SINCRONIZACIÓN BASE DE DATOS ORIXE CON BASE DE DATOS DESTINO

Éber Cordeiro Martínez

Índice

1. Introdución

Este proxecto xorde da problemática que teñen algúns buques de pesca, os cales, carecen de conexión a Internet que é necesaria para levar a cabo a transmisión de datos dende o buque ata a base de datos central ou orixe. A información que necesitan transmitir é a relacionada coa xestión dos mantementos do barco, a súa tripulación, etc.

Dado que os buques carecen de conexión a Internet necesitan levar consigo o seu propio servidor local. Neste servidor estará a base de datos destino, na que se irá almacenando a información, e cando o buque chegue a porto, mediante unha conexión wifi, transmitirá esta información a base de datos orixe.

A sincronización realizarase a través dunha aplicación en linguaxe PHP, utilizando como motor de base de datos MySQL. Esta aplicación será a encargada de realizar en primeiro lugar, a verificación da existencia das mesmas táboas en orixe e en destino, e de non ser así, da creación das táboas que non se crearan en destino. En segundo lugar, verificará que a estrutura de cada táboa é idéntica, isto quere dicir, comprobará os nomes dos campos, o tipo de datos, e as constraints de cada táboa, e de non ser idénticas, realizará as modificacións oportunas para que a estrutura da táboa destino teña idéntica estrutura a táboa orixe. Unha vez verificados estes dous pasos realizarase a transmisión dos datos de orixe a destino, no caso de que non existan en destino ou tiveran algunha modificación. E por último, a aplicación tamén se encargará de sincronizar os datos da base de datos destino coa base de datos orixe, no caso de novos datos introducidos ou modificacións nos rexistros.

2. Manual de usuario

Para poder comezar a utilizar este programa, necesitamos ter creada unha base de datos orixe, a cal, estará formada polas táboas e datos correspondentes. E por outro lado, teremos que crear unha base de datos destino terá unha configuración previa, á cal, consiste na creación das táboas ***log_sincro*** e ***t_eliminar***. A continuación, executaremos o programa e automaticamente replicaranse as táboas e os datos da base de datos orixe na base de datos destino.

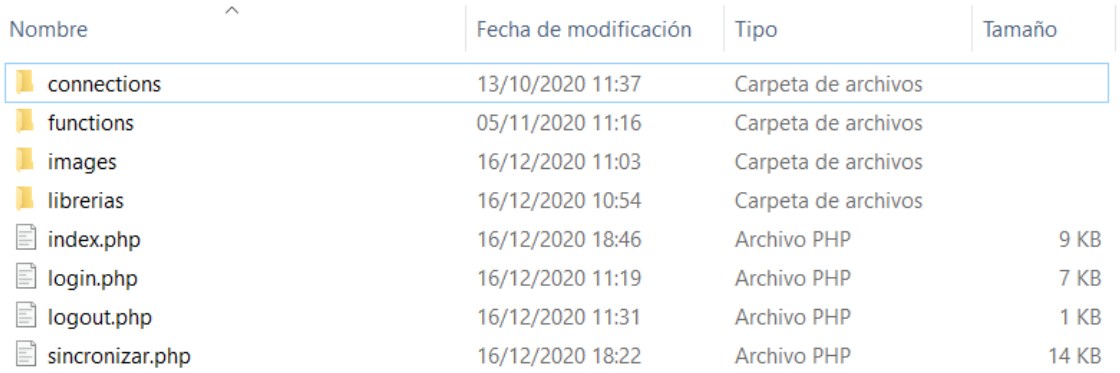
A situación inicial é que partimos dun servidor web local, no cal, teremos instalado o XAMPP. A configuración inicial necesaria, consistirá na creación dunha base de datos orixe que estará formada polas súas táboas e datos correspondentes, esta base de datos recibirá o nome de ***SincroOrigen***, e o seu usuario vai a ser ***sincroorigen***. E por outro lado, no servidor local do buque teremos a mesma situación de partida, e crearemos unha base de datos valeira que recibirá o nome de ***SincroDestino***, e o seu usuario vai ser ***sincrodestino***.

A ruta onde se almacenará todos os arquivos necesarios para o funcionamento da aplicación será ***C:\xampp\htdocs\SincroBarcos***.

3. Documentación técnica

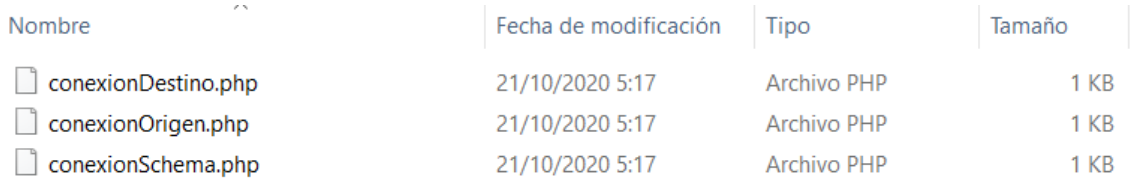
3.1. Estrutura código

A estrutura do código deste proxecto está formada polas carpetas ***connections***, ***functions***, ***images***, ***librerias*** e polos arquivos ***index.php***, este arquivo está deseñado para amosar a información das bases de datos en dúas datatables e permite seleccionar a táboa da que desexamos obter a información, o arquivo ***login.php*** consta dun login que só permite o acceso a información os usuarios dados de alta na base de datos, o arquivo ***logout.php*** permite pechar a sesión dun usuario que xa se puido loguear, e por último, o arquivo ***sincronizar.php*** este arquivo é o encargado de sincronizar as diferentes táboas.



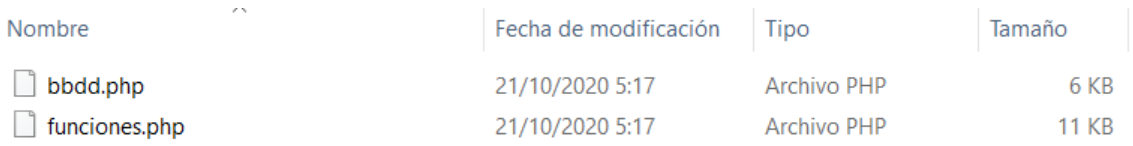
Nombre	Fecha de modificación	Tipo	Tamaño
connections	13/10/2020 11:37	Carpeta de archivos	
functions	05/11/2020 11:16	Carpeta de archivos	
images	16/12/2020 11:03	Carpeta de archivos	
librerias	16/12/2020 10:54	Carpeta de archivos	
index.php	16/12/2020 18:46	Archivo PHP	9 KB
login.php	16/12/2020 11:19	Archivo PHP	7 KB
logout.php	16/12/2020 11:31	Archivo PHP	1 KB
sincronizar.php	16/12/2020 18:22	Archivo PHP	14 KB

A carpeta ***connections*** está formada polos arquivos ***conexionDestino.php*** onde se establece a configuración de conexión a base de datos destino, ***conexionOrigen.php*** onde se establece a configuración de conexión a base de datos orixe, e por último, o arquivo ***conexionSchema.php*** onde se estable a configuración de conexión a base de datos SCHEMA, a cal, se utiliza para obter os datos referentes as FOREIGN KEYS.



Nombre	Fecha de modificación	Tipo	Tamaño
conexionDestino.php	21/10/2020 5:17	Archivo PHP	1 KB
conexionOrigen.php	21/10/2020 5:17	Archivo PHP	1 KB
conexionSchema.php	21/10/2020 5:17	Archivo PHP	1 KB

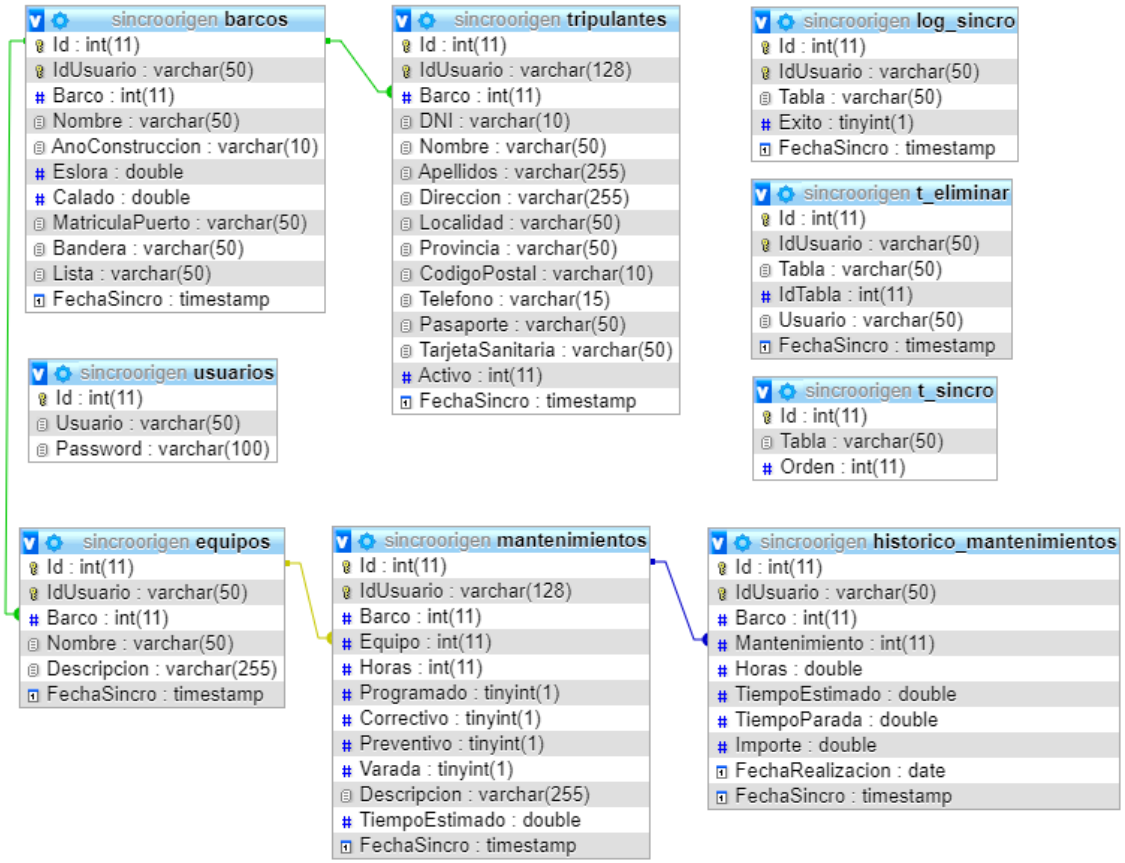
A carpeta ***functions*** está formada polos arquivos ***bbdd.php*** onde se atopan definidas todas as funcións que fan referencia a base de datos, e o arquivo ***funciones.php*** onde se atopan definidas diferentes funcións xerais.



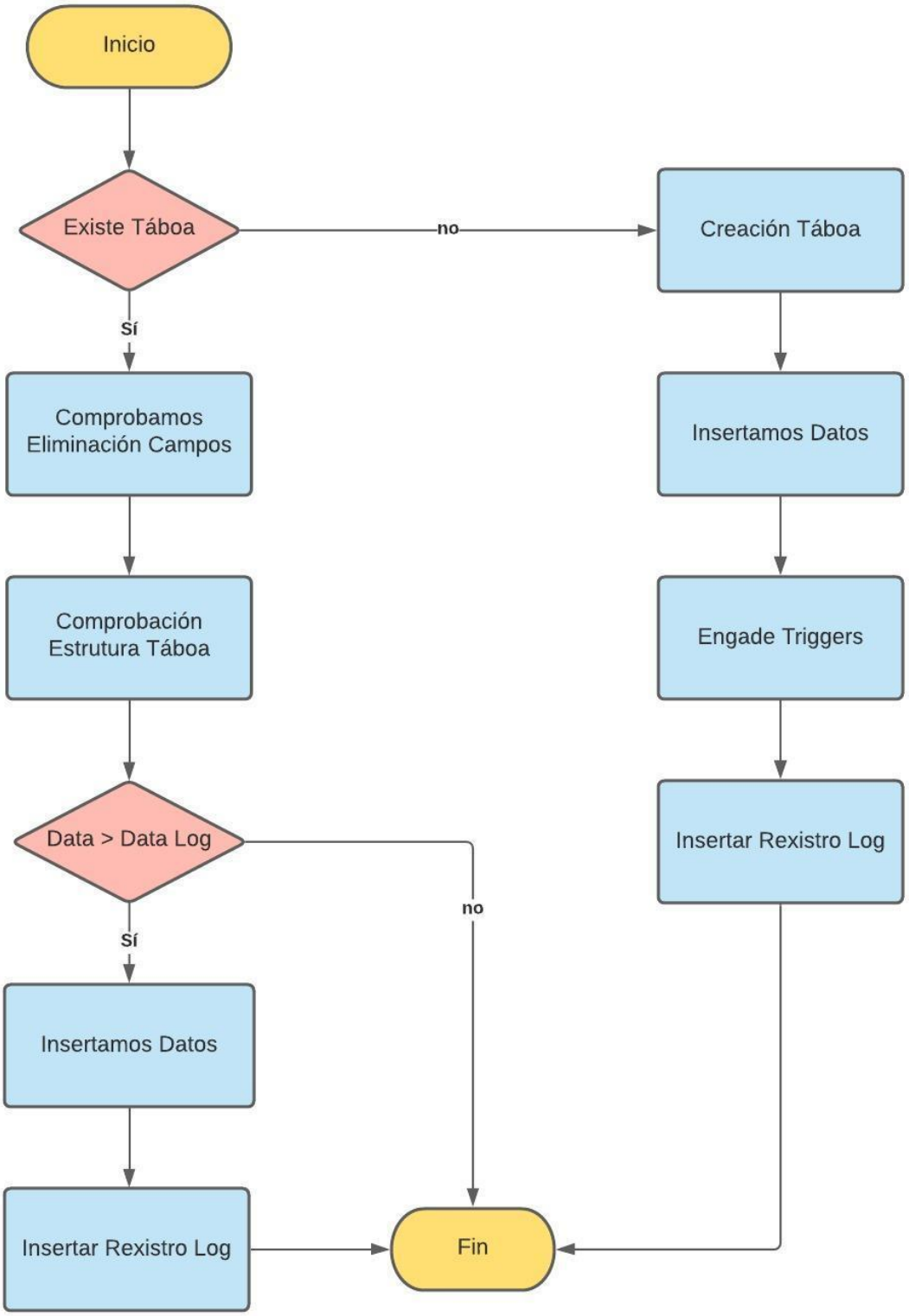
Nombre	Fecha de modificación	Tipo	Tamaño
bbdd.php	21/10/2020 5:17	Archivo PHP	6 KB
funciones.php	21/10/2020 5:17	Archivo PHP	11 KB

A carpeta ***images*** contén as imaxes que se utilizan dentro da páxina, e a carpeta ***librerias*** contén as diferentes librerías necesarias para o correcto funcionamento da páxina, como por exemplo, Bootstrap, Datatables, jQuery, etc.

3.2. Base de datos

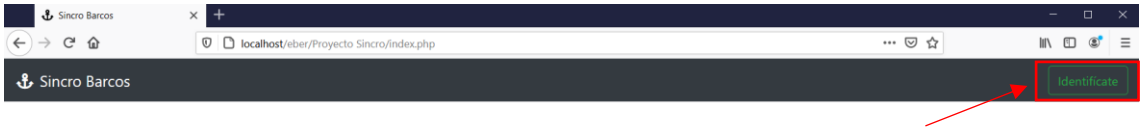


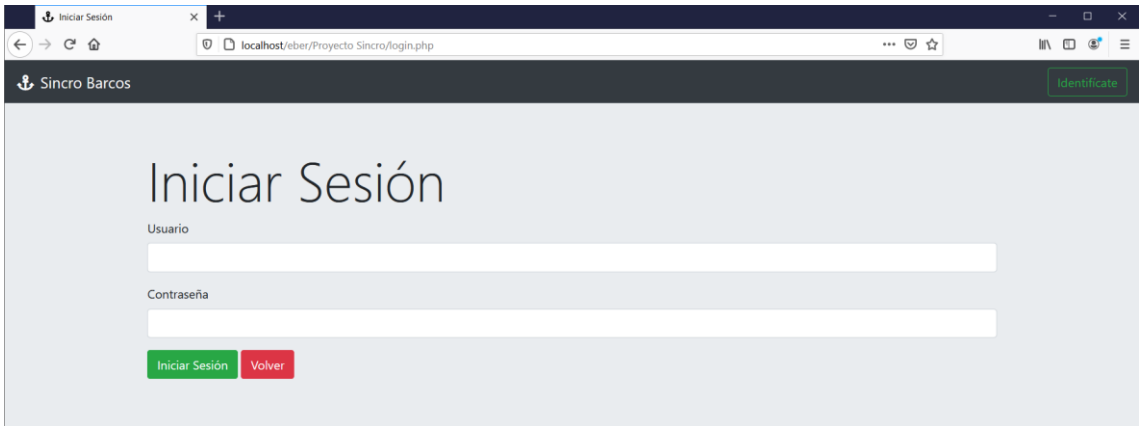
3.3. Diagrama fluxo programa

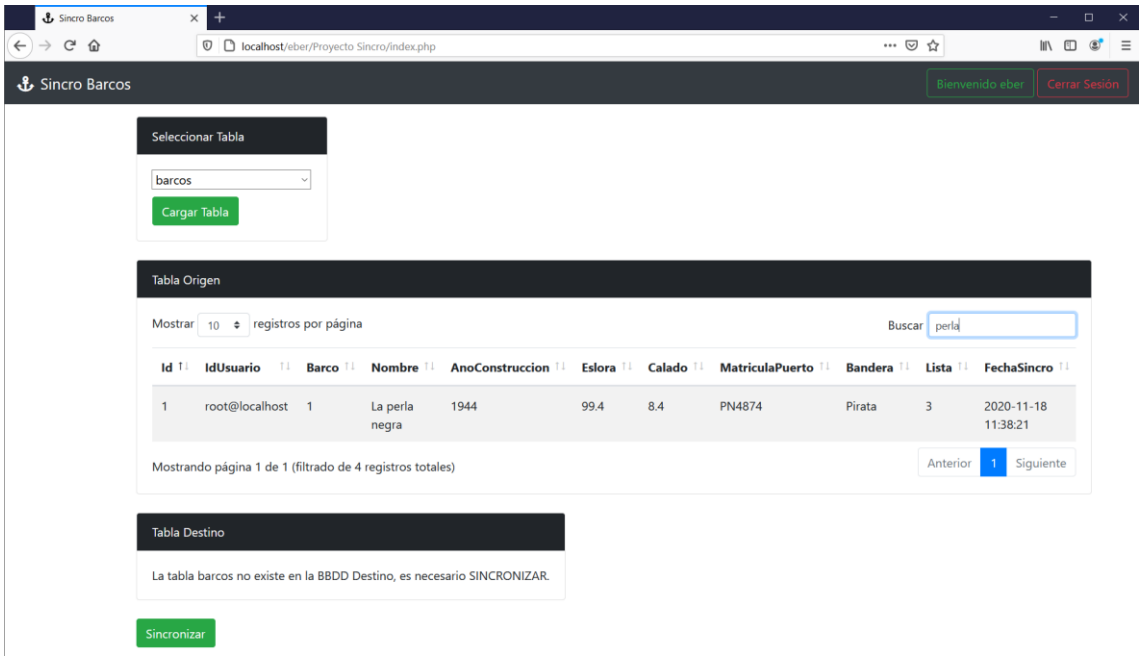


3.4. Proceso de sincronización

Antes de comezar co proceso de sincronización, é necesario realizar un proceso de autenticación mediante un login para acceder a aplicación. Unha vez logueado amosa a información actual da táboa que se seleccione nun despregable, e para rematar, debemos premer no botón de Sincronizar para que o proceso se inicie.







O proceso de sincronización comeza consultando os rexistros da táboa ***t_sincro***, nesta táboa almacénanse o nome das táboas que necesitan ser sincronizadas e a orde de creación das mesmas. Esta información almacenarase nun array indexado que vai ser recorrido, e que marca o comezo do programa.

```
// FUNCION QUE OBTIENE LAS TABLAS DE LA BASE DE DATOS ORIGEN
function obtenerTablasBDO($SyncroOrigen, $databaseSyncroO){
    mysqli_select_db($SyncroOrigen, $databaseSyncroO);
    $sqlEstruBDO = sprintf("SELECT Tabla FROM t_sincro ORDER BY Orden");
    $estruBDO = mysqli_query($SyncroOrigen, $sqlEstruBDO) or die(mysqli_error($SyncroOrigen));
    $rowsEstruBDO = mysqli_fetch_all($estruBDO);

    return $rowsEstruBDO;
}
```

Neste punto podemos dividir o fluxo do programa en dúas partes ben diferenciadas, por un lado se a táboa xa existe en destino ou se polo contrario non existe a táboa.

Táboa existe en destino

Cando a táboa xa existe en destino, o programa obtén a estrutura das táboas, tanto da táboa orixe como a táboa destino, e en primeiro lugar comproba se se eliminou algún campo da táboa orixe e de ser así, elimina dito campo na táboa destino. Deste proceso encargase a función ***comprobarEliminacionCampos***.

```
// FUNCION QUE COMPRUEBA SI SE HA ELIMINADO ALGUN CAMPO EN LA ESTRUCTURA DE LA TABLA ORIGEN
function comprobarEliminacionCampos($SyncroDestino, $databaseSyncroD, $rowsEstructuraO, $rowsEstructuraD,
$totalRowsEstructuraO, $totalRowsEstructuraD, $tabla){
    if($totalRowsEstructuraD > $totalRowsEstructuraO){
        $camposO=array();
        $camposD=array();

        for($i=0;$i<$totalRowsEstructuraO;$i++){
            $camposO[$i]=$rowsEstructuraO[$i][0];
        }

        for($j=0;$j<$totalRowsEstructuraD;$j++){
            $camposD[$j]=$rowsEstructuraD[$j][0];
        }

        $resultado=array_diff($camposD,$camposO);
        sort($resultado); // ordenar array

        // ELIMINAR CAMPO EN DESTINO
        if($resultado != NULL){
            mysqli_select_db($SyncroDestino, $databaseSyncroD);
            for($k=0;$k<count($resultado);$k++){
                $dropColumD = sprintf("ALTER TABLE ".$tabla." DROP ".$resultado[$k]);
                echo "<p>".$dropColumD."</p>";
                $columD = mysqli_query($SyncroDestino, $dropColumD) or die(mysqli_error($SyncroDestino));
            }
        }
    }
}
```

A continuación, compara a estrutura da táboa orixe coa da táboa destino, no caso de non ser idénticas podemos ter dous casos, que se trate dunha modificación dun campo existente ou por outro lado que teñamos que engadir un novo campo ou unha nova constraint. Deste proceso encargase a función ***comprobarEstructuraTablaD***.

```
// FUNCION QUE COMPRUEBA SI LA ESTRUCTURA DE LA TABLA ORIGEN ES IGUAL A LA TABLA DESTINO
function comprobarEstructuraTablaD($SyncroOrigen, $databaseSyncroO, $SyncroSchema, $databaseSchema,
$SyncroDestino, $databaseSyncroD, $rowsEstructuraO, $rowsEstructuraD, $totalRowsEstructuraO,
$totalRowsEstructuraD, $tabla){
    for($i=0;$i<$totalRowsEstructuraO;$i++){
        $coincide=false;

        for($j=0;$j<$totalRowsEstructuraD;$j++){
            if($rowsEstructuraO[$i]==$rowsEstructuraD[$j] && $coincide==false){
                $coincide=true;

            } elseif($rowsEstructuraO[$i][0]==$rowsEstructuraD[$j][0] && $coincide==false){
                echo "<p>Entro en modificar la columna ".$rowsEstructuraO[$i][0]."</p>";
                $coincide=true;
                $campo=$rowsEstructuraO[$i][0];
                $tipo=$rowsEstructuraO[$i][1];

                if($rowsEstructuraO[$i][2]=="NO"){
                    $null="NOT NULL";
                } else{
                    $null="";
                }

                switch($rowsEstructuraO[$i][3]){
                    case "PRI":
                        $key="PRIMARY KEY";
                        break;
                    case "UNI":
                        $key="UNIQUE";
                        break;
                    case "MUL":
                        $key="FOREIGN KEY";
                        break;
                    default:
                        $key="";
                }

                if($rowsEstructuraO[$i][4] != ""){
                    $default="DEFAULT ".$rowsEstructuraO[$i][4];
                } else{
                    $default="";
                }
                $extra=$rowsEstructuraO[$i][5];
```

```php
// AÑADIR/MODIFICAR CONSTRAINTS
if($key=="FOREIGN KEY"){
    $datosFK=obtenerDatosFKO($databaseSyncroO, $SyncroSchema, $databaseSchema, $tabla);
    $tablaFK=$datosFK[0][0];
    $campoFK=$datosFK[0][1];

    if($rowsEstructuraD[$j][3] == "MUL"){
        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        $addConstraintD = sprintf("ALTER TABLE ".$tabla." MODIFY ".$key."(".$campo.") REFERENCES ".$tablaFK."(".
        $campoFK.")");
        echo "<p>".$addConstraintD."</p>";
        $constraintD = mysqli_query($SyncroDestino, $addConstraintD) or die(mysqli_error($SyncroDestino));
    } else{
        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        $addConstraintD = sprintf("ALTER TABLE ".$tabla." ADD ".$key."(".$campo.") REFERENCES ".$tablaFK."(".$campoFK.
        ")");
        echo "<p>".$addConstraintD."</p>";
        $constraintD = mysqli_query($SyncroDestino, $addConstraintD) or die(mysqli_error($SyncroDestino));
    }

} elseif($key == "PRIMARY KEY"){
    if($rowsEstructuraO[$j][3] == "PRI" && $rowsEstructuraD[$j][3] == "" && $campo != $rowsEstructuraO[0][0]){
        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        $addConstraintD = sprintf("ALTER TABLE ".$tabla." DROP ".$key.", ADD ".$key."(".$rowsEstructuraO[0][0].", ".
        $campo.")");
        echo "<p>".$addConstraintD."</p>";
        $constraintD = mysqli_query($SyncroDestino, $addConstraintD) or die(mysqli_error($SyncroDestino));
    } else{
        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        $addConstraintD = sprintf("ALTER TABLE ".$tabla." ADD ".$key."(".$campo.")");
        echo "<p>".$addConstraintD."</p>";
        $constraintD = mysqli_query($SyncroDestino, $addConstraintD) or die(mysqli_error($SyncroDestino));
    }

} elseif($key != NULL){
    if($rowsEstructuraD[$j][3] != NULL){
        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        $addConstraintD = sprintf("ALTER TABLE ".$tabla." MODIFY ".$key."(".$campo.")");
        echo "<p>".$addConstraintD."</p>";
        $constraintD = mysqli_query($SyncroDestino, $addConstraintD) or die(mysqli_error($SyncroDestino));
    } else{
        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        $addConstraintD = sprintf("ALTER TABLE ".$tabla." ADD ".$key."(".$campo.")");
        echo "<p>".$addConstraintD."</p>";
        $constraintD = mysqli_query($SyncroDestino, $addConstraintD) or die(mysqli_error($SyncroDestino));
    }

}


        // MODIFICAR CAMPO
        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        $addColumD = sprintf("ALTER TABLE ".$tabla." MODIFY ".$campo." ".$tipo." ".$extra." ".$null." ".$default);
        echo "<p>".$addColumD."</p>";
        $columD = mysqli_query($SyncroDestino, $addColumD) or die(mysqli_error($SyncroDestino));
    }
}
```

O seguinte paso, unha vez comprobada a estrutura da táboa, é comprobar se existen datos que actualizar, para isto, obteremos a data máis recente da táboa ***log_sincro*** na base de datos orixe, esta data pasámoslla a función ***obtenerDatosO*** que será a encargada de obter os datos que non están actualizados.

```php
// FUNCION QUE OBTIENE LOS DATOS QUE HAN SIDO MODIFICADOS DESDE LA ULTIMA SINCRONIZACION EN ORIGEN
function obtenerDatosO($SyncroOrigen, $databaseSyncroO, $tabla, $fecha, $idBarco){
    mysqli_select_db($SyncroOrigen, $databaseSyncroO);
    $consulDatosO = sprintf("SELECT * FROM ".$tabla." WHERE FechaSincro > %s AND Barco = %s",
                            GetSQLValueString($fecha, 'text'),
                            GetSQLValueString($idBarco, 'text'));
    $datosO = mysqli_query($SyncroOrigen, $consulDatosO) or die(mysqli_error($SyncroOrigen));
    $rowDatosO = mysqli_fetch_all($datosO);

    return $rowDatosO;
}
```

Se a función nos devolve algún resultado, é dicir, o que nos devolve non é nulo pasaremos a inserción destes datos na táboa destino. Para isto utilizaremos as funcións ***obtenerCamposInsert*** e ***obtenerSentenciaUpdate***. E por último, engadiremos un novo rexistro na táboa ***log_sincro*** para deixar constancia da correcta sincronización da táboa.

```php
if($datosO != NULL){
    // PREPARACION SENTENCIA PARA INSERTAR DATOS EN LA TABLA DESTINO
    $estructuraOrigen=obtenerEstructuraOrigen($SyncroOrigen, $databaseSyncroO, $tabla);
    $rowsTablasOrigen=count($estructuraOrigen);

    // CONVERTIR EN ARRAY LOS CAMPOS
    $campos=obtenerCamposInsert($estructuraOrigen, $rowsTablasOrigen);
    $cadenaCampos=substr($campos, 1, -1);
    $arrayCampos=explode(",", $cadenaCampos);

    $sentencia=obtenerSentenciaUpdate($datosO, $arrayCampos, $rowsTablasOrigen, $tabla);
    echo "<p>".$sentencia[0]."</p>";

    // INSERTAR LOS DATOS EN LA TABLA DESTINO
    if($sentencia != NULL){
        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        for($k=0;$k<count($sentencia);$k++){
            $addTablaD = sprintf($sentencia[$k]);
            $tablaD = mysqli_query($SyncroDestino, $addTablaD) or die(mysqli_error($SyncroDestino));
        }
    }

    // INSERTAR EN EL LOG
    $sentenciaLog=obtenerSentenciaLog($tabla);
    if($sentenciaLog != NULL){
        mysqli_select_db($SyncroOrigen, $databaseSyncroO);
        $addTablaLogO = sprintf($sentenciaLog);
        $tablaLogO = mysqli_query($SyncroOrigen, $addTablaLogO) or die(mysqli_error($SyncroOrigen));

        mysqli_select_db($SyncroDestino, $databaseSyncroD);
        $addTablaLogD = sprintf($sentenciaLog);
        $tablaLogD = mysqli_query($SyncroDestino, $addTablaLogD) or die(mysqli_error($SyncroDestino));
    }
}

// FUNCION QUE OBTIENE LOS CAMPOS NECESARIOS PARA REALIZAR LA INSERCION DE DATOS
function obtenerCamposInsert($estructura, $rowsTablas){
    $campos="";
    $contador=0;

    do{
        if($campos == ""){
            $campos="(".$estructura[$contador][0];
        } else{
            $campos.=", ".$estructura[$contador][0];
        }
        $contador++;
    } while($contador < $rowsTablas);

    $campos.=")";

    return $campos;
}
```

```php
// FUNCION QUE OBTIENE LOS VALORES DE LOS CAMPOS NECESARIOS PARA REALIZAR LA INSERCION DE DATOS
function obtenerValuesInsert($contenido, $numContenido, $rowsTablas){
    $values="VALUES";
    $contador=0;
    $contador2=0;

    do{
        if($values == "VALUES"){
            $values.="(";
        } else{
            $values.="), (";
        }

        $sentencia="";
        $contador2=0;
        do{
            if($sentencia == ""){
                $sentencia.=GetSQLValueString($contenido[$contador][$contador2], 'text');
            } else{
                $sentencia.=", ".GetSQLValueString($contenido[$contador][$contador2], 'text');
            }
            $contador2++;

        } while($contador2 < $rowsTablas);
        $contador++;
        $values.=$sentencia;

    } while($contador < $numContenido);

    $values.=")";

    return $values;
}
```

Táboa non existe en destino

Por outro lado, cando a táboa non existe na base de datos destino crearemos dita táboa obtendo a creación da táboa orixe. Este proceso será realizado pola función ***obtenerCreacionTabla***.

```php
// FUNCION QUE OBTIENE LA SENTENCIA DE CREACION DE UNA TABLA
function obtenerCreacionTabla($SyncroOrigen, $databaseSyncroO, $tabla){
    mysqli_select_db($SyncroOrigen, $databaseSyncroO);
    $consulCreateO = sprintf("SHOW CREATE TABLE ".$tabla);
    $createO = mysqli_query($SyncroOrigen, $consulCreateO) or die(mysqli_error($SyncroOrigen));
    $rowsCreateO = mysqli_fetch_all($createO);

    return $rowsCreateO;
}
```

Unha vez creada a táboa insertaremos os datos que esteen almacenados na táboa orixe, utilizando as funcións ***obtenerCamposInsert*** e ***obtenerValuesInsert*** (xa mencionadas anteriormente).

O seguinte paso, consiste en engadir a táboa unha serie de triggers necesarios para o correcto funcionamento do programa. Por unha banda, engadiremos un trigger encargado de engadir o usuario actual cando se inserte un novo rexistro na táboa, e por outro lado, engadiremos outro trigger encargado de gardar a información esencial (nome da táboa, id e usuario), cando se elimine un rexistro na táboa e inserte eses datos na táboa ***t_eliminar***. E por último, engadiremos un novo rexistro na táboa ***log_sincro*** para deixar constancia da correcta sincronización da táboa.

```
// INSERTAR DATOS EN LA TABLA DESTINO
mysqli_select_db($SyncroDestino, $databaseSyncroD);
$consulInsertD = "INSERT INTO ".$tabla." ".$campos." ".$values;
$insertD = mysqli_query($SyncroDestino, $consulInsertD) or die(mysqli_error($SyncroDestino));

// PREPARACION SENTENCIA PARA INSERTAR TRIGGERS EN LA TABLA DESTINO
$sentenciaTriggerUsuarioD=crearTriggerUsuarioBDD($tabla);
$sentenciaTriggerEliminarD=crearTriggerEliminarBDD($tabla);
echo "<p>".$sentenciaTriggerUsuarioD."</p>";
echo "<p>".$sentenciaTriggerEliminarD."</p>";


// AÑADIR TRIGGERS BASE DATOS DESTINO
if($sentenciaTriggerUsuarioD != NULL){
    $db=new mysqli($hostnameSyncroD, $usernameSyncroD, $passwordSyncroD, $databaseSyncroD);
    $db->multi_query($sentenciaTriggerUsuarioD);
}

if($sentenciaTriggerEliminarD != NULL){
    $db=new mysqli($hostnameSyncroD, $usernameSyncroD, $passwordSyncroD, $databaseSyncroD);
    $db->multi_query($sentenciaTriggerEliminarD);
}

// INSERTAR EN EL LOG
$sentenciaLog=obtenerSentenciaLog($tabla);
if($sentenciaLog != NULL){
    mysqli_select_db($SyncroOrigen, $databaseSyncroO);
    $addTablaLogO = sprintf($sentenciaLog);
    $tablaLogO = mysqli_query($SyncroOrigen, $addTablaLogO) or die(mysqli_error($SyncroOrigen));

    mysqli_select_db($SyncroDestino, $databaseSyncroD);
    $addTablaLogD = sprintf($sentenciaLog);
    $tablaLogD = mysqli_query($SyncroDestino, $addTablaLogD) or die(mysqli_error($SyncroDestino));
}

// FUNCION QUE CREARA EL TRIGGER NECESARIO PARA SABER EL USUARIO ACTUAL EN CADA TABLA DE LA BASE DE DATOS DESTINO
function crearTriggerUsuarioBDD($tabla){

    $sentencia="CREATE TRIGGER `usuario_".$tabla."` BEFORE INSERT ON `".$tabla."` FOR EACH ROW BEGIN IF NEW.
    IdUsuario = '' THEN SET NEW.IdUsuario = (SELECT USER()); END IF; END;";

    return $sentencia;
}

// FUNCION QUE CREARA EL TRIGGER NECESARIO PARA GUARDAR LOS DATOS IMPRESCINDIBLES PARA ELIMINAR UN REGISTRO EN
CADA TABLA DE LA BASE DE DATOS DESTINO
function crearTriggerEliminarBDD($tabla){

    if($tabla != 't_eliminar' || $tabla != 'log_sincro'){
        $sentencia="CREATE TRIGGER `eliminar_".$tabla."` BEFORE DELETE ON `".$tabla."` FOR EACH ROW BEGIN INSERT
        INTO `t_eliminar` (Tabla, IdTabla, Usuario) VALUES('".$tabla."', OLD.Id, OLD.IdUsuario); END;";

        return $sentencia;
    }

}
```

O derradeiro paso do programa, consiste en comprobar se temos datos en destino que temos que sincronizar cara orixe. Para isto, utilizamos a mesma dinámica que cando sincronizabamos os datos de orixe cara destino.

En primeiro lugar obtemos a fecha máis actual da táboa ***log_sincro*** da base de datos destino e pasámoslla a función ***obtenerDatosD***, que será a encargada de obter os datos que non están actualizados en destino.

```
// FUNCION QUE OBTIENE LOS DATOS QUE HAN SIDO MODIFICADOS DESDE LA ULTIMA SINCRONIZACION EN DESTINO
function obtenerDatosD($SyncroDestino, $databaseSyncroD, $tabla, $fecha){
    mysqli_select_db($SyncroDestino, $databaseSyncroD);
    $consulDatosD = sprintf("SELECT * FROM ".$tabla." WHERE FechaSincro > %s",
                            GetSQLValueString($fecha, 'text'));
    $datosD = mysqli_query($SyncroDestino, $consulDatosD) or die(mysqli_error($SyncroDestino));
    $rowDatosD = mysqli_fetch_all($datosD);

    return $rowDatosD;
}
```

Se a función nos devolve algún resultado, é dicir, o que nos devolve non é nulo pasaremos a inserción destes datos na táboa destino. Para isto utilizaremos as funcións ***obtenerCamposInsert*** e ***obtenerSentenciaUpdate*** (xa mencionadas anteriormente). E por último, engadiremos un novo rexistro na táboa ***log_sincro*** para deixar constancia da correcta sincronización da táboa.

```
// PREPARACION SENTENCIA PARA INSERTAR DATOS EN LA TABLA ORIGEN
$estructuraDestino=obtenerEstructuraDestino($SyncroDestino, $databaseSyncroD, $tabla);
$rowsTablasDestino=count($estructuraDestino);

// CONVERTIR EN ARRAY LOS CAMPOS DEVUELTOS
$campos=obtenerCamposInsert($estructuraDestino, $rowsTablasDestino);
$cadenaCampos=substr($campos, 1, -1);
$arrayCampos=explode(",", $cadenaCampos);

$sentencia=obtenerSentenciaUpdate($datosD, $arrayCampos, $rowsTablasDestino, $tabla);
echo "<p>".$sentencia[0]."</p>";

// INSERTAR LOS DATOS EN LA TABLA ORIGEN
if($sentencia != NULL){
    mysqli_select_db($SyncroOrigen, $databaseSyncroO);
    for($k=0;$k<count($sentencia);$k++){
        $addTablaO = sprintf($sentencia[$k]);
        $tablaO = mysqli_query($SyncroOrigen, $addTablaO) or die(mysqli_error($SyncroOrigen));
    }
}

// INSERTAR EN EL LOG
$sentenciaLog=obtenerSentenciaLog($tabla);
if($sentenciaLog != NULL){
    mysqli_select_db($SyncroOrigen, $databaseSyncroO);
    $addTablaLogO = sprintf($sentenciaLog);
    $tablaLogO = mysqli_query($SyncroOrigen, $addTablaLogO) or die(mysqli_error($SyncroOrigen));

    mysqli_select_db($SyncroDestino, $databaseSyncroD);
    $addTablaLogD = sprintf($sentenciaLog);
    $tablaLogD = mysqli_query($SyncroDestino, $addTablaLogD) or die(mysqli_error($SyncroDestino));
}
```

4. Presuposto

	Coste	Horas	Total (€)
Aluguer dominio	10 €/ano	1	10 €
Aluguer hosting	10 €/mes	12	120 €
Planificación, busca de información e deseño	30 €/hora	10	300 €
Proceso de codificación	30 €/hora	40	1.200 €
Documentación	30 €/hora	3	90 €
		IVE(21%)	361,20 €
			TOTAL: 2081,20 €

5. Melloras

Ademais do xa proposto neste proxecto, podemos engadir funcionalidades en futuras versións.

- Crear un servizo de transmisión de documentos e arquivos a través dun servidor FTP.
- Administrar unha listaxe coas táboas que se queren sincronizar, así como, en que dirección desexamos que se realice a sincronización, é dicir, de orixe a destino, de destino a orixe, ou en ambas direccións.
- Permitir dende as Datatables poder engadir, modificar ou eliminar un rexistro.