usuario
Asigna
Rol
id
activa
destino
cuotaDestino
RutaPuntoControl
ruta
puntoControl
Asigna
username
username
password
nombre
activo
rol
DatosGlobales
id
tarifaOperacion
precioLibra
id
nombre
PuntoControl
Tiene
Tiene
id
localizacion
paquetesMaximos
tarifa
operador
Tiene
Pertenece
Cliente
nit
nombre
Pedido
Tiene
Paquete
Tiene
Proceso
id
tarifaOperacion
cuotaDestino
cliente
precioLibra
enBodega
ruta
PuntoControl
id
libras
id
tiempo
paquete
puntoControl

Diagrama de tablas



Mapeo físico de la DB



Requerimientos para la ejecución del programa

- Una computadora
- Se necesita tener go instalado en una computador
- Alguna terminal para poder interpretar comandos, si se encuentra en windows con el cmd o el windows powershell será suficiente.

Modelos

Son archivos creados en Goland para simular las clases y de esa manera usarlos para el backend.

```
packag  model
e      s
type PuntoControl struc  {
    Id            t int
    Localizacio     strin
    MaquetesMaximo  gint
    Tarif           float3
    Operado         2strin
}   r               g
```

```
packag  model
e      s
type DatosGlobales struc  {
    TarifaOperacio tfloat3
    PrecioLibr      2float3
}   a               2
```

```
packag  model
e      s
type RutaModel struc  {
    Id         t int
    Activ        bool
    Destin       int
    CuotaDestin  float3
}   o            2
```

Controllers

Sirven para poder crear funciones que se ejecutan en cada ruta

```
packag  controller
e       s
impor   (
t    "backend/initializer "
     "backend/model "
     "database/sq "
     "fmt"
     "net/htt "
      p
     "github.com/gin-gonic/gi "
)     n

func handleErro (c *gin.Context, statusCod  int,
messag  strin , err error) {      e
e    fmt.Printl ("Erro  ", err)
     c.JSON(statusCod , gin.H "error": messag })
}          e         {               e

func CreateUsuari (c *gin.Context)
     var body models.Usuari
              o
     if err := c.BindJSO (&body); err != nil {
          handleErro (c, http.StatusBadReques , "
Invalid request bod ", err) t
y         retur
     }    n

     sqlStatemen  := "
INSERT INTO usuario (username, password, nombre, ac
tivo, rol) VALUES (?, ?, ?, ?, ?)
"

     stmt, err := initializer .DB.Prepar (
sqlStatemen )     s               e
t    if err != nil {
          handleErro (c, http.
StatusInternalServerErro , "
Error preparing statemen ", err)
t         retur
     }    n
     defer stmt.Close()

     _, err = stmt.Exec(body.Usernam , body.Passwor
, body.Nombr , body.Activ , body.Rol)       d
     if err != nil { o
          handleErro (c, http.
StatusInternalServerErro , "
Error executing statemen ", err)
t         retur
     }    n
```

Enrutamiento

A través del framework de gin se crea el enrutamiento para que dependiendo de la ruta se ejecute cierta función.

Conección a base de datos

```
packag  initializer
e       s
impor   (
t    "database/sq "
     "log"
     "os"

     _ "github.com/go-sql-driver/mysq "
)       l

var DB *sql.D
         B
func ConnectToD () {
     vBr err error
     dsn := os.Geten ("DB_UR ")
                v       L
     // Open database connectio
     DBn err = sql.Open("mysql", dsn)
     if err != nil {
         log.Fatal("
Error connecting to the databas  ", err)
e:   }

     // Ping database to verify connectio
     err = DB.Ping()
     if err != nil {
         log.Fatal("Error pinging databas  ", err)
     }              e:

     log.Printl ("Connected to the databas ")
}        n         e
```

Cargar variables locales

```
packag  initializer
e       s
impor   (
t    "log"

     "github.com/joho/godoten "
)     v

func LoadEnvVariable () {
     err := godoten .Load()
     if err != nil {
          log.Fatal("Error loading .env fil ")
     }              e

}
```