

Universidad Mariano Gálvez de Guatemala

**Ingeniería en Sistemas
Análisis de Sistemas II
Ángel Atilio Maltez Cifuentes**

MANUAL TECNICO

Luis Francisco Gómez Rodríguez

Carnet: 3090-19-6864

CONEXIÓN BASE DATOS

The screenshot shows the Visual Studio Code interface with the 'modelo_conexion.php' file open in the center editor tab. The code implements a PDO connection to a MySQL database with specific parameters like host, user, password, and database name. It includes exception handling and a function to close the connection.

```
<?php
class conexionBD
{
    public function conexionPDO(){
        $host = "localhost";
        $usuario = "root";
        $contrasena = "";
        $dbName = "asilo";
        try{
            $pdo = new PDO("mysql:host=$host;dbname=$dbName",$usuario,$contrasena);
            $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            $pdo->exec("set names utf8");
        }catch(exception $e){
            echo "la conexión ha fallado";
        }
    }
    function cerrarConexion(){
        $this->$pdo=null;
    }
}
```

NUESTRO CONTROLADOR CAJA DE REGISTRO

The screenshot shows the Visual Studio Code interface with the 'controlador_caja_registro.php' file open in the center editor tab. The code defines a controller class for managing a 'Caja' entity. It includes methods for listing, creating, updating, and deleting entries, utilizing a 'Modelo_Caja' instance to interact with the database.

```
require '../model/model_caja.php';
$MU = new Modelo_Caja(); //Instaciamos
$id = strtoupper($_POST['id']);
$fe = strtoupper($_POST['fe']);
$mo = strtoupper($_POST['mo']);
$t1 = strtoupper($_POST['t1']);
$d1 = strtoupper($_POST['d1']);
$consulta = $MU->Registrar_Caja($id,$fe,$mo,$t1,$d1);
echo $consulta;
```

NUESTRO MODELO DE CAJA

The screenshot shows the Visual Studio Code interface with the file `model_caja.php` open. The code implements a `Modelo_Caja` class that extends `conexionBD`. It contains two main public functions: `Listar_Caja` and `Registrar_Caja`. The `Listar_Caja` function uses a stored procedure to fetch all boxes between specified dates and tipologies. The `Registrar_Caja` function uses another stored procedure to insert a new box record.

```
<?php
require_once 'modelo_conexion.php';

class Modelo_Caja extends conexionBD{

    public function Listar_Caja($inicio,$fin,$tip){
        $c = conexionBD::conexionPDO();
        $sql = "CALL SP_LISTAR_CAJA(?, ?, ?, ?)";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->bindParam(1,$inicio);
        $query->bindParam(2,$fin);
        $query->bindParam(3,$tip);
        $query->execute();
        $resultado = $query->fetchAll(PDO::FETCH_ASSOC);
        foreach($resultado as $resp){
            $arreglo["data"][] = $resp;
        }
    }

    return $arreglo;
    conexionBD::cerrarConexion();
}

public function Registrar_Caja($id,$fe,$mo,$ti,$de){
    $c = conexionBD::conexionPDO();

    $sql = "CALL SP_REGISTRAR_CAJA(?, ?, ?, ?, ?, ?)";
    $query = $c->prepare($sql);
    $query->bindParam(1,$id);
    $query->bindParam(2,$fe);
    $query->bindParam(3,$mo);
    $query->bindParam(4,$ti);
    $query->bindParam(5,$de);
    $resultado = $query->execute();
    if($resultado){
        return 1;
    }else{
        return 0;
    }
    conexionBD::cerrarConexion();
}
```

The screenshot shows the Visual Studio Code interface with the file `model_caja.php` open. The code implements a `Modelo_Caja` class that extends `conexionBD`. It contains two main public functions: `Listar_Caja` and `Modificar_Caja`. The `Listar_Caja` function is identical to the one in the previous screenshot. The `Modificar_Caja` function uses a stored procedure to update an existing box record based on its ID and other parameters.

```
$query->bindParam(3,$mo);
$query->bindParam(4,$ti);
$query->bindParam(5,$de);
$resultado = $query->execute();
if($resultado){
    return 1;
}else{
    return 0;
}
conexionBD::cerrarConexion();}

public function Modificar_Caja($id,$fe,$mo,$ti,$de,$idca){
    $c = conexionBD::conexionPDO();

    $sql = "CALL SP_MODIFICAR_CAJA(?, ?, ?, ?, ?, ?)";
    $query = $c->prepare($sql);
    $query->bindParam(1,$id);
    $query->bindParam(2,$fe);
    $query->bindParam(3,$mo);
    $query->bindParam(4,$ti);
    $query->bindParam(5,$de);
    $query->bindParam(6,$idca);
    $resultado = $query->execute();
    if($resultado){
        return 1;
    }else{
        return 0;
    }
    conexionBD::cerrarConexion();
}
```

JS DE CAJA

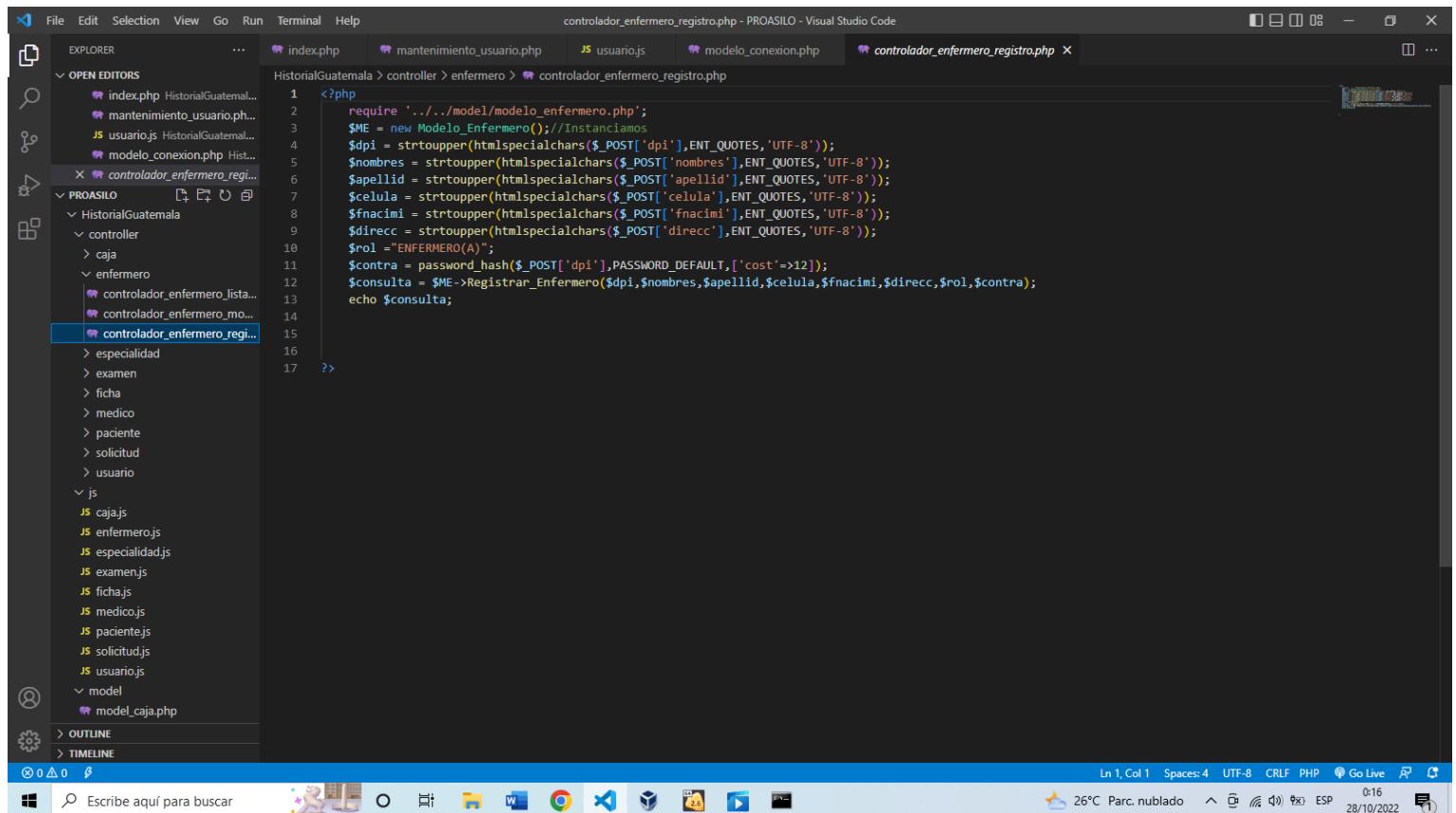
The screenshot shows the Visual Studio Code interface with the 'caja.js' file open in the center editor tab. The code is a JavaScript function named 'listado_caja' that initializes a DataTable for managing cash boxes. It includes logic to validate date inputs and handle asynchronous requests to a controller. The code is well-formatted with line numbers and syntax highlighting.

```
1 var tbl_caja;
2 function listado_caja(){
3     let ini = document.getElementById('txt_fechainicio').value;
4     let fin = document.getElementById('txt_fechafin').value;
5     let tip = document.getElementById('select_tipo_busqueda').value;
6     let nombre = "Caja Todos";
7     if(tip!=""){
8         nombre=tip;
9     }
10    if(fin<ini){
11        return Swal.fire("Mensaje de Advertencia","La fecha final no debe ser menor a la de inicio","warning");
12    }
13    tbl_caja = $("#tabla_caja").DataTable({
14        "ordering":false,
15        "bLengthChange":true,
16        "searching": { "regex": false },
17        "lengthMenu": [ [ 10, 25, 50, 100, -1], [10, 25, 50, 100, "All"] ],
18        "pageLength": 100,
19        "destroy":true,
20        "async": false ,
21        "processing": true,
22        "ajax":{
23            "url":"../controller/caja/controlador_caja_listar.php",
24            "type:'POST',
25            "data:{
26                ini:ini,
27                fin:fin,
28                tip:tip
29            }
30        },
31        "columns":[
32            {"defaultContent":""},
33            {"data":"caja_registro"},
34            {"data":"caja_monto"},
35            {"data":"caja_tipo"},
36            {"data":"caja_descripcion"},
37            {"defaultContent":"<button class='editar btn btn-primary btn-sm'><i class='fa fa-edit'></i></button>"}
38        ],
39        "fnRowCallback": function( nRow, aData, iDisplayIndex, iDisplayIndexFull ) {
40            $($.nRow).find("td")[1].css('text-align', 'center');
41            $($.nRow).find("td")[2].css('text-align', 'center');
42            $($.nRow).find("td")[3].css('text-align', 'center');
43            $($.nRow).find("td")[5].css('text-align', 'center');
44        },
45        "language":idioma_espanol,
46        select: true,
47        dom: 'Bfrtip',
48        buttons: [
49            {
50                extend: 'excelHtml5',
51                title: 'Listado de '+nombre,
52                exportOptions: [
53                    columns: [ 1, 2, 3,4 ]
54                ],
55            },
56            {
57                extend: 'pdfHtml5',
58                title: 'Listado de '+nombre,
59                pageSize: 'LEGAL',
60                customize: function ( doc ) {
61
62                    doc.content[1].table.widths =
63                        Array(doc.content[1].table.body[0].length + 1).join('*').split('');
64                    var rowCount = doc.content[1].table.body.length;
65                    for ( i = 1; i < rowCount; i++ ) {
66
67                        doc.content[1].table.body[i][0].alignment = 'center';
68                        doc.content[1].table.body[i][1].alignment = 'center';
69                        doc.content[1].table.body[i][2].alignment = 'center';
70                        doc.content[1].table.body[i][3].alignment = 'center';
71
72                }
73            }
74        ]
75    }
76}
```

This screenshot shows the same 'caja.js' file in Visual Studio Code, but with a different section of the code visible. The code continues from the previous snippet, focusing on the 'buttons' configuration for the DataTable. It includes logic for generating Excel and PDF reports, setting column widths, and aligning table data. The code is annotated with comments explaining its purpose.

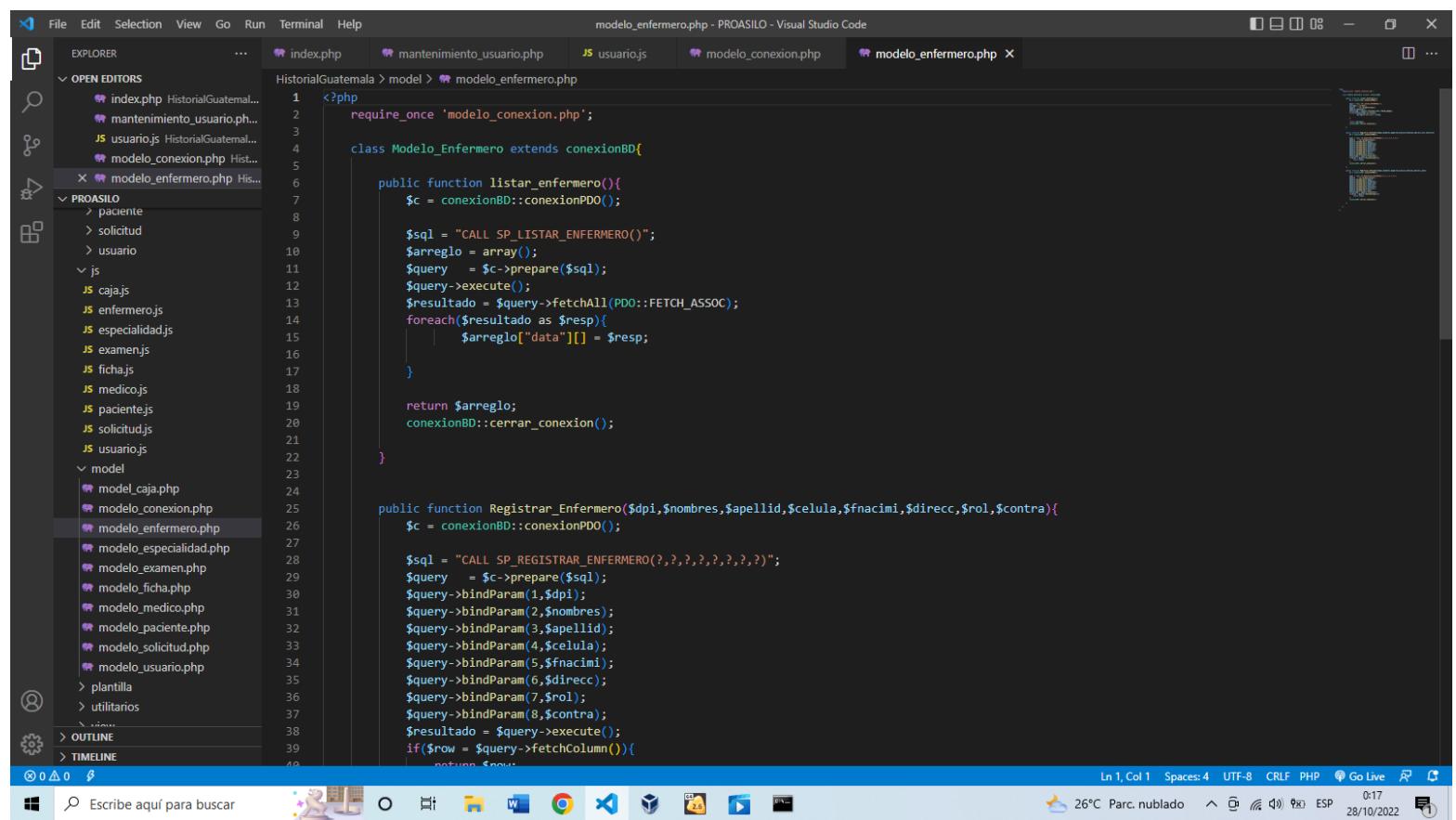
```
1 var tbl_caja;
2 function listado_caja(){
3     let ini = document.getElementById('txt_fechainicio').value;
4     let fin = document.getElementById('txt_fechafin').value;
5     let tip = document.getElementById('select_tipo_busqueda').value;
6     let nombre = "Caja Todos";
7     if(tip!=""){
8         nombre=tip;
9     }
10    if(fin<ini){
11        return Swal.fire("Mensaje de Advertencia","La fecha final no debe ser menor a la de inicio","warning");
12    }
13    tbl_caja = $("#tabla_caja").DataTable({
14        "ordering":false,
15        "bLengthChange":true,
16        "searching": { "regex": false },
17        "lengthMenu": [ [ 10, 25, 50, 100, -1], [10, 25, 50, 100, "All"] ],
18        "pageLength": 100,
19        "destroy":true,
20        "async": false ,
21        "processing": true,
22        "ajax":{
23            "url":"../controller/caja/controlador_caja_listar.php",
24            "type:'POST',
25            "data:{
26                ini:ini,
27                fin:fin,
28                tip:tip
29            }
30        },
31        "columns":[
32            {"defaultContent":""},
33            {"data":"caja_registro"},
34            {"data":"caja_monto"},
35            {"data":"caja_tipo"},
36            {"data":"caja_descripcion"},
37            {"defaultContent":"<button class='editar btn btn-primary btn-sm'><i class='fa fa-edit'></i></button>"}
38        ],
39        "fnRowCallback": function( nRow, aData, iDisplayIndex, iDisplayIndexFull ) {
40            $($.nRow).find("td")[1].css('text-align', 'center');
41            $($.nRow).find("td")[2].css('text-align', 'center');
42            $($.nRow).find("td")[3].css('text-align', 'center');
43            $($.nRow).find("td")[5].css('text-align', 'center');
44        },
45        "language":idioma_espanol,
46        select: true,
47        dom: 'Bfrtip',
48        buttons: [
49            {
50                extend: 'excelHtml5',
51                title: 'Listado de '+nombre,
52                exportOptions: [
53                    columns: [ 1, 2, 3,4 ]
54                ],
55            },
56            {
57                extend: 'pdfHtml5',
58                title: 'Listado de '+nombre,
59                pageSize: 'LEGAL',
60                customize: function ( doc ) {
61
62                    doc.content[1].table.widths =
63                        Array(doc.content[1].table.body[0].length + 1).join('*').split('');
64                    var rowCount = doc.content[1].table.body.length;
65                    for ( i = 1; i < rowCount; i++ ) {
66
67                        doc.content[1].table.body[i][0].alignment = 'center';
68                        doc.content[1].table.body[i][1].alignment = 'center';
69                        doc.content[1].table.body[i][2].alignment = 'center';
70                        doc.content[1].table.body[i][3].alignment = 'center';
71
72                }
73            }
74        ]
75    }
76}
```

CONTROLADOR DE ENFERMER@



```
<?php
require '../../../../../model/modelo_enfermero.php';
$ME = new Modelo_Enfermero(); //Instanciamos
$dpi = strtoupper(htmlspecialchars($_POST['dpi'],ENT_QUOTES,'UTF-8'));
$nombres = strtoupper(htmlspecialchars($_POST['nombres'],ENT_QUOTES,'UTF-8'));
$apellid = strtoupper(htmlspecialchars($_POST['apellid'],ENT_QUOTES,'UTF-8'));
$celula = strtoupper(htmlspecialchars($_POST['celula'],ENT_QUOTES,'UTF-8'));
$fnacimi = strtoupper(htmlspecialchars($_POST['fnacimi'],ENT_QUOTES,'UTF-8'));
$direcc = strtoupper(htmlspecialchars($_POST['direcc'],ENT_QUOTES,'UTF-8'));
$rol = "ENFERMERO(A)";
$contra = password_hash($_POST['dpi'],PASSWORD_DEFAULT,[ 'cost'=>12]);
$consulta = $ME->Registrar_Enfermero($dpi,$nombres,$apellid,$celula,$fnacimi,$direcc,$rol,$contra);
echo $consulta;
echo ">>>
```

NUESTRO MODELO ENFERMER@



```
<?php
require_once 'modelo_conexion.php';

class Modelo_Enfermero extends conexionBD{

    public function listar_enfermero(){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_LISTAR_ENFERMERO();";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->execute();
        $resultado = $query->fetchAll(PDO::FETCH_ASSOC);
        foreach($resultado as $resp){
            $arreglo["data"][] = $resp;
        }

        return $arreglo;
        conexionBD::cerrarConexion();
    }

    public function Registrar_Enfermero($dpi,$nombres,$apellid,$celula,$fnacimi,$direcc,$rol,$contra){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_REGISTRAR_ENFERMERO(?, ?, ?, ?, ?, ?, ?)";
        $query = $c->prepare($sql);
        $query->bindParam(1,$dpi);
        $query->bindParam(2,$nombres);
        $query->bindParam(3,$apellid);
        $query->bindParam(4,$celula);
        $query->bindParam(5,$fnacimi);
        $query->bindParam(6,$direcc);
        $query->bindParam(7,$rol);
        $query->bindParam(8,$contra);
        $resultado = $query->execute();
        if($row = $query->fetchColumn()){
            return $row;
        }
    }
}
```

JS ENFERMER@

The screenshot shows the Visual Studio Code interface with the 'enfermero.js' file open in the center editor tab. The code is a JavaScript function for listing nurses using a DataTable plugin. It includes handling for row click events to edit data and a modal for registration.

```
1 var tbl_enfermero;
2 function listar_enfermero(){
3     tbl_enfermero = $("#tbl_enfermero").DataTable({
4         "ordering":false,
5         "lengthChange":true,
6         "searching": { "regex": false },
7         "lengthMenu": [ [10, 25, 50, 100, -1], [10, 25, 50, 100, "All"] ],
8         "pageLength": 10,
9         "destroy":true,
10        "async": false ,
11        "processing": true,
12        "ajax":{
13            "url":"../controller/enfermero/controlador_enfermero_listar.php",
14            "type": "POST"
15        },
16        "columns":[
17            {"defaultContent":""},
18            {"data":"personal_salud_dpi"},
19            {"data":"enfermero"},
20            {"data":"personal_salud_ceular"}, 
21            {"data":"personal_salud_fnacimiento"}, 
22            {"data":"personal_salud_direccion"}, 
23            {"defaultContent":"><button class='editar btn btn-primary btn-sm'><i class='fa fa-edit'></i></button>"}
24        ],
25        "fnRowCallback": function( nRow, aData, iDisplayIndex, iDisplayIndexFull ) {
26            $($.nRow).find("td")[3]).css('text-align', 'center' );
27            $($.nRow).find("td")[4]).css('text-align', 'center' );
28            $($.nRow).find("td")[6]).css('text-align', 'center' );
29        },
30        "language":idioma_espanol,
31        select: true
32    });
33    tbl_enfermero.on('draw.dt',function(){
34        var PageInfo = $('#tbl_enfermero').DataTable().page.info();
35        tbl_enfermero.column(0, {page: 'current'}).nodes().each(function(cell, i){
36            cell.innerHTML = i + 1 + PageInfo.start;
37        });
38    });
39 });
40 
```

Below the code editor, the status bar shows 'Ln 1, Col 1 Spaces: 2 UTF-8 CRLF () JavaScript Go Live'. The bottom navigation bar includes icons for file operations like Open, Save, and Close, as well as other application icons.

This screenshot shows the same 'enfermero.js' file in Visual Studio Code, but with more of the code visible. It includes the modal registration logic and additional validation for empty fields before performing an AJAX POST request.

```
1 ...
2 ...
3 ...
4 ...
5 ...
6 ...
7 ...
8 ...
9 ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 ...
19 ...
20 ...
21 ...
22 ...
23 ...
24 ...
25 ...
26 ...
27 ...
28 ...
29 ...
30 ...
31 ...
32 ...
33 ...
34 ...
35 ...
36 ...
37 ...
38 ...
39 ...
40 ...
41 ...
42 ...
43 ...
44 ...
45 ...
46 ...
47 ...
48 ...
49 ...
50 ...
51 ...
52 ...
53 ...
54 ...
55 ...
56 ...
57 ...
58 ...
59 ...
60 ...
61 ...
62 ...
63 ...
64 ...
65 ...
66 ...
67 ...
68 ...
69 ...
70 ...
71 ...
72 ...
73 ...
74 ...
75 ...
76 ...
77 ...
78 ...
79 
```

The status bar at the bottom indicates 'Ln 1, Col 1 Spaces: 2 UTF-8 CRLF () JavaScript Go Live'. The bottom navigation bar is identical to the first screenshot.

CONTROLADOR ESPECIALIDAD

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** controlador_especialidad_registro.php - PROASILo - Visual Studio Code.
- Explorer:** Shows the project structure under 'PROASILo'. The 'controller' folder contains files like index.php, mantenimiento_usuario.php, usuario.js, modeloConexion.php, enfermero.js, and controlador_especialidad_registro.php. The 'controlador_especialidad_registro.php' file is currently selected.
- Editor:** Displays the PHP code for 'controlador_especialidad_registro.php'. The code includes imports for 'Modelo_Especialidad' and 'Modelo_Conexion', and logic for registering a specialization.
- Bottom Status Bar:** Shows file path (HistorialGuatemala > controller > especialidad > controlador_especialidad_registro.php), line 1, column 1, spaces: 4, encoding: UTF-8, CRLF, PHP, and Go Live button.
- Taskbar:** Includes icons for File Explorer, Search, Task List, and others.
- System Tray:** Shows weather (26°C), battery status, and date (28/10/2022).

MODELO ESPECIALIDAD

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** modelo_especialidad.php - PROASILo - Visual Studio Code.
- Explorer:** Shows the project structure under 'PROASILo'. The 'model' folder contains files like index.php, mantenimiento_usuario.php, usuario.js, modeloConexion.php, enfermero.js, and modelo_especialidad.php. The 'modelo_especialidad.php' file is currently selected.
- Editor:** Displays the PHP code for 'modelo_especialidad.php'. The code defines a 'Modelo_Especialidad' class that extends 'conexionBD'. It includes methods for listing specializations and registering new ones.
- Bottom Status Bar:** Shows file path (HistorialGuatemala > model > modelo_especialidad.php), line 1, column 1, spaces: 4, encoding: UTF-8, CRLF, PHP, and Go Live button.
- Taskbar:** Includes icons for File Explorer, Search, Task List, and others.
- System Tray:** Shows weather (26°C), battery status, and date (28/10/2022).

JS ESPECIALIDAD

The screenshot shows the Visual Studio Code interface with the 'especialidad.js' file open. The code is a JavaScript function for a DataTable named 'tbl_especialidad'. It includes logic for rendering data, handling row edit buttons, and setting up a modal for editing rows. The code is annotated with comments explaining its purpose.

```
var tbl_especialidad;
function listado_especialidad(){
    tbl_especialidad = $('#tabla_especialidad').DataTable({
        "ordering":false,
        "bLengthChange":true,
        "searching": { "regex": false },
        "lengthMenu": [ [10, 25, 50, 100, -1], [10, 25, 50, 100, "All"] ],
        "pageLength": 10,
        "destroy":true,
        "async": false ,
        "processing": true,
        "ajax":{
            "url": "../controller/especialidad/controlador_especialidad_listar.php",
            "type": "POST"
        },
        "columns":[
            {"defaultContent":""},
            {"data": "especialidad_nombre"},
            {"data": "especialidad_fregistro"},
            {"data": "especialidad_estatus",
                render: function(data,type,row){
                    if(data=='ACTIVO'){
                        return '<span class="badge bg-success">ACTIVO</span>';
                    }else{
                        return '<span class="badge bg-danger">INACTIVO</span>';
                    }
                }
            },
            {"defaultContent": "<button class='editar btn btn-primary btn-sm'><i class='fa fa-edit'></i></button>"}
        ],
        "fnRowCallback": function( nRow, aData, iDisplayIndex, iDisplayIndexFull ) {
            $(nRow).find("td")[2]).css('text-align', 'center' );
            $(nRow).find("td")[3]).css('text-align', 'center' );
            $(nRow).find("td")[4]).css('text-align', 'center' );
        },
        "language":idioma_espanol,
        select: true
    });
    tbl_especialidad.on('draw.dt', function() {
        ...
    });
}
```

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF ⓘ JavaScript ⓘ Go Live ⓘ 0:23 26°C Parc. nublado ⓘ 28/10/2022 ⓘ

The screenshot shows the continuation of the 'especialidad.js' file in Visual Studio Code. The code adds event listeners for table rows and columns, handles modal opening for editing, and implements a registration function for new specializations. It uses jQuery and Bootstrap modal components.

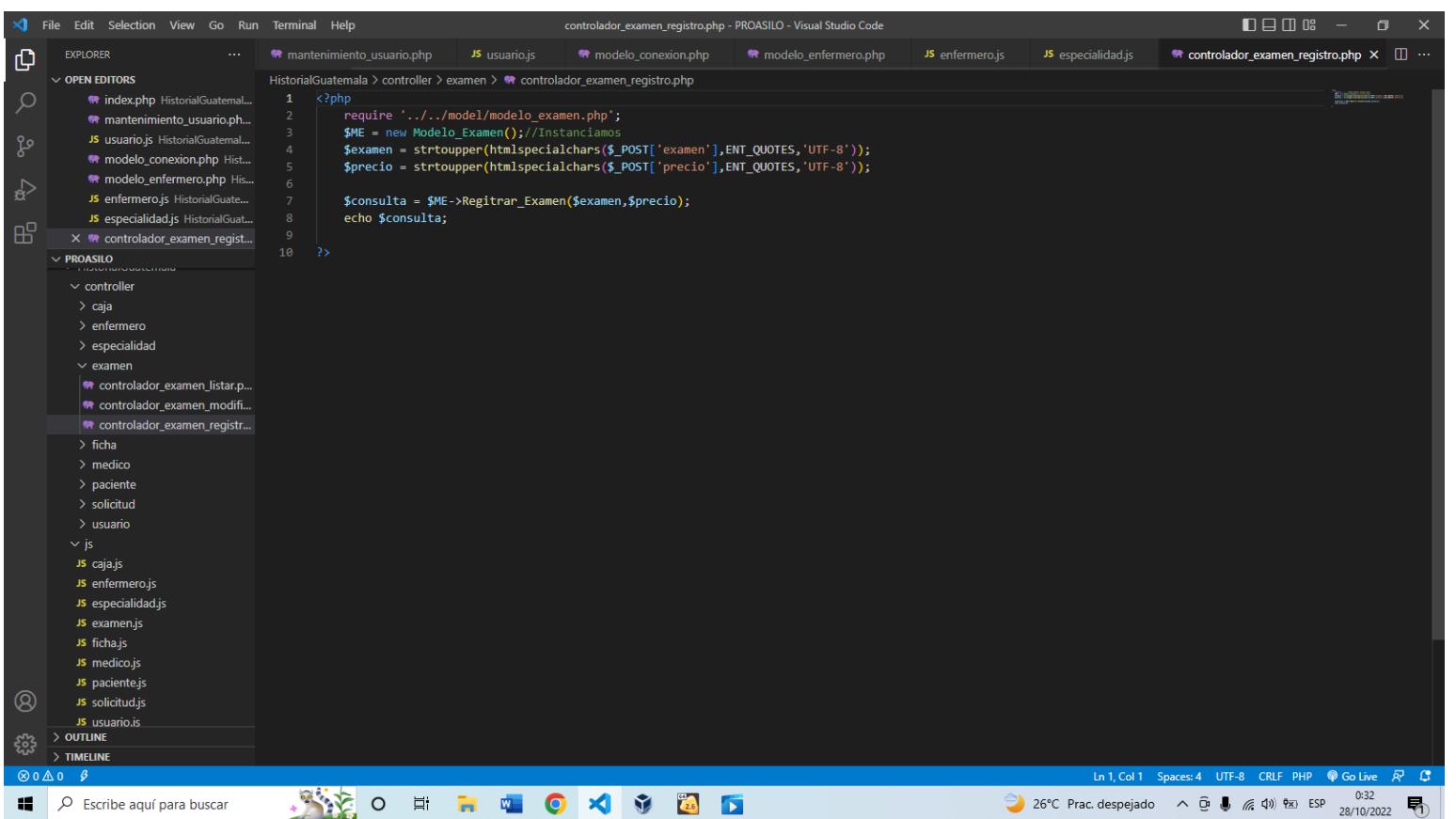
```
tbl_especialidad.on('click','.editar',function(){
    var data = tbl_especialidad.row($(this).parents('tr')).data(); //En tamaño escritorio
    if(tbl_especialidad.row(this).child.isShown()){
        var data = tbl_especialidad.row(this).data();
    } //Permite llevar los datos cuando es tamaño celular y usas el responsive de datatable
    $("#modal_editar_especialidad").modal('show');
    document.getElementById('id especialidad').value=data.especialidad_id;
    document.getElementById('txt_especialidad_editar').value=data.especialidad_nombre;
    $("#select_estatus").select2().val(data.especialidad_estatus).trigger('change.select2');
})

function AbrirModalRegistroEspecialidad(){
    $("#modal_registro_especialidad").modal({backdrop:'static',keyboard:false})
    $("#modal_registro_especialidad").modal('show');
}

function Registrar_Especialidad(){
    let esp = document.getElementById('txt_especialidad').value;
    if(esp.length==0){
        return Swal.fire("Mensaje de Advertencia","Tiene algunos campos vacios","warning");
    }
    $.ajax({
        url: '../controller/especialidad/controlador_especialidad_registro.php',
        type: 'POST',
        data: {
            esp:esp
        }
    }).done(function(resp){
        ...
    })
}
```

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF ⓘ JavaScript ⓘ Go Live ⓘ 0:27 26°C Prac. despejado ⓘ 28/10/2022 ⓘ

CONTROLADOR EXAMEN

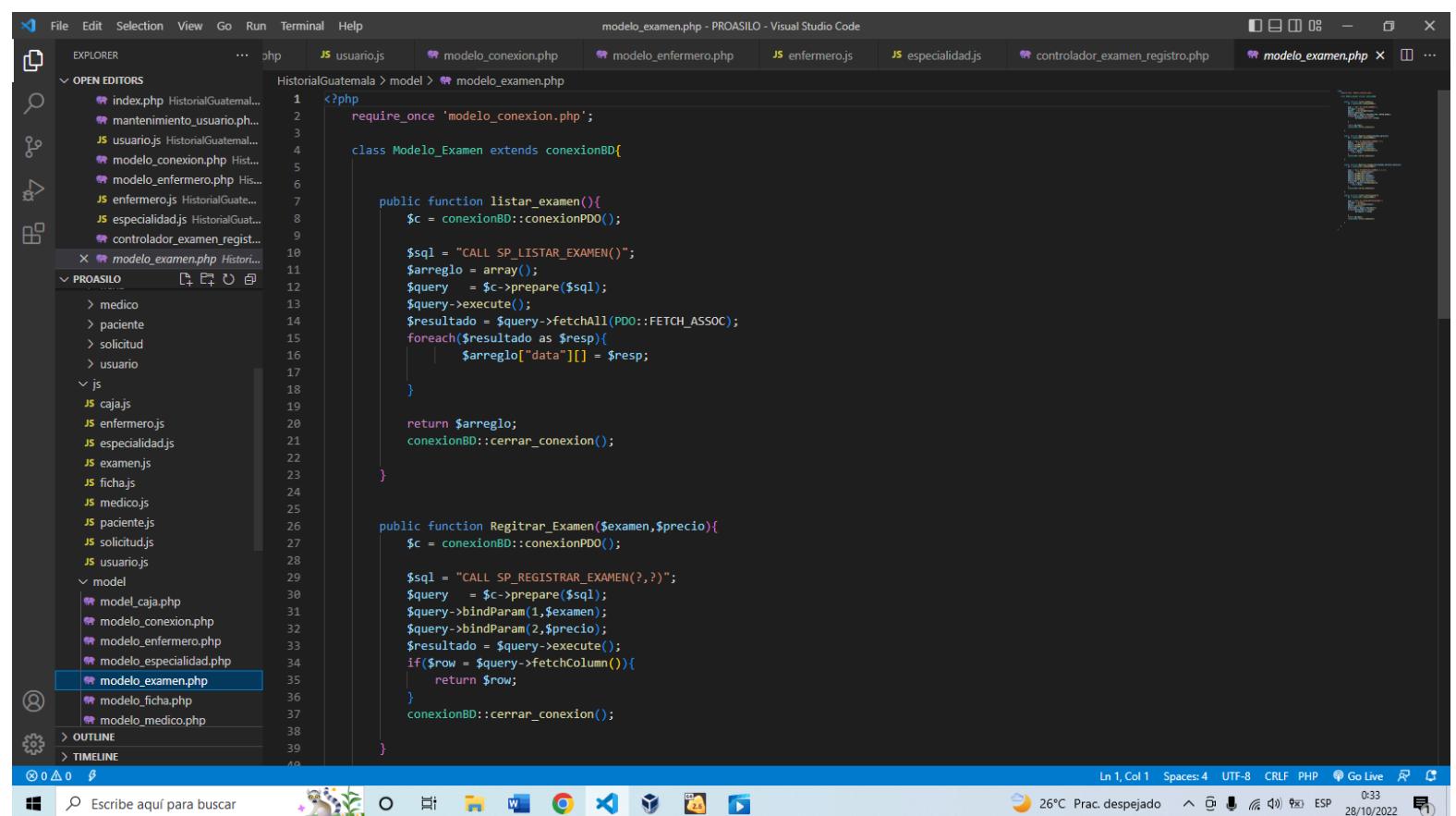


```
<?php
require '../../model/modelo_exam.php';
$ME = new Modelo_Examen(); //Instanciamos
$examen = strtoupper(htmlspecialchars($_POST['examen'],ENT_QUOTES,'UTF-8'));
$precio = strtoupper(htmlspecialchars($_POST['precio'],ENT_QUOTES,'UTF-8'));

$consulta = $ME->Registrar_Examen($examen,$precio);
echo $consulta;
```

The screenshot shows the Visual Studio Code interface with the 'controlador_examen_registro.php' file open. The code registers an exam with a specified price. The file is located in the 'HistoriaGuatemala > controller > examen' directory. The code uses the 'strtoupper' function to handle special characters and the 'htmlspecialchars' function to prevent SQL injection. It then calls the 'Registrar_Examen' method of the 'Modelo_Examen' class.

MODELO EXAMEN



```
<?php
require_once 'modelo_conexion.php';

class Modelo_Examen extends conexionBD{

    public function listar_examenes(){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_LISTAR_EXAMEN()";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->execute();
        $resultado = $query->fetchAll(PDO::FETCH_ASSOC);
        foreach($resultado as $resp){
            $arreglo["data"][] = $resp;
        }

        return $arreglo;
        conexionBD::cerrar_conexion();
    }

    public function Registrar_Examen($examen,$precio){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_REGISTRAR_EXAMEN(?,?)";
        $query = $c->prepare($sql);
        $query->bindParam(1,$examen);
        $query->bindParam(2,$precio);
        $resultado = $query->execute();
        if($row = $query->fetchColumn()){
            return $row;
        }
        conexionBD::cerrar_conexion();
    }
}
```

The screenshot shows the Visual Studio Code interface with the 'modelo_exam.php' file open. This file contains two main functions: 'listar_examenes' and 'Registrar_Examen'. The 'listar_examenes' function retrieves all exams from the database using a stored procedure. The 'Registrar_Examen' function inserts a new exam into the database using another stored procedure. Both functions use the 'conexionBD' class to manage the database connection.

JS EXAMEN

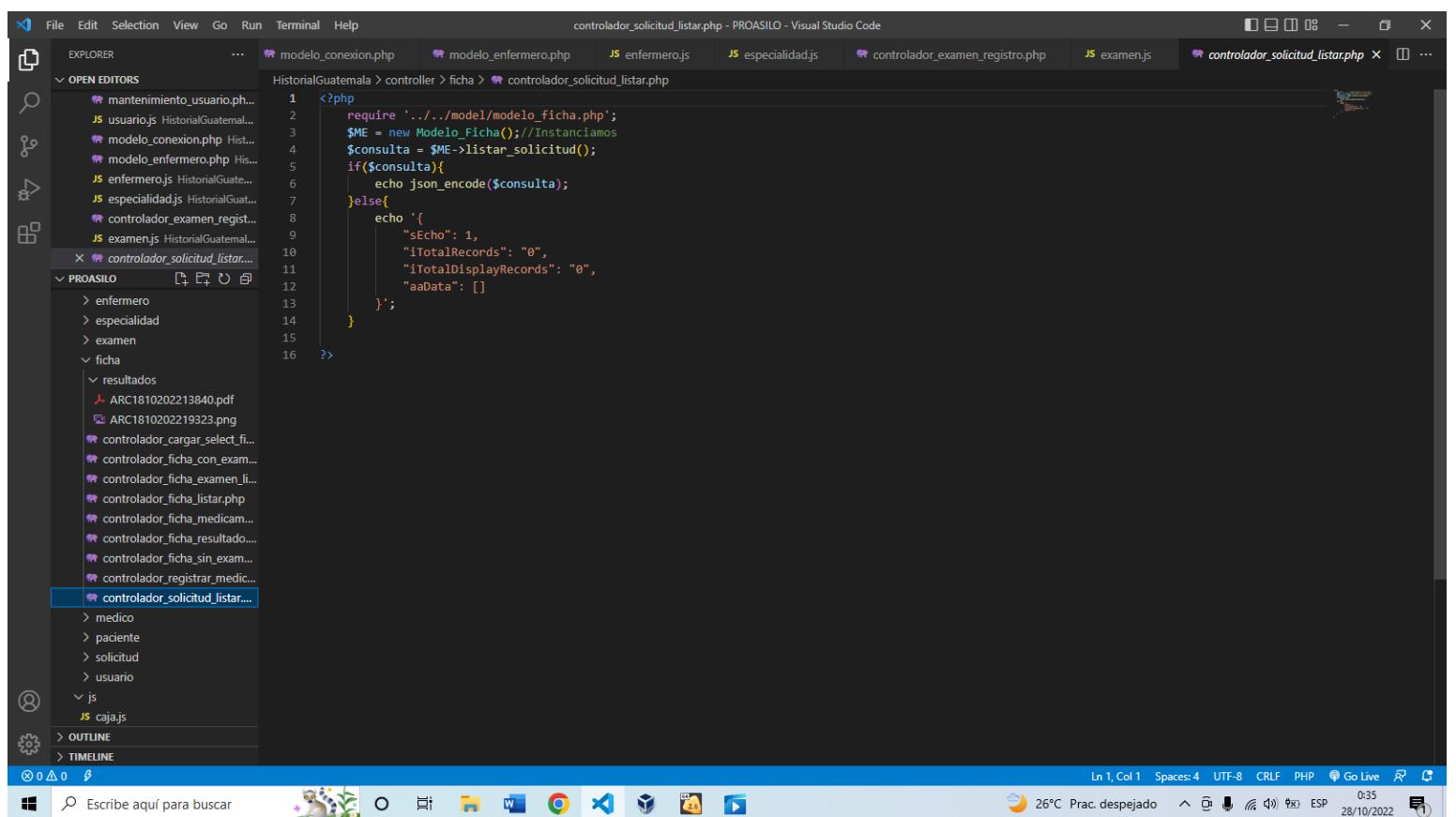
The screenshot shows the Visual Studio Code interface with the 'examen.js' file open. The code is a DataTable configuration for listing exams. It includes columns for examen_nombre, examen_precio, examen_fregistro, and examen_estatus, with a custom render function for the latter. The file also contains a fnRowCallback function to style the table rows and a language setting for Spanish.

```
HistorialGuatemala > js > JS examen.js > ...
1 var tbl_examen;
2 
3 function listar_examen(){
4     tbl_examen = $("#tabla_examen").DataTable({
5         "ordering":false,
6         "bLengthChange":true,
7         "searching": { "regex": false },
8         "lengthMenu": [ [10, 25, 50, 100, -1], [10, 25, 50, 100, "All"] ],
9         "pageLength": 10,
10        "destroy":true,
11        "async": false ,
12        "processing": true,
13        "ajax":{
14            "url": "../controller/examen/controlador_examen_listar.php",
15            "type': 'POST'
16        },
17        "columns":[
18            {"defaultContent":""},
19            {"data":"examen_nombre"},
20            {"data":"examen_precio"},
21            {"data":"examen_fregistro"},
22            {"data":"examen_estatus",
23                render: function(data,type,row){
24                    if(data=='ACTIVO'){
25                        return '<span class="badge bg-success">ACTIVO</span>';
26                    }else{
27                        return '<span class="badge bg-danger">INACTIVO</span>';
28                    }
29                },
30                {"defaultContent":<button class='editar btn btn-primary btn-sm'><i class='fa fa-edit'></i></button>"}
31            ],
32            "fnRowCallback": function( nRow, aData, iDisplayIndex, iDisplayIndexFull ) {
33                $(nRow).find("td")[3]).css('text-align', 'center' );
34                $(nRow).find("td")[4]).css('text-align', 'center' );
35                $(nRow).find("td")[5]).css('text-align', 'center' );
36                $(nRow).find("td")[2]).css('text-align', 'center' );
37            },
38            "language":idioma_espanol,
39            select: true
40        }
41    });
42   tbl_examen.on('draw.dt',function(){
43        var pageInfo = $("#tabla_examen").DataTable().page.info();
44        tbl_examen.column(0, {page: 'current'}).nodes().each(function(cell, i){
45            cell.innerHTML = i + 1 + pageInfo.start;
46        });
47    });
48
49 $('#tabla_examen').on('click','.editar',function(){
50     var data = tbl_examen.row($(this).parents('tr')).data(); //En tamaño escritorio
51     if(tbl_examen.row(this).child.isShown()){
52         var data = tbl_examen.row(this).data();
53     } //Permite llevar los datos cuando es tamaño celular y usas el responsive de datatable
54     $('#modal_editar_examen').modal('show');
55     document.getElementById('idexamen').value=data.examen_id;
56     document.getElementById('txt_examen_editar').value=data.examen_nombre;
57     document.getElementById('txt_precio_editar').value=data.examen_precio;
58     $('#select_estatus').select2().val(data.examen_estatus).trigger('change.select2');
59 })
60
61 function AbrirModalRegistroExamen(){
62     $('#modal_registro_examen').modal({backdrop:'static',keyboard:false})
63     $('#modal_registro_examen').modal('show');
64 }
65
66 function Registrar_Examen(){
67     let examen = document.getElementById('txt_examen').value;
68     let precio = document.getElementById('txt_precio').value;
69     if(examen.length==0 || precio.length==0){
70         return Swal.fire("Mensaje de Advertencia","Tiene algunos campos vacios","warning");
71     }
72     $.ajax({
73         url:'../controller/examen/controlador_examen_registro.php',
74         type:'POST',
75         data:{
```

This screenshot shows the same 'examen.js' file in Visual Studio Code, but with different code content. It includes functions for opening a registration modal and registering an exam. The modal is styled with 'static' backdrop and 'show' methods. The registration function uses AJAX to post data to a controller.

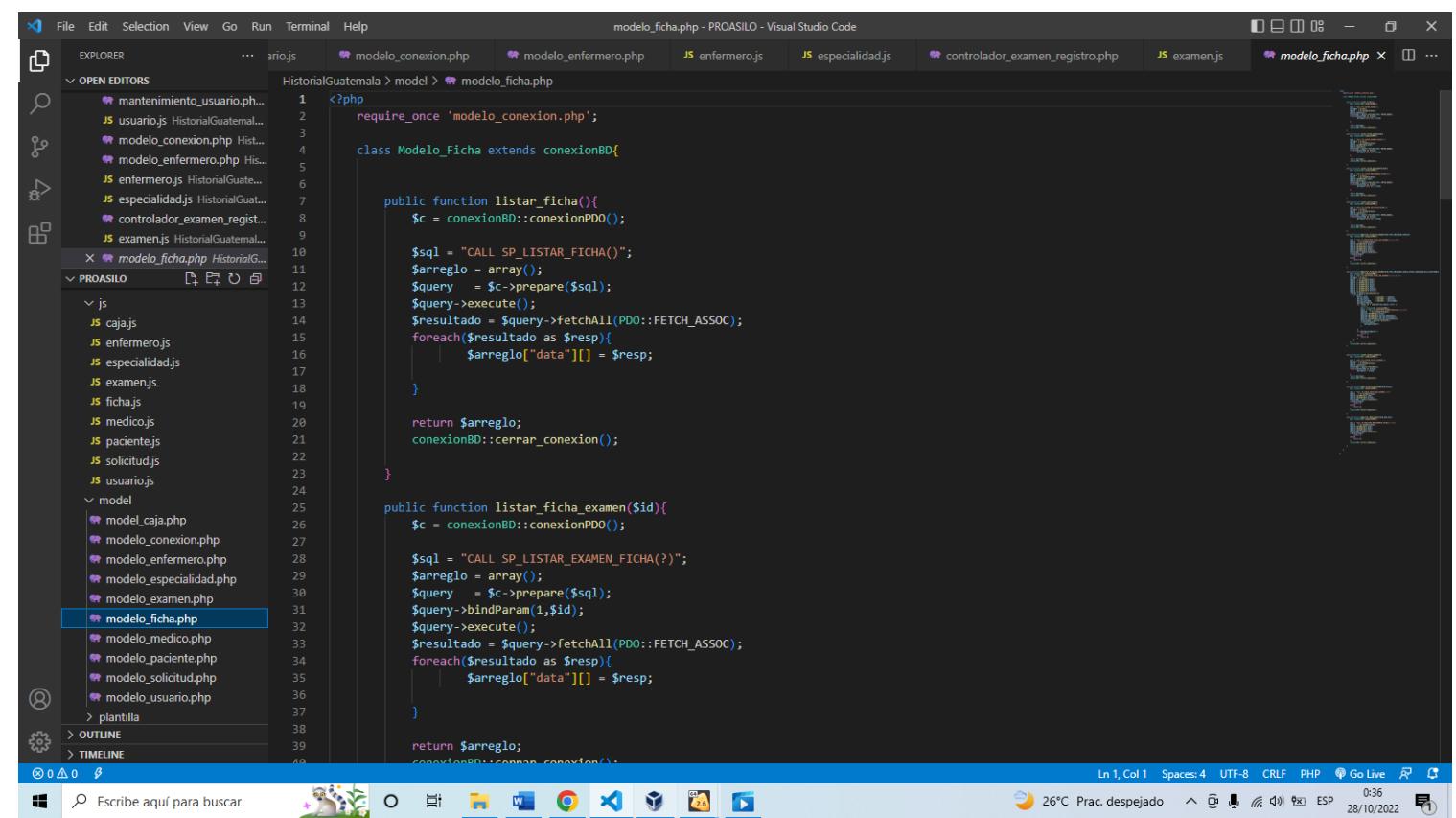
```
HistorialGuatemala > js > JS examen.js > ...
38     "language":idioma_espanol,
39     select: true
40 );
41 tbl_examen.on('draw.dt',function(){
42     var pageInfo = $("#tabla_examen").DataTable().page.info();
43     tbl_examen.column(0, {page: 'current'}).nodes().each(function(cell, i){
44         cell.innerHTML = i + 1 + pageInfo.start;
45     });
46 });
47
48 $('#tabla_examen').on('click','.editar',function(){
49     var data = tbl_examen.row($(this).parents('tr')).data(); //En tamaño escritorio
50     if(tbl_examen.row(this).child.isShown()){
51         var data = tbl_examen.row(this).data();
52     } //Permite llevar los datos cuando es tamaño celular y usas el responsive de datatable
53     $('#modal_editar_examen').modal('show');
54     document.getElementById('idexamen').value=data.examen_id;
55     document.getElementById('txt_examen_editar').value=data.examen_nombre;
56     document.getElementById('txt_precio_editar').value=data.examen_precio;
57     $('#select_estatus').select2().val(data.examen_estatus).trigger('change.select2');
58 })
59
60 function AbrirModalRegistroExamen(){
61     $('#modal_registro_examen').modal({backdrop:'static',keyboard:false})
62     $('#modal_registro_examen').modal('show');
63 }
64
65 function Registrar_Examen(){
66     let examen = document.getElementById('txt_examen').value;
67     let precio = document.getElementById('txt_precio').value;
68     if(examen.length==0 || precio.length==0){
69         return Swal.fire("Mensaje de Advertencia","Tiene algunos campos vacios","warning");
70     }
71     $.ajax({
72         url:'../controller/examen/controlador_examen_registro.php',
73         type:'POST',
74         data:{
```

CONTROLADOR FICHA MEDICA



```
<?php
require '../model/modelo_ficha.php';
$ME = new Modelo_Ficha(); //Instanciamos
$consulta = $ME->listar_solicitud();
if($consulta){
    echo json_encode($consulta);
} else {
    echo [
        "sEcho": 1,
        "iTotalRecords": "0",
        "iTotalDisplayRecords": "0",
        "aaData": []
    ];
}
```

MODELO FICHA MEDICA



```
<?php
require_once 'modelo_conexion.php';

class Modelo_Ficha extends conexionBD{

    public function listar_ficha(){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_LISTAR_FICHA();";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->execute();
        $resultado = $query->fetchAll(PDO::FETCH_ASSOC);
        foreach($resultado as $resp){
            $arreglo["data"][] = $resp;
        }

        return $arreglo;
        conexionBD::cerrarConexion();
    }

    public function listar_ficha_examen($id){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_LISTAR_EXAMEN_FICHA(?);";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->bindParam(1,$id);
        $query->execute();
        $resultado = $query->fetchAll(PDO::FETCH_ASSOC);
        foreach($resultado as $resp){
            $arreglo["data"][] = $resp;
        }

        return $arreglo;
        conexionBD::cerrarConexion();
    }
}
```

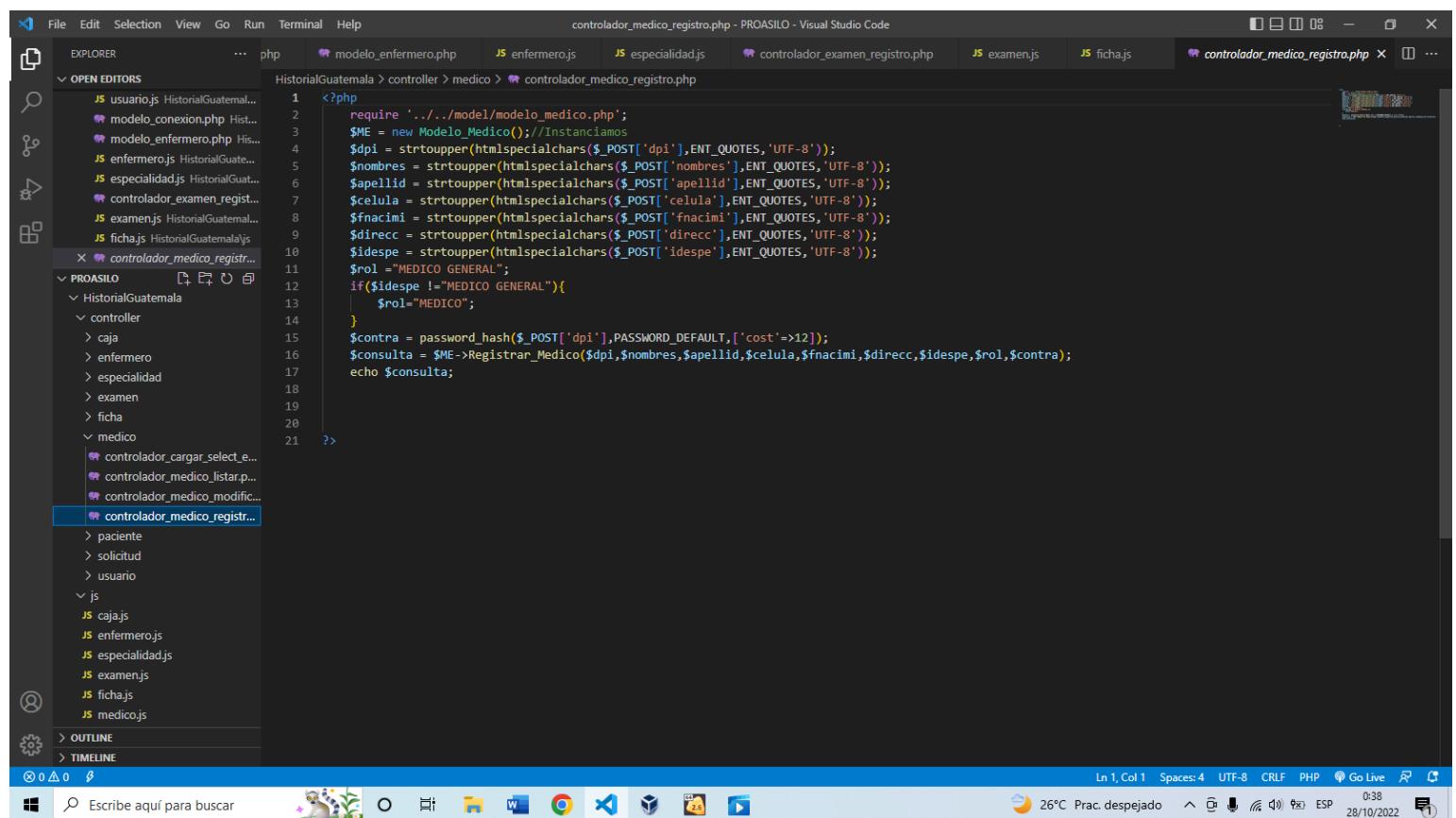
JS FICHA MEDICA

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** ficha.js - PROASIRO - Visual Studio Code
- File Explorer:** Shows the project structure under 'PROASIRO'. The 'ficha.js' file is selected.
- Code Editor:** Displays the JavaScript code for the 'ficha.js' file. The code uses jQuery and DataTables to handle a table named '#tabla_ficha'. It includes functions for listing records and handling row click events to open modal dialogs for assigning medications and viewing results.
- Bottom Status Bar:** Shows file path (HistorialGuatemala > js > ficha.js), line 1, column 1, spaces 2, UTF-8, CRLF, and the current file is JavaScript.
- Taskbar:** Includes icons for Windows, search, file operations, and other recent files.
- System Tray:** Shows weather (26°C), battery level, and system status.

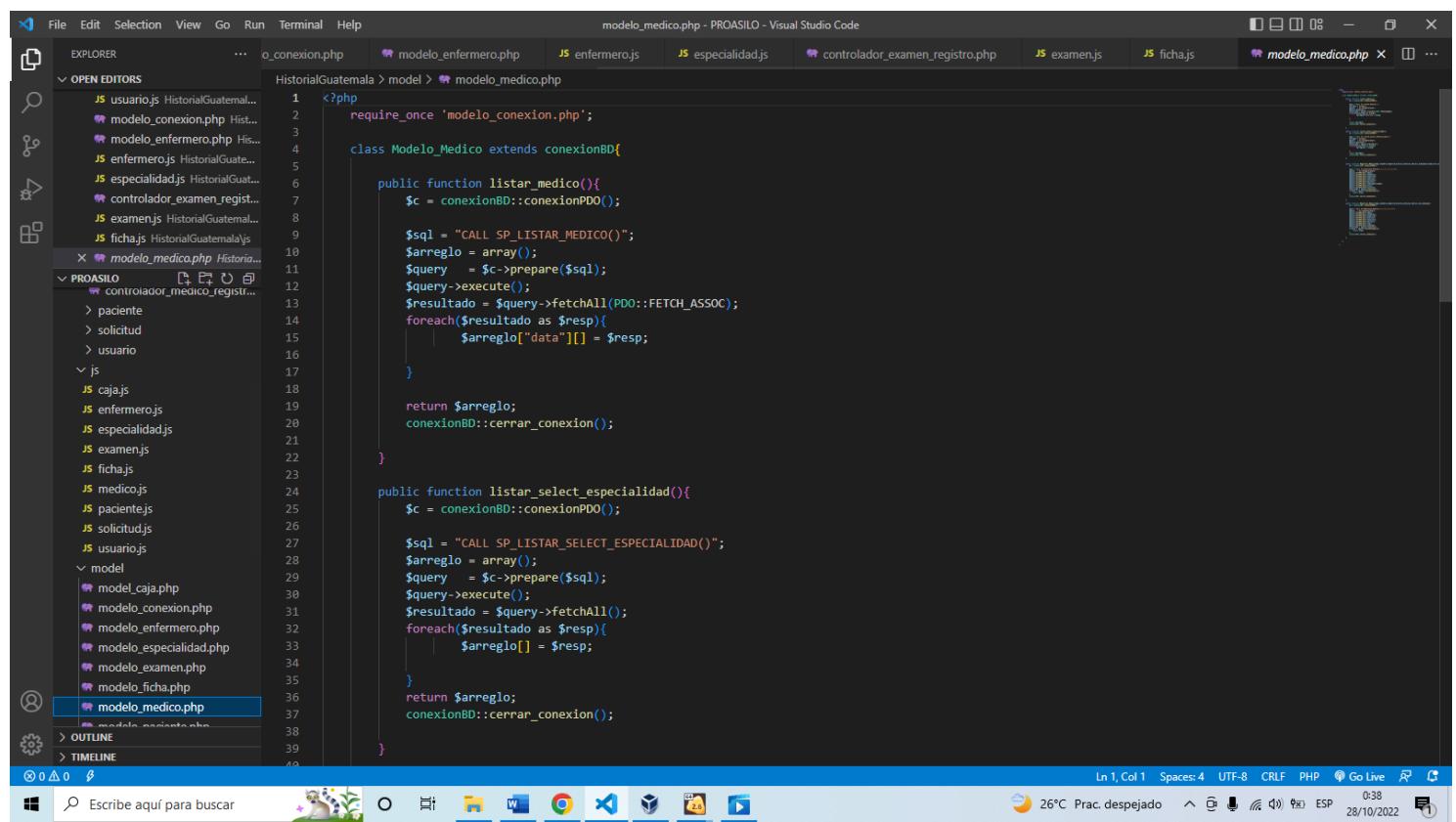
This screenshot is identical to the one above, showing the 'ficha.js' file in the Visual Studio Code interface. The code editor displays the same JavaScript logic for managing patient records and their associated exams and medications.

CONTROLADOR MEDICO



```
<?php
require '../model/modelo_medico.php';
$ME = new Modelo_Medico(); //Instanciamos
$dpi = strtoupper(htmlspecialchars($_POST['dpi'], ENT_QUOTES, 'UTF-8'));
$nombres = strtoupper(htmlspecialchars($_POST['nombres'], ENT_QUOTES, 'UTF-8'));
$apellido = strtoupper(htmlspecialchars($_POST['apellido'], ENT_QUOTES, 'UTF-8'));
$celula = strtoupper(htmlspecialchars($_POST['celula'], ENT_QUOTES, 'UTF-8'));
$fnacimi = strtoupper(htmlspecialchars($_POST['fnacimi'], ENT_QUOTES, 'UTF-8'));
$direcc = strtoupper(htmlspecialchars($_POST['direcc'], ENT_QUOTES, 'UTF-8'));
$idespe = strtoupper(htmlspecialchars($_POST['idespe'], ENT_QUOTES, 'UTF-8'));
$rol = "MEDICO GENERAL";
if($idespe != "MEDICO GENERAL"){
    $rol = "MEDICO";
}
$contra = password_hash($_POST['dpi'], PASSWORD_DEFAULT, ['cost'=>12]);
$consulta = $ME->Registrar_Medico($dpi, $nombres, $apellido, $celula, $fnacimi, $direcc, $idespe, $rol, $contra);
echo $consulta;
```

MODELO MEDICO



```
<?php
require_once 'modelo_conexion.php';

class Modelo_Medico extends conexionBD{

    public function listar_medico(){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_LISTAR_MEDICO();";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->execute();
        $resultado = $query->fetchAll(PDO::FETCH_ASSOC);
        foreach($resultado as $resp){
            $arreglo["data"][] = $resp;
        }

        return $arreglo;
        conexionBD::cerrarConexion();
    }

    public function listar_select_especialidad(){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_LISTAR_SELECT_ESPECIALIDAD();";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->execute();
        $resultado = $query->fetchAll();
        foreach($resultado as $resp){
            $arreglo[] = $resp;
        }

        return $arreglo;
        conexionBD::cerrarConexion();
    }
}
```

JS MEDICO

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "OPEN EDITORS".
 - HistorialGuatemala (JavaScript files): usuario.js, modelo_conexion.php, enfermero.js, especialidad.js, controlador_examen_registro.php, examen.js, fichajs, medico.js.
 - PROASILo (PHP files): paciente.php, solicitud.php, usuario.php, model (model_caja.php, modelo_conexion.php, modelo_enfermero.php, modelo_especialidad.php, modelo_exam.php, modelo_ficha.php, modelo_medico.php), paciente.php, solicitud.php, usuario.php.
- Code Editor:** The active file is "medico.js" located in the PROASILo folder. The code implements a DataTable for listing doctors, handling AJAX requests, and setting up language switching for Spanish.
- Bottom Status Bar:** Shows the current file path as "medico.js - PROASILo - Visual Studio Code", along with other status indicators like "JavaScript" and "Go Live".

This screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "OPEN EDITORS".
- Code Editor:** The active file is "medico.js" located in the PROASILo folder. The code continues from the previous screenshot, including logic for handling clicks on the table and opening modal dialogs for editing and registration.
- Bottom Status Bar:** Shows the current file path as "medico.js - PROASILo - Visual Studio Code", along with other status indicators like "JavaScript" and "Go Live".

CONTROLADOR PACIENTE

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "PROASILo". The "paciente" folder is expanded, showing files like "controlador_listar_examen_p...", "controlador_modificar_pacie...", and "controlador_paciente_listar...". The "controlador_registrar_pacien..." file is currently selected.
- Code Editor:** Displays the PHP code for "controlador_registrar_paciente.php". The code includes imports, variable declarations, and a main processing block. It uses strToUpper() for sanitizing POST variables and a Model class to register a patient.
- Bottom Bar:** Includes a search bar, a decorative owl icon, and various system icons.
- Status Bar:** Shows the file path ("controlador_registrar_paciente.php - PROASILo - Visual Studio Code"), line 1, column 1, spaces: 4, encoding: UTF-8, and date/time (28/10/2022 0:42).

MODELO PACIENTE

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "PROASILo". The "model" folder is expanded, showing files like "conexion.php", "model_caja.php", and "modelo_paciente.php". The "modelo_paciente.php" file is currently selected.
- Code Editor:** Displays the PHP code for "modelo_paciente.php". The code defines a "Modelo_Paciente" class extending "conexionBD". It contains methods for listing patients and registering a new patient. Both methods use prepared statements and fetchAll() to handle the results.
- Bottom Bar:** Includes a search bar, a decorative owl icon, and various system icons.
- Status Bar:** Shows the file path ("modelo_paciente.php - PROASILo - Visual Studio Code"), line 1, column 1, spaces: 4, encoding: UTF-8, and date/time (28/10/2022 0:42).

JS PACIENTE

The screenshot shows the Visual Studio Code interface with the following details:

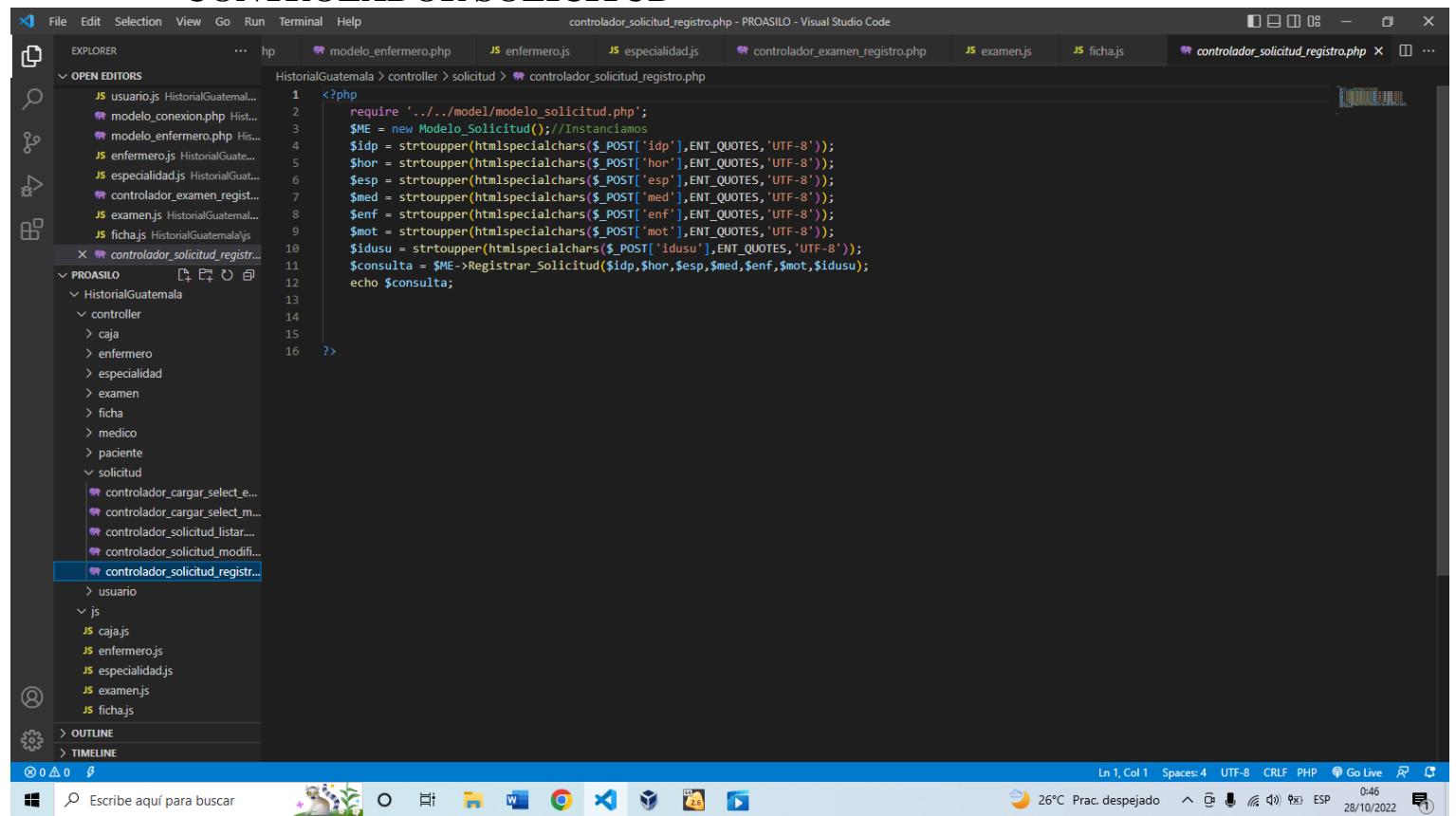
- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** Shows a tree view of files under 'PROASILo'. The 'paciente.js' file is selected and highlighted in blue.
- Editor:** The main area displays the JavaScript code for 'paciente.js'. The code includes functions for listing patients, handling patient edits, and displaying exam results.
- Bottom Bar:** Includes a search bar ('Escribe aquí para buscar'), a toolbar with various icons (File, Save, Find, etc.), and status indicators like 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'CRLF', 'JavaScript', 'Go Live', and system status (26°C, 04:43, 28/10/2022).

```
1 var tbl_paciente;
2 function listar_paciente (){
3     tbl_paciente = $("#tbl_paciente").DataTable({
4         "ordering":false,
5         "bLengthChange":true,
6         "searching": { "regex": false },
7         "lengthMenu": [ [ 10, 25, 50, 100, -1], [ 10, 25, 50, 100, "All" ] ],
8         "pageLength": 10,
9         "destroy":true,
10        "async": false ,
11        "processing": true,
12        "ajax":{
13            "url": "../controller/paciente/controlador_paciente_listar.php",
14            "type:'POST'
15        },
16        "columns":[
17            {"defaultContent":""},
18            {"data":"paciente_dpi"},
19            {"data":"pacil"},
20            {"data":"paciente_celular"}, 
21            {"data":"paciente_fnacimiento"}, 
22            {"data":"paciente_direccion"}, 
23            {"defaultContent": "<button class='editar btn btn-primary btn-sm'><i class='fa fa-edit'></i>&ampnbspEditar</button>&ampnbsp<button class='ver btn btn-primary btn-sm'><i class='fa fa-eye'></i>&ampnbspVer</button>"}
24        ],
25        "fnRowCallback": function( nRow, aData, iDisplayIndex, iDisplayIndexFull ) {
26            $(nRow).find("td")[3].css('text-align', 'center');
27            $(nRow).find("td")[4].css('text-align', 'center');
28            $(nRow).find("td")[6].css('text-align', 'center');
29        },
30        "language":idioma_espanol,
31        select: true
32    });
33    tbl_paciente.on('draw.dt',function(){
34        var PageInfo = $("#tbl_paciente").DataTable().page.info();
35        tbl_paciente.column(0, {page: 'current'}).nodes().each(function(cell, i){
36            cell.innerHTML = i + 1 + PageInfo.start;
37        });
38    });
39 },
```

This screenshot shows the same Visual Studio Code environment as the first one, but with more of the 'paciente.js' code visible at the bottom of the editor window.

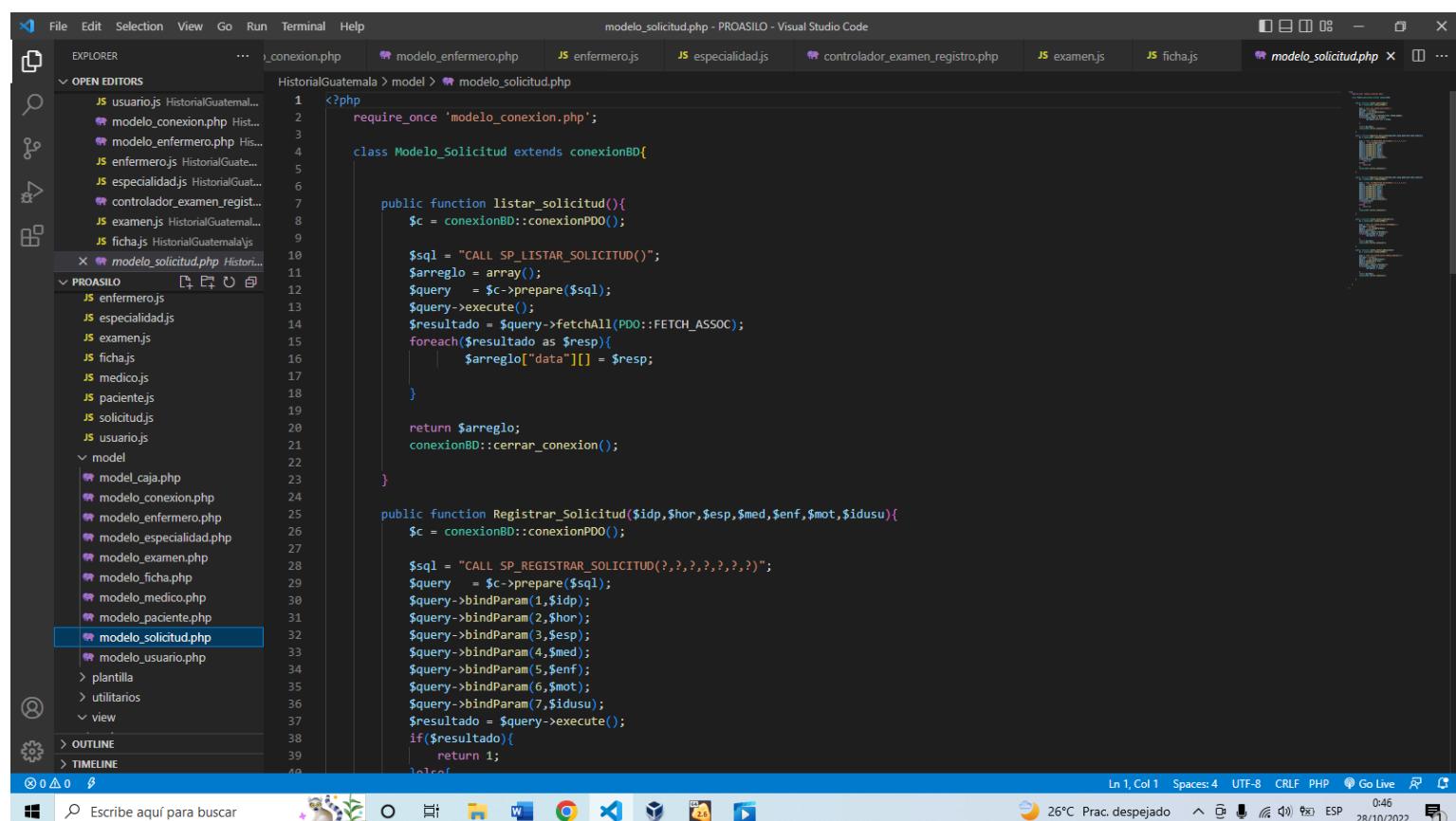
```
40 });
41 $('#tbl_paciente').on('click','.editar',function(){
42     var data = tbl_paciente.row($(this).parents('tr')).data(); //En tamaño escritorio
43     if(tbl_paciente.row(this).child.isShown()){
44         var data = tbl_paciente.row(this).data();
45     } //Permite llevar los datos cuando es tamaño celular y usas el responsive de datatable
46     $('#modal_modificar_paciente').modal('show');
47     document.getElementById('idpaciente').value=data.paciente_id;
48     document.getElementById('lbl_nombre').innerHTML=data.paci;
49     document.getElementById('txt_dpi_editar').value=data.paciente_dpi;
50     document.getElementById('txt_nombres_editar').value=data.paciente_nombre;
51     document.getElementById('txt_apellidos_editar').value=data.paciente_apellido;
52     document.getElementById('txt_celular_editar').value=data.paciente_celular;
53     document.getElementById('txt_fnacimiento_editar').value=data.paciente_fnacimiento;
54     document.getElementById('txt_direccion_editar').value=data.paciente_direccion;
55 }
56 });
57 });
58 $('#tbl_paciente').on('click','.ver',function(){
59     var data = tbl_paciente.row($(this).parents('tr')).data(); //En tamaño escritorio
60     if(tbl_paciente.row(this).child.isShown()){
61         var data = tbl_paciente.row(this).data();
62     } //Permite llevar los datos cuando es tamaño celular y usas el responsive de datatable
63     $('#modal_ver_examen').modal(' show ');
64     listar_examenes_paciente(data.paciente_id);
65 });
66 });
67 });
68 });
69 });
70 var tbl_paciente_examenes;
71 function listar_examenes_paciente(id){
72     tbl_paciente_examenes = $("#tbl_examenes_paciente").DataTable({
73         "ordering":false,
74         "bLengthChange":true,
75         "searching": { "regex": false },
76         "lengthMenu": [ [ 10, 25, 50, 100, -1], [ 10, 25, 50, 100, "All" ] ],
77         "pageLength": 10,
78         "destroy":true,
79         "async": false ,
```

CONTROLADOR SOLICITUD



```
<?php
require '../model/modelo_solicitud.php';
$ME = new Modelo_Solicitud(); //Instanciamos
$idp = strtoupper(htmlspecialchars($_POST['idp'],ENT_QUOTES,'UTF-8'));
$hor = strtoupper(htmlspecialchars($_POST['hor'],ENT_QUOTES,'UTF-8'));
$esp = strtoupper(htmlspecialchars($_POST['esp'],ENT_QUOTES,'UTF-8'));
$med = strtoupper(htmlspecialchars($_POST['med'],ENT_QUOTES,'UTF-8'));
$enf = strtoupper(htmlspecialchars($_POST['enf'],ENT_QUOTES,'UTF-8'));
$mot = strtoupper(htmlspecialchars($_POST['mot'],ENT_QUOTES,'UTF-8'));
$idusu = strtoupper(htmlspecialchars($_POST['idusu'],ENT_QUOTES,'UTF-8'));
$consulta = $ME->Registrar_Solicitud($idp,$hor,$esp,$med,$enf,$mot,$idusu);
echo $consulta;
```

MODELO SOLICITUD



```
require_once 'modelo_conexion.php';

class Modelo_Solicitud extends conexionBD{

    public function listar_solicitud(){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_LISTAR_SOLICITUD();";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->execute();
        $resultado = $query->fetchAll(PDO::FETCH_ASSOC);
        foreach($resultado as $resp){
            $arreglo["data"][] = $resp;
        }

        return $arreglo;
        conexionBD::cerrarConexion();
    }

    public function Registrar_Solicitud($idp,$hor,$esp,$med,$enf,$mot,$idusu){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_REGISTRAR_SOLICITUD(?,?,?,?,?,?,?,?)";
        $query = $c->prepare($sql);
        $query->bindParam(1,$idp);
        $query->bindParam(2,$hor);
        $query->bindParam(3,$esp);
        $query->bindParam(4,$med);
        $query->bindParam(5,$enf);
        $query->bindParam(6,$mot);
        $query->bindParam(7,$idusu);
        $resultado = $query->execute();
        if($resultado){
            return 1;
        }
    }
}
```

JS SOLICITUD

This screenshot shows the Visual Studio Code interface with the 'solicitud.js' file open in the editor. The file contains JavaScript code for a DataTable plugin, specifically for listing requests ('listar_solicitud'). The code includes handling for ordering, searching, and destroy operations, as well as an AJAX call to a controller. It also defines columns for various data fields like 'paciente', 'nombre', 'especialidad', etc., and includes a button for editing. The code is annotated with line numbers from 1 to 40.

```
1 var tbl_solicitud;
2 function listar_solicitud(){
3     tbl_solicitud = $("#tbl_solicitud").DataTable({
4         "ordering":false,
5         "bLengthChange":true,
6         "searching": { "regex": false },
7         "lengthMenu": [ [ 10, 25, 50, 100, -1], [ 10, 25, 50, 100, "All"] ],
8         "pageLength": 10,
9         "destroy":true,
10        "async": false ,
11        "processing": true,
12        "ajax":{
13            "url":"../controller/solicitud/controlador_solicitud_listar.php",
14            "type: POST"
15        },
16        "columns":[
17            {"defaultContent":""},
18            {"data":"solicitud_registro"},  
...  
40 },  
        "language":idioma_espanol,  
        select: true  
    });
   tbl_solicitud.on('draw.dt',function(){  
        var PageInfo = $('#tbl_solicitud').DataTable().page.info();  
        tbl_solicitud.column(0, {page: 'current'}).nodes().each(function(cell, i){  
            cell.innerHTML = i + 1 + ' ' + PageInfo.start;  
        });  
    });
}
```

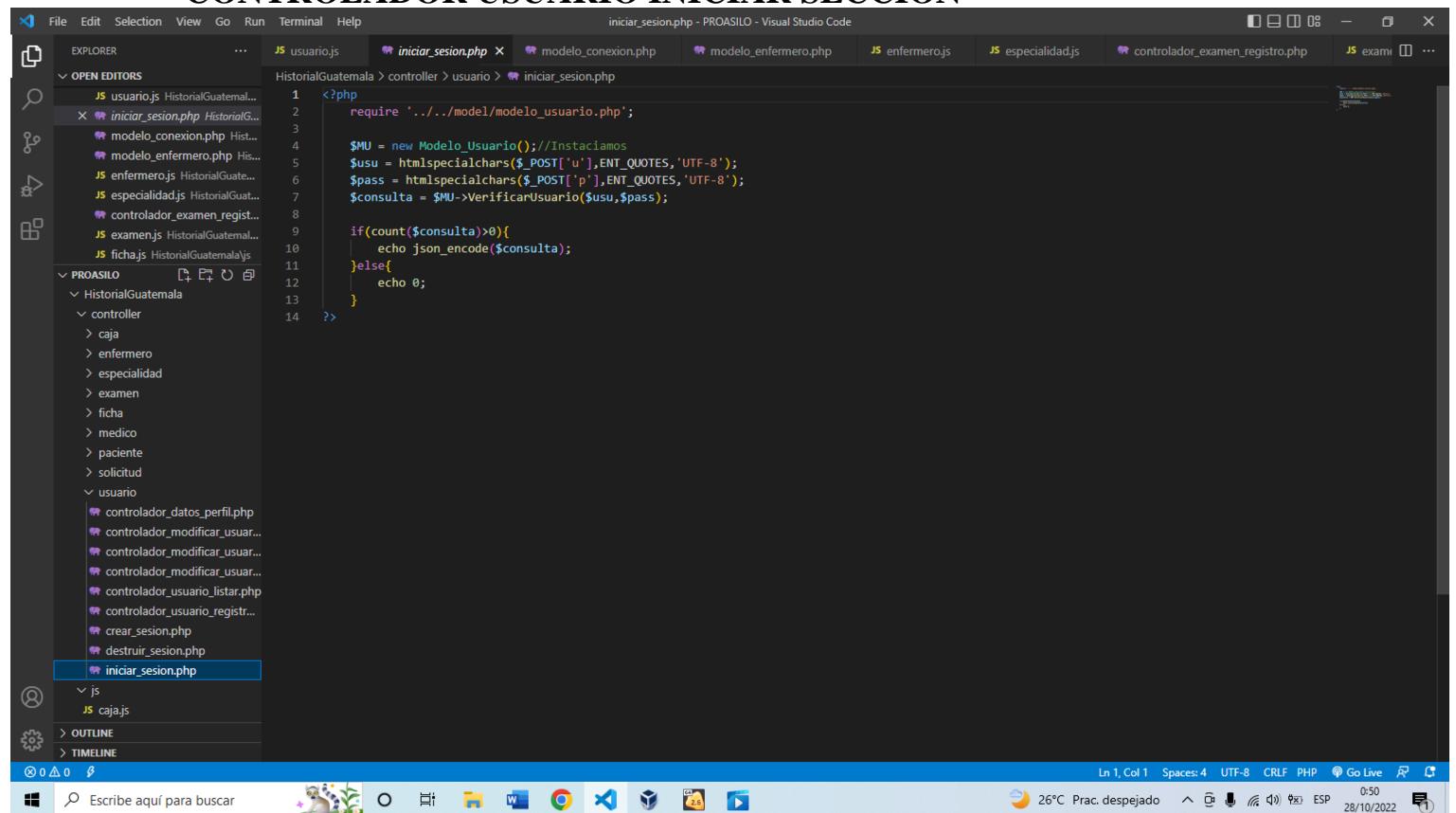
The Explorer sidebar shows other files in the project, including 'HistoriaGuatemala.js' which contains links to 'usuario.js', 'modelos.js', and 'solicitud.js'. The status bar at the bottom right shows the date as 28/10/2022 and the time as 04:47.

This screenshot shows the same 'solicitud.js' file in Visual Studio Code with a dark theme applied. The code remains the same as in the previous screenshot, containing the logic for listing and editing requests. The Explorer sidebar and status bar are also visible, showing the same project structure and system information.

```
43 }
44
45 $("#tbl_solicitud").on('click','.editar',function(){
46     var data = tbl_solicitud.row($(this).parents('tr')).data();//En tamaño escritorio
47     if(tbl_solicitud.row(this).child.isShown()){
48         var data = tbl_solicitud.row(this).data();
49     } //Permite llevar los datos cuando es tamaño celular y usas el responsive de datatable
50     $('#modal_editar').modal('show');
51     document.getElementById('txt_paciente_dpi').value=data.paciente_dpi;
52     document.getElementById('txt_paciente').value=data.paci;
53     document.getElementById('txt_horario').value=data.solicitud_horario;
54     document.getElementById('txt_motivo').value=data.solicitud_motivo;
55     $('#select_especialidad').select2().val(data.especialidad_id).trigger('change.select2');
56     cargar_select_medico_buscar_editar(data.especialidad_id,data.medico_id);
57     $('#select_enfermero').select2().val(data.enfermero_id).trigger('change.select2');
58     document.getElementById('txtid solicitud').value=data.solicitud_id;
59 }
60
61 function Registrar_Solicitud(){
62     let idp = document.getElementById('txtpacienteid').value;
63     let hor = document.getElementById('txt_horario').value;
64     let esp = document.getElementById('select_especialidad').value;
65     let med = document.getElementById('select_medico').value;
66     let enf = document.getElementById('select_enfermero').value;
67     let mot = document.getElementById('txt_motivo').value;
68     let idusu = document.getElementById('txtidprincipal').value;
69
70     if(idp.length==0){
71         return Swal.fire("Mensaje de Advertencia","Seleccione un paciente","warning");
72     }
73
74     if(esp.length==0){
75         return Swal.fire("Mensaje de Advertencia","Seleccione una especialidad","warning");
76     }
77
78     if(med.length==0){
79         return Swal.fire("Mensaje de Advertencia","Seleccione un medico","warning");
80     }
81
82     if(enf.length==0){
83 }
```

The Explorer sidebar shows the same project structure as the first screenshot. The status bar at the bottom right shows the date as 28/10/2022 and the time as 04:47.

CONTROLADOR USUARIO INICIAR SECCION

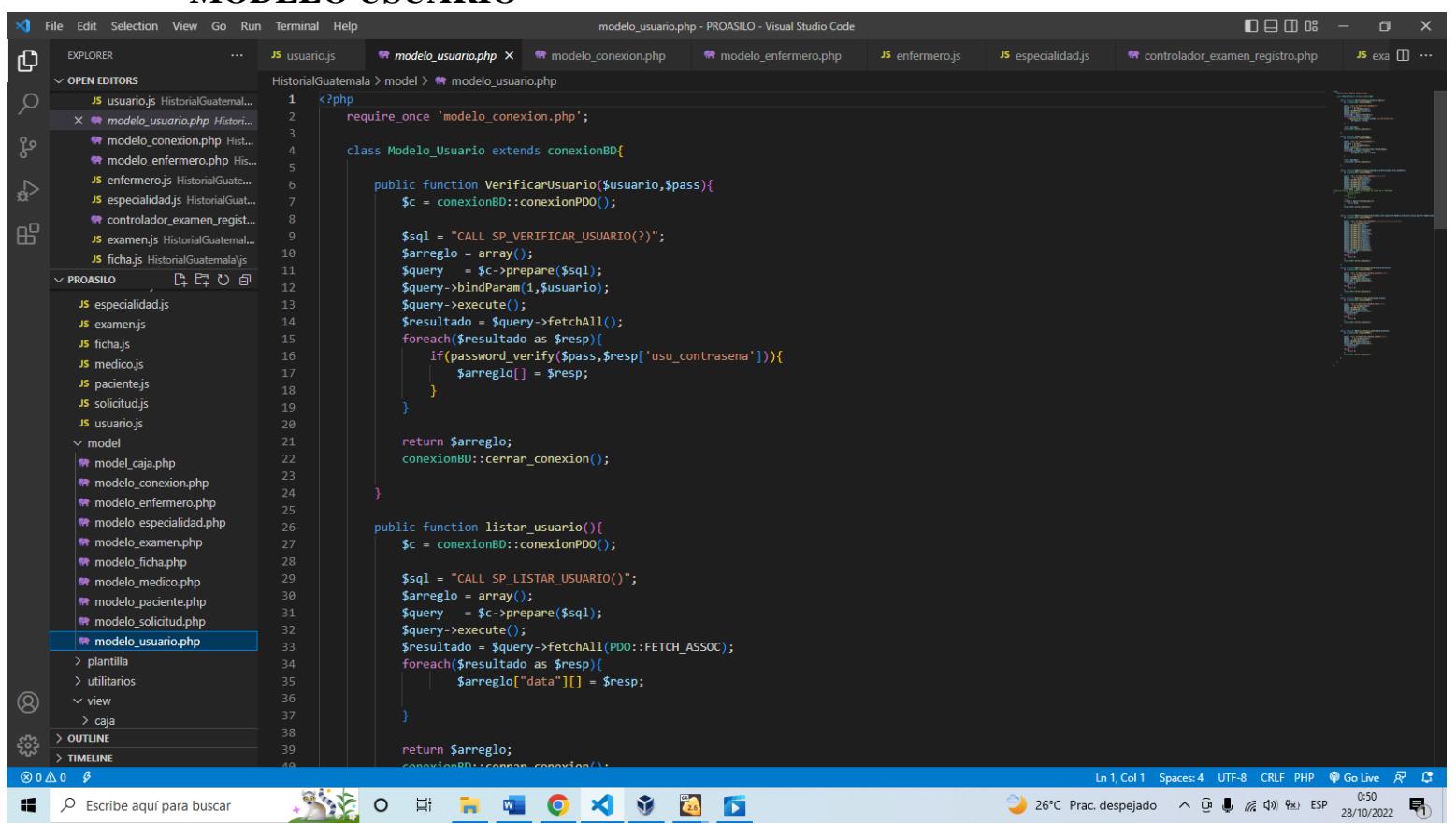


```
<?php
require '../model/modelo_usuario.php';

$MU = new Modelo_Usuario(); //Instaciamos
$usu = htmlspecialchars($_POST['u'],ENT_QUOTES,'UTF-8');
$pass = htmlspecialchars($_POST['p'],ENT_QUOTES,'UTF-8');
$consulta = $MU->VerificarUsuario($usu,$pass);

if(count($consulta)>0){
    echo json_encode($consulta);
} else{
    echo 0;
}>
```

MODELO USUARIO



```
<?php
require_once 'modelo_conexion.php';

class Modelo_Usuario extends conexionBD{

    public function VerificarUsuario($usuario,$pass){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_VERIFICAR_USUARIO?";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->bindParam(1,$usuario);
        $query->execute();
        $resultado = $query->fetchAll();
        foreach($resultado as $resp){
            if(password_verify($pass,$resp['usu_contraseña'])){
                $arreglo[] = $resp;
            }
        }

        return $arreglo;
        conexionBD::cerrarConexion();
    }

    public function listar_usuario(){
        $c = conexionBD::conexionPDO();

        $sql = "CALL SP_LISTAR_USUARIO";
        $arreglo = array();
        $query = $c->prepare($sql);
        $query->execute();
        $resultado = $query->fetchAll(PDO::FETCH_ASSOC);
        foreach($resultado as $resp{
            $arreglo["data"][] = $resp;
        }

        return $arreglo;
        conexionBD::cerrarConexion();
    }
}
```

JS USUARIO

The screenshot shows the 'usuario.js' file open in Visual Studio Code. The code is a JavaScript file that handles user login and account creation. It uses the Swal library for modal dialogs and the jQuery \$.ajax method for making POST requests to PHP controllers.

```
function Iniciar_Sesion(){
    let usu = document.getElementById('txt_usuario').value;
    let pass = document.getElementById('txt_contra').value;
    if(usu.length==0 || pass.length==0){
        return Swal.fire('Mensaje de Advertencia','Llene los campos de la sesion','warning');
    }

    $.ajax({
        url: 'controller/usuario/iniciar_sesion.php',
        type: 'POST',
        data: {
            usu,
            pass
        }
    }).done(function(resp){
        let data = JSON.parse(resp);
        if(data.length>0){
            if(data[0][4]=='INACTIVO'){
                return Swal.fire('Mensaje de Advertencia','Lo sentimos el usuario '+usu+' se encuentra desactivado, comuníquese con el administrador','error');
            }

            $.ajax({
                url: 'controller/usuario/crear_sesion.php',
                type: 'POST',
                data: {
                    idusuario:data[0][0],
                    usuario:data[0][1],
                    rol:data[0][4]
                }
            }).done(function(r){
                let timerInterval
                Swal.fire({
                    title: 'Bienvenido al sistema',
                    html: 'Sera redireccion en <b></b> milliseconds.',
                    timer: 2000,
                    heightAuto:false,
                    timerProgressBar: true,
                    allowOutsideClick: false,
                    didOpen: () => {
                        Swal.showLoading()
                        timerInterval = setInterval(() => {
                            const content = Swal.getContent()
                            if (content) {
                                const b = content.querySelector('b')
                                if (b) {
                                    b.textContent = Swal.getTimerLeft()
                                }
                            }
                        }, 100)
                    },
                    willClose: () => {
                        clearInterval(timerInterval)
                    }
                }).then((result) => {
                    /* Read more about handling dismissals below */
                    if (result.dismiss === Swal.DismissReason.timer) {
                        location.reload();
                    }
                })
            })
        }
    })
}

else{
    Swal.fire({
        icon: 'error',
        title: 'Mensaje de Advertencia',
        text: 'Usuario o contraseña incorrecta',
        heightAuto:false
    })
}
```

The code includes logic for validating user input, performing asynchronous requests to a PHP controller, and displaying confirmation or error messages using the Swal library.

This screenshot shows the continuation of the 'usuario.js' file from the previous one. It contains the completion of the 'Iniciar_Sesion' function and the start of another function, likely for account creation.

```
}).done(function(r){
    let timerInterval
    Swal.fire({
        title: 'Bienvenido al sistema',
        html: 'Sera redireccion en <b></b> milliseconds.',
        timer: 2000,
        heightAuto:false,
        timerProgressBar: true,
        allowOutsideClick: false,
        didOpen: () => {
            Swal.showLoading()
            timerInterval = setInterval(() => {
                const content = Swal.getContent()
                if (content) {
                    const b = content.querySelector('b')
                    if (b) {
                        b.textContent = Swal.getTimerLeft()
                    }
                }
            }, 100)
        },
        willClose: () => {
            clearInterval(timerInterval)
        }
    }).then((result) => {
        /* Read more about handling dismissals below */
        if (result.dismiss === Swal.DismissReason.timer) {
            location.reload();
        }
    })
})

else{
    Swal.fire({
        icon: 'error',
        title: 'Mensaje de Advertencia',
        text: 'Usuario o contraseña incorrecta',
        heightAuto:false
    })
})
```

The code continues to handle user input validation and interaction with the UI using the Swal library.