Documentación del Sitio CIMEQH

Contenido

[Componentes del Framework 3](#_Toc488604252)

[ Librerías: 3](#_Toc488604253)

[o Parameters.php: 3](#_Toc488604254)

[o Template\_engine.php: 5](#_Toc488604255)

[ Vistas: 8](#_Toc488604256)

[ Controladores: 11](#_Toc488604257)

[Estructura de un Controlador 11](#_Toc488604258)

[o Site.nw.php: 12](#_Toc488604259)

[o Verificar.nw.php: 13](#_Toc488604260)

[o Index.php: 13](#_Toc488604261)

[ Modelos: 14](#_Toc488604262)

[Validaciones: 14](#_Toc488604263)

[Diseño del sitio: 14](#_Toc488604264)

[Base de Datos 15](#_Toc488604265)

[Tabla de Roles 15](#_Toc488604266)

[Tabla de Estados de Cuenta 15](#_Toc488604267)

[Tablas de Estados de Recepción, Factibilidad, Despeje y Aprobación 16](#_Toc488604268)

El Sistema fue desarrollado sobre un servidor Apache y utilizando MySQL (es el servidor de base de datos relacionales más popular) como gestor de base de datos y PHP.

Se implementó el uso de varios framework que servirán de pilares en el desarrollo del sistema, a continuación, se explican en detalle:

* MVC (Modelo Vista Controlador): Es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Models y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación.
* Bootstrap: Es un framework que tiene como objetivo facilitar el diseño web, permite crear de forma sencillas webs de diseño adaptable es decir que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien.

# Componentes del Framework

A continuación, se enumeran los componentes principales del framework utilizado:

* Librerías: En esta carpeta se encuentran las funciones principales que hacen que la aplicación funcione, a continuación, se explican las principales librerías:
* Parameters.php: Esta librería contiene los datos necesarios para que la aplicación se pueda conectar con la base de datos.

A continuación, se detallas los elementos de este archivo:

* **$server =** “Aquí se especifica la dirección del servidor donde se aloja la base de datos”
* **$user =** “El nombre de usuario con el que se accede a esa base de datos”
* **$pswd =** “Aquí se especifica la contraseña”
* **$database =** “El nombre de la base de datos a utilizar”
* **$port =** “El número de puerto del servidor”
* Dao.php: Esta librería se encuentran todas las funciones que permiten interactuar con la base de datos, ya sea para guardar los datos en ella o para obtener los datos solicitados.

A continuación, se detallan los elementos y funciones más importantes:

* **Require\_once(“libs/parameters.php”):** Aquí se importan los parámetros de conexión a la base de datos.
* **$conexion = new mysqli( $server, $user, $pswd, $database, $port ):** Con esta línea de código se crea la conexión a la base de datos.
* **obtenerRegistros(“Aquí se le pasa el query a ejecutar”):** Esta función se utiliza cuando se requiere ejecutar un query que se espera que retorne varias filas o registros. Esta función retorna un arreglo con los datos obtenidos al ejecutar el query.
* **obtenerUnRegistro(“Aquí se le pasa el query a ejecutar”):** Esta función se utiliza cuando se requiere ejecutar un query que se espera que retorne un solo registro o una sola fila. Esta función retorna un arreglo con los datos obtenidos al ejecutar el query.
* **ejecutarNonQuery(“Aquí se le pasa el query a ejecutar”):** Esta función se utiliza cuando se requiere ejecutar un query de tipo **INSERT, UPDATE** o **DELETE**. Esta función retorna la cantidad de filas afectadas al ejecutar el query.
* **ejecutarNonQueryConErrores(“Aquí se le pasa el query a ejecutar”):** Esta función se utiliza cuando se requiere ejecutar un query de tipo **INSERT, UPDATE** o **DELETE**. Esta función retorna el mensaje de error de MySql en caso de que algo fallara, esta función es ideal para encontrar errores y depurar código.
* **getLastInsetId(“No requiere parámetros”):** Esta función retorna la última llave primaria insertada de una conexión activa.

La librería de dao.php utiliza una sola conexión a la base de datos y se encarga de cerrar la misma automáticamente para la ejecución de todas las funciones lo que prevé que se rebalse la piscina de conexiones .

* Template\_engine.php: Renderiza todas las vistas que se soliciten (se detallara más adelante que es una vista) así como los datos obtenidos de la base de datos a través de un controlador.

A continuación, se detallan los elementos y funciones más importantes:

* **renderizar($vista, $datos, $layoutFile = "layout.view.tpl"):** Esta función se ejecuta en el controlador de cada vista y se encarga de renderizar las vistas en el explorador. Los parámetros que utiliza esta función se detallan a continuación:
  + **$vista =** “Este parámetro es el nombre de la view que se desea renderizar ejemplo: [home.view.tpl](http://home.view.tpl)”
  + **$datos=** ”Aquí se pasan los datos obtenidos en el controlador y que se desean utilizar o mostrar en la vista, este parámetro debe ser un arreglo, en caso de pasar ningún dato se deberá pasar como un arreglo vacío ejemplo: **array()**”
  + **$layout=**”Aquí se especifica que layout utilizará la vista que se renderizará (más adelante se daráan más detalles acerca de como funciona un layout) este parámetro se puede omitir, en caso de omitirlo la vista tomará por defecto un layout predeterminado que en este caso es layout.view.tpl”

En el template.engine también ofrece funciones para poder manipular los datos extraídos del controlador dentro de las vistas las funciones que ofrece este platillero son las siguientes:

* If
* For
* Ifnot

A continuación, se explica el funcionamiento de las funciones del plantillero en las vistas.

**{{}}**

Si se requiere mostrar un dato único en las vistas se coloca le nombre del dato que se pasó en la función renderizar entre corchetes. Ejemplo:

|  |
| --- |
| //En el controlador  renderizar(“holaMundo.view.tpl”, array(“mensaje”=>”Hola Mundo”))  //En la vista  <h1>{{mensaje}}</h1>  //Lo que muestra en pantalla  **Hola Mundo** |

{{foreach}}

{{endfor}}

Si se desea iterar un dato dentro de una lista se utiliza la función foreach seguida del nombre del dato a iterar y al terminar se escribe endfor seguido nuevamente del nombre del dato, todo dentro de corchetes. Ejemplo:

|  |
| --- |
| //En el controlador  $diasDeLaSemana=array(“dia”=>”Lunes”,“dia”=>”Martes”, “dia”=>”Miercoles”, “dia”=>”Jueves”, “dia”=>”Viernes”);  renderizar(“holaMundo.view.tpl”, array(“mensaje”=>”Días de La Semana”, “diasSemana”=>$diasDeLaSemana))  //En la vista  <h1>{{mensaje}}</h1>  <ul>  {{ foreach diasSemana }}  <li>{{ dia }}</li>  {{ endfor diasSemana }}  </ul>  //Lo que muestra en pantalla  **Días de la Semana**   * Lunes * Martes * Miercoles * Jueves * Viernes |

**{{if}}**

**{{endif}}**

**{{ifnot}}**

**{{endifnot}}**

Si se desea realizar una decisión dentro de una vista se utiliza la función if o ifnot dentro de corchetes seguido del nombre del dato, al final de la decisión se debe ingresar un endif o un endifnot respectivamente, seguido nuevamente por el mismo dato.

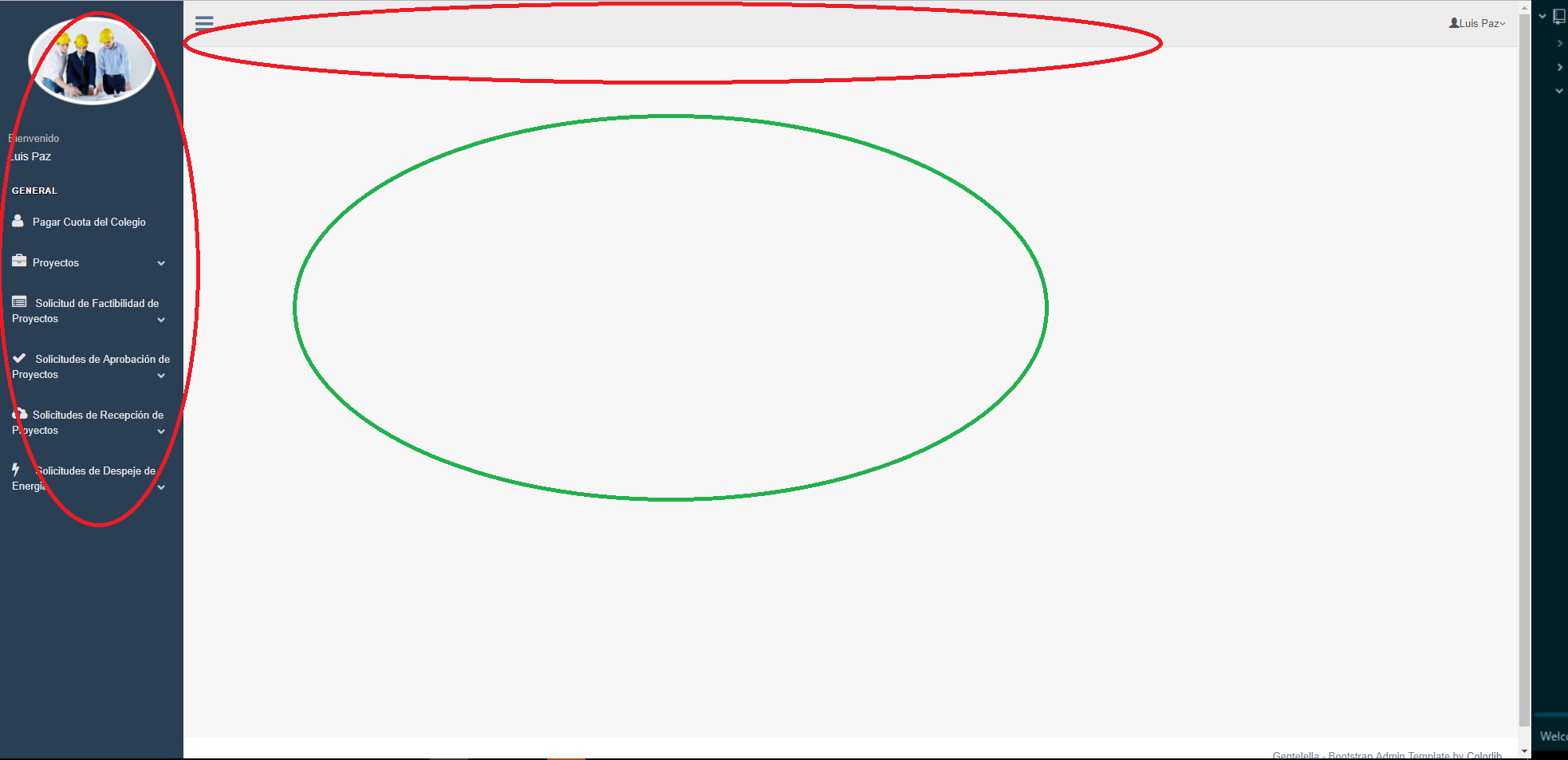
* **Nota.** Estas funciones solo funcionan para evaluar datos de tipo booleano.

|  |
| --- |
| //En el controlador  $edad= 21  $esMayorDeEdad=false  If( $edad >= 21){  $esMayorDeEdad= true  }  renderizar(“holaMundo.view.tpl”,array(“esMayorDeEdad”=> sMayorDeEdad))  //En la vista  {{ if esMayorDeEdad }}  <h1>Puede Entrar a la Disco</h1>  {{ endif esMayorDeEdad }}  //Lo que muestra en pantalla  **Puede Entrar a la Disco** |

* Vistas: Es la primera capa de la aplicación y es llamada la capa cliente, esta se encarga de comunicar al usuario con el Sistema, y en ella se registran las instruciones del usuario retornando las respuestas a la aplicación.

Una de las vistas de mayor relevancia es **“layout.view.tpl”** debido a que contiene el diseño general de todas las vistas así como los elementos básicos con los que contaran todas las demás vistas, por ejemplo:

* Menú
* Footer



Este es un ejemplo del layout predeterminado utilizado en el sitio, las partes dentro de los círculos rojos son las partes propias del layout, son elementos que se repiten en la mayoría de las vistas, dentro del círculo verde es donde se renderizan las vistas individuales.

Dentro del layout se deberán incluir las librerías y scripts que sean necesarias para el funcionamiento del sitio, de este modo se evita estar haciendo referencia a las librerías en cada vista que se cree.

A continuación, se muestra el código de un layout simple para explicar sus elementos.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | <!DOCTYPE html> | |  | <html> | |  | <head> | |  | <meta charset="utf-8" /> | |  | <title>{{page\_title}}</title> | |  | <meta name="viewport" content="width=device-width, initial-scale=1"/> | |  | <link href='https://fonts.googleapis.com/css?family=Roboto:400,700' rel='stylesheet' type='text/css'> | |  | <link rel="stylesheet" href="public/css/estilo1.css" /> | |  | |  | |  | |  | </head> | |  | <body> | |  | <div class="menu"> | |  | <ul> | |  | <li><a href="index.php?page=home">Home</a></li> | |  | <li><a href="index.php?page=althome">Alt Home</a></li> | |  | <li class="title"><b>{{page\_title}}</b></li> | |  | <li><a href>Menu Item 4</a></li> | |  | <li><a href>Menu Item 5</a></li> | |  | </ul> | |  | </div> | |  | <div class="contenido"> | |  | {{{page\_content}}} | |  | </div> | |  | <div class="footer"> | |  | Derechos Reservados 2015 | |  | </div> | |  | </body> | |  | </html> | |

* **{{ page\_title }}** = En este elemento se renderiza el title del sitio web, más adelante se explicará cómo definir este parámetro.
* **{{ page\_content }} =** Dentro de este bloque es donde se renderiza todas las vistas que se manden a llama

En el sistema de CIMEQH se utilizarón ocho layouts que se visualizan según el usuario que está usando el sistema los cuales son:

1. **layoutPublico.view.tpl:** Es el layout que visualizan los usuarios que no están registrados pero quiere consultar el estado de un proyecto.
2. **layoutCimeqh.view.tpl:** Es el layout que visualiza el usuario que tiene el rol de CIMEQH.
3. **layoutEnee.view.tpl:** Es el layout que visualiza el usuario que tiene el rol de ENEE.
4. **layoutSinSesion.view.tpl:** Es el layout que utiliza el formulario del login.
5. **layoutSinSesion2.view.tpl:** Es el layout que utiliza el formulario de registro de usuarios.
6. **layoutStripe.view.tpl:** Es el layout que utilizan todos los formularios en las que se puede realizar un pago.
7. **layoutLanding.view.tpl:** Es el layout que utiliza el landing page del sitio.
8. **Layout.view.tpl:** Es el layout que visualizan los ingenieros que realizan los trámites con sus proyectos.

* Controladores: Es la capa intermedia de la aplicación y se encarga de ser un Puente entre las vistas y los modelos, los controladores reciben la información de las vistas y estos a su vez se la envían a los modelos y viceversa.

### Estructura de un Controlador

**<?php**

**//Siempre se debe a esta librerí, contiene la arquitectura del sitio**

**require\_once( "libs/template\_engine.php" );**

**//Si se require se hace un llamado al modelo donde contiene las funciones para manipular los datos**

**require\_once( "models/categorias.model.php" );**

**//Dentro de esta función se escriben todos los procedimientos que se requieren realizer.**

**function run(){**

**$categorias = array();**

**$categorias = obtenerCategorias();**

**//Con esta función se renderiza el sitio.**

**renderizar("categorias", array("categorias" => $categorias));**

**}**

**//Aquí es donde se ejecuta la función que se declaró arriba**

**run();**

**?>**

Existe un controlador por cada vista, pero hay 3 controladores en especial que se encargan del funcionamiento de la aplicación:

* Site.nw.php: Es el middleware de configuración de todo el sitio, se encarga de inicializar la aplicación web, en él se pueden renderizar datos que se utilizaran en toda la página web. Por ejemplo:

|  |
| --- |
| **<?php**  **function site\_init(){**  **//Aquí se inicializa la variable title para que esta pueda ser utilizada en todo el sitio.**  **addToContext( "title","CIMEQH" );**  **}**  **site\_init();**  **?>**  **//Vista**  **<h1> {{ title }} </h1>**  **//Lo que se muestra**  **CIMEQH** |

* Verificar.nw.php: Es el middleware de verificación de todo el sitio, este se encarga de verificar el inicio de sesión de los usuarios. A continuación se explican las funciones que contienen este controlador:
  + **mw\_estaLogueado():** No recibe ningún parámetro, retorna verdadero si el usuario tiene una sesión activa y falso si el usuario no ha iniciado sesión.
  + **mw\_estaLogueado( ):** Sirve para inicializar la información de sesión del usuario, los parámetros que recibe son de acuerdo a las necesidades del programador de lo que ocupe guardar.
  + **mw\_redirectToLogin( $to ):** Sirve para redigirir al usuario a la vista de login y el único parámetro que recibe es la dirección del login hacia donde quiere que se redirija al usuario.
* Index.php: Es uno de los elementos más importantes de la aplicación, se de enrutar las url del sito y conectar la url de la página solicitada con su respectivo controlador y vista. A continuación se detallan los elementos más importantes de este controlador:
  + **$pageRequest = “** **landingPage** **”:** En esta variable se especifica cuál es la página de inicio que tendrá el sitio.
  + **switch( $pageRequest ):** Este switch es la función de enrutamiento del sitio, a continuación se explica su funcionamiento:

Por cada vista del sitio se deberá crear un case con el nombre de la vista y la dirección de su respectivo controlador, por ejemplo:

Si de en el browser se intenta ingresar a una vista a través de una url como: [www.ejemplo.com/home](http://www.ejemplo.com/home). Se deberá crear la ruta de esa dirección de la siguiente manera:

**case "home":**

//llamar al controlador

**require\_once("**controllers/home.control.php**");**

**break;**

* Modelos: Se encarga de comunicar a la aplicación con la base de datos, los modelos reciben la información de los controladores para posteriormente enviar los datos a la base de datos y esta retorne los resultados.

Validaciones: Para las validaciones se utilizó un modelo de validación en 2 capas, donde se validan los datos en la capa cliente, la capa del servidor y la base de datos, esto con el fin de tener la mayor protección posible sobre la integridad de los datos.

* Validación de la capa Cliente: Para esta validación se utilizó un plugin llamado Bootstrap validator el cual está basado en JQuery el cual permite la validación de los campos de un formulario mostrándose tanto en tiempo real también cuando el usuario trata de enviar la información.

Puede encontrar toda la documentación de este plugin en el siguiente enlace:

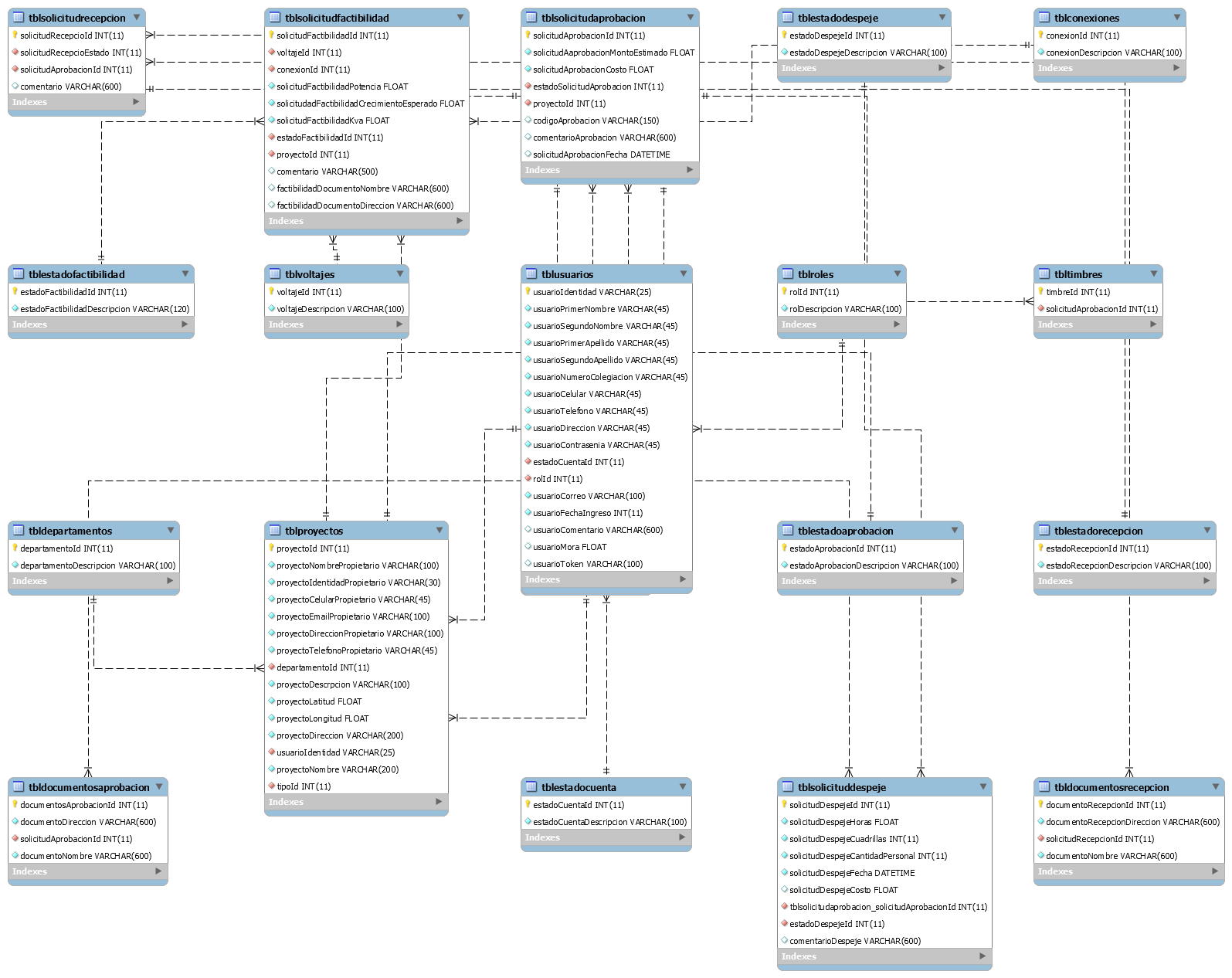
* <http://1000hz.github.io/bootstrap-validator/>

Diseño del sitio:Para el diseño del sitio se utilizó una plantilla gratuita de bootstrap llamada Getelella. Puede encontrar toda la documentación de esta plantilla en el siguiente enlace:

* https://github.com/puikinsh/gentelella

# Base de Datos

A continuación se muestra el diagrama entidad-relación de la base de datos.



Hay algunas tablas que tienen valores por defecto y en los cuales se utiliza estos datos por defectos para realizar procedimientos con ellos. Las tablas con valores por defecto son los siguientes:

## Tabla de Roles

|  |  |
| --- | --- |
| rolId | rolDescripcion |
| 1 | cimeqh |
| 2 | enee |
| 3 | publico |
| 4 | Ingeniero |

## Tabla de Estados de Cuenta

|  |  |
| --- | --- |
| id | descripcion |
| 1 | aprobado |
| 2 | denegado |
| 3 | suspendido |
| 4 | En Revision |

## Tablas de Estados de Recepción, Factibilidad, Despeje y Aprobación

|  |  |
| --- | --- |
| id | descripcion |
| 1 | Aprobado CIMEQH |
| 2 | Aprobado ENEE |
| 3 | Rechazado |
| 4 | En Revision |