Documentación

Servicios y Procesos

Programa: Proyecto Servidor Chat con Flask

Autor: Cristina Fuster García

# **1. Documentación de usuario final:**

## **1.1. Guía de usuario**

En este proyecto trabajamos con la librería de python Flask, creando un servidor con python, nos va a permitir poder recibir y enviar mensajes en tiempo real. Cada usuario se podrá conectar en la dirección que ofrece el servidor.

Requisitos Para el Sistema

* Tener Python instalados
* Instalas las dependencias de Flask y Flask-CORS
* Navegador web que sea compatible.

Accesos:

Asegurarse de que el servidor Flask este ejecutando, para ello tenemos que abrir la línea de comandos de nuestro ordenador y ejecutar el archivo de python.

Una vez corriendo el servidor nos dará una url donde la abriremos en la url.

Interfaz de Usuario

En la pantalla lo que nos vamos a encontrar es un campo donde tendremos que ingresar el nombre de usuario, tendremos una sección que aparece vacia y será donde se vayan mostrando los mensajes. Y abajo del todo tenemos un input que el usuario va a poder ingresar el mensaje.

Una vez envie el mensaje el programa le asignará un color según la numeración de su nombre.

## **1.2. FAQ (Preguntas frecuentes)**

¿Qué pasa si no arrancamos el servidor?

Si el servidor no está corriendo el chat aparecerá en el navegador y no nos dejara utilizarlo, pero tendremos un mensaje en la consola de error.

¿Que pasa si cierro la ventana?

Mientras el servidor siga corriendo, podrás volver a meterte y recuperar la información.

¿Como generamos los colores de cada usuario?

Los colores son generados mediante una operación cogiendo el valor numérico de ASCII.

# **2. Documentación técnica para desarrolladores**

## **2.1. Introducción al software**

El proyecto nos proporciona una plataforma de chat con servidor Flask, permitiendo a los usuarios enviar y recibir mensajes en tiempo real y organizar las interacciones con la interfaz de html.

Características Principales

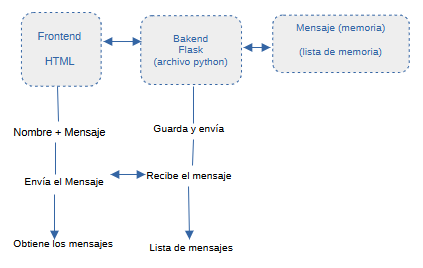
* Los mensajes de entrada y salida serán gestionados por el servidor Flask.
* Actualizara a tiempo real mediante la función del bucle que se va actualizando mediante el temporizador implementado.
* También tenemos dos peticiones fetch llamando a las funciones del servidor dame y toma que devuelve un json como resuelta.

Tecnologías Utilizadas

Backend: Flask, Flask-CORS

Fronted: HTML, JS

## **2.2. Arquitectura del sistema**

Diagrama de Flujo del Sistema 

El frontend captura la entrada del usuario y se comunica con el backend para enviar y recibir los mensajes.

El backend (Flask) es el que va a manejar las solicitudes y guardar los mensajes en una lista de memoria.

Los mensajes se envian y reciben a traves de las rutas /toma que la utilizaremos para enviar los mensajes y /dame para recibirlos al servidor, entre el frontend y el backend

El Flujo de Trabajo:

el usuaruario va a introducir el nombre de usuario y escribe el mensaje en los campos indicados, el mensaje se envia al servidor utilizando la solicitud POST a la ruta toma y dame.

El servidor recibe el mensaje y lo guarda en la lista de memoria y lo envia otra vez a traves de las rutas ndicadas.

En el fronted hace la solicitud GET a las rutas para obtener los mensajes.

## **2.3.Estructura del código**

* 006.chat.py implementa el servidor de Flask con rutas para recibir y enviar mensajes
* 016.chat.html es la interfaz de usuario para enviar y recibir los mensajes en tiempo real.

# **3. Guías de instalación y despliegue**

## **3.1. Requisitos previos**

Python

Bibliotecas:

* flask
* Flask-CORS

Un navegador

## **3.2. Instrucciones de instalación**

1.Descargar los archivos del proyecto

2. A través del la línea de comando tenemos que dirigirnos a la ruta donde esta el archivo e introducir python nombre\_proyecto.py y entonces veremos que se ejecuta el servidor.

3.Ir a la ruta que nos da el servidor y ya podremos utilizar el chat.

# **4. Mantenimiento y soporte**

## **4.1. Manejo de errores**

Que el servidor no arranque, eso podría significar que no hemos instalado bien las librerías o que el puerto está ocupado.

Que los mensajes no se vean o se repitan, tendríamos que ir a ver al codigo donde se encuentra la función del bucle.

## **4.2. Registro de cambios (changelog)**

version5: empecemos a trabajar con el servidor para que guardará los mensajes en un array.

version7: ya implementamos las rutas tanto de dame como de toma el propio servidor es el que se encarga de guardar la información y en el caso de dama los retorna en formato de json.

version16: la funcionalidad ya era aplicar un poco más de estilo

La suma de los colores ASCII

También vemos que hemos trabajado en el scroll para que los mensajes si llegan hasta arriba el que entre desde 0 les salgan los mensajes más recientes.

## **4.3. Soporte y contacto**

Verificar la consola del navegador si surgiera algún problema y asegurarnos de que el servidor está corriendo.