INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1 - D

# MANUAL TÉCNICO

RODRIGO ALEJANDRO HERNÁNDEZ DE LEÓN 201900042

U-BLOG

# Contenido

I. Introducción....2
II. Objetivos....2
III. Dirigido....2
IV. Especificación Técnica....3
1. Requisitos de Hardware....3
2. Requisitos de Software....3
V. Lógica del Programa....4
VI. Flujo de la Aplicación....12
Comunicación entre Frontend y Backend....12
VII. Créditos....12

# I. Introducción

La aplicación web de UBlog esta hecha por medio de un frontend desarrollado con html y css para la buena experiencia del usuario al usar la aplicación y con JavaScript para la funcionalidad de los botones y ciertas funciones para la conexión de comunicación con la api; y backend que fue desarrollada en Python donde estaba la parte lógica de toda la pagina web. En el desarrollo de la API se tuvo que utilizar el paradigma de programación orientada a objetos donde fue muy útil para las respuestas hacia el servidor.

# II. Objetivos

El objetivo de este manual es hacer que los desarrolladores que se encuentren en el campo de la programación web puedan comprender la forma en que fue desarrollada la API para hacer que funcione el frontend de mejor manera.

El objetivo del sistema es poder aprender a desarrollar por medio de un framework que es Flask y fue utilizado en Python para poder levantar la API y poder ser utilizada para las distintas operaciones lógicas del programa y no tener ningún problema con las respuestas enviadas al frontend. Asimismo, poder aprender también del uso de Cloud Computing ya que esta API así como su frontend esta subida en la plataforma de Heroku que es útil para el alcance de todo el mundo usar esta aplicación.

# III. Dirigido

Este manual esta orientado hacia aquellos desarrolladores enfocados en programación web y especialmente en el desarrollo de API's para el funcionamiento del backend y así mismo a los auxiliares de catedra para entender el funcionamiento de las aplicaciones de los estudiantes.

# IV. Especificación Técnica

## 1. Requisitos de Hardware

- Computadora de escritorio o portátil.
- Mínimo 4GB de Memoria RAM.
- 250 GB de almacenamiento de Disco Duro o superior.
- Procesador Intel Core i3 o superior.
- Resolución grafica mínimo 1024 x 768 pixeles.

## 2. Requisitos de Software

- Sistema Operativo Windows 7 o superior.
- Tener instalado Python en la versión 3.9.2
- Tener instalada la librería Flask.
- Tener instalada la librería Flask_cors.
- Editor de Texto como Visual Studio Code.
- Lector de PDF como Acrobat Reader.
- Cuenta registrada en Heroku.

# V. Lógica del Programa

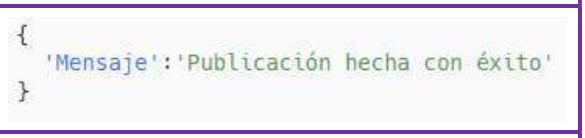
<table>
<tr><th>METODO</th><th>DIRECCION Y TIPO DE MÉTODO</th><th>BODY</th><th>RESPUESTA</th></tr>
<tr><td rowspan="3">Registrar Usuario<br><br>Recibe un JSON con los datos que el usuario ingresa desde el frontend, verifica si el usuario esta en uso y también le validación si su contraseña es válida, si todo está bien entonces registra al usuario agregando un nuevo objeto de tipo Usuario y lo agrega dentro del arreglo de Usuarios, de lo contrario retorna mensajes de error para el ingreso correcto de datos.</td><td>/Usuarios/Registrar</td><td rowspan="3"><pre>{
    "email":"rod@gmail.com",
    "gender":"m",
    "name":"Rodrigo Hernández",
    "password":"Contraseña1234@",
    "username":"rodri.ale"
}</pre></td><td><pre>{
    "Mensaje": "Se agregó el usuario correctamente",
    "estado": "true"
}</pre></td></tr>
<tr><td>POST</td><td><pre>{
    "Mensaje": "Ingrese un usuario existente",
    "estado": "false"
}</pre></td></tr>
<tr><td><pre>{
    "Mensaje": "Su contraseña debe ser mayor a 8 caracteres, contener un número y un símbolo",
    "estado": "false"
}</pre></td></tr>
<tr><td rowspan="2">Mostrar Usuarios<br>Esta función es la encargada de mostrar todos los usuarios del sistema.</td><td>/Usuarios</td><td rowspan="2"></td><td rowspan="2"><pre>[
    {
        "email": "admin@ipc1.com",
        "gender": "M",
        "name": "Abner Cardona",
        "no": 0,
        "password": "admin@ipc1",
        "username": "admin"
    },
    {
        "email": "rod@gmail.com",
        "gender": "M",
        "name": "Rodrigo Hernández",
        "no": 1,
        "password": "Contraseña1234@",
        "username": "rodrialeh"
    }
]</pre></td></tr>
<tr><td>GET</td></tr>
</table>

<table>
<tr>
<td rowspan="2">Mostrar Usuarios Tabla<br>Esta función esta encargada de mostrar a los usuarios exceptuando al admin.</td>
<td>/Usuarios/Tabla</td>
<td rowspan="2"></td>
<td rowspan="2"><pre>[
    {
        "email": "rod@gmail.com",
        "gender": "M",
        "name": "Rodrigo Hernández",
        "no": 1,
        "password": "Contraseña1234@"
        "username": "rodrialeh"
    }
]</pre></td>
</tr>
<tr>
<td>GET</td>
</tr>
<tr>
<td rowspan="3">Login<br>Esta función se encarga de verificar si al entrar existe el usuario en la lista de usuarios y si entra retorna un mensaje de bienvenida de lo contrario tira mensaje de error.</td>
<td rowspan="2">/Login</td>
<td rowspan="3"><pre>{
  "username": "rodrialeh",
  "password": "Contraseña1234@"
}</pre></td>
<td><pre>{
    "Login": "true",
    "Mensaje": "Bienvenido Administrador",
    "Tipo": "Administrador"
}</pre></td>
</tr>
<tr>
<td><pre>{
    "Login": "true",
    "Mensaje": "Bienvenido rodrialeh",
    "Tipo": "Usuario",
    "user": "rodrialeh"
}</pre></td>
</tr>
<tr>
<td>POST</td>
<td><pre>{
    "Login": "false",
    "Mensaje": "Credenciales Incorrectas",
    "Tipo": "Error Effesinda"
}</pre></td>
</tr>
<tr>
<td rowspan="2">Actualizar Usuario<br>Esta función se encarga de actualizar los datos del usuario recibidos en un JSON y actualizando el objeto que va a modificar sus datos,si no encuentra el usuario muestra imagen de error.</td>
<td>/Usuarios/Modificar/<string:user></td>
<td rowspan="2"></td>
<td><pre>{
    "Mensaje": "Se actulizó correctamente el usuario, inicia sesión de nuevo.",
    "Mensajea": "Se actulizó correctamente el usuario",
    "estado": "true"
}</pre></td>
</tr>
<tr>
<td>PUT</td>
<td><pre>{
    "username":"rodrialeh",
    "email":"rodrialeh@gmail.com",
    "gender":"M",
    "name":"Rod",
    "password":"Contrarodri12@"
}</pre></td>
<td><pre>{
    "Mensajea": "No se encontró el usuario",
    "estado": "false"
}</pre></td>
</tr>
</table>

<table>
<tr><td rowspan="2">Mostrar Usuario Individual<br>Retorna los datos del usuario solicitado en formato JSON y por medio de la ruta el nombre de usuario el cual se le está solicitando los atributos de este objeto.</td><td>/Usuarios/<string:user></td><td rowspan="2"></td><td rowspan="2"><pre>{
    "email": "rod@gmail.com",
    "gender": "M",
    "name": "Rodrigo Hernández",
    "password": "Contraseña1234@",
    "user": "rodrialeh"
}</pre></td></tr>
<tr><td>GET</td></tr>
<tr><td rowspan="2">Carga Masiva de Usuarios<br>Se recibe en formato JSON el contenido del archivo cargado en el frontend y se castea de tal forma que se convierta en un archivo JSON y lea los datos de los usuarios y los registre al sistema.</td><td>/Usuarios/CargaMasiva</td><td rowspan="2"><pre>{
    "users": contenido del archivo
}</pre></td><td rowspan="2"><pre>{
    "Mensaje": "Se cargo correctamente"
}</pre></td></tr>
<tr><td>POST</td></tr>
<tr><td rowspan="2">Carga Masiva de Publicaciones<br>Se recibe en formato JSON el contenido del archivo cargado desde el frontend, se castea de tal forma que la API lo tome como JSON y obtenga los datos de las publicaciones y las registre al sistema.</td><td>/Publicaciones/CargaMasiva</td><td rowspan="2"><pre>{
    "publicaciones": contenido del archivo
}</pre></td><td rowspan="2"><pre>{
    "Mensaje": "Se cargo correctamente"
}</pre></td></tr>
<tr><td>POST</td></tr>
</table>

| | | | |
|---|---|---|---|
| **Eliminar Usuario** Este método manda busca al usuario dentro de la lista de Usuarios para borrarlo del sistema, también llama a otros métodos donde se borra el este objeto dentro de las otras listas que fue asignado el usuario. | /Usuarios/Eliminar/<string:user> | | { "Mensaje":"Se eliminó el usuario exitosamente" } |
| | DELETE | | { "Mensaje":"No se encontró el usuario" } |
| **Contador de Usuarios** Esta función se encarga de retornar la cantidad de usuario registrados en el sistema exceptuando al administrador | /Usuarios/Contador | | { "Cantidad":1 } |
| | GET | | |
| **Contador de Publicaciones** Esta función se encarga de retornar la cantidad de publicaciones registradas al sistema. | /Publicaciones/Contador | | { "Cantidad":1 } |
| | GET | | |
| **Tabla de Publicaciones** Esta función se encarga de mostrar todas las publicaciones registradas al sistema. | /Publicaciones/Tabla | | [ { "category": "Flowers", "date": "17/10/2021", "id": 1, "tipo": "Imagen", "url": "https://image.shutterstock.com/image-vector/seamless-vector-floral-pattern-background-000w-1867320830.jpg", "username": "casandbb" } ] |
| | GET | | |

| | | | |
|---|---|---|---|
| **Mostrar Publicación Individual**<br>Esta función se encarga de solicitar por medio de el ID de la publicación todos los atributos de esa publicación. | /Publicaciones/<string:idP><br>GET | | <pre>{<br>"category": "Soccer",<br>"date": "17/10/2021",<br>"id": 3,<br>"tipo": "Imagen",<br>"url": "https://fondosmil.com/fondo/7562.jpg"<br>}</pre> |
| **Editar una Publicación**<br>Esta función se encarga de pedir por medio de la ruta y su id modificar los datos de la publicación entrando un mensaje por medio de un JSON para editar la publicación. | /Publicaciones/Modificar/<int:idP><br>PUT | <pre>{<br>"url":"https://cdn.pixabay.com/photo/2015/04/23/22/00/tree-736885__340.jpg",<br>"date":"4/11/2021",<br>"category":"happy"<br>}</pre> | <pre>{<br>"Mensaje": "Se actulizó correctamente la publicación"<br>}</pre> |
| **Eliminar Publicación**<br>Esta función se encarga de eliminar la publicación por medio del id de la publicación escrita en la ruta y luego borrándola de todos los arreglos donde se encontraba la publicación | /Publicaciones/Eliminar/<int:idP><br>DELETE | | <pre>{<br>"Mensaje": "Se eliminó la publicacion exitosamente"<br>}</pre> |

| | | | |
|---|---|---|---|
| **Mostrar Publicaciones en el Inicio** Esta función se encarga de que la petición del frontend sea obtener todas las publicaciones registradas en el sistema en formato JSON array. | /Publicaciones/Inicio | |  |
| | GET | | |
| **Ranking de Publicaciones** Esta función se encarga de llamar a una función de ordenamiento de publicaciones por cantidad de likes y retorna el JSON con los datos de las publicaciones en orden de cantidad de "me gusta". | /Ranking | |  |
| | GET | | |
| **Crear Publicación** Esta función se encarga de recibir los datos de un JSON para agregarlo a la lista de Publicaciones. | /Publicaciones/Nuevo |  |  |
| | POST | |  |
| **Publicaciones por Usuario** Esta función se encarga de que se retorne las publicaciones por usuario registrado. | /Publicaciones/<string:user> | |  |
| | GET | | |

| | | | |
|---|---|---|---|
| **Reacciones** Esta función se encarga de retornar la cantidad de "me gusta" de cada publicación únicamente obteniendo el id y su cantidad. | /Reacciones<br>GET | | [ { "id": 2, "likes": 1 }, { "id": 3, "likes": 2 }, { "id": 4, "likes": 0 } ] |
| **TOP 5 DE PUBLICACIONES CON MÁS ME GUSTA** Esta función retorna las primeras 5 publicaciones con más cantidad de "me gusta" únicamente retornando su id y cantidad de "me gusta" y este orden se obtuvo llamando a una función que retornaba el arreglo ya ordenado. | /Publicaciones/Reacciones<br>GET | | [ { "id": 8, "likes": 10, "posicion": "1" }, { "id": 5, "likes": 7, "posicion": "2" }, { "id": 11, "likes": 6, "posicion": "3" }, { "id": 3, "likes": 4, "posicion": "4" }, { "id": 1, "likes": 3, "posicion": "5" } ] |
| **TOP 5 DE CANTIDAD DE PUBLICACIONES POR USUARIO** Esta función retorna únicamente los primeros 5 usuarios y su cantidad de publicaciones junto con su posición, se llama al método que hace el ordenamiento. | /Publicaciones/Usuarios<br>GET | | [ { "cantidad": 2, "posicion": "1", "username": "casandR8@" }, { "cantidad": 1, "posicion": "2", "username": "valero74*" }, { "cantidad": 1, "posicion": "3", "username": "richardB4@" }, { "cantidad": 1, "posicion": "4", "username": "0dalysP25@" }, { "cantidad": 1, "posicion": "5", "username": "rosaAg45@" } ] |

| | | | |
|---|---|---|---|
| **AÑADIR REACCION** Esta función hace que la cantidad de me gusta que tenga la publicación se le sumará 1 más . | /Reacción/Añadir |  |  |
| | POST | | |

# VI. Flujo de la Aplicación

## Comunicación entre Frontend y Backend


Donde el usuario al interactuar con el frontend manda las peticiones al backend(Python) y retorna las respuestas al frontend.

# VII. Créditos

Elaborado por el alumno Rodrigo Alejandro Hernández de León para el curso de IPC1, en el país de Guatemala con fecha desde el 8 de octubre de 2021 al 7 de noviembre de 2021.