Trabajo Práctico Integrador
"Hotel Sol de las Sierras"

ENTREGA FINAL "ALGORITMOS Y ESTRUCTURA DE DATOS"

Institución: UTN - FRVM

Carrera: Ingeniería en Sistemas de información

Materia: Algoritmo y estructura de datos

Empresa: Hotel Sol de las Sierras

Título: Presentación del prototipado

Profesores de Cátedra:

- Ing. Rinaldi, Mario
- Ing. Simieli, Paola
- Ing. Parruccia, Pablo

Grupo: Anarqui-Team (Grupo 2) – (anarquiteam@protonmail.com)

Integrantes:

- COSTAMAGNA, Leonel Osvaldo (leonel18costamagna@gmail.com)
- DEMARCHI, Andrés (demarchi.andres128@gmail.com)
- FANTINI, Marcial Domingo (marcialfantini@gmail.com)
- GONZÁLEZ, Daniel Agustín (danielito.2405@hotmail.com)
- MAZA BIANCHI, Lucas (mazalucas@hotmail.com)
- MOSCA, Sebastián Jesús (sebastian.mosca@outlook.com)
- VARGAS, Milagros Asunción (milagros-a-vargas-@hotmail.com)

Año de cursado: 1er Año - Ingeniería en Sistemas de Información

ÍNDICE

Contenido

ARCHIVOS

En los principios de la computación, hubo un problema fatal a la utilización práctica y eficaz de una computadora, y ese problema fue el de inconstantibilidad.
Debido a la naturaleza volátil, degradable y de escaso almacenamiento de nuestras memorias principales, la data almacenada en variables es impersistente. La impersistencia es incierta, enigmática, e inverosímil.

Para lidiar con este problema, creamos memorias de mayor capacidad, más económicas, no volátiles e inalterable en condiciones propicias, y en estas memorias, almacenamos **archivos** (aunque si se indaga a mayor profundidad sobre el verdadero uso de los mismos, se podría entablar una relación entre los archivos y cualquier tipo de memoria, sin necesidad de ser no volátiles)

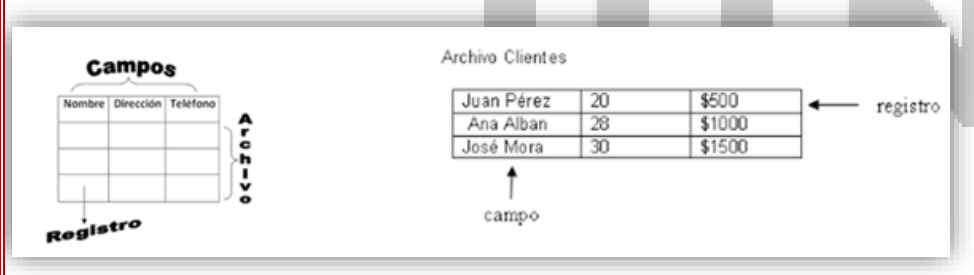
Un archivo o fichero es un conjunto de datos estructurados en una colección de entidades elementales denominadas registros o artículos, que son de igual tipo y constan a su vez de diferentes entidades de nivel más bajo denominadas campos

Registro

Es una colección de información, normalmente relativa a una entidad particular, y de campos lógicamente relacionados, que pueden
ser tratados como una unidad por algún programa.
Un ejemplo de un registro puede ser la información de un determinado empleado que contiene los campos de nombre, dirección, fecha de nacimiento, estudios, salario, etc.

Campo

Se lo puede clasificar como un ítem o elemento de datos elementales, tales como un nombre, número de empleados, ciudad, número de identificación, etc, cuyo propósito es formar divisiones dentro de los propios registros a modo de clasificación de los bytes de datos almacenados. Suele estar caracterizado por su **nombre**, **tamaño o longitud** y su **tipo de datos** (cadena de caracteres, entero, lógico, etcét era.)



Como se puede ver en la imagen, los campos están presentes como la "clasificación" de los datos, mientras que los registros son una "colección" de los mismos

Organización de los archivos

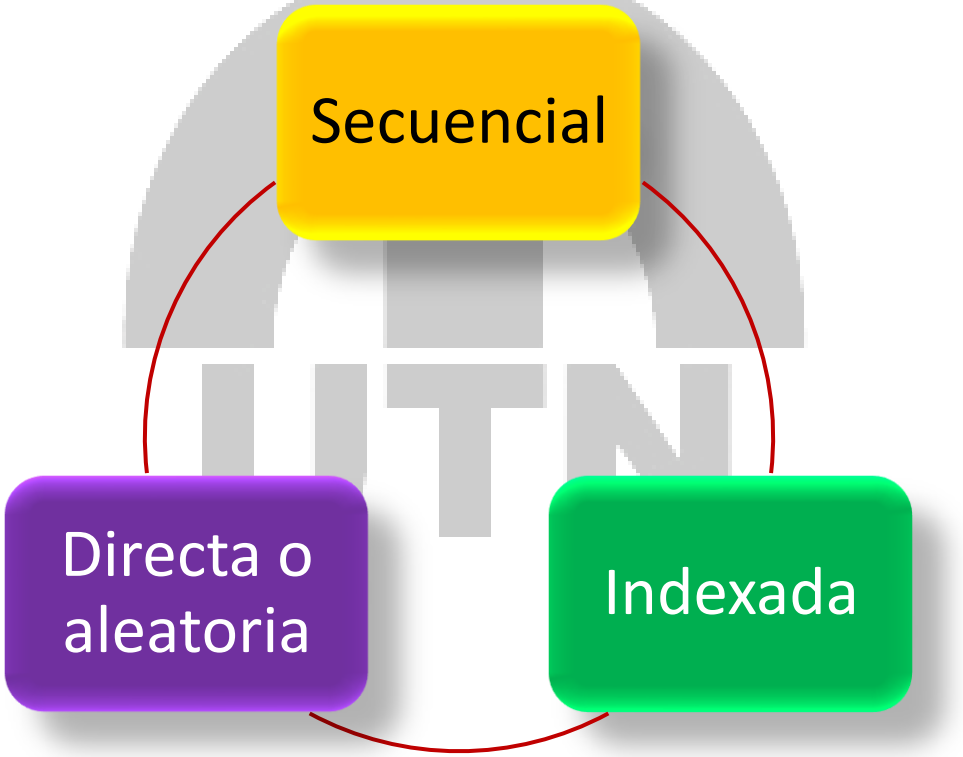
La organización de los archivos esta fundamentalmente ligada a la tecnología empleada para la distribución de la data e información en el medio físico, este medio físico suele referirse como ***soporte***, y podemos hablar de soportes **secuenciales**, y de soportes **direccionables**.

Los soportes secuenciales fueron los primeros desarrollados y la tecnología consiste en organizar la data-información y meta data continuamente ([data1, data2, data3, data4...). Para poder acceder al registro 5, necesitarías acceder al registro 1, registro 2, registro 3, registro 4, y recién ahí uno podría acceder al registro 5.

Mientras que los soportes direccionables proveen a los registros de *un campo clave*, una ***key*** que los identifica, una dirección. Esta dirección nos permite acceder a la información de un archivo sin tener que atravesar toda la secuencia de información.

Estos distintos soportes junto a la organización de los registros de los archivos, implican la existencia de dos tipos de acceso a los archivos, secuencial y directo, respectivamente. La significación de estos y cómo funcionan ya fue descrita en la definición de los distintos soportes.

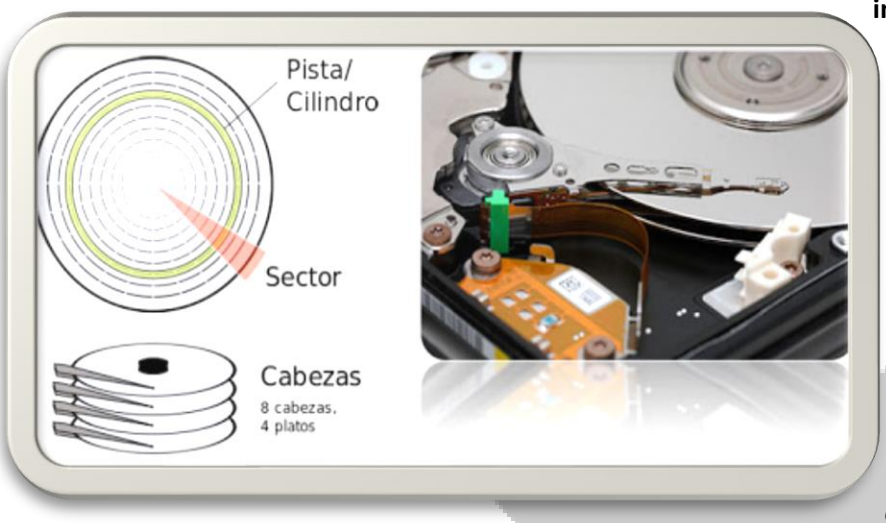
Lo que no abordamos, es la organización de los registros en los archivos.
Generalmente, tratamos con tres tipos de formas en las que podemos organizar los archivos:



Organización secuencial

- ✓ Los datos se almacenan secuencialmente dentro del archivo (uno después del otro), y es **imposible** acceder a los registros de forma directa sin tener que atravesar toda la serie de datos hasta encontrar lo buscado.
  - ✓ En el último registro de esta organización, se encuentra un EOF (End of File) FDA.
  - ✓ Como ventaja al organizar los datos uno tras otro, aprovecha mejor los espacios, siendo muy difícil que deje espacios en blanco o huecos

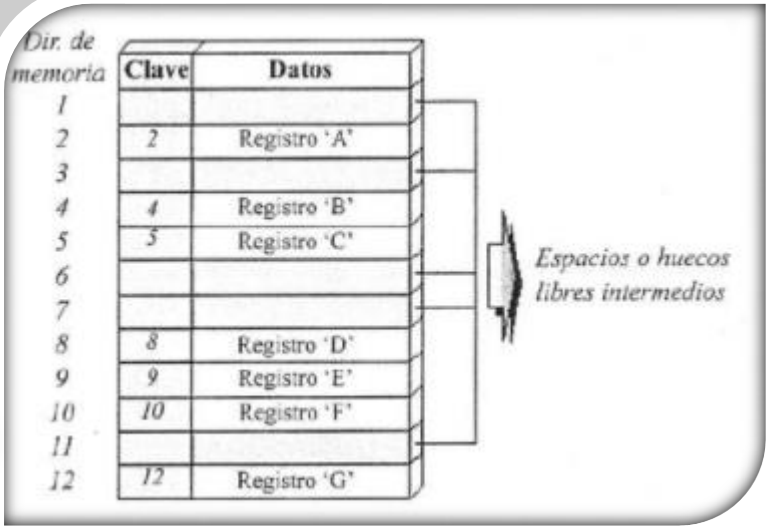
Un ejemplo claro de este tipo de organización se puede visualizar en un disco magnético



Organización directa o aleatoria

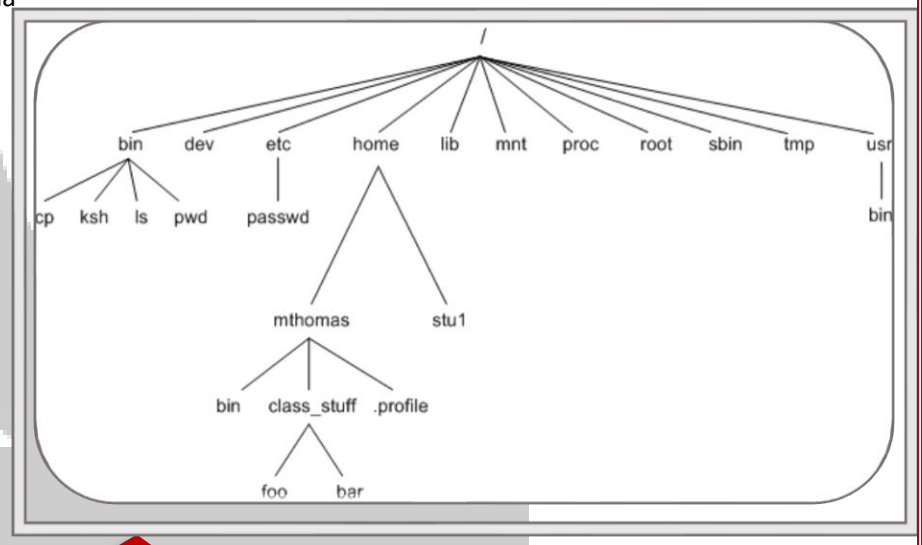
- ✓ Los registros deben tener un campo ***key*** donde se almacena un dato único identificador de este.
- ✓ Orden físico no corresponde al orden lógico de los archivos.
- ✓ Una gran desventaja de este método es que el fichero contiene gran cantidad de huecos o espacios, por lo tanto el algoritmo para la conversión de las claves y el algoritmo necesario para el almacenamiento y tratamiento de sinónimos han de ser creados de modo que dejen el menor número de huecos libres y se genere el menor número de sinónimos.

Como se puede visualizar, entre medio de los registros suelen haber espacios en blanco



Organización secuencial indexada

- ✓ El tipo de sus registros contiene un campo clave identificador.
- ✓ Los registros están situados en un soporte direccionable por el orden de los valores indicados por la clave.
- ✓
- ✓ Un índice para cada posición direccionable, la dirección de la posición y el valor de la clave; en esencia, el índice contiene la clave del último registro y la dirección de acceso al primer registro del bloque.



Unix File System, un claro ejemplo de esta forma de organización

Clasificación de los archivos según su función

Los archivos pueden tener muchas funcionalidades, y todas dependen de cómo lo desee utilizar su usuario, qué datos decida guardar dentro de los registros del mismo, en que contexto lo utilice luego del almacenamiento de datos, etc. Pero para simplificar las subjetividades que puede llegar a generar esta cuestión, existen clasificaciones entabladas objetivamente como las siguientes:

Maestros

- Almacenan datos permanentes o históricos

De movimiento

- Pueden estar clasificados como auxiliares. Suelen contener los registros necesarios para actualizarse a sí mismo como archivos permanentes

De maniobras

- Suelen ser efímeros (de corta duración) y a la vez auxiliares. Contienen información en base a registros seleccionados o semielaborados

De informe

- Contienen datos para ser presentados a lo usuarios

Operaciones sobre los archivos

Los archivos suelen ser permanentes y generalmente de simple acceso al usuario. Una persona desea guardar datos, para luego transformar un conjunto de los mismos en información, y consecuentemente la misma es sumamente valiosa para utilizarla en miles de formas y procesos, y para ello uno debe poder manipular fácilmente su contenido.

Operaciones básicas que se pueden llevar a cabo sobre los archivos

Nombre característico	*Acción*
Apertura	Comunicación del archivo lógico con el físico
Creación	Definición del archivo
Lectura / Consulta	Acceder al archivo para ver su contenido
Fin de Archivo	Detecta el final del archivo
Destrucción	Borra completamente el Archivo
Reorganización	Ordena la información almacenada en los registros para una mayor optimización
Fusión	Reúne varios archivos en uno solo, logrando que de la misma manera los datos almacenados también queden en conjunto
Cierre	Cerrar la conexión

Actualización de archivos

Corresponden al clásico **ABM** (Alta, Baja y Modificación de los archivos o información)

Actualización - Alta	• Adición de los registros dentro del archivo
Actualización - Baja	• Eliminación o borrado lógico de registros
Actualización – Modificación	• Altera información de contenido

FLUJO DE ARCHIVOS

Los flujos determinan el sentido de la comunicación (lectura, escritura, o lectura/escritura), la posibilidad de posicionamiento directo o no en un determinado registro y la forma de operar el archivo. Cerrar el archivo implica cerrar el flujo, por ejemplo.

Los archivos se clasifican en función del flujo de los datos o por el modo de acceso a los datos del archivo. En función de la dirección del flujo de los datos son de:

Entrada

- Aquellos cuyos datos se leen por parte del programa (archivos de lectura)

Salida

- Archivos que escribe el programa (archivos de escritura)

Entrada/Salida

- Archivos en los que se puede leer y escribir

La determinación del tipo de archivo se realiza en el momento de la creación del archivo.

Por ejemplo, aquí podemos observar un algoritmo en Python 3.8 donde el flujo de datos opera en función de Entrada/Salida (write & read) de un archivo “file.txt” ya existente:

```
#!/usr/bin/env python3


# Declaramos la variable archivo donde abrimos file.txt en modo Lectura Y Escritura:
archivo = open("file.txt", mode="w+")
## Es aquí donde podemos apreciar el flujo de datos...
## ... El "mode" w+ declara un flujo de datos de Lectura Y Escritura en Python.


# Escribimos sobre el archivo:
archivo.write("Hola mundo")

# Leemos el contenido del archivo:
contenido = archivo.read()
# Y lo printeamos:
print(contenido)

# Finalmente, cerramos el archivo:
archivo.close()
```

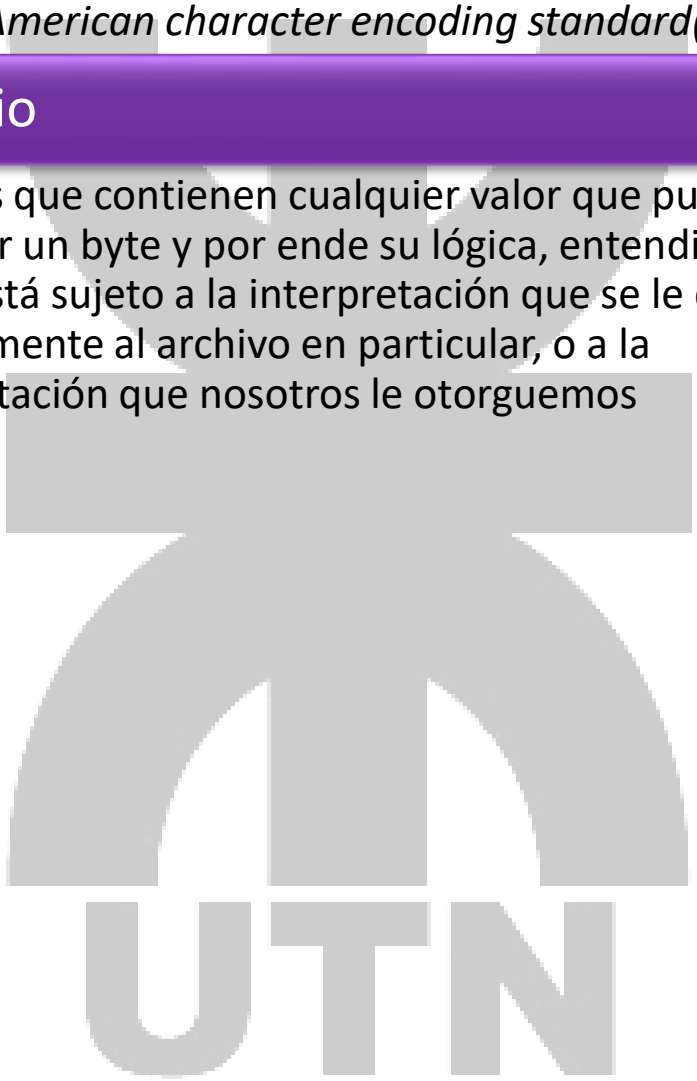
Cuando lidiamos con archivos a un nivel más elevado, nos topamos que se presentan de dos formas:

En texto

- Archivos específicamente construidos para interpretarse bajo el *American character encoding standard(ASCII).*

En binario

- Archivos que contienen cualquier valor que pueda contener un byte y por ende su lógica, entendimiento de estos, está sujeto a la interpretación que se le dio originalmente al archivo en particular, o a la interpretación que nosotros le otorguemos



Presentación del Sistema de Información

Nuestro prototipo del sistema de información cuenta con 4 solapas: “Personal”,” Habitaciones”, “Reservas” y “Pasajeros”, cada una de ellas indispensables ya sea para el recepcionista, como el encargado de reserva y el encargado de personal del hotel.

En su pantalla principal cuenta con las estadísticas que se actualizan de forma automática dependiendo de los ingresos de información que se producen en la Base de Datos referentes a las edades de los pasajeros registrados, como las diferentes nacionalidades o proveniencias de los mismo.

Hemos implementado el uso de muchas herramientas para completar el sistema de información que se va a describir a continuación. Se ha desarrollado a la completitud el sistema con las siguientes tecnologías:

- ✓ PHP
- ✓ HTML
- ✓ CSS
- ✓ JavaScript
- ✓ JQuery (librería)
- ✓ CanvasJS (librería)
- ✓ Bootstrap (FrameWork)
- ✓ MariaDB
- ✓ phpMyAdmin
- ✓ SQL

Facilitamos una web para la prueba de nuestro software donde se puede interactuar con el sistema sin necesidad de instalar nada. Si quieres utilizar este sistema en tu propio servidor, debes instalar una distribución de Apache mínimamente con PHP y una base de datos MySQL para su funcionamiento correcto.

Demostración del código

```html
<!DOCTYPE HTML>

<html>
    <head>
        <title>Empleados - Sol de las Sierras</title>
    </head>

    <body>
        <div class="card m-3">
            <div class="card-header">
                Administrador de Empleados
            </div>
            <div class="card-body">
                <p>En esta sección se pueden agregar, modificar, eliminar o revisar empleados del hotel
registrados en la base de datos.</p>
            </div>
        </div>
    </body>
</html>
```

HTML - Descripción de la Sección Empleados y Head

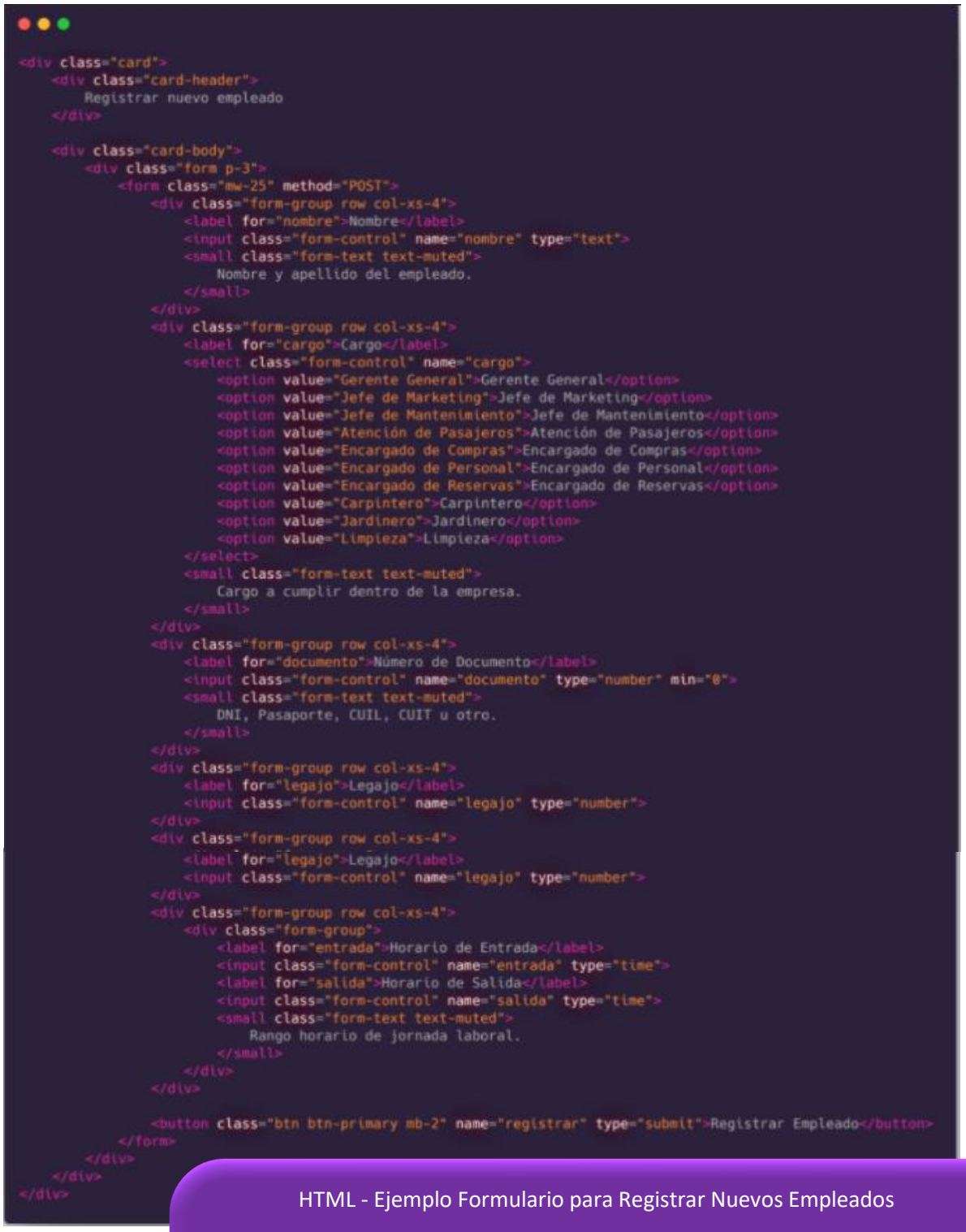
Dentro de la etiqueta <head> generalmente se ingresan datos no visibles al usuario, y dentro de <body> se ingresan todos los elementos visuales de la página. Aquí solo se construyó un encabezado para la pestaña de empleados, y un cuadro de texto con una pequeña descripción

```html
<div class="col">
    <div class="card">
        <div class="card-header">
            Vista de Empleados
        </div>

        <div class="card-body">
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">Nombre</th>
                        <th scope="col">Cargo</th>
                        <th scope="col">Documento</th>
                        <th scope="col">Legajo</th>
                        <th scope="col">Entrada</th>
                        <th scope="col">Salida</th>
                        <th scope="col">Acciones</th>
                    </tr>
                </thead>
                <tbody>
                  <!-- Vacío. Aquí irían los registros de los empleados para
                    rellenar la tabla -->
                </tbody>
            </table>
        </div>
    </div>
</div>
```

HTML - Tabla de la Sección Empleados

Aquí se ensambla la tabla donde se visualizarán en un futuro todos los datos de los empleados ingresados al sistema. La misma se conforma generando una etiqueta principal <table>, y luego dentro de la misma mediante <thead> se nos sitúa en la cabecera de la tabla, <tr> para crear una fila, y finamente con <th> se generan los campos uno por uno, indicando su contenido



```
<div class="card">
    <div class="card-header">
        Registrar nuevo empleado
    </div>

    <div class="card-body">
        <div class="form p-3">
            <form class="mw-25" method="POST">
                <div class="form-group row col-xs-4">
                    <label for="nombre">Nombre</label>
                    <input class="form-control" name="nombre" type="text">
                    <small class="form-text text-muted">
                        Nombre y apellido del empleado.
                    </small>
                </div>
                <div class="form-group row col-xs-4">
                    <label for="cargo">Cargo</label>
                    <select class="form-control" name="cargo">
                        <option value="Gerente General">Gerente General</option>
                        <option value="Jefe de Marketing">Jefe de Marketing</option>
                        <option value="Jefe de Mantenimiento">Jefe de Mantenimiento</option>
                        <option value="Atención de Pasajeros">Atención de Pasajeros</option>
                        <option value="Encargado de Compras">Encargado de Compras</option>
                        <option value="Encargado de Personal">Encargado de Personal</option>
                        <option value="Encargado de Reservas">Encargado de Reservas</option>
                        <option value="Carpintero">Carpintero</option>
                        <option value="Jardinero">Jardinero</option>
                        <option value="Limpieza">Limpieza</option>
                    </select>
                    <small class="form-text text-muted">
                        Cargo a cumplir dentro de la empresa.
                    </small>
                </div>
                <div class="form-group row col-xs-4">
                    <label for="documento">Número de Documento</label>
                    <input class="form-control" name="documento" type="number" min="0">
                    <small class="form-text text-muted">
                        DNI, Pasaporte, CUIL, CUIT u otro.
                    </small>
                </div>
                <div class="form-group row col-xs-4">
                    <label for="legajo">Legajo</label>
                    <input class="form-control" name="legajo" type="number">
                </div>
                <div class="form-group row col-xs-4">
                    <label for="legajo">Legajo</label>
                    <input class="form-control" name="legajo" type="number">
                </div>
                <div class="form-group row col-xs-4">
                    <div class="form-group">
                        <label for="entrada">Horario de Entrada</label>
                        <input class="form-control" name="entrada" type="time">
                        <label for="salida">Horario de Salida</label>
                        <input class="form-control" name="salida" type="time">
                        <small class="form-text text-muted">
                            Rango horario de jornada laboral.
                        </small>
                    </div>
                </div>

                <button class="btn btn-primary mb-2" name="registrar" type="submit">Registrar Empleado</button>
            </form>
        </div>
    </div>
</div>
```

HTML - Ejemplo Formulario para Registrar Nuevos Empleados

Aquí se puede visualizar un ejemplo de como realizamos los diferentes formularios que se encuentran en nuestro sistema. Mediante la etiqueta <form> se indica que lo que se encuentre dentro va a ser parte de un formulario de ingreso de datos. Para cada entrada se utiliza la etiqueta <input> y dentro de la misma se indica a que tipo de entrada pertenece (número, correo electrónico, texto simple, etc). En este conjunto de entradas se utilizó la etiqueta <select>, la cual crea una casilla en forma de lista desplegable con diferentes opciones a elegir, definidas cada una por <option>

```
<script>
    window.onload = function() {
        let countries = [],
            ages      = [];

        function PushData(arr, toArr) {
            let coun = [];

            arr.forEach(function (e) {
                if (!coun.includes(e))
                    coun.push(e);
            });

            let nums = [];
            for (let i = 0; i < coun.length; i++)
                nums[i] = 0;

            arr.forEach(function (e) {
                nums[coun.indexOf(e)] += 1;
            });

            for (let i = 0; i < coun.length; i++) {
                toArr.push({
                    y: nums[i],
                    label: coun[i]
                });
            }
        }

        $.getJSON("link_a_la_pagina_con_el_json", function (data) {
            PushData(data, countries);

            let coChart = new CanvasJS.Chart('passanger-countries', {
                animationEnable: true,
                theme: 'light2',
                zoomEnabled: true,
                title: {
                    text: 'Paises de Proveniencia de Pasajeros Registrados'
                },
                axisY: {
                    title: 'Cantidad'
                },
                data: [{
                    type: 'column',
                    yValueFormatString: '# pasajero(s)',
                    dataPoints: countries
                }]
            });

            coChart.render();
        });
    }
</script>
```

JavaScript - Script para Renderizado de Gráficas Estadísticas en CanvasJS

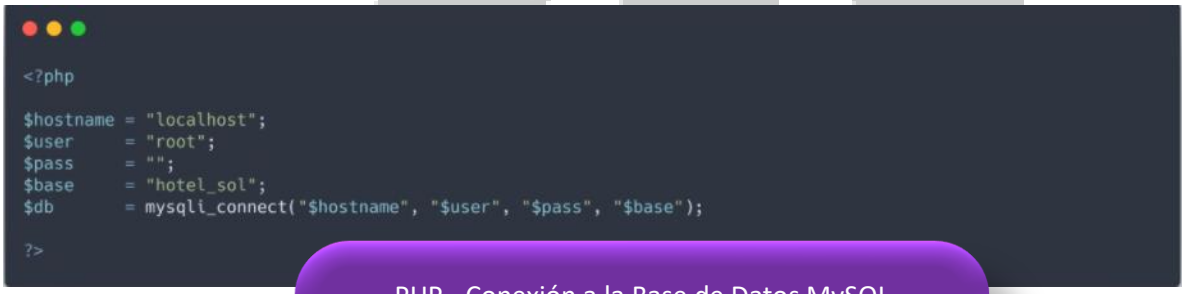
Esta parte del código básicamente lo que hace es tomar un JSON de las dos API REST y convierte los datos contenidos en el mismo en un formato aceptado por CanvasJS, y asi poder generar debidamente la gráfica presente en la página principal del sistema

JSON - API REST 1

Representan el formato en el que llegan los datos provenientes de la API REST de los países de proveniencia de los pasajeros

JSON - Convertido para CanvasJS

Representa el JSON convertido para ser interpretado finalmente por el CanvasJS



PHP - Conexión a la Base de Datos MySQL

Acá se puede visualizar como nos podemos conectar generalmente a una Base de Datos perteneciente a MySQL para realizar el almacenamiento de la información generada por el sistema

```php
<nav class="navbar navbar-expand-lg navbar-light bb" style="background-color: #549fcf; border-bottom: 5px solid #3e87bd;">
    <a class="navbar-brand" href="/">
        <img src="img/logo.svg" width="60px" data-toggle="tooltip" title="Home">
    </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-
controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <ul class="navbar-nav mr-2">
            <li class="nav-item active ml-5" data-toggle="tooltip" title="Empleados">
                <a href="empleados.php">
                    <img src="img/empleados.svg" width="50px">
                </a>
            </li>
            <li class="nav-item active ml-5" data-toggle="tooltip" title="Habitaciones">
                <a href="habitaciones.php">
                    <img src="img/habitaciones.svg" width="50px">
                </a>
            </li>
            <li class="nav-item active ml-5" data-toggle="tooltip" title="Pasajeros">
                <a href="pasajeros.php">
                    <img src="img/pasajeros.svg" width="50px">
                </a>
            </li>
            <li class="nav-item active ml-5" data-toggle="tooltip" title="Reservas">
                <a href="reservas.php">
                    <img src="img/reservas.svg" width="50px">
                </a>
            </li>
        </ul>
    </div>
</nav>
```

PHP - Ejemplo de un Módulo (Barra Navegable)

Parte del código donde se representa el armado de una barra navegable (la que se encuentra presente en la parte superior de la página en todo momento). La misma fue ensamblada principalmente con la biblioteca Bootstrap (muy utilizada en la creación de páginas webs por su simplicidad en el armado de interfaces y parte frontal de la misma). Además, cada botón está conectado a su respectivo archivo .php, en el cual dentro del mismo se encuentra toda la codificación encargada de llevar a cabo la parte “backend” o funcional del sistema

```php
<?php
    include("mod-header.php");
?>
```

PHP - Ejemplo de la Importación de un Módulo

Asi se importan los módulos dentro de las páginas, mediante include()

```php
<?php
    $q = "SELECT * FROM empleados";
    $r = mysqli_query($db, $q);

    if ($r) {
        while ($row = $r->fetch_array()) {
            $id     = $row['id'];
            $name   = $row['nombre'];
            $cargo  = $row['cargo'];
            $dni    = $row['documento'];
            $leg    = $row['legajo'];
            $en     = date('h:i A', strtotime($row['inicio_jornada']));
            $sa     = date('h:i A', strtotime($row['final_jornada']));

        ?>
            <tr>
                <td><?php echo $name; ?></td>
                <td><?php echo $cargo; ?></td>
                <td><?php echo $dni; ?></td>
                <td><?php echo $leg; ?></td>
                <td><?php echo $en; ?></td>
                <td><?php echo $sa; ?></td>
                <td>
                    <button type="button" class="btn" data-toggle="modal" data-target="#delete-modal"
                            data-id="<?php echo $id; ?>"
                            data-name="<?php echo $name; ?>">
                        <img src="img/icon/delete.svg" width="25px">
                    </button>
                    <button type="button" class="btn" data-toggle="modal" data-target="#edit-modal"
                            data-id="<?php echo $id; ?>"
                            data-name="<?php echo $name; ?>"
                            data-cargo="<?php echo $cargo; ?>"
                            data-dni="<?php echo $dni; ?>"
                            data-legajo="<?php echo $leg; ?>"
                            data-entry="<?php echo $row['inicio_jornada']; ?>"
                            data-exit="<?php echo $row['final_jornada']; ?>">
                        <img src="img/icon/edit.svg" width="25px">
                    </button>
                </td>
            </tr>
        <?php
        }
    }
?>
```

PHP - Ejemplo de Query MySQL para Imprimir Datos en la Aplicación

Pide datos de la base, los trae, itera sobre todos los resultados y los imprime en código HTML para luego mostrarlos en la tabla de empleados que se puede visualizar en la página

```php
<?php

  if (isset($_POST['editUser'])) {
      $id     = $_POST['id'];
      $name   = $_POST['nombre'];
      $cargo  = $_POST['cargo'];
      $dni    = $_POST['documento'];
      $legajo = $_POST['legajo'];
      $en     = $_POST['entrada'];
      $sa     = $_POST['salida'];

      $q = "UPDATE empleados SET nombre = '$name', cargo = '$cargo', documento = '$dni', legajo = $legajo, " .
           "inicio_jornada = '$en', final_jornada = '$sa' " .
           "WHERE id = $id";
      mysqli_query($db, $q);

      echo "<meta http-equiv='refresh' content='0'>";
  }

  if (isset($_POST['deleteUser'])) {
      $id = $_POST['id'];

      $q = "DELETE FROM empleados WHERE id = $id";
      mysqli_query($db, $q);

      echo "<meta http-equiv='refresh' content='0'>";
  }

  if (isset($_POST['registrar'])) {
      $name   = $_POST['nombre'];
      $cargo  = $_POST['cargo'];
      $dni    = $_POST['documento'];
      $legajo = $_POST['legajo'];
      $en     = $_POST['entrada'];
      $sa     = $_POST['salida'];

      $q = "INSERT INTO empleados (nombre, cargo, documento, legajo, inicio_jornada, final_jornada) " .
           "VALUES ('$name', '$cargo', '$dni', $legajo, '$en', '$sa')";
      mysqli_query($db, $q);

      echo "<meta http-equiv='refresh' content='0'>";
  }

?>
```

PHP - Detección de Envío de Formularios e Inserción en la Base de Datos de los mismos

Aquí lo que se hace es detectar cuando el usuario del sistema realizó un envío de información a través de un formulario; posteriormente a ello, los datos son ingresados dentro de la Base de Datos perteneciente a la pestaña en la cual se encuentre. Además de registrar datos, también se realiza las actualizaciones y/o eliminaciones de los mismos dentro de la BBDD

```sql
-- Insertar datos en una tabla de la base de datos
INSERT INTO empleados (nombre, apellido) VALUES ("Albertx", "Fernández")

-- Modificar datos existentes de una tabla de la base de datos
UPDATE empleados SET nombre = "Alberto" WHERE apellido = "Fernández"

-- Eliminar un registro específico de la base de datos
DELETE FROM empleados WHERE nombre = "Alberto"

-- Obtener datos de la base de datos

SELECT nombre FROM empleados WHERE apellido = "Fernández"
-- El resultado es una fila con una única columna llamada "nombre" que tiene como valor "Alberto"

SELECT * FROM empleados
-- El resultado es absolutamente todo el contenido de la tabla "empleados".
```

MySQL - Queries de Ejemplo y Comandos que utilizamos en el prototipo

Aquí decidimos mostrar a modo de ejemplificaciones los comandos que se suelen utilizar para realizar trabajos con MySQL de forma externa al mismo, ya sea para realizar ingresos de información nueva, eliminar registros determinados de la tabla, actualizar los mismos, o seleccionar un registro en particular para traerlo al programa para visualizar sus datos