

Appendice 1 - Administracion de Header y Ayuda de Pantallas - Codigo

Este es el codigo para el header de todas las pantallas. Dentro del mismo se encuentran los scripts JavaScript que controlan el modal para despliegue de ayuda de pantalla.

```
function writePageHeader($showUser = false) {
    global $pctCompanyName, $pctLoggedInUserID, $screenName, $screenTitle;

    $user = ($showUser && !empty($pctLoggedInUserID)) ? $pctLoggedInUserID : "---";
    $company = $pctCompanyName ?? "Su Empresa";
    $screen = $screenName ?? "Desconocido";

    if ($screen === "index.php") {
        $screen = "Login";
    }

    $rowhtml = "";

    #####
    ##
    # Banner (3 columnas: Marca izquierda, Acciones centro, Informacion pantalla derecha)

    #####
    ##
    $rowhtml = "<div style='border:2px solid #2596be; background:#eaf6fb; padding:10px 12px;
margin:20px auto; width:80%; border-radius:12px;'>";
    $rowhtml .= " <div style='display:flex; justify-content:space-between; align-items:center; flex-
wrap:nowrap; gap:14px;'>";

    #####
    ##
    # COLUMNA IZQUIERDA: Marca combinada (logo + nombre sistema)

    #####
    ##
    $rowhtml .= " <div style='flex:0 0 320px; display:flex; align-items:center; justify-content:flex-
start;'>";
    $rowhtml .= " <div style='display:flex; align-items:center; gap:12px; background:#e3f2fd;
border:1px solid #cfe7f3; border-radius:12px; padding:10px 14px;'>";
    $rowhtml .= " <div style='flex:0 0 72px; height:72px; display:flex; align-items:center; justify-
content:center; overflow:hidden;'>";
    $rowhtml .= " <img src='/images/BookBrandingLogo.png' alt='Logo' style='max-
height:72px; max-width:72px; width:auto; height:auto; object-fit:contain;'>";
    $rowhtml .= " </div>";
    $rowhtml .= " <div style='flex:1 1 auto; font-size:20px; font-weight:600; color:#2596be; line-
```

```
height:1.1; white-space:nowrap;*>Sistema Corporativo de Inteligencia</div>";
$rowhtml .= "    </div>";
$rowhtml .= " </div>";
```

```
#####
##
# COLUMNA CENTRAL: 3 lineas
# 1) Home + Ayuda Pantalla + icono ayuda(?)
# 2) Buscar Manual para Ayuda + icono buscar
# 3) Chat Conmigo + icono chat
```

```
#####
##
$rowhtml .= " <div style='flex:1 1 auto; display:flex; flex-direction:column; justify-
content:center; align-items:center; gap:8px; min-width:260px; min-width:0;'>";
```

```
#####
##
# Linea Central 1: Home + Ayuda Pantalla + icono ayuda
```

```
#####
##
$rowhtml .= " <div style='display:flex; align-items:center; justify-content:center; gap:10px;
flex-wrap:nowrap;'>";
```

```
if (strtolower($screen) !== "logon" && strtolower($screenName ?? "") !== "logon") {
    $rowhtml .= "    <a href='Menu_Main.php' ";
    $rowhtml .= "        style='display:inline-block; padding:4px 12px; background-color:#2596be;
color:white; text-decoration:none; border-radius:6px; font-size:13px;'>Inicio</a>";
}
```

```
$rowhtml .= "    <span style='font-size:14px; color:#2596be; font-weight:bold;'>Ayuda
Pantalla</span>";
```

```
$rowhtml .= "    <img src='/images/questionmark.png' alt='Ayuda' title='Ayuda Pantalla' ";
$rowhtml .= "        style='cursor:pointer; width:18px; height:18px;' ";
$rowhtml .= "        onclick=\"showScreenHelp('\" . htmlspecialchars($screen, ENT_QUOTES,
'UTF-8') . \"')\">";
$rowhtml .= "    </div>";
```

```
#####
##
# Linea Central 2: Caja busqueda + icono buscar
```

```
#####
```

```

##
$rowhtml .= " <div style='display:flex; align-items:center; justify-content:center; gap:6px;
width:100%;'>";
$rowhtml .= " <input type='text' name='txt_Question' id='txt_Question'
placeholder='Buscar Ayuda en Manual' ";
$rowhtml .= " style='max-width:520px; width:75%; padding:5px 10px; border:1px solid
#ccc; border-radius:6px; font-size:14px;' ";
$rowhtml .= " onkeydown=\"if(event.key==='Enter'){submitHeaderQuestion();}\">";
$rowhtml .= " <img src='/images/Search.png' alt='Buscar' title='Buscar Ayuda' ";
$rowhtml .= " style='cursor:pointer; width:18px; height:18px; vertical-align:middle;' ";
$rowhtml .= " onclick='submitHeaderQuestion()'>";
$rowhtml .= " </div>";

```

```

#####
##

```

```

# Linea Central 3: Chat Conmigo + icono chat

```

```

#####
##

```

```

$rowhtml .= " <div style='display:flex; align-items:center; justify-content:center; gap:8px;'>";
$rowhtml .= " <a href='Chat.php' target='_blank' style='display:flex; align-items:center;
gap:8px; text-decoration:none;'>";
$rowhtml .= " <span style='font-size:14px; color:#2596be; font-weight:bold;'>Chat
Conmigo</span>";
$rowhtml .= " <img src='/images/chat.png' alt='Chat' title='Chat' ";
$rowhtml .= " style='width:26px; height:26px; vertical-align:middle;'>";
$rowhtml .= " </a>";
$rowhtml .= " </div>";
$rowhtml .= " </div>";

```

```

#####
##

```

```

# COLUMNA DERECHA: 2 lineas con inputs readonly
# 1) Nombre Pantalla
# 2) Descripcion

```

```

#####
##

```

```

$rowhtml .= " <div style='flex:0 0 340px; display:flex; flex-direction:column; justify-
content:center; gap:8px;'>";

$rowhtml .= " <div style='display:flex; align-items:center; justify-content:space-between;
gap:8px;'>";
$rowhtml .= " <div style='color:#2596be; font-size:13px; font-weight:bold; white-
space:nowrap;'>Nombre Pantalla</div>";
$rowhtml .= " <input type='text' readonly='readonly' value='' .

```

```
htmlspecialchars($screenName ?? "", ENT_QUOTES, 'UTF-8') . "" ";
$rowhtml .= " style='width:220px; padding:4px 8px; border:1px solid #ccc; border-
radius:6px; font-size:13px; background:#f7fbfe;'>";
$rowhtml .= " </div>";
```

```
$rowhtml .= " <div style='display:flex; align-items:center; justify-content:space-between;
gap:8px;'>";
```

```
$rowhtml .= " <div style='color:#2596be; font-size:13px; font-weight:bold; white-
space:nowrap;'>Descripcion</div>";
```

```
$rowhtml .= " <input type='text' readonly='readonly' value="" .
```

```
htmlspecialchars($screenTitle ?? "", ENT_QUOTES, 'UTF-8') . "" ";
```

```
$rowhtml .= " style='width:220px; padding:4px 8px; border:1px solid #ccc; border-
radius:6px; font-size:13px; background:#f7fbfe;'>";
```

```
$rowhtml .= " </div>";
```

```
$rowhtml .= " </div>";
```

```
#####
##
```

```
# Cerrar contenedores banner
```

```
#####
##
```

```
$rowhtml .= " </div>";
```

```
$rowhtml .= "</div>";
```

```
echo $rowhtml;
```

```
#####
##
```

```
# Modal ayuda + scripts
```

```
#####
##
```

```
echo <<<EOT
```

```
<script>
```

```
function submitHeaderQuestion() {
```

```
var el = document.getElementById('txt_Question');
```

```
if (!el) return;
```

```
var q = (el.value || "").trim();
```

```
if (q.length === 0) { return; }
```

```
var url = 'Search.php?Question=' + encodeURIComponent(q);
```

```
var win = window.open(url, '_blank', 'noopener,noreferrer');
```

```
if (win) { win.opener = null; }
```

```

}

function openHeaderChat() {
    var win = window.open('Chat.php', '_blank', 'noopener,noreferrer');
    if (win) { win.opener = null; }
}

function showScreenHelp(screen) {
    console.log("Pantalla solicitada:", screen);

    var modal = document.getElementById('helpModal');
    var body = document.getElementById('helpModalBody');
    if (!modal || !body) return;

    body.innerHTML = "<div style='font-size:12px;color:#666;'>Cargando...</div>";

    fetch('gethelp.php?screen=' + encodeURIComponent(screen))
        .then(function(response){ return response.JSON(); })
        .then(function(data){
            if (!data || data.status !== 'ok') {
                var msg = (data && data.message) ? data.message : 'No existe ayuda disponible para esta
pantalla.';
                body.innerHTML = escapeHtml(msg);
                modal.style.display = 'block';
                return;
            }
            }

            var mode = data.HelpMode || 'Text';
            var chapterName = data.ChapterName || '';
            var sectionName = data.SectionName || '';
            var showHeader = (data.ShowHeader === 'Y');
            var content = data.Content || '';

            var html = '';

            if (mode === 'HTML') {
                if (showHeader && (chapterName || sectionName)) {
                    html += "<div style='margin:0 0 8px 0;'><strong>" +
                        escapeHtml(chapterName) + " - " + escapeHtml(sectionName) +
                        "</strong></div>";
                }
                html += content;
            } else {
                html += escapeHtml(content).replace(/\n/g, "<br>");
            }

            body.innerHTML = html;

```

```

var imgs = body.querySelectorAll('img');
imgs.forEach(function(img) {
    img.style.maxWidth = "100%";
    img.style.height = "auto";
    img.style.cursor = "zoom-in";
    img.addEventListener('click', function() {
        var src = img.getAttribute('src');
        if (src) {
            var win = window.open(src, '_blank', 'noopener,noreferrer');
            if (win) { win.opener = null; }
        }
    });
});

modal.style.display = 'block';
})
.catch(function(error) {
    console.error("Error Fetch:", error);
    body.innerHTML = "Error cargando ayuda.";
    modal.style.display = 'block';
});
}

```

```

function closeHelpModal() {
    document.getElementById('helpModal').style.display = 'none';
}

```

```

function escapeHtml(s) {
    s = String(s | | "");
    return s
        .replace(/&/g, "&amp;")
        .replace(/</g, "&lt;")
        .replace(/>/g, "&gt;")
        .replace(/"/g, "&quot;")
        .replace(/'/g, "&#039;");
}

```

```

</script>

```

```

<div id="helpModal" style="display:none; position:fixed; z-index:9999; left:0; top:0; width:100%; height:100%; overflow:auto; background-color:rgba(0,0,0,0.4);">

```

```

    <div style="background-color:#fff; margin:10% auto; padding:20px; border:1px solid #888; width:50%; border-radius:10px;">

```

```

        <span onclick="closeHelpModal()" style="color:#aaa; float:right; font-size:28px; font-weight:bold; cursor:pointer;">&times;</span>

```

```

        <h4>Ayuda Pantalla</h4>

```

```

        <div id="helpModalBody" style="margin-top:10px;">

```

```

        </div>

```

```

    </div>

```

```
</div>  
EOT;  
}
```

Apendice 2 - Codigo de Barra de Menu Controlada por Tablas

Este es el codigo utilizado para el header de todas las pantallas. Dentro de este codigo se incluyen los scripts JavaScript que controlan el modal utilizado para mostrar la ayuda de pantalla.

```
function writePageHeader($showUser = false) {
    global $pctCompanyName, $pctLoggedOnUserID, $screenName, $screenTitle;

    $user = ($showUser && !empty($pctLoggedOnUserID)) ? $pctLoggedOnUserID : "---";
    $company = $pctCompanyName ?? "Su Empresa";
    $screen = $screenName ?? "Desconocido";

    if ($screen === "index.php") {
        $screen = "Logon";
    }

    $rowhtml = "";

    #####
    # Banner (3 columnas: Marca izquierda, Acciones centro, Informacion de pantalla derecha)
    #####
    $rowhtml = "<div style='border:2px solid #2596be; background:#eaf6fb; padding:10px 12px;
margin:20px auto; width:80%; border-radius:12px;'>";
    $rowhtml .= " <div style='display:flex; justify-content:space-between; align-items:center; flex-
wrap:nowrap; gap:14px;'>";

    #####
    # COLUMNA IZQUIERDA: identificacion combinada de marca (logo + nombre del sistema)
    #####
    $rowhtml .= " <div style='flex:0 0 320px; display:flex; align-items:center; justify-content:flex-start;'>";
    $rowhtml .= " <div style='display:flex; align-items:center; gap:12px; background:#e3f2fd;
border:1px solid #cfe7f3; border-radius:12px; padding:10px 14px;'>";
    $rowhtml .= " <div style='flex:0 0 72px; height:72px; display:flex; align-items:center; justify-
content:center; overflow:hidden;'>";
    $rowhtml .= " <img src='/images/BookBrandingLogo.png' alt='Logo' style='max-height:72px; max-
width:72px; width:auto; height:auto; object-fit:contain;'>";
    $rowhtml .= " </div>";
    $rowhtml .= " <div style='flex:1 1 auto; font-size:20px; font-weight:600; color:#2596be; line-
height:1.1; white-space:nowrap;'>Sistema Corporativo de Inteligencia</div>";
    $rowhtml .= " </div>";
    $rowhtml .= " </div>";

    #####
    # COLUMNA CENTRAL: 3 lineas
    # 1) Inicio + Ayuda de Pantalla + icono de ayuda
    # 2) Buscar Manual de Ayuda + icono de busqueda
    # 3) Chatear Conmigo + icono de chat
    #####
```

```
$rowhtml .= " <div style='flex:1 1 auto; display:flex; flex-direction:column; justify-content:center; align-items:center; gap:8px; min-width:260px; min-width:0;*>";
```

```
#####
```

```
# Linea central 1: Inicio + Ayuda de Pantalla + icono de ayuda
```

```
#####
```

```
$rowhtml .= " <div style='display:flex; align-items:center; justify-content:center; gap:10px; flex-wrap:nowrap;*>";
```

```
if (strtolower($screen) != "logon" && strtolower($screenName ?? "") != "logon") {
```

```
    $rowhtml .= " <a href='Menu_Main.php' ";
```

```
    $rowhtml .= " style='display:inline-block; padding:4px 12px; background-color:#2596be; color:white; text-decoration:none; border-radius:6px; font-size:13px;*>Inicio</a>";
```

```
}
```

```
$rowhtml .= " <span style='font-size:14px; color:#2596be; font-weight:bold;*>Ayuda de Pantalla</span>";
```

```
$rowhtml .= " <img src='/images/questionmark.png' alt='Ayuda' title='Ayuda de Pantalla' ";
```

```
$rowhtml .= " style='cursor:pointer; width:18px; height:18px; ">";
```

```
$rowhtml .= " onclick=\"showScreenHelp(' . htmlspecialchars($screen, ENT_QUOTES, 'UTF-8') . \" )\">";
```

```
$rowhtml .= " </div>";
```

```
#####
```

```
# Linea central 2: Caja de busqueda + icono de busqueda
```

```
#####
```

```
$rowhtml .= " <div style='display:flex; align-items:center; justify-content:center; gap:6px; width:100%;*>";
```

```
$rowhtml .= " <input type='text' name='txt_Question' id='txt_Question' placeholder='Buscar Manual de Ayuda' ";
```

```
$rowhtml .= " style='max-width:520px; width:75%; padding:5px 10px; border:1px solid #ccc; border-radius:6px; font-size:14px; ">";
```

```
$rowhtml .= " onkeydown=\"if(event.key==='Enter'){submitHeaderQuestion();}\">";
```

```
$rowhtml .= " <img src='/images/Search.png' alt='Buscar' title='Buscar Ayuda' ";
```

```
$rowhtml .= " style='cursor:pointer; width:18px; height:18px; vertical-align:middle; ">";
```

```
$rowhtml .= " onclick='submitHeaderQuestion()'>";
```

```
$rowhtml .= " </div>";
```

```
#####
```

```
# Linea central 3: Chatear Conmigo + icono de chat
```

```
#####
```

```
$rowhtml .= " <div style='display:flex; align-items:center; justify-content:center; gap:8px;*>";
```

```
$rowhtml .= " <a href='Chat.php' target='_blank' style='display:flex; align-items:center; gap:8px; text-decoration:none;*>";
```

```
$rowhtml .= " <span style='font-size:14px; color:#2596be; font-weight:bold;*>Chatear Conmigo</span>";
```

```
$rowhtml .= " <img src='/images/chat.png' alt='Chat' title='Chat' ";
```

```

$rowhtml .= "          style='width:26px; height:26px; vertical-align:middle;';>";
$rowhtml .= "    </a>";
$rowhtml .= "  </div>";
$rowhtml .= " </div>";

#####
# COLUMNA DERECHA: 2 lineas con campos solo lectura (sin postback)
# 1) Nombre de Pantalla
# 2) Descripcion
#####
$rowhtml .= " <div style='flex:0 0 340px; display:flex; flex-direction:column; justify-content:center;
gap:8px;';>";

    $rowhtml .= "  <div style='display:flex; align-items:center; justify-content:space-between;
gap:8px;';>";
    $rowhtml .= "    <div style='color:#2596be; font-size:13px; font-weight:bold; white-
space:nowrap;';>Nombre de Pantalla</div>";
    $rowhtml .= "      <input type='text' readonly='readonly' value='' . htmlspecialchars($screenName ??
'', ENT_QUOTES, 'UTF-8') . '' ";
    $rowhtml .= "        style='width:220px; padding:4px 8px; border:1px solid #ccc; border-radius:6px;
font-size:13px; background:#f7fbfe;';>";
    $rowhtml .= "      </div>";

    $rowhtml .= "  <div style='display:flex; align-items:center; justify-content:space-between;
gap:8px;';>";
    $rowhtml .= "    <div style='color:#2596be; font-size:13px; font-weight:bold; white-
space:nowrap;';>Descripcion</div>";
    $rowhtml .= "      <input type='text' readonly='readonly' value='' . htmlspecialchars($screenTitle ?? '',
ENT_QUOTES, 'UTF-8') . '' ";
    $rowhtml .= "        style='width:220px; padding:4px 8px; border:1px solid #ccc; border-radius:6px;
font-size:13px; background:#f7fbfe;';>";
    $rowhtml .= "      </div>";

    $rowhtml .= "  </div>";

#####
# Cerrar contenedores del banner
#####
$rowhtml .= " </div>";
$rowhtml .= " </div>";

echo $rowhtml;

#####
# Modal de ayuda + scripts
#####
echo <<<EOT
<script>

```

```

function submitHeaderQuestion() {
    var el = document.getElementById('txt_Question');
    if (!el) return;

    var q = (el.value || '').trim();
    if (q.length === 0) { return; }

    var url = 'Search.php?Question=' + encodeURIComponent(q);
    var win = window.open(url, '_blank', 'noopener,noreferrer');
    if (win) { win.opener = null; }
}

function openHeaderChat() {
    var win = window.open('Chat.php', '_blank', 'noopener,noreferrer');
    if (win) { win.opener = null; }
}

function showScreenHelp(screen) {
    console.log("Pantalla solicitada:", screen);

    var modal = document.getElementById('helpModal');
    var body = document.getElementById('helpModalBody');
    if (!modal || !body) return;

    body.innerHTML = "<div style='font-size:12px;color:#666;'>Cargando...</div>";

    fetch('gethelp.php?screen=' + encodeURIComponent(screen))
        .then(function(response){ return response.JSON(); })
        .then(function(data){
            if (!data || data.status !== 'ok') {
                var msg = (data && data.message) ? data.message : 'No hay ayuda disponible para esta
pantalla.';
                body.innerHTML = escapeHtml(msg);
                modal.style.display = 'block';
                return;
            }
        })

    var mode = data.HelpMode || 'Text';
    var chapterName = data.ChapterName || '';
    var sectionName = data.SectionName || '';
    var showHeader = (data.ShowHeader === 'Y');
    var content = data.Content || '';

    var html = "";

    if (mode === 'HTML') {
        if (showHeader && (chapterName || sectionName)) {
            html += "<div style='margin:0 0 8px 0;'>" +

```

```

        escapeHtml(chapterName) + " - " + escapeHtml(sectionName) +
        "</strong></div>";
    }
    html += content;
} else {
    html += escapeHtml(content).replace(/\n/g, "<br>");
}

body.innerHTML = html;

var imgs = body.querySelectorAll('img');
imgs.forEach(function(img){
    img.style.maxWidth = "100%";
    img.style.height = "auto";
    img.style.cursor = "zoom-in";
    img.addEventListener('click', function(){
        var src = img.getAttribute('src');
        if (src) {
            var win = window.open(src, '_blank', 'noopener,noreferrer');
            if (win) { win.opener = null; }
        }
    });
});

modal.style.display = 'block';
})
.catch(function(error){
    console.error("Fetch Error:", error);
    body.innerHTML = "Error cargando ayuda.";
    modal.style.display = 'block';
});
}

function closeHelpModal() {
    document.getElementById('helpModal').style.display = 'none';
}

function escapeHtml(s) {
    s = String(s || "");
    return s
        .replace(/&/g, "&amp;")
        .replace(/</g, "&lt;")
        .replace(/>/g, "&gt;")
        .replace(/"/g, "&quot;")
        .replace(/'/g, "&#039;");
}
</script>

```

```
<div id="helpModal" style="display:none; position:fixed; z-index:9999; left:0; top:0; width:100%;
height:100%; overflow:auto; background-color:rgba(0,0,0,0.4);">
  <div style="background-color:#fff; margin:10% auto; padding:20px; border:1px solid #888; width:50%;
border-radius:10px;">
    <span onclick="closeHelpModal()" style="color:#aaa; float:right; font-size:28px; font-weight:bold;
cursor:pointer;">&times;</span>
    <h4>Ayuda de Pantalla</h4>
    <div id="helpModalBody" style="margin-top:10px;">
      </div>
    </div>
  </div>
</div>
EOT;
}
```

Apendice 3 - Codigo Para Hacer Sus Programas Independientes del Dominio

El siguiente codigo se proporciona en PHP para mostrar funcionalidad que puede convertirse al lenguaje de programacion de su preferencia. Estas funciones toman y asignan variables para evitar hardcoding de nombres de dominio, nombres de cPanel y directorios principales de archivos para acceso interno y descarga externa, asi como configuraciones globales de conexion de base de datos, email y otros parametros generales.

```
/*
/*
=== Identidad del Documento ===
DocType: Program
Category: Runtime Configuration Loader
Subsystem: Administracion Central de Configuracion Independiente del Website
programName: Runtime Configuration Functions
Audience: Both
Overview: Lee archivos de configuracion que hacen que la aplicacion sea portable entre dominios,
cuentas cPanel, bases de datos, sistemas de email, rutas de filesystem y proveedores de Inteligencia
Artificial.
Keywords: configuracion, independiente del dominio, configuracion runtime, filesystem, base de datos,
email, inteligencia artificial
=== Fin de Identidad del Documento ===
*/
```

```
/*
=== Proposito del Programa ===
Este modulo carga todos los valores especificos del ambiente requeridos por la aplicacion.
```

La misma base de codigo puede moverse hacia otro website cambiando archivos seguros de configuracion de texto en lugar de editar el codigo del programa.

```
=== Fin de Proposito del Programa ===
*/
/*
=== loadRuntimeConfig() ===
```

Resumen:
Cargador maestro para todas las secciones de configuracion runtime.

Aspectos Destacados:

- Inicializacion centralizada de configuraciones
- Carga de filesystem y rutas
- Configuracion de base de datos
- Configuracion de email
- Configuracion de proveedores de Inteligencia Artificial
- Variables globales del ambiente

```
*/
```

```

function loadRuntimeConfig()
{
    if (!setFileConfig()) { fatalConfigError("Unable to load file configuration."); }
    if (!loadDatabaseConfig()) { fatalConfigError("Unable to load database configuration."); }
    if (!loadSystemConfig()) { fatalConfigError("Unable to load system configuration."); }
    if (!loadEmailConfig()) { fatalConfigError("Unable to load email configuration."); }
    if (!loadAIConfig()) { fatalConfigError("Unable to load AI configuration."); }

    return true;
}

/*
=== fatalConfigError() ===
Displays a browser or command-line configuration error and stops execution.
*/
function fatalConfigError($message)
{
    $isWeb = (isset($_SERVER['HTTP_HOST']) || isset($_SERVER['REQUEST_METHOD']));

    if ($isWeb) {
        if (function_exists('ob_get_length')) {
            if (ob_get_length()) { ob_clean(); }
        }

        echo "<div style='margin:30px auto; max-width:900px; border:2px solid #dc3545;
background:#fff1f1; padding:18px; font-family:Arial, Helvetica, sans-serif; border-radius:8px;'>";
        echo "<h3 style='color:#dc3545; margin-top:0;'>System Configuration Error</h3>";
        echo "<p style='font-size:15px; color:#111;'>" . htmlspecialchars((string)$message, ENT_QUOTES,
'UTF-8') . "</p>";
        echo "</div>";
    } else {
        echo "SYSTEM CONFIGURATION ERROR: " . $message . PHP_EOL;
    }

    exit;
}

/*
=== setFileConfig() ===
Derives the cPanel ID, domain folder, physical root, and URL base from __DIR__.
Stores the results in globals and session variables.
*/
function setFileConfig()
{
    global $cpandid, $domain, $EnvironmentMode;
    global $FilePhysicalBase, $FileUrlBase;

    $cpandid = "";

```

```

$domain = "";
$EnvironmentMode = 'legacy';
$FilePhysicalBase = "";
$FileUrlBase = "";

$path = rtrim(str_replace('\\', '/', __DIR__), '/');

$parts = explode('/', $path);
$parts = array_values(array_filter($parts, 'strlen'));

if (count($parts) < 4) {
    fatalConfigError("Unable to determine website path from __DIR__: " . __DIR__);
}

if (strtolower($parts[0]) !== 'home') {
    fatalConfigError("Unexpected server path format: " . __DIR__);
}

$cpanelid = $parts[1] ?? "";
$siteFolder = $parts[2] ?? "";

if ($cpanelid === "") {
    fatalConfigError("Unable to determine cpanelid from path: " . __DIR__);
}

if ($siteFolder === "") {
    fatalConfigError("Unable to determine domain folder from path: " . __DIR__);
}

if (strpos($siteFolder, '.') !== false) {
    $tmp = explode('.', $siteFolder);
    $domain = $tmp[0];
} else {
    $domain = $siteFolder;
}

$FilePhysicalBase = '/home/' . $cpanelid . '/' . $siteFolder;

if (isset($parts[3]) && strtolower($parts[3]) === 'public_html') {
    $FilePhysicalBase .= '/public_html';
}

$FileUrlBase = 'https://' . $domain;

if (isset($_SESSION)) {
    $_SESSION['cpanelid'] = $cpanelid;
    $_SESSION['domain'] = $domain;
    $_SESSION['FilePhysicalBase'] = $FilePhysicalBase;
}

```

```

    $_SESSION['FileUrlBase'] = $FileUrlBase;
    $_SESSION['EnvironmentMode'] = $EnvironmentMode;
}

return true;
}

/*
=== loadDatabaseConfig() ===
Lee secure/database.txt y carga configuraciones operacionales y de base de datos company.
Los valores operacionales faltantes utilizan valores company como fallback y viceversa.
Configurado para acceso dual de bases de datos.
*/function loadDatabaseConfig($file=null)
{
    global $EnvironmentMode;
    global $DB_SERVER,$DB_USER,$DB_PASSWORD,$DB_NAME,$DB_PORT;
    global $CDB_SERVER,$CDB_USER,$CDB_PASSWORD,$CDB_NAME,$CDB_PORT;
    global $FilePhysicalBase;

    $fileRoot = rtrim((string)$FilePhysicalBase, '\\');
    if ($fileRoot === '') { fatalConfigError("FilePhysicalBase is blank before loadDatabaseConfig()."); }

    $defaultDir = $fileRoot . '/secure';

    if ($file === null || trim((string)$file) === '') {
        $file = $defaultDir . '/database.txt';
    } else {
        if (strpos($file, '/') === false && strpos($file, '\\') === false) {
            $file = $defaultDir . '/' . $file;
        }
    }

    if (!file_exists($file)) { fatalConfigError("Missing database config file: " . $file); }

    $lines = file($file, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
    if ($lines === false) { fatalConfigError("Unable to read database config file: " . $file); }

    foreach ($lines as $ln) {
        $ln = trim($ln);
        if ($ln === '') { continue; }
        if ($ln[0] == '#' || $ln[0] == ';') { continue; }
        if (strpos($ln, '=') === false) { continue; }

        list($k,$v) = array_map('trim', explode('=', $ln, 2));

        switch ($k) {
            case 'EnvironmentMode': $EnvironmentMode = $v; break;

```

```

    case 'DB_SERVER': $DB_SERVER = $v; break;
    case 'DB_USER': $DB_USER = $v; break;
    case 'DB_PASSWORD': $DB_PASSWORD = $v; break;
    case 'DB_NAME': $DB_NAME = $v; break;
    case 'DB_PORT': $DB_PORT = $v; break;

    case 'CDB_SERVER': $CDB_SERVER = $v; break;
    case 'CDB_USER': $CDB_USER = $v; break;
    case 'CDB_PASSWORD': $CDB_PASSWORD = $v; break;
    case 'CDB_NAME': $CDB_NAME = $v; break;
    case 'CDB_PORT': $CDB_PORT = $v; break;
}
}

if (!$DB_SERVER) { $DB_SERVER = $CDB_SERVER; }
if (!$DB_USER) { $DB_USER = $CDB_USER; }
if (!$DB_PASSWORD) { $DB_PASSWORD = $CDB_PASSWORD; }
if (!$DB_NAME) { $DB_NAME = $CDB_NAME; }
if (!$DB_PORT) { $DB_PORT = $CDB_PORT; }

if (!$CDB_SERVER) { $CDB_SERVER = $DB_SERVER; }
if (!$CDB_USER) { $CDB_USER = $DB_USER; }
if (!$CDB_PASSWORD) { $CDB_PASSWORD = $DB_PASSWORD; }
if (!$CDB_NAME) { $CDB_NAME = $DB_NAME; }
if (!$CDB_PORT) { $CDB_PORT = $DB_PORT; }

if (isset($_SESSION)) {
    $_SESSION['EnvironmentMode'] = $EnvironmentMode;

    $_SESSION['DB_SERVER'] = $DB_SERVER;
    $_SESSION['DB_USER'] = $DB_USER;
    $_SESSION['DB_PASSWORD'] = $DB_PASSWORD;
    $_SESSION['DB_NAME'] = $DB_NAME;
    $_SESSION['DB_PORT'] = $DB_PORT;

    $_SESSION['CDB_SERVER'] = $CDB_SERVER;
    $_SESSION['CDB_USER'] = $CDB_USER;
    $_SESSION['CDB_PASSWORD'] = $CDB_PASSWORD;
    $_SESSION['CDB_NAME'] = $CDB_NAME;
    $_SESSION['CDB_PORT'] = $CDB_PORT;
}

return true;
}

/*
=== loadSystemConfig() ===
Reads secure/config.txt and loads application directory settings.

```

```

*/
function loadSystemConfig($file=null)
{
    global $cftTempUpload,$cftTempDownload,$cftBackups;
    global $cftManual,$cftManuallImages;
    global $cftDocFileUploads,$cftDocImageUploads;
    global $FilePhysicalBase;

    $fileRoot = rtrim((string)$FilePhysicalBase, '/\');
    if ($fileRoot === '') { fatalConfigError("FilePhysicalBase is blank before loadSystemConfig()."); }

    $defaultDir = $fileRoot . '/secure';

    if ($file === null || trim((string)$file) === '') {
        $file = $defaultDir . '/config.txt';
    } else {
        if (strpos($file, '/') === false && strpos($file, '\\') === false) {
            $file = $defaultDir . '/' . $file;
        }
    }
}

if (!file_exists($file)) { fatalConfigError("Missing system config file: " . $file); }

$lines = file($file, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
if ($lines === false) { fatalConfigError("Unable to read system config file: " . $file); }

foreach ($lines as $ln) {
    $ln = trim($ln);
    if ($ln === '') { continue; }
    if ($ln[0] == '#' || $ln[0] == ';') { continue; }
    if (strpos($ln, '=') === false) { continue; }

    list($k,$v) = array_map('trim', explode('=', $ln, 2));
    $k = strtolower($k);
    $v = rtrim($v, '/') . '/';

    if ($k === 'tempupload') { $cftTempUpload = $v; }
    if ($k === 'tempdownload') { $cftTempDownload = $v; }
    if ($k === 'backups') { $cftBackups = $v; }
    if ($k === 'manual') { $cftManual = $v; }
    if ($k === 'manualimages') { $cftManuallImages = $v; }
    if ($k === 'docfileuploads') { $cftDocFileUploads = $v; }
    if ($k === 'docimageuploads') { $cftDocImageUploads = $v; }
}

if (isset($_SESSION)) {
    $_SESSION['cftTempUpload'] = $cftTempUpload;
    $_SESSION['cftTempDownload'] = $cftTempDownload;
}

```

```

    $_SESSION['cftBackups'] = $cftBackups;
    $_SESSION['cftManual'] = $cftManual;
    $_SESSION['cftManuallImages'] = $cftManuallImages;
    $_SESSION['cftDocFileUploads'] = $cftDocFileUploads;
    $_SESSION['cftDocImageUploads'] = $cftDocImageUploads;
}

return true;
}

/*
=== loadEmailConfig() ===
Lee secure/emailconfig.txt y carga configuraciones de remitente, reply-to, administrador, soporte
y configuraciones SMTP.
*/
function loadEmailConfig($file=null)
{
    global $mailer,$fromName,$fromEmail,$replyToEmail;
    global $authorEmail,$salesEmail,$supportEmail,$infoEmail,$testEmail,$bounceEmail,$adminEmail;
    global $smtpHost,$smtpPort,$smtpUser,$smtpPassword,$smtpSecure;
    global $FilePhysicalBase;

    $fileRoot = rtrim((string)$FilePhysicalBase, '\\');
    if ($fileRoot === '') { fatalError("FilePhysicalBase is blank before loadEmailConfig()."); }

    $defaultDir = $fileRoot . '/secure';

    if ($file === null || trim((string)$file) === '') {
        $file = $defaultDir . '/emailconfig.txt';
    } else {
        if (strpos($file, '/') === false && strpos($file, '\\') === false) {
            $file = $defaultDir . '/' . $file;
        }
    }
}

if (!file_exists($file)) { fatalError("Missing email config file: " . $file); }

$lines = file($file, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
if ($lines === false) { fatalError("Unable to read email config file: " . $file); }

foreach ($lines as $ln) {
    $ln = trim($ln);
    if ($ln === '') { continue; }
    if ($ln[0] == '#' || $ln[0] == ';') { continue; }
    if (strpos($ln, '=') === false) { continue; }

    list($k,$v) = array_map('trim', explode('=', $ln, 2));
    $k = strtolower($k);

```

```

if ($k === 'mailer') { $mailer = $v; }
if ($k === 'fromname') { $fromName = $v; }
if ($k === 'fromemail') { $fromEmail = $v; }
if ($k === 'replyto') { $replyToEmail = $v; }

if ($k === 'authoremail') { $authorEmail = $v; }
if ($k === 'salesemail') { $salesEmail = $v; }
if ($k === 'supportemail') { $supportEmail = $v; }
if ($k === 'infoemail') { $infoEmail = $v; }
if ($k === 'testemail') { $testEmail = $v; }
if ($k === 'bounceemail') { $bounceEmail = $v; }
if ($k === 'adminemail') { $adminEmail = $v; }

if ($k === 'smtp_host') { $smtpHost = $v; }
if ($k === 'smtp_port') { $smtpPort = $v; }
if ($k === 'smtp_user') { $smtpUser = $v; }
if ($k === 'smtp_password') { $smtpPassword = $v; }
if ($k === 'smtp_secure') { $smtpSecure = $v; }
}

if (trim((string)$mailer) === '') { $mailer = 'phpmail'; }
if (trim((string)$smtpPort) === '') { $smtpPort = '587'; }
if (trim((string)$smtpSecure) === '') { $smtpSecure = 'tls'; }

if (isset($_SESSION)) {
    $_SESSION['mailer'] = $mailer;
    $_SESSION['fromName'] = $fromName;
    $_SESSION['fromEmail'] = $fromEmail;
    $_SESSION['replyToEmail'] = $replyToEmail;

    $_SESSION['authorEmail'] = $authorEmail;
    $_SESSION['salesEmail'] = $salesEmail;
    $_SESSION['supportEmail'] = $supportEmail;
    $_SESSION['infoEmail'] = $infoEmail;
    $_SESSION['testEmail'] = $testEmail;
    $_SESSION['bounceEmail'] = $bounceEmail;
    $_SESSION['adminEmail'] = $adminEmail;

    $_SESSION['smtpHost'] = $smtpHost;
    $_SESSION['smtpPort'] = $smtpPort;
    $_SESSION['smtpUser'] = $smtpUser;
    $_SESSION['smtpPassword'] = $smtpPassword;
    $_SESSION['smtpSecure'] = $smtpSecure;
}

return true;
}

```

```

/*
=== loadAIConfig() ===
Lee secure/aihost.txt y carga el proveedor de Inteligencia Artificial, host, API key,
modelo de chat y modelo de embeddings.
*/
function loadAIConfig($file=null)
{
    global $cftAIProvider,$cftAIHost,$cftAPIKey,$cftAIChatModel,$cftAIEmbedModel;
    global $FilePhysicalBase;

    $fileRoot = rtrim((string)$FilePhysicalBase, '\\');
    if ($fileRoot === '') { fatalConfigError("FilePhysicalBase is blank before loadAIConfig()."); }

    $defaultDir = $fileRoot . '/secure';

    if ($file === null || trim((string)$file) === '') {
        $file = $defaultDir . '/aihost.txt';
    } else {
        if (strpos($file, '/') === false && strpos($file, '\\') === false) {
            $file = $defaultDir . '/' . $file;
        }
    }

    if (!file_exists($file)) { fatalConfigError("Missing AI config file: " . $file); }

    $lines = file($file, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
    if ($lines === false) { fatalConfigError("Unable to read AI config file: " . $file); }

    foreach ($lines as $ln) {
        $ln = trim($ln);
        if ($ln === '') { continue; }
        if ($ln[0] == '#' || $ln[0] == ';') { continue; }
        if (strpos($ln, '=') === false) { continue; }

        list($k,$v) = array_map('trim', explode('=', $ln, 2));
        $k = strtolower($k);

        if ($k === 'provider') { $cftAIProvider = $v; }
        if ($k === 'host') { $cftAIHost = $v; }
        if ($k === 'apikey') { $cftAPIKey = $v; }
        if ($k === 'chatmodel') { $cftAIChatModel = $v; }
        if ($k === 'embedmodel') { $cftAIEmbedModel = $v; }
    }

    if (isset($_SESSION)) {
        $_SESSION['cftAIProvider'] = $cftAIProvider;
        $_SESSION['cftAIHost'] = $cftAIHost;
    }
}

```

```
    $_SESSION['cftAIApiKey'] = $cftAIApiKey;
    $_SESSION['cftAIChatModel'] = $cftAIChatModel;
    $_SESSION['cftAIEmbedModel'] = $cftAIEmbedModel;
}

return true;
}
```

Apendice 4 - Funciones de Logon e Inicio del Sistema – Codigo

Lo siguiente representa codigo PHP real que controla la funcion de Logon de Usuarios y el modulo incluye de inicio que es incluido dentro de todos los programas GUI front end. Se proporciona solamente como ejemplo; sin embargo, la funcionalidad servira como guia para convertirla al lenguaje de programacion de su preferencia.

login.php

```
<?php
```

```
/*
```

```
=== Identidad del Documento ===
```

```
DocType: Program
```

```
Category: Login.php
```

```
Subsystem: Autenticacion Central de Plataforma e Inicializacion de Sesiones
```

```
programName: Login.php
```

```
Audience: Both
```

```
Overview: Autentica usuarios, inicializa controles de sesion, carga configuracion runtime y redirecciona logons exitosos hacia el menu principal.
```

```
Keywords: login, logon, autenticacion, sesion, seguridad
```

```
=== Fin de Identidad del Documento ===
```

```
*/
```

```
/*
```

```
=== Proposito del Programa Login ===
```

```
Este programa es la pantalla publica de login para la aplicacion.
```

```
Inicia la sesion, carga rutinas compartidas, verifica el estado actual del login,
```

```
valida credenciales enviadas y muestra el formulario de login cuando resulta necesario.
```

```
=== Fin de Proposito del Programa Login ===
```

```
*/
```

```
ini_set('display_errors', 1);
```

```
ini_set('display_startup_errors', 1);
```

```
error_reporting(E_ALL);
```

```
/* Configure session storage and lifetime before session_start(). */
```

```
ini_set('session.save_path', __DIR__ . '/sessionfiles');
```

```
ini_set('session.gc_maxlifetime', 28800);
```

```
ini_set('session.cookie_lifetime', 28800);
```

```
/* Establecer reglas de cookies del navegador para la sesion PHP. */
```

```
session_set_cookie_params([
```

```
    'lifetime' => 28800,
```

```
    'path' => '/',
```

```
    'secure' => false,
```

```
    'httponly' => true,
```

```
    'samesite' => 'Lax'
```

```
]);
```

```
session_start();
```

```

/* Crear una cookie de seguimiento de mensajes del navegador si aun no existe. */
if (!isset($_COOKIE['messageUID'])) {
    $messageUID = uniqid('msg_', true);

    setcookie('messageUID', $messageUID, [
        'expires' => time() + 86400,
        'path' => '/',
        'secure' => false,
        'httponly' => true,
        'samesite' => 'Lax'
    ]);

    $_COOKIE['messageUID'] = $messageUID;
}

/* Cargar variables comunes, rutinas de base de datos y rutinas compartidas de utilidades. */
require_once(getcwd()."/include/A_DBUserVariables.php");
require_once(getcwd()."/include/A_DBUserRoutines.php");
require_once(getcwd()."/include/A_CommonRoutines.php");

/* Si ya existe login activo, omitir el formulario e ir al menu principal. */
if (!empty($_SESSION['loggedonuserid'])) {
    header("Location: Menu_Main.php");
    exit;
}

/* Almacenar informacion de identidad de sesion para uso posterior del programa. */
$sessionID = session_id();
$_SESSION['sessionID'] = $sessionID;
$sessionIP = $_SERVER['REMOTE_ADDR'];
$_SESSION['sessionIP'] = $sessionIP;

/* Inicializar token CSRF para seguridad de formulario y sesion. */
if (!isset($_SESSION['csrfToken']) || $_SESSION['csrfToken'] == '') {
    $_SESSION['csrfToken'] = bin2hex(random_bytes(32));
}

/* Inicializar seguimiento de intentos fallidos de login y controles temporales de bloqueo. */
if (!isset($_SESSION['loginAttemptCount'])) {
    $_SESSION['loginAttemptCount'] = 0;
}

if (!isset($_SESSION['loginLockUntil'])) {
    $_SESSION['loginLockUntil'] = 0;
}

/* Runtime configuration globals populated by loadRuntimeConfig(). */
global $FilePhysicalBase;

```

```

global $FileUrlBase;
global $EnvironmentMode, $DB_SERVER, $DB_USER, $DB_PASSWORD,$DB_NAME, $DB_PORT;
global $CDB_SERVER,$CDB_USER,$CDB_PASSWORD, $CDB_NAME,$CDB_PORT;
global $cfgTempUpload,$cfgTempDownload,$cfgBackups,$cfgManual,$cfgManuallImages;
global $cfgAIProvider, $cfgAIHost,$cfgAIApiKey,$cfgAIChatModel,$cfgAIEmbedModel;

/* Cargar configuraciones runtime de base de datos, filesystem, URL e Inteligencia Artificial. */
loadRuntimeConfig();

/* Verificacion secundaria de login utilizando la variable alterna de usuario de sesion. */
if (!isset($_SESSION['LoggedOnUserID'])) { $_SESSION['LoggedOnUserID'] = ''; }

if (trim((string)$_SESSION['LoggedOnUserID']) !== "") {
    addMessage("Usuario ya conectado.");
    header("Location: Menu_Main.php");
    exit;
}

/* Inicializar variables de procesamiento de login. */
global $userid, $password, $lcUser, $errsw;

$userid = "";
$password = "";
$lcUser = "";
$errsw = 0;

/* Procesar formulario de login enviado. */
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['btnLogon'])) {
    if (!isset($_SESSION['errmsg'])) { $_SESSION['errmsg'] = ''; }

    $currentTime = time();

    if ($_SESSION['loginLockUntil'] > $currentTime) {
        $remainingSeconds = $_SESSION['loginLockUntil'] - $currentTime;
        $remainingMinutes = ceil($remainingSeconds / 60);
        addMessage("Too many failed login attempts. Try again in " . $remainingMinutes . " minutes.");
    } else {
        if ($_SESSION['loginLockUntil'] > 0 && $_SESSION['loginLockUntil'] <= $currentTime) {
            $_SESSION['loginAttemptCount'] = 0;
            $_SESSION['loginLockUntil'] = 0;
        }
    }

    if (isset($_POST['pctUserID'])) { $userid = trim((string)$_POST['pctUserID']); }
    if (isset($_POST['pctPassword'])) { $password = trim((string)$_POST['pctPassword']); }

    if ($userid === "") { addMessage("UserID is required."); }
    if ($password === "") { addMessage("Password is required."); }
}

```

```

    $lcUser = strtolower($userid);

    if (empty($_SESSION['errmsg'])) {
        if (loginUser($userid, $password)) {
            $_SESSION['loginAttemptCount'] = 0;
            $_SESSION['loginLockUntil'] = 0;

            $lcUser = strtolower($userid);
            $_SESSION['loggedonuserid'] = $lcUser;
            $_SESSION['LoggedInUserID'] = $lcUser;

            $sec40 = "";
            if (isset($_SESSION['secgroups_array'][40])) { $sec40 =
(string)$_SESSION['secgroups_array'][40]; }

            addMessage("Welcome " . $lcUser . " You have successfully logged in as " .
$_SESSION['UserType'] . ".");

            header("Location: Menu_Main.php");
            exit;
        } else {
            $_SESSION['loginAttemptCount']++;

            if ($_SESSION['loginAttemptCount'] >= 5) {
                $_SESSION['loginLockUntil'] = time() + 600;
                $_SESSION['loginAttemptCount'] = 0;
                addMessage("Too many failed login attempts. Wait 10 minutes before trying again.");
            } else {
                addMessage("UserID and Password do not match - Try Again.");
            }
        }
    }
}
?>
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Production Retrieval-Augmented AI Systems | Ivan Rodriguez</title>
    <meta name="description" content="Engineering AI Knowledge Systems: From Documentation to
Intelligent Assistants. A production-ready blueprint for ingesting organizational content into hybrid
search and retrieval pipelines.">
    <meta name="keywords" content="<?php echo htmlspecialchars($metaKeywords, ENT_QUOTES,
"UTF-8"); ?>">
    <meta name="robots" content="index, follow">

```

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
```

```
<style>
body{
  margin:0;
  font-family: Arial, Helvetica, sans-serif;
  background: linear-gradient(180deg, #e3f2fd 0%, #f7fbfe 55%, #ffffff 100%);
  color:#0f1720;
}
```

```
.wrapLogin{ max-width: 980px; margin: 0 auto; padding: 18px 14px 40px 14px; }
```

```
.topbarLogin{
  width:100%;
  background: rgba(255,255,255,0.78);
  border: 1px solid rgba(37,150,190,0.22);
  border-radius: 14px;
  box-shadow: 0 14px 28px rgba(0,0,0,0.06);
  padding: 18px 16px;
  margin-bottom: 18px;
}
```

```
.brandCenter{
  display:flex;
  align-items:center;
  justify-content:center;
  gap:14px;
  flex-wrap:wrap;
  text-align:center;
}
```

```
.brandLogo{
  width: 96px;
  height: 96px;
  object-fit: contain;
}
```

```
.brandText{ line-height:1.10; }
```

```
.brandTitle{
  font-weight: 900;
  letter-spacing: 0.4px;
  font-size: 34px;
  color:#2596be;
  margin:0;
}
```

```
.brandSub{
  margin-top:6px;
  font-size: 15px;
  color:#26323e;
}
```

```

}

.loginCard{
  max-width: 520px;
  margin: 0 auto;
  border: 2px solid #2596be;
  background-color: #e3f2fd;
  border-radius: 14px;
  box-shadow: 0 12px 24px rgba(0,0,0,0.06);
  padding: 16px 16px 14px 16px;
}
.welcomeBox{
  background: rgba(255,255,255,0.80);
  border: 1px solid rgba(37,150,190,0.30);
  border-radius: 12px;
  padding: 10px 12px;
  margin-bottom: 12px;
  color:#0f1720;
  font-size: 14px;
}

.loginLabel{ font-weight:700; color:#0f1720; margin-bottom:4px; }
.loginInput{ border-radius: 10px; border: 1px solid rgba(37,150,190,0.45); }
.loginRow{ margin-bottom: 10px; }

.btnRow{ display:flex; justify-content:center; margin-top: 8px; }
</style>
</head>

<body>

<div class="wrapLogin">

  <div class="topbarLogin">
    <div class="brandCenter">
      
      <div class="brandText">
        <div class="brandTitle">Production Retrieval-Augmented AI Systems</div>
        <div class="brandSub">AI Knowledge Demo System</div>
      </div>
    </div>
  </div>

  <center><input type='hidden' maxlength='1' size='1' name='upds'w' readonly='readonly' value='<?php
echo $upds; ?>'>
  <?php if (isset($_SESSION['errmsg'])) {echo '<span style="color: red;">' . $_SESSION['errmsg'] .
'</span>';unset($_SESSION['errmsg']); } ?>
  </center>

```

```
<div class="loginCard">

  <div class="welcomeBox">
    Welcome. You may login as userid <b>guest</b> password <b>guest</b> to view our system.
  </div>
  <form method="post" action="">
  <div class="loginRow">
    <div class="loginLabel">UserID</div>
    <input type="text" name="pctUserID" id="pctUserID" class="form-control loginInput" value="">
  </div>

  <div class="loginRow">
    <div class="loginLabel">Password</div>
    <input type="password" name="pctPassword" id="pctPassword" class="form-control loginInput"
value="">
  </div>

  <div class="btnRow">
    <button type="submit" name="btnLogon" id="btnLogon" class="btn btn-primary btn-sm">Logon to
Demo Website</button>
  </div>
  </form>
</div>

</div>

</body>
</html>
```

startup.php

```
<?php
/*
=== Document Identity ===
DocType: Program
Category: startup.php
Subsystem: Platform Core Session Start up All Programs included in all front end programs
programName: startup.php
Audience: Both
Overview: Login to the system, set session variables, and redirect to main menu if already logged in. Also
initializes session configuration and includes necessary database and utility files.
Keywords: login, logon, connect
=== End Document Identity ===
*/
```

```
/*
=== Program Purpose ===
startup.php is the common startup entry point for the front end application.
```

This program prepares the PHP runtime session, loads the standard database and utility include files, loads the runtime environment configuration, and establishes the base conditions required before the user is allowed to continue into the application.

Because this program is included at the beginning of the front end flow, it acts as a platform-level control point. Any program that depends on a valid session, database configuration, shared routines, or login state depends on this startup process being completed first.

```
=== End Program Purpose ===
*/
```

```
/*
=== Session Initialization ===
Only start a session when PHP has not already started one.
```

The custom `session_save_path` points PHP session storage to the application-level `sessionfiles` directory. This keeps session files under the application directory structure instead of relying on the server default session location.

The `dirname(__DIR__)` expression moves one directory above the current include directory so the `sessionfiles` directory can be shared consistently across the application.

```
*/
if (session_status() === PHP_SESSION_NONE) {
    session_save_path(dirname(__DIR__) . "/sessionfiles");
    session_start();
}
```

```
/*  
=== Output Buffering ===  
Start output buffering before any page output is produced.
```

This allows the startup process and downstream programs to safely perform redirects, set headers, or build page content before anything is sent to the browser.

```
*/  
ob_start();
```

```
/*  
=== Core Includes ===  
Load the standard platform include files required by the application.
```

A_DBUserVariables.php:
Defines common database and user-level variables used by the system.

A_DBUserRoutines.php:
Provides database access routines and user/security related database helpers.

A_CommonRoutines.php:
Provides shared application routines used across front end programs.

These includes are loaded from the current working directory so front end programs can use a consistent include path relative to the executing program.

```
*/  
require_once(getcwd()."/include/A_DBUserVariables.php");  
require_once(getcwd()."/include/A_DBUserRoutines.php");  
require_once(getcwd()."/include/A_CommonRoutines.php");
```

```
/*  
=== Runtime Configuration Load ===  
Load the runtime configuration after the core include files are available.
```

This step initializes the environment-specific settings such as database connection values, file paths, URL paths, and other deployment configuration values required by the application.

Keeping this logic inside loadRuntimeConfig() allows the same source code to run in different hosting environments without hard-coding server-specific values inside the front end programs.

```
*/  
loadRuntimeConfig();
```

```
?>
```

Apendice 5 - Esquemas JSON de Instrucciones API – Código

Este apéndice proporciona un esquema simplificado de instrucciones API utilizado con fines educativos y arquitectónicos. No representa un perfil productivo completo de instrucciones. Cada organización debe diseñar, probar, ajustar, proteger y gobernar sus propios perfiles de instrucciones API según sus requerimientos operacionales, reglas de seguridad, dominios empresariales y objetivos de retrieval.

Las estructuras JSON y nombres de campos dentro de los perfiles de instrucciones API permanecen en inglés para mantener consistencia técnica, compatibilidad entre plataformas y estabilidad de procesamiento. Sin embargo, cuando el sistema opera en ambientes Español LATAM, todas las preguntas de usuarios, fragmentos generados, palabras clave, metadata descriptiva, resúmenes y respuestas conversacionales deben procesarse y generarse en Español LATAM utilizando las reglas definidas por la tabla oficial de traducción técnica.

Las instrucciones internas del esquema pueden permanecer en inglés cuando representan contratos técnicos o estructuras de control.

Todo contenido orientado al usuario debe generarse en Español LATAM.

Los términos técnicos deben seguir la tabla oficial de traducción técnica aprobada por el proyecto.

Las respuestas no deben mezclar inglés y español excepto cuando el término técnico deba preservarse por compatibilidad operacional o por ser parte del esquema JSON.

El propósito de este apéndice consiste en mostrar la estructura de un contrato de instrucciones API sin exponer lógica propietaria de procesamiento.

```
{
  "requestTemplate": {
    "requestInfo": {
      "requestId": "...",
      "schemaVersion": "1.0",
      "instructionProfile": "SAMPLE_DOCUMENT_PROCESSING",
      "instructionVersion": "1.0",
      "requestTimestamp": "...",
      "workerType": "PARSE",
      "operationType": "DOCUMENT_PROCESSING"
    },
    "fileInfo": {
      "documentFormat": "PLAIN_TEXT",
      "mimeType": "text/plain",
      "fileSizeBytes": 0,
      "contentTransportFormat": "EXTRACTED_TEXT",
      "fileContentText": ""
    },
    "documentClassification": {
      "businessDomainHint": "",
      "documentTypeHint": ""
    },
    "structureAndOutput": {
      "responseMode": "FULL_JSON",
      "returnProcessingInfo": "Y",
      "returnUsageInfo": "Y",
      "returnContentInfo": "Y"
    },
    "metadataGuidance": {
      "metadataProfile": "SAMPLE_DOCUMENT_PROCESSING",
      "metadataHint": "Instructions may remain in English for schema consistency; however, all generated user-facing content, chunk material, keywords, metadata descriptions, summaries, and conversational responses must be produced in Spanish LATAM."
    }
  }
}
```

```
},
"promptInstructionBlocks": {
  "requestInfoInstructions": {
    "section": "requestInfo",
    "purpose": "Define the request identity, instruction profile, worker type, operation type, and response contract
for this run.",
    "rules": [
      "You must use requestInfo as the control contract for the run.",
      "You must return valid JSON only.",
      "You must not return markdown.",
      "You must not return explanatory prose outside the JSON response.",
      "You must return exactly one processingInfo object.",
      "You must return exactly one usagelInfo object.",
      "You must return only the top-level objects defined by the response schema."
    ]
  },
  "fileInfoInstructions": {
    "section": "fileInfo",
    "purpose": "Define the source text that must be processed.",
    "rules": [
      "You must use fileInfo.fileContentText as the authoritative source text when contentTransportFormat is
EXTRACTED_TEXT.",
      "You must not retrieve or infer missing file content.",
      "You must not rewrite, summarize, or expand the supplied source text unless the active instruction profile
explicitly requires it.",
      "If fileContentText is blank or unusable, you must return processingInfo.status as ERROR."
    ]
  },
  "documentClassificationInstructions": {
    "section": "documentClassification",
    "purpose": "Provide optional classification hints for processing.",
    "rules": [
      "You may use businessDomainHint as optional context when it is supplied.",
      "You must not invent a business domain if none is supplied.",
      "You must not allow businessDomainHint to override the supplied source text."
    ]
  },
  "processingInfoInstructions": {
    "section": "processingInfo",
    "purpose": "Return processing status, stop reason, warning text, and error text.",
    "rules": [
      "processingInfo.status must be SUCCESS, PARTIAL, or ERROR.",
      "processingInfo.stopReasonCode must contain a short machine-readable value.",
      "processingInfo.stopReasonText must contain a short human-readable explanation.",
      "processingInfo.hasWarningsSw must be Y or N.",
      "processingInfo.hasErrorsSw must be Y or N.",
      "processingInfo.warningText must contain concise warning text when warnings exist.",
      "processingInfo.errorText must contain concise error text when errors exist.",
      "If processing completed fully, status must be SUCCESS.",
      "If processing completed only part of the supplied input, status must be PARTIAL.",
      "If processing failed or the input is unusable, status must be ERROR.",
      "You must not report SUCCESS when required content was omitted or only partially processed."
    ]
  }
}
```

```
},
"usageInfoInstructions": {
  "section": "usageInfo",
  "purpose": "Return token and usage values known from the run.",
  "rules": [
    "usageInfo.inputTokens must be numeric.",
    "usageInfo.outputTokens must be numeric.",
    "usageInfo.totalTokens must be numeric.",
    "If exact token counts are unavailable, return 0.",
    "You must not invent token counts.",
    "You must not invent cost values.",
    "The caller system is responsible for calculating cost from stored pricing rules."
  ]
},
"contentInfoInstructions": {
  "section": "contentInfo",
  "purpose": "Return limited sample content result information.",
  "rules": [
    "You must return contentInfo only in the format required by the response schema.",
    "You must not include unsupported fields.",
    "You must not include markdown.",
    "You must not include explanatory prose outside the JSON response."
  ]
}
},
"responseSchema": {
  "processingInfo": {
    "status": "",
    "stopReasonCode": "",
    "stopReasonText": "",
    "hasWarningsSw": "N",
    "hasErrorsSw": "N",
    "warningText": "",
    "errorText": "",
    "processingNotes": ""
  },
  "usageInfo": {
    "provider": "",
    "model": "",
    "inputTokens": 0,
    "outputTokens": 0,
    "totalTokens": 0
  },
  "contentInfo": {
    "contentStatus": "",
    "contentItemCount": 0,
    "contentSummary": ""
  }
}
}
```

Appendix 6 - Ejemplos Estandar de CRUD y Move – Codigo

Esto describe el estandar para funciones CRUD y Move, las llamadas hacia escritura de auditoria de base de datos y la conectividad de base de datos.

```
/*  
=== Identidad del Documento ===  
DocType: Program Example  
Category: Standard Database CRUD and Audit Pattern  
Subsystem: Acceso a Base de Datos y Registro de Auditoria  
programName: Standard CRUD Database Access Example  
Audience: Both  
Overview: Demuestra seleccion de conexion de base de datos, llamadas genericas  
create/read/update/delete, registro automatico de auditoria y rutinas wrapper CRUD especificas por  
tabla.  
Keywords: crud, base de datos, conexion, auditoria, insert, select, update, delete  
=== Fin de Identidad del Documento ===  
*/
```

```
/*  
=== Arquitectura CRUD Estandar ===  
Este ejemplo posee dos capas.
```

La primera capa es la capa generica de acceso a base de datos. Abre la conexion correcta de base de datos, construye la instruccion SQL, ejecuta la instruccion y cierra la conexion.

La segunda capa es la capa wrapper especifica por tabla. Mapea los pct globals actuales dentro de arrays de campos y llama las rutinas genericas.

Los registros de auditoria de base de datos son escritos dentro de las rutinas genericas createRecord(), updateRecord() y deleteRecord(). Ese diseno mantiene centralizado el registro de auditoria para que cada rutina CRUD especifica por tabla no necesite su propio codigo de auditoria.

```
=== Fin de Arquitectura CRUD Estandar ===  
*/
```

```
/*  
=== ConnectToDB() ===  
Abre la conexion default hacia la base de datos operacional.
```

Legacy mode utiliza la conexion mysqli estandar de cuatro argumentos.

Non-legacy mode utiliza la conexion mysqli de cinco argumentos con soporte para puerto.

```
/*  
function ConnectToDB() {  
    global $EnvironmentMode, $DB_PORT;  
    global $DB_SERVER, $DB_USER, $DB_PASSWORD, $DB_NAME;  
    global $conn;
```

```

if ($EnvironmentMode === 'legacy') {
    $conn = new mysqli($DB_SERVER, $DB_USER, $DB_PASSWORD, $DB_NAME);

    if ($conn->connect_error) {
        die("Legacy DB connection failed: " . $conn->connect_error);
    }

    return $conn;
}

$portToUse = (!empty($DB_PORT)) ? (int)$DB_PORT : 3306;

$conn = new mysqli($DB_SERVER, $DB_USER, $DB_PASSWORD, $DB_NAME, $portToUse);

if ($conn->connect_error) {
    die("Cluster DB connection failed: " . $conn->connect_error .
        " (Host=$DB_SERVER, Port=$portToUse)");
}

return $conn;
}

/*
=== ConnectToCDB() ===
Abre la conexion hacia la base de datos company.

Las rutinas CRUD genericas llaman esta funcion cuando connectionType es "company".
*/
function ConnectToCDB() {
    global $EnvironmentMode, $DB_PORT;
    global $CDB_SERVER, $CDB_USER, $CDB_PASSWORD, $CDB_NAME;
    global $conn;

    if ($EnvironmentMode === 'legacy') {
        $conn = new mysqli($CDB_SERVER, $CDB_USER, $CDB_PASSWORD, $CDB_NAME);

        if ($conn->connect_error) {
            die("Legacy DB connection failed: " . $conn->connect_error);
        }

        return $conn;
    }

    $portToUse = (!empty($DB_PORT)) ? (int)$DB_PORT : 3306;

    $conn = new mysqli($CDB_SERVER, $CDB_USER, $CDB_PASSWORD, $CDB_NAME, $portToUse);

    if ($conn->connect_error) {

```

```

        die("Cluster DB connection failed: " . $conn->connect_error .
            " (Host=$CDB_SERVER, Port=$portToUse)");
    }

    return $conn;
}

```

```

/*
=== createRecord() ===
Rutina generica INSERT.

```

Esta rutina selecciona la conexion de base de datos, construye el SQL INSERT desde el array de campos, ejecuta el insert y posteriormente escribe el registro de auditoria de base de datos.

```

PUNTO DE AUDITORIA:
writeDatabaseAudit() es llamada aqui despues de que el INSERT es ejecutado.
*/
/*
=== createRecord() ===
Rutina generica INSERT.

```

Esta rutina selecciona la conexion de base de datos, construye el SQL INSERT desde el array de campos, ejecuta el insert y posteriormente escribe el registro de auditoria de base de datos.

```

PUNTO DE AUDITORIA:
writeDatabaseAudit() es llamada aqui despues de que el INSERT es ejecutado.
*/
function createRecord($table, $fields, $connectionType = "default") {
    $conn = ($connectionType === "company") ? ConnectToCDB() : ConnectToDB();

```

```

    $columns = array_keys($fields);
    $values = [];

```

```

    foreach ($fields as $value) {
        $escapedValue = $conn->real_escape_string((string)$value ?? "");
        $values[] = ""$escapedValue"";
    }

```

```

    $sql = "INSERT INTO $table (" . implode(" ", $columns) . ") VALUES (" . implode(" ", $values) . ")";

```

```

    $result = $conn->query($sql);

```

```

/*
=== Escritura de Auditoria de Base de Datos ===
Almacena el evento insert, tabla objetivo, valores de campos, tipo de conexion
y resultado de exito o error.

```

```

*/
writeDatabaseAudit([
    "routineName" => "createRecord",
    "tableName" => $table,
    "fields" => $fields,
    "connectionType" => $connectionType,
    "resultStatus" => $result ? "SUCCESS" : "ERROR"
]);

$conn->close();
return $result;
}

```

```

/*
=== readRecord() ===
Rutina generica SELECT.

```

Esta rutina selecciona la conexion de base de datos, construye la clausula WHERE desde el array de primary keys, ejecuta el SELECT, devuelve una fila y cierra la conexion.

El acceso de solo lectura no escribe un registro de auditoria de base de datos.

```

*/
function readRecord($table, $primaryKeys, $connectionType = "default") {
    $conn = ($connectionType === "company") ? ConnectToCDB() : ConnectToDB();

    $conditions = [];

    foreach ($primaryKeys as $key => $value) {
        $conditions[] = "$key = '$value'";
    }

    $sql = "SELECT * FROM $table WHERE " . implode(" AND ", $conditions);

    $result = $conn->query($sql);
    $data = $result ? $result->fetch_assoc() : null;

    $conn->close();
    return $data;
}

```

```

/*
=== updateRecord() ===
Rutina generica UPDATE.

```

Esta rutina selecciona la conexion de base de datos, construye la clausula SET desde el array de campos, construye la clausula WHERE desde el array de llaves, ejecuta el update y posteriormente escribe el registro de auditoria de base de datos.

PUNTO DE AUDITORIA:

writeDatabaseAudit() es llamada aqui despues de que el UPDATE es ejecutado.

```
*/
function updateRecord($table, $fields, $primaryKeys, $connectionType = "default") {
    $conn = ($connectionType === "company") ? ConnectToCDB() : ConnectToDB();

    $updates = [];
    $conditions = [];

    foreach ($fields as $key => $value) {
        $escapedValue = $conn->real_escape_string((string)$value ?? "");
        $updates[] = "$key = '$escapedValue'";
    }

    foreach ($primaryKeys as $key => $value) {
        $escapedKeyVal = $conn->real_escape_string((string)$value ?? "");
        $conditions[] = "$key = '$escapedKeyVal'";
    }

    $sql = "UPDATE $table SET " . implode(", ", $updates) . " WHERE " . implode(" AND ", $conditions);

    $result = $conn->query($sql);

    /*
    === Escritura de Auditoria de Base de Datos ===
    Almacena el evento update, tabla objetivo, valores modificados de campos,
    primary keys, tipo de conexion y resultado de exito o error.
    */
    writeDatabaseAudit([
        "routineName" => "updateRecord",
        "tableName" => $table,
        "fields" => $fields,
        "primaryKeys" => $primaryKeys,
        "connectionType" => $connectionType,
        "resultStatus" => $result ? "SUCCESS" : "ERROR"
    ]);

    $conn->close();
    return $result;
}

/*
=== deleteRecord() ===
Rutina generica DELETE.

```

Esta rutina selecciona la conexion de base de datos, construye la clausula WHERE desde el array de primary keys, ejecuta el delete y posteriormente escribe el registro de auditoria

de base de datos.

PUNTO DE AUDITORIA:

writeDatabaseAudit() es llamada aqui despues de que el DELETE es ejecutado.

*/

```
function deleteRecord($table, $primaryKeys, $connectionType = "default") {  
    $conn = ($connectionType === "company") ? ConnectToCDB() : ConnectToDB();
```

```
    $conditions = [];
```

```
    foreach ($primaryKeys as $key => $value) {  
        $escapedValue = $conn->real_escape_string((string)$value ?? "");  
        $conditions[] = "$key = '$escapedValue'";  
    }  
}
```

```
$sql = "DELETE FROM $table WHERE " . implode(" AND ", $conditions);
```

```
$result = $conn->query($sql);
```

```
/*
```

```
=== Escritura de Auditoria de Base de Datos ===
```

```
Almacena el evento delete, tabla objetivo, primary keys, tipo de conexion  
y resultado de exito o error.
```

```
*/
```

```
writeDatabaseAudit([  
    "routineName" => "deleteRecord",  
    "tableName" => $table,  
    "primaryKeys" => $primaryKeys,  
    "connectionType" => $connectionType,  
    "resultStatus" => $result ? "SUCCESS" : "ERROR"  
]);
```

```
$conn->close();
```

```
return $result;
```

```
}
```

```
/*
```

```
=== createManual_Chapter_Table() ===
```

```
Wrapper CREATE especifico por tabla.
```

Mapea los pct globals de Manual_Chapter_Table dentro de un array de campos y llama createRecord().

El registro de auditoria es escrito dentro de createRecord().

```
*/
```

```
function createManual_Chapter_Table() {  
    global $pctBookName, $pctChapterNumber, $pctChapterName, $pctPlacementSw, $pctReadFromSw,  
    $pctEditor, $pctLastDate;
```

```
    $fields = [
```

```

    'BookName' => $pctBookName,
    'ChapterNumber' => $pctChapterNumber,
    'ChapterName' => $pctChapterName,
    'PlacementSw' => $pctPlacementSw,
    'ReadFromSw' => $pctReadFromSw,
    'Editor' => $pctEditor,
    'LastDate' => $pctLastDate
  ];

  return createRecord('Manual_Chapter_Table', $fields, "company");
}

/*
=== readManual_Chapter_Table() ===
Wrapper READ especifico por tabla.

Llama readRecord() para recuperar una fila de Manual_Chapter_Table.
*/
function readManual_Chapter_Table($keys) {
  return readRecord('Manual_Chapter_Table', $keys, "company");
}

/*
=== updateManual_Chapter_Table() ===
Wrapper UPDATE especifico por tabla.

Mapea pct globals dentro del array de campos update y llama updateRecord().
El registro de auditoria es escrito dentro de updateRecord().
*/
function updateManual_Chapter_Table($keys) {
  global $pctBookName, $pctChapterNumber, $pctChapterName, $pctPlacementSw, $pctReadFromSw,
  $pctEditor, $pctLastDate;

  $fieldsToUpdate = [
    'BookName' => $pctBookName,
    'ChapterNumber' => $pctChapterNumber,
    'ChapterName' => $pctChapterName,
    'PlacementSw' => $pctPlacementSw,
    'ReadFromSw' => $pctReadFromSw,
    'Editor' => $pctEditor,
    'LastDate' => $pctLastDate
  ];

  return updateRecord('Manual_Chapter_Table', $fieldsToUpdate, $keys, "company");
}

/*
=== deleteManual_Chapter_Table() ===

```

Wrapper DELETE específico por tabla.

Llama deleteRecord() para una fila de Manual_Chapter_Table.

El registro de auditoría es escrito dentro de deleteRecord().

```
*/
```

```
function deleteManual_Chapter_Table($keys) {  
    return deleteRecord('Manual_Chapter_Table', $keys, "company");  
}
```

```
/*
```

```
=== moveManual_Chapter_Table() ===
```

Rutina MOVE específica por tabla.

Mueve una fila seleccionada de base de datos dentro de los pct globales utilizados por la pantalla.

```
*/
```

```
function moveManual_Chapter_Table($data) {  
    global $pctBookName, $pctChapterNumber, $pctChapterName, $pctPlacementSw, $pctReadFromSw,  
    $pctEditor, $pctLastDate;
```

```
    if (isset($data[0]) && is_array($data[0])) { $data = $data[0]; }
```

```
    $pctBookName = $data['BookName'];
```

```
    $pctChapterNumber = $data['ChapterNumber'];
```

```
    $pctChapterName = $data['ChapterName'];
```

```
    $pctPlacementSw = $data['PlacementSw'];
```

```
    $pctReadFromSw = $data['ReadFromSw'];
```

```
    $pctEditor = $data['Editor'];
```

```
    $pctLastDate = $data['LastDate'];
```

```
}
```

Appendix 7 - Rutinas Estandar de Seguridad –Codigo

odos los sistemas deben tener seguridad. Aqui se presenta un ejemplo entre posiblemente miles de rutinas existentes. Es simple y funcional. La seguridad es controlada mediante numeros de grupos funcionales, registros de pantallas que designan el grupo funcional al cual pertenece la funcion de pantalla y el registro de Usuario que define por grupo funcional de seguridad los niveles Read, Write y No Access. Adicionalmente, los niveles de seguridad del 1 al 5, donde 1 representa el nivel mas alto de seguridad, son utilizados para controlar funciones internas granulares de seguridad.

```
// -----
// Retornar nombre de pantalla (sin .php)
// -----
function getCleanScreenName() {
    return pathinfo(basename($_SERVER['PHP_SELF']), PATHINFO_FILENAME);
}

// -----
// Verificar seguridad completa, controlar errores,
// validar pantalla y establecer nivel de acceso del usuario
// -----
function initSecurityMinimum($requiredLevel = 'R') {
    global $screenName, $screenTitle;
    global $pctSecurityLevel, $pctReadOK, $pctWriteOK;
    global $pctLoggedOnUserID, $pctLoggedOnFirstName, $pctLoggedOnLastName;
    global $pctSecurityGroupsArray, $pctSecurityGroupsString;
    global $pctCompanyID;

    // 1. No conectado? Mostrar error y salir
    if (!isUserLoggedIn()) {
        $_SESSION['errmsg'] = "No ha iniciado sesion.";
        echo "<div class='container mt-4'><center><span style='color:red;'>{$_SESSION['errmsg']}<br><a
href='index.php'>Retornar a Login</a></span></center></div>";
        exit;
    }

    // 2. Cargar valores de usuario y sesion
    loadSecurityGlobalsFromSession();

    // 3. Obtener nombre de pantalla si aun no existe
    if (empty($screenName)) {
        $screenName = getCleanScreenName();
    }

    // 4. Obtener numero de grupo desde base de datos
    $group = getScreenGroup($screenName);

    if ($group < 1 || $group > 40) {
        $_SESSION['errmsg'] = "Grupo de Seguridad no configurado para pantalla: $screenName.";
        echo "<div class='container mt-4'><center><span style='color:red;'>{$_SESSION['errmsg']}<br><a
href='index.php'>Retornar</a></span></center></div>";
    }
}
```

```

    exit;
}

// 5. Obtener nivel de acceso desde array de seguridad
$access = $pctSecurityGroupsArray[$group] ?? 'N';
$pctSecurityLevel = $access;

// 6. Establecer indicadores read/write
$pctReadOK = in_array($access, ['R', 'U']);
$pctWriteOK = ($access === 'U');

// 7. Aplicar nivel requerido
$levels = ['N' => 0, 'R' => 1, 'W' => 2];

if ($levels[$access] < $levels[$requiredLevel]) {
    $_SESSION['errmsg'] = "Acceso Denegado: No posee permisos para acceder a la pantalla
$screenName</strong>.";
    echo "<div class='container mt-4'><center><span style='color:red;'>{$_SESSION['errmsg']}<br><a
href='index.php'>Retornar</a></span></center></div>";
    exit;
}

return $access;
}

// -----
// Convertir string de seguridad R/U/N de 40 caracteres
// dentro de array [1 => 'R', 2 => 'N'...]
// -----
function parseSecurityString($string) {
    $result = [];

    for ($i = 0; $i < 40; $i++) {
        $result[$i + 1] = strtoupper($string[$i] ?? 'N');
    }

    return $result;
}

// -----
// Verificar si usuario esta conectado
// -----
function isUserLoggedIn() {
    return isset($_SESSION['loggedonuserid']) && !empty($_SESSION['loggedonuserid']);
}

// -----
// Almacenar sesion de usuario despues de login exitoso

```

```

// -----
function initializeSecuritySession($data) {

    $_SESSION['loggedonuserid'] = $data['Userid'];
    $_SESSION['firstname'] = $data['FirstName'];
    $_SESSION['lastname'] = $data['LastName'];
    $_SESSION['secllevel'] = $data['SecLevel'];
    $_SESSION['UserType'] = $data['WorkType'];

    $secString = "";

    for ($i = 1; $i <= 40; $i++) {
        $field = 'SecGroup' . $i;
        $secString .= $data[$field] ?? 'N';
    }

    $_SESSION['secgroups'] = $secString;
    $_SESSION['secgroups_array'] = parseSecurityString($secString);
}

// -----
// Cargar Variables de Seguridad
// -----
function loadSecurityGlobalsFromSession() {

    global $pctLoggedOnUserID, $pctLoggedOnFirstName, $pctLoggedOnLastName;
    global $pctSecurityLevel, $pctSecurityGroupsString, $pctSecurityGroupsArray;

    $pctLoggedOnUserID    = $_SESSION['loggedonuserid']    ?? "";
    $pctLoggedOnFirstName = $_SESSION['firstname']        ?? "";
    $pctLoggedOnLastName  = $_SESSION['lastname']         ?? "";
    $pctSecurityLevel     = $_SESSION['secllevel']        ?? "";
    $pctSecurityGroupsString = $_SESSION['secgroups']      ?? str_repeat('N', 40);
    $pctSecurityGroupsArray = $_SESSION['secgroups_array'] ?? array_fill(1, 40, 'N');
}

// -----
// Login y validacion de credenciales utilizando hash
// -----
function loginUser($userid, $password) {

    $keys = ['Userid' => $userid];
    $data = readRecord('Sec_User_Table', $keys);

    // Validar existencia de usuario y password hash
    if (!$data || !password_verify($password, $data['PassWord'])) {
        $_SESSION['errmsg'] = "User ID o Password invalido.";
        return false;
    }
}

```

```

}

initializeSecuritySession($data);
return true;
}

// -----
// Obtener grupo de seguridad asignado a pantalla
// -----
function getScreenGroup($screenName) {

    $sql = "SELECT SecGroupNumber FROM Sec_Screen_Table WHERE ScreenName = '$screenName'";
    $result = getRecordsFullSQL($sql, "company");

    if ($result && isset($result[0]['SecGroupNumber'])) {
        return (int)$result[0]['SecGroupNumber'];
    }

    return 0;
}

// -----
// Verificar nivel de acceso del usuario hacia pantalla:
// R, U o N
// -----
function checkAccess($screenName) {

    if (!isset($_SESSION['secgroups_array'])) return 'N';

    $group = getScreenGroup($screenName);

    if ($group < 1 || $group > 40) return 'N';

    return $_SESSION['secgroups_array'][$group] ?? 'N';
}

// -----
// Aplicar nivel requerido de acceso a pantalla
// -----
function requireSecurity($screenName, $requiredLevel = 'R') {

    $access = checkAccess($screenName);

    $levels = ['N' => 0, 'R' => 1, 'U' => 2];

    if ($levels[$access] < $levels[$requiredLevel]) {
        $_SESSION['errmsg'] = "Acceso Denegado: No posee permisos para acceder a esta pantalla.";
    }
}

```

```

    echo "<div class='container mt-4'><center><span
style='color:red;'>{$_SESSION['errmsg']}</span></center></div>";
    exit;
}

return true;
}

// -----
// Verificar si usuario posee acceso 'W'
// hacia un numero de funcion (1-40)
// -----
function checkFunctionAccessByNumber($functionNumber) {

    if (!isset($_SESSION['secgroups_array'])) {
        $_SESSION['errmsg'] = "Sesion de seguridad no inicializada.";
        return false;
    }

    if ($functionNumber < 1 || $functionNumber > 40) {
        $_SESSION['errmsg'] = "Numero de funcion invalido.";
        return false;
    }

    $access = $_SESSION['secgroups_array'][$functionNumber] ?? 'N';

    return ($access === 'W');
}

// -----
// Verificacion opcional de acceso por funcion
// para botones y funcionalidades
// -----
function checkFunctionAccess($groupNumber, $requiredLevel = 'U') {

    if (!isset($_SESSION['secgroups_array'])) return false;

    $current = $_SESSION['secgroups_array'][$groupNumber] ?? 'N';
    $levels = ['N' => 0, 'R' => 1, 'U' => 2];

    return $levels[$current] >= $levels[$requiredLevel];
}

// Obtener nivel de acceso de usuario
// para grupo funcional especifico
function getAccessLevel($groupNumber) {

    if (!isset($_SESSION['secgroups_array'])) {

```

```

    return 'N';
}

if ($groupNumber < 1 || $groupNumber > 40) {
    return 'N';
}

return $_SESSION['secgroups_array'][$groupNumber] ?? 'N';
}

// -----
// Enviar numero de grupo y nivel requerido N R W
// opcionalmente nivel SecLevel requerido
// retorna true o false y envia mensaje al usuario
// -----
function requireGroupAccess($groupNumber, $requiredLevel, $requiredSecLevelMax = 0)
{
    global $screenName, $secLevel;

    $groupNumber = (int)$groupNumber;

    // Normalizar requiredLevel
    $requiredLevel = strtoupper(trim((string)$requiredLevel));

    if ($requiredLevel !== 'N' && $requiredLevel !== 'R' && $requiredLevel !== 'W') {
        $requiredLevel = 'N';
    }

    // Obtener nivel actual desde sesion
    $haveLevel = 'N';

    if (isset($_SESSION['secgroups_array']) && is_array($_SESSION['secgroups_array'])) {

        if (isset($_SESSION['secgroups_array'][(string)$groupNumber])) {
            $haveLevel = strtoupper(trim((string)$_SESSION['secgroups_array'][(string)$groupNumber]));
        }
        elseif (isset($_SESSION['secgroups_array'][$groupNumber])) {
            $haveLevel = strtoupper(trim((string)$_SESSION['secgroups_array'][$groupNumber]));
        }
    }

    if ($haveLevel !== 'N' && $haveLevel !== 'R' && $haveLevel !== 'W') {
        $haveLevel = 'N';
    }

    // Comparar niveles N/R/W
    $rank = array('N' => 0, 'R' => 1, 'W' => 2);

```

```

$groupOk = ($rank[$haveLevel] >= $rank[$requiredLevel]);

// Validacion opcional SecLevel
// 1 es maxima autoridad
$secOk = true;

$requiredSecLevelMax = (int)$requiredSecLevelMax;

if ($requiredSecLevelMax > 0) {

    $userSecLevel = (int)$secLevel;

    $secOk = ($userSecLevel > 0 && $userSecLevel <= $requiredSecLevelMax);
}

if ($groupOk && $secOk) {
    return true;
}

// Obtener nombre de grupo desde company DB
$groupName = "";

$where = "SecGroupNumber=".$groupNumber;

$groupName = trim((string)getSingleValue("Sec_Group_Table", "SecGroupName", $where,
"company"));

$screen = trim((string)$screenName);

if ($screen === '') { $screen = 'UnknownScreen'; }

// Construir mensaje compacto
$line1 = "No autorizado. Pantalla: ".$screen.".";
$line2 = "Grupo ".$groupNumber.(($groupName !== " ? " ".$groupName : "").": Su nivel ".$haveLevel.",
Requerido ".$requiredLevel.".";

if ($requiredSecLevelMax > 0) {

    $userSecLevel = (int)$secLevel;

    $line3 = "SecLevel: Su nivel ".$userSecLevel.", Requerido ".$requiredSecLevelMax.". Contacte su
Administrador de Seguridad para actualizar su perfil.";

    addCommonMessage($line1." ".$line2." ".$line3);
}
else {

    $line3 = "Contacte su Administrador de Seguridad para actualizar su perfil para este grupo.";
}

```

```
    addCommonMessage($line1." ".$line2." ".$line3);  
  }  
  
  return false;  
}
```

Appendix 8 - Búsqueda con Índice Invertido – Código

Esta sección representa en código PHP, adaptable a cualquier lenguaje, el módulo básico de búsqueda con Índice Invertido.

```
/*=== Identidad del Documento ===
```

```
DocType: Program
```

```
Category: Pantallas y UI
```

```
Subsystem: Índice Invertido
```

```
ProgramName: Search.php
```

```
Audience: Both
```

```
Overview: Pantalla de búsqueda presentada al cliente cuando se invoca búsqueda por palabras clave desde el menú principal
```

```
Keywords: búsqueda, índice invertido, retrieval de candidatos, JSON
```

```
=== Fin de Identidad del Documento === */
```

```
/**
```

```
 * SearchInclude.php
```

```
 * -----
```

```
 * Motor de búsqueda runtime contra el índice invertido de manuales/procedimientos.
```

```
 *
```

```
 * Punto de entrada:
```

```
 * array searchExecute(string $query, int $offset, int $limit, array $options = [])
```

```
 * Retorna:
```

```
 * [
```

```
 *   'rows' => [ // resultados ordenados y paginados listos para UI
```

```
 *   [
```

```
 *     'DocID' => int,
```

```
 *     'SourceType' => 'SCR'|'PROC',
```

```
 *     'SourceName' => string,
```

```
 *     'Title' => string,
```

```
 *     'BookName' => string,
```

```
 *     'ChapterName' => string,
```

```
 *     'SectionNumber' => string,
```

```
 *     'SectionName' => string,
```

```
 *     'HelpMode' => 'Text'|'HTML',
```

```
 *     'LastIndexed' => 'YYYY-MM-DD HH:MM:SS',
```

```
 *     'score' => float,
```

```
 *     'matchMask' => int, // OR de máscaras de campos a través de términos coincidentes
```

```
 *     'snippet' => string // texto plano sin entidades HTML
```

```
 *   ],
```

```
 *   ...
```

```
 * ],
```

```
 * 'total' => int, // total de documentos coincidentes antes de paginación
```

```
 * 'tookMs' => int // milisegundos transcurridos
```

```
 * ]
```

```
 *
```

```
 * Notas:
```

```
 * - Semántica AND a través de todos los tokens de consulta; siempre busca todo.
```

* - Peso de campos mediante FieldMask. Asignaciones actuales de mascara:

*

* 1 = title (t)

* 2 = section (s)

* 4 = chapter (c)

* 8 = meta (m)

* 16 = body (b)

* 32 = alt (a)

* 64 = bookName (n)

* 128 = docType (d)

* 256 = metaData (x)

* 512 = keywords (k)

*

* - Los tokens pueden originarse desde cualquiera de los canales anteriores.

* Los tokens de consulta deben coincidir a traves del documento utilizando

* semantica AND independientemente del campo.

*

* - El refuerzo de frase/proximidad utiliza el JSON Positions almacenado.

* Positions puede contener las siguientes llaves de campo:

*

* t,s,c,m,b,a,n,d,x,k

*

* - El snippet se construye aqui, uniendo nuevamente con tablas fuente, y se

* retorna como texto plano sin entidades HTML.

*

* - Todo acceso a base de datos utiliza funciones helper y la conexion company.

*

* - Logging de consultas hacia Search_QueryLog_Table utilizando generacion de

* llave getMaxValue()+1.

*

* Configuracion (opcional, tomada desde Search_Config_Table; se usan defaults si falta):

*

* search.maxAllLimit default 500 // limite duro cuando UI pasa "All"

* search.maxCandidateDocs default 5000 // limite seguro de union, raro en modo AND

*

* search.weights.title default 5.0

* search.weights.section default 2.0

* search.weights.chapter default 1.5

* search.weights.meta default 1.2

* search.weights.body default 1.0

* search.weights.alt default 0.5

*

* search.weights.bookName default 0.75

* search.weights.docType default 0.50

* search.weights.metaData default 0.90

* search.weights.keywords default 1.10

*

* search.idfFloor default 0.1

```
* search.phraseBoost      default 1.25
*
* search.snippet.preferHitLines default 2
* search.snippet.fallbackLines default 3
* search.snippet.maxLength default 240
*/
```

```
/* =====
REGLAS IMPORTANTES DE TOKENIZACION
```

1. Los tokens de búsqueda se normalizan a lowercase a-z0-9 solamente.
Underscores y puntuacion se eliminan durante tokenizacion.

2. Campos identificadores como:
ScreenName
ProcedureName
SourceName
NO DEBEN normalizarse como tokens.

Estos valores frecuentemente contienen underscores, por ejemplo SN_W_Add,
y deben preservarse exactamente para búsquedas SQL.

3. Por lo tanto:
- Las listas de tokens utilizan helpers de tokens normalizados.
- Las listas de nombres utilizan `_srch_sql_quote_list()` y preservan `'_'`.

```
===== */
```

```
function searchExecute($query, $offset, $limit, $options = []) {
    // Proposito: Ejecutar una búsqueda runtime AND sobre el indice invertido,
    // OR dentro de grupos de sinonimos, clasificar coincidencias con pesos de campo
    // mas IDF y refuerzo opcional de frase/proximidad, y luego retornar resultados
    // ordenados y paginados listos para la UI de Search.php.
    $t0 = microtime(true);

    // ----- Normalize input & tokens -----
    $qraw = (string)$query;
    $qnorm = _srch_norm_text($qraw);
    $tokens = _srch_tokenize($qnorm);

    // Stopword filter
    $stop = _srch_load_stopwords(); // set of normalized stopwords
    $tokens = array_values(array_filter($tokens, function($tok) use ($stop){
        return $tok !== '' && !isset($stop[$tok]);
    }));

    if (count($tokens) === 0) {
```

```

    $tookMs = (int)round((microtime(true)-$t0)*1000);
    _srch_log_query($qraw, 0, $tookMs);
    return ['rows'=>[], 'total'=>0, 'tookMs'=>$tookMs];
}

// ----- Config -----
$config = _srch_load_config([
    'search.maxAllLimit'      => '500',
    'search.maxCandidateDocs' => '5000',

    'search.weights.title'    => '5.0',
    'search.weights.section'  => '2.0',
    'search.weights.chapter'  => '1.5',
    'search.weights.meta'     => '1.2',
    'search.weights.body'     => '1.0',
    'search.weights.alt'      => '0.5',

    // New index channels (safe defaults even if not present in Search_Config_Table yet)
    'search.weights.bookName' => '0.75',
    'search.weights.docType'  => '0.50',
    'search.weights.metaData' => '0.90',
    'search.weights.keywords' => '1.10',

    'search.idfFloor'         => '0.1',
    'search.phraseBoost'      => '1.25',
    'search.snippet.preferHitLines' => '2',
    'search.snippet.fallbackLines' => '3',
    'search.snippet.maxLength' => '240',
]);

$maxAllLimit = isset($options['maxAllLimit']) ? (int)$options['maxAllLimit'] :
(int)$config['search.maxAllLimit'];
$maxCandidateDocs = isset($options['maxCandidateDocs']) ? (int)$options['maxCandidateDocs'] :
(int)$config['search.maxCandidateDocs'];

$weights = [
    'title' => (float)$config['search.weights.title'],
    'section' => (float)$config['search.weights.section'],
    'chapter' => (float)$config['search.weights.chapter'],
    'meta' => (float)$config['search.weights.meta'],
    'body' => (float)$config['search.weights.body'],
    'alt' => (float)$config['search.weights.alt'],

    'bookName' => (float)$config['search.weights.bookName'],
    'docType' => (float)$config['search.weights.docType'],
    'metaData' => (float)$config['search.weights.metaData'],
    'keywords' => (float)$config['search.weights.keywords'],
];

```

```

$idfFloor = max(0.000001, (float)$cfg['search.idfFloor']);
$phraseBoost = max(1.0, (float)$cfg['search.phraseBoost']);

$preferHitLines = max(0, (int)$cfg['search.snippet.preferHitLines']);
$fallbackLines = max(1, (int)$cfg['search.snippet.fallbackLines']);
$snippetMax = max(60, (int)$cfg['search.snippet.maxLength']);

// ----- Synonym expansion -> groups of terms -----
// OR within a group, AND across groups
$exp = _srch_expand_synonyms($tokens);
$queryGroupsTerms = $exp['groups'];
$origSet = $exp['origSet'];

// Resolve all terms in one SQL -> TokenID
$allTermsSet = [];
foreach ($queryGroupsTerms as $g) {
    foreach ($g as $t) { $allTermsSet[$t] = true; }
}
$allTerms = array_keys($allTermsSet);

if (empty($allTerms)) {
    $tookMs = (int)round((microtime(true)-$t0)*1000);
    _srch_log_query($qraw, 0, $tookMs);
    return ['rows'=>[], 'total'=>0, 'tookMs'=>$tookMs];
}

// Tokens are normalized a-z0-9; safe to quote list
$inList = _srch_sql_quote_list($allTerms);

$rowsTok = getRecordsFullSQL(
    "SELECT TokenID, Token, DocFreq
    FROM Search_Token_Table
    WHERE Token IN ($inList)",
    'company',
    'search.syn.tokresolve'
);

$tokIdByTerm = []; // term => TokenID
$dfByTokId = []; // TokenID => DocFreq
$termByTokId = []; // TokenID => term

if (is_array($rowsTok)) {
    foreach ($rowsTok as $r) {
        if (!is_array($r) || !isset($r['Token'])) continue;

        $tid = isset($r['TokenID']) ? (int)$r['TokenID'] : 0;
        if ($tid <= 0) continue;
    }
}

```

```

        $tk = strtolower((string)$r['Token']);
        $tokIdByTerm[$tk] = $tid;
        $dfByTokId[$tid] = max(1, isset($r['DocFreq']) ? (int)$r['DocFreq'] : 1);
        $termByTokId[$tid] = $tk;
    }
}

// Rebuild groups in TokenID space (drop unknown terms)
$queryGroups = [];
foreach ($queryGroupsTerms as $g) {
    $lst = [];
    foreach ($g as $t) {
        if (isset($tokIdByTerm[$t])) $lst[] = $tokIdByTerm[$t];
    }
    if (!empty($lst)) $queryGroups[] = array_values(array_unique($lst));
}

if (empty($queryGroups)) {
    $tookMs = (int)round((microtime(true)-$t0)*1000);
    _srch_log_query($qraw, 0, $tookMs);
    return ['rows'=>[], 'total'=>0, 'tookMs'=>$tookMs];
}

// ----- Grouped AND matching (build $docSet, $docAgg) -----
$idfCache = []; // tokenId => idf
$docSet = []; // DocID => true
$docAgg = []; // DocID => ['score'=>float, 'mask'=>int, 'pos'=> [tokenId => positions] ]

$synFactorExact = 1.0;
$synFactorAlias = 0.85;

$Ndocs = _srch_total_docs();

// Seed: union of first group
$g0 = $queryGroups[0];

foreach ($g0 as $tid) {

    if (!isset($idfCache[$tid])) {
        $df = isset($dfByTokId[$tid]) ? $dfByTokId[$tid] : 1;
        $idfCache[$tid] = _srch_idf($Ndocs, $df, $idfFloor);
    }

    $posts = _srch_load_postings_for_token($tid);

    $term = isset($termByTokId[$tid]) ? $termByTokId[$tid] : "";
    $isExact = ($term !== "" && isset($origSet[$term]));
}

```

```

$synW = $isExact ? $synFactorExact : $synFactorAlias;

foreach ($posts as $p) {

    $docId = (int)$p['DocID'];
    $tf = (int)$p['TermFreq'];
    $mask = (int)$p['FieldMask'];
    $posJ = $p['Positions'];

    $docSet[$docId] = true;

    $add = $tf * _srch_weight_from_mask($mask, $weights) * $idfCache[$tid] * $synW;

    if (!isset($docAgg[$docId])) {

        $docAgg[$docId] = [
            'score' => $add,
            'mask' => $mask,
            'pos' => [ $tid => _srch_positions_decode($posJ) ],
        ];

    } else {

        $docAgg[$docId]['score'] += $add;
        $docAgg[$docId]['mask'] |= $mask;

        if (!isset($docAgg[$docId]['pos'][$tid])) {
            $docAgg[$docId]['pos'][$tid] = _srch_positions_decode($posJ);
        }
    }
}

// For each remaining group: union within group, then INTERSECT with running set
for ($gi = 1; $gi < count($queryGroups); $gi++) {

    if (empty($docSet)) break;

    $grpSet = [];
    $grpAgg = [];

    $currentDocs = array_keys($docSet);

    foreach ($queryGroups[$gi] as $tid) {

        if (!isset($idfCache[$tid])) {
            $df = isset($dfByTokId[$tid]) ? $dfByTokId[$tid] : 1;
            $idfCache[$tid] = _srch_idf($Ndocs, $df, $idfFloor);
        }
    }
}

```

```

}

$post = _srch_load_postings_for_token_limited($tid, $currentDocs);

$term = isset($termByTokId[$tid]) ? $termByTokId[$tid] : '';
$isExact = ($term !== '' && isset($origSet[$term]));
$synW = $isExact ? $synFactorExact : $synFactorAlias;

foreach ($posts as $p) {

    $docId = (int)$p['DocID'];
    $tf = (int)$p['TermFreq'];
    $mask = (int)$p['FieldMask'];
    $posJ = $p['Positions'];

    $grpSet[$docId] = true;

    $add = $tf * _srch_weight_from_mask($mask, $weights) * $idfCache[$tid] * $synW;

    if (!isset($grpAgg[$docId])) {

        $grpAgg[$docId] = [
            'score' => $add,
            'mask' => $mask,
            'pos' => [ $tid => _srch_positions_decode($posJ) ],
        ];

    } else {

        if ($add > $grpAgg[$docId]['score']) $grpAgg[$docId]['score'] = $add;
        $grpAgg[$docId]['mask'] |= $mask;

        if (!isset($grpAgg[$docId]['pos'][$tid])) {
            $grpAgg[$docId]['pos'][$tid] = _srch_positions_decode($posJ);
        }
    }
}

// AND: intersect with running set
$newSet = array_intersect_key($docSet, $grpSet);

if (empty($newSet)) {
    $docSet = [];
    $docAgg = [];
    break;
}

```

```

// Merge this group's contributions into aggregate (only for surviving docs)
$kept = [];

foreach ($newSet as $docId => $_) {

    $docId = (int)$docId;

    if (!isset($docAgg[$docId]) || !isset($grpAgg[$docId])) continue;

    $prev = $docAgg[$docId];
    $ga = $grpAgg[$docId];

    $prev['score'] += $ga['score'];
    $prev['mask'] |= $ga['mask'];

    foreach ($ga['pos'] as $tid => $posArr) {
        if (!isset($prev['pos'][$tid])) $prev['pos'][$tid] = $posArr;
    }

    $kept[$docId] = $prev;
}

$docSet = $newSet;
$docAgg = $kept;

// Defensive cap
if (count($docSet) > $maxCandidateDocs) {

    $docSetIds = array_slice(array_keys($docSet), 0, $maxCandidateDocs);
    $docSet = array_fill_keys($docSetIds, true);
    $docAgg = array_intersect_key($docAgg, $docSet);
}
}

// No matches?
$total = count($docSet);

if ($total === 0) {
    $tookMs = (int)round((microtime(true)-$t0)*1000);
    _srch_log_query($qraw, 0, $tookMs);
    return ['rows'=>[], 'total'=>0, 'tookMs'=>$tookMs];
}

// ----- Phrase/proximity boost -----
if ($phraseBoost > 1.0 && !empty($queryGroups)) {

    $tokSet = [];
    foreach ($queryGroups as $g) { foreach ($g as $tid) { $tokSet[$tid] = true; } }
}

```

```

$tokenIds = array_keys($tokSet);

// Keep phrase checks on original text-like fields only
$fieldKeys = ['t','s','c','m','b','a'];

foreach ($docAgg as $docId => $agg) {

    $adjFound = false;

    foreach ($fieldKeys as $fk) {

        $perTok = [];

        foreach ($tokenIds as $tid) {
            if (isset($agg['pos'][$tid][$fk]) && is_array($agg['pos'][$tid][$fk])) {
                $perTok[$tid] = $agg['pos'][$tid][$fk];
            }
        }

        if (count($perTok) >= 2) {

            $tids = array_keys($perTok);

            for ($i = 0; $i < count($tids) && !$adjFound; $i++) {
                for ($j = $i + 1; $j < count($tids) && !$adjFound; $j++) {
                    if (_srch_has_adjacent($perTok[$tids[$i]], $perTok[$tids[$j]])) {
                        $adjFound = true;
                        break;
                    }
                }
            }

            if ($adjFound) break;
        }

        if ($adjFound) $docAgg[$docId]['score'] *= $phraseBoost;
    }
}

// ----- Join to doc metadata & sort -----
$docIds = array_keys($docAgg);
$docsMeta = _srch_load_docs_meta($docIds);

$items = [];

foreach ($docAgg as $docId => $agg) {

```

```

if (!isset($docsMeta[$docId])) continue;

$m = $docsMeta[$docId];

$items[] = [
    'DocID'    => $docId,
    'SourceType' => $m['SourceType'],
    'SourceName' => $m['SourceName'],
    'BookName'  => isset($m['BookName']) ? $m['BookName'] : '',
    'Title'     => $m['Title'],
    'ChapterName' => $m['ChapterName'],
    'SectionNumber' => $m['SectionNumber'],
    'SectionName' => $m['SectionName'],
    'HelpMode'  => '',
    'LastIndexed' => $m['LastIndexed'],
    'score'     => (float)$agg['score'],
    'matchMask' => (int)$agg['mask'],
];
}

usort($items, function($a, $b) {

    if ($a['score'] !== $b['score']) return ($a['score'] > $b['score']) ? -1 : 1;

    if ($a['LastIndexed'] !== $b['LastIndexed']) {
        return strcmp($b['LastIndexed'], $a['LastIndexed']);
    }

    $t = strcmp((string)$a['Title'], (string)$b['Title']);
    if ($t !== 0) return $t;

    return $a['DocID'] <=> $b['DocID'];
});

// ----- Paging -----
$totalHits = count($items);
$pageLimit = ($limit > 0) ? (int)$limit : min($totalHits, $maxAllLimit);
$pageOffset = max(0, (int)$offset);

if ($pageOffset >= $totalHits) $pageOffset = 0;

$pageSlice = array_slice($items, $pageOffset, $pageLimit);

// ----- Load text & build snippets for page slice -----
$scrNames = [];
$procNames = [];

foreach ($pageSlice as $it) {

```

```

    if ($it['SourceType'] === 'SCR') $scrNames[] = $it['SourceName'];
    if ($it['SourceType'] === 'PROC') $procNames[] = $it['SourceName'];
}

$scrMap = _srch_load_screen_texts($scrNames);
$procMap = _srch_load_proc_texts($procNames);

foreach ($pageSlice as &$it) {

    $nm = $it['SourceName'];

    if ($it['SourceType'] === 'SCR') {

        $hm = isset($scrMap[$nm]) ? $scrMap[$nm]['HelpMode'] : 'Text';
        $tx = isset($scrMap[$nm]) ? $scrMap[$nm]['Text'] : "";

        $it['HelpMode'] = $hm;
        $it['snippet'] = _srch_make_snippet($tx, $tokens, $preferHitLines, $fallbackLines, $snippetMax);

    } else {

        $hm = isset($procMap[$nm]) ? $procMap[$nm]['HelpMode'] : 'Text';
        $tx = isset($procMap[$nm]) ? $procMap[$nm]['Text'] : "";

        $it['HelpMode'] = $hm;
        $it['snippet'] = _srch_make_snippet($tx, $tokens, $preferHitLines, $fallbackLines, $snippetMax);
    }
}

unset($it);

// ----- Log & return -----
$tookMs = (int)round((microtime(true)-$t0)*1000);
_srch_log_query($qraw, (int)$total, $tookMs);

return ['rows'=>$pageSlice, 'total'=>$total, 'tookMs'=>$tookMs];
}

```

```

// *****
// Funciones helper y globals para SearchInclude.php
// *****

```

```

/* ===== Helpers ===== */

function _srch_norm_text($s) {
    // Proposito: Normalizar texto de entrada hacia una forma consistente
    // para procesamiento de busqueda.
    // - Decodificar entidades HTML
    // - Lowercase
    // - Reemplazar puntuacion por espacios
    // - Compactar espacios en blanco
    //
    // NOTA: Normalizar query string de entrada en Search.php antes de llamar searchExecute().

    $s = html_entity_decode($s, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
    $s = mb_strtolower($s, 'UTF-8');
    $s = preg_replace('/[^\p{L}\p{Nd}]+/u', '', $s);
    $s = preg_replace('/\s+/u', ' ', $s);
    return trim($s);
}

function _srch_tokenize($s) {
    // Proposito: Convertir query string normalizado dentro de una lista estable de tokens.
    // - Utiliza _srch_norm_text() primero para que los limites de tokens coincidan con reglas de
    // normalizacion
    // - Produce solamente tokens a-z0-9
    // - Elimina duplicados preservando orden

    $s = _srch_norm_text((string)$s);
    if ($s === "") return [];

    $parts = preg_split('/\s+/u', $s, -1, PREG_SPLIT_NO_EMPTY);

    $tokens = [];
    foreach ($parts as $p) {
        $t = preg_replace('/[^a-z0-9]/', '', strtolower((string)$p));
        if ($t !== "") $tokens[] = $t;
    }

    $seen = [];
    $out = [];
    foreach ($tokens as $t) {
        if (isset($seen[$t])) continue;
        $seen[$t] = true;
        $out[] = $t;
    }
    return $out;
}

/**

```

```

* Expandir tokens de consulta utilizando Search_Synonym_Table.
* Retorna:
* [
*   'groups' => [ ['tokA','tokB',...], ['tokC'], ... ],
*   'origSet' => [ 'tokA' => true, 'tokC' => true, ... ]
* ]
*
* Notas:
* - Espera tokens ya normalizados igual que el indice.
* - getRecordsFullSQL requiere connectionType = 'company' aqui porque estas tablas
*   fueron movidas a diferentes bases de datos y esta tabla de sinonimos reside en company DB.
*/
function _srch_expand_synonyms(array $tokens) {
    $out = ['groups' => [], 'origSet' => []];
    if (empty($tokens)) return $out;

    // Mantener originales para poder reducir levemente el peso de sinonimos posteriormente
    foreach ($tokens as $t) { $out['origSet'][$t] = true; }

    // Construir IN (...) con seguridad usando helper existente
    $inList = _srch_sql_quote_list($tokens);

    // Recuperar todos los tokens de los grupos de sinonimos que contienen cualquier token de consulta
    $sql = "
        SELECT s2.SynGroupID, s2.Token AS SynTok
        FROM Search_Synonym_Table s1
        JOIN Search_Synonym_Table s2 ON s2.SynGroupID = s1.SynGroupID
        WHERE s1.Token IN ($inList)
    ";
    $rows = getRecordsFullSQL($sql, 'company', 'search.syn.expand');

    $groupMap = [];
    $seenInAnyGroup = [];

    if (is_array($rows)) {
        foreach ($rows as $r) {
            if (!is_array($r)) continue;

            $gid = isset($r['SynGroupID']) ? (int)$r['SynGroupID'] : 0;
            $tok = isset($r['SynTok']) ? strtolower((string)$r['SynTok']) : "";

            // Normalizar token sinonimo hacia el mismo formato a-z0-9 utilizado en el sistema
            $tok = preg_replace('/[^a-z0-9]/', '', $tok);
            if ($gid <= 0 || $tok === "") continue;

            if (!isset($groupMap[$gid])) $groupMap[$gid] = [];
            $groupMap[$gid][$tok] = true;
            $seenInAnyGroup[$tok] = true;
        }
    }
}

```

```

    }
}

// Agregar cada grupo de sinonimos encontrado
foreach ($groupMap as $gid => $set) {
    $out['groups'][] = array_values(array_unique(array_keys($set)));
}

// Agregar grupos individuales para tokens no presentes en ningun grupo de sinonimos
foreach ($tokens as $t) {
    if (!isset($seenInAnyGroup[$t])) {
        $out['groups'][] = [$t];
    }
}

return $out;
}

function _srch_load_stopwords() {
    // Proposito: Cargar stopwords activas y normalizarlas al mismo formato
    // usado por tokenizacion de consultas.
    // NOTA: getRecordsFullSQL requiere connectionType = 'company' porque
    // estas tablas de soporte de busqueda se almacenan en company DB.

    $set = [];

    $sql = "SELECT Token FROM Search_Stopword_Table WHERE IsActive='Y'";
    $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.stopwords');

    foreach ($rows as $r) {

        if (!is_array($r) || !isset($r['Token'])) continue;

        $tok = strtolower((string)$r['Token']);

        // Normalizar al mismo formato que tokens de consulta
        $tok = preg_replace('/^[a-z0-9]/', '', $tok);

        if ($tok === '') continue;

        $set[$tok] = true;
    }

    return $set;
}

function _srch_load_config($defaults) {
    // Proposito: Cargar overrides de configuracion desde Search_Config_Table,

```

```

// utilizando defaults proporcionados cuando falten valores.
// NOTA: getRecordsFullSQL requiere connectionType = 'company' porque
// esta tabla de configuracion reside en company DB.

$keys = array_keys($defaults);
$in = _srch_sql_quote_list($keys);

$sql = "SELECT ConfigKey, ConfigVal FROM Search_Config_Table WHERE ConfigKey IN ($in)";
$rows = getRecordsFullSQL($sql, 'company', 'searchExecute.config');

$map = $defaults;

foreach ($rows as $r) {
    if (!is_array($r) || !isset($r['ConfigKey'])) continue;
    $map[(string)$r['ConfigKey']] = (string)$r['ConfigVal'];
}

return $map;
}

function _srch_total_docs() {
    // Proposito: Retornar cantidad total de documentos indexados para calculos IDF.
    // NOTA: getSingleValueFromSQL requiere connectionType = 'company' porque
    // Search_Doc_Table reside en company DB.

    static $N = null;

    if ($N !== null) return $N;

    $N = (int)getSingleValueFromSQL(
        "SELECT COUNT(*) FROM Search_Doc_Table",
        'company',
        'searchExecute.totalDocs'
    );

    return $N;
}

function _srch_idf($N, $df, $floor) {
    // Proposito: Calcular componente de score tipo IDF con floor para evitar valores extremos.

    $df = max(1, (int)$df);
    $N = max($df, (int)$N);

    $idf = log(($N + 1.0) / ($df + $floor)) + 1.0;

    return $idf;
}

```

```

function _srch_weight_from_mask($mask, $w) {
    // Proposito: Convertir bitset FieldMask en suma de pesos de campos para scoring.
    //
    // Bits de mascara:
    // 1 = title (t)
    // 2 = section (s)
    // 4 = chapter (c)
    // 8 = meta (m)
    // 16 = body (b)
    // 32 = alt (a)
    // 64 = bookName (n)
    // 128 = docType (d)
    // 256 = metaData (x)
    // 512 = keywords (k)

    $sum = 0.0;

    if (($mask & 1) !== 0) $sum += $w['title'];
    if (($mask & 2) !== 0) $sum += $w['section'];
    if (($mask & 4) !== 0) $sum += $w['chapter'];
    if (($mask & 8) !== 0) $sum += $w['meta'];
    if (($mask & 16) !== 0) $sum += $w['body'];
    if (($mask & 32) !== 0) $sum += $w['alt'];

    if (($mask & 64) !== 0) $sum += $w['bookName'];
    if (($mask & 128) !== 0) $sum += $w['docType'];
    if (($mask & 256) !== 0) $sum += $w['metaData'];
    if (($mask & 512) !== 0) $sum += $w['keywords'];

    return $sum > 0 ? $sum : $w['body'];
}

function _srch_positions_decode($JSON) {
    // Proposito: Decodificar Positions JSON hacia estructura estable para verificaciones de
    // frase/proximidad.
    //
    // Llaves posibles:
    // t,s,c,m,b,a,n,d,x,k

    $empty = [
        't'=>[], 's'=>[], 'c'=>[], 'm'=>[], 'b'=>[], 'a'=>[],
        'n'=>[], 'd'=>[], 'x'=>[], 'k'=>[]
    ];

    if ($JSON === null || $JSON === "") {
        return $empty;
    }
}

```

```

    $arr = JSON_decode($JSON, true);

    if (!is_array($arr)) return $empty;

    foreach (['t','s','c','m','b','a','n','d','x','k'] as $k) {
        if (!isset($arr[$k]) || !is_array($arr[$k])) {
            $arr[$k] = [];
        }
    }

    return $arr;
}

function _srch_has_adjacent($a, $b) {
    // Proposito: Retornar true si cualquier posicion en $a esta adyacente (+/- 1)
    // a cualquier posicion en $b para un pequeno refuerzo de frase/proximidad.

    if (empty($a) || empty($b)) return false;

    // Usar set para busqueda mas rapida
    $set = array_fill_keys($a, true);

    foreach ($b as $pos) {
        $p = (int)$pos;
        if (isset($set[$p - 1]) || isset($set[$p + 1])) return true;
    }
    return false;
}

function _srch_sql_quote_list($vals) {
    // Proposito: Construir lista SQL IN(...) con valores entre comillas desde tokens.
    // Se espera que los tokens esten normalizados a [a-z0-9].
    // Se aplica nuevamente aqui por seguridad y determinismo.

    $out = [];

    foreach ($vals as $v) {
        $t = strtolower((string)$v);
        $t = preg_replace('/^[^a-z0-9]/', '', $t);
        if ($t === "") continue;
        $out[] = "" . $t . "";
    }

    return empty($out) ? "" : implode(',', $out);
}

function _srch_sql_in_numbers($nums) {

```

```

// Proposito: Construir lista numerica SQL IN(...) desde un array de valores.

$numms = array_values(array_filter($numms, function($n){ return is_numeric($n); }));
if (empty($numms)) return 'NULL';

return implode(',', array_map('intval', $numms));
}

function _srch_load_postings_for_token($tokenId) {
// Proposito: Cargar todas las filas postings para un TokenID desde Search_Posting_Table.
// NOTA: getRecordsFullSQL requiere connectionType = 'company' porque estas tablas
// de indice de busqueda se almacenan en company DB.

$sql = "SELECT TokenID, DocID, TermFreq, FieldMask, Positions
FROM Search_Posting_Table
WHERE TokenID = " . (int)$tokenId;

$rows = getRecordsFullSQL($sql, 'company', 'searchExecute.postings1');

$out = [];

foreach ($rows as $r) {
if (!is_array($r)) continue;

$out[] = [
'TokenID' => isset($r['TokenID']) ? (int)$r['TokenID'] : (int)$tokenId,
'DocID' => isset($r['DocID']) ? (int)$r['DocID'] : 0,
'TermFreq' => isset($r['TermFreq']) ? (int)$r['TermFreq'] : 0,
'FieldMask' => isset($r['FieldMask']) ? (int)$r['FieldMask'] : 0,
'Positions' => isset($r['Positions']) ? (string)$r['Positions'] : "",
];
}

return $out;
}

function _srch_load_postings_for_token_limited($tokenId, $docIds) {
// Proposito: Cargar postings para un TokenID limitado a un conjunto conocido de DocIDs.
// Se utiliza despues de reducir el conjunto de candidatos.
// NOTA: getRecordsFullSQL requiere connectionType = 'company' porque estas tablas
// de indice de busqueda se almacenan en company DB.

if (empty($docIds)) return [];

$out = [];
$chunkSize = 900;

for ($i = 0; $i < count($docIds); $i += $chunkSize) {

```

```

$chunk = array_slice($docIds, $i, $chunkSize);
$in = _srch_sql_in_numbers($chunk);

$sql = "SELECT TokenID, DocID, TermFreq, FieldMask, Positions
FROM Search_Posting_Table
WHERE TokenID = " . (int)$tokenId . " AND DocID IN ($in)";

$rows = getRecordsFullSQL($sql, 'company', 'searchExecute.postingsL');

foreach ($rows as $r) {
    if (!is_array($r)) continue;

    $out[] = [
        'TokenID' => isset($r['TokenID']) ? (int)$r['TokenID'] : (int)$tokenId,
        'DocID' => isset($r['DocID']) ? (int)$r['DocID'] : 0,
        'TermFreq' => isset($r['TermFreq']) ? (int)$r['TermFreq'] : 0,
        'FieldMask' => isset($r['FieldMask']) ? (int)$r['FieldMask'] : 0,
        'Positions' => isset($r['Positions']) ? (string)$r['Positions'] : '',
    ];
}
}

return $out;
}

function _srch_load_docs_meta($docIds) {
    // Proposito: Cargar filas de metadata documental desde Search_Doc_Table
    // para un conjunto de DocIDs.
    // NOTA: getRecordsFullSQL requiere connectionType = 'company' porque estas tablas
    // de busqueda e indice se almacenan en company DB.

    $map = [];
    if (empty($docIds)) return $map;

    $chunkSize = 900;
    $docIds = array_values(array_unique($docIds));

    for ($i = 0; $i < count($docIds); $i += $chunkSize) {

        $chunk = array_slice($docIds, $i, $chunkSize);
        $in = _srch_sql_in_numbers($chunk);

        $sql = "SELECT DocID, SourceType, SourceName, BookName, ChapterName, SectionNumber,
SectionName, Title, LastIndexed
FROM Search_Doc_Table
WHERE DocID IN ($in)";

```

```

$rows = getRecordsFullSQL($sql, 'company', 'searchExecute.docMeta');

foreach ($rows as $r) {
    if (!is_array($r) || !isset($r['DocID'])) continue;

    $docId = (int)$r['DocID'];

    $map[$docId] = [
        'SourceType' => isset($r['SourceType']) ? (string)$r['SourceType'] : "",
        'SourceName' => isset($r['SourceName']) ? (string)$r['SourceName'] : "",
        'BookName' => isset($r['BookName']) ? (string)$r['BookName'] : "",
        'ChapterName' => isset($r['ChapterName']) ? (string)$r['ChapterName'] : "",
        // Mantener como string para preservar formato como 140.000
        'SectionNumber' => isset($r['SectionNumber']) ? (string)$r['SectionNumber'] : "",
        'SectionName' => isset($r['SectionName']) ? (string)$r['SectionName'] : "",
        'Title' => isset($r['Title']) ? (string)$r['Title'] : "",
        'LastIndexed' => isset($r['LastIndexed']) ? (string)$r['LastIndexed'] : "",
    ];
}

return $map;
}

function _srch_load_screen_texts($names) {
    // Proposito: Cargar HelpMode + ScreenHelp para valores ScreenName coincidentes.
    // NOTA: Los valores ScreenName incluyen '_' y no deben normalizarse como tokens.
    // NOTA: getRecordsFullSQL requiere connectionType = 'company' porque estas tablas
    // se almacenan en company DB.

    $map = [];
    if (empty($names)) return $map;

    $names = array_values(array_unique($names));
    $in = _srch_sql_quote_list($names);

    $sql = "SELECT ScreenName, HelpMode, ScreenHelp
        FROM Sec_Screen_Table
        WHERE ScreenName IN ($in)";

    $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.scrText');

    foreach ($rows as $r) {
        if (!is_array($r) || !isset($r['ScreenName'])) continue;

        $name = (string)$r['ScreenName'];

        $map[$name] = [

```

```

        'HelpMode' => isset($r['HelpMode']) ? (string)$r['HelpMode'] : "",
        'Text'    => isset($r['ScreenHelp']) ? (string)$r['ScreenHelp'] : "",
    ];
}

return $map;
}

function _srch_load_proc_texts($names) {
    // Proposito: Cargar HelpMode + ProcedureText para valores ProcedureName coincidentes.
    // NOTA: Los valores ProcedureName pueden incluir '_' y no deben normalizarse como tokens.
    // NOTA: getRecordsFullSQL requiere connectionType = 'company' porque estas tablas
    // se almacenan en company DB.

    $map = [];
    if (empty($names)) return $map;

    $names = array_values(array_unique($names));
    $in = _srch_sql_quote_list($names);

    $sql = "SELECT ProcedureName, HelpMode, ProcedureText
            FROM Proc_Procedure_Table
            WHERE ProcedureName IN ($in)";

    $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.procText');

    foreach ($rows as $r) {
        if (!is_array($r) || !isset($r['ProcedureName'])) continue;

        $name = (string)$r['ProcedureName'];

        $map[$name] = [
            'HelpMode' => isset($r['HelpMode']) ? (string)$r['HelpMode'] : "",
            'Text'     => isset($r['ProcedureText']) ? (string)$r['ProcedureText'] : "",
        ];
    }

    return $map;
}

function _srch_plain_text($html) {
    // Proposito: Convertir contenido almacenado HTML/text help a texto plano normalizado
    // adecuado para extraccion de snippet, preservando saltos de linea basicos.

    if ($html === null) $html = "";
    if (!is_string($html)) $html = "";

    $s = html_entity_decode($html, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');

```

```

if (!is_string($s)) $s = "";

// Preservar estructura basica como nuevas lineas antes de eliminar tags
$r = preg_replace('#<\s*br\s*/?>#i', "\n", $s); $s = is_string($r) ? $r : "";
$r = preg_replace('#<\s*p\s*>#i', "\n", $s); $s = is_string($r) ? $r : "";

$s = strip_tags($s);
if (!is_string($s)) $s = "";

// Normalizar espacios
$r = preg_replace('/\r\n|\r|\n/u', "\n", $s); $s = is_string($r) ? $r : "";
$r = preg_replace('/[\t]+/u', ' ', $s); $s = is_string($r) ? $r : "";
$r = preg_replace('/\n{3,}/u', "\n\n", $s); $s = is_string($r) ? $r : "";

return trim($s);
}

function _srch_make_snippet($rawHtml, $qTokens, $preferHitLines, $fallbackLines, $maxLen) {
    // Proposito: Construir snippet corto de texto plano favoreciendo lineas
    // que contienen cualquier token de consulta.

    $txt = _srch_plain_text($rawHtml);
    if ($txt === "") return "";

    $lines = preg_split('/\n/u', $txt);
    $lines = array_map('trim', $lines);
    $lines = array_values(array_filter($lines, function($l){ return $l !== ""; }));

    $hits = [];
    if ($preferHitLines > 0 && !empty($qTokens)) {

        // Tokens de consulta ya estan normalizados a-z0-9; construir lookup rapido
        $qSet = [];
        foreach ($qTokens as $t) {
            $t = strtolower((string)$t);
            if ($t !== "") $qSet[$t] = true;
        }

        foreach ($lines as $ln) {

            // Tokenizar la linea igual que la consulta para mantener coincidencia consistente
            $lnTokens = _srch_tokenize($ln);

            $found = false;
            foreach ($lnTokens as $lt) {
                if (isset($qSet[$lt])) { $found = true; break; }
            }
        }
    }
}

```

```

        if ($found) {
            $hits[] = $ln;
            if (count($hits) >= $preferHitLines) break;
        }
    }
}

$pick = !empty($hits) ? $hits : array_slice($lines, 0, (int)$fallbackLines);

$snippet = trim(implode(' ', $pick));

if (mb_strlen($snippet, 'UTF-8') > $maxLen) {
    $snippet = mb_substr($snippet, 0, $maxLen, 'UTF-8') . '...';
}

return $snippet;
}

function _srch_log_query($queryText, $results, $latencyMs) {
    // Proposito: Registrar una ejecucion de consulta de busqueda dentro de
    // Search_QueryLog_Table para analitica.

    // User id opcional desde sesion si esta disponible
    $userId = null;
    if (isset($_SESSION['loggedonuserid'])) {
        $userId = (string)$_SESSION['loggedonuserid'];
    } elseif (isset($_SESSION['UserID'])) {
        $userId = (string)$_SESSION['UserID'];
    }

    $qid = (int)getValue('Search_QueryLog_Table', 'QueryID', '1=1', 'company') + 1;

    $fields = [
        'QueryID' => $qid,
        'QueryText' => (string)$queryText,
        'UserID' => $userId,
        'Results' => (int)$results,
        'LatencyMs' => (int)$latencyMs,
        'QueryDate' => date('Y-m-d H:i:s'),
    ];

    createRecord('Search_QueryLog_Table', $fields, 'company');
}

```

Appendix 9 - Programa Modelo GUI Front End – Código

Para mantener uniformidad dentro de todo el conjunto de programas del sistema, se utiliza un programa modelo para los front ends GUI. Este es el código de ejemplo.

```
/*=== Identidad del Documento ===
```

```
DocType: Program
```

```
Category: Menu Principal del Sistema
```

```
Subsystem: Menu y Navegacion
```

```
ProgramName: Sample.php
```

```
Audience: Both
```

```
Overview: Muestra el menu principal despues del login del usuario, presentando modulos y opciones disponibles segun permisos del usuario. Proporciona enlaces de navegacion hacia funciones centrales del sistema y sirve como centro principal para acceder diferentes partes de la aplicacion.
```

```
Keywords: menu, menu principal, navegacion, permisos de usuario, modulos, dashboard
```

```
=== Fin de Identidad del Documento === */
```

```
$debug="N";
```

```
require_once(getcwd()."/include/A_DBUserVariables.php") ;
```

```
require_once(getcwd()."/include/A_DBUserRoutines.php") ;
```

```
require_once(getcwd()."/include/A_CommonRoutines.php") ;
```

```
require_once(getcwd()."/include/startup.php") ;
```

```
// MAIN GLOBAL STATEMENT
```

```
global $accessLevel, $accessLevelRequired, $accessAllow, $accessLevelNum, $accessLevelRequiredNum, $accessMessage;
```

```
global $secLevel, $secLevelRequired, $pctSecurityLevel, $groupNumber,$moduleAccessLevel, $pctReadOK, $pctWriteOK;
```

```
global $sesPreFix, $sesPrefix,$activePane, $pctActivePane,$panelID;
```

```
global $screenName, $screen, $screenTitle, $functionName, $HELP_KEY;
```

```
global $NewpctUserID, $origPctUserID, $pctLoggedOnUserID, $pctUserID, $pctUserid;
```

```
global $updsw, $addsw, $btnhide, $pksw, $pctreturn, $pctReturnTo;
```

```
// -----
```

```
// Identidad de pantalla - Reemplazar Sample con el nombre de pantalla, por ejemplo SN_ABC
```

```
// Reemplazar el prefijo XXX con el prefijo de pantalla, es decir,
```

```
// la abreviacion para esta pantalla, por ejemplo SN_ABC = SA,
```

```
// SN_DEF = SD, etc. Esto mantiene las llaves de sesion mas cortas y consistentes.
```

```
// Establecer parametros de seguridad para esta pantalla
```

```
// numero de grupo, nivel de acceso requerido y nivel de seguridad requerido.
```

```
// Estos valores seran utilizados en la verificacion de seguridad abajo
```

```
// y deben establecerse segun las necesidades de esta pantalla y los roles
```

```
// de usuario que deben tener acceso.
```

```
// -----$screenName = getCleanScreenName();
```

```
$screenTitle = "Sample";
```

```
$HELP_KEY = $screenName;
```

```
$panelID = "Sample";$functionName = $panelID;
```

```
$sesPrefix = "XXX";$sesPreFix = $sesPrefix;
```

```
$groupNumber = 1; // security group number for this screen
```

```
$accessLevelRequired = "R"; // required security level for this screen
```

```

$secLevelRequired = "5"; // required security level for this screen

// -----
// Identidad de usuario
// -----
// El session dump muestra: $_SESSION['loggedonuserid'] es la llave funcional.
// NewpctUserID se estaba utilizando sin establecerse dentro de este snippet.
// Establecerlo desde sesion para que $pctUserid / $pctLoggedOnUserID no queden en blanco.

//$NewpctUserID = $_SESSION['loggedonuserid'] ?? "";

$pctUserID = $_SESSION['loggedonuserid'] ?? "";
$pctUserid = $pctUserID;
$pctLoggedOnUserID = $pctUserID;

// -----
// VERIFICAR SEGURIDAD
// -----

// Paso 1: Inicializar acceso minimo segun necesidades generales del modulo
$moduleAccessLevel = initSecurityMinimum($accessLevelRequired);
$pctSecurityLevel = $moduleAccessLevel;

if ($moduleAccessLevel == "N") { $pctReadOK = false; $pctWriteOK = false; $accessMessage = "No esta
autorizado para esta funcion"; }
if ($moduleAccessLevel == "R") { $pctReadOK = true; $pctWriteOK = false; $accessMessage = "Esta
autorizado con acceso read para esta funcion"; }
if ($moduleAccessLevel == "W") { $pctReadOK = true; $pctWriteOK = true; $accessMessage = "Esta
autorizado con acceso update para esta funcion"; }

// Paso 2: Verificacion granular de seguridad basada en grupo
$accessLevel = $_SESSION['secgroups_array'][$groupNumber] ?? 'N';
$secLevel = $_SESSION['secllevel'];

// Mapear niveles de acceso hacia rangos numericos: N < R < W
$accessLevelNum = 0;
if ($accessLevel === "R") { $accessLevelNum = 1; }
if ($accessLevel === "W") { $accessLevelNum = 2; }

$accessLevelRequiredNum = 0;
if ($accessLevelRequired === "R") { $accessLevelRequiredNum = 1; }
if ($accessLevelRequired === "W") { $accessLevelRequiredNum = 2; }

// Seguridad descendente: 1 es el privilegio mas alto, por lo tanto usuario debe ser <= requerido
$accessAllow = "N";
if ($accessLevelNum > 0 && $accessLevelNum >= $accessLevelRequiredNum && $secLevel <=
$secLevelRequired) {

```

```

    $accessAllow = "Y";
}

// -----
// switches
// -----$updsw    = "N";
$addsw    = "N";
$btnhide  = "N";
$pksw     = "N";
$pctreturn = "N"; // Return button switch (Y = show button after save/add)

// Todos los globals del programa deben inicializarse arriba de esta linea
// por motivos de seguridad, consistencia y claridad operacional.

global $pctCSRFToken; // Hacer disponible globalmente el token CSRF para uso en formularios y
validaciones

// Manejo de postback
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    // Obtener campos POST
    //
    $updsw = "N";
    $updsw = isset($_POST['updsw']);

    // Limpiar mensaje de error
    if (!isset($_SESSION['errmsg'])) {
        $_SESSION['errmsg'] = "";
    } else {
        $_SESSION['errmsg'] = ""; // Limpiar mensajes existentes en cada carga
    }

    // issets para pantalla Sample

    if (isset($_POST['pctFieldName'])) { $pctFieldName = trim($_POST['pctFieldName']);
handleSessionNS($sesPreFix, 'set','pctFieldName',$pctFieldName); }

    $errsw = false; // switch de error para impedir procesamiento si falla validacion

    // Validar token CSRF al enviar formulario
    if (!isset($_POST['csrfToken']) || $_POST['csrfToken'] !== $_SESSION['csrfToken']) {
        addMessage("Fallo validacion de seguridad. Por favor refresque e intente nuevamente.");
        $errsw = true;
    }

    /* manejador de boton sample */
    if (isset($_POST['btn_Sample']))
    {

```

```

If (!$errsw)
{
    $ok = ValidateSample();
    if (!$ok)
    {
        $errsw = true;
        addMessage("Fallo validacion para Sample. Por favor revise su entrada. ");
    }
}
If (!$errsw)
{
    // verificar seguridad para update (DEBE ser primero) grupo W R N con nivel sec 1-5
    if (!requireGroupAccess(4, "W", 3)) {$errsw = true;}

    $ok = ProcessSample();
    if (!$ok)
    {
        $errsw = true;
        addMessage("Fallo procesamiento para Sample. Por favor revise su entrada. ");
    }
}

} // fin de manejador de boton sample */

} // Fin de manejo de postback

function validateSample()
{
    global $accessLevel, $accessLevelRequired, $accessAllow, $accessLevelNum,
    $accessLevelRequiredNum, $accessMessage;
    global $secLevel, $secLevelRequired, $pctSecurityLevel, $groupNumber, $moduleAccessLevel,
    $pctReadOK, $pctWriteOK;
    global $sesPreFix, $sesPrefix, $activePane, $pctActivePane, $panelID;
    global $screenName, $screen, $screenTitle, $functionName, $HELP_KEY;
    global $NewpctUserID, $origPctUserID, $pctLoggedOnUserID, $pctUserID, $pctUserid;
    global $upds, $addsw, $btnhide, $pksw, $pctreturn, $pctReturnTo;

    // globals adicionales aqui

    // Configurar array de errores para validaciones
    $errs = [];

    // -----
    // Normalizar prefijo de sesion
    // Estandar Ivan: SWU, sin guion
    // -----
    if (empty($sesPreFix) && !empty($sesPrefix))

```

```

{
    $sesPreFix = "XXX";
}

// -----
// Procesar validaciones para campos de pantalla Sample.
// Ejemplo mostrado para campo numerico.
// Ajustar segun tipos de campos y reglas de validacion reales.
// -----
$pctFieldName = (int)$pctFieldName;

if ($pctFieldName <= 0) {
    $errs[] = "FieldName esta vacio. No se puede actualizar.";
}

// -----
// Procesamiento final de errores
// -----
if (!empty($errs)) {

    foreach ($errs as $msg) {
        addMessage($msg);
    }

    return false;
}

// Validacion aprobada
// Restablecer session vars desde globals
handleSessionNS($sesPreFix,'set','pctFieldName',$pctFieldName);

return true;
}

function processSample()
{
    global $accessLevel, $accessLevelRequired, $accessAllow, $accessLevelNum,
    $accessLevelRequiredNum, $accessMessage;
    global $secLevel, $secLevelRequired, $pctSecurityLevel, $groupNumber, $moduleAccessLevel,
    $pctReadOK, $pctWriteOK;
    global $sesPreFix, $sesPrefix, $activePane, $pctActivePane, $panelID;
    global $screenName, $screen, $screenTitle, $functionName, $HELP_KEY;
    global $NewpctUserID, $origPctUserID, $pctLoggedOnUserID, $pctUserID, $pctUserid;
    global $updsW, $addsw, $btnhide, $pksw, $pctreturn, $pctReturnTo;

    // globals adicionales aqui

```

```

// -----
// Normalizar prefijo de sesion
// Estandar Ivan: SWU, sin guion
// -----
if (empty($sesPreFix) && !empty($sesPrefix)) {
    $sesPreFix = "XXX";
}

$errsw = false; // switch de error para impedir procesamiento si falla validacion

// Asegurarse que todos los globals sean enviados aqui para procesamiento

// Llamar funciones createTableName_Table()
// UpdateTableName_Table($keys)
// deleteTableName_Table($keys)
// segun necesidades de esta pantalla y estructura de base de datos.
//
// Ejemplo mostrado para update con llave unica.
// Ajustar segun requerimientos reales de procesamiento.

// Probar error de retorno de update
$errsw = true;

// Ejemplo de establecer switch de error en true
// Ajustar logica segun funciones reales de procesamiento y manejo de errores

if (!$errsw)
{
    addMessage("Proceso completado exitosamente.");
} else
{
    addMessage("Ocurrio un error durante procesamiento. Por favor revise su entrada e intente
nuevamente.");
    $errsw = true;
}
}

function generateSampleHTML()
{
    global $accessLevel, $accessLevelRequired, $accessAllow, $accessLevelNum,
$accessLevelRequiredNum, $accessMessage;
    global $secLevel, $secLevelRequired, $pctSecurityLevel, $groupNumber, $moduleAccessLevel,
$pctReadOK, $pctWriteOK;

```

```

global $sesPreFix, $sesPrefix, $activePane, $pctActivePane, $panelID;
global $screenName, $screen, $screenTitle, $functionName, $HELP_KEY;
global $NewpctUserID, $origPctUserID, $pctLoggedOnUserID, $pctUserID, $pctUserid;
global $updswh, $addsw, $btnhide, $pkswh, $pctreturn, $pctReturnTo;

global $pctCSRFToken; // Hacer disponible globalmente el token CSRF para formularios y validacion

$pctCSRFToken = handleSessionNS($sesPreFix, 'get', 'pctCSRFToken', (string)($_SESSION['csrfToken'] ??
''));

// Todos los globals de pantalla aqui

// Restablecer globals desde session vars aqui
$pctFieldName = handleSessionNS($sesPreFix, 'get', 'pctFieldName', $pctFieldName ?? '');

$html = "";

$html .= "<div class='property-pane' id='Sample' style='display:block;'>";
$html .= "<form method='post'>";

// Al generar formulario incluir token CSRF como hidden input
$html .= "<input type='hidden' name='csrfToken' value='".
htmlspecialchars((string)($_SESSION['csrfToken'] ?? ''), ENT_QUOTES) . "'>";

$html .= "<div style='text-align:center; padding:30px 10px;'>";

$html .= "<div style='margin-bottom:12px;'>";
$html .= "<img src='/images/BookBrandingLogo.png' alt='Logo' style='max-height:130px;
width:auto;'>";
$html .= "</div>";

$html .= "<div style='font-size:18px; color:#2596be; font-weight:600; margin-bottom:6px;'>";
$html .= "Bienvenido al Sitio Web Demo para";
$html .= "</div>";

$html .= "<div style='font-size:26px; color:#2596be; font-weight:700; margin-bottom:6px;'>";
$html .= "Ingenieria de Sistemas de Conocimiento con IA";
$html .= "</div>";

$html .= "<div style='font-size:18px; color:#2596be; font-weight:600;'>";
$html .= "Desde Ingestion de Datos hasta Aplicaciones Impulsadas por LLM";
$html .= "</div>";

$html .= "</div>";

$html .= "</form>";

return $html;

```

```

}

// Verificacion de seguridad allowAccess Y o N
function generateSecurityHTML()
{
    global $accessLevel, $secLevel, $accessAllow;

    $html = "";

    if ($accessAllow === "Y") {

        // Usuario autorizado
        return "";

    } else {

        // Usuario no autorizado
        $html = "<div style='margin: 2rem auto; padding: 2rem; max-width: 600px;";
        $html .= " background-color: #fff3cd; border: 1px solid #ffeeba;";
        $html .= " border-radius: 8px; color: #856404; font-size: 1.1rem;'>";

        $html .= " ⓧ Usted no esta autorizado para esta funcion.</strong><br><br>";
        $html .= " Por favor seleccione otra opcion del menu.<br><br>";
        $html .= " <em>Su Nivel de Acceso:</em> $accessLevel</strong><br>";
        $html .= " <em>Su Nivel de Seguridad:</em> $secLevel</strong><br>";
        $html .= " <em>Acceso Permitido:</em> $accessAllow</strong><br>";

        // Declaracion nueva de nivel de acceso
        global $accessLevelRequired, $secLevelRequired;

        $html .= "El Nivel de Acceso requerido es " . $accessLevelRequired . " y el Nivel de Seguridad
requerido es " . $secLevelRequired . "<br>";

        $html .= "</div>";

        return $html;
    }
}


?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title><?php echo $screenTitle; ?></title>

```

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
<meta name="description" content="<?php echo $screenTitle; ?>">
<meta name="keywords" content="<?php echo $screenTitle; ?>">
<! — Link css styles —>
<link rel="stylesheet" type="text/css" href="style/header.css" />
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
```

```
<script src="/javascripts/writehtml.js"></script>
```

```
<?php ListStyle(); ?>
```

```
<style>
/*  Grid Table Styling */
.grid-table {
  width: 100%;
  border-collapse: collapse;
  font-size: 10pt;
  font-family: Arial, sans-serif;
}
.grid-table th, .grid-table td {
  border: 1px solid #ccc;
  padding: 4px 6px;
  text-align: left;
}
.grid-table th {
  background-color: #e3f2fd; /* Light blue header */
  color: #000;
}
.grid-table tr:hover {
  background-color: #f1f1f1;
  cursor: pointer;
}
</style>
```

```
<style>
.lease-pane {
  background-color: #e3f2fd; /* Light blue background */
  padding: 16px; /* Comfortable padding */
  border-radius: 10px; /* Rounded corners */
  border: 1px solid #ccc; /* Light gray border */
  margin-top: 12px; /* Space above each pane */
  min-height: 90vh; /* 90% of screen height */
  display: none; /* Start hidden, shown via JS */
}

```

```
.lease-pane h4 {
text-align: center;

```

```

font-weight: bold;
color: #1976d2; /* Deeper blue for header */
margin-bottom: 12px;
}
.tab-btn {
padding: 6px 12px;
margin-right: 6px;
border: none;
background-color: #bbdefb;
border-radius: 8px;
font-size: 10pt;
color: #000;
}

.tab-btn.active {
background-color: #1976d2;
color: white;
font-weight: bold;
}

</style>
<style>
/* apply lease look to property panes */
.property-pane{
background-color:#e3f2fd; /* light blue */
padding:16px;
border-radius:10px;
border:1px solid #ccc;
margin:12px auto 0;
max-width:1100px; /* keeps content centered */
display:none; /* JS toggles to block */
min-height:90vh;
}
.property-pane h5{
text-align:center;
font-weight:600;
color:#1976d2; /* header blue */
margin-bottom:12px;
}

/* toolbar (property name + buttons) */
.prop-toolbar{
width:100%;
margin:8px 0 12px 0;
display:flex;
justify-content:center; /* <- center everything */
align-items:center;
gap:12px;

```

```

    flex-wrap:wrap;
  }
  .prop-toolbar label{ margin:0; }
  .prop-toolbar input[type="text"]{ width:30ch; max-width:100%; }
</style>

</head>
<body>

<!-- Header -->
<?php writePageHeader(true); ?>

<div class="container mt-4"> <!-- Division principal del cuerpo -->

    <center><input type='hidden' maxlength='1' size='1' name='updswh' readonly='readonly' value='<?php
echo $updswh; ?>'>
    <?php if (isset($_SESSION['errmsg'])) {echo '<span style="color: red;">' . $_SESSION['errmsg'] .
'</span>';unset($_SESSION['errmsg']); } ?>
    </center>

<!-- Seccion de menu dropdown -->
<?php ListMainMenu(); ?>

<?php
// Verificar seguridad y bloquear si corresponde
echo $html = generateSecurityHTML();
?>

<!-- Pantalla principal -->
<div class="container mt-3"> <!-- Inicio de div class container mt 3 -->

<!-- Seccion principal de pane -->

<?php
if ($accessAllow === "Y")
{
$functionName = "generateSampleHTML";
echo '<div class="property-screen">';
$html = $functionName();
echo $html;
echo '</div>';
}
?>

</div> <!-- Fin de pantalla principal -->

```

```
<!-- Bloque modal de una sola vez, ya existente dentro del patron --><div id="helpModal"
style="display:none; position:fixed; top:10%; left:10%; width:80%; background:#fff; border:2px solid
#ccc; padding:20px; z-index:1000;">
  <div id="helpModalBody" style="max-height:500px; overflow-y:auto; font-size:0.95rem;"></div>
  <div style="text-align:right; margin-top:10px;">
    <button onclick="document.getElementById('helpModal').style.display='none'" class="btn btn-
secondary btn-sm">Close</button>
  </div>
</div>
```

```
<script>
function copyGridToClipboard(tableId) {
  const table = document.getElementById(tableId);
  if (!table) {
    alert("Table not found.");
    return;
  }

  let text = "";
  for (let row of table.rows) {
    let rowText = [];
    for (let cell of row.cells) {
      rowText.push(cell.innerText.trim());
    }
    text += rowText.join('\t') + '\n'; // Tab-separated
  }

  navigator.clipboard.writeText(text).then(() => {
    alert("Entire grid copied to clipboard!");
  }).catch(err => {
    alert("Error copying grid: " + err);
  });
}
</script>
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
```

```
<br>
</body>
</html>
```

Appendix 10 - Sitio Web Demo

Un sitio web demo ha sido desarrollado para mostrar como funcionan todos los elementos presentados en este libro. Puede acceder al mismo en Datoscorporativosai.com. Utilice el boton ubicado en la esquina superior derecha para registrarse. El sistema solicitara su nombre y direccion de correo electronico. Una vez registrado, un correo electronico sera enviado con su guest userid y password. Luego nuevamente en la esquina superior derecha puede presionar login para ingresar al sistema. Todas las pantallas del sistema son visibles para usted; sin embargo, las actualizaciones hacia la base de datos no estan permitidas. Recomendamos ampliamente utilizar el Agente Conversacional de IA. El sistema contiene un Manual de Recursos Humanos de ejemplo, un Manual de Practicas Contables y un Manual de Politicas y Procedimientos de Tecnologia de Informacion. Presentados solamente como ejemplo, a continuacion se muestran algunas preguntas y respuestas reales proporcionadas por el sistema.

Recursos Humanos

Pregunta

Cual es la politica de horas extras de mi empresa

Respuesta Flujo Base de Datos

La politica de horas extras de su empresa, basada en el manual de politicas de Recursos Humanos proporcionado, es la siguiente:

1. Limite de horas extras
 - Las horas extras aplican para empleados de tiempo completo cuando trabajan mas de 8 horas en un dia o mas de 44 horas en una semana.
2. Manejo de horas extras
 - En lugar de pago inmediato, las horas extras se acumulan como tiempo compensatorio a razon de 1.5 veces las horas realmente trabajadas.
 - Ejemplo: si usted trabaja 2 horas extras, acumula 3 horas de tiempo libre.
3. Momento de utilizacion del tiempo compensatorio
 - El tiempo acumulado debe utilizarse en una fecha acordada mutuamente entre el Director Ejecutivo y el empleado.
4. Si el tiempo acumulado no es utilizado
 - Si el tiempo acumulado no se utiliza dentro de 6 meses, el empleador pagara dichas horas extras a razon de 1.5 veces el salario regular del empleado.
5. Maximo diario para ciertos viajes o conferencias
 - El maximo de tiempo laboral reclamable es de 12 horas por dia para viajes fuera de la ciudad o conferencias.
6. Requisito de aprobacion
 - Los empleados deben ser autorizados previamente para trabajar horas extras con el fin de recibir credito por ellas.
 - La excepcion corresponde a situaciones de emergencia donde no sea posible obtener aprobacion previa; en dicho caso, el empleado debe informar al Director Ejecutivo tan pronto como sea posible.
7. Formulario requerido
 - Los empleados deben presentar un formulario completado de aprobacion de horas extras para aprobacion del Director Ejecutivo.

Nota importante:

- La misma seccion de politica indica que debe alinearse con los estandares laborales provinciales y puede ajustarse a las practicas de la Fundacion. Por lo tanto, este parece ser el texto

documentado actual de la política; sin embargo, podría tratarse de una plantilla que debe confirmarse contra el manual final de la empresa o las reglas específicas de la provincia.

Respuesta Flujo Público

En general, las políticas de horas extras normalmente definen:

- cuando comienzan las horas extras,
- si las horas extras se pagan o se convierten en tiempo compensatorio,
- la tarifa de horas extras, normalmente 1.5 veces el salario regular,
- si se requiere aprobación previa,
- límites máximos de horas reclamables, y
- formularios u hojas de tiempo requeridas para aprobación y nómina.

La política interna específicamente establece que las horas extras comienzan después de 8 horas en un día o 44 horas en una semana para empleados de tiempo completo, se acumulan como tiempo compensatorio a razón de 1.5 veces y deben ser preautorizadas excepto en emergencias.

Pregunta

Cuántas horas en una semana debo trabajar para ser elegible para horas extras

Respuesta Flujo Base de Datos

Basado en el manual interno de Recursos Humanos proporcionado, no existe un número específico de horas semanales indicado para elegibilidad de horas extras. El manual indica que la Fundación debe referirse a la legislación provincial de estándares laborales para horas de trabajo y muestra espacios en blanco para la jornada laboral diaria y semanal, por lo tanto el límite de horas extras aún no ha sido completado.

La respuesta interna más precisa es:

su elegibilidad para horas extras depende de la provincia y de las reglas aplicables de estándares laborales, además de la política final completada por su empleador.

El documento hace referencia a estándares laborales provinciales como fuente principal. Una sección seleccionada menciona los estándares laborales de Nova Scotia, lo cual sugiere que Nova Scotia podría ser relevante; sin embargo, el material interno aún no proporciona el límite exacto de horas extras.

Si lo desea, puedo ayudarle a encontrar el límite exacto de horas extras una vez indique su provincia y si usted es empleado por hora, asalariado o pertenece a una categoría administrativa/exenta.

Respuesta Flujo Público

La elegibilidad para horas extras normalmente se basa en legislación provincial o territorial de empleo y no solamente en el manual de la empresa.

En Canadá, el límite semanal de horas extras depende de la ubicación donde trabaja. Por ejemplo, muchas jurisdicciones utilizan 44 horas por semana, mientras otras utilizan reglas diarias o semanales diferentes. Algunos empleados también pueden estar exentos de reglas de horas extras, como ciertos administradores o profesiones específicas.

Debido a que el documento interno no contiene el número exacto, el siguiente paso correcto consiste en confirmar:

1. Su provincia o territorio
2. Si su trabajo está regulado provincial o federalmente
3. Si su rol está exento de reglas de horas extras

Si usted se encuentra en Nova Scotia, el material interno apunta hacia Nova Scotia Labour Standards Code como fuente principal, aunque no incluye el límite exacto.

Indique su provincia y puedo proporcionarle la regla exacta de horas extras.

Pregunta

Puedo utilizar mis horas extras y tiempo compensatorio para licencia administrativa

Respuesta Flujo Base de Datos

Basado en el texto interno de políticas de Recursos Humanos proporcionado:

1. Horas extras / tiempo compensatorio

- Si, la politica permite que las horas extras sean utilizadas como tiempo compensatorio en lugar de pago inmediato.
- Las horas extras se acumulan a razon de 1.5 veces.
- Dicho tiempo debe utilizarse en una fecha acordada mutuamente entre el Director Ejecutivo y el empleado.
- Si el tiempo acumulado no se utiliza dentro de 6 meses, el empleador pagara dicho tiempo a razon de 1.5 veces el salario regular del empleado.
- Las horas extras deben ser autorizadas previamente, excepto en situaciones de emergencia.

2. Licencia / licencia administrativa

- No se observa una politica interna clara que otorgue un derecho general para utilizar licencia administrativa de la misma manera que el tiempo compensatorio.
- La seccion de licencias menciona licencia de emergencia sin pago, posible licencia personal sin pago dependiendo de necesidades operacionales y licencias medicas/discapacidad.
- El texto no define claramente una categoria separada llamada licencia administrativa.

Por lo tanto, la respuesta interna mas precisa es:

- Probablemente usted puede utilizar tiempo compensatorio acumulado, pero solamente con aprobacion y acuerdo de programacion con el Director Ejecutivo.
- No puedo confirmar a partir de la politica proporcionada que usted pueda utilizar licencia administrativa de la misma forma.

Appendix 11 - Lecturas Recomendadas y Referencias de la Industria

Las siguientes referencias representan libros seleccionados, documentos técnicos, discusiones arquitectónicas y recursos de la industria relacionados con sistemas empresariales de retrieval, Generación Aumentada por Retrieval, arquitectura de búsqueda, ingeniería de metadatos, retrieval mejorado mediante grafos y diseño operacional de Inteligencia Artificial. Estas referencias influyeron en partes del pensamiento arquitectónico discutido a través de este manuscrito y son recomendadas para lectores interesados en una exploración más profunda de sistemas empresariales de conocimiento e ingeniería de retrieval.

1.

Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich
Introduction to Information Retrieval
Cambridge University Press

Uno de los textos modernos fundamentales sobre retrieval de información, indexación, clasificación, puntuación, arquitectura de búsqueda e ingeniería de sistemas de retrieval. Particularmente valioso para comprender los fundamentos históricos y técnicos que soportan los sistemas empresariales de búsqueda.

2.

Jurafsky, Daniel; Martin, James H.
Speech and Language Processing
Pearson

Ampliamente reconocido como una de las referencias más importantes en Procesamiento de Lenguaje Natural. Cubre procesamiento estadístico del lenguaje, análisis semántico, embeddings, conceptos de retrieval y lingüística computacional relevantes para sistemas modernos de Inteligencia Artificial.

3.

Russell, Stuart; Norvig, Peter
Artificial Intelligence: A Modern Approach
Pearson

Una referencia amplia de Inteligencia Artificial que cubre sistemas de razonamiento, búsqueda, aprendizaje automático, representación de conocimiento y conceptos arquitectónicos de IA fundamentales para sistemas empresariales de Inteligencia Artificial.

4.

Lewis, Patrick; Perez, Ethan; Piktus, Aleksandra; et al.
Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
Documento de Investigación Meta AI

Uno de los primeros documentos influyentes que describió formalmente arquitecturas de Generación Aumentada por Retrieval y la integración de sistemas de retrieval con modelos de lenguaje de gran escala.

5.

White, Tom
Hadoop: The Definitive Guide
O'Reilly Media

Lectura importante para comprender sistemas de datos de gran escala, conceptos de procesamiento distribuido, infraestructura de indexación y arquitecturas escalables de información relevantes para sistemas empresariales de conocimiento.

6.

Knuth, Donald E.
The Art of Computer Programming
Addison-Wesley

Aunque no está enfocado directamente en retrieval con IA, esta serie fundamental refuerza fuertemente ingeniería disciplinada de sistemas, pensamiento algorítmico, conceptos de indexación, métodos de ordenamiento y principios de arquitectura computacional.

7.

Hearst, Marti A.

Search User Interfaces

Cambridge University Press

Una referencia altamente valiosa que discute como los usuarios interactuan con sistemas de retrieval, comportamiento de busqueda, estructuras de navegacion, filtrado y diseno de presentacion de retrieval.

8.

Chen, Wenhui; Wang, Hexiang; et al.

Graph-Based Retrieval-Augmented Generation

Discusiones de Investigacion Industrial y Publicaciones Emergentes iniciadas alrededor de 2024

Un creciente cuerpo de investigacion que discute arquitecturas de retrieval mejoradas mediante grafos, sistemas de retrieval conscientes de relaciones, estructuras conectadas de conocimiento y razonamiento contextual a traves de ambientes empresariales de informacion.

9.

Silberschatz, Abraham; Korth, Henry F.; Sudarshan, S.

Database System Concepts

McGraw-Hill

Una referencia fundamental que cubre arquitectura de bases de datos relacionales, normalizacion, indexacion, sistemas de consultas, modelado de datos y administracion estructurada de informacion empresarial.

10.

Baeza-Yates, Ricardo; Ribeiro-Neto, Berthier

Modern Information Retrieval

Addison-Wesley

Una de las referencias classicas sobre teoria de retrieval de informacion, modelos de clasificacion, indexacion, arquitectura de busqueda, puntuacion de relevancia y diseno de sistemas de retrieval. Las arquitecturas empresariales de retrieval discutidas a traves de este manuscrito tambien fueron influenciadas por experiencia operacional practica relacionada con:

- sistemas empresariales de contabilidad
- ambientes operacionales de reportes
- analisis empresarial impulsado por hojas de calculo
- repositorios documentales intensivos en metadata
- sistemas procedimentales de flujo de trabajo
- ingenieria de motores de busqueda
- pipelines de ingestion documental de gran escala
- experimentacion de retrieval hibrido
- arquitecturas de retrieval conscientes de relaciones
- restricciones operacionales empresariales del mundo real

El autor recomienda a los lectores explorar tanto investigacion academica de retrieval como literatura practica de ingenieria de sistemas empresariales. Muchos de los avances mas importantes en Inteligencia Artificial empresarial no emergen solamente de modelos aislados de IA, sino de la integracion de ingenieria de retrieval, arquitectura operacional, sistemas de metadata, estructuras de gobierno y organizacion contextual del conocimiento trabajando conjuntamente como sistemas unificados.