



CRUD

Python - MySQL - Flask



Retrospectiva del proyecto

CRUD

Proyecto de gestión de datos

CRUD (acrónimo): Gestión de bases de datos digitales.

- **Create** (Crear registros)
- **Read/Retrieve** (Leer/recuperar registros)
- **Update** (Actualizar registros)
- **Delete/Destroy** (Borrar registros)

CRUD

Proyecto de gestión de datos

El patrón MVC: Modelo-Vista-Controlador (*Model-View-Controller*), que son las capas o grupos de componentes en los que organizaremos nuestras aplicaciones bajo este paradigma.

- **Modelo de datos:** En la capa **Modelo** encontraremos siempre una representación de los datos del dominio, es decir, aquellas entidades que nos servirán para almacenar información del sistema que estamos desarrollando. En este caso son los datos que almacenaremos en la base de datos.
- **Vista:** Los componentes de la **Vista** son los responsables de generar la interfaz de nuestra aplicación, es decir, de componer las pantallas, páginas, o cualquier tipo de resultado utilizable por el usuario o cliente del sistema.
- **Controlador:** La misión principal de los componentes incluidos en el **Controlador** es actuar como *intermediarios entre el usuario y el sistema*. Serán capaces de capturar las acciones de éste sobre la Vista, como puede ser la pulsación de un botón o la selección de una opción de menú, interpretarlas y actuar en función de ellas. En el controlador se gestionan las peticiones de la app Web.

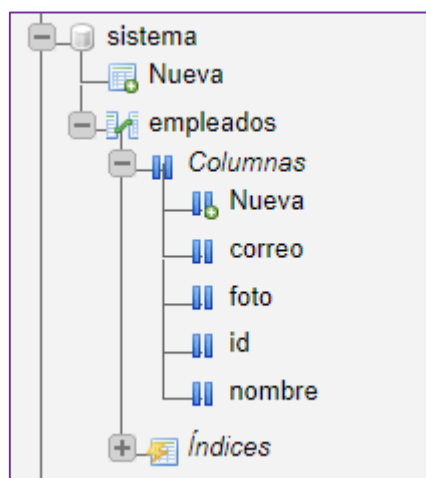
Para ampliar: <https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx>

CRUD | Partes del proyecto

Modelo de datos

La base de datos “sistema”, creada con *phpMyAdmin*, contiene una sola tabla llamada **empleados**. Sus campos son:

- **Id** (*int*): Clave principal, autonumérico, longitud 10
- **Nombre** (*varchar*): longitud 255
- **Correo** (*varchar*): longitud 255
- **Foto** (*varchar*): longitud 5000



id	nombre	correo	foto
1	Héctor	hector@empresa.com	2021145938señor1.jpg
2	Ana	ana@empresa.com	2021150039señora1.jpg
3	Luis	luis@empresa.com	2021150055señor2.jpg
4	Laura	laura@empresa.com	2021150118señora2.jpg
5	Fernando	fernando@empresa.com	2021150141señor3.jpg
6	Emilia	emilia@empresa.com	2021150216señora3.jpg




CRUD | Partes del proyecto

Vistas

Gestión de empleados Empleados

Ingresar nuevo empleado

index.html

#	Foto	Nombre	Correo	Acciones
1		Héctor	hector@empresa.com	Editar Eliminar
2		Ana	ana@empresa.com	Editar Eliminar
3		Luis	luis@empresa.com	Editar Eliminar

La página principal (*index.html*) muestra el listado de la tabla empleados y permite acceder a crear un nuevo registro.

Cada uno de los registros puede ser modificado (*edit.html*) o eliminado. Tras esta última acción se redireccionará a la página *index.html*

Nota: Las tres vistas comparten *header* y *footer*.

Gestión de empleados Empleados

Ingresar empleados

create.html

Datos del empleado

Nombre:

Correo:

Foto:

[Seleccionar archivo](#) Ningún archivo seleccionado

[Agregar](#) [Regresar](#)

Gestión de empleados Empleados

Formulario para editar

Editar empleado

edit.html


Datos del empleado

ID:

Nombre:

Correo:

Foto:



[Seleccionar archivo](#) Ningún archivo seleccionado

[Modificar](#) [Regresar](#)

CRUD | Partes del proyecto

Controlador

```
app.py > ...
1  from flask import Flask
2  from flask import render_template, request, redirect, url_for, flash
3  from flaskext.mysql import MySQL
4  from datetime import datetime
5  import os
6  from flask import send_from_directory
7
8  app = Flask(__name__)
9  app.secret_key="ClaveSecreta"
10
11  mysql = MySQL()
12  app.config['MYSQL_DATABASE_HOST']='localhost'
13  app.config['MYSQL_DATABASE_USER']='root'
14  app.config['MYSQL_DATABASE_PASSWORD']=''
15  app.config['MYSQL_DATABASE_BD']='sistemados'
16  mysql.init_app(app)
17
18  CARPETA= os.path.join('uploads')
19  app.config['CARPETA']=CARPETA
20
21  @app.route('/uploads/<nombreFoto>')
22  def uploads(nombreFoto):
23      return send_from_directory(app.config['CARPETA'], nombreFoto)
```

Aquí tendremos el código que nos permitirá realizar las acciones del CRUD, además de la importación de las librerías necesarias, la conexión con la base de datos y una validación básica de los datos.

*El controlador responderá a **eventos** (acciones del usuario) e invocará peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un registro en una base de datos). También puede **enviar comandos** a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo', por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'*

CRUD | Elementos necesarios

Aplicaciones, paquetes y extensiones

- Python (por instalador desde <https://www.python.org/downloads/>)
- Visual Studio Code (por instalador desde <https://code.visualstudio.com/>)
- XAMPP (por instalador desde <https://www.apachefriends.org/es/index.html>)
- Flask (desde la terminal de VSC con **pip install flask**)
- MySQL (desde la terminal de VSC con **pip install Flask-MySQL**)
- Jinja2 (desde la terminal de VSC con **pip install jinja2**)
- Bootstrap v4 Snippets ([más info](#))
- Flask Snippets ([más info](#))
- Flask-snippets ([más info](#))
- Jinja2 Snippet Kit ([más info](#))
- Palenight Theme ([más info](#)) y Andromeda ([más info](#)) (opcionales)
- [phpMyAdmin](#) (gestor de bases de datos web)

pip: sistema de gestión de paquetes sencillo utilizado para la instalación y administración de paquetes ([más info](#))

CRUD | Elementos necesarios

Librerías necesarias

- **De flask:**
 - **Flask:** Permite importar el framework
 - **render_template:** Permite mostrar los templates (páginas index, edit y create)
 - **request:** Nos servirá para acceder a la información que nos envíe el cliente (nombre, correo, foto, ID)
 - **redirect:** Permite redirigir a un usuario de una pagina a otra (utilizado cuando se hacen modificaciones de datos)
 - **url_for:** Recibe como parámetro el nombre del método y nos devuelve la ruta para redireccionar
 - **flash:** Se utiliza para generar mensajes informativos
 - **send_from_directory:**
Permite buscar si el archivo está en el directorio. Se le pasan dos argumentos: un directorio y un nombre de fichero
- **De flaskext.mysql**
 - MySQL Nos permite conectarnos a la BD MySQL
- **De datetime**
 - datetime: Proporciona clases para manipular fechas y horas (nos permitirá darle el nombre a la foto)
- **De os:** Permite acceder a funcionalidades dependientes del SO, en nuestro caso para trabajar con archivos y directorios

CRUD | Funciones



C

Create

```
def create()  
def storage()
```

- **def create():** Redirecciona a la página **create.html**
- **def storage():** Permitirá almacenar los datos en la tabla, redireccionando a la página **index.html**

R

Read

```
def index()
```

- **def index():** Permitirá desplegar los datos de la tabla, retornando el template **index.html**

U

Update

```
def edit(id)  
def update()  
def uploads(nombreFoto)
```

- **def edit(id):** Mostrará los datos de la tabla para ser modificados, retornando el template **edit.html**
- **def update():** Actualizará los datos de la tabla enviados al formulario por **def edit()**, redireccionando a la página **index.html**
- **def uploads(nombreFoto):** Enviará el archivo de la foto a la carpeta uploads.

D

Delete

```
def destroy(id)
```

- **def destroy(id):** Eliminará un registro a partir de su ID, redireccionando a la página **index.html**

CRUD | Funciones | Create

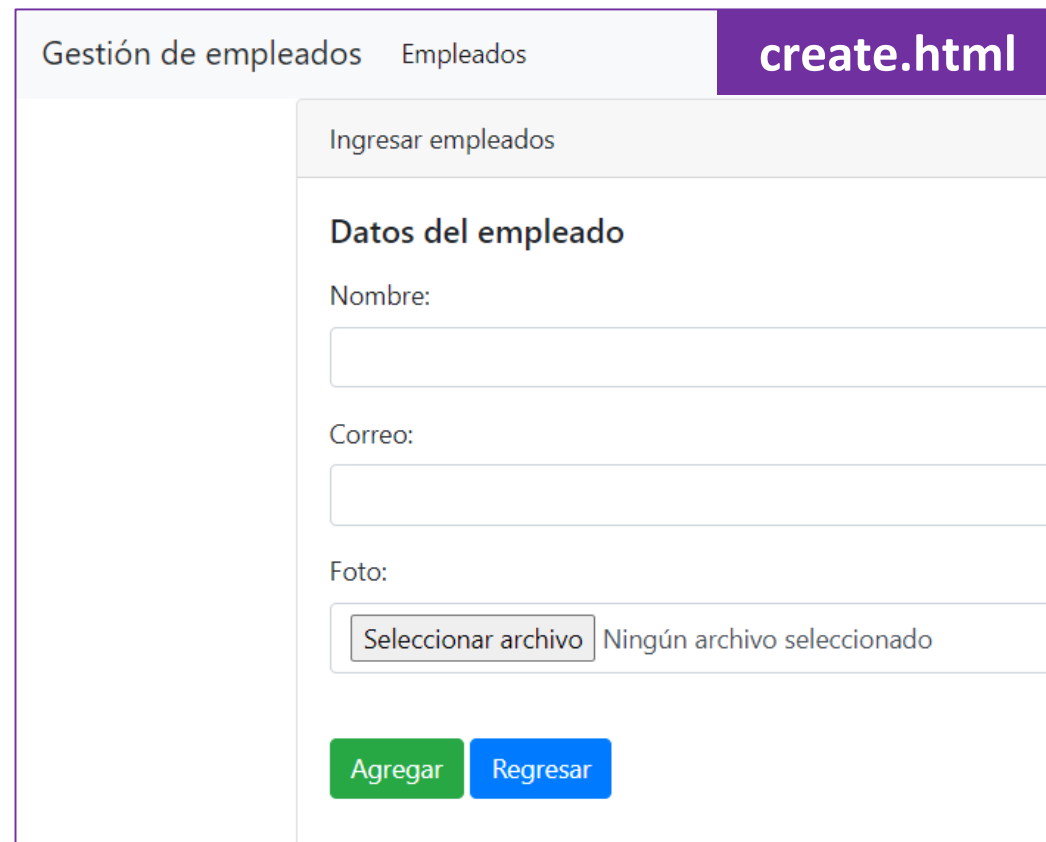
Create

Ingresar nuevo empleado

- Para dar de alta un empleado accedemos a **Ingresar nuevo empleado**. La función **def create()** se encarga de desplegar la página **create.html**.
- La función **def storage()** es la que permite almacenar los datos en la base de datos, básicamente de esta manera:
 - Toma los datos del formulario.
 - Valida si se cargaron todos los datos.
 - Guarda la foto cargada en la carpeta uploads.
 - Genera la sentencia SQL de inserción.
 - Realiza la conexión con la BD
 - Ejecuta la consulta y cierra la conexión

Esta función se activa con el botón **Agregar** y luego retorna a la página **index.html**

El botón Regresar permite volver a **index.html**.



The screenshot shows a web application interface for managing employees. At the top, there's a navigation bar with 'Gestión de empleados' and 'Empleados' on the left, and 'create.html' on the right. Below this, the main content area is titled 'Ingresar empleados'. Underneath, there's a section 'Datos del empleado' with three input fields: 'Nombre:', 'Correo:', and 'Foto:'. The 'Foto:' field has a file selection button labeled 'Seleccionar archivo' and a status message 'Ningún archivo seleccionado'. At the bottom of the form, there are two buttons: 'Agregar' (green) and 'Regresar' (blue).

CRUD | Funciones | Read




Read

- La función **def index()** se encarga de mostrar la información de la tabla Empleados una vez que se carga la página principal (**index.html**), básicamente de esta manera:
 - Genera la sentencia SQL de selección de todos los registros
 - Realiza la conexión con la BD
 - Ejecuta la consulta
 - Recupera las filas del resultado de la consulta
 - Cierra la conexión

Gestión de empleados Empleados

index.html


Ingresar nuevo empleado

#	Foto	Nombre	Correo	Acciones
1		Héctor	hector@empresa.com	Editar Eliminar
2		Ana	ana@empresa.com	Editar Eliminar
3		Luis	luis@empresa.com	Editar Eliminar

CRUD | Funciones | Update

Update

- Para actualizar un registro debemos hacerlo desde el botón **Editar** que le corresponde y se encuentra a la misma altura en la última columna, junto con el botón Eliminar.
- La operación de actualización está dividida en dos partes:
 - a) El despliegue de los datos que se desean modificar dentro del formulario de edición, a cargo de la función **def edit(id)**
 - b) La actualización propiamente dicha, que modificará los datos de la tabla Empleados, a cargo de la función **def update()**

#	Foto	Nombre	Correo	Acciones
1		Héctor	hector@empresa.com	<div>Editar</div> <div>Eliminar</div>

Gestión de empleados Empleados

edit.html

Formulario para editar


Editar empleado

Datos del empleado

ID:

Nombre: Héctor

Correo: hector@empresa.com

Foto: 

Seleccionar archivo

Ningún archivo seleccionado

Modificar

Regresar


CRUD | Funciones | Update

Update

a) **def edit(id)** se encargará de mostrar los datos del registro a modificar, básicamente de esta manera:

- Realiza la conexión con la BD
- Ejecuta la consulta SQL de selección tomando como criterio el id recibido como argumento
- Recupera las filas del resultado de la consulta
- Cierra la conexión
- Despliega la página **edit.html** que muestra en el formulario los datos del registro.

El usuario se encargará de modificar el nombre, el correo y la foto, seleccionando un nuevo archivo y luego presionando el botón **Modificar**. También es posible presionar **Regresar** sin realizar cambios.

#	Foto	Nombre	Correo	Acciones
1		Héctor Luis	hectorluis@empresa.com	<div> <div>Editar</div> <div>Eliminar</div> </div>

Gestión de empleados

Empleados

edit.html

Formulario para editar


Editar empleado

Datos del empleado

ID:

Nombre: Héctor Luis

Correo: hectorluis@empresa.com

Foto: 

Seleccionar archivo

señor5.jpg

Modificar

Regresar

CRUD | Funciones | Update

Update

b) **def update()** actualizará los datos en la tabla Empleados una vez que se presione el botón **Modificar**, básicamente de esta manera:

Modificar

- Toma los datos del formulario (id, nombre, correo y foto)
- Genera la sentencia SQL de actualización (nombre y correo)
- Realiza la conexión con la BD
- Genera un string que guarda el año, hora, minuto y segundo de actualización de la foto
- Pregunta si la foto también fue actualizada y si es así realiza lo siguiente:
 - Crea un nuevo nombre para la foto (string + nombre) y la guarda
 - Ejecuta la sentencia SQL de selección de la foto anterior y recupera la fila del resultado de la consulta
 - Elimina el archivo de la foto anterior
 - Ejecuta la consulta de actualización solamente para la foto nueva
 - Cierra la conexión
- Si la foto no fue actualizada ejecuta la consulta SQL de actualización (nombre y correo)
- Cierra la conexión y redirecciona a **index.html**

Nota: La consulta no contempla la actualización del Id ya que no es necesario, es un campo autonumérico.

c) **def uploads(nombreFoto):** Nos habilita el acceso a la carpeta uploads, con `send_from_directory`, que nos permite enviar un archivo específico desde un el directorio uploads.

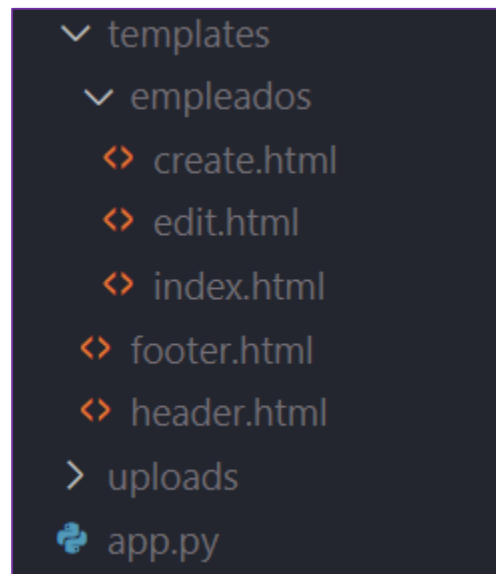
CRUD | Funciones | Delete

Delete

- Para dar de baja un registro debemos hacerlo desde el botón **Eliminar** que le corresponde al empleado y se encuentra a la misma altura en la última columna, junto con el botón Editar.
- La función **def destroy(id)** es la que permite eliminar el registro deseado, a partir del id pasado como argumento, básicamente de esta manera:
 - Realiza la conexión con la BD
 - Ejecuta la consulta SQL de selección tomando como criterio el id recibido como argumento (para la foto)
 - Recupera la fila del resultado de la consulta
 - Elimina el archivo de la foto
 - Ejecuta la consulta SQL de eliminación tomando como criterio el id recibido como argumento
 - Cierra la conexión

Tras presionar **Eliminar** se retorna a la página **index.html**, de esta manera no solamente se elimina el registro sino también la foto asociada al registro en la carpeta uploads.

CRUD | Estructura del proyecto y vistas



Templates: Contiene los archivos **header.html** y **footer.html** que contendrán el encabezado y pie de página que se cargarán como plantilla en las páginas **index.html**, **create.html** y **edit.html**.

Además contiene la carpeta **empleados** que, a su vez, contiene las páginas anteriormente nombradas.

Uploads: Guarda las imágenes que corresponden a cada registro una vez que son creadas o modificadas.

Header y Footer

header.html contiene el encabezado de las páginas y parte del cuerpo. En el encabezado se incluye la referencia a Bootstrap. En el cuerpo se incluye la barra de navegación y la apertura del contenedor div dentro del cual se desplegará la tabla de empleados y los formularios.

footer.html contiene el cierre del contenedor, y los cierres del cuerpo y el documento HTML.

CRUD | Estructura del proyecto y vistas

Index

Además de incluir el encabezado (header.html) y pie (footer.html) esta vista muestra una tabla donde se despliegan los datos de la tabla **Empleados**, incluyendo un ciclo **for** que recorre la tabla para mostrar cada registro, uno por cada fila de la tabla. Cada registro, además, contiene los botones Editar y Eliminar que se corresponden con las acciones de actualizar y eliminar el registro.

Create

Además de incluir el encabezado (header.html) y pie (footer.html) esta vista muestra una *card* de Bootstrap que permite agregar un nuevo empleado a través de un formulario, con un mensaje de alerta que aparecerá en caso de que no se incluyan datos obligatorios. En el formulario deberán completarse el nombre, el correo y seleccionar una fotografía.

Edit

Además de incluir el encabezado (header.html) y pie (footer.html) esta vista muestra una *card* de Bootstrap que permite modificar los datos de un empleado existente a través de un formulario. En el formulario se muestran el nombre, el correo y seleccionar la fotografía, pudiendo ser modificados y confirmados a través del botón Modificar.