**Manual Técnico**

# **Propósito.**

La creación de este documento está fundamentada en mostrar cómo fue diseño, qué características tiene y cómo guiarse por especificaciones para una intervención de mantenimiento futura en el sistema.

# **Alcance.**

Este documento está dirigido a una persona con conocimientos informáticos, más específicamente un ingeniero en sistemas de la información.

# **Uso del manual técnico.**

El uso de este documento sólo debe ser para el mantenimiento o intervención extraordinaria para la solución de problemas que se puedan presentar.

# **Descripción de módulos.**

* Nombre del módulo: Registro.

Funcionalidad/Propósito: Este módulo tiene como objetivo registrar todos los usuarios que manejaran los procesos del sistema.

* Nombre del módulo: Registro Afiliados.

Funcionalidad/Propósito: Este módulo tiene como objetivo registrar todos los afiliados de la Asociación. Así mismo los Apiarios, ubicaciones de estos y otorgarles los apiarios a los afiliados.

Dependencias funcionales: Debe existir Usuario.

* Nombre del módulo: Recepción.

Funcionalidad/Propósito: Este módulo tiene como objetivo realizar una recepción de miel y cera cada vez que los afiliados lleguen a ofrecer la misma.

Dependencias funcionales: Debe existir Afiliado y Usuario.

* Nombre del módulo: Planta.

Funcionalidad/Propósito: Este módulo tiene como objetivo realizar los registros de estañones y la recepción de estañón.

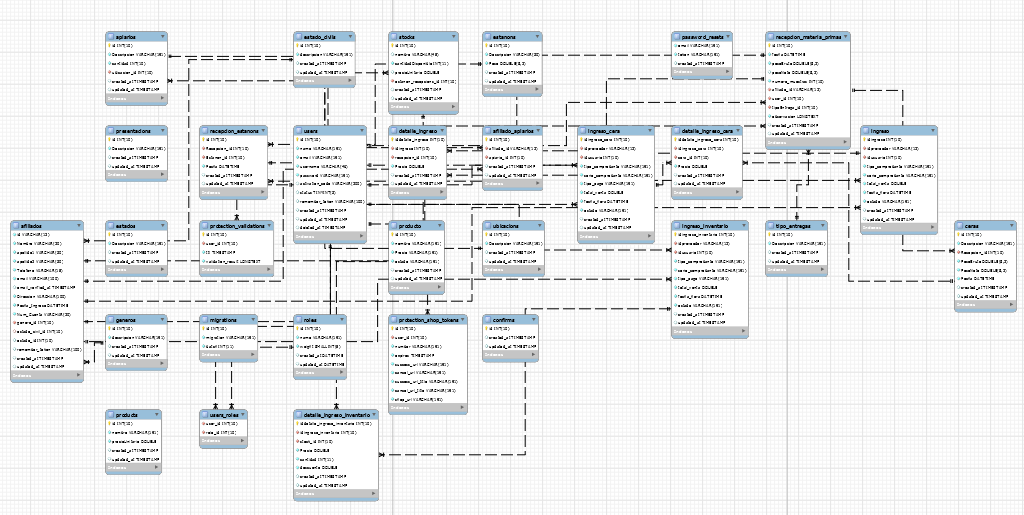
Dependencias funcionales: Debe existir una recepción.

* Nombre del módulo: Inventario.

Funcionalidad/Propósito: Este módulo tiene como objetivo realizar el inventario de los productos, los servicios de cera y miel.

Dependencias funcionales: Debe existir recepción, Afiliado y Usuario.

**Modelo entidad-relación de la base de datos.**

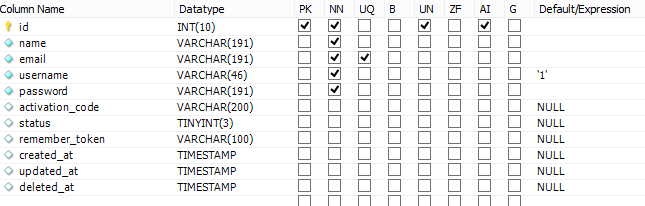


Como trabajamos con Laravel nuestra base de datos no necesita procedimientos almacenados, todo las funciones se hacen desde el controlador.

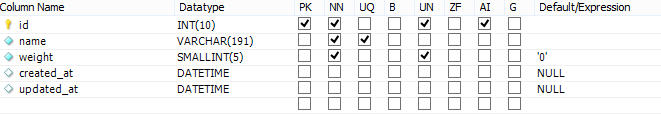
|  |  |
| --- | --- |
| Nombre de función | Descripción |
| RegistrarController | Registrar Usuarios mediante confirmación de correo. |
| protected *function* create(*array* $data)  {  /\*\* @var $user User \*/  $user = *User*::create([    ]);  if (config('auth.users.default\_role')) {  $user->roles()->attach(*Role*::firstOrCreate(['name' => config('auth.users.default\_role')]));  }  return $user;  }  protected *function* register(*Request* $request)  {    /\*\* @var *User* $user \*/  $validatedData = $request->validate([  'name' => 'required|string|max:255',  'email' => 'required|string|email|max:255|unique:users',  'password' => 'required|string|min:6|confirmed',  ]);  try {  $validatedData['password'] = bcrypt(array\_get($validatedData, 'password'));  $validatedData['activation\_code'] = str\_random(30).time();  $user  = app(*User*::class)->create($validatedData);      } catch (\*Exception* $exception) {  dd($exception);  logger()->error($exception);  return redirect()->back()->with('message', 'imposible crear una cuenta.');  }  if (config('auth.users.default\_role')) {  $user->roles()->attach(*Role*::firstOrCreate(['name' => config('auth.users.default\_role')]));  }  $user->notify(new *UserActivate*($user));      return redirect()->back()->with('message', 'Se ha creado exitosamente una nueva cuenta. Por favor revise su correo para activar su cuenta.');    }  /\*\*  \* Activate the user with given activation code.  \* @param string $activationCode  \* @return string  \*/  public *function* activateUser(*string* $activationCode)  {  try {  $user = app(*User*::class)->where('activation\_code', $activationCode)->first();  if (!$user) {  return "El código no existe en nuestro sistema.";  }  $user->status = 1;  $user->activation\_code = null;  $user->save();  auth()->login($user);  } catch (\*Exception* $exception) {  logger()->error($exception);  return "Whoops! sucedió un error.";  }  return redirect()->to('/dashboard');  } | |
| LoginController | Iniciar sesión del usuario autenticado en el sistema. |
| protected $redirectTo = '/dashboard';  /\*\*  \* Create a new controller instance.  \*  \* @return void  \*/  protected *function* credentials(*Request* $request)  {  $field = $this->field($request);  return [  $field => $request->get($this->username()),  'password' => $request->get('password'),  'status' => *User*::STATUS,  ];  }  /\*\*  \* Determine if the request field is email or username.  \*  \* @param \Illuminate\Http\*Request* $request  \* @return string  \*/  public *function* field(*Request* $request)  {  $email = $this->username();      return filter\_var($request->get($email), FILTER\_VALIDATE\_EMAIL) ? $email : 'username';  }  /\*\*  \* Validate the user login request.  \*  \* @param \Illuminate\Http\*Request* $request  \* @return void  \*/  protected *function* validateLogin(*Request* $request)  {  $field = $this->field($request);  $messages = ["{$this->username()}.exists" => 'La cuenta que está intentado ingresar no ha sido activado.'];  $this->validate($request, [  $this->username() => "required|exists:users,{$field},status," . *User*::STATUS,  'password' => 'required',  ], $messages);  } | |
| UserController | Modificar o eliminar los usuarios registrados asignando el rol que sea necesario para el manejo del sistema. |
| public *function* edit(*User* $user)  {  return view('users.edit', ['user' => $user, 'roles' => *Role*::get()]);  }  /\*\*  \* Update the specified resource in storage.  \*  \* @param \Illuminate\Http\*Request* $request  \* @param *User* $user  \* @return mixed  \*/  public *function* update(*Request* $request, *User* $user)  {      $user->name = $request->get('name');  $user->email = $request->get('email');    if ($request->has('password')) {  $user->password = bcrypt($request->get('password'));  }  $user->status = $request->get('status', 0);    $user->save();  //roles  if ($request->has('roles')) {  $user->roles()->detach();  if ($request->get('roles')) {  $user->roles()->attach($request->get('roles'));  }  }    return redirect()->intended(route('users.index'))->with('success', 'Profile updated!');;    }  /\*\*  \* Remove the specified resource from storage.  \*  \* @param int $id  \* @return \Illuminate\Http\*Response*  \*/  /\*DELETEEEEEE\*/  public *function* deleteUser(*request* $request){    $afi = *User*::find($request->id);  $afi->delete();    } | |
| AfiliadoController | Añadir, actualizar y eliminar Afiliados en el sistema. |
| public *function* addAfiliado(*Request* $request){  $rules = array(    'id' => 'min:6|max:9|required',  'Nombre' => 'min:1|max:120|required',  'apellido1' => 'required',  'apellido2' => 'required',  'Telefono' => 'numeric|required',  'email' => 'required',  'Direccion' => 'required',  'Fecha\_Ingreso' => 'required',  'Num\_Cuenta' => 'numeric|required',  'genero\_id' => 'required',  'estado\_civil\_id' => 'required',  'estado\_id' => 'required'    );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));    else {  $afi = new *Afiliado*;  $afi->id= $request->id;  $afi->Nombre = $request->Nombre;  $afi->apellido1 = $request->apellido1;  $afi->apellido2 = $request->apellido2;  $afi->Telefono = $request->Telefono;  $afi->email = $request->email;  $afi->Direccion = $request->Direccion;  $afi->Fecha\_Ingreso = $request->Fecha\_Ingreso;  $afi->Num\_Cuenta = $request->Num\_Cuenta;  $afi->genero\_id = $request->genero\_id;  $afi->estado\_civil\_id = $request->estado\_civil\_id;  $afi->estado\_id = $request->estado\_id;  $afi->save();  return response()->json(['success' => 'Se ha creado un Afiliado correctamente']);  }  }  public *function* editAfiliado(*request* $request){  $rules = array(  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));    else {  $afi = *Afiliado*::find ($request->id);    $afi->Nombre = $request->Nombre;  $afi->apellido1 = $request->apellido1;  $afi->apellido2 = $request->apellido2;  $afi->Telefono = $request->Telefono;  $afi->email = $request->email;  $afi->Direccion = $request->Direccion;  $afi->Fecha\_Ingreso = $request->Fecha\_Ingreso;  $afi->Num\_Cuenta = $request->Num\_Cuenta;  $afi->genero\_id = $request->genero\_id;  $afi->estado\_civil\_id = $request->estado\_civil\_id;  $afi->estado\_id = $request->estado\_id;  $afi->save();  return response()->json($afi);  }  }  public *function* deleteAfiliado(*request* $request){    $afi = *Afiliado*::find ($request->id);  $afi->delete();    } | |
| UbicacionController | Agregar, editar y eliminar ubicaciones en el sistema. |
| public *function* addUbicacion(*Request* $request){  $rules = array(  'Descripcion' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {  $ubicacion = new *Ubicacion*;  $ubicacion->Descripcion = $request->Descripcion;  $ubicacion->save();    return response()->json($ubicacion,['success' => 'Se ha creado una ubicación correctamente']);  }  }  public *function* editUbicacion(*request* $request){  $rules = array(  'Descripcion' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {  $ubicacion =*Ubicacion*::find ($request->id);  $ubicacion->Descripcion = $request->Descripcion;  $ubicacion->save();  return response()->json(['success' => 'Se ha editado correctamente']);  }  }  public *function* deleteUbicacion(*request* $request){    $ubicacion = *Ubicacion*::find($request->id);  $ubicacion->delete();    } | |
| ApiarioController | Agregar, editar y eliminar los Apiarios del sistema. |
| public *function* addApiario(*Request* $request){    $validator = *Validator*::make($request->all(), [    'Descripcion' => 'required',  'cantidad' => 'required',  'ubicacion\_id' => 'required'  ]);    if ($validator->fails())  return *Response*::json(['errors' => $error->errors()->all()]);  else {    $ubicacion\_id = *input*::get('ubicacion\_id');  $api = new *Apiario*;  $api->Descripcion = $request->Descripcion;  $api->cantidad = $request->cantidad;  $api->ubicacion\_id = $request->ubicacion\_id;    $api->save();      return response()->json(['success' => 'Se ha creado un Apiario correctamente']);    }  }  public *function* find(*Request* $request)  {  $term = trim($request->q);  if (empty($term)) {  return \*Response*::json([]);  }  $tags = *Ubicacion*::search($term)->limit(5)->get();  $formatted\_tags = [];  foreach ($tags as $tag) {  $formatted\_tags[] = ['id' => $tag->id, 'text' => $tag->name];  }  return \*Response*::json($formatted\_tags);  }  public *function* editApiario(*request* $request){  $rules = array(  'Descripcion' => 'required',  'cantidad' => 'required',  'ubicacion\_id' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {  $ubicaciones = *Ubicacion*::all();  $api = *Apiario*::find($request->id);  $api->Descripcion = $request->Descripcion;  $api->cantidad = $request->cantidad;  $api->ubicacion\_id = $request->ubicacion\_id;  $api->save();  return response()->json($api);  }  }  public *function* deleteApiario(*request* $request){    $api = *Apiario*::find ($request->id);  $api->delete();  } | |
| AfiliadoApiario | Añadir , editar y eliminar los registros de los afiliados con su respectivo apiario en el sistema. |
| public *function* addApiario(*Request* $request){    $validator = *Validator*::make($request->all(), [    'Descripcion' => 'required',  'cantidad' => 'required',  'ubicacion\_id' => 'required'  ]);    if ($validator->fails())  return *Response*::json(['errors' => $error->errors()->all()]);  else {    $ubicacion\_id = *input*::get('ubicacion\_id');  $api = new *Apiario*;  $api->Descripcion = $request->Descripcion;  $api->cantidad = $request->cantidad;  $api->ubicacion\_id = $request->ubicacion\_id;    $api->save();    return response()->json(['success' => 'Se ha creado un Apiario correctamente']);    }  }  public *function* find(*Request* $request)  {  $term = trim($request->q);  if (empty($term)) {  return \*Response*::json([]);  }  $tags = *Ubicacion*::search($term)->limit(5)->get();  $formatted\_tags = [];  foreach ($tags as $tag) {  $formatted\_tags[] = ['id' => $tag->id, 'text' => $tag->name];  }  return \*Response*::json($formatted\_tags);  }  public *function* editApiario(*request* $request){  $rules = array(  'Descripcion' => 'required',  'cantidad' => 'required',  'ubicacion\_id' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {  $ubicaciones = *Ubicacion*::all();  $api = *Apiario*::find($request->id);  $api->Descripcion = $request->Descripcion;  $api->cantidad = $request->cantidad;  $api->ubicacion\_id = $request->ubicacion\_id;  $api->save();  return response()->json($api);  }  }  public *function* deleteApiario(*request* $request){    $api = *Apiario*::find ($request->id);  $api->delete();  } | |
| RecepcionMateriaPrimaController | Añadir , editar y eliminar las recepciones de miel en el sistema. |
| public *function* addRecepcionMateriaPrima(*Request* $request){  $rules = array(    );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {  $recepcion = new *RecepcionMateriaPrima*;    $recepcion->fecha= $request->fecha;  $recepcion->pesoBruto = $request->pesoBruto;  $recepcion->pesoNeto = $request->discount;  $recepcion->numero\_muestras = $request->numero\_muestras;  $recepcion->afiliado\_id = $request->afiliado\_id;  $recepcion->user\_id = $request->user\_id;  $recepcion->tipoEntrega\_id = $request->tipoEntrega\_id;  $recepcion->observacion = $request->observacion;  $recepcion->save();  return response()->json(['success' => 'Se ha creado una Recepción de Materia Prima correctamente']);  }  }  public *function* addRecepcion(*Request* $request){  $rules = array(    'Recepcion\_id' => 'required',  'Estanon\_id' => 'required',  'Fecha' => 'required',    );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {      $recepcionEst = new *RecepcionEstanon*;  $recepcionEst->Recepcion\_id = $request->Recepcion\_id;  $recepcionEst->Estanon\_id = $request->Estanon\_id;  $recepcionEst->Fecha = $request->Fecha;  $recepcionEst->save();  return back()->with('flash','Recepcion Guardada');  return response()->json($recepcionEst);  }  }  public *function* editRecepcion(*request* $request){  $rules = array(    );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {  $recepcion =*RecepcionMateriaPrima*::find ($request->id);  $recepcion->fecha= $request->fecha;  $recepcion->pesoBruto = $request->pesoBruto;  $recepcion->pesoNeto = $request->discount;  $recepcion->numero\_muestras = $request->numero\_muestras;  $recepcion->afiliado\_id = $request->afiliado\_id;  $recepcion->user\_id = $request->user\_id;  $recepcion->tipoEntrega\_id = $request->tipoEntrega\_id;  $recepcion->observacion = $request->observacion;  $recepcion->save();  return response()->json($recepcion);  }  }  public *function* deleteRecepcionMateriaPrima(*request* $request){    $recepcion= *RecepcionMateriaPrima*::find ($request->id);  $recepcion->delete();  } | |
| CeraController | Añadir , editar y eliminar las recepciones de cera en el sistema. |
| public *function* addCera(*Request* $request){  $rules = array(  'Descripcion' => 'required',  'Recepcion\_id' => 'required',  'PesoBruto' => 'required',  'PesoNeto' => 'required',  'Fecha' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {      $cera = new *Cera*;  $cera->Descripcion = $request->Descripcion;  $cera->Recepcion\_id = $request->Recepcion\_id;  $cera->PesoBruto = $request->PesoBruto;  $cera->PesoNeto = $request->PesoNeto;  $cera->Fecha = $request->Fecha;    $cera->save();  return response()->json(['success' => 'Se ha creado una Recepción de Cera correctamente']);    }  }  public *function* editCera(*request* $request){  $rules = array(  'Descripcion' => 'required',  'Recepcion\_id' => 'required',  'PesoBruto' => 'required',  'PesoNeto' => 'required',  'Fecha' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {    $cera = *Cera*::find($request->id);  $cera->Descripcion = $request->Descripcion;  $cera->Recepcion\_id = $request->Recepcion\_id;  $cera->PesoBruto = $request->PesoBruto;  $cera->PesoNeto = $request->PesoNeto;  $cera->Fecha = $request->Fecha;  $cera->save();  return response()->json($cera);  }  }  public *function* deleteCera(*request* $request){    $cera = *Cera*::find ($request->id);  $cera->delete();  } | |
| sp2017ConsultaEstAprobacion | Obtener los estados de solicitudes |
| USE [PatentesIng2v9]  GO  /\*\*\*\*\*\* Object: StoredProcedure [dbo].[sp2017ConsultaEstAprobacion] Script Date: 06/06/2017 02:40:03 a.m. \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  ALTER procedure [dbo].[sp2017ConsultaEstAprobacion]  as  select t.CP29IEst, t.CP29DesEst from CP29EST t where t.CP29IEst between 2 and 3 | |
| EstanonController | Añadir , editar y eliminar los estañones en el sistema. |
| public *function* addEstanon(*Request* $request){  $rules = array(  'Descripcion' => 'required',  'Peso' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {  $estanon = new *Estanon*;  $estanon->Descripcion = $request->Descripcion;  $estanon->Peso = $request->Peso;  $estanon->save();  return response()->json(['success' => 'Se ha creado un Estañón correctamente']);  }  }  public *function* editEstanon(*request* $request){  $rules = array(  'Descripcion' => 'required',  'Peso' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {  $estanon = new *Estanon*;  $estanon->Descripcion = $request->Descripcion;  $estanon->Peso = $request->Peso;  $estanon->save();  return response()->json($estanon);  }  }  public *function* deleteEstanon(*request* $request){    $estanon = *Estanon*::find ($request->id);  $estanon->delete();    } | |
| RecepcionEstañon | Añadir , editar y eliminar las recepciones de estañón del sistema. |
| public *function* addRecepcion(*Request* $request){  $rules = array(    'Recepcion\_id' => 'required',  'Estanon\_id' => 'required',  'Fecha' => 'required',    );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {    $recepcionEst = new *RecepcionEstanon*;  $recepcionEst->Recepcion\_id = $request->Recepcion\_id;  $recepcionEst->Estanon\_id = $request->Estanon\_id;  $recepcionEst->Fecha = $request->Fecha;    $recepcionEst->save();  return response()->json(['success' => 'Se ha creado una Recepción de Estañón correctamente']);  }  }  public *function* editRecepcion(*request* $request){  $rules = array(    'Recepcion\_id' => 'required',  'Estanon\_id' => 'required',  'Fecha' => 'required',  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));  else {    $recepcionEst = *RecepcionEstanon*::find($request->id);  $recepcionEst->Recepcion\_id = $request->Recepcion\_id;  $recepcionEst->Estanon = $request->Estanon;  $recepcionEst->Fecha = $request->Fecha;  $recepcionEst->save();  return response()->json($recepcionEst);  }  }  public *function* deleteApiario(*request* $request){      $recepcionEst = *RecepcionEstanon*::find($request->id);  $recepcionEst->delete();  return response()->json($recepcionEst);  } | |
| StockController | Añadir , editar y eliminar los stock del sistema. |
| public *function* addStock(*Request* $request){  $rules = array(      'nombre' => 'required',  'cantidadDisponible' => 'required',  'precioUnitario' => 'required',  'estanon\_recepcions\_id' => 'required'  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));    else {  $sto = new *Stock*;  $sto->id= $request->id;  $sto->nombre = $request->nombre;  $sto->cantidadDisponible = $request->cantidadDisponible;  $sto->precioUnitario = $request->precioUnitario;  $sto->estanon\_recepcions\_id = $request->estanon\_recepcions\_id;  $sto->save();  return response()->json(['success' => 'Se ha creado un Stock correctamente']);  }  }  public *function* editStock(*request* $request){  $rules = array(  );  $validator = *Validator*::make ( *Input*::all(), $rules);  if ($validator->fails())  return *Response*::json(array('errors'=> $validator->getMessageBag()->toarray()));    else {  $sto = *Stock*::find ($request->id);    $sto->nombre = $request->nombre;  $sto->cantidadDisponible = $request->cantidadDisponible;  $sto->precioUnitario = $request->precioUnitario;  $sto->estanon\_recepcions\_id = $request->estanon\_recepcions\_id;    $sto->save();  return response()->json($sto);  }  }  public *function* deleteStock(*request* $request){    $sto = *Stock*::find ($request->id);  $sto->nombre = $request->nombre;  $sto->cantidadDisponible = $request->cantidadDisponible;  $sto->precioUnitario = $request->precioUnitario;  $sto->estanon\_recepcions\_id = $request->estanon\_recepcions\_id;  $sto->delete();  return response()->json();  } | |
| IngresoCeraController | Crear los servicios de cera ingresados dependiendo de la recepción de cera del afiliado en el sistema. |
| public *function* create()  {  $personas=*DB*::table('afiliados')  ->get();  $usuarios=*DB*::table('users')  ->get();  $ceras = *DB*::table('ceras as art')  ->select(*DB*::raw('CONCAT(art.id ," - ",art.Recepcion\_id ) AS ceras'),'art.id','art.PesoNeto')  ->groupBy('ceras','art.id','art.PesoNeto')  ->where('art.PesoNeto','>','0')  ->get();  return view ("IngresoCera.create",["personas"=>$personas,"usuarios"=>$usuarios,"ceras"=>$ceras]);  }  public *function* store (*IngresoCeraFormRequest* $request)  {  try{  *DB*::beginTransaction();  $ingreso=new *IngresoCera*;  $ingreso->idproveedor=$request->get('idproveedor');  $ingreso->idusuario=$request->get('idusuario');  $ingreso->tipo\_comprobante=$request->get('tipo\_comprobante');  $ingreso->serie\_comprobante=$request->get('serie\_comprobante');  $ingreso->tipo\_pago=$request->get('tipo\_pago');  $ingreso->total\_venta=$request->get('total\_venta');  $mtyime= *Carbon*::now('America/Costa\_Rica');  $ingreso->fecha\_hora=$mtyime->toDateTimeString();  $ingreso->estado='Activo';  $ingreso->save();  $cera\_id = $request->get('cera\_id');  $Precio= $request->get('Precio');    $cont= 0;  while ($cont < count($cera\_id)){  $detalle = new *DetalleIngresoCera*();  $detalle->idingreso\_cera=$ingreso->idingreso\_cera;  $detalle->cera\_id=$cera\_id[$cont];  $detalle->Precio=$Precio[$cont];  $detalle->save();  $cont=$cont+1;    }  *DB*::commit();  }catch(\*Exception* $e)  {  *DB*::rollback();  }  return *Redirect*::to('IngresoCera');  }  public *function* show ($id)  {  $ingresos=*DB*::table('ingreso\_cera as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso\_cera as di','i.idingreso\_cera','=','di.idingreso\_cera')  ->select('i.idingreso\_cera','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->where('i.idingreso\_cera','=',$id)  ->groupBy('i.idingreso\_cera','i.fecha\_hora','p.nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->first();  $detalles=*DB*::table('detalle\_ingreso\_cera as d')  ->join ('ceras as a','d.cera\_id','=','a.id')  ->select('a.Descripcion as ceras','d.Precio','a.PesoNeto')  ->where('d.idingreso\_cera','=',$id)  ->get();  return view("IngresoCera.show",["ingresos"=>$ingresos,"detalles"=>$detalles]);  }  public *function* edit($id)  {  $ingresos=*DB*::table('ingreso\_cera as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso\_cera as di','i.idingreso\_cera','=','di.idingreso\_cera')  ->select('i.idingreso\_cera','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->where('i.idingreso\_cera','=',$id)  ->groupBy('i.idingreso\_cera','i.fecha\_hora','p.nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->first();  $detalles=*DB*::table('detalle\_ingreso\_cera as d')  ->join ('ceras as a','d.cera\_id','=','a.id')  ->select('a.Descripcion as ceras','d.Precio','a.PesoNeto')  ->where('d.idingreso\_cera','=',$id)  ->get();    $ingresos = *PDF*::loadView("IngresoCera.edit",["ingresos"=>$ingresos,"detalles"=>$detalles]);  return $ingresos->download('IngresoCera.edit');    }  public *function* destroy ($id)  {  $ingresoC=*IngresoCera*::findOrFail($id);  $ingresoC->Estado='Cancelado';  $ingresoC->update();  return *Redirect*::to('IngresoCera');  } | |
| IngresoController | Crear, editar y eliminar los servicios de miel procesada dependiendo de la recepción de miel del afiliado en el sistema. |
| public *function* store (*IngresoFormRequest* $request)  {  try{  *DB*::beginTransaction();  $ingreso=new *Ingreso*;  $ingreso->idproveedor=$request->get('idproveedor');  $ingreso->idusuario=$request->get('idusuario');  $ingreso->tipo\_comprobante=$request->get('tipo\_comprobante');  $ingreso->serie\_comprobante=$request->get('serie\_comprobante');  $ingreso->total\_venta=$request->get('total\_venta');  $mtyime= *Carbon*::now('America/Costa\_Rica');  $ingreso->fecha\_hora=$mtyime->toDateTimeString();  $ingreso->estado='Activo';  $ingreso->save();  $recepcion\_id= $request->get('recepcion\_id');  $Precio = $request->get('Precio');    $cont= 0;  while ($cont < count($recepcion\_id)){  $detalle = new *DetalleIngreso*();  $detalle->idingreso=$ingreso->idingreso;  $detalle->recepcion\_id=$recepcion\_id[$cont];  $detalle->Precio=$Precio[$cont];    $detalle->save();  $cont=$cont+1;    }  *DB*::commit();  }catch(\*Exception* $e)  {  *DB*::rollback();  }  return *Redirect*::to('Ingreso');  }  public *function* show ($id)  {  $ingresos=*DB*::table('ingreso as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso as di','i.idingreso','=','di.idingreso')  ->select('i.idingreso','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.total\_venta','i.estado')  ->where('i.idingreso','=',$id)  ->groupBy('i.idingreso','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.total\_venta','i.estado')  ->first();  $detalles=*DB*::table('detalle\_ingreso as d')  ->join ('recepcion\_materia\_primas as a','d.recepcion\_id','=','a.id')  ->select('a.id as producto','d.Precio','a.PesoBruto')  ->where('d.idingreso','=',$id)  ->get();  return view("Ingreso.show",["ingresos"=>$ingresos,"detalles"=>$detalles]);  }  public *function* edit($id)  {  $ingresos=*DB*::table('ingreso as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso as di','i.idingreso','=','di.idingreso')  ->select('i.idingreso','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.total\_venta','i.estado')  ->where('i.idingreso','=',$id)  ->groupBy('i.idingreso','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.total\_venta','i.estado')  ->first();  $detalles=*DB*::table('detalle\_ingreso as d')  ->join ('recepcion\_materia\_primas as a','d.recepcion\_id','=','a.id')  ->select('a.id as producto','d.Precio','a.PesoBruto')  ->where('d.idingreso','=',$id)  ->get();    $ingresos = *PDF*::loadView("Ingreso.edit",["ingresos"=>$ingresos,"detalles"=>$detalles]);  return $ingresos->download('Ingreso.edit');    }  public *function* destroy ($id)  {  $ingreso=*Ingreso*::findOrFail($id);  $ingreso->Estado='Cancelado';  $ingreso->update();  return *Redirect*::to('Ingreso');  } | |
| IngresoInventarioController | Crear , editar y eliminar servicios de inventario dependiendo de los stock en el sistema. |
| public *function* create()  {  $personas=*DB*::table('afiliados')  ->get();  $usuarios=*DB*::table('users')  ->get();  $stocks = *DB*::table('stocks as art')  ->select(*DB*::raw('CONCAT(art.id," - ", art.nombre) AS stocks'),'art.id','art.cantidadDisponible')  ->groupBy('stocks','art.id','art.cantidadDisponible')  ->get();  return view ("IngresoInventario.create",["personas"=>$personas,"usuarios"=>$usuarios,"stocks"=>$stocks]);  }  public *function* store (*IngresoInventarioFormRequest* $request)  {  try{  *DB*::beginTransaction();  $ingreso=new *IngresoInventario*;  $ingreso->idproveedor=$request->get('idproveedor');  $ingreso->idusuario=$request->get('idusuario');  $ingreso->tipo\_comprobante=$request->get('tipo\_comprobante');  $ingreso->serie\_comprobante=$request->get('serie\_comprobante');  $ingreso->tipo\_pago=$request->get('tipo\_pago');  $ingreso->total\_venta=$request->get('total\_venta');  $mtyime= *Carbon*::now('America/Costa\_Rica');  $ingreso->fecha\_hora=$mtyime->toDateTimeString();  $ingreso->estado='Activo';  $ingreso->save();  $stock\_id = $request->get('stock\_id');  $Precio= $request->get('Precio');  $cantidad=$request->get('cantidad');      $cont= 0;  while ($cont < count($stock\_id)){  $detalle = new *DetalleIngresoInventario*();  $detalle->idingreso\_inventario=$ingreso->idingreso\_inventario;  $detalle->stock\_id=$stock\_id[$cont];  $detalle->Precio=$Precio[$cont];  $detalle->cantidad=$cantidad[$cont];    $detalle->save();  $cont=$cont+1;    }  *DB*::commit();  }catch(\*Exception* $e)  {  *DB*::rollback();  }  return *Redirect*::to('IngresoInventario');  }  public *function* show ($id)  {  $ingresos=*DB*::table('ingreso\_inventario as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso\_inventario as di','i.idingreso\_inventario','=','di.idingreso\_inventario')  ->select('i.idingreso\_inventario','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->where('i.idingreso\_inventario','=',$id)  ->groupBy('i.idingreso\_inventario','i.fecha\_hora','p.nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->first();  $detalles=*DB*::table('detalle\_ingreso\_inventario as d')  ->join ('stocks as a','d.stock\_id','=','a.id')  ->select('a.producto\_id as stocks','d.Precio','d.cantidad','a.cantidadDisponible')  ->where('d.idingreso\_inventario','=',$id)  ->get();  return view("IngresoInventario.show",["ingresos"=>$ingresos,"detalles"=>$detalles]);  }  public *function* edit($id)  {  $ingresos=*DB*::table('ingreso\_inventario as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso\_inventario as di','i.idingreso\_inventario','=','di.idingreso\_inventario')  ->select('i.idingreso\_inventario','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->where('i.idingreso\_inventario','=',$id)  ->groupBy('i.idingreso\_inventario','i.fecha\_hora','p.nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->first();  $detalles=*DB*::table('detalle\_ingreso\_inventario as d')  ->join ('stocks as a','d.stock\_id','=','a.id')  ->select('a.producto\_id as stocks','d.Precio','d.cantidad','a.cantidadDisponible')  ->where('d.idingreso\_inventario','=',$id)  ->get();    $ingresos = *PDF*::loadView("IngresoInventario.edit",["ingresos"=>$ingresos,"detalles"=>$detalles]);  return $ingresos->download('IngresoInventario.edit');    }  public *function* destroy ($id)  {  $ingresoC=*IngresoInventario*::findOrFail($id);  $ingresoC->Estado='Cancelado';  $ingresoC->update();  return *Redirect*::to('IngresoInventario');  } | |
| UserController | Listar todos los usuarios o un Usuario registrados en el sistema. |
| public *function* index(*Request* $request)  {    if($request){  $query=trim($request->get('searchText')); //valida si la peticion trae el campo de busqueda  $users= *User*::with('roles')->where('name','LIKE','%'.$query.'%')  ->orderby('id','desc')  ->paginate(4);      $roles = *Role*::all();  return view('users.index', compact('users', 'roles'), ['users'=>$users,"searchText"=>$query]);  }  } | |
| AfiliadoController | Listar todos los Afiliados o un afiliado registrados en el sistema. |
| public *function* index(*Request* $request)  {  if($request){  $query=trim($request->get('searchText')); //valida si la peticion trae el campo de busqueda  $afi= *Afiliado*::with('Genero', 'Estado\_Civil')  ->where('Nombre','LIKE','%'.$query.'%')  ->orderby('id','desc')  ->paginate(7);  $genero = *Genero*::all();  $estadoC = *Estado\_Civil*::all();  return view('Afiliado.index', compact('afi','genero','estadoC'), ['afi'=>$afi,"searchText"=>$query]);  }    return view('Afiliado.index',compact('afi','genero','estadoC','esta'));    } | |
| UbicacionController | Listar todas o una sola ubicación registrada en el sistema. |
| ublic *function* index(*Request* $request)  {  if ($request)  {  $query=trim($request->get('searchText'));  $ubicacion=*DB*::table('ubicacions')->where('id','LIKE','%'.$query.'%')  ->orwhere('descripcion','LIKE','%'.$query.'%')  ->orderby('id','desc')  ->paginate(10);  return view('Ubicacion.index',["ubicacion"=>$ubicacion,"searchText"=>$query]);  }  // $ubicacion = Ubicacion::paginate(10);  // return view('Ubicacion.index',compact('ubicacion'));    } | |
| ApiarioController | Listar todos o un solo Apiario registrado en el sistema. |
| public *function* index(*Request* $request)  {  if($request){  $query=trim($request->get('searchText')); //valida si la peticion trae el campo de busqueda  $api = *Apiario*::with('Ubicacion')  ->where('Descripcion','LIKE','%'.$query.'%')  ->orderby('id','desc')  ->paginate(7);  $ubicaciones = *Ubicacion*::all();    return view('Apiario.index', compact('api', 'ubicaciones'), ['api'=>$api,"searchText"=>$query]);  }    } | |
| AfiliadoApiarioController | Listar todos o un solo Afiliado con su Apiario en el sistema. |
| public *function* index(*Request* $request)  {    $afiliadoapiario = *AfiliadoApiario*::paginate(10);  $afiliados=*Afiliado*::all();  $apiarios=*Apiario*::all();  return view('AfiliadoApiario.index',compact('afiliadoapiario','afiliados','apiarios'));    } | |
| RecepcionMateriaPrima | Listar todas o una sola recepción de miel en el sistema. |
| public *function* index(*Request* $request){  if($request){  $query=trim($request->get('searchText')); //valida si la peticion trae el campo de busqueda  $recepcion = *RecepcionMateriaPrima*::paginate(10);  $afiliado = *Afiliado*::all();  $estanon = *Estanon*::all();  $recepciones = *RecepcionMateriaPrima*::all()->sortBy("fecha");  $user = *User*::all();  $tipoEntrega = *TipoEntrega*::all();  return view('RecepcionMateriaPrima.index', compact('afiliado','recepciones','estanon','user','tipoEntrega','recepcion'), ['recepcion'=>$recepcion,"searchText"=>$query]);  }} | |
| CeraController | Listar todas o una recepción de cera en el sistema. |
| public *function* index(*Request* $request)  {  if($request){  $query=trim($request->get('searchText')); //valida si la petición trae el campo de busqueda  $cera = *Cera*::with('RecepcionMateriaPrima')  ->where('Descripcion','LIKE','%'.$query.'%')  ->orderby('id','desc')  ->paginate(7);  $recepciones = *RecepcionMateriaPrima*::all();    return view('Cera.index', compact('cera', 'recepciones'), ['cera'=>$cera,"searchText"=>$query]);  }    } | |
| EstanonController | Lista todos o un solo estañón en el sistema. |
| public *function* index(*Request* $request)  {  if($request){  $query=trim($request->get('searchText')); //valida si la petición trae el campo de búsqueda  $estanon = *Estanon*::paginate(10);  return view('Estanon.index',compact('estanon'), ['estanon'=>$estanon,"searchText"=>$query]);  }  } | |
| RecepcionEstanonController | Listar todas o una sola recepción de estañón en el sistema. |
| public *function* index(*Request* $request)  {  if($request){  $query=trim($request->get('searchText')); //valida si la petición trae el campo de búsqueda  $recepcionEst = *RecepcionEstanon*::with('RecepcionMateriaPrima', 'Estanon')  ->where('Fecha','LIKE','%'.$query.'%')  ->orderby('id','desc')  ->paginate(7);  $recepciones = *RecepcionMateriaPrima*::all();  $estanon = *Estanon*::all();    return view('RecepEstanon.index', compact('cera', 'recepciones', 'estanon'), ['recepcionEst'=>$recepcionEst,"searchText"=>$query]);  }  } | |
| IngresoCeraController | Listar todos o solo un servicio de cera en el sistema. |
| public *function* index(*Request* $request)  {  if ($request)  {  $query=trim($request->get('searchText'));  $ingresos=*DB*::table('ingreso\_cera as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso\_cera as di','i.idingreso\_cera','=','di.idingreso\_cera')  ->select('i.idingreso\_cera','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->where('i.idingreso\_cera','LIKE','%'.$query. '%')  ->orderBy ('i.idingreso\_cera','desc')  ->groupBy('i.idingreso\_cera','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->paginate(7);  return view ('IngresoCera.index',["ingresos"=>$ingresos,"searchText"=>$query]);  }  } | |
| IngresoinventarioController | Listar todos o un solos servicio en el sistema. |
| public *function* index(*Request* $request)  {  if ($request)  {  $query=trim($request->get('searchText'));  $ingresos=*DB*::table('ingreso\_inventario as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso\_inventario as di','i.idingreso\_inventario','=','di.idingreso\_inventario')  ->select('i.idingreso\_inventario','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->where('i.idingreso\_inventario','LIKE','%'.$query. '%')  ->orderBy ('i.idingreso\_inventario','desc')  ->groupBy('i.idingreso\_inventario','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.tipo\_pago','i.total\_venta','i.estado')  ->paginate(7);  return view ('Ingresoinventario.index',["ingresos"=>$ingresos,"searchText"=>$query]);  }  } | |
| IngresoController | Listar todos o un solo servicio de miel procesada en el sistema. |
| public *function* index(*Request* $request)  {  if ($request)  {  $query=trim($request->get('searchText'));  $ingresos=*DB*::table('ingreso as i')  ->join('afiliados as p','i.idproveedor','=','p.id')  ->join('users as u','i.idusuario','=','u.id')  ->join('detalle\_ingreso as di','i.idingreso','=','di.idingreso')  ->select('i.idingreso','i.fecha\_hora','p.Nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.total\_venta','i.estado')  ->where('i.idingreso','LIKE','%'.$query. '%')  ->orderBy ('i.idingreso','desc')  ->groupBy('i.idingreso','i.fecha\_hora','p.nombre','p.apellido1','p.apellido2','u.name','i.tipo\_comprobante', 'i.serie\_comprobante','i.total\_venta','i.estado')  ->paginate(7);  return view ('Ingreso.index',["ingresos"=>$ingresos,"searchText"=>$query]);  }  } | |

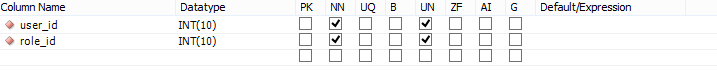
**Tablas.**

**Users**

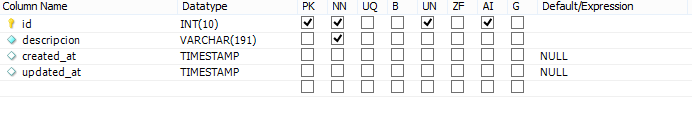


roles

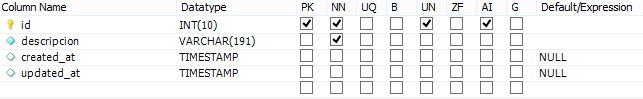


users\_roles

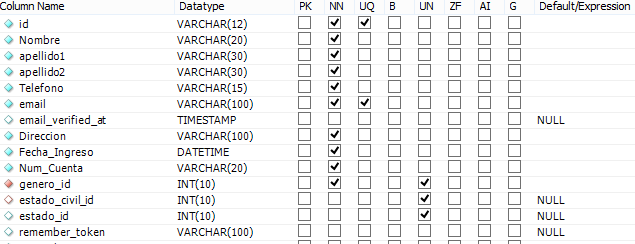
**generos.**



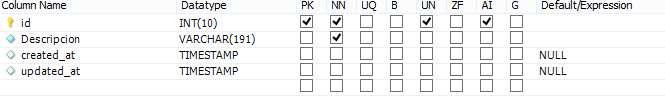
**estado\_civils**



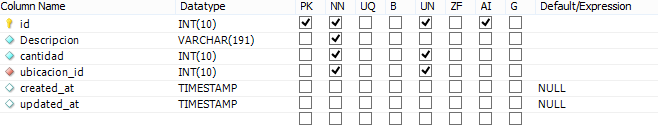
**afiliados**



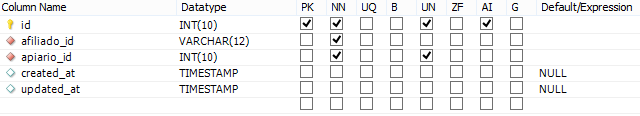
**Ubicacions**



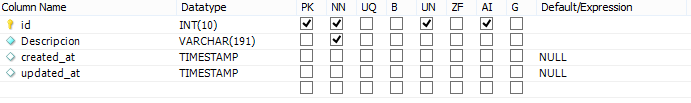
**apiarios**



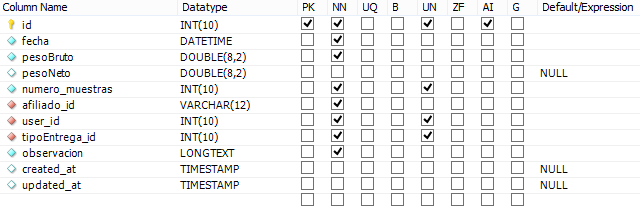
**afiliado\_apiarios**



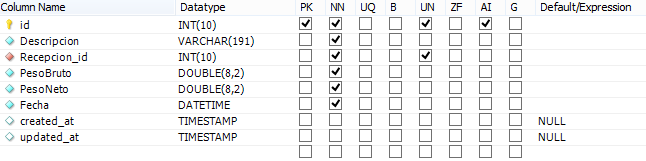
**tipo\_entregas**



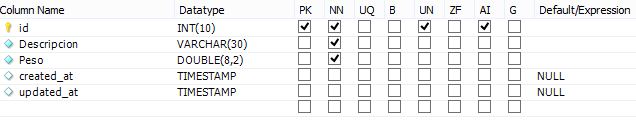
**recepcion\_materia\_primas**



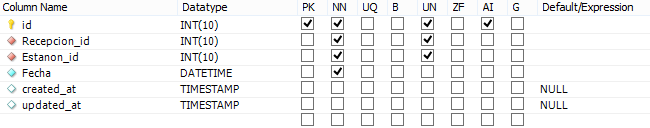
**ceras.**



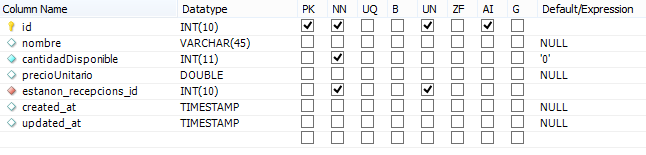
**estanons**



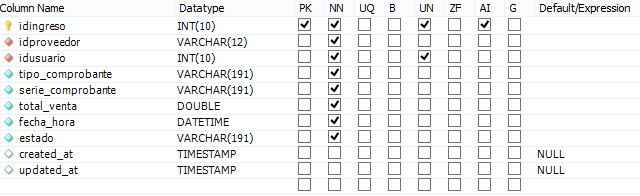
**recepcion\_estanons**



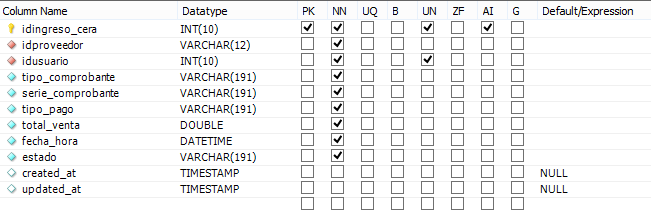
**stocks**



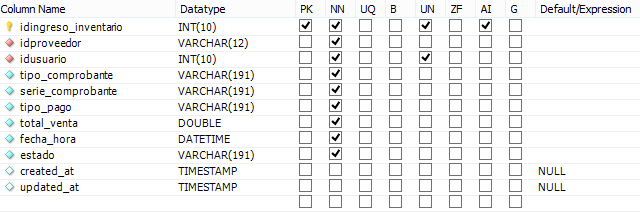
**ingreso.**



**ingreso\_cera**



ingreso\_inventario.



**Instalación y configuración**

Requisitos generales preinstalación