José Sebastián Rodríguez Fonseca
joserodriguezfonseca10.03@gmail.com

Solución Prueba Técnica Backend Developer (Chat Integrations)

Documentación de la solución

Para esta solución se hizo uso de la plataforma de mensajería https://www.weavy.com/, la cual es de uso gratuito.

Se creo una solución en .NET Core, la cual contiene tres proyectos:

- **Hunty.Chat.Back**: Proyecto backend, permite el envío, recepción y lectura de mensajes.
- **Hunty.Chat.Front**: Proyecto frontend, permite exponer el chat que ofrece ***Weavy*** con una interfaz amigable.
- **Hunty.Chat.Transverse**: Proyecto helper, permite compartir utilidades y modelos a los dos proyectos anteriores.

Hunty.Chat.Back

Consta de tres capas:

- **Hunty.Chat.Back.API**, es la primera capa que se encarga de exponer los endpoint.
- **Hunty.Chat.Back.Aplication**, se encarga de manejar toda la lógica necesaria del backend y de implementar la integración con ***Weavy***.
- **Hunty.Chat.Back.DataBase**, se encarga de acceder al repositorio de base de datos

Expone principalmente 3 endpoints:

- **POST** `/api/v1/messages`:
  Este endpoint permite enviar y recibir mensajes.

  No maneja autenticación

  - Para enviar mensajes se utiliza la siguiente estructura JSON:

```
Content-Type: application/json
{
   “nameUser”: “sebastian”,
   “codeUser”: “sebastian”,
   “textMessage”: “Sending a test message”,
}
```

  - Para recibir mensajes se utiliza la siguiente estructura JSON:
  Para recibir los mensajes de la plataforma ***Weavy***, con anterioridad se debe de registrar este mismo endpoint dentro de la misma plataforma https://www.weavy.com/docs/learn/webhooks, cabe aclarar que este endpoint tiene que estar publico y expuesto a internet para que pueda ser consumido por el webhook.

José Sebastián Rodríguez Fonseca
joserodriguezfonseca10.03@gmail.com

- Adicionalmente, este endpoint se encarga de crear, autenticar y agregar los usuarios al chat, adicionalmente valida si el mensaje ya existe o no para no almacenar mensajes repetidos.

- **GET** `/api/v1/messages`:
  Este endpoint permite leer únicamente los mensajes que se recibieron.

- **GET** `/api/v1/user/getAccessToken`:
  Este endpoint retorna el AccessToken de un usuario definido únicamente para el proyecto *Hunty.Chat.Front*.

Hunty.Chat.Front

Este proyecto facilita la prueba, este se encarga de utilizar un CDN que ofrece ***Weavy*** https://cdn.jsdelivr.net/npm/@weavy/dropin-js@17.0.3/dist/weavy.js, el cual hace el render del chat que se esta utilizando en la prueba, donde muestra todos los mensajes que se han enviado a este chat.

Cuando se envíen mensajes por medio del chat este proyecto, el CDN de ***Weavy*** es el único que se encarga de enviar los mensajes.

Los mensajes que se escriben por este proyecto son los que se disparan por medio del webhook de ***Weavy*** y que su vez llegan al endpoint anteriormente mencionado.

Configurar Webhook:

```
POST    https://1ca4a5e812a34d64a17560d00f8b6181.weavy.io/api/webhooks

Authorization: Bearer wys_ZDw23OqeDXFUjkaXxjbQ8L3Np9oVIv2SJibk

{
  "payload_url": "{URLDomain}/api/v1/messages ",
  "triggers": [
    "messages"
  ],
  "enabled": true
}
```

Weavy

Para hacer uso de la plataforma Weavy, previamente se tuvo que crear una cuanta en esta plataforma y adicionalmente, se creo un ambiente dentro de la misma, esto para poder generar los datos de las entidades del chat, por ejemplo, Conversations, Messages, Webhook, Users.

José Sebastián Rodríguez Fonseca
joserodriguezfonseca10.03@gmail.com

Adicionalmente ***Weavy*** brinda por cada ambiente las siguientes credenciales para poder hacer uso del API:

URL: ***https://1ca4a5e812a34d64a17560d00f8b6181.weavy.io***
API Key: wys_ZDw23OqeDXFUjkaXxjbQ8L3Np9oVIv2SJibk

Los anteriores datos son esenciales para la integración con la plataforma.

Integración Hunty.Chat.Back y Weavy

Dentro del proyecto Hunti.Chack.Back.API en el archivo de configuraciones *appsettings.json* se encuentran los endpoints y el API Key que se utilizaron para integrar estos dos sistemas en el nodo `WeavyChatPlatformAPI`.

Base de datos

Para esta solución se implementó una base de datos relacional desarrollada en SQL Server 2022 y adicionalmente, se utilizó la tecnología de *EntityFrameworkCore* para permitir la interacción entre el proyecto de .NET Core y la base de datos.

Pruebas

Pruebas Unitarias

Se realizaron pruebas de funcionalidad y de consistencia de datos dentro de cada capa del proyecto, para asegurar que cada capa se encargará de la responsabilidad que se le había establecido y que así mismo fuera capas de manejar cualquier cambio inesperado de datos o de funcionalidad de otra capa.

Pruebas de Integración

Durante el desarrollo, dentro de la capa de aplicación donde se conecta con sistemas externos, se implementaron pruebas de caja negra, donde se iba comprobando si el sistema estaba recibiendo los request de manera exitosa, así mismo, por medio de Serilog se iba teniendo control de la recepción y envio de los datos a ***Weavy***.