\*Sistema Desarrollo11\*

El presente documento expone el manual técnico del \*sistema\* con la finalidad de contar con una explicación clara y precisa del código.

A continuación, se enlistarán las tecnologías ocupadas, así como el patrón de programación utilizado en el \*sistema\*

Tecnologías

* PHP 7.1.20
* MySQL Community Server – GPL
* MaterializeCSS 1.0.0-beta
* XAMPP
* HTML 5
* CSS 3
* JavaScript
* jQuery 3.3.1
* Ajax
* SweetAlert

Patrón

* MVC
* Orientado a objetos.

Es importante remarcar que no se usó ningún framework para el desarrollo de este proyecto, por lo que es necesario que quien lo manipule cuente con los conocimientos adecuados de PHP nativo, así como JavaScript.

**Estructura del proyecto**

Debido al uso del patrón MVC el sistema cuenta con tres carpetas principales, así como carpetas de recursos, en la siguiente figura se muestra el árbol principal del proyecto.

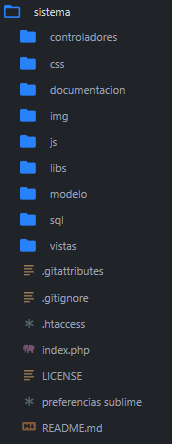


Figura 1 Estructura del proyecto.

Sistema: es la carpeta raíz del proyecto.

Controladores (Figura 2) contiene la lógica.

CSS (Figura 3) Hojas de estilo.

Documentacion (Figura 4) Manuales.

Img (Figura 5) imágenes del proyecto.

Js (Figura 6) Scripts de Javascript.

Libs (Figura 7) librerías nativas del proyecto.

Modelo (Figura 8) contiene las consultas.

Sql (Figura 9) script y modelo de la BD.

Vistas (Figura 10) HTML para la interfaz.

Clases de github.

.htaccess (Figura 11) especificaciones.

Index.php (Figura 12) HTML inicial.

Archivos pertenecientes a materialize.

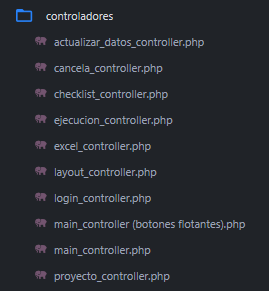
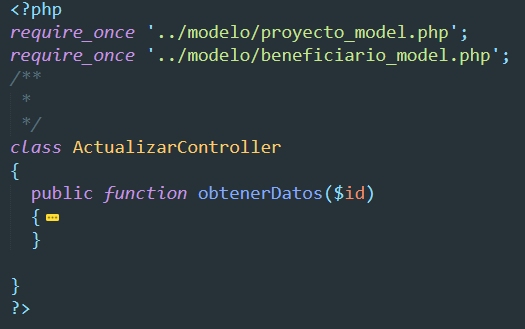


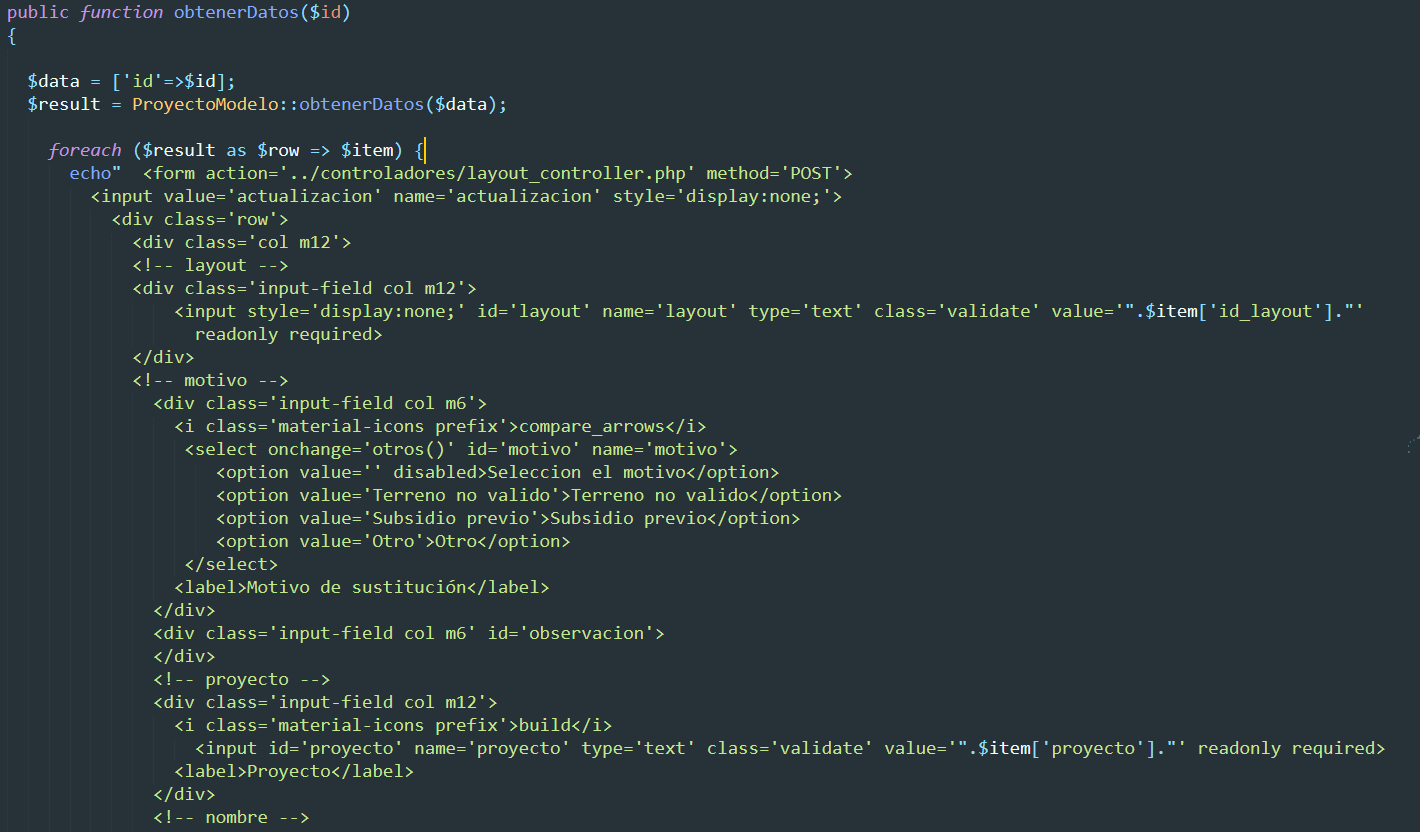
Figura 2 Controladores

Cada controlador se liga una vista, con la finalidad de mantener un orden por lo que en esta sección se explicarán paso a paso cada controlador y sus funciones, empezando por orden alfabético:

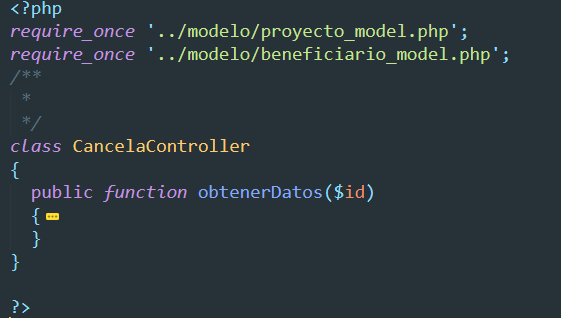
**Actualizar\_datos\_controller.php**:



Este archivo únicamente cuenta con una clase llamada ActualizarController, la cual contiene una sola función “ObtenerDatos” que recibe un id el cual pertenece al layout del usuario seleccionado. A grandes rasgos la función busca la información existente el usuario y la convierte en un formulario de actualización de la siguiente forma:



El id del layout se almacena en el arreglo ‘$data’ con la clave ‘id’, dicho arreglo es enviado a la función “obtenerDatos” de la clase “ProyectoModelo”, los resultados obtenidos en esta clase se guardan en la variable “$result” para después ser recorrida por medio de un foreach debido a que al ser un arreglo de tipo “FetchAll” la única forma para manipular esta información es por medio del foreach, en cada iteración se genera un “echo” en el cual se agregan las etiquetas HTML necesarias para generar el formulario de actualización de las datos, agregando los valores obtenidos de la base de datos en el atributo “value” de cada input con la indicación “$item[‘nombre\_de\_la\_columna’]” y agregando el atributo “readonly” para evitar que el usuario manipule campos específicos que no pueden ser modificaos después de capturados.

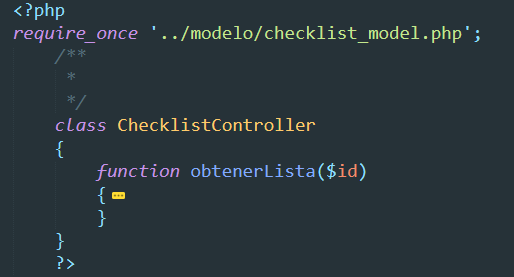
**Cancela\_controller.php**

En el aspecto funcional este archivo es igual al anterior, sólo cuenta con una función que recibe un parámetro para obtener datos de un usuario en concreto.



Dentro de la función la única parte que vale la pena mencionar es la sección del motivo, ya que aquí se captura la opción de un “select” que representa el motivo por el cual el se está cancelando a este usuario. A diferencia del formulario anterior este sólo trae el nombre de la persona, su CURP y su teléfono, datos meramente informativos ya que no pueden ser modificados, sólo están ahí para que el usuario verifique que es el beneficiario que quiere cancelar.

**Checlist\_controller.php**



Al igual que los anteriores archivos sólo existe una función en esta clase, que también se utilizar para traer datos desde la base.



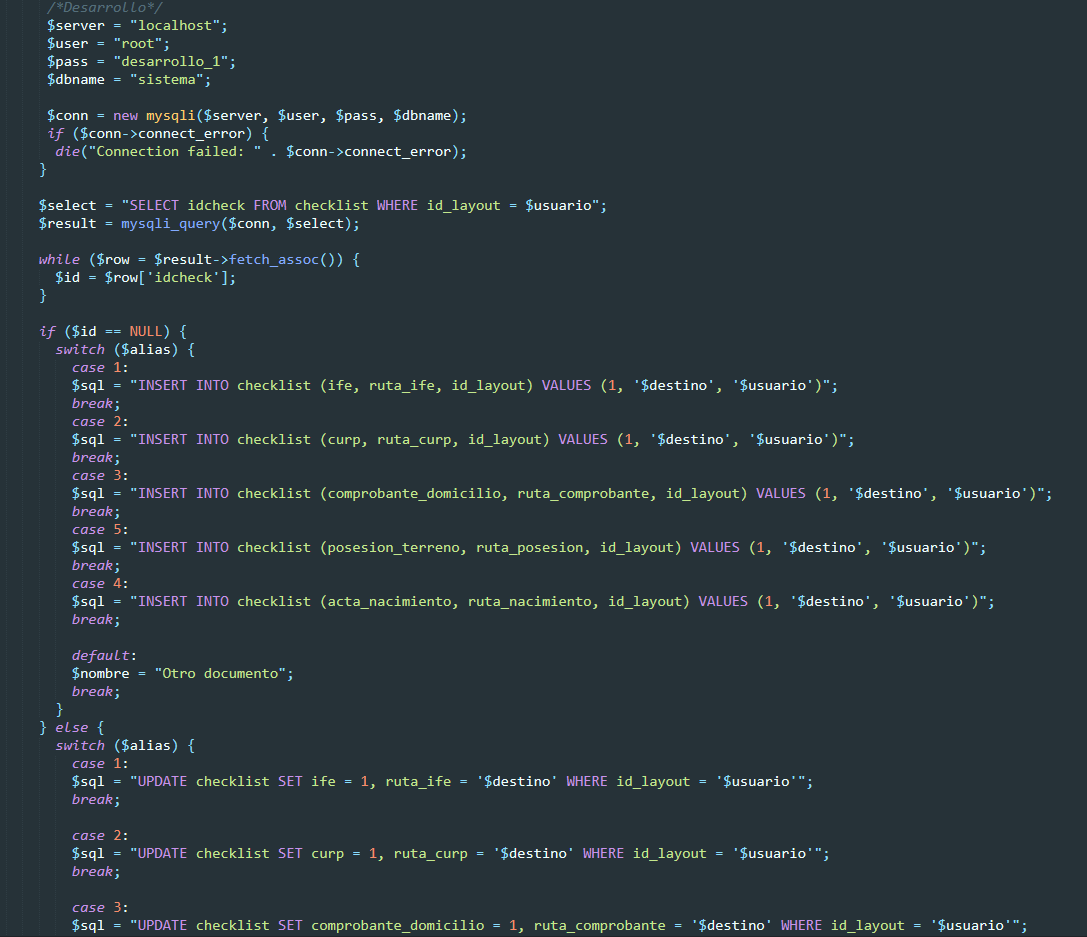
A diferencia de las dos funciones anteriores, la tarea de esta función es crear un checklist regido por una condición especifica que es “Si el campo obtenido de la base de datos es mayor a 0, entonces el check debe aparecer ya marcado, sino aparecerá vacío” acompañado de un botón para adjuntar archivos, una vez adjuntado el archivo el checkbox se marcará de manera automática, así que no hay forma de desmarcarlo después, de esta forma se puede llevar un mejor control de la documentación de los beneficiarios.



Para el desarrollo de las consultas se ocupa PDO, implementando las 4 principales estructuras: *Select, Update, Insert* y *Store Procedure*, en el caso de los *Delete* no son utilizados debido a que no se eliminará ningún registro de la base de datos para mantener el control de los datos ingresados en un historial. Se sustituye la eliminación con un cambio de estado del elemento, normalmente representado por la palabra “inactivo” o “oculto”. Entre las especificaciones de PDO que se deben tener en cuenta, se encuentra la transferencia de datos desde la vista hacia el controlador, y de este al modelo, es importante saber que para enviar los datos al controlador se utiliza el método POST, guardándolos en un arreglo que seguido de eso será dirigido al modelo, ya que al tratar de enviar variables normales la función bindParam de PDO no será capaz de reconocerlas es por esta razón que usamos el arreglo y lo manipulamos con la instrucción: **$stmt -> bindParam (‘:llave’, $arreglo[‘llave’], PDO::PARAM\_STR);** de esta forma el valor obtenido del arreglo y convertido a un String será asignado a la llave encontrada en la consulta.

Este comportamiento es propio de PDO por lo tanto puede encontrarse en su documentación siguiendo el siguiente link [PDO](http://php.net/manual/es/book.pdo.php) de igual forma si usted conoce una forma de agilizar el proceso, esta en total libertad de implementar su conocimiento.

Existen tres archivos que utilizan una estructura diferente a PDO, en este caso [mysqli](http://php.net/manual/es/book.mysqli.php) tiene una estructura más sencilla: **$query = “SELECT \* FROM usuario WHERE nombre = ‘“.$nombre.”’;** en este caso se genera la consulta como se haría en SQL, y se anexa la variable requerida directamente dentro de comillas seguidas de un punto para concatenarla y de esta forma generar la consulta que después será ejecutada por medio de **mysqli\_execute** a diferencia de pdo que se ejecuta con la siguiente sentencia **$stmt->execute();** después de recibir los parámetros necesarios para cumplir con la condición requerida. Método que podemos observar en el siguiente ejemplo:



En este ejemplo podemos ver tres de las cuatro sintaxis básicas que usaremos: *Select, Insert* y *Update* estas sentencias se pueden encontrar en el archivo *subir2.php* que se utiliza para cargar la documentación del beneficiario en el sistema guardando únicamente la ruta donde está alojado el archivo, de igual forma existe un archivo llamado *subir.php* utilizado para la misma función, pero orientado al checklist de la documentación inicial.

Es importante tener en cuenta la estructura de este switch antes de decidir modificarlo, ya que al cargar los documentos uno por uno se puede preguntar si ya existe algún registro para este beneficiario o es el primer documento que se cargará de este, así como la inserción de su respectivo bit para dar por existente el documento, la función del switch es preguntar por cada documento cargado, logrando de esta forma ordenarlos sin la necesidad de crear un registro por cada documento, sólo actualizando un solo registro para el beneficiario.

**Restricciones de usuario**

Existen tres tipos de usuario:

* *Administrador*
* *Operador*
* *Invitado*

***Administrador:***

Cuenta con todos los permisos y un panel exclusivo para el manejo de los usuarios, su creación, vista y baja. Así como el control de los usuario el administrador es el único usuario con permiso para eliminar proyectos y con ellos eliminar a los beneficiarios que están relacionados con este, en la parte técnica como se había mencionado anteriormente no se elimina ningún dato, sólo cambian de estado por inactivo, con la finalidad de más adelante obtener un reporte desde la base de datos de todos los proyectos anteriormente realizados, reporte que deberá ser generado directamente en SQL ya que actualmente no existe una función que nos permita obtenerlo de forma gráfica, puesto que no es una función primordial, aunque como encargado del sistema hay que tener en cuenta la cantidad de almacenamiento de la que disponemos, por lo que mi consejo es realizar un respaldo periódico para después limpiar la base del servidor y de esta forma esos respaldos puedan ser manipulados en un dispositivo local únicamente para obtener el reporte pertinente, a pesar de eso usted es libre de tomar sus decisiones con respecto al manejo de respaldos.

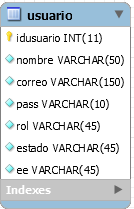
***Operador:***

El operador a diferencia del administrador cuenta con un perfil un poco más limitado ya que las opciones anteriormente mencionadas no le están permitidas al operador, en cambio sí cuenta con permisos como cancelar algún beneficiario, sustituir sus datos por los de algún familiar del mismo o incluso crear proyectos, ya que son funciones que aunque requieren de cierta responsabilidad el operador se encuentra capacitado para tomar estas decisiones, además de eso el sistema está diseñado para capturar al usuario que ha realizado el cambio sin que este lo note.

***Invitado:***

A diferencia de los dos roles anteriores, este es el usuario con más restricciones, ya que únicamente puede visualizar y exportar los datos referentes a su entidad ejecutora, esto debido a que los invitados normalmente están vinculados a una financiera en específico. Los invitados no tienen permiso de actualizar ni agregar datos al proyecto para no comprometer la información, en cuanto el invitado inicia sesión es dirigido automáticamente a la lista de proyectos.

El control de los usuarios es posible dado a la estructura de la tabla usuario:



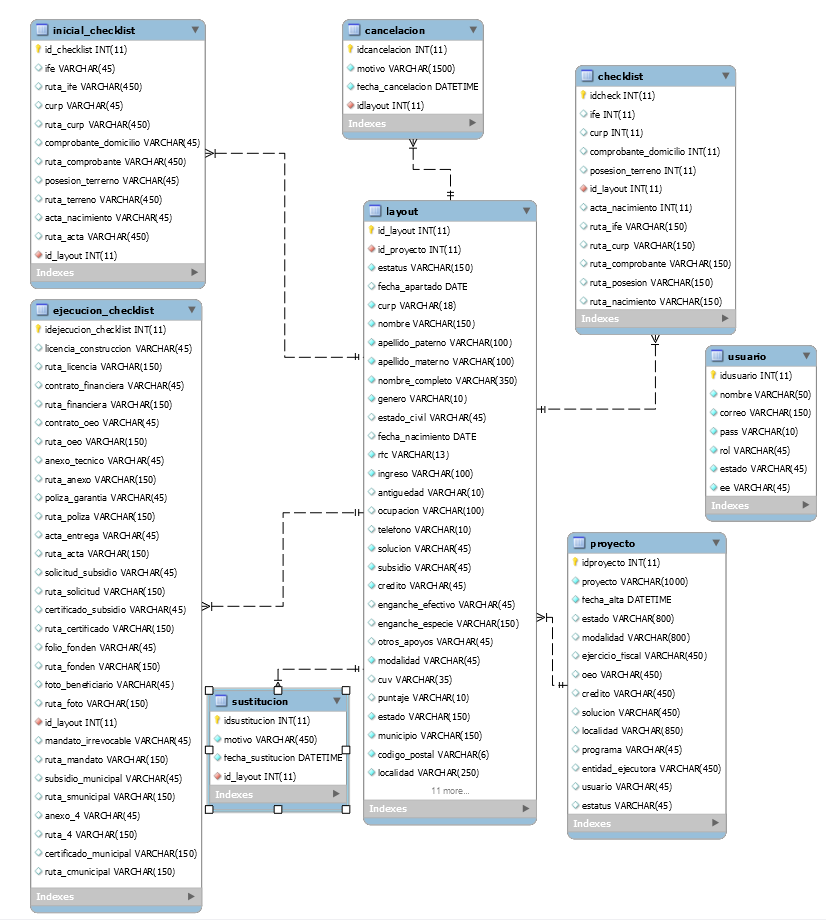
* El *idusuario* es la llave primaria de la tabla.
* Seguida del *nombre* del usuario.
* El *correo* para que el usuario pueda acceder al sistema, el correo no necesariamente debe ser real, sólo contener la sintaxis especifica ([ejemplo@ejemplo.com](mailto:ejemplo@ejemplo.com)).
* *Pass* es la columna destinada para la contraseña del usuario.
* El *rol* aquí se define si el usuario será invitado u operador, tomando esta columna como base para que el sistema muestre el contenido pertinente de cada usuario.
* *Estado* es una columna para dos valores, ‘activo’ o ‘inactivo’ sólo para definir si el usuario tiene o no acceso al sistema.
* Por último, la columna *ee* almacena la entidad financiera a la que pertenece el usuario, en caso de ser un usuario invitado, de otra forma se guardará la frese “N/F” para distinguir que es un usuario interno.

Después de ver el modelado de la tabla *usuario* podemos entender mejor como se genera el filtrado de usuarios en pocas palabras la responsabilidad recae en las columnas *estado, ee* y *rol¸* ya que son las encargadas de dar proveer las especificaciones del usuario al sistema.

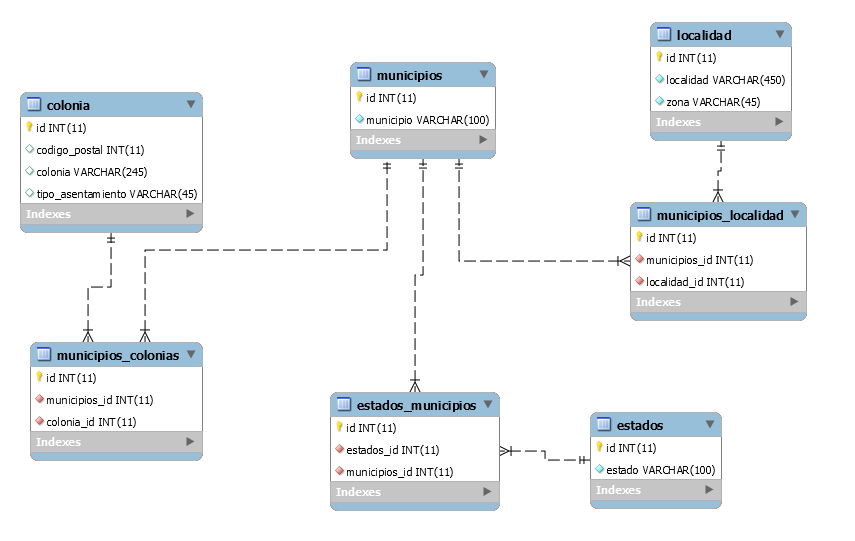
La mayoría de estas restricciones son efectuadas por medio de If’s por lo que no es difícil identificarlas de forma rápida, es importante saber que los resultados de esta tabla como *rol, nombre* y *ee* son guardados en variables de [$\_SESSION](http://php.net/manual/es/reserved.variables.session.php) para poder manipularlos y usarlos en cualquier parte del código permitiéndonos así realizar las comparaciones sin tener que recurrir a la base de datos en cada lugar que lo necesiten. Las columnas ya mencionadas se llenan por medio de *selects* con opciones predeterminadas para evitar la confusión si el usuario trata de capturar la respuesta correcta y de esta forma evitar confusiones en el sistema por mayúsculas o abreviaturas.

***Especificaciones del servidor actual***

Actualmente el proyecto se encuentra en el servidor gratuito 000webHost que provee de 10GB para almacenamiento de la base de datos, espacio para dos esquemas, los cuales ya han sido ocupados ya que este sistema trabaja con dos bases de datos separadas, la primera que cuenta con el esquema principal del sistema:



Y el segundo esquema pertenece al almacenamiento de todos los estados, municipios, localidades, colonias y códigos postales del país:



La razón por la que estos esquemas se encuentran separados es por la cantidad de información que almacena el esquema de los estados, de igual forma es necesario ser cuidadoso con esta base ya que es información publica es complicado juntar toda esta información por lo que lo mejor es mantener las bases separadas evitando así comprometer la información de las tablas, como dato curiosos gran parte de las tablas de esta base cuentan con más de 300,000 registros. Al igual que en los puntos anteriores si usted cuenta con una alternativa que nos permita mejorar este proceso es libre de implementarla. Mi consejo es siempre mantener un respaldo de esta base.

En el caso del almacenamiento del código, hay dos formas de subir cambios, la primera por filezilla y la segunda arrastrando los archivos modificados directamente al cPanel del servidor

Datos de conexión:

*Usuario:* ***sistema11***

*Contraseña:* ***\*Será entregada personalmente para evitar comprometer el documento\****

*Puerto de conexión:* ***21***

*Servidor:* ***ftp.000webhost.com***

*URL del sitio:* ***sistema11.000webhostapp.com***

Estos son los datos necesarios para realizar la conexión desde filezilla para ver el árbol del proyecto y subir los cambios desde esta opción.

Al ser un sistema completamente interno no hay ningún problema con usar un servidor gratuito, si en un futuro el programa planea liberarse para un uso más publico habría que considerar la idea de obtener un hosting, como sugerencia personal el uso de Hostinger suena razonable tomando en cuenta su precio y funciones que ofrece.