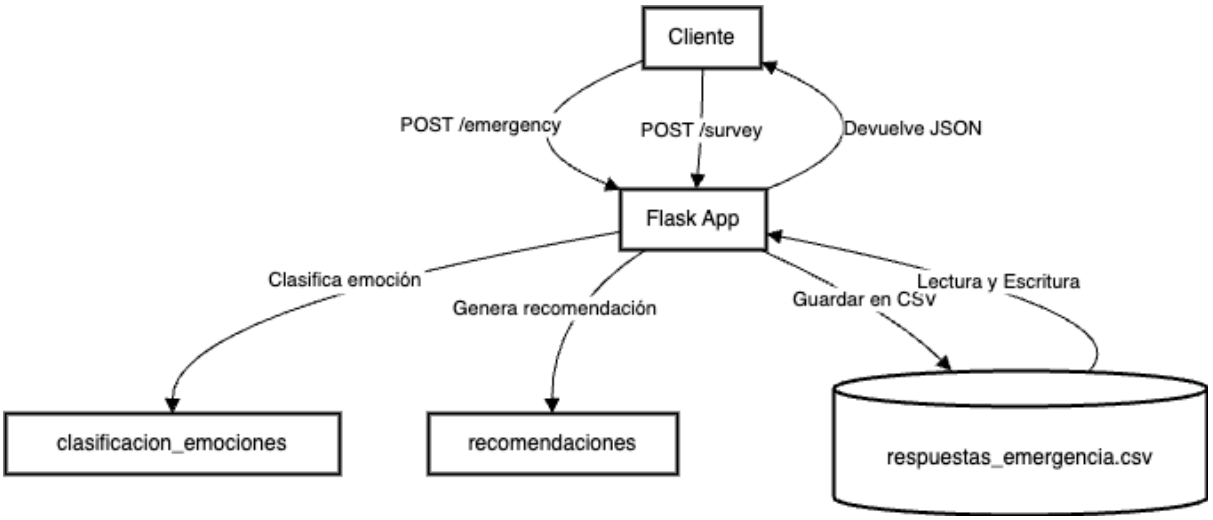
## CalmResponse

El proyecto se centra en el diseño y desarrollo de un sistema de inteligencia artificial (IA) para apoyar la salud mental de los estudiantes del Politécnico Colombiano Jaime Isaza Cadavid (PCJIC). Utilizando algoritmos avanzados de análisis de datos, el sistema recopilará y analizará información relevante, como el comportamiento en línea, los patrones de interacción social y el rendimiento académico de los estudiantes. A partir de estos datos, el sistema identificará indicadores de problemas de salud mental y ofrecerá recomendaciones personalizadas, como recursos de apoyo psicológico, grupos de discusión y actividades de bienestar. El proceso de diseño y desarrollo del sistema incluye la selección de algoritmos y la implementación de recomendaciones, seguido por la evaluación y validación en un entorno piloto con estudiantes reales. Finalmente, se discutirán las implicaciones prácticas y las oportunidades futuras del proyecto en el ámbito de la salud mental estudiantil, ofreciendo recomendaciones para su implementación en otras instituciones educativas.

## Arquitectura




## Desarrollo

### *Clasificación de Emociones*

```
texto_preprocesado = prueba.preprocess_text([texto])
```

La función clasificar_emocion es la encargada de predecir la emoción de un texto proporcionado por el usuario. Aquí se explica su funcionamiento paso a paso:

Se preprocesa el texto usando la función preprocess_text del módulo prueba. Esto generalmente implica limpieza del texto, tokenización, eliminación de stopwords, etc.

***Vectorización del Texto***

```
vectorizer = load("vectorizer_entrenado.pkl")
X = vectorizer.transform(texto_preprocesado)
```

El texto preprocesado se convierte en una representación numérica mediante un vectorizador (HashingVectorizer o TfidfVectorizer) previamente entrenado y guardado en un archivo `vectorizer_entrenado.pkl`.

***Carga de modelo y predicción:***

```
model = load("modelo_entrenado.pkl")
print("Modelo cargado correctamente.")
probabilidades = model.predict(X)
print("Probabilidades:", probabilidades)
```

El modelo de clasificación de emociones, guardado en modelo_entrenado.pkl, se carga y se usa para predecir la emoción del texto vectorizado. Este modelo puede ser un modelo de TensorFlow/Keras o un clasificador como Logistic Regression.

***Devolver la Emoción Predicha***

```
emocion_predicha = probabilidades
return emocion_predicha
```

Finalmente, se devuelven las probabilidades predichas, que representan la emoción clasificada del texto.

Función de pruebas

```
def prueba_clasificacion():
    textos_prueba = [
```

```
        "Estoy muy feliz hoy",
        "Me siento muy triste por lo sucedido",
        "Estoy lleno de amor por mi familia",
        "Me enfurece la injusticia que veo",
        "Siento mucho miedo cuando estoy solo por la noche"
    ]

    for texto in textos_prueba:
        emocion_predicha = clasificar_emocion(texto)
        print(f"Texto: '{texto}', Emoción predicha:
{emocion_predicha}")

prueba_clasificacion()
```

La función `prueba_clasificacion` se usa para probar la clasificación de emociones con textos de ejemplo

***Entrenamiento del Modelo***

```
def entrenar_modelo_emergencia(respuestas_emergencia):
    textos = [respuesta[0] for respuesta in respuestas_emergencia]
    emociones = [respuesta[1] for respuesta in
respuestas_emergencia]

    vectorizer = TfidfVectorizer()
    X_train = vectorizer.fit_transform(textos)

    clf = LogisticRegression()
    clf.fit(X_train, emociones)

    joblib.dump(vectorizer, "vectorizer.pkl")
    joblib.dump(clf, "modelo_emergencia.pkl")
```

La función `entrenar_modelo_emergencia` se utiliza para entrenar un modelo de clasificación de emociones desde cero y guardarlo junto con el vectorizador