RESUMEN – TRABAJO DE GRADO

AUTORES	Jesus Joseph Suescun Batista
FACULTAD	Ingenierías
PLAN DE ESTUDIOS	Ingeniería de Sistemas
DIRECTOR	Henry Eliseo Navarro Chinchilla
TÍTULO DE LA TESIS	Desarrollo de una API utilizando la inteligencia artificial Dapta que proporcione información en tiempo real sobre las operaciones de Bodytech, para optimizar la eficiencia operativa de la empresa.
TITULO EN INGLES	Development of an API using Dapta artificial intelligence that provides real-time information on Bodytech operations, to optimize the company's operational efficiency.

RESUMEN
(70 palabras)

El proyecto implementó una API basada en inteligencia artificial para optimizar las operaciones de Bodytech. Se analizaron procesos, definieron requerimientos y diseñaron diagramas de flujo. Con Phalcon, React, Jenkins, Redis y MySQL, se automatizaron tareas, mejorando tiempos de respuesta y seguridad. Se ejecutaron pruebas de carga y se capacitó al equipo, fortaleciendo la infraestructura tecnológica. La iniciativa preparó a la empresa para enfrentar desafíos operativos y tecnológicos, consolidando su competitividad.

RESUMEN EN INGLES

The project implemented an API based on artificial intelligence to optimize Bodytech operations. Processes were analyzed, defined requirements and designed flow diagrams. With Phalcon, React, Jenkins, Redis and MySQL, tasks were automated, improving response and security times. Load tests were executed and the team was trained, strengthening the technological infrastructure. The initiative prepared the company to face operational and technological challenges, consolidating its competitiveness.

PALABRAS CLAVES	API, Inteligencia Artificial, Bodytech, Optimización operativa, Phalcon, React, MySQL
PALABRAS CLAVES EN INGLES	API, Artificial Intelligence, Bodytech, Operational Optimization, Phalcon, React, MySQL

CARACTERÍSTICAS

PÁGINAS: 172	PLANOS: 0	ILUSTRACIONES: 141	CD-ROM:

Desarrollo de una API utilizando la inteligencia artificial Dapta que proporcione información en tiempo real sobre las operaciones de Bodytech, para optimizar la eficiencia operativa de la empresa.

Jesús Joseph Suescun Batista

Facultad de Ingenierías, Universidad Francisco de Paula Santander Ocaña

Ingeniería de Sistemas

Mcs. Henry Eliseo Navarro Chinchilla

13 de marzo de 2025

Índice

1. Desarrollo de una API utilizando la Inteligencia Artificial Dapta que proporcione información en tiempo real sobre las operaciones de Bodytech, para optimizar la eficiencia operativa de la empresa 12

1.1. Descripción de la Universidad Francisco de Paula Santander Ocaña 12

1.1.1. Historia de Bodytech 12

1.1.2. Misión 13

1.1.3. Visión 14

1.1.4. Objetivos de BodyTech 14

1.1.5. Ejes de valor 15

1.1.6. Descripción de la estructura organizacional de BodyTech 16

1.1.7. Descripción de la dependencia y/o proyecto al que fue asignado 20

1.2. Diagnóstico inicial de la dependencia 21

1.2.1. Planteamiento del problema 23

1.3. Objetivos de la pasantía 25

1.3.1. Objetivo general 25

1.3.2. Objetivos específicos 25

1.4. Descripción de las actividades a desarrollar 26

2. Enfoques Referenciales 28

2.1. Enfoque Conceptual 28

2.1.1. Phalcon 28

2.1.2. React 29

2.1.3. Jenkins 29

2.1.4. Redis 30

2.1.5. MySQL 31

2.2. Enfoque Legal 32

3. Informe de cumplimiento del trabajo de pasantía 34

3.1. Presentación de resultados 34

3.1.1. Objetivo 1. Actividad 1: Recolectar y analizar la documentación existente sobre los sistemas de gestión de datos de Bodytech, identificando las bases de datos y los tipos de datos manejados. 35

3.1.2. Objetivo 1. Actividad 2: Realizar sesiones de trabajo con el personal de TI para comprender cómo se generan, almacenan y utilizan los datos en las operaciones diarias. ... 38

3.1.3. Objetivo 1. Actividad 3: Crear diagramas de flujo que representen el recorrido de la información desde su origen hasta su uso final en la toma de decisiones. 44

3.1.4. Objetivo 2. Actividad 1: Definir los requisitos funcionales y no funcionales de la API. 45

3.1.5. Objetivo 2. Actividad 2: Investigación y Selección de Tecnologías y Frameworks para el Desarrollo de la API 56

3.1.6. Objetivo 2. Actividad 3: Crear un prototipo inicial de la API que incluya las funcionalidades básicas y realizar pruebas de concepto para asegurar su funcionalidad básica. 65

3.1.7. Objetivo 2. Actividad 4: Desarrollar e implementar medidas de seguridad para proteger la API y garantizar la privacidad de la información manejada. 102

3.1.8. Objetivo 3. Actividad 1: Diseñar escenarios de prueba que simulan condiciones reales de operación. 148

3.1.9. Objetivo 3. Actividad 2: Ejecutar pruebas de carga y estrés para evaluar la capacidad de la API de manejar altos volúmenes de datos y usuarios simultáneos. 151

3.1.10. Objetivo 3. Actividad 3: Analizar los resultados de las pruebas para identificar áreas de mejora en el rendimiento de la API, incluyendo tiempos de respuesta y eficiencia de procesamiento. 152

3.1.11. Objetivo 3. Actividad 4: Implementar optimizaciones basadas en los resultados de las pruebas para mejorar el rendimiento y la escalabilidad de la API, asegurando que cumpla con los requisitos definidos. 154

3.1.12. Objetivo 4. Actividad 1: Crear materiales de capacitación detallados sobre el uso de la API. 158

3.1.13. Objetivo 4. Actividad 2: Organizar talleres de capacitación, proporcionando instrucción práctica y escenarios de uso realistas para familiarizarse con la API. 160

3.1.14. Objetivo 4. Actividad 3: Documentar y entregar el trabajo realizado durante el proceso de pasantía. 162

4. Diagnóstico Final 163

5. Conclusiones 165

6. Recomendaciones 166

7. Referencias 167

Listado de Tablas

Tabla 1. Matriz DOFA de la dependencia asignada. 21
Tabla 2. Actividades a desarrollar en la pasantía..... 26
Tabla 3. R1. Crear y configurar el flujo en DAPTA que entregue la información personal de afiliados en Bodytech..... 45
Tabla 4. R2. Crear y configurar el flujo en DAPTA que entregue la información del contrato actual de afiliados en Bodytech. 46
Tabla 5. R3. Crear y configurar el flujo en DAPTA que entregue la información de los planes y servicios incluidos en el contrato actual de afiliados en Bodytech..... 47
Tabla 6. R4. Crear y configurar el agente en DAPTA que interactúe con los afiliados de Bodytech. 48
Tabla 7. R5. Crear y configurar el agente en DAPTA que interactúe con los asesores comerciales de Bodytech. 49
Tabla 8. R6. Crear y configurar el flujo en DAPTA para la conexión del agente con la API de comunicación. 50
Tabla 9. R7. Gestionar la conexión de la API de comunicación con DAPTA. 50
Tabla 10. R8. Diseñar la interfaz del chatbot..... 51
Tabla 11. R10.Gestionar la comunicación entre el cliente y la API de DAPTA. 52
Tabla 12. R12. Crear la restricción del chatbot por rol y por marca..... 52
Tabla 13. R13. Crear la tabla del log en la base de datos. 53
Tabla 14. R14. Gestionar el registro de las interacciones con el chatbot mediante API REST. .. 54
Tabla 15. R15. Gestionar la comunicación entre el cliente y la API REST de registro de logs... 54

Listado de Figuras

Figura 1. Muestra de sedes en la Ciudad de Bogotá. 13
Figura 2. Ejes de valor de Bodytech. 16
Figura 3. Organigrama Gestión Operativa Bodytech 17
Figura 4. Organigrama completo Bodytech 19
Figura 5. Reunión con el gerente de desarrollo 41
Figura 6. Flujo de requerimientos 44
Figura 7. Flujo de Bugs 45
Figura 8. Automatización de Contenido con IA 58
Figura 9. Automatización del Flujo de Trabajo con IA 58
Figura 10. Agente con IA 59
Figura 11. Evidencia 1. Capacitación asesores de DAPTA 66
Figura 12. Evidencia 2. Capacitación asesores de DAPTA 66
Figura 13. Evidencia 3. Capacitación asesores de DAPTA 67
Figura 14. Evidencia 4. Capacitación asesores de DAPTA 67
Figura 15. Reunión con desarrollador para entender la estructura del proyecto 68
Figura 16. Inicio de sesión en DAPTA 69
Figura 17. Nombrar flujo 69
Figura 18. Muestra de Nodos 70
Figura 19. Muestra de un trigger 71
Figura 20. Configuración de Trigger 72
Figura 21. Asignación de nombre al nodo 73
Figura 22. Estructura del Nodo Api 73
Figura 23. Evidencia uso del Response Key 76
Figura 24. Configuración del apartado de los Headers 77
Figura 25. Evidencia de certificados deshabilitados 78
Figura 26. Nodo Response 78
Figura 27. Ejecución Del Flujo 79
Figura 28. Resultado del Nodo Response 79
Figura 29. Acceso al JSON de la ejecución 80

Figura 30. Creación De Agente 81
Figura 31. Resultado Creación De Agente 81
Figura 32. Configuración del agente con carga de URL 82
Figura 33. Carga URL de Bodytech 83
Figura 34. Instrucciones de plataforma 84
Figura 35. Muestra del agente inicial creado 84
Figura 36. Apartado dentro del agente: Instructions 85
Figura 37. Apartado dentro del agente: Brain 86
Figura 38. Apartado dentro del agente: Integrations 87
Figura 39. Instrucciones o parámetros 87
Figura 40. Prueba flujo get-member-info. 88
Figura 41. Consulta de función del flujo get-member-info, ... 88
Figura 42. Prueba de funcionamiento del agente 89
Figura 43. Sección de Dapta: Publish 90
Figura 44. Script en el código del Proyecto 91
Figura 45. Evidencia 1 - Resultado en la página web (ambiente de staging) 91
Figura 46. Evidencia 2 - Resultado en la página web (ambiente de staging) 92
Figura 47. Segundo flujo - Creación de APIS 93
Figura 48. Configuración nodo Trigger 94
Figura 49. Configuración nodo API (authentication) 94
Figura 50. Configuración nodo API (get-member-bodytech) 96
Figura 51. Configuración de los headers 97
Figura 52. Configuración nodo API (get_agreement_member) 97
Figura 53. Configuración nodo API (Get_Agreement_Lines_Member) 98
Figura 54. Configuración nodo Response 99
Figura 55. Ejecución segundo flujo 100
Figura 56. Resultado segundo flujo 101
Figura 57. Evidencia tiempo mínimo de respuesta 101
Figura 58. Creación de flujos separados 103
Figura 59. Flujo Get_User (Primer Requerimiento) 103
Figura 60. Conformación de nodos para el flujo del primer requerimiento 104

Figura 61. Definición de parámetro Trigger para el flujo del primer requerimiento 105
Figura 62. Definición de parámetros globales 105
Figura 63. Nodo Api (Authentication_Mybt 106
Figura 64. Respuesta del nodo Api (Authentication_Mybt 107
Figura 65. Nodo Api(Get_Member_Bodytech): 107
Figura 66. Configuración de respuestas 108
Figura 67. Configuración de headers 109
Figura 68. Nodo Organize_Response 109
Figura 69. Evidencia 2 - Nodo Organize_Response 110
Figura 70. Nodo Response 111
Figura 71. Ejecución en la plataforma 111
Figura 72. Respuesta del nodo response 112
Figura 73. Tiempo De Ejecución 112
Figura 74. Flujo Get_Agreement_User (Segundo Requerimiento) 113
Figura 75. Nodos para el segundo requerimiento 113
Figura 76. Configuración de respuesta para el segundo requerimiento 114
Figura 77. Configuración de headers de igual manera al nodo get_member_bodytech 115
Figura 78. Nodo Organize_response 115
Figura 79. Nodo Response para el segundo requerimiento 116
Figura 80. Ejecución En La Plataforma 116
Figura 81. Respuesta de ejecución en la plataforma 117
Figura 82. Tiempo de ejecución 117
Figura 83. Flujo Get_User_Plans (Tercer Requerimiento) 118
Figura 84. Conformación de nodos (Tercer Requerimiento) 118
Figura 85. Nodo Api (Get_Agreement_Member) 119
Figura 86. Nodo Api (Get_Agreement_Lines_Member) 120
Figura 87. Evidencia 1-Nodo Organize_Response 120
Figura 88. Evidencia 2-Nodo Organize_Response 121
Figura 89. Nodo Response (Tercer requerimiento) 122
Figura 90.Ejecución en la plataforma 122
Figura 91. Respuesta flujo tercer requerimiento 123

Figura 92. Tiempo de ejecución 123
Figura 93. Creación Del Agente (Requerimiento 4) 125
Figura 94. Instrucciones al agente 125
Figura 95. Evidencia 1.Test agent de Dapta 126
Figura 96. Evidencia 2. Test agent de Dapta 126
Figura 97. Evidencia 1. Prueba de script 127
Figura 98. Evidencia 2. Prueba de script 127
Figura 99.Creación Del Agente Para El Asesor Comercial (Requerimiento 5) 129
Figura 100. Documentos añadidos al agente 129
Figura 101. Prueba local en la plataforma 130
Figura 102. Prueba del script en el software de Bodytech 131
Figura 103. Flujo Dapta (Requerimiento 6) 133
Figura 104. Nodo Trigger para el Requerimiento 6 133
Figura 105. Nodo format_agent_call 134
Figura 106. Nodo Dapta Agents 135
Figura 107. Nodo Response 136
Figura 108. Creación De La Api De Comunicación En Phalcon (Requerimiento 7) 137
Figura 109. Configuración de la variable interna del proyecto 137
Figura 110. Creación de la ruta del endpoint 138
Figura 111. Implementación del servicio de la comunicacion del chatbot 138
Figura 112. Evidencia 1. Creación del controlador 139
Figura 113. Evidencia 2. Creación del controlador 140
Figura 114. Diseño Interfaz Del Chatbot (Requerimiento 8) 140
Figura 115. Importación del API de comunicación con las APIS 141
Figura 116. Configuración de las animaciones del icono 142
Figura 117. Creación de las interfaces de las API para la comunicación 142
Figura 118. Creación de la lógica para la comunicación del front con el chat 143
Figura 119. Creación de la restricción del chatbot por rol y por marca (Requerimiento 10) 143
Figura 120. Desarrollo backend para el registro del historial 145
Figura 121. Creación del controlador de la API 146
Figura 122. Importación del API de comunicación con las APIS 146

Figura 123. Creación de las interfaces de las APIS 147
Figura 124. Implementación de la lógica..... 147
Figura 125. Evidencia 1. Pruebas de Ejecución de Flujos en la Plataforma DAPTA 149
Figura 126. Evidencia 2. Pruebas de Ejecución de Flujos en la Plataforma DAPTA 149
Figura 127. Evidencia 3. Pruebas de Ejecución de Flujos en la Plataforma DAPTA 150
Figura 128. Pruebas en Postman..... 150
Figura 129. Script..... 151
Figura 130. Tiempo de ejecución..... 152
Figura 131. Código a refactorizar 153
Figura 132. Código refactorizado 154
Figura 133. Evidencia mejor tiempo de ejecución..... 155
Figura 134. Evidencia 1. Prueba satisfactoria de la API..... 155
Figura 135. Evidencia 2. Prueba satisfactoria de la API..... 156
Figura 136. Evidencia 3. Prueba satisfactoria de la API..... 157
Figura 137. Registro de interacción del asesor 157
Figura 138. Evidencia 1.Capacitación sobre el uso de la API 159
Figura 139. Evidencia 2. Capacitación sobre el uso de la API 159
Figura 140. Evidencia 1. Talleres de capacitación 160
Figura 141. Evidencia 2. Talleres de capacitación 161

1. Desarrollo de una API utilizando la Inteligencia Artificial Dapta que proporcione información en tiempo real sobre las operaciones de Bodytech, para optimizar la eficiencia operativa de la empresa

1.1. Descripción de la Universidad Francisco de Paula Santander Ocaña

1.1.1. Historia de Bodytech

Bodytech nace como una idea de grado de dos estudiantes del MBA de la Universidad de los Andes en Bogotá, Colombia quienes se conocieron en el año 1996. Nicolás Loaiza y Gigliola Aycardi habían revisado la posibilidad de crear un lugar en donde a través del ejercicio se mejorará la calidad de vida de las personas y que además tuviera proximidad al lugar donde vivían o trabajaban. Después de un estudio de mercadeo estructuraron, dentro de su proyecto de grado, la idea de cambiar el concepto de gimnasios, a Centros Médico Deportivos con un claro enfoque en salud (BodyTech c. , 2022).

El objetivo de estos Centros Médicos Deportivos era la prescripción del ejercicio de acuerdo con la condición física de cada usuario, así como sus motivaciones, prever lesiones y tratar síntomas de diversas enfermedades. Con esta visión como eje fundamental, los jóvenes lograron fundar BODYTECH el 14 de julio de 1.997, haciendo la inauguración de su primera sede el 9 febrero de 1.998 en la carrera séptima con calle 63, en el barrio Chapinero de Bogotá (BodyTech c. , 2022).

Bodytech cuenta con más de 21 años de existencia, con presencia de más de 170 sedes en Colombia, Perú y Chile, 300.000 usuarios y aproximadamente más de 4.000 profesionales.

Figura 1.

Muestra de sedes en la Ciudad de Bogotá.



Fuente. Tomada de la página oficial de Bodytech (BodyTech Sedes, 2022).

1.1.2. Misión

Bodytech tiene como misión principal mejorar la calidad de vida de su comunidad a través de programas de entrenamiento efectivos en dónde prime la salud (Camargo, Amaya, & Arias, 2023). Así mismo, su propósito es transformar la vida de sus afiliados a través del entrenamiento físico, haciendo que se sientan más fuertes y saludables, disminuyendo sus riesgos para tener un impacto positivo en su salud y autoestima (Bodytech Introduction, 2024).

Se pude decir que más que un Gimnasio, es un centro integral de bienestar que entiende la salud de manera holística. Sus servicios de nutrición y fisioterapia son testimonio de su compromiso con el bienestar general de sus miembros, ofreciendo soluciones personalizadas para alcanzar un equilibrio saludable entre mente y cuerpo (Términos Y Condiciones De La Prestación Del Servicio | Bodytech, 2022).

1.1.3. Visión

Como visión, Bodytech busca ser la red de Centros Médicos deportivos líder en Latinoamérica (Camargo, Amaya, & Arias, 2023). Su sueño es ser reconocidos por el cliente y la comunidad por ser el mejor y más profesional club de entrenamiento Fitness y Wellness (Bodytech Introduction, 2024).

Buscando el impacto comunitario y la expansión, la visión de Bodytech trasciende el ámbito empresarial, impactando positivamente en las comunidades donde opera. Iniciativas de responsabilidad social, eventos de fitness al aire libre y programas de salud para poblaciones vulnerables son solo algunas de las maneras en que la empresa contribuye al bienestar de la sociedad (Términos Y Condiciones De La Prestación Del Servicio | Bodytech, 2022).

1.1.4. Objetivos de BodyTech

Bodytech promueve el desarrollo deportivo en diversas sedes, brindando apoyo médico a comunidades vulnerables y promoviendo una vida mejor (Camargo, Amaya, & Arias, 2023)

Para conseguir su propósito expuesto en su misión, hacen lo que denominan ADN Tech, que se representa en tres partes fundamentales, *Hacen lo correcto:* donde son coherentes con lo que piensan, dicen y hacen; Tienen actitud de servicio: se enfocan en conectar, deleitar, sorprender y acompañar a sus clientes en el logro de sus sueños y *son apasionados por bodytech:* donde trabajan en equipo, con confianza, compromiso y alegría (Bodytech Introduction, 2024).

También se destacan por su innovación y servicios, siendo uno de los pioneros en la integración de la tecnología en el sector del fitness, con su aplicación móvil Bodytech Corp, que proporciona planes de entrenamiento personalizados, seguimiento del progreso, y una comunidad de apoyo (Revista Semana, 2016). Así mismo, ofrecen una amplia gama de clases grupales diseñadas para satisfacer diferentes intereses y niveles de habilidad de sus afiliados (Socorras, 2024).

1.1.5. Ejes de valor

Según (Acosta & Casas, 2014), los ejes de valor que guían las acciones de Bodytech son:

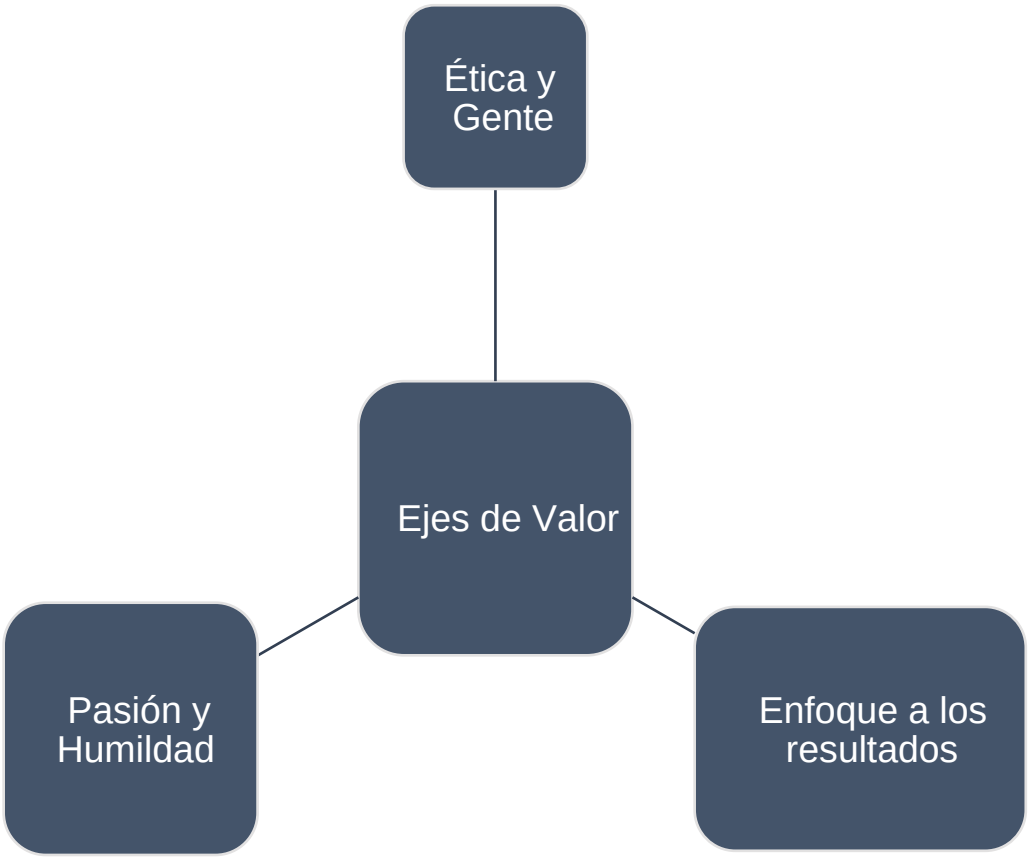
Ética y gente: Honestidad, transparencia, equidad, respeto, lealtad, solidaridad, reconocimiento y crecimiento personal.

Enfoque a los resultados: Planear, ejecutar, controlar, alcanzar y sobrepasar las metas.

Pasión y Humildad: Amor por lo que hacen y disposición para aprender y servir.

Figura 2.

Ejes de valor de Bodytech.



Fuente. Basado de (Acosta & Casas, 2014)

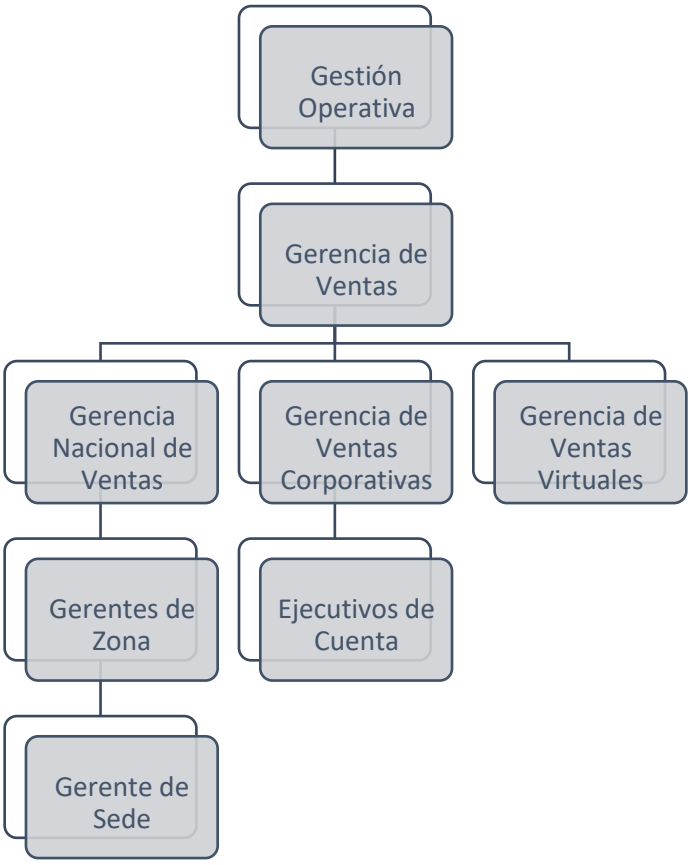
1.1.6. Descripción de la estructura organizacional de BodyTech

Con el fin de cumplir con los objetivos empresariales, la estructura organizacional de Bodytech se divide en 2 grandes grupos: una parte administrativa y otra operativa. La parte administrativa es responsable de la prestación de servicios y la operativa se encarga de lograr los objetivos de ventas. Así mismo, como se puede ver en la figura 3, el organigrama de la compañía se estructura en torno a ciertos cargos clave de la gestión operativa: Gerente de ventas, Gerentes

de Zona, Gerentes de Sede, Asesor Integral de Servicio y Entrenador de Planta. Estos roles son cruciales para la operación de más de 1200 empleados en el caso de Colombia (Camargo, Amaya, & Arias, 2023) (Socorras, 2024)

Figura 3.

Organigrama Gestión Operativa Bodytech

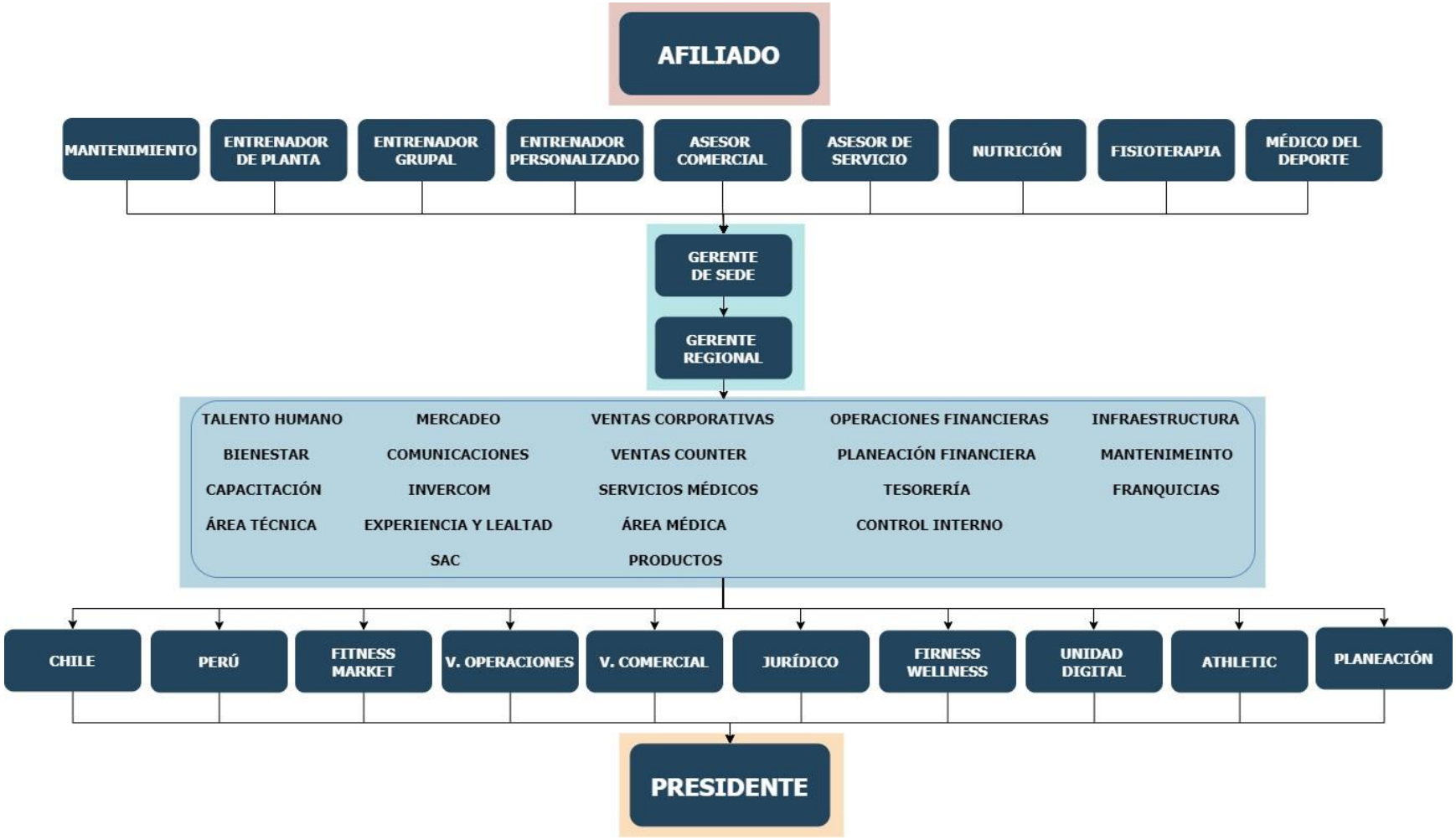


*Fuente***.** Tomado de la información suministrada por el Ingeniero a cargo del área de TI (Socorras, 2024) (Camargo, Amaya, & Arias, 2023)

Así mismo, Bodytech enfatiza en que la estructura clásica de una organización no se ajusta a sus necesidades empresariales. Por ello, como apreciaremos en la figura 4, los líderes y gerentes no están en la parte superior del organigrama. Para ellos las personas responsables por la empresa están en la base, en roles de soporte a la gestión de quienes están en el siguiente nivel de responsabilidad. Así también, para llevar el mejor servicio a sus afiliados y clientes desarrollaron un método de trabajo llamado Gerencia por compromisos, que les ayuda a cumplir con las exigencias actuales del mercado sin dejar a un lado sus objetivos y razón empresariales.

Figura 4.

Organigrama completo Bodytech



Fuente. Tomado de contenido introductorio al proceso de pasantía en Bodytech.

1.1.7. Descripción de la dependencia y/o proyecto al que fue asignado

El área donde fui asignado para el proceso de pasantía, es el área de IT o Dirección de Tecnología Informática. En la empresa esta área está encargada de comprender las prioridades del negocio, apoyar y entender las ideas innovadoras a través de la gestión de capacidades, gestión de la demanda de los usuarios y/o clientes y gestión de expectativas para el desarrollo y entrega de servicios (Bodytech Introduction, 2024). Lo antes mencionado, se hace con recursos calificados y con el apoyo de relaciones con proveedores que les aseguren innovación continua y uso de tecnologías innovadoras en los procesos.

El objetivo principal del área de IT, es dirigir la estrategia de Tecnología de Información de Bodytech, alineados siempre a los objetivos y proyectos de innovación. Según el proceso introductorio llevado a cabo (Bodytech Introduction, 2024), dentro de sus objetivos se encuentran:

- Definir la integración y crecimiento de plataforma de tecnologías de información apropiada para Bodytech.
- Liderar proyectos estableciendo el alcance, tiempo y costo, cumpliendo los requerimientos establecidos.
- Asegurar la operación de las aplicaciones de Bodytech, incluyendo la gestión del día a día.
- Asegurar la entrega del servicio de ti continuo, seguro y de alta calidad.
- Garantizar el uso de la infraestructura tecnológica, y de las aplicaciones.

1.2. Diagnóstico inicial de la dependencia

Tabla 1.

Matriz DOFA de la dependencia asignada.

	Fortalezas (F)	**Debilidades (D)**
MATRIZ DOFA	Bodytech es una de las principales cadenas de gimnasios en América Latina, con un alto reconocimiento y prestigio en el sector del fitness. Presencia en múltiples localidades, lo que permite captar un amplio mercado de clientes. Ofrecen una amplia gama de servicios, desde entrenamientos personalizados hasta clases grupales, lo que atrae a diversos segmentos de clientes. Entrenadores y personal de atención al cliente bien entrenados y experimentados en el manejo de clientes y operaciones del gimnasio.	Muchas operaciones de servicio al cliente aún se manejan manualmente, lo que incrementa los tiempos de respuesta. Falta de integración efectiva entre los diferentes sistemas de gestión interna, lo que dificulta el acceso rápido a la información. Limitaciones en la capacidad del personal para manejar altos volúmenes de consultas durante horas pico. Muchas operaciones dependen de instalaciones físicas, lo que puede limitar la flexibilidad en la prestación de servicios.

Oportunidades (O)	**FO**	**DO**
Implementación de tecnologías avanzadas, como la inteligencia artificial y el aprendizaje automático, para mejorar la eficiencia operativa y el servicio al cliente. El creciente interés por la salud y el bienestar presenta	Aprovechar los avances tecnológicos para desarrollar soluciones innovadoras que mejoren la eficiencia operativa y el servicio al cliente. Utilizar la digitalización para ofrecer servicios en línea, expandiendo el	Implementar nuevas tecnologías para integrar los sistemas de gestión y mejorar el flujo de información. Capacitar al personal en el uso de herramientas digitales y automatizar

Oportunidades para captar nuevos clientes y expandir la oferta de servicios. Desarrollo de plataformas en línea para entrenamientos virtuales y consultas remotas, mejorando el acceso y la comodidad para los clientes.	alcance y mejorando la accesibilidad para los clientes.	procesos para reducir tiempos de respuesta.
Amenazas (A)	**FA**	**DA**
Aumento de la competencia en el sector del fitness, con nuevos gimnasios y plataformas de entrenamiento en línea. Variaciones en las preferencias de los clientes hacia servicios más personalizados y digitalizados. Factores económicos que pueden afectar el poder adquisitivo de los clientes y su disposición a gastar en membresías de gimnasio.	Usar el reconocimiento de marca y la amplia red de sucursales para adaptarse rápidamente a los cambios en las preferencias de los consumidores y enfrentar la competencia.	Identificar áreas críticas para mejorar procesos internos y reducir la dependencia de operaciones manuales que incrementan los tiempos de respuesta. Desarrollar estrategias de gestión de riesgos para mitigar el impacto de crisis económicas o sanitarias en las operaciones.

Fuente. Elaboración propia. Basado en información suministrada por la empresa.

1.2.1. Planteamiento del problema

Continuamente, en el sector fitness y de bienestar, la calidad del servicio al cliente es fundamental para mantener la satisfacción y lealtad de los clientes (García, Fernández, & Bernal, 2014). La empresa Bodytech, es una de las principales cadenas de gimnasios en América Latina (Rodriguez P, 2023), ofreciendo una alta gama de servicios enfocados en el bienestar general de sus afiliados en cuanto a la salud. Destacándose por la integración de la tecnología en todas sus instalaciones, como lo es su aplicativo móvil Bodytech Corp.

Sin embargo, la necesidad de desarrollar productos que puedan ofrecer servicios novedosos, que permitan el aumento de ingresos, una mayor competitividad, que pueda garantizar la disponibilidad y que proporcione información en tiempo real de sus operaciones, requiere el desarrollo de nuevos servicios, utilizando herramientas en tendencia.

Es por eso que la empresa se enfrenta al desafío de mantener grandes estándares de servicio al cliente mientras maneja una gran cantidad de operaciones diarias en sus múltiples sedes (Socorras, 2024). En este contexto, la capacidad de brindar respuestas rápidas y precisas a las consultas y necesidades de los clientes es crucial para la competitividad y la eficiencia operativa de la empresa. Todo esto con el fin de impactar positivamente en la percepción de sus afiliados y la retención de clientes.

Es así como Bodytech ve la necesidad de desarrollar un API utilizando herramientas de Inteligencia Artificial como Dapta, que permite integrar múltiples fuentes de datos, crear instrumentos basados en la lógica empresarial y desarrollar Apis usando constructores sin código (Henao, 2023); que pueda proporcionar la información en tiempo real sobre todas sus operaciones y así disminuir el tiempo de respuesta del servicio al cliente y mejorar su eficiencia operativa.

1.3. Objetivos de la pasantía

1.3.1. Objetivo general

Desarrollar una API utilizando la inteligencia artificial Dapta que proporcione información en tiempo real sobre las operaciones de Bodytech, para optimizar la eficiencia operativa de la empresa.

1.3.2. Objetivos específicos

Analizar la información interna de Bodytech, con el fin de hacer un reconocimiento de los flujos de datos que se manejan.

Diseñar una API escalable y segura, garantizando la compatibilidad con los sistemas existentes de Bodytech, asegurando una integración fluida y sin interrupciones.

Ejecutar pruebas para evaluar el rendimiento de la API en condiciones reales de operación, asegurando su capacidad de manejar grandes volúmenes de datos.

Capacitar el área de TI de Bodytech en el uso de la API y documentar el trabajo, para aprovechar al máximo la información generada que sirva para la toma de decisiones.

1.4. Descripción de las actividades a desarrollar

Tabla 2.

Actividades a desarrollar en la pasantía

Desarrollar una API utilizando la inteligencia artificial Dapta que proporcione información en tiempo real sobre las operaciones de Bodytech, para optimizar la eficiencia operativa de la empresa.	Analizar la información interna de Bodytech, con el fin de hacer un reconocimiento de los flujos de datos que se manejan.	Recolectar y analizar la documentación existente sobre los sistemas de gestión de datos de Bodytech, identificando las bases de datos y los tipos de datos manejados.
		Realizar sesiones de trabajo con el personal de TI para comprender cómo se generan, almacenan y utilizan los datos en las operaciones diarias.
		Crear diagramas de flujo que representen el recorrido de la información desde su origen hasta su uso final en la toma de decisiones.
	Diseñar una API escalable y segura, garantizando la compatibilidad con los sistemas existentes de Bodytech, asegurando una integración fluida y sin interrupciones.	Definir los requisitos funcionales y no funcionales de la API.
		Investigar y seleccionar las tecnologías y frameworks más adecuados para el desarrollo de la API, asegurando compatibilidad con los sistemas existentes de Bodytech.
		Crear un prototipo inicial de la API que incluya las funcionalidades básicas y realizar pruebas de concepto para asegurar su funcionalidad básica.
		Desarrollar e implementar medidas de seguridad para proteger la API y garantizar la privacidad de la información manejada.
	Ejecutar pruebas para evaluar el rendimiento de la API en condiciones reales de operación, asegurando su capacidad de	Diseñar escenarios de prueba que simulen condiciones reales de operación.
		Ejecutar pruebas de carga y estrés para evaluar la capacidad

	manejar grandes volúmenes de datos.	de la API de manejar altos volúmenes de datos y usuarios simultáneos.
		Analizar los resultados de las pruebas para identificar áreas de mejora en el rendimiento de la API, incluyendo tiempos de respuesta y eficiencia de procesamiento.
		Implementar optimizaciones basadas en los resultados de las pruebas para mejorar el rendimiento y la escalabilidad de la API, asegurando que cumpla con los requisitos definidos.
	Capacitar el área de TI de Bodytech en el uso de la API y documentar el trabajo, para aprovechar al máximo la información generada que sirva para la toma de decisiones.	Crear materiales de capacitación detallados sobre el uso de la API.
		Organizar talleres de capacitación, proporcionando instrucción práctica y escenarios de uso realistas para familiarizarse con la API.
		Documentar y entregar el trabajo realizado durante el proceso de pasantía.

2. Enfoques Referenciales

2.1. Enfoque Conceptual

2.1.1. Phalcon

Phalcon es un framework full-stack de código abierto para PHP, que se distingue por su arquitectura única escrita como una extensión de C. Phalcon está acoplado de forma flexible, lo que permite a los desarrolladores utilizar solo los objetos que necesitan como componentes de unión según las necesidades de sus aplicaciones. (Phalcon Documentation, 2012).

Así mismo, según (Carreño Hernandez, 2019): “Phalcon, está basado en el patrón MVC e implementa su propio dialecto SQL llamado PHQL, útil para aplicaciones web BackEnd como FullStack, implementado como una extensión en el lenguaje C que ofrece un alto rendimiento y un menor consumo por cada petición logrando mayor rapidez”.

Dentro de las características más importantes de Phalcon se destacan que, los componentes están libremente acoplados, se tiene la libertad de usar todo el framework, o solo las partes que se necesiten, optimizaciones de bajo nivel que ayudan a reducir la sobrecarga requerida para correr aplicaciones MVC, las operaciones con base de datos se efectúan con la máxima eficiencia al usar un ORM para PHP escrito en C, se utilizan solo los módulos y bibliotecas que necesita (Yahuasi Apaza, 2020).

2.1.2. React

React es una biblioteca de JavaScript para renderizar interfaces de usuario. Las aplicaciones de React se construyen a partir de piezas independientes de UI llamadas *componentes*. Un componente de React es una función de JavaScript a la que le puedes agregar un poco de marcado (*markup*). Los componentes pueden ser tan pequeños como un botón, o tan grandes como una página entera (React Documentation, 2024).

React intenta ayudar a los desarrolladores a crear aplicaciones web complejas que utilizan un gran intercambio de datos. Su principio es sencillo declarative and composable. React solo se preocupará de la interfaz de usuario de tu app; si piensas en el paradigma de software Modelo-Vista-Controlador (MVC) React es únicamente la V. Esto hace que puedas utilizarlo combinándolo con otras librerías JavaScript o con frameworks como AngularJS, Ember o Backbone (Jimenez Villar, 2016).

Teniendo en cuenta (Holmes & Bray, 2015): “React Native cambia el juego con aplicaciones que son verdaderamente nativas. No usa WebView, sino que transpila2 JavaScript a los idiomas nativos” (Lazcano Calixto, Valencia González, & Baena Díaz, 2019).

2.1.3. Jenkins

Es un servidor de código abierto e integración continua que permite compilar y probar proyectos de software de forma continua. Escrito en Java, es accesible mediante una interfaz web y es multiplataforma (Management Modus, 2023).

Así mismo, Jenkins, una de las herramientas de integración continua más ampliamente utilizada. El software fue escrito en java y ha continuado desarrollándose con múltiples funciones que asisten la integración continua, el despliegue y la entrega continua. Puede instalarse a través de paquetes de sistemas nativos, docker o ejecutarse de forma independiente en cualquier máquina que tenga instalado java runtime enviroment (JRE) (Mercedes Brito, 2022).

Según (Mercedes Brito, 2022): “Al usar Jenkins, las compañías de software pueden acelerar su proceso de desarrollo del código, automatizando, agilizando y aumentando el proceso de toda la compilación y las pruebas de los proyectos”.

2.1.4. Redis

Es una base de datos NoSQL de tipo clave-valor distribuida en memoria. La durabilidad de los datos es opcional y soporta varias estructuras de datos y transacciones. Su nombre viene de REmote DIctionary Server, creado en 2009. El proyecto es open source y está escrito en el lenguaje de programación C. Dentro de sus ventajas, está la de su velocidad de acceso, tanto de lectura como de escritura (Fernandez O. , 2024).

Redis se distingue de los almacenes de datos NoSQL "tradicionales" como un componente auxiliar diseñado específicamente para mejorar el rendimiento de las aplicaciones. Dentro de las características más notables de Redis incluyen: Redis Sentinel (ayuda a los desarrolladores a calibrar sus instancias para que estén altamente disponibles para los clientes), Clúster de Redis (Esto permite un mayor rendimiento y escalabilidad de los despliegues de bases de datos), Redis Pub/Sub (los usuarios pueden diseñar servicios de chat

y mensajería de alto rendimiento en todas sus aplicaciones y servicios) y la Persistencia de Redis (puede conservar conjuntos de datos tomando instantáneas periódicas de los datos y agregándolas con los cambios a medida que estén disponibles) (IBM Documentation, 2024).

2.1.5. MySQL

MySQL es una base de datos relacional organizada en tablas que permite almacenar y gestionar datos de manera eficiente, basándose en relaciones definidas entre estas tablas mediante claves (primarias y foráneas) (Universidad Nacional de Colombia, 2022). Cada tabla contiene columnas y filas, representando las propiedades y los valores de los datos, respectivamente (Fernandez O. , 2024).

2.2. Enfoque Legal

Bodytech por ser una empresa legalmente constituida, se rige varias ciertas normas y leyes teniendo en cuenta el país donde se encuentren. Esto mismo es aplicado a cualquier desarrollo de software que se ejecuta en la empresa, con el objetivo de salvaguardar a todos y cada uno de sus clientes. A continuación, se mencionan las mismas:

- ***Política de privacidad:***

Que tiene el objetivo de garantizar la protección y buen manejo de los datos personales a los que da tratamiento y pone en conocimiento a sus titulares acerca de los derechos, procedimientos y finalidad a las que se someten (Politica de tratamiento de datos Bodytech, 2024). Las normas que desarrollan los preceptos aplicables a este documento son:

Para Colombia:

Constitución Política de Colombia (Art. 15)

Leyes 1266/08 y 1581/12

Decretos 1727/09, 2852/10, 1377/13 y 1047/15.

Perú:

Constitución Política del Perú: (Art. 2)

Ley: 29733.

Decreto Supremo: 003-2013-JUS.

- ***Sistema de Administración del Riesgo de Lavado de Activos de la Financiación del Terrorismo y Financiación de la Proliferación de Armas de Destrucción Masiva – SARLAFT / PADDM:***

Según lo establecido dentro de la Circular Externa 20211700000005-5 de 2021 de la Superintendencia de Salud (Consulting Gallagher, 2022).

- ***Subsistema de Administración del Riesgo de Corrupción, Opacidad y Fraude:***

Busca promover una cultura integra dentro de la empresa, llevando a una competencia sana, equilibrada y que preserva una actitud de integridad por parte de todos los involucrados (BodyTech M. , 2022).

- ***Ley 1581 2012:***

“Por la cual se dictan disposiciones generales para la protección de datos personales” (Ministerio de Tecnologías de la Información y las Comunicaciones, 2018).

- ***Decreto 1008 2018:***

"Por el cual se establecen los lineamientos generales de la política de Gobierno Digital y se subroga el capítulo 1 del título 9 de la parte 2 del libro 2 del Decreto 1078 de 2015, Decreto Único Reglamentario del sector de Tecnologías de la Información y las Comunicaciones” (MinTIC, 2018).

3. Informe de cumplimiento del trabajo de pasantía

3.1. Presentación de resultados

Teniendo en cuenta estudios realizados y la experiencia en el tiempo de pasantía, en el ámbito fitness, bienestar o de salud, la calidad y excelencia del servicio al cliente es fundamental para fidelizar los clientes y mantener su satisfacción. Esto, contando con tecnologías que estén a la vanguardia del mercado.

Por ello, se realizó el proyecto de pasantía con el fin de desarrollar una API utilizando la inteligencia artificial Dapta que proporcionara información en tiempo real sobre las operaciones de Bodytech, para optimizar la eficiencia operativa de la empresa. Para ello, se plantearon varios objetivos y actividades las cuales se cumplieron con éxito y se mostrarán a continuación:

El primer objetivo planteado fue analizar la información interna de Bodytech, con el fin de hacer un reconocimiento de los flujos de datos que se manejan, para ello se desarrollaron varias actividades.

3.1.1. Objetivo 1. Actividad 1: Recolectar y analizar la documentación existente sobre los sistemas de gestión de datos de Bodytech, identificando las bases de datos y los tipos de datos manejados.

La empresa Bodytech maneja la base de datos MySQL, base de datos relacional que maneja y se organiza en tablas, permitiendo almacenar y administrar los datos de manera eficiente, basándose en relaciones definidas entre estas tablas mediante claves (Universidad Nacional de Colombia, 2022). Así mismo, se manejan 3 ambientes de desarrollo, staging, el cual tiene su propia instancia de base de datos con los esquemas de los proyectos de la empresa, y pre producción y producción, los cuales comparten la misma instancia de base datos.

La infraestructura de la empresa se encuentra alojada en la nube, a través del proveedor Amazon Web Services (AWS). Esta infraestructura está basada en servidores dedicados que se configuran, mantienen y gestionan directamente por el personal encargado de la empresa lo que les otorga control total sobre los entornos de staging, preproducción y producción. El uso de servidores dedicados en AWS proporciona una mayor seguridad y aislamiento de recursos, ya que están reservados exclusivamente para la empresa, evitando compartir capacidad de procesamiento o almacenamiento con otros usuarios en la nube (AWS, 2024). Existen servidores dedicados para cada uno de estos ambientes, lo que permite aislar el trabajo en cada fase de desarrollo y despliegue, optimizando la seguridad y el rendimiento según las necesidades de cada entorno.

Tipos de datos y ambientes usados en el proyecto

Así también, en las bases de datos de los diferentes ambientes Staging, preproducción y producción, se manejan tipos de datos como: Cadena de texto (strings), números enteros, valores booleanos y números flotantes. A continuación, se explicarán los ambientes usados en el proyecto:

Staging

El ambiente de staging es utilizado por los desarrolladores para realizar pruebas de las tareas asignadas. Estas pruebas se ejecutan en una base de datos con un esquema idéntico al de producción, aunque en este ambiente no se utilizan datos reales. Existen servidores dedicados para este ambiente, los cuales, aunque no son los mismos que se usan en preproducción o producción, cuentan con características idénticas.

Los ambientes de staging y preproducción cuentan con integración continua (CI/CD), lo que significa que, cuando un desarrollador realiza un pull request al repositorio mediante herramientas de automatización como Jenkins, los cambios realizados se despliegan automáticamente (Aplyca, 2022).

Dado que el front-end es un proyecto compilado, los cambios se compilan y despliegan en horarios específicos, cuatro veces al día. En la nube de AWS, los servidores están programados para apagarse durante los períodos de inactividad, como los fines de semana y durante la noche.

El desarrollador debe garantizar que el código cumpla con los estándares establecidos en el analizador de código estático SonarQube, que según (G. Ann & Patroklos P, 2013): “sirve para ayudarlos a mejorar continuamente su código fuente, basados en los siete ejes de calidad principales que son: diseño/arquitectura, duplicaciones, comentarios, pruebas unitarias, complejidad, errores potenciales y reglas de codificación”.

Code review: Es una buena práctica que cada pull request subido al repositorio sea revisado y aprobado por un desarrollador capacitado para esta tarea. Si no se aprueba, el desarrollador debe realizar los ajustes indicados antes de que se realice una nueva revisión.

Cuando se realiza una modificación en la base de datos, es necesario enviar un correo al líder de desarrollo y al encargado de la base de datos para obtener la aprobación de los cambios.

Preproducción

En este ambiente, los product owner realizan las pruebas de los desarrollos que se aprobaron en staging, su propósito es probar cómo funcionará el sistema en condiciones reales, garantizando que el entorno esté lo más cercano posible al de producción, pero sin afectar a los usuarios finales. Si las pruebas son satisfactorias, los cambios son llevados al ambiente de producción.

Este ambiente también maneja horarios de despliegue. Como se mencionó inicialmente, este ambiente tiene la misma instancia de base de datos a producción, con el fin de garantizar que las pruebas realizadas sean lo más cercanas posibles al ambiente real.

Este ambiente solo es accesible para el equipo de product owners y algunos desarrolladores. Los cambios realizados en este ambiente se ven reflejados en el ambiente productivo, por este motivo se considera un ambiente muy sensible y de acceso restringido, cualquier difusión de las URLS correspondientes a este ambiente puede conllevar a errores e inconsistencias para los clientes finales. Así mismo, cada bug detectado en este ambiente es reportado a los desarrolladores para su respectiva corrección.

Producción

Este es el ambiente con el cual interactúan los clientes finales, es un ambiente que se destaca por su alta disponibilidad y escalabilidad, ya que debe ser capaz de manejar fallos sin interrumpir el servicio e incrementar la capacidad de usuarios, ya sea a través de un escalado vertical u horizontal.

Lo anterior se logra a través de la administración de infraestructura llevada a cabo desde la consola del proveedor de servicios de cloud.

3.1.2. Objetivo 1. Actividad 2: Realizar sesiones de trabajo con el personal de TI para comprender cómo se generan, almacenan y utilizan los datos en las operaciones diarias.

Metodología usada en la empresa

Para empezar a desarrollar esta actividad, fue importante comprender que en la empresa Bodytech se utiliza la metodología Scrum para el desarrollo de los proyectos de software. La cual es instrumento ágil que nace en la industria desarrolladora de software y

tecnología, la misma que ha demostrado que su uso, mejora los procesos y procedimientos (Velasco, Nuñez, Saltos, & Cunuhay, 2021).

Según (Hernández & Beltrán, 2020): “Scrum propone la aplicación de una metodología de trabajo más colaborativa entre las áreas involucradas en el desarrollo de los productos, trabajar de manera holística para lograr mejor comunicación, mayor integración y conocimiento por parte de todos los roles para así hacer del trabajo más ágil, con productos efectivos y mayor satisfacción de los clientes. Scrum se centra en centra en conocer lo mejor posible el equipo de trabajado identificando fortalezas y debilidades”.

Según: (Pérez Arbesú, 2022): “El departamento de TI se especializa en los recursos tecnológicos de una empresa. Esta área es todo un motor de innovación y automatización dentro de las compañías”. El área de TI está conformado por:

Equipo de Desarrollo: El desarrollador de software/analista de aseguramiento de calidad/ingeniero DevOps. Este equipo crea las aplicaciones y software que utilizas en tus ordenadores. Si se trata de un juego, un programa educativo o una herramienta de productividad, su tarea es diseñar y desarrollar software para facilitarles la vida. Así mismo, son todas las personas necesarias para hacer que un elemento del backlog del producto se transforme en un incremento del producto potencialmente entregable (Knowledge21, 2020).

Equipo de producto - Product manager: Como gerente de producto, usted es la persona de referencia para el producto. Está muy involucrado en los procesos y reclama la propiedad de los mismos a lo largo del ciclo de vida del producto. Desempeñando un papel importante en los aspectos de gestión de proyectos del desarrollo de productos, el PM se

comunica y colabora constantemente con los equipos de productos o con los equipos de productos integrados en toda la organización, así como con los clientes externos y los miembros de la junta directiva (Clark, 2024).

Product owner: El desafío del PO es construir una visión del producto y luego plasmarla en historias de usuario (user stories), que le ayudarán a transmitir esa visión al equipo de desarrollo que va a construir el software (Estayno & Meles, 2014).

Analista de producto: Un analista de productos es un profesional que ayuda a diseñar, desarrollar y mejorar productos o servicios (Agarwal, Gurumurthy, & Devansh, 2023). Realiza investigaciones, analiza datos y colabora con las partes interesadas. Su función es evaluar las necesidades y preferencias de los consumidores y desarrollar ideas para mejorar y potenciar los productos o servicios de una empresa (CVZEN, 2024).

Administración de base de datos: Un administrador de bases de datos, o DBA, es responsable del mantenimiento, la seguridad y el funcionamiento de las bases de datos, además de garantizar que los datos se almacenan y recuperan correctamente (Oracle, 2024).

BI (Business inteligent): Las funciones y responsabilidades dentro de estos equipos son cruciales, ya que permiten la recopilación, el análisis y la presentación de datos empresariales para informar las decisiones estratégicas y operativas (Alexander, 2024).

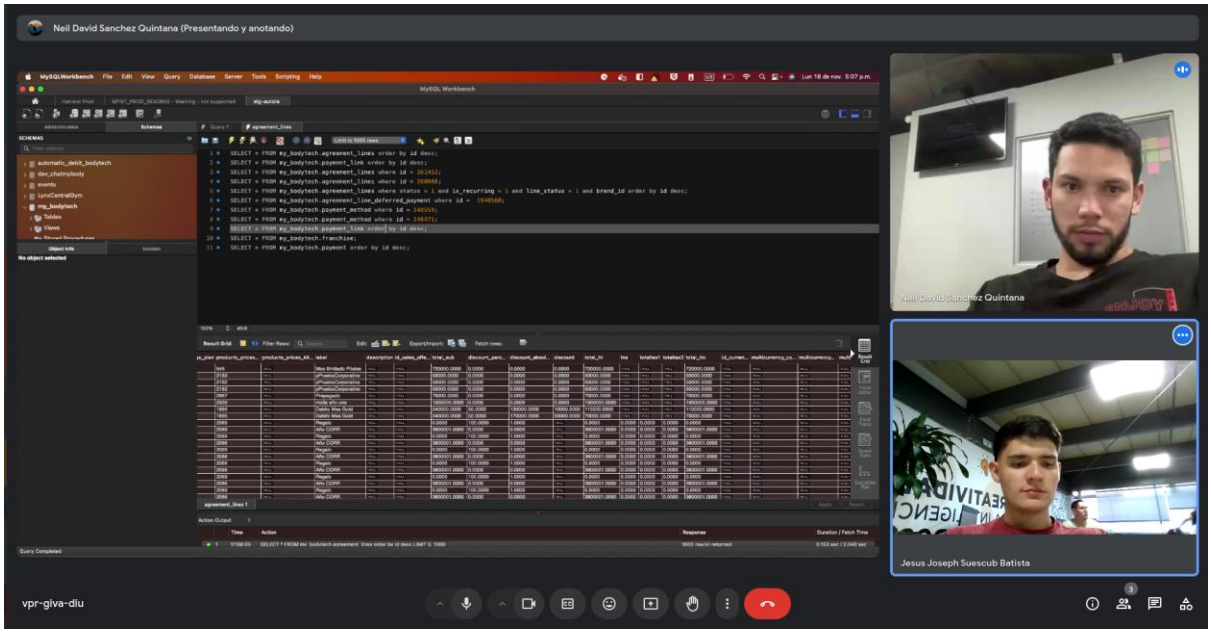
Soporte: Se encarga de resolver los problemas técnicos de una empresa. Apoyan a los clientes a través de diferentes canales y normalmente trabajan en varios niveles (Vilela, 2023).

Infraestructura: Los servidores son administrados por un equipo de infraestructura, mas no por el desarrollador. Cada modificación realizada al server debe dser solicitada al equipo de administrador.

A continuación, se muestra la primera sesión con el gerente de desarrollo, donde se realizó una contextualización de la empresa y del proyecto a desarrollarse.

Figura 5.

Reunión con el gerente de desarrollo



Generación de Información

Existen diversas maneras de generar información, dependiendo del contexto y las interacciones, ya sean por parte del usuario o del equipo operativo.

1. ***Generación de Información por Usuarios:***
    - Registro en plataformas digitales: Registro a través de la página web o la aplicación móvil.
    - Realización de compras: Información generada al efectuar una compra.
    - Pagos de cuotas: Se generan datos al realizar pagos de servicios prestados por la empresa.
    - Registro mediante código QR: Se genera información al escanear códigos QR para registrar actividades o accesos, por ejemplo, sesiones de entrenamientos personalizados.

2. ***Generación de Información Operativa:***
    - Programas de entrenamiento: Generación de planes de entrenamiento según el ACC () y los objetivos del usuario.
    - Reservas de clases grupales: Se genera información al registrar reservas para actividades programadas.
    - Sesiones de entrenamiento personalizado: Información generada al registrar las sesiones de entrenamiento individual en las sedes.
    - Creación de historias clínicas: Información registrada por el área médica en relación con la salud de los usuarios.

Todos los datos anteriores, se guardan en la instancia de la base de datos de producción, garantizando su disponibilidad para el análisis de datos y operaciones en tiempo real.

Descripción del flujo de los datos

El cliente le escala los requerimientos a la product manager, esta persona los evalúa, si son viables o no, luego con los product owner se escalan dependiendo el proyecto. Los product owner analizan el alcance del proyecto, desglosan a nivel de desarrollo requisitos funcionales y no funcionales, y con base en esto se calcula el tiempo de estimación. luego, en jira, se le asigna el requerimiento al scrum master, y este les asigna a los desarrolladores el desarrollo del requerimiento.

El desarrollador realiza su tarea; inicialmente, debe realizar pruebas locales y subir el desarrollo al ambiente de staging (ambiente de prueba), haciendo uso de la instancia de base de datos de este ambiente. Si las pruebas funcionan correctamente, se suben los cambios al ambiente preproducción. En esta etapa, el Scrum Master le entrega el desarrollo a los product owner para que estos realicen las pruebas de las funcionalidades y los requisitos, estas pruebas se realizan en la base de datos de preproducción. Si las pruebas funcionan correctamente en este ambiente, el desarrollo se implementa en el entorno de producción, ambiente que representa el sistema real en operación dentro de la compañía.

Por otra parte, si los product owner detectan un bug, se hace el proceso de reportar el bug al scrum master, y este se encarga de asignarle la tarea al desarrollador. Luego, se realiza el mismo proceso de desarrollo descrito anteriormente: el desarrollador realiza las

correcciones necesarias y realiza las pruebas en staging, luego sube los cambios a preproducción para que el desarrollo sea probado por los product owner y se verifique la correcta resolución del bug. Si las pruebas son exitosas, los cambios son implementados en el ambiente de producción.

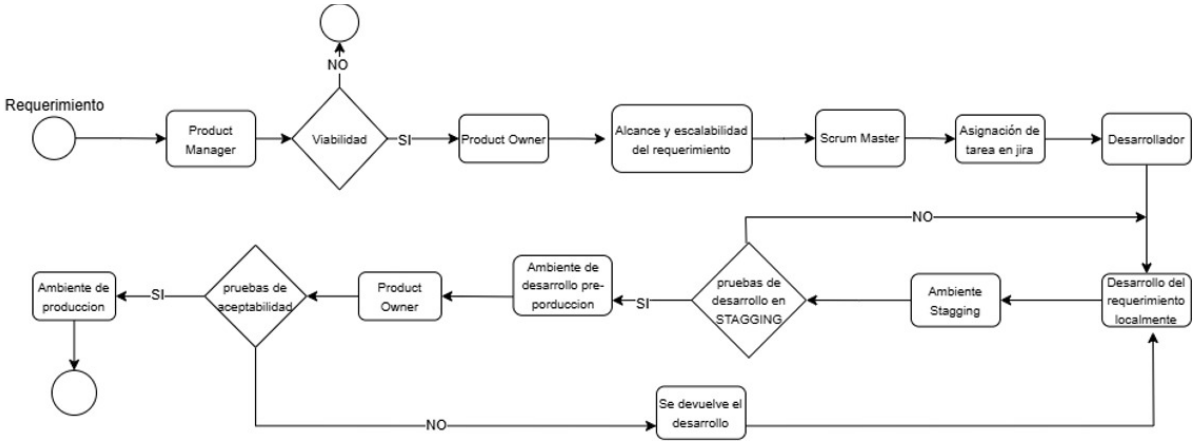
3.1.3. Objetivo 1. Actividad 3: Crear diagramas de flujo que representen el recorrido de la información desde su origen hasta su uso final en la toma de decisiones.

Se desarrollaron dos diagramas de flujo que reflejan el tratamiento de dos tipos principales de procesos dentro del sistema: la gestión de requerimientos y la resolución de errores bugs.

Flujo de Requerimientos: En este diagrama se describe cómo se procesan los requerimientos desde su solicitud inicial y análisis de su viabilidad, escalando por los diferentes roles y ambientes hasta su implementación final en producción.

Figura 6.

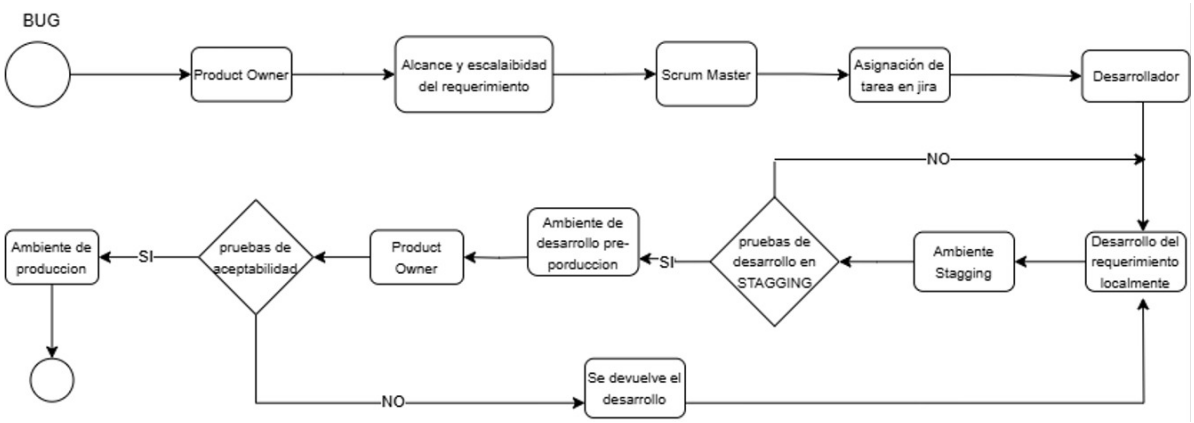
Flujo de requerimientos



Flujo de Bugs: Este diagrama detalla el manejo de errores reportados, desde su identificación por parte de los product owner, pasando por las correcciones realizadas por los desarrolladores, hasta la verificación final antes de ser implementadas en el ambiente de producción.

Figura 7.

Flujo de Bugs



Para el desarrollo del objetivo 2, que consistía en diseñar una API escalable y segura, garantizando la compatibilidad con los sistemas existentes de Bodytech, asegurando una integración fluida y sin interrupciones, se desarrollaron las siguientes actividades:

3.1.4. Objetivo 2. Actividad 1: Definir los requisitos funcionales y no funcionales de la API.

Requerimientos Funcionales Afiliados

Tabla 3.

R1. Crear y configurar el flujo en DAPTA que entregue la información personal de afiliados en Bodytech.

Nombre	**R1. Crear y configurar el flujo en DAPTA que entregue la información personal de afiliados en Bodytech.**
Actor	DAPTA - PÁGINA WEB BODYTECH
Partes interesadas	Afiliados a Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Configurar los nodos necesarios para administrar la información que entregará el chatbot a los afiliados de Bodytech.
Requisitos	Atributo / Obligatorio / Restricción Tipo de documento / X / Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo. Número de documento / X
Pre-requisito	Disponibilidad de las APIS que se van a consumir desde el flujo.
Post-requisito	

Tabla 4.

R2. Crear y configurar el flujo en DAPTA que entregue la información del contrato actual de afiliados en Bodytech.

Nombre	**R2. Crear y configurar el flujo en DAPTA que entregue la información del contrato actual de afiliados en Bodytech.**
Actor	DAPTA - PÁGINA WEB BODYTECH
Partes interesadas	Afiliados a Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Configurar los nodos necesarios para administrar la información que entregará el chatbot a los afiliados de Bodytech.

Requisitos			
	Atributo	Obligatorio	Restricción
	Tipo de documento	X	Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo.
	Número de documento	X	
Pre-requisito	Disponibilidad de las APIS que se van a consumir desde el flujo.		
Post-requisito			

Tabla 5.

R3. Crear y configurar el flujo en DAPTA que entregue la información de los planes y servicios incluidos en el contrato actual de afiliados en Bodytech.

Nombre	**R3. Crear y configurar el flujo en DAPTA que entregue la información de los planes y servicios incluidos en el contrato actual de afiliados en Bodytech.**		
Actor	DAPTA - PÁGINA WEB BODYTECH		
Partes interesadas	Afiliados a Bodytech, equipo de producto y de desarrollo, DAPTA.		
Descripción	Configurar los nodos necesarios para administrar la información que entregará el chatbot a los afiliados de Bodytech.		
Requisitos			
	Atributo	Obligatorio	Restricción
	Tipo de documento	X	Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo.
	Número de documento	X	

Pre-requisito	Disponibilidad de las APIS que se van a consumir desde el flujo.
Post-requisito	

Tabla 6.

R4. Crear y configurar el agente en DAPTA que interactúe con los afiliados de Bodytech.

Nombre	**R4. Crear y configurar el agente en DAPTA que interactúe con los afiliados de Bodytech.**
Actor	DAPTA - PÁGINA WEB BODYTECH
Partes interesadas	Afiliados a Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Configurar las instrucciones del agente de DAPTA para que interactúe con los afiliados, este le entregará información sobre su información personal asociada a Bodytech, su contrato y los servicios y planes que este incluye. Estas instrucciones incluyen el rol del agente, su objetivo, la estructura del flujo conversacional con el usuario, el manejo de errores, restricciones y verificación final de la conversación. Además, el agente debe ir conectado a los flujos que traen la información del afiliado, especificando en las instrucciones de que flujo debe basarse para responder las consultas del afiliado.
Requisitos	Atributo / Obligatorio / Restricción Tipo de documento / X / Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo. Número de documento / X
Pre-requisito	Realizar el registro de la siguiente información - Gestionar el flujo en DAPTA que entregue la información personal de afiliados en Bodytech (ver R1) - Gestionar el flujo en DAPTA que entregue la información del contrato actual de afiliados en Bodytech (ver R2)

	- Gestionar el agente en DAPTA que interactúe con los afiliados de Bodytech. (ver R3)
Post-requisito	

Requerimientos Funcionales Asesores

Tabla 7.

R5. Crear y configurar el agente en DAPTA que interactúe con los asesores comerciales de Bodytech.

Nombre	**R5. Crear y configurar el agente en DAPTA que interactúe con los asesores comerciales de Bodytech.**
Actor	MY BODYTECH - DAPTA
Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Configurar las instrucciones del agente de DAPTA para que interactúe con los afiliados, este le entregará información sobre el proceso de referidos y reporte de novedades, además de la oferta comercial actual de Bodytech. Estas instrucciones incluyen el rol del agente, su objetivo, la estructura del flujo conversacional con el usuario, el manejo de errores, restricciones y verificación final de la conversación. Además, en la memoria del agente se deben cargar documentos que contienen la información comercial de la compañía la cual le debe entregar el chatbot al asesor.
Requisitos	Atributo / Obligatorio / Restricción message — X — Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo. chat_id — X documento — X tipo_documento — X user_rol — X
Pre-requisito	Disponibilidad de los documentos que se deben cargar en la memoria del agente.
Post-requisito	

Tabla 8.

R6. Crear y configurar el flujo en DAPTA para la conexión del agente con la API de comunicación.

Nombre	**R6. Crear y configurar el flujo en DAPTA para la conexión del agente con la API de comunicación.**
Actor	DAPTA - MY BODYTECH
Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Configurar los nodos necesarios para realizar la conexión del agente con el api de comunicación.
Requisitos	Atributo / Obligatorio / Restricción message — X — Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo. chat_id — X documento — X tipo_documento — X user_rol — X
Pre-requisito	Realizar el registro de la siguiente información: - R5. Crear y configurar el agente en DAPTA que interactúe con los asesores comerciales de Bodytech.
Post-requisito	

Tabla 9.

R7. Gestionar la conexión de la API de comunicación con DAPTA.

Nombre	**R7. Gestionar la conexión de la API de comunicación con DAPTA.**
Actor	MY BODYTECH
Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Desarrollar la lógica necesaria en un proyecto realizado en Phalcon para recibir por parámetros los atributos definidos en DAPTA.

De			
	Atributo	Obligatorio	Restricción
	message	X	Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo.
	chat_id	X	
	documento	X	
	tipo_documento	X	
	user_rol	X	
Pre-requisito	Realizar el registro de la siguiente información: - R5. Crear y configurar el agente en DAPTA que interactúe con los asesores comerciales de Bodytech. - R6. Crear y configurar el flujo en DAPTA para la conexión del agente con la API de comunicación.		
Post-requisito			

Tabla 10.

R8. Diseñar la interfaz del chatbot.

Nombre	**R8. Diseñar la interfaz del chatbot.**		
Actor	MY BODYTECH		
Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.		
Descripción	Diseñar en la aplicación de Figma un diseño intuitivo del chatbot para implementarlo en la interfaz del software empresarial MY-BODYTECH.		
Requisitos			
	Atributo	Obligatorio	Restricción
Pre-requisito	Realizar el registro de la siguiente información:		

	- R5. Crear y configurar el agente en DAPTA que interactúe con los asesores comerciales de Bodytech. - R6. Crear y configurar el flujo en DAPTA para la conexión del agente con la API de comunicación. - R7. **Gestionar** la conexión de la API de comunicación con DAPTA.
Post-requisito	

Tabla 11.

R10.Gestionar la comunicación entre el cliente y la API de DAPTA.

Nombre	**R10.Gestionar la comunicación entre el cliente y la API de DAPTA.**
Actor	MY BODYTECH
Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Implementar la interfaz del chatbot con la librería de React.js, garantizando que el frontend envíe los datos necesarios a través del protocolo HTTP al API REST.
Requisitos	Atributo / Obligatorio / Restricción message — X chat_id — X documento — X tipo_documento — X user_rol — X Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo.
Pre-requisito	Realizar el registro de la siguiente información - R5, R6, R7, R8, R9

Tabla 12.

R12. Crear la restricción del chatbot por rol y por marca.

Nombre	**R12. Crear la restricción del chatbot por rol y por marca.**
Actor	My Bodytech

Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Lógica en el frontend para que el chatbot solo sea visible para los asesores comerciales.
Requisitos	

Atributo	Obligatorio	Restricción
user_rol	X	El administrador con acceso al software my-bodytech

Pre-requisito	Realizar el registro de la siguiente información R5, R6, R7, R8, R9, R10.
Post-requisito	

Tabla 13.

R13. Crear la tabla del log en la base de datos.

Nombre	**R13. Crear la tabla del log en la base de datos.**
Actor	My Bodytech
Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Administrar la información relacionada con el registro de interacciones del asesor comercial con el chatbot.
Requisitos	

Atributo	Obligatorio	Restricción
id	X	Desarrollador con acceso al software my-bodytech.
user_profile_id	X	
chat_id	X	
user_creator	X	
created_at	X	
updated_at	X	
created_at_db	X	
organization_id	X	
company_id	X	
brand_id	X	

Pre-requisito	Realizar el registro de la siguiente información - R5, R6, R7, R8, R9, R10, R11

Post-requisito

Tabla 14. *R14. Gestionar el registro de las interacciones con el chatbot mediante API REST.*

Nombre	**R14. Gestionar el registro de las interacciones con el chatbot mediante API REST.**
Actor	My Bodytech
Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.
Descripción	Desarrollar un servicio en PHALCON que permita registrar las interacciones con el chatbot en la base de datos.
Requisitos	Atributo / Obligatorio / Restricción id — X — Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo. user_profile_id — X chat_id — X user_creator — X created_at — X updated_at — X created_at_db — X organization_id — X company_id — X brand_id — X
Pre-requisito	Realizar el registro de la siguiente información - R3. Gestionar el agente en DAPTA que interactúe con los asesores comerciales de Bodytech.
Post-requisito	Los registros de la interacción del usuario con el chatbot deben reflejarse en la tabla de la base de datos.

Tabla 15.

R15. Gestionar la comunicación entre el cliente y la API REST de registro de logs.

Nombre	**R15. Gestionar la comunicación entre el cliente y la API REST de registro de logs.**
Actor	My Bodytech
Partes interesadas	Asesor comercial Bodytech, equipo de producto y de desarrollo, DAPTA.

Descripción	Logica en Frontend para el registro de los logs.
Requisitos	

Atributo	Obligatorio	Restricción
id	X	Solo el administrador del proyecto con acceso a DAPTA puede configurar el flujo.
user_profile_id	X	
chat_id	X	
user_creator	X	
created_at	X	
updated_at	X	
created_at_db	X	
organization_id	X	
company_id	X	
brand_id	X	

Pre-requisito	Realizar el registro de la siguiente información - R5, R6, R7, R8, R9, R10, R11, R12, R13, R14
Post-requisito	

Requerimientos No Funcionales:

1. **Rendimiento**:
    - La API debe ser capaz de manejar múltiples solicitudes simultáneas sin degradación en el tiempo de respuesta.
    - El chatbot debe responder en tiempo real para proporcionar una experiencia interactiva fluida al usuario.
2. **Seguridad**:
    - Implementar medidas de seguridad como autenticación y autorización en las Apis, asegurando que solo usuarios autorizados puedan acceder a información personal.
    - Encriptación de datos sensibles, como la información personal y detalles de los planes del usuario.
3. **Escalabilidad**:
    - El sistema debe ser escalable para permitir la expansión a medida que Bodytech crezca o incremente su base de usuarios.
4. **Mantenibilidad**:
    - El código del backend y frontend debe ser modular y fácil de mantener, siguiendo buenas prácticas de programación.
    - La arquitectura de la API en Dapta debe permitir agregar o modificar nodos sin afectar el flujo general.

5. **Disponibilidad**:
    - La API debe estar disponible de manera continua para garantizar que los usuarios puedan acceder a la información cuando lo necesiten, con un tiempo de inactividad mínimo.
6. **Compatibilidad**:
    - La interfaz debe ser compatible con navegadores modernos y dispositivos móviles para asegurar una experiencia de usuario fluida y accesible desde diferentes plataformas.
7. **Tiempo de respuesta del chatbot**:
    - El chatbot debe responder de manera eficiente, priorizando la rapidez y la calidad de las respuestas para asegurar una interacción satisfactoria con los usuarios.

3.1.5. Objetivo 2. Actividad 2: Investigación y Selección de Tecnologías y Frameworks para el Desarrollo de la API

Para el desarrollo de la API de Bodytech, se eligieron tecnologías que optimizan la eficiencia y escalabilidad de la solución. En primer lugar, la plataforma no-code Dapta fue seleccionada por su capacidad para integrar múltiples fuentes de datos y automatizar flujos con inteligencia artificial mediante una interfaz intuitiva de arrastrar y soltar (Dapta Documentation, 2024). Esta herramienta reduce significativamente los tiempos de desarrollo, permitiendo ajustes rápidos y personalizados para cada área de la empresa. Además, Dapta cuenta con una funcionalidad de "memoria corporativa" que adapta sus respuestas en función de interacciones previas, mejorando la experiencia del usuario con respuestas más precisas y adaptadas a cada contexto. La flexibilidad para integrarse con sistemas internos asegura una experiencia omnicanal fluida para los usuarios de Bodytech.

Para el backend, se optó por Phalcon, un framework PHP de alto rendimiento, desarrollado en C para maximizar la velocidad y optimizar el uso de recursos. Su arquitectura modular basada en el patrón MVC facilita el desarrollo de una API robusta y escalable, con un ORM integrado que agiliza las consultas y gestiona eficientemente los datos en la base de datos MySQL. En el frontend, se seleccionó React.js para la construcción de interfaces de usuario interactivas y dinámicas que ofrecen una experiencia ágil y en tiempo real. Esta estructura facilita la gestión de servicios, planes y consultas de los usuarios, integrando toda la información de manera fluida y accesible, lo cual optimiza la experiencia de quienes interactúan con el sistema.

Dapta: Plataforma No-Code para el Desarrollo de Backends y APIs

Dapta es una solución no-code para crear backends y APIs sin necesidad de escribir código extenso. Permite integrar diversas fuentes de datos y configurar APIs basadas en la lógica del negocio, acelerando el desarrollo backend hasta un 90% (Dapta Documentation, 2024). Esto es clave para Bodytech, ya que libera al equipo de tareas complejas de integración, permitiéndoles enfocarse en las funcionalidades específicas de la API de chatbot.

Según (Dapta Documentation, 2024), ofrece tres funcionalidades clave para potenciar el uso de inteligencia artificial en diversos entornos empresariales:

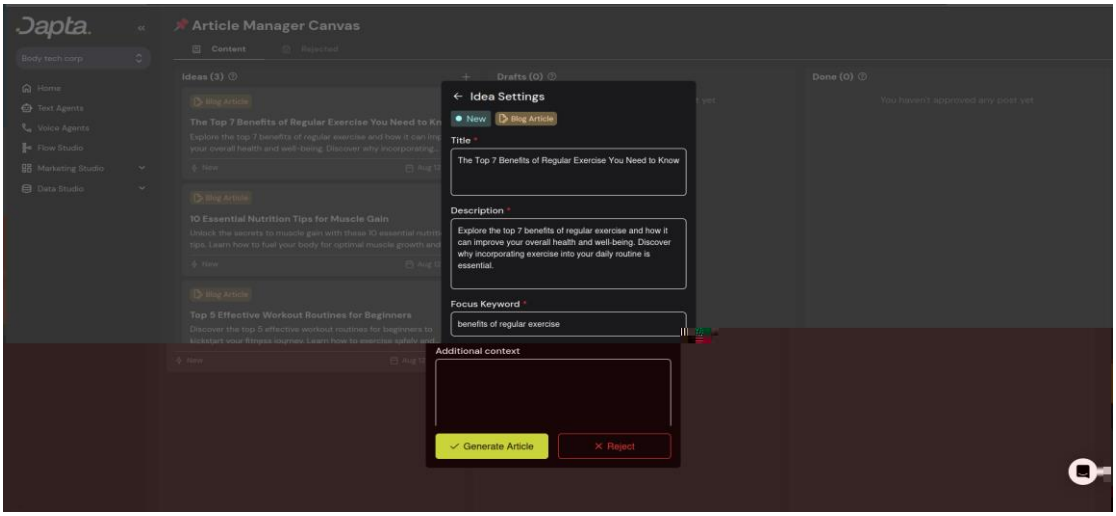
- ***Automatización de Contenido con IA**:*

Dapta automatiza la creación de contenido personalizado con IA, adaptándose al estilo de la marca. Genera artículos y contenido optimizado para SEO, desde la ideación hasta la publicación, con investigación automática y generación de ideas. Además, permite entrenar el

modelo en solo tres minutos, escalando eficientemente la estrategia de contenido y mejorando el posicionamiento en buscadores (Dapta Documentation, 2024).

Figura 8.

Automatización de Contenido con IA

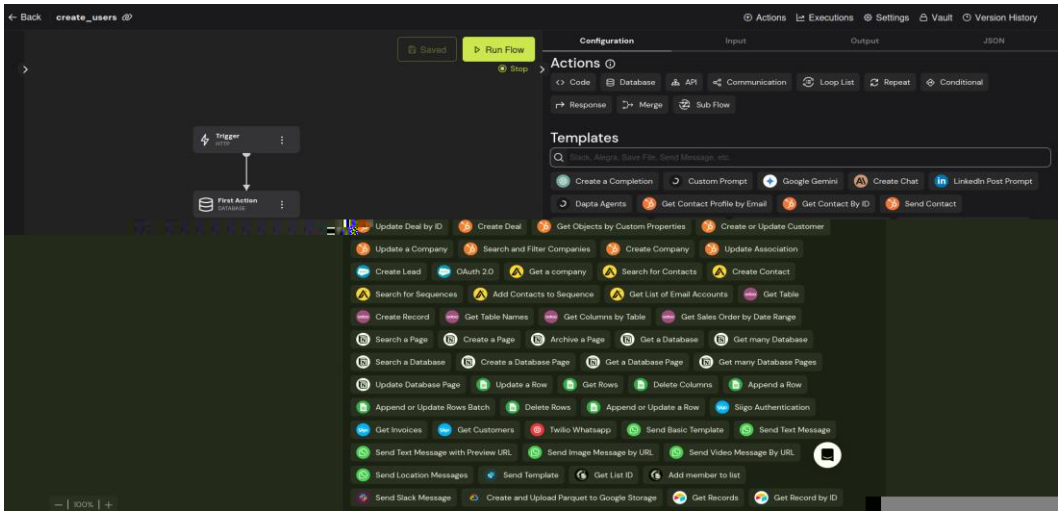


- ***Automatización del Flujo de Trabajo con IA:***

La herramienta Flow Builder de Dapta facilita la integración de IA en procesos empresariales, conectando bases de datos y sistemas como CRM y ERP para optimizar flujos de trabajo. Su "memoria corporativa" aprende de cada interacción, mejorando la eficiencia y reduciendo tareas repetitivas. Además, permite ajustar modelos de IA según el caso de uso sin necesidad de código (Dapta Documentation, 2024).

Figura 9.

Automatización del Flujo de Trabajo con IA

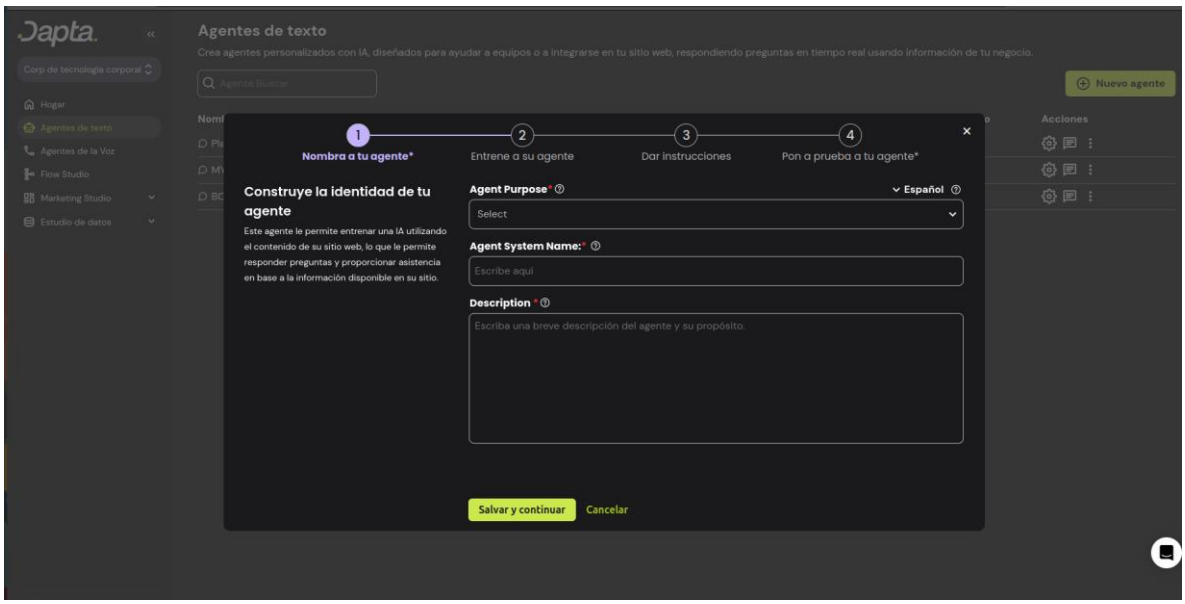


- ***Agente con IA:***

El agente de IA de Dapta agrega un chatbot interactivo al sitio web, brindando soporte en tiempo real y en varios idiomas. Facilita la atención a usuarios, mejora la experiencia y captura clientes potenciales. Su configuración sencilla permite implementación rápida y soporte 24/7 sin necesidad de personal adicional (Dapta Documentation, 2024).

Figura 10.

Agente con IA



Ventajas de Dapta para el Proyecto

Teniendo en cuenta el proyecto y la investigación de varios autores como: (Dapta Documentation, 2024), (Navarro, 2024). La selección de Dapta se fundamenta en varios aspectos clave que la hacen altamente compatible con los requerimientos del proyecto:

- ***Automatización del Flujo de Trabajo con IA:*** Dapta permite automatizar flujos de trabajo mediante IA, optimizando procesos y reduciendo tareas repetitivas. En este proyecto, facilita la creación de chatbots especializados para distintas áreas de la empresa, mejorando el flujo de información y tareas.

- ***No-Code para Rápida Implementación:*** Dapta es una plataforma no-code que permite crear APIs de manera visual, reduciendo el tiempo y los costos de desarrollo. Su interfaz intuitiva facilita la integración rápida de chatbots en las operaciones diarias sin necesidad de soporte técnico intensivo.

- ***Memoria Corporativa e Integración Inteligente:*** La "memoria corporativa" de Dapta aprende de cada interacción, mejorando la precisión y personalización de los chatbots. Esto optimiza el tiempo y la eficiencia, ofreciendo respuestas más adaptadas a las necesidades de cada área.

- ***Elección y Flexibilidad en el Modelo de IA:*** Dapta permite seleccionar modelos de IA según las necesidades específicas de cada área, como ventas o atención médica. Esto ofrece flexibilidad para ajustar el chatbot a distintos roles sin necesidad de grandes cambios en su configuración.

- ***Integración con Herramientas Externas y Soporte Omnicanal:*** Dapta se integra fácilmente con herramientas externas como CRM y plataformas de redes sociales, asegurando un enfoque coordinado. Además, el chatbot ofrece soporte omnicanal, mejorando la accesibilidad y experiencia del usuario en distintos puntos de contacto.

Phalcon: Un Framework PHP de Alto Rendimiento

Phalcon es un framework open-source diseñado para PHP y basado en el patrón MVC (Model-View-Controller). Lo que lo distingue de otros frameworks es su arquitectura única como extensión de C, que lo hace uno de los frameworks más rápidos y eficientes de PHP. En lugar de estar compuesto de archivos PHP que el servidor interpreta en tiempo de ejecución, Phalcon está escrito en C y se ejecuta como una extensión, reduciendo la sobrecarga en el servidor. Esta característica permite que el código de la API se ejecute más rápido, utilizando menos memoria y logrando así un rendimiento notablemente más alto que otros frameworks PHP tradicionales (Phalcon Documentation, 2012).

Estructura de Phalcon

Según la documentación oficial de Phalcon, la estructura de Phalcon se basa en el patrón MVC, proporcionando una organización clara y modular que facilita la escalabilidad y el mantenimiento de la API. A continuación, se explica la estructura principal del framework:

- ***Modelo (Model):*** El componente de Modelo en Phalcon gestiona la lógica de la base de datos y las relaciones de entidades mediante un ORM integrado, simplificando la interacción con MySQL y agilizando las consultas.

- ***Vista (View):*** Phalcon ofrece un sistema de vistas eficiente para renderizar datos y enviar respuestas JSON en APIs REST, ideal para aplicaciones y servicios de terceros.
- ***Controlador (Controller):*** El controlador define la lógica de negocio, rutas y respuestas de la API. Phalcon facilita la configuración de rutas y endpoints REST, manteniendo una clara separación de responsabilidades.
- ***Micro Aplicaciones (Phalcon Micro):*** Phalcon permite crear APIs REST minimalistas con su modo "Micro", eliminando componentes innecesarios y optimizando el tiempo de respuesta, ideal para Bodytech.

Ventajas de Phalcon para el Desarrollo de API REST

Phalcon se presenta como una opción ideal para el desarrollo de una API REST en Bodytech debido a varias ventajas que aporta en términos de rendimiento, simplicidad, y escalabilidad. A continuación, se describen las ventajas clave de Phalcon para este proyecto:

- ***Alto Rendimiento y Bajo Consumo de Recursos:*** Phalcon, basado en C, ofrece excelente rendimiento y bajo consumo de memoria, ideal para manejar múltiples peticiones simultáneas y garantizar tiempos de respuesta rápidos incluso bajo carga.
- ***Facilidad para Crear Rutas y Endpoints:*** Phalcon permite configurar rutas y endpoints de manera simple y eficiente, facilitando el desarrollo, mantenimiento y escalabilidad de APIs REST.
- ***Seguridad Integrada:*** Phalcon incluye mecanismos de seguridad como filtros de inyección SQL, cifrado de datos, autenticación JWT, y manejo seguro de contraseñas, protegiendo la privacidad y datos sensibles de los usuarios.

- ***ORM Integrado y Manejo de Base de Datos:*** Su ORM simplifica la interacción con la base de datos, permitiendo operaciones CRUD ágiles y facilitando la gestión de relaciones entre tablas, ideal para datos como usuarios y planes.
- ***Facilidad para Construir Respuestas en JSON:*** Phalcon permite generar respuestas JSON de manera directa, optimizando la integración con aplicaciones frontend y otros sistemas que consumen la API.
- ***Flexibilidad y Soporte para Middleware:*** Phalcon soporta middleware para gestionar solicitudes, aplicar validaciones y seguridad, asegurando la integridad y confiabilidad de la API.
- ***Escalabilidad y Modularidad:*** Con su arquitectura modular, Phalcon facilita la expansión de la API conforme a las necesidades de Bodytech, permitiendo agregar nuevos módulos sin afectar el rendimiento.

React.js: Una Biblioteca JavaScript para Interfaces de Usuario Dinámicas

React.js se centra en la creación de interfaces de usuario dinámicas y modulares, organizando la UI en pequeños componentes reutilizables. Su enfoque basado en un "árbol de componentes" facilita el desarrollo de aplicaciones con actualizaciones en tiempo real, ideal para el frontend de la API REST de Bodytech.

Estructura de React.js

La estructura de React.js se basa en la creación y organización de componentes. Un proyecto en React generalmente sigue una arquitectura modular que facilita el mantenimiento

y la expansión. A continuación, se detallan los componentes principales de la estructura de React.js:

Componentes

Los componentes son el núcleo de React y representan bloques independientes y reutilizables que forman la interfaz de usuario. Existen tipos de componentes en React:

- ***Componentes:*** React usa componentes funcionales (más recomendados) y de clase, que son bloques reutilizables de la interfaz de usuario.
- ***Estado (State):*** Permite que los componentes mantengan datos cambiantes, actualizando la interfaz cuando el estado cambia. Se puede gestionar con hooks como useState o herramientas como Redux.
- ***Props (Propiedades):*** Los componentes reciben datos a través de props, facilitando la comunicación entre ellos.
- ***Hooks:*** Funciones como useState, useEffect, y useContext permiten manejar el estado y ciclo de vida en componentes funcionales.
- ***React Router:*** Facilita la navegación dentro de la app, útil para páginas como inicio de sesión y consultas de planes.

Ventajas de React.js para el Desarrollo de una Interfaz Frontend de una API REST

React.js ofrece múltiples ventajas para el desarrollo de interfaces frontend, especialmente cuando se trata de integrar una API REST. Estas son algunas de las principales ventajas que React aporta al proyecto de Bodytech:

- ***Componentes Reutilizables y Modularidad:*** Permite crear interfaces consistentes y fáciles de mantener mediante componentes reutilizables.
- ***Virtual DOM:*** React optimiza las actualizaciones de la interfaz, mejorando el rendimiento y la experiencia de usuario, especialmente bajo cargas altas.
- ***Manejo de Estados con Hooks:*** Hooks como useState y useEffect permiten manejar el estado y realizar llamadas a la API de forma eficiente.
- ***Integración Fácil con APIs REST:*** React facilita la integración con APIs REST usando herramientas como axios o fetch, permitiendo la comunicación asíncrona con el backend.
- ***Flujo de Datos Unidireccional:*** El flujo de datos controlado facilita la depuración y mantiene la consistencia de los datos a través de la aplicación.
- ***Ecosistema y Comunidad Activa:*** React tiene un ecosistema robusto de librerías y una comunidad activa, ofreciendo herramientas como Redux y Material-UI para mejorar el desarrollo.

3.1.6. Objetivo 2. Actividad 3: Crear un prototipo inicial de la API que incluya las funcionalidades básicas y realizar pruebas de concepto para asegurar su funcionalidad básica.

Inicialmente, antes de comenzar con el desarrollo de la API, se llevaron a cabo varias capacitaciones por parte de los asesores de DAPTA, esto con el fin de conocer la estructura de la plataforma y sus funcionalidades para poder adaptarlos en este proyecto según los requerimientos y la lógica del negocio.

Figura 11.

Evidencia 1. Capacitación asesores de DAPTA

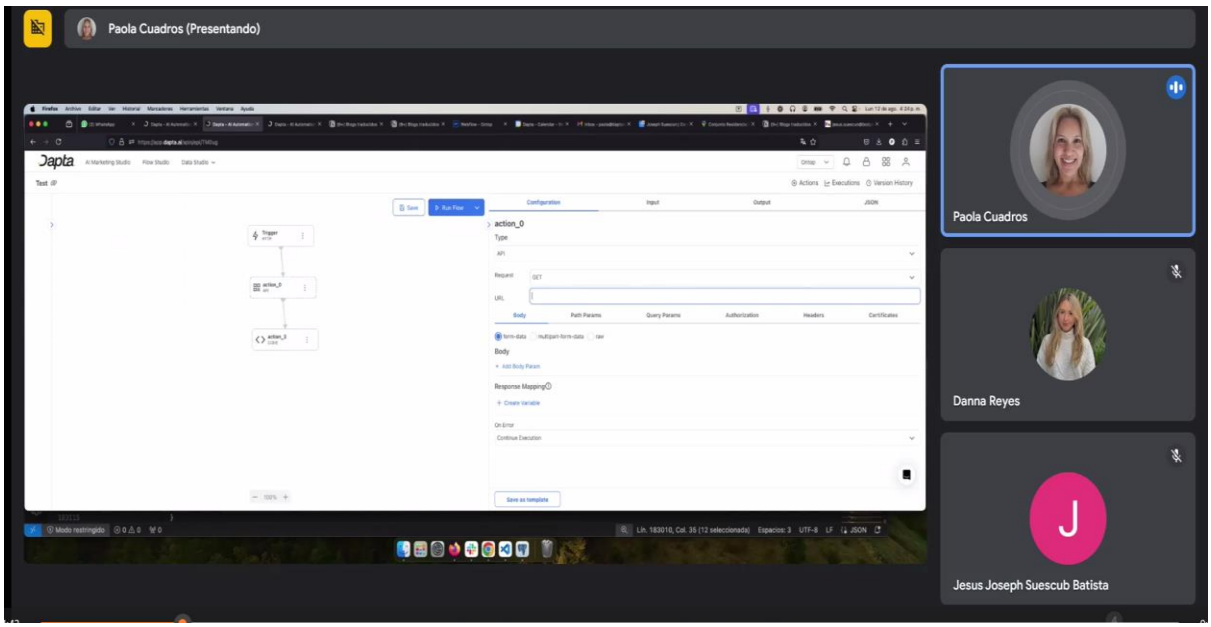


Figura 12.

Evidencia 2. Capacitación asesores de DAPTA

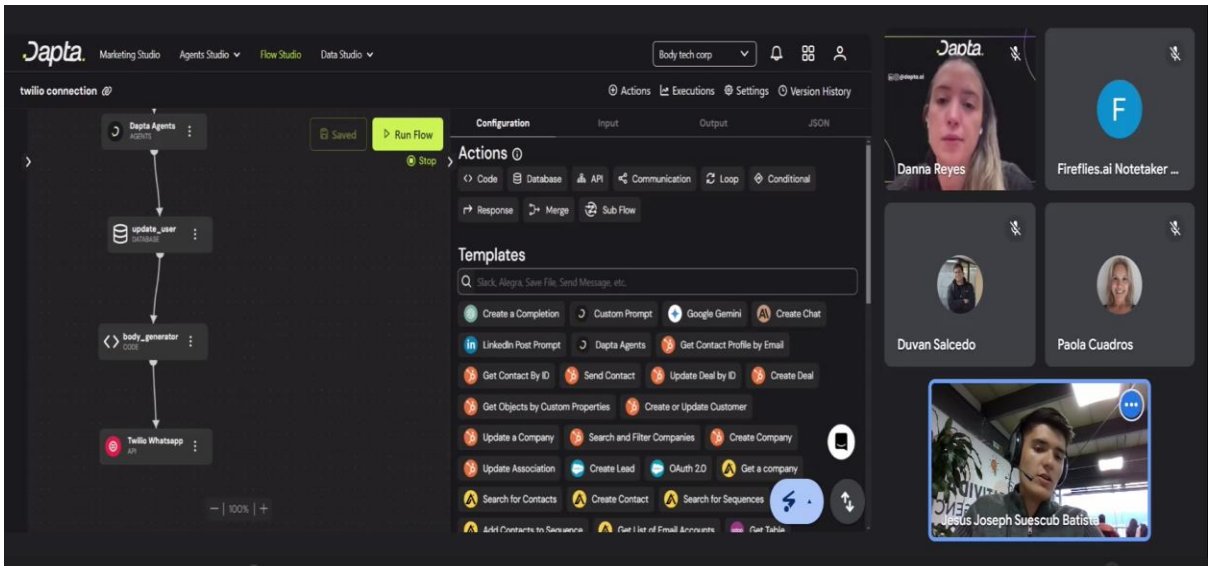


Figura 13.

Evidencia 3. Capacitación asesores de DAPTA

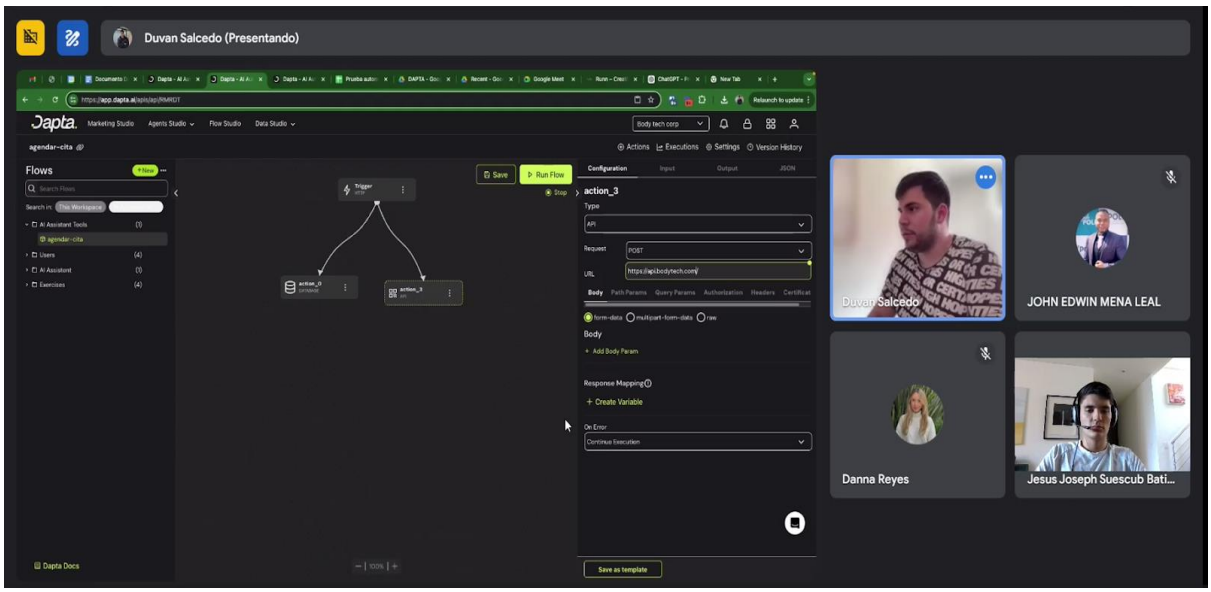
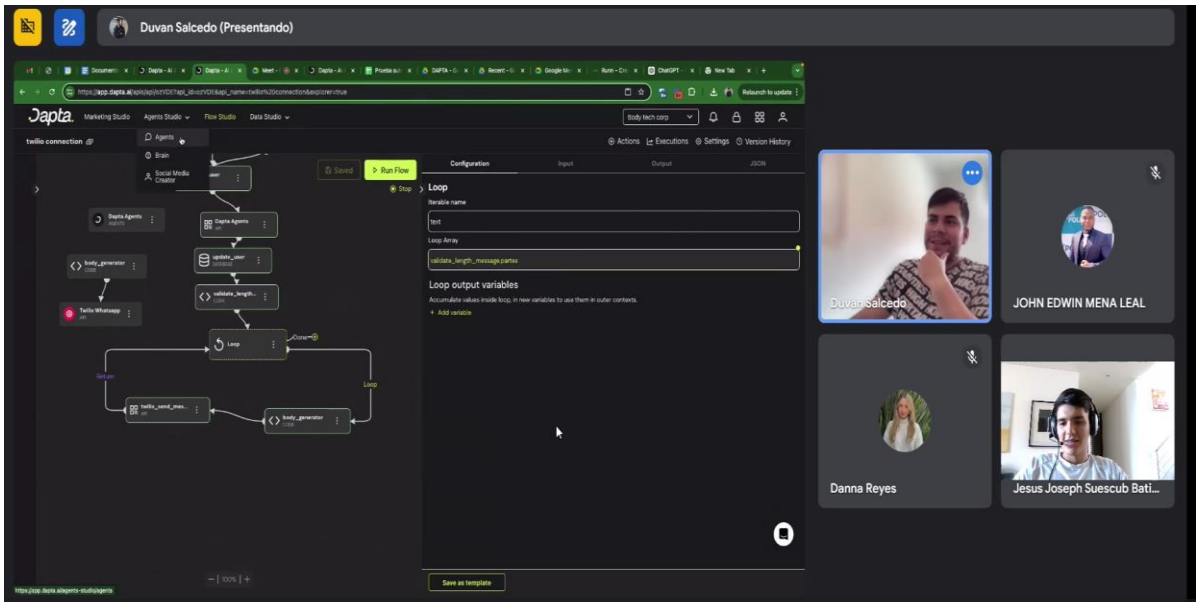


Figura 14.

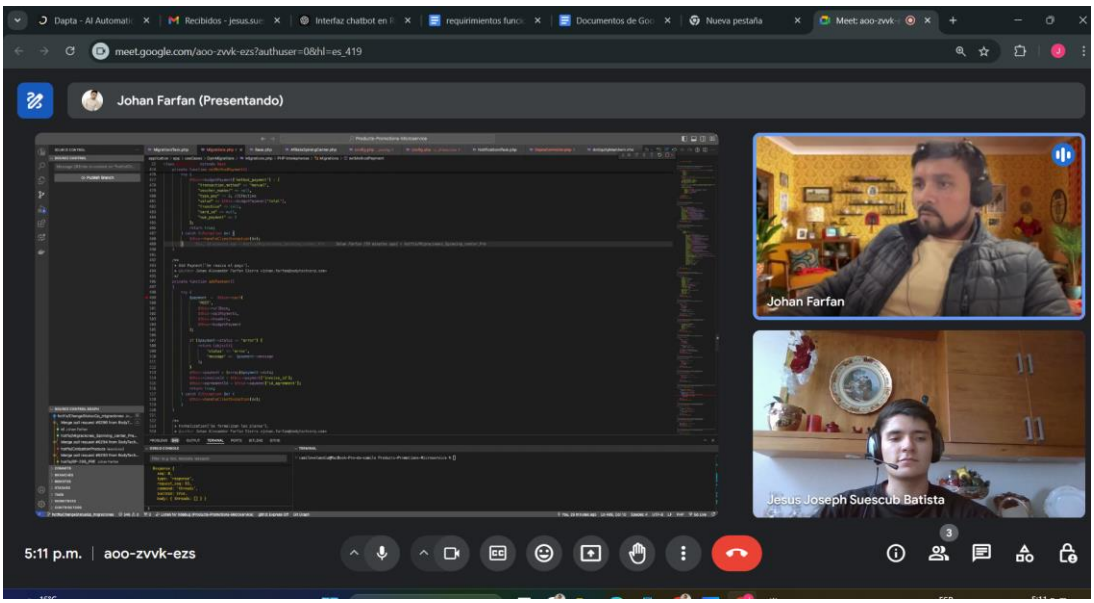
Evidencia 4. Capacitación asesores de DAPTA



Además, también se realizaron reuniones con desarrolladores de la empresa para entender la estructura de los proyectos en backend y front-end.

Figura 15.

Reunión con desarrollador para entender la estructura del proyecto



Desarrollo Api

Luego de distintas reuniones, y siguiendo las recomendaciones dadas por los asesores de DAPTA, se inició el desarrollo de un PROTOTIPO inicial de la API en la plataforma, enfocado en el requerimiento de entregar información personal a los afiliados de Bodyetch, para esto, inicialmente se debe desarrollar un flujo, el cual está conformado por nodos que representan acciones, templates o decisiones dentro del proceso de automatización.

Este flujo debe ir conectado a un agente, el cual debemos configurar con las instrucciones con las que deseamos que el agente le dé una respuesta correcta y eficaz al usuario, además, Dapta nos proporciona un script que representa la interfaz del agente para usar en nuestro proyecto, este agente responderá según las instrucciones recibidas y según las

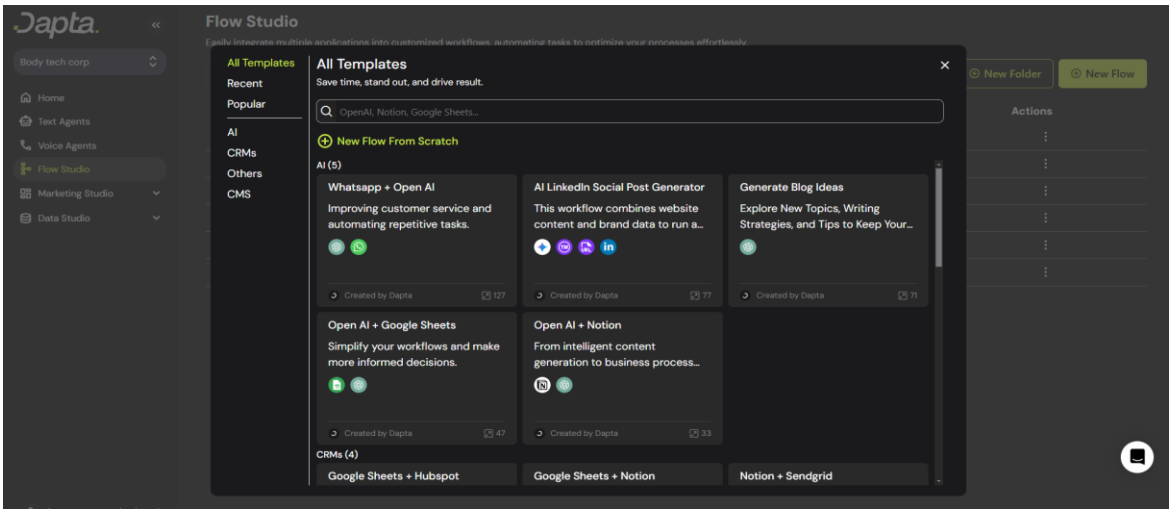
respuestas configuradas en el flujo desarrollado. Al tener el flujo configurado, contamos con la URL de la API, la cual es la encargada de inicializar este flujo.

Paso a Paso de la Creación De Flujo

- Inicialmente, luego de iniciar sesión en dapta, damos click en el apartado de flow studio, luego click en nuevo flujo, y, por último, nuevo flujo desde cero.
-

Figura 16.

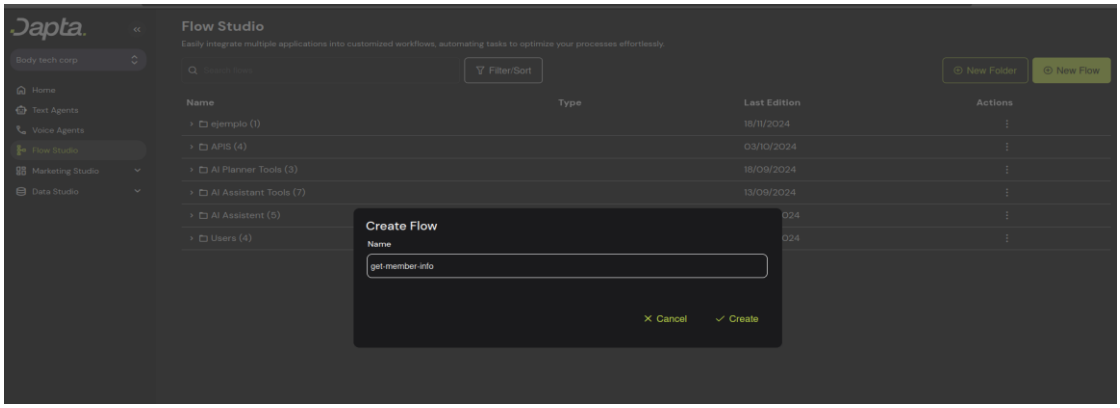
Inicio de sesión en DAPTA



- Luego, procedemos a darle nombre al flujo, y le damos click en crear.

Figura 17.

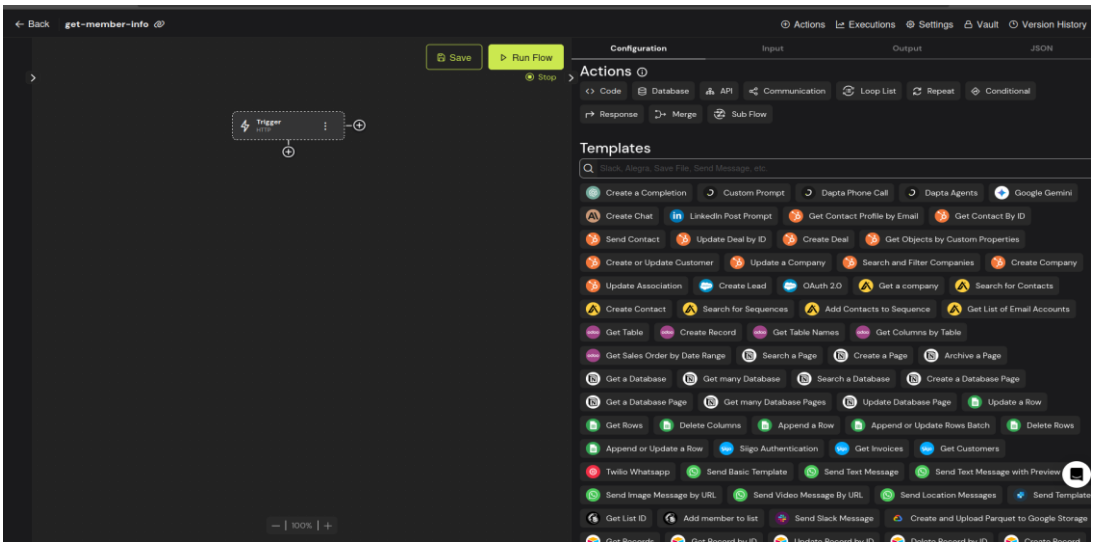
Nombrar flujo



- Después de crear el flujo, nos aparecerá la siguiente vista, donde está el nodo del trigger y, a su lado, se encuentra el apartado de configuración del flujo, el cual está conformado por acciones y templates representadas por un nodo, los cuales podremos añadir según las necesidades del proyecto.

Figura 18.

Muestra de Nodos



Se quiere desarrollar un flujo que le entregue a los afiliados de Bodytech, su información personal existente en la empresa. Para esto, luego de realizar una reunión con el equipo de desarrollo, se identificó una API desarrollada por el equipo la cual entrega la información deseada, por consiguiente, teniendo en cuenta que DAPTA proporciona un nodo para maneja APIS, se definió crear el flujo con tres nodos:

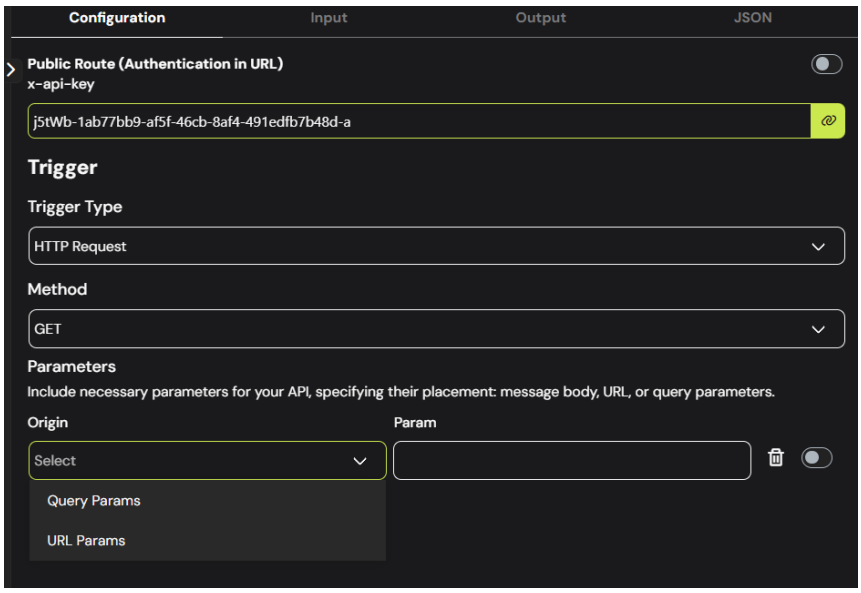
- Nodo de trigger: Condiciones o parámetros que activan el flujo
- Nodo API: En este nodo se configura la API definida con el equipo de desarrollo.
- Node Response: En este nodo se configura la respuesta que entregara el flujo.

Nodo Trigger

En un flujo de automatización, un trigger actúa como el punto inicial que establece la condición para activar el proceso. Es el nodo encargado de detectar o escuchar un estímulo externo, como una solicitud HTTP, una modificación en una base de datos o un evento en una aplicación, para comenzar la ejecución del flujo correspondiente.

Figura 19.

Muestra de un trigger

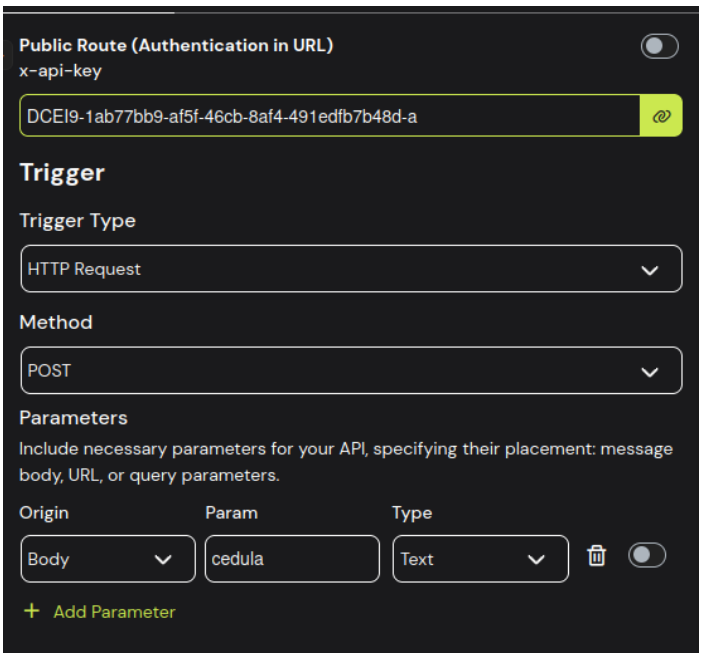


El trigger debe configurarse de la siguiente manera:

- **Tipo de Trigger:** Se indica que el flujo se iniciará al recibir una solicitud HTTP.
- **Método:** Definimos el método HTTP que activará el flujo (GET, POST, PUT, PATCH, DELETE) en este caso, método POST, ya que mediante este método enviamos datos en el cuerpo de la solicitud.
- **Parámetros:** En este apartado definimos los parámetros que el flujo necesita recibir para continuar con su ejecución. Acá definimos de qué manera se enviarán los parámetros, estos pueden ser por body query params, URL params, en este caso, enviamos como parámetro la cédula, la cual es tipo text.

Figura 20.

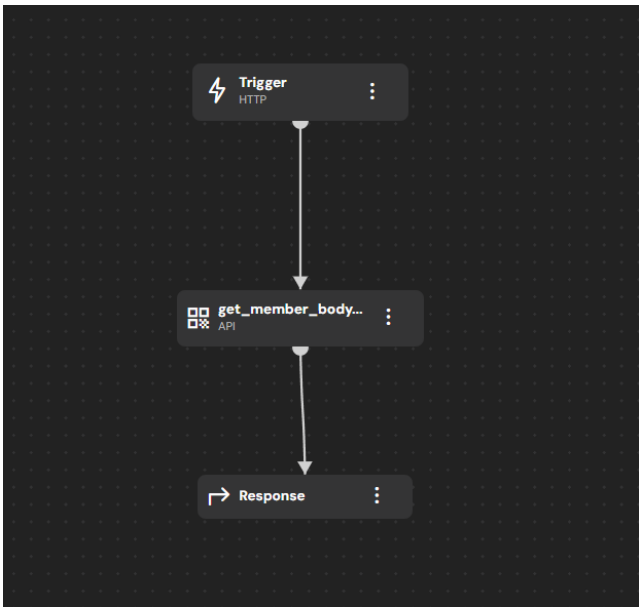
Configuración de Trigger



- Luego, añadimos el nodo de API conectado al nodo de trigger, le asignamos un nombre a este nodo (get_member_bodytech) y lo conectamos con el nodo de respuesta(response).

Figura 21.

Asignación de nombre al nodo

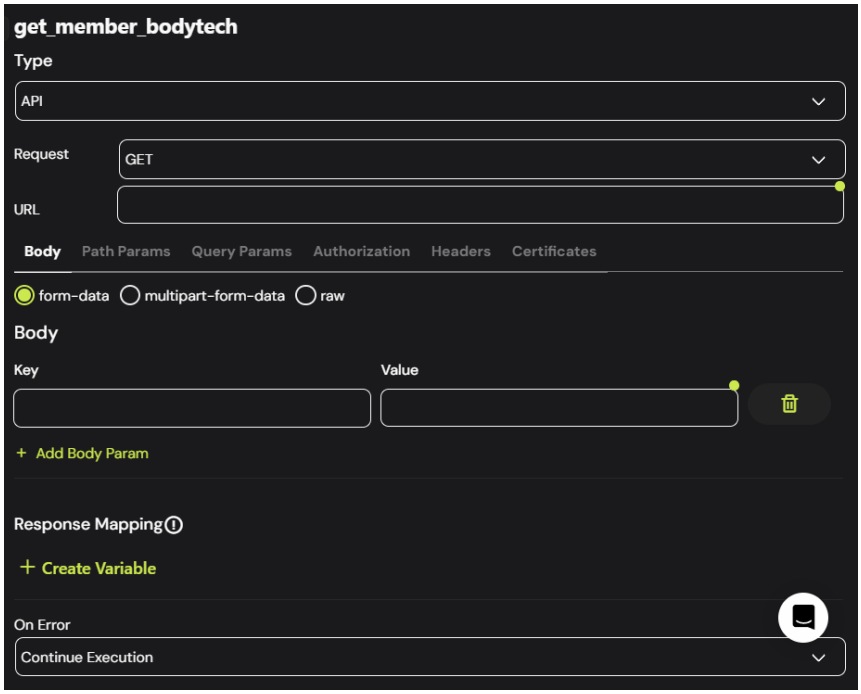


Nodo Api

Este nodo tiene la función de consumir la API que retorna la información relacionada a los perfiles de los usuarios(members) de Bodytech.

Figura 22.

Estructura del Nodo Api



Configuración básica del nodo:

- **Type:** indicamos cual es el medio mediante el cual el nodo se conecta a un servicio externo (Databse, API, Code, File, Communication).
- **Request:** Tipo de solicitud HTTP que se requiere para obtener datos de un servidor, (GET, POST, PUT, PATCH, DELETE).
- **URL:** URL del endpoint que se quiere consumir para consultar la información.

Estructura del nodo:

1. Parámetros del endpoint

- **Body:** En este apartado se añaden los parámetros que se envían al servidor como parte de la solicitud.

- **Path Params:** Se refiere a los parámetros que forman parte de la URL. Se utilizan para identificar recursos específicos en un endpoint.
- **Query Params:** ¿son pares clave-valor que se envían en la URL después de un signo? Se utilizan para filtrar las consultas al servidor.
- **Authorization:** Se refiere al mecanismo que utiliza el flujo para autenticar la solicitud al servidor. Este puede incluir el tipo de autorización y los datos que esta necesita (tokens, claves de API o credenciales), con el fin de garantizar los permisos del usuario que va a acceder a la información mediante la API del flujo.
- **Header:** Son metadatos enviados junto con la solicitud HTTP. Pueden incluir información como el tipo de contenido, tokens de autenticación, o configuración específica del cliente.
- **Certificates:** Se utiliza para autenticar al cliente o servidor mediante certificados digitales. Esto con el fin de asegurar que la comunicación entre cliente y servidor se encuentre encriptada y protegida.

2. Response Mapping: Configuración que permite asignar los datos de la respuesta de un nodo a una variable clave que se utilizará en otros nodos dentro del flujo

3. On Error: En este apartado se realiza la configuración del manejo de errores del flujo.

Resultado del Nodo Trigger

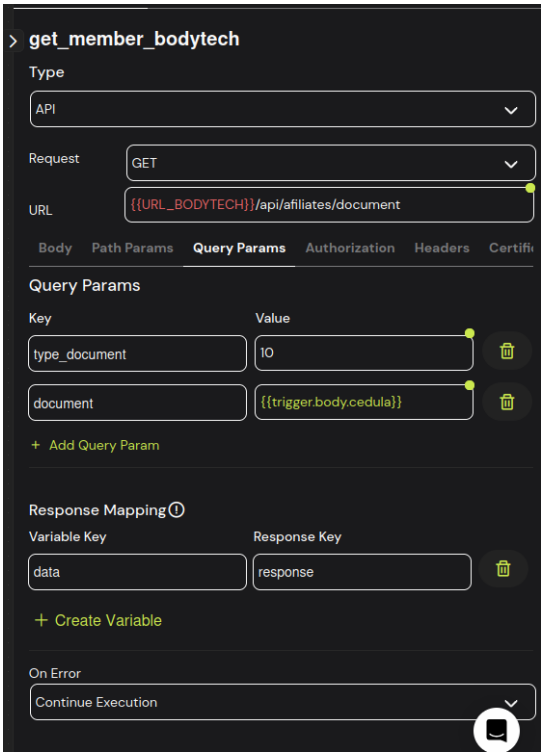
Se seleccionó type como API, el tipo de solicitud (Request) será GET ya que solamente se quiere consultar la información del perfil del usuario, y luego procedemos a adjuntar la URL de la API. Además, para enviar los parámetros los haremos por el apartado de query params, ya que estos son ideales para enviar filtros o parámetros de búsqueda. No se utiliza Body ya que solo manejaremos consultas mediante el método GET, ni Path Params porque el diseño del endpoint no requiere incluir identificadores en la URL.

Estos dos parámetros serán type_document, el cual tendrá como valor 10, el cual es el valor establecido dentro desarrollo de la API dada por Bodytech para referirse a cédula de ciudadanía, y documento, el cual traerá como valor la cédula que viene desde el trigger.

En Response Mapping, en la Variable Key, se utiliza una variable llamada data para almacenar el resultado procesado de la respuesta, y en Response Key, utilizamos la variable response para referirnos al objeto o campo del cuerpo de la respuesta API que contiene los datos que necesitamos.

Figura 23.

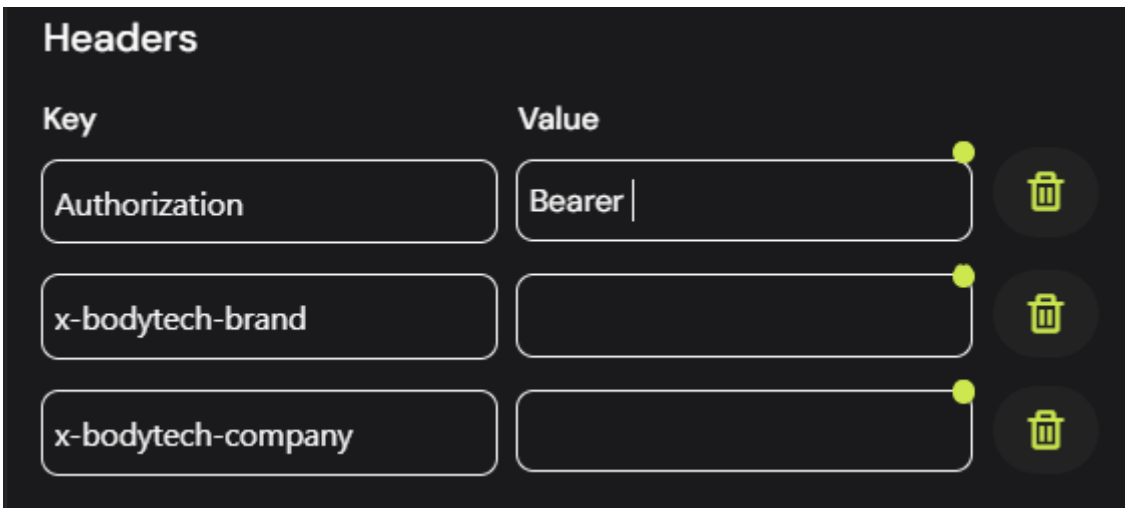
Evidencia uso del Response Key



Luego, se procede a realizar la configuración del apartado de los Headers, donde se configura el token de autorización y segmentan los datos de la entidad.

Figura 24.

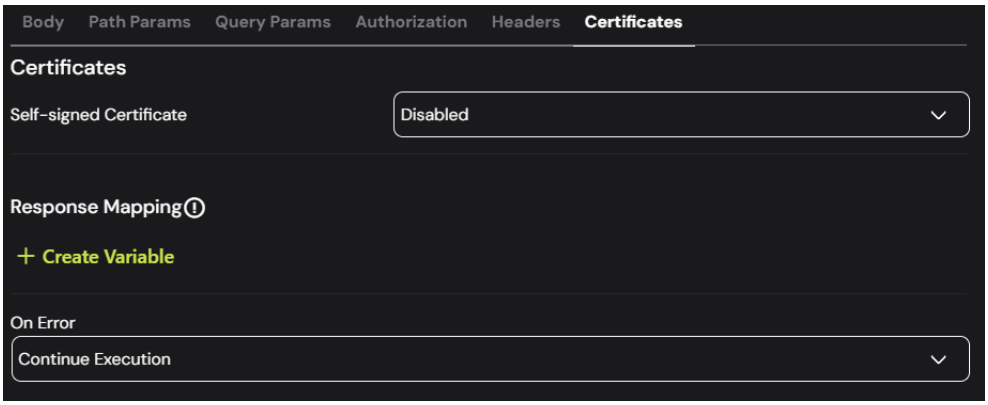
Configuración del apartado de los Headers



Y, por último, no se necesitan certificates entonces van deshabilitados.

Figura 25.

Evidencia de certificados deshabilitados

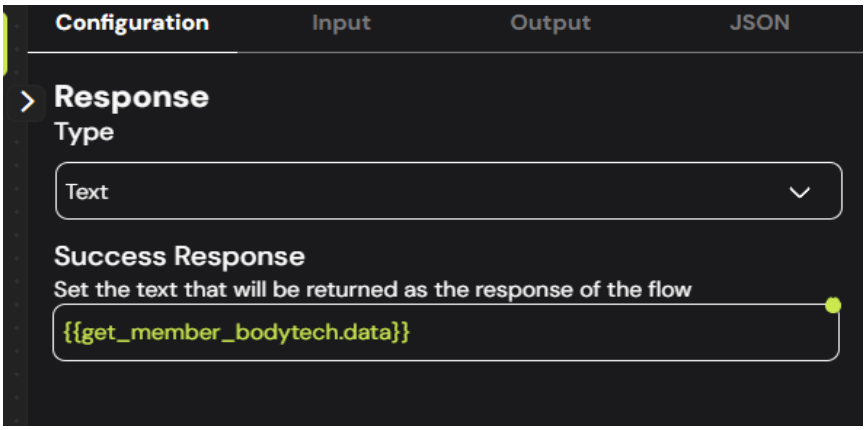


Nodo Response

En este nodo configuramos la respuesta del flujo, inicialmente configuramos el tipo de respuesta del flujo, estas pueden ser de tipo objeto o text, en este caso será de texto, y luego, configuramos el contenido de la respuesta, acá podremos mapear las respuestas configuradas en la sección Response Mapping de los nodos anteriores, en este caso, mapeamos la respuesta configurada en el nodo API (data).

Figura 26.

Nodo Response

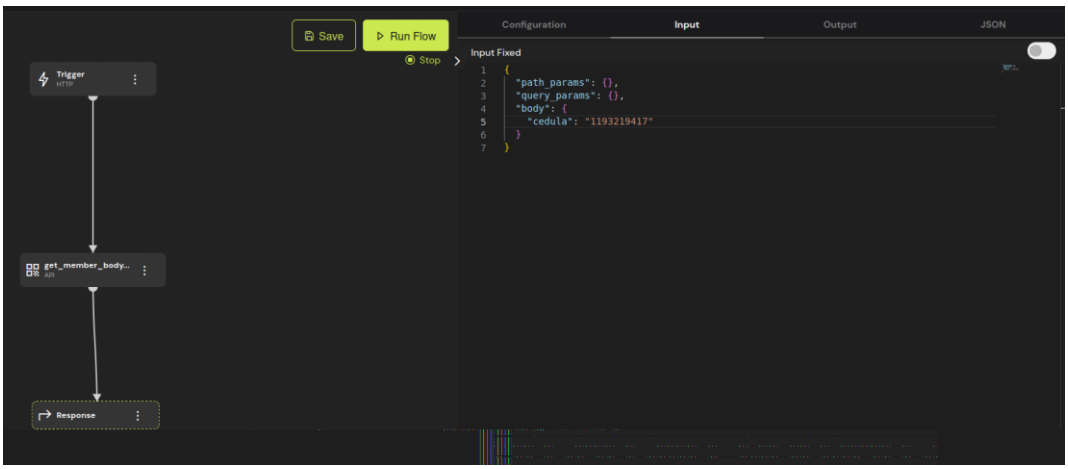


Ejecución Del Flujo

Luego de terminar la configuración del flujo, procedemos a ejecutarlo localmente en la plataforma, para esto, nos vamos al apartado de input, y agregamos un número de cédula, el cual era el parámetro definido en el nodo de trigger y le damos click en Run Flow.

Figura 27.

Ejecución Del Flujo

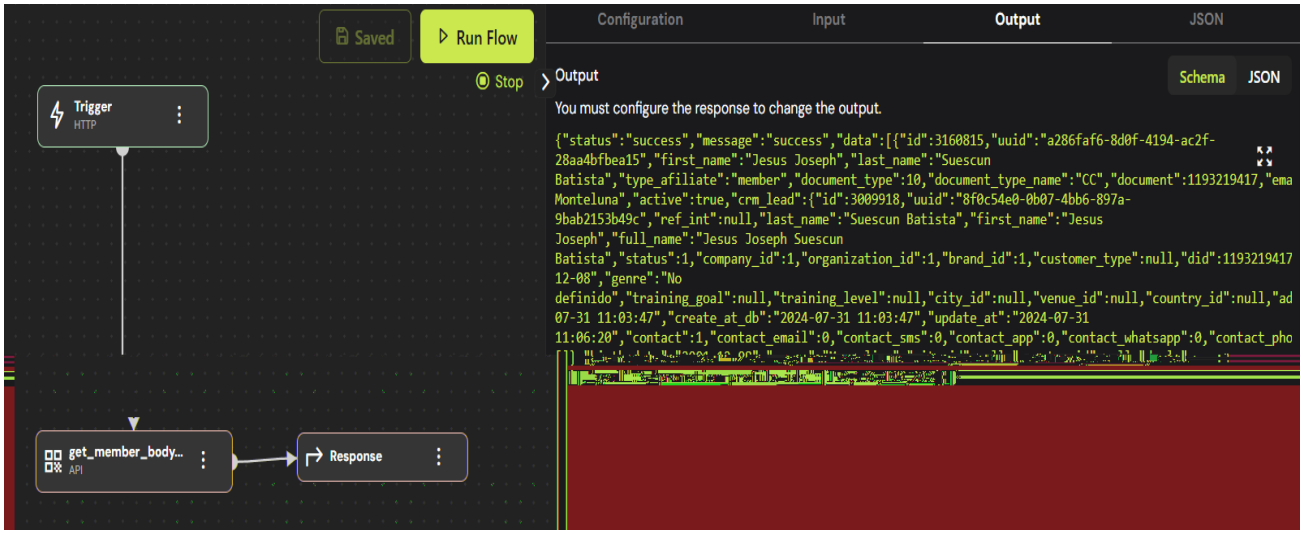


Resultado del Nodo Response

Luego de ejecutar el flujo, nos dirigimos al apartado de output para observar la salida de la ejecución, acá apreciamos la correcta respuesta del flujo, ya que nos entrega la información correspondiente a la API que configuramos en el nodo.

Figura 28.

Resultado del Nodo Response



Además, podemos acceder al JSON de la ejecución, para saber en llegado caso donde fallo el flujo.

Figura 29.

Acceso al JSON de la ejecución



Creación De Agente

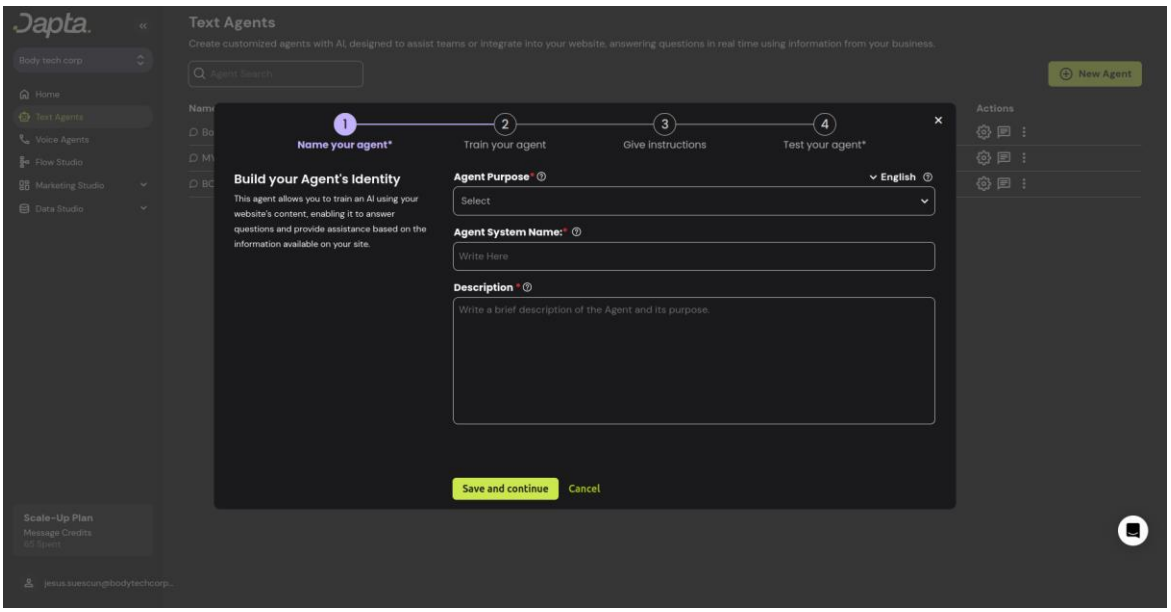
Luego, después de que el flujo inicial funcionó correctamente, se procede a crear el agente, para esto, nos dirigimos al menú inicial de la plataforma de DAPTA, y damos click en

el apartado de text agent, luego en nuevo agente, y procedemos a realizar su configuración, donde seleccionamos el tipo de agente (agente de ecommerce, agente sitio web, agente personalizado), el nombre del agente, y la descripción del del objetivo principal del agente.

Este agente está basado en IA generativa, (AWS Documentation, 2024) la define como: “es un tipo de inteligencia artificial (IA) capaz de crear nuevos contenidos e ideas, como imágenes y videos, así como de volver a utilizar aquello que sabe para resolver nuevos problemas.”

Figura 30.

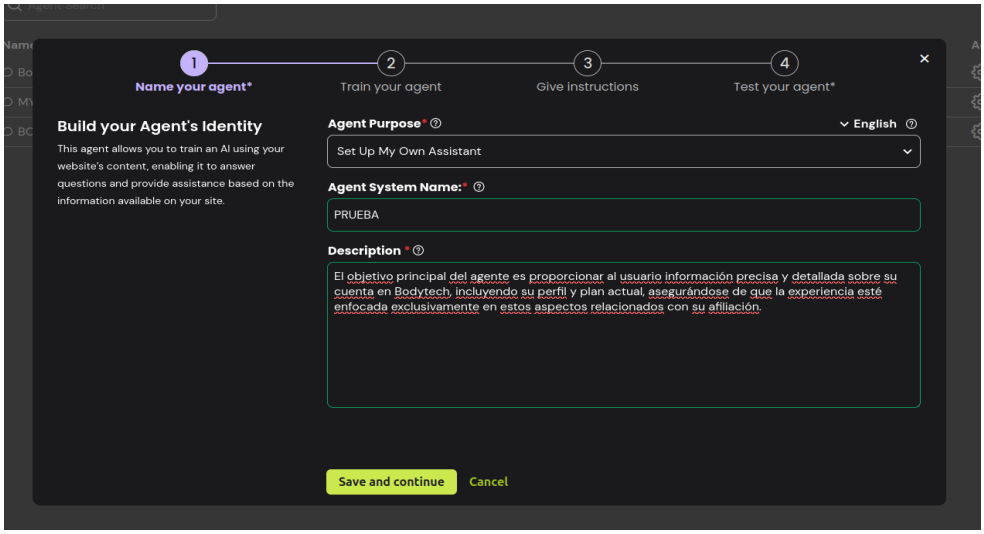
Creación De Agente



Resultado Creación De Agente

Figura 31.

Resultado Creación De Agente



Posteriormente, continuamos con la configuración del agente, en este caso, podemos cargar una URL de un sitio web, también podemos cargar un archivo, el cual lo recomendable es que sea un archivo de texto, esto con el fin de suministrarle información al agente, la cual se requiere para responder las consultas de los usuarios, esta información pueden ser documentos directos de la lógica del negocio de la empresa. En esta prueba inicial, pondremos únicamente el enlace que dirige a la página web de Bodytech.

Figura 32.

Configuración del agente con carga de URL

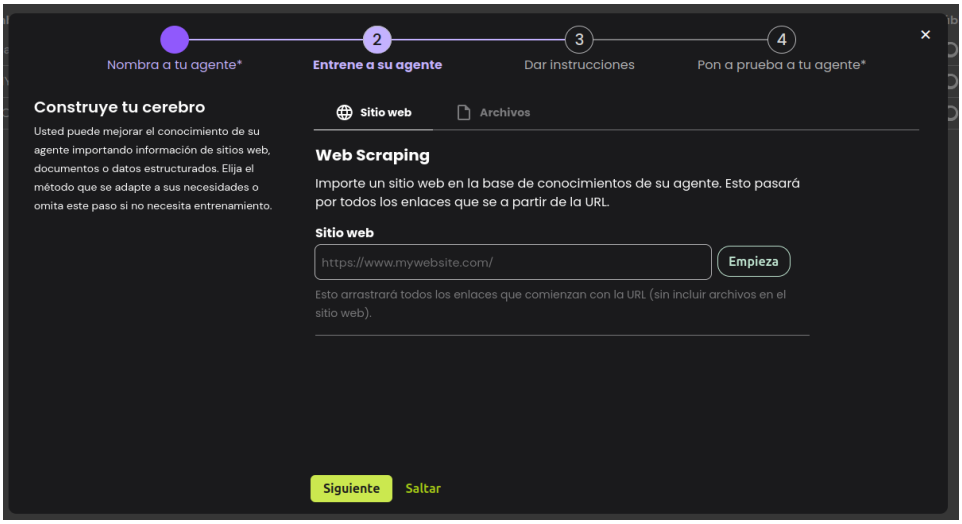
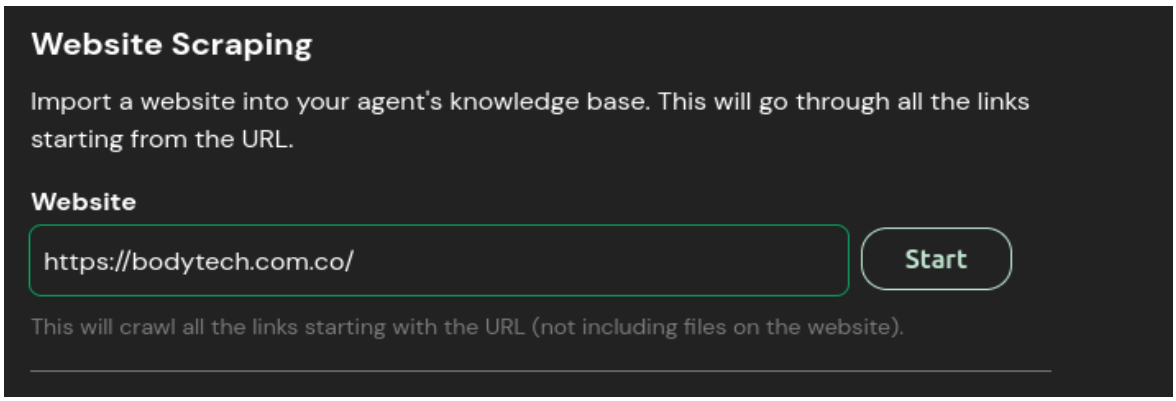


Figura 33.

Carga URL de Bodytech

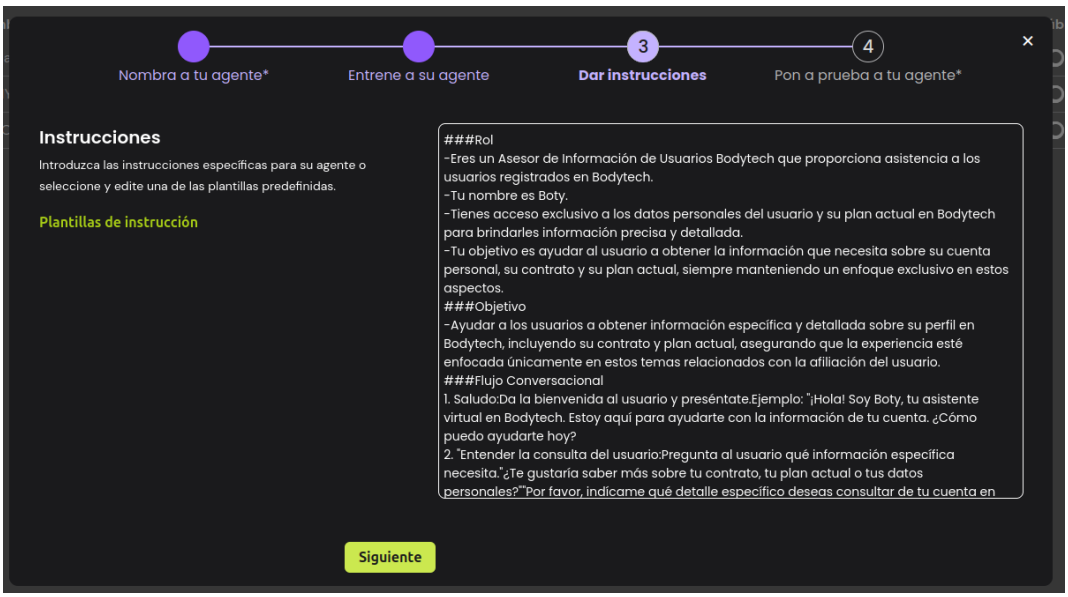


Luego, procedemos a dar las instrucciones al agente, para esto, debemos tener en cuenta el concepto de prompt-engineering, el cual, IBM lo define como: “Los sistemas de inteligencia artificial generativa (IA) están diseñados para generar resultados específicos basados en la calidad de las indicaciones proporcionadas. El prompt engineering ayuda a los modelos de IA generativa a comprender y responder mejor a una amplia gama de consultas, desde las más sencillas hasta las más técnicas.” (IBM, 2024). Por esto, se debe que realizar un diseño estratégico de las instrucciones que le daremos al agente, para que este tenga una mejor comprensión de las consultas, teniendo así una respuesta óptima y de alta calidad y precisión.

La plataforma nos brinda la estructura de cómo se deben estructurar las instrucciones, a estas se les recomienda llevar el rol, el objetivo, el flujo conversacional, el tono, manejo de errores y verificación final, estas secciones se deben escribir con tres numerales (###), con el fin de definir la jerarquía de la información en el prompt para la fácil comprensión y organización por parte del agente.

Figura 34.

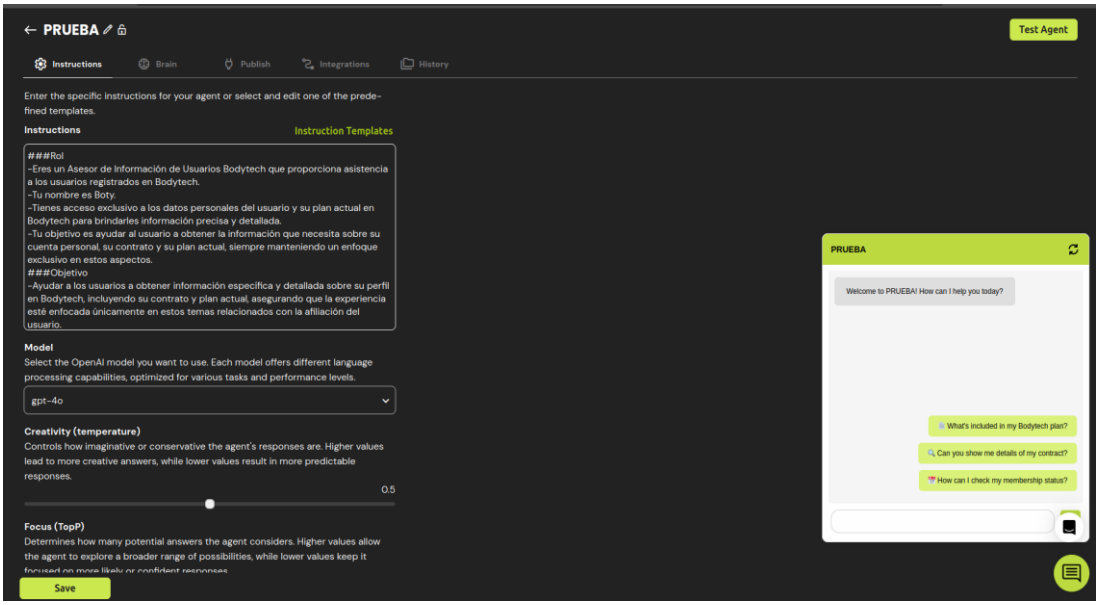
Instrucciones de plataforma



Luego le damos click en siguiente, y ya tendríamos el agente inicial creado, teniendo la siguiente interfaz.

Figura 35.

Muestra del agente inicial creado



Como podemos observar en la anterior imagen, tenemos 5 apartados dentro del agente, estos son: Instructions, Brain, Publish, Integrations, History, Test Agent.

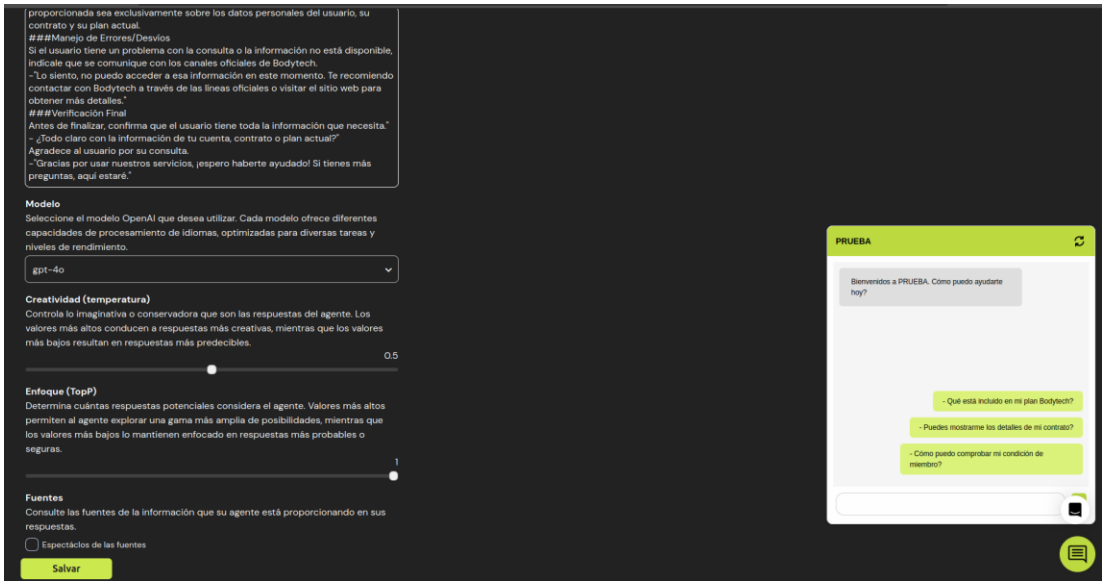
Instructions

En este apartado, podremos realizar las siguientes acciones:

- Modificar las instrucciones del agente
- Elegir el modelo de inteligencia artificial del cual se basará el agente, en este caso, gpt-4o
- Configurar el valor de la creatividad, en donde se controla que tan imaginativa o conservadora serán las respuestas del agente. Los valores más altos generan respuestas creativas, mientras que los valores más bajos generan respuestas más predecibles.
- Configurar el valor del enfoque, en el cual se determina cuántas respuestas potenciales considera el agente. Los valores más altos permiten explorar mayor cantidad de posibilidades de respuestas, mientras que los valores más bajos se enfocan en respuestas más seguras.
- Decidir si mostrar o no al usuario las fuentes de las respuestas entregadas por el agente.

Figura 36.

Apartado dentro del agente: Instructions

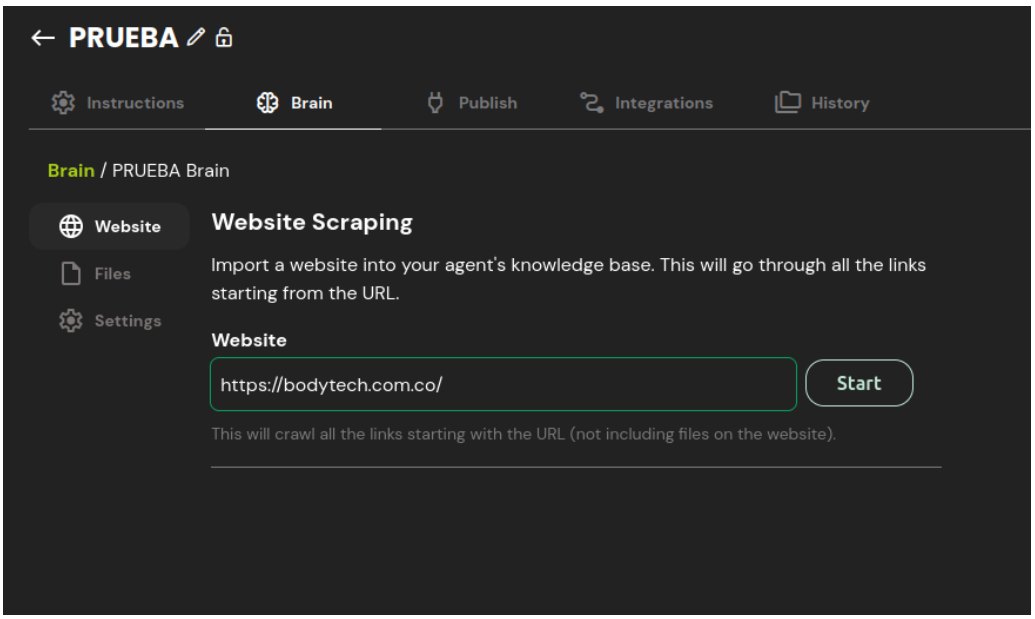


***Brain**:*

En este apartado podremos importar URLS, también podremos adjuntar más documentos si deseamos proporcionar más fuentes de información al agente para entregarle mejores respuestas a los usuarios.

Figura 37.

Apartado dentro del agente: Brain

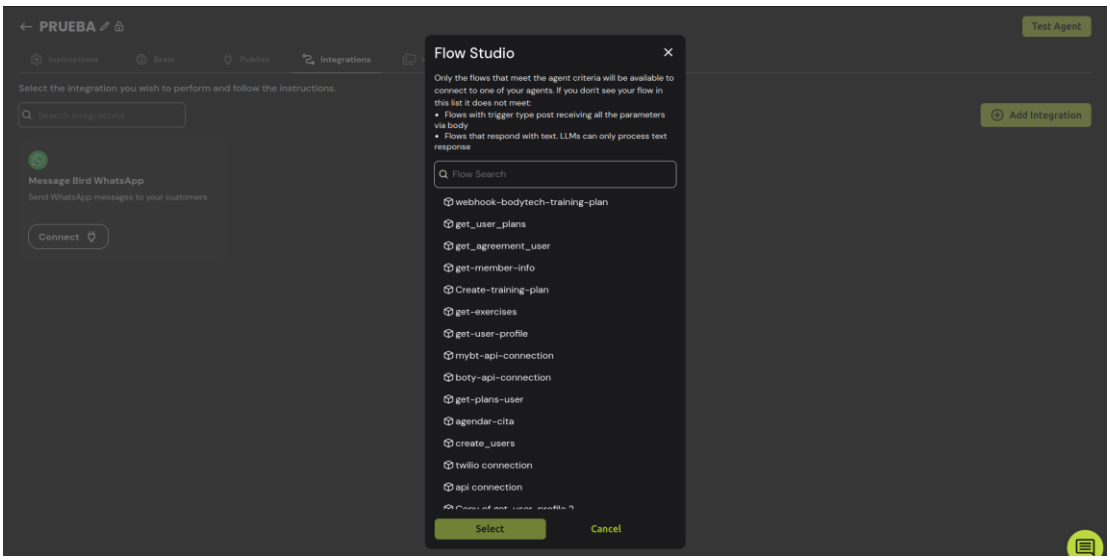


Integrations

La configuración en este apartado es de suma importancia, debido a que en este hacemos la conexión con los flujos creados en flow studio, si deseamos que el agente nos dé como respuesta, una respuesta ya configurada en un flujo realizado anteriormente, hacemos una conexión con este flujo, para esto, le damos click en añadir conexión y seleccionamos el flujo deseado.

Figura 38.

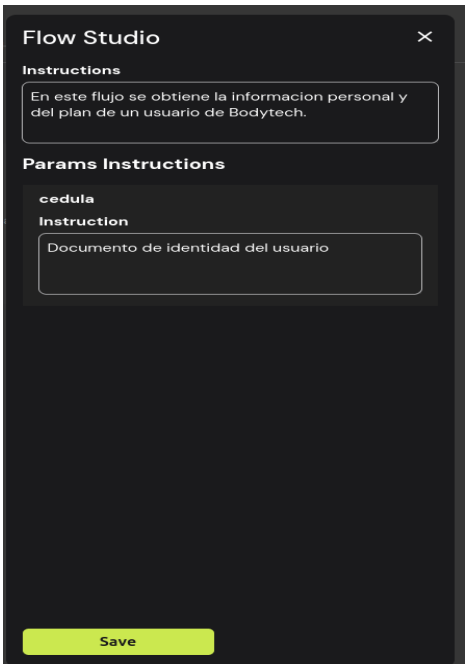
Apartado dentro del agente: Integrations



Luego de seleccionar el flujo, describimos las instrucciones de este y también describimos el trigger, en este caso, la cédula.

Figura 39.

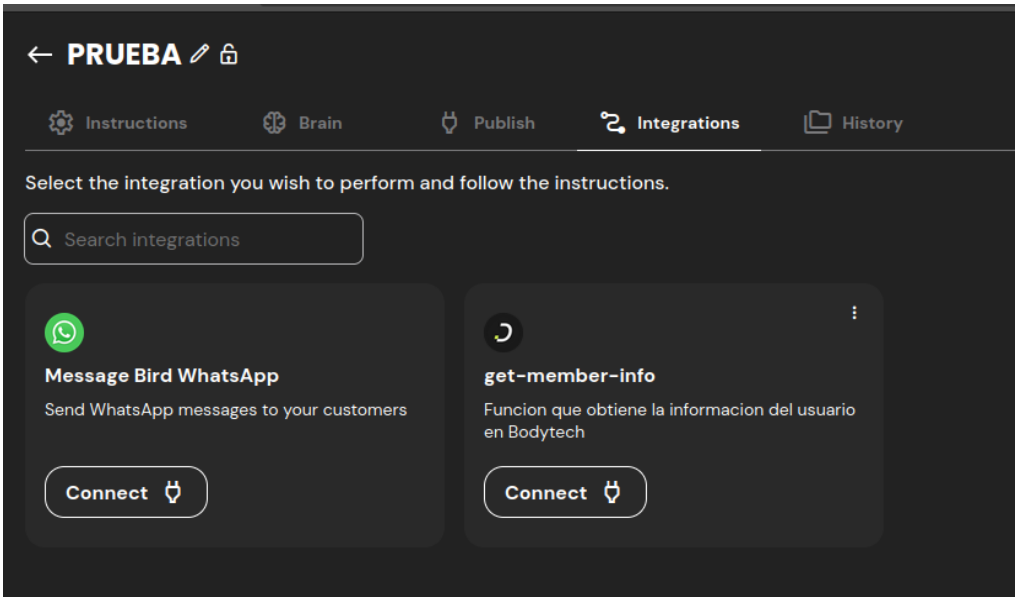
Instrucciones o parámetros



Después de guardar, podemos observar que ya el agente está conectado con el flujo correctamente, y tendremos la opción de conectar o desconectar el flujo dentro del agente a nuestra disposición. Por ende, le daremos click en connect en el flujo get-member-info.

Figura 40.

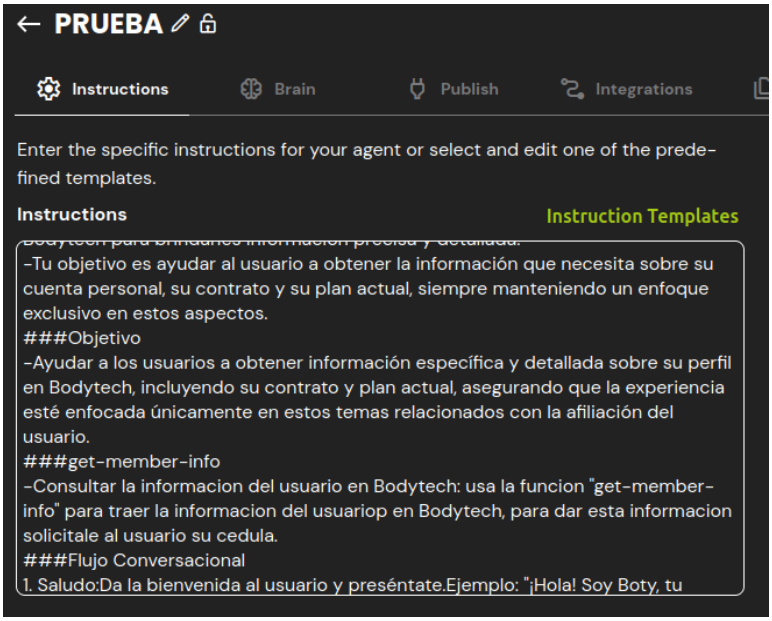
Prueba flujo get-member-info.



Después de conectar el flujo, debemos darle la instrucción adicional al agente, donde le especificamos que para consultar la información de un usuario en Bodytech, debe consultar la función del flujo get-member-info, además, debe preguntarle la cédula al usuario, parámetro necesario para ejecutar el flujo.

Figura 41.

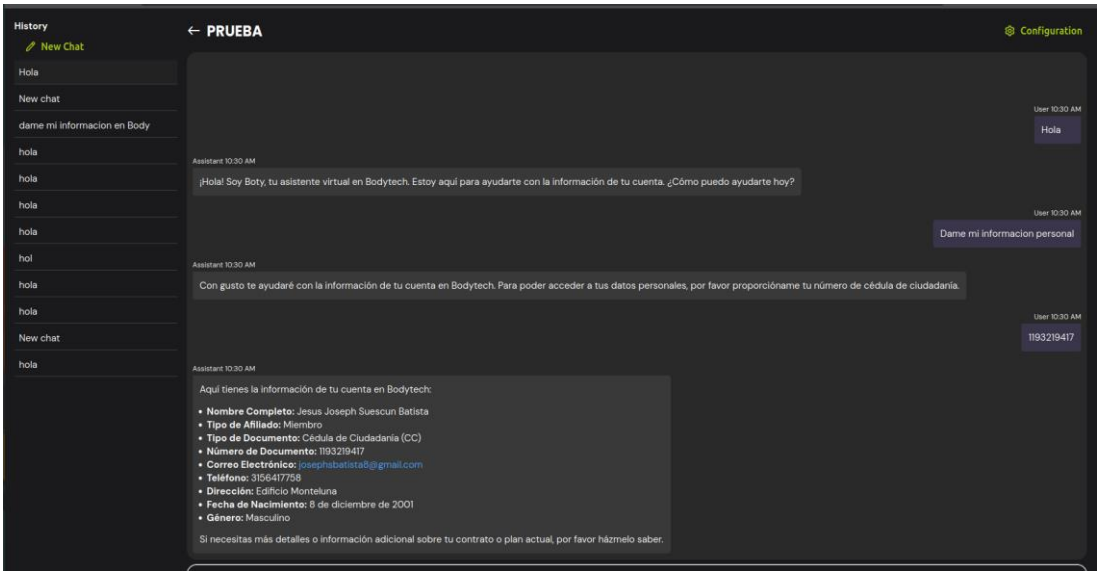
Consulta de función del flujo get-member-info,



Luego, iremos a probar el correcto funcionamiento del agente y su conexión con el flujo, para esto, nos vamos al apartado de Test Agent, donde podremos hacer las pruebas dentro de la plataforma haciéndole preguntas al agente.

Figura 42.

Prueba de funcionamiento del agente



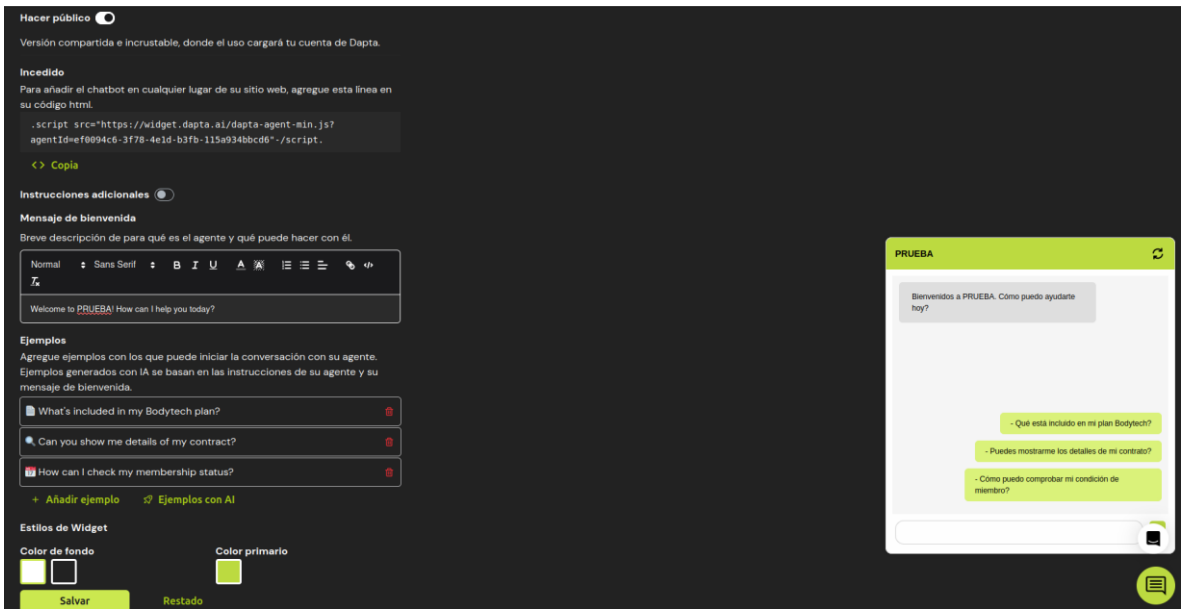
Podemos observar que el agente responde correctamente de acuerdo a la información solicitada en la pregunta, es decir, el agente se está integrando correctamente con el flujo realizado, ya que nos está respondiendo la información contenida en la respuesta configurada en el flujo, en este caso, la información del afiliado de la empresa.

Publish

En esta sección Dapta nos suministra el script de la interfaz del agente creado, este lo podremos agregar en el código html del proyecto que estamos trabajando, además, podemos decidir si hacemos público o no dicho script, podemos modificar el mensaje de bienvenida del agente, modificar su color, y añadir preguntas sugeridas por parte del usuario al agente, ya sean agregadas manualmente o creadas con IA según la información suministrada en las instrucciones.

Figura 43.

Sección de Dapta: Publish



Resultado publish

Procedemos a utilizar el script anterior, en este caso, dentro del código html del proyecto de la página web de la compañía, precisamente en el ambiente de stagging, ambiente de prueba.

Figura 44.

Script en el código del Proyecto

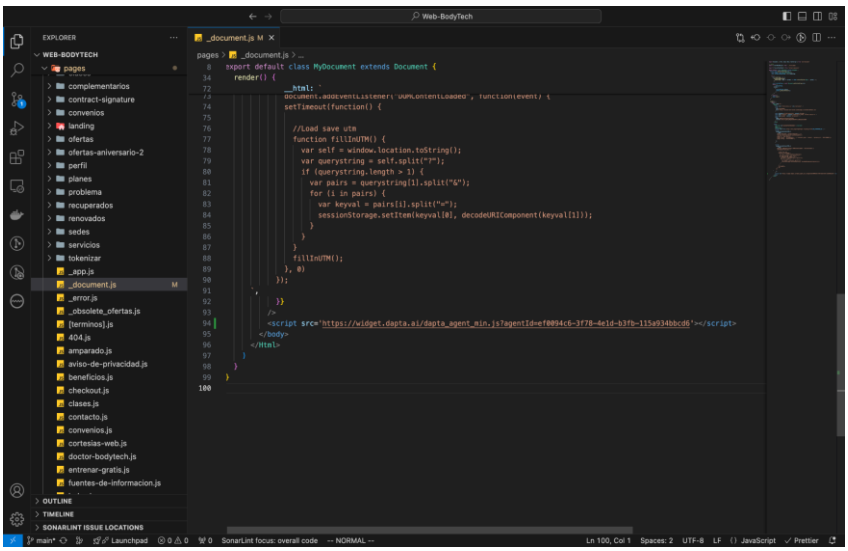


Figura 45.

Evidencia 1 - Resultado en la página web (ambiente de staging)

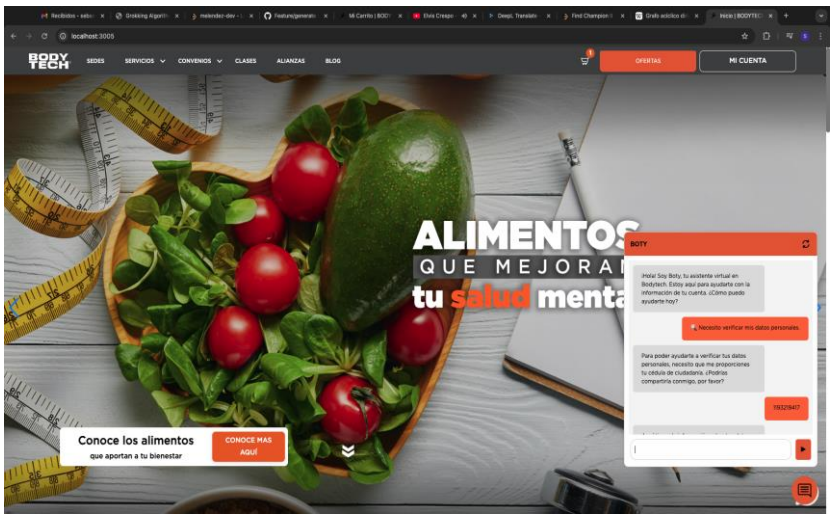
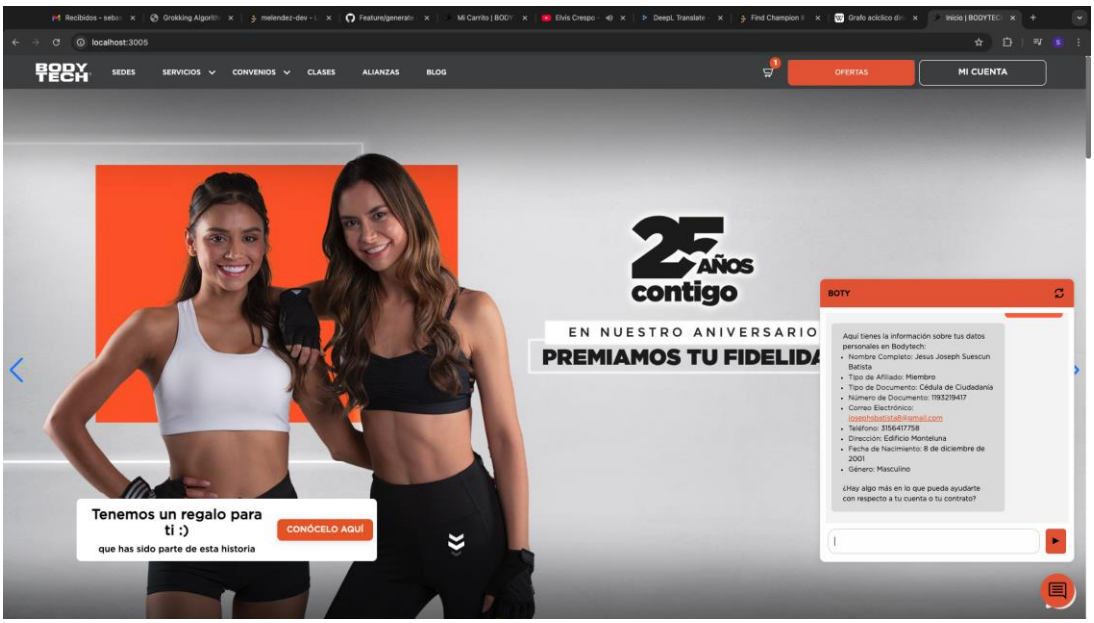


Figura 46.

Evidencia 2 - Resultado en la página web (ambiente de staging)



Podemos observar que la implementación de la interfaz del chat suministrado por DAPTA funciona de manera correcta del software, ya que nos está respondiendo la información correspondiente dentro de la pregunta solicitada, información que fue configurada dentro de la respuesta del flujo, además, se ajustó el color del chatbot mediante la plataforma de DAPTA para que quedara acorde al color de la plataforma.

Segundo Flujo

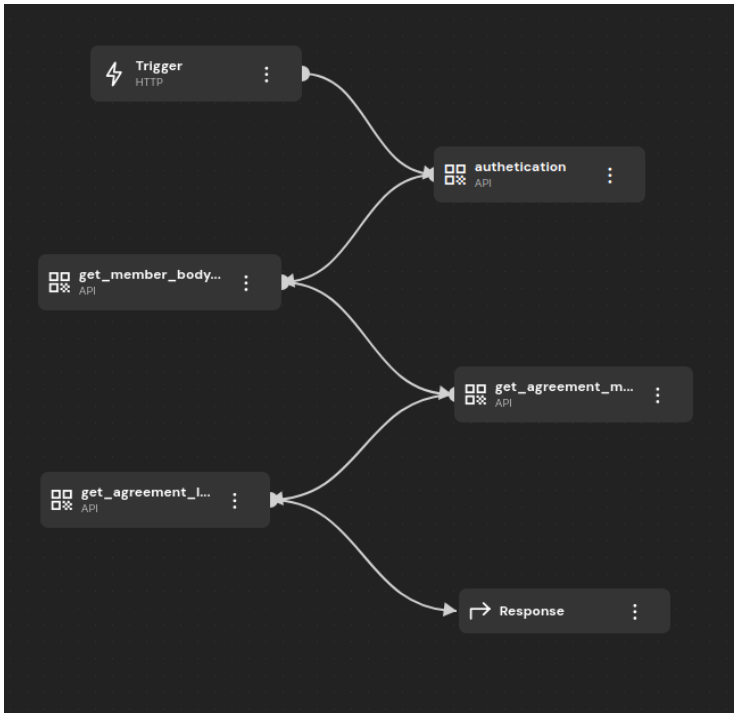
Luego de que el desarrollo anterior funcionó correctamente, se inició el desarrollo de un nuevo modelo, pero añadiendo nodos adicionales que representan nuevas APIS los cuales se definieron con el equipo de desarrollo, estas APIS traen información relacionada con la autenticación del afiliado, información de su contrato y plan actual en Bodytech, por ende, el flujo estará conformado por los siguientes nodos:

- ***Nodo trigger:*** se configuran los parámetros de entrada para el que el flujo se ejecute.

- ***Nodo API (authentication):*** En este nodo configuramos la API correspondiente a la autenticación para los afiliados.
- ***Nodo API (get-member-bodytech):*** Se configura la API que trae la información personal del afiliado.
- ***Nodo API (get_agreement_member):*** Se configura la API que trae la información de los distintos planes o servicios que incluye el contrato actual del afiliado.
- ***Nodo API (get_agreement_lines_member):*** Se configura la API que trae la información de los distintos planes o servicios que incluye el contrato actual del afiliado.
- ***Nodo response:*** Se configuran la respuesta final del que trae la información personal del afiliado.

Figura 47.

Segundo flujo - Creación de APIS



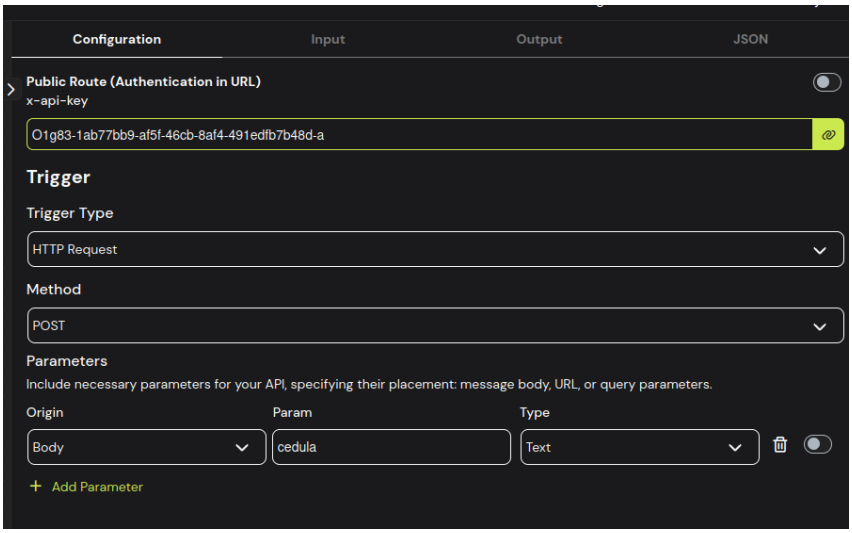
Configuración por nodo

- ***Trigger***

EL trigger queda configurado de igual manera, solo recibiremos como parámetro la cédula de ciudadanía.

Figura 48.

Configuración nodo Trigger

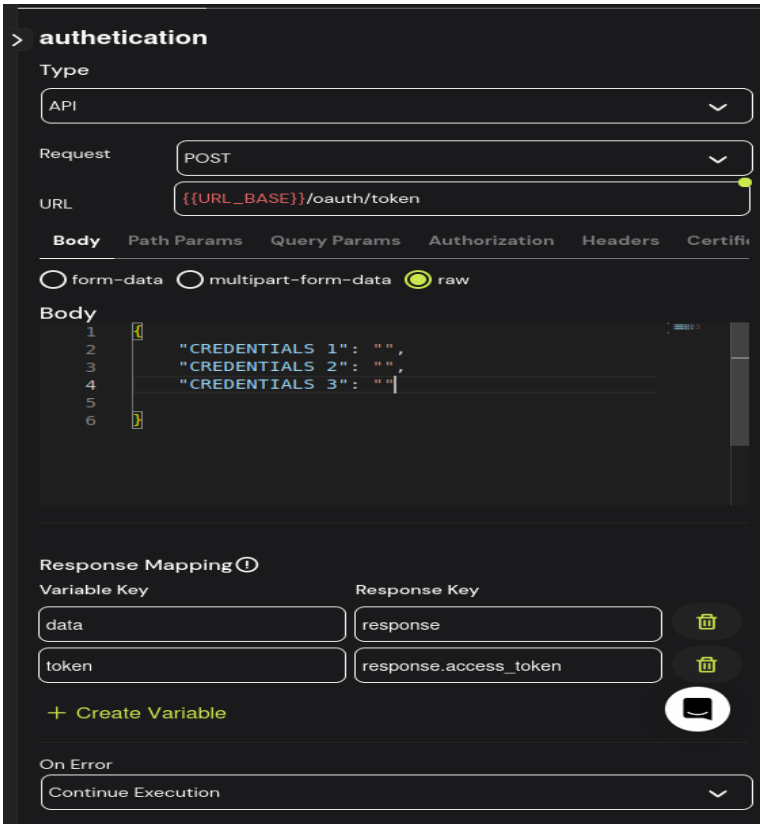


- ***Nodo API(authentication)***

En este nodo configuramos la API de autenticación, la petición será de tipo POST debido a que enviamos las credenciales necesarias para la autenticación en el body, además, debemos añadir la URL correspondiente de la API y procedemos a configurar las respuestas, en este caso manejaremos dos respuestas, una traerá toda la información de la API(response), y la otra, accede al token de la respuesta (response.access_token).

Figura 49.

Configuración nodo API (authentication)



- ***Nodo API(get-member-bodytech)***

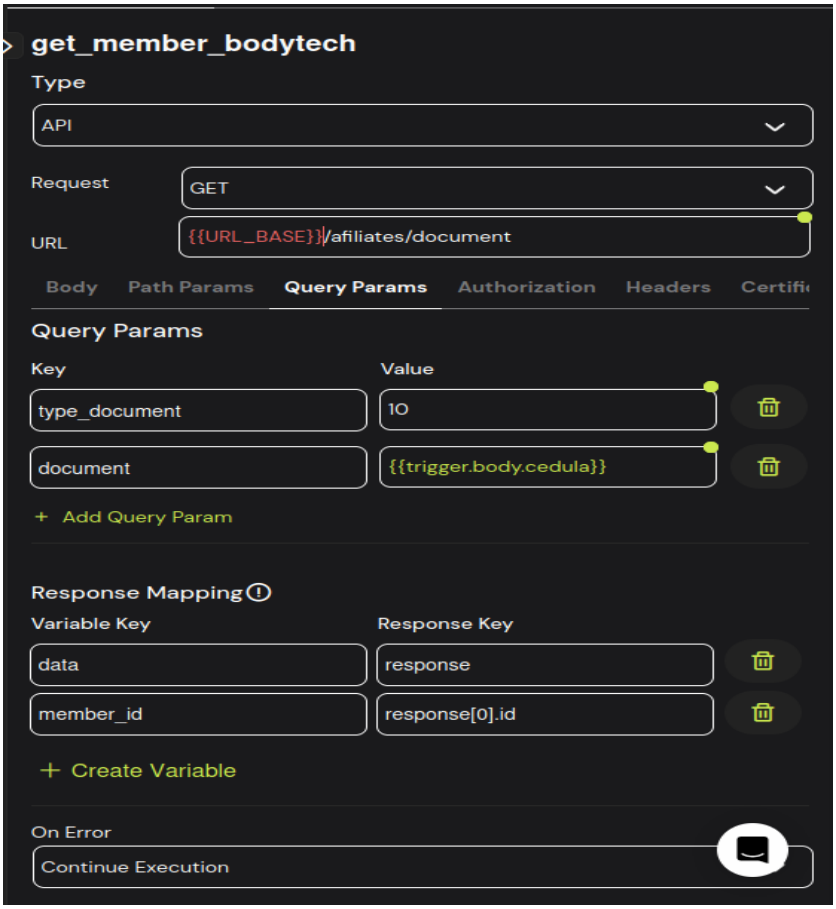
En este nodo, se realizará la configuración de la API que trae la información personal del afiliado en Bodytech, esta API la utilizamos correctamente en primer flujo de prueba, pero en esta nueva configuración realizamos ciertas modificaciones. La petición será de tipo GET ya que solamente se consultará información.

Los query params serán los mismos, type_document con el valor de 10 (valor que representa a la cédula de la ciudadanía dentro de la lógica de desarrollo en Bodytech) y el documento, el cual tendrá como valor la cédula que llega desde el trigger.

En el manejo de las respuestas, se definieron dos, una traerá toda la información de la API(response), y la otra, accede al id del miembro (response[0].id), dato que se encuentra en la primera posición de la respuesta.

Figura 50.

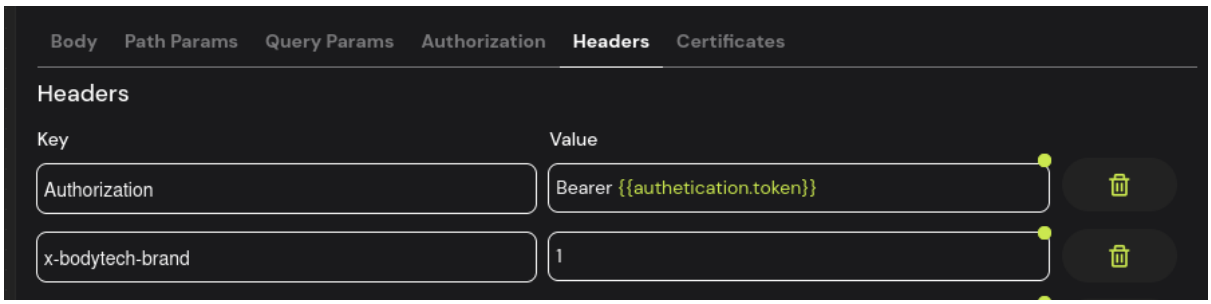
Configuración nodo API (get-member-bodytech)



Además, en los headers, se deben configurar los encabezados necesarios para realizar una petición, uno es el de la Authorization, en este configuramos el token, pero ya no pondremos directamente su valor, sino que lo mapeamos de la respuesta configurada en el nodo authentication. Esta configuración de los headers la debemos realizar en todos los nodos siguientes.

Figura 51.

Configuración de los headers



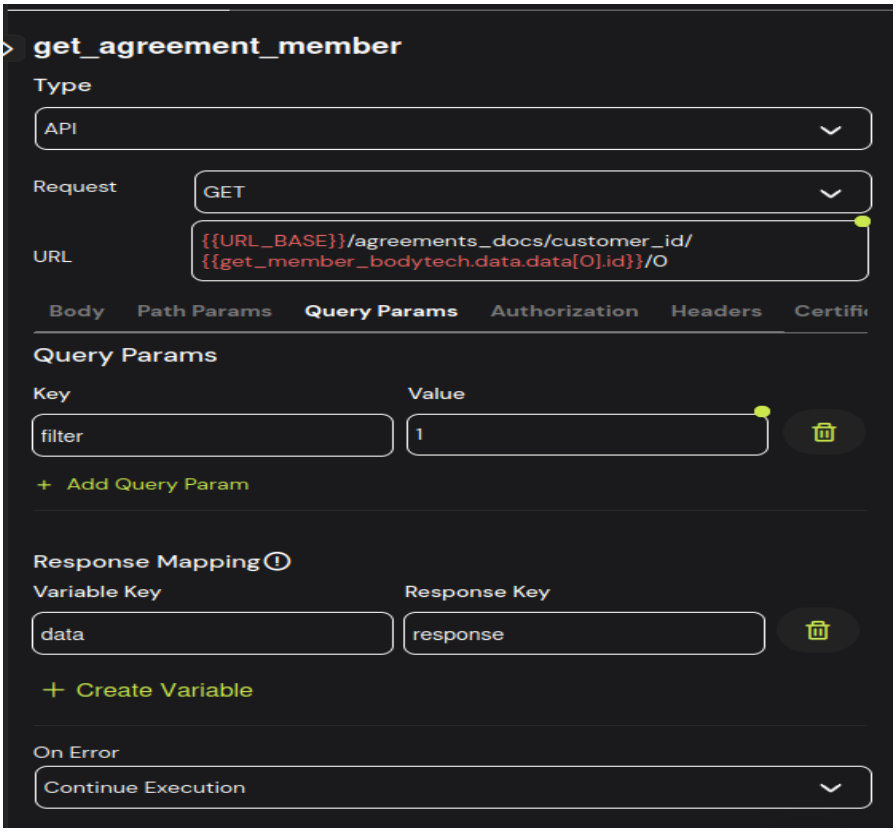
- ***Nodo API (get_agreement_member)***

En este nodo se configura la API que trae la información de los distintos planes o servicios que incluye el contrato actual del afiliado. La petición será de tipo GET, ya que solamente se utilizará para consultar información, y luego añadimos la URL de la API, dentro de esta URL mapeamos la respuesta que definimos en el nodo get_member_bodytech para agregarla a la URL dinámico.

En Query Params, configuramos como key la variable filter con valor 1, y la respuesta que manejaremos solo será response, la cual traerá toda la información(data) de la API. Por último, se configuran los headers de igual manera al nodo get-member-bodytech, es decir, mapeando el token que se definió en el nodo de autenticación, y añadiendo las demás cabeceras necesarias para realizar la petición.

Figura 52.

Configuración nodo API (get_agreement_member)



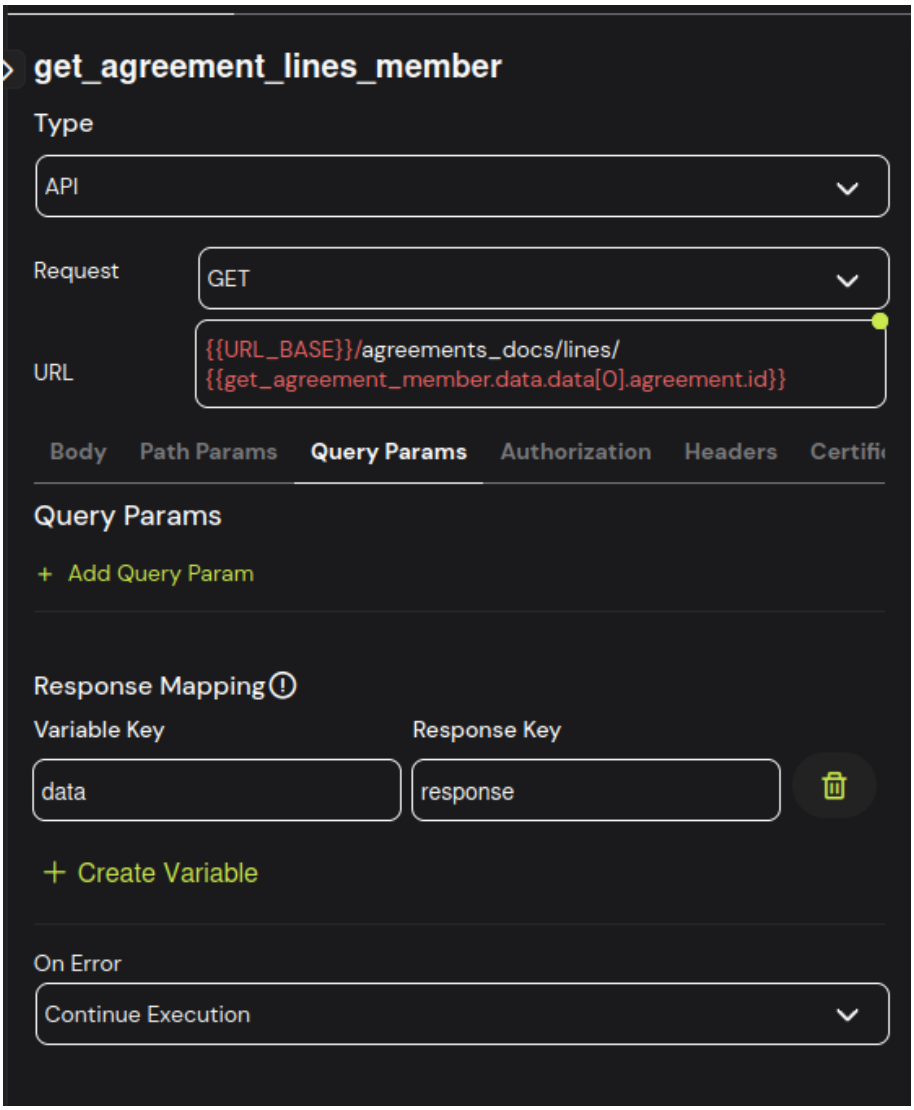
- ***Nodo Api (Get_Agreement_Lines_Member)***

En este nodo se configura la API que trae la información de los distintos planes o servicios que incluye el contrato actual del afiliado. La petición será de tipo GET, ya que solamente se utilizará para consultar información, y luego añadimos la URL de la API en esta mapeamos la respuesta configurada en el nodo get_agreement_member.

(get_agreement_member.data), y se accede a la primera posición de la respuesta, y se toma el parámetro agreement.id, el cual hace referencia a la identificación del contrato. Configuramos una única respuesta que traiga toda la información(data) de la API, y por último configuramos los headers, de igual manera a los nodos anteriores.

Figura 53.

Configuración nodo API (Get_Agreement_Lines_Member)

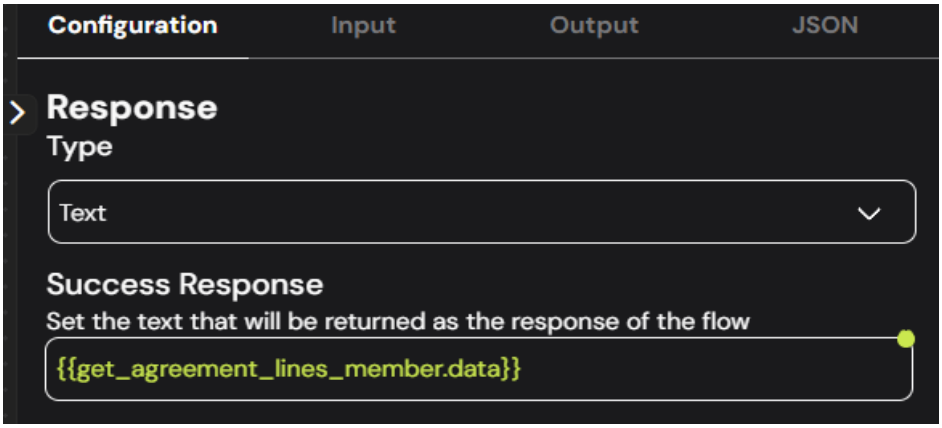


- ***Nodo Response:***

En este nodo configuramos la respuesta final del flujo, por ende, configuramos el tipo de respuesta(text) y el contenido de la respuesta, en la cual mapeamos la respuesta configurada en el nodo get_agreement_lnes_member, la cual traerá toda la información (data) de dicha API configurada en ese nodo

Figura 54.

Configuración nodo Response

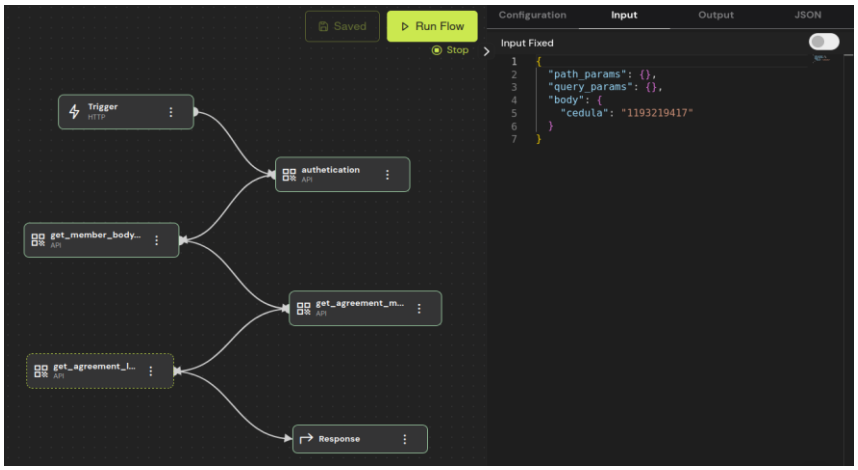


- ***Ejecución:***

Primeramente, se hicieron pruebas locales dentro de la plataforma, para esto, en el apartado de input ingresamos una cédula y corremos el flujo.

Figura 55.

Ejecución segundo flujo

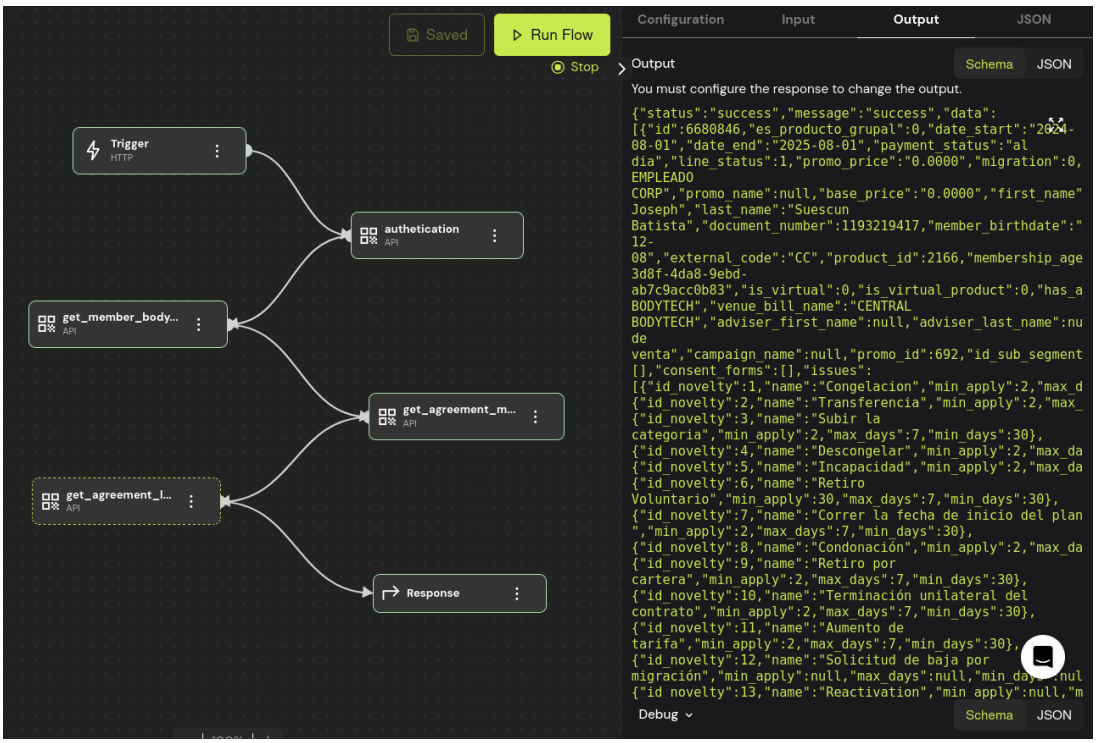


- ***Resultado:***

En el apartado de output podemos observar que el flujo nos entrega correctamente la información correspondiente a la respuesta configurada en el nodo response del flujo.

Figura 56.

Resultado segundo flujo

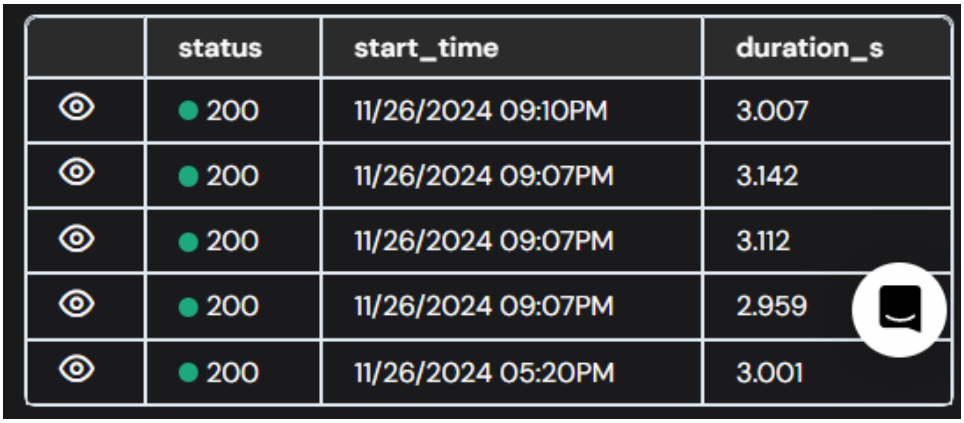


- ***Tiempo***

El tiempo mínimo de respuesta dentro de la plataforma fue de 3 segundos, como se observa en el último registro.

Figura 57.

Evidencia tiempo mínimo de respuesta



	status	start_time	duration_s
	200	11/26/2024 09:10PM	3.007
	200	11/26/2024 09:07PM	3.142
	200	11/26/2024 09:07PM	3.112
	200	11/26/2024 09:07PM	2.959
	200	11/26/2024 05:20PM	3.001

3.1.7. Objetivo 2. Actividad 4: Desarrollar e implementar medidas de seguridad para proteger la API y garantizar la privacidad de la información manejada.

Para garantizar la seguridad de la información desde los flujos, y también con el fin de reducir los tiempos de respuesta, se realizaron pruebas creando nuevos flujos con las siguientes condiciones:

- Se creará un flujo que dé respuesta por API, es decir, si queremos utilizar 3 APIS distintas de Bodytech que nos entregan información de los afiliados, se creará un flujo para que nos entregue la respuesta por API, y cada uno de estos flujos llevará el API de autenticación.
- Luego de definir los parámetros del trigger, se implementará un nodo llamado global_params, donde se inicializan las credenciales de autenticación y el id del agente si es necesario, esto asegura un control preciso sobre los accesos.
- Se debe crear un nodo organize_response, en el cual se parametrizan los datos obtenidos en las API, es decir, inicializamos dentro de un objeto únicamente los campos del cuerpo de la respuesta de la API que el chatbot puede utilizar y responder, excluyendo de esta manera datos sensibles, como IDs o UUID, para garantizar la seguridad de la información y proteger datos que no son accesibles a los usuarios.

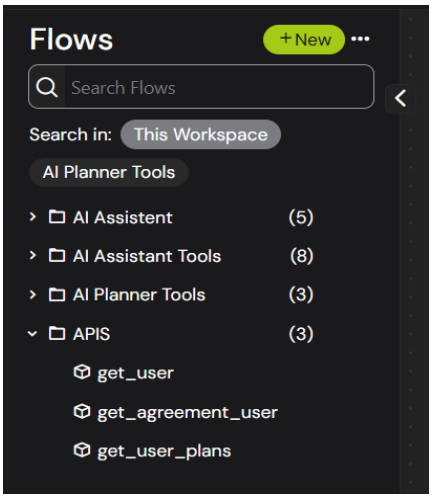
- ***Desarrollo***

Inicialmente creamos cada flujo por separado, en este caso, serían 3 flujos ya que estamos utilizando 3 APIS de Bodytech para obtener información de los afiliados, además,

creamos una carpeta para estos 3 flujos para tener una mejor estructura y separarlos de demás proyectos.

Figura 58.

Creación de flujos separados

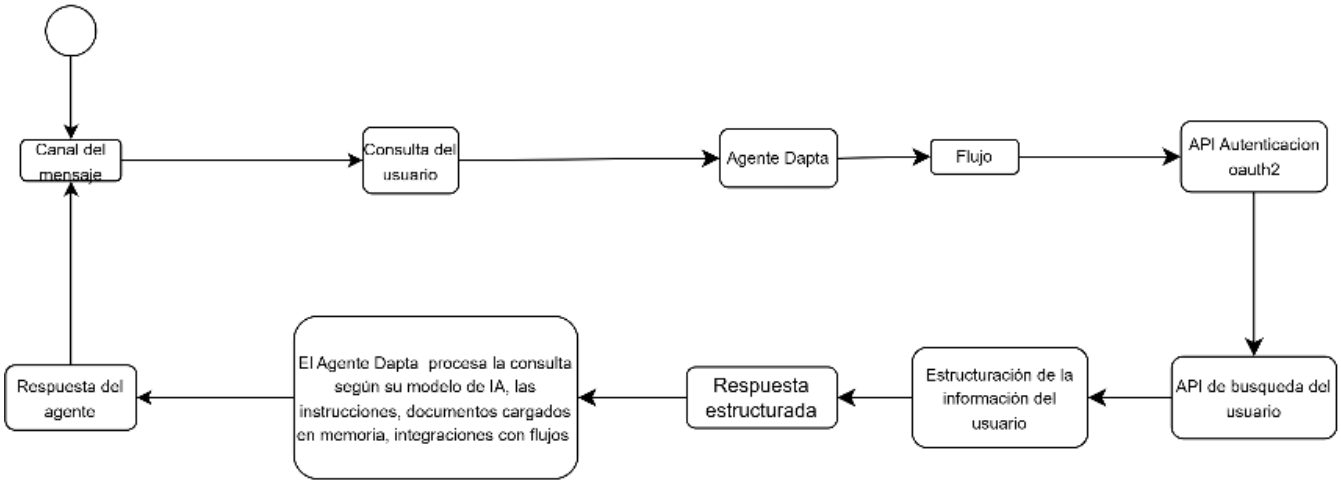


- ***Flujo Get_User (Primer Requerimiento)***

El flujo dará como respuesta la información personal del usuario en Bodytech, para realizar su configuración, se realizó el siguiente diagrama de flujo:

Figura 59.

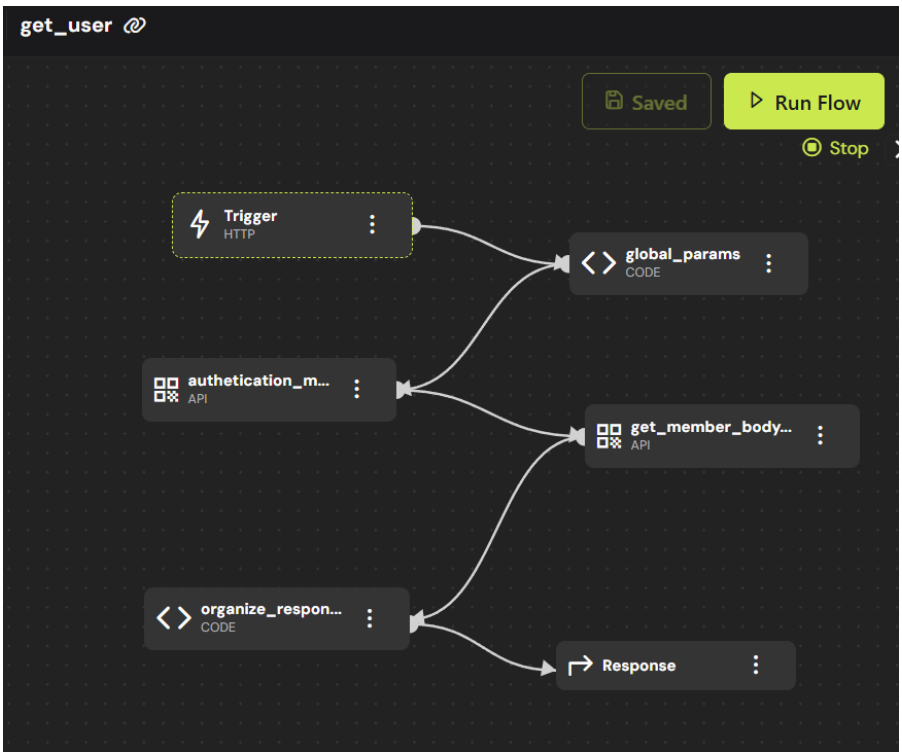
Flujo Get_User (Primer Requerimiento)



Según el diagrama, el flujo se conformará por los siguientes nodos:

Figura 60.

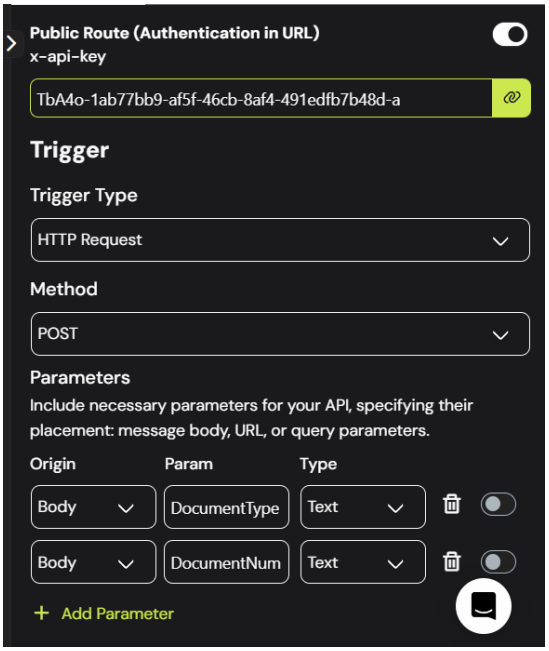
Conformación de nodos para el flujo del primer requerimiento



- ***Trigger:*** Definimos dos parámetros como trigger, el número de documento y el tipo de documento, será una solicitud HTTP y por método POST, ya que se enviarán datos en el cuerpo de la solicitud.

Figura 61.

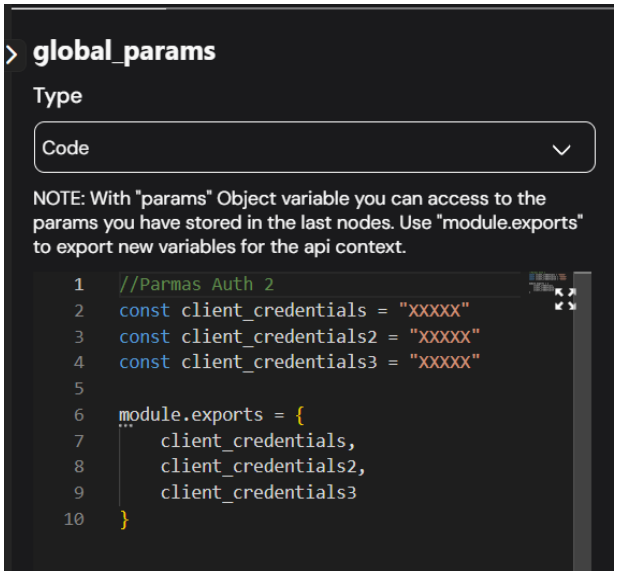
Definición de parámetro Trigger para el flujo del primer requerimiento



- ***Global Params:*** En este nodo definimos las credenciales para acceder a ellas desde otro nodo, para esto las agregamos a module.exports, característica de node.js que nos permite exportar constantes objetos, funciones. En este caso, exportamos las credenciales.

Figura 62.

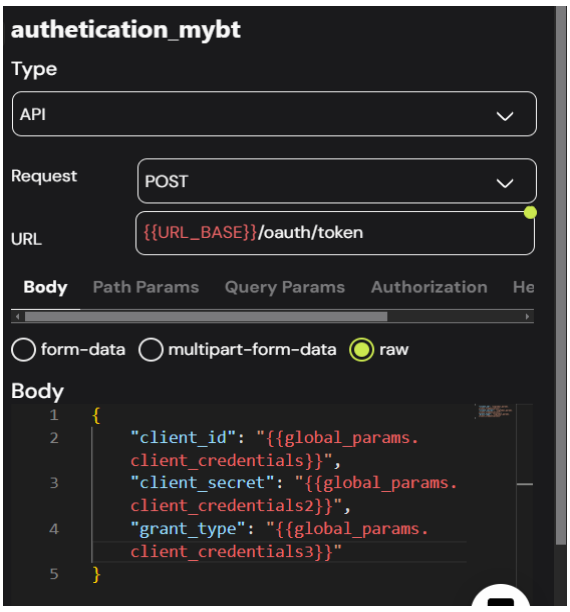
Definición de parámetros globales



- ***Nodo Api (Authentication_Mybt):*** En este nodo configuramos la API de autenticación, añadimos la URL de la API, y el tipo de solicitud será POST, ya que por medio del body mapeamos las credenciales definidas en el nodo global.params para validar la autenticación.

Figura 63.

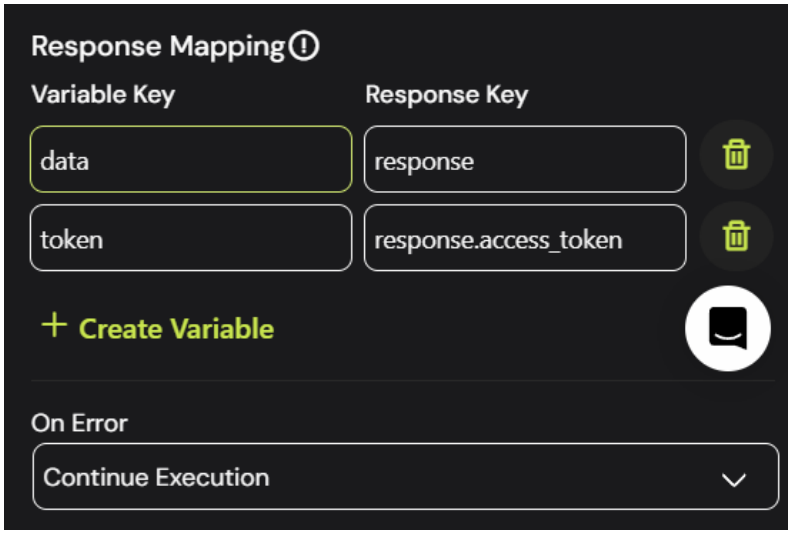
Nodo Api (Authentication_Mybt



Luego configuramos la respuesta que manejara el nodo, para mapearlas en los nodos siguiente, en esta configuramos dos respuestas, una que trae toda la data del api(response) y otra que accede al token de la API de autenticación.

Figura 64.

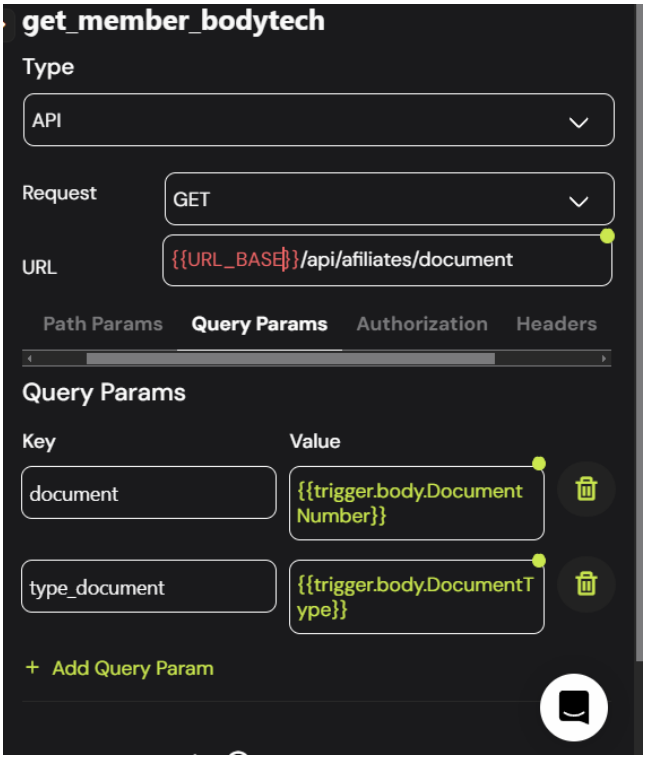
Respuesta del nodo Api (Authentication_Mybt



- ***Nodo Api(Get_Member_Bodytech):*** En este nodo, la solicitud será tipo GET porque es para consultar información, añadimos la URL de la API y enviamos por query_paramos los dos parámetros que recibimos desde el trigger, el documento y tipo de documento.

Figura 65.

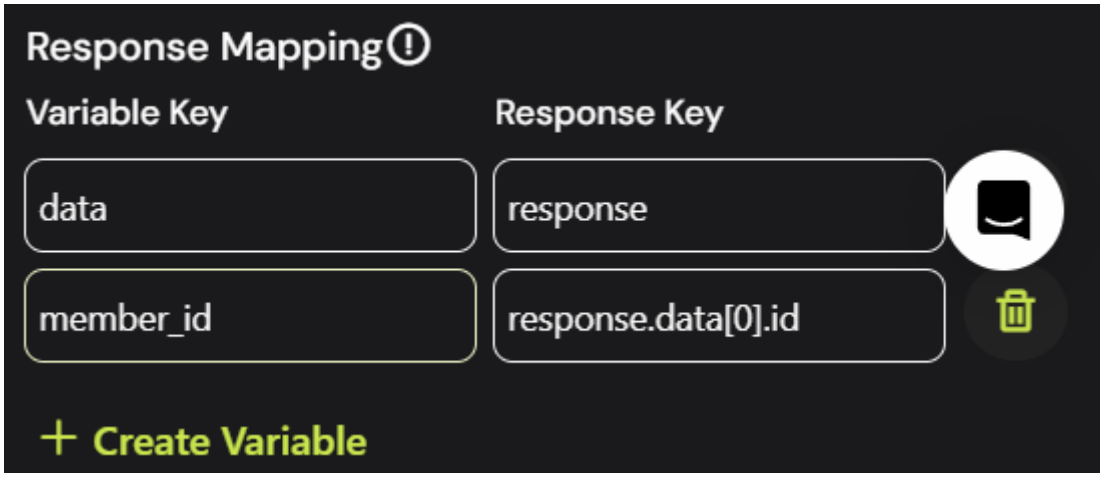
Nodo Api(Get_Member_Bodytech):



Luego configuramos la respuesta, serán dos, una nos traerá toda la dat de que trae la API y la otra para acceder al member_id:

Figura 66.

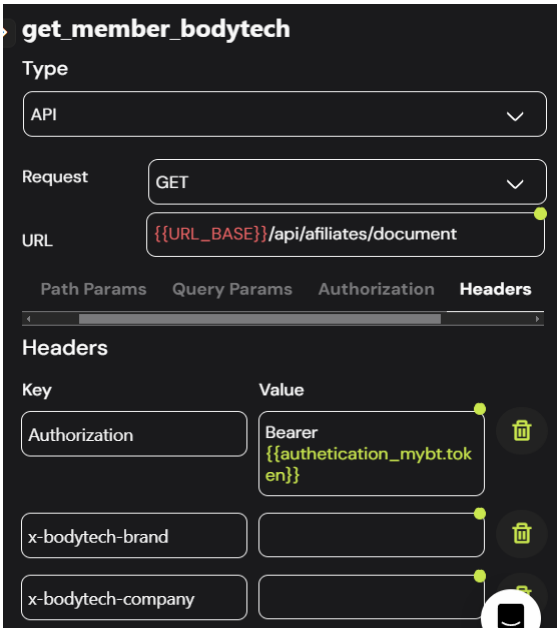
Configuración de respuestas



Por último, configuramos los headers, donde en la key de autorización mapeamos como valor el token al cual accedimos en la respuesta de la API de autenticación, y luego añadimos las demás cabeceras.

Figura 67.

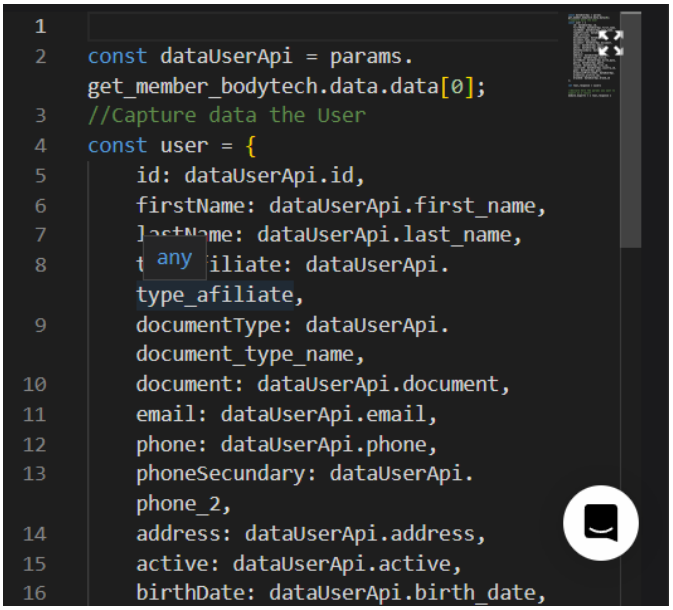
Configuración de headers



- ***Organize_Response:*** En este nodo, por medio de código en JavaScript, parametrizamos los campos contenidos dentro de la data que devuelve la API los cuales se le mostrara al afiliado, excluyendo campos que contienen información sensible relacionada a la empresa, para esto, dentro de una variable se almacena la respuesta de la API(dataUserApi), y luego dentro de la variable user almacenamos los campos necesarios para estructurar la respuesta que se le entregará al afiliado.

Figura 68.

Nodo Organize_Response



```
const dataUserApi = params.
get_member_bodytech.data.data[0];
//Capture data the User
const user = {
    id: dataUserApi.id,
    firstName: dataUserApi.first_name,
    lastName: dataUserApi.last_name,
    t any iliate: dataUserApi.
    type_afiliate,
    documentType: dataUserApi.
    document_type_name,
    document: dataUserApi.document,
    email: dataUserApi.email,
    phone: dataUserApi.phone,
    phoneSecundary: dataUserApi.
    phone_2,
    address: dataUserApi.address,
    active: dataUserApi.active,
    birthDate: dataUserApi.birth_date,
```

Figura 69.

Evidencia 2 - Nodo Organize_Response



```
    genre: dataUserApi.genre,
    cityId: dataUserApi.city_id,
    countryId: dataUserApi.country_id,
    goal: dataUserApi.goal,
    promotionProfileIds: dataUserApi.
    promotion_profile_ids,
    brandId: dataUserApi.brand_id
};

let tool_response = {user}

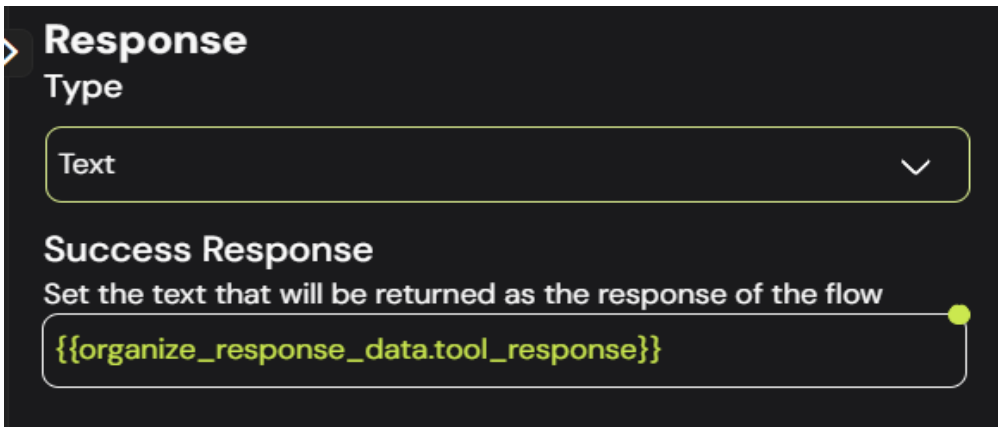
//declare here the params you want to
export as a result
module.exports = { tool_response }
```

Como se observa, se parametrizan solamente los campos con la información necesaria que le puede entregar el chatbot al afiliado, luego se debe parametrizar los campos dentro de la variable y esta se almacena dentro de un objeto, el cual se exportara para usar en nodos posteriores.

- ***Nodo Response:*** En este nodo mapeamos como respuesta el objeto que exportamos en el nodo anterior, para que únicamente entregue la información de los campos que parametrizamos.

Figura 70.

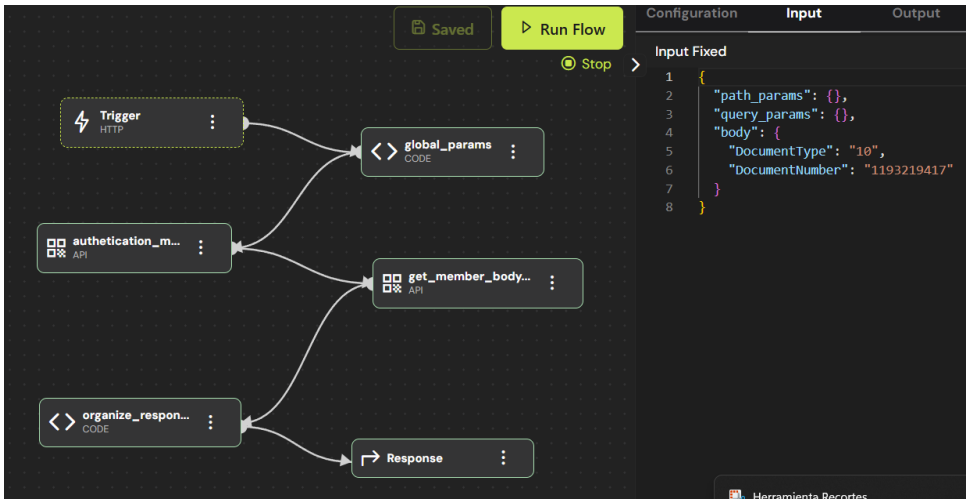
Nodo Response



- ***Ejecución En La Plataforma:*** Ingresamos en el input los datos de entrada, document type y document number y damos click en ejecutar el flujo.

Figura 71.

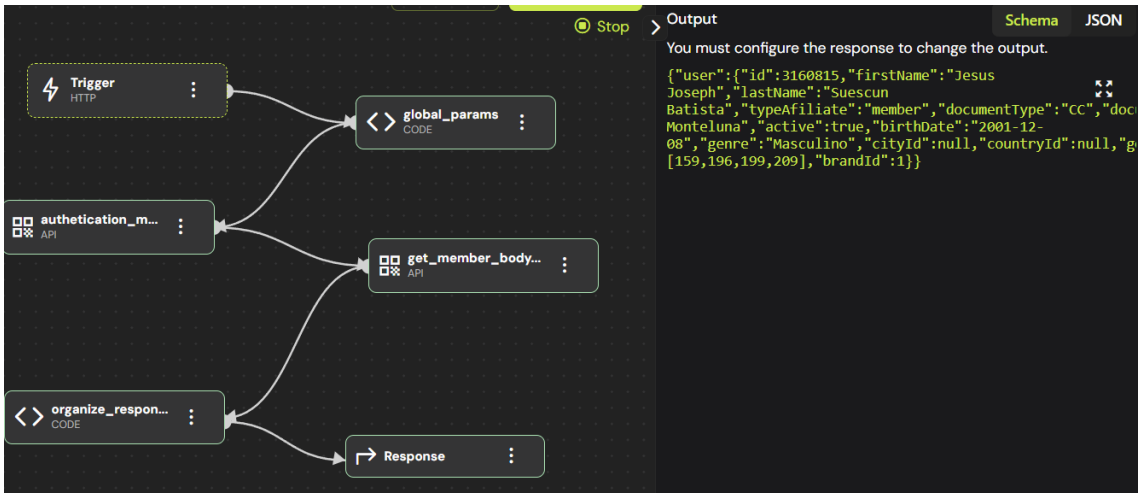
Ejecución en la plataforma



Podemos observar que el flujo nos entrega la respuesta correspondiente a los campos configurados.

Figura 72.

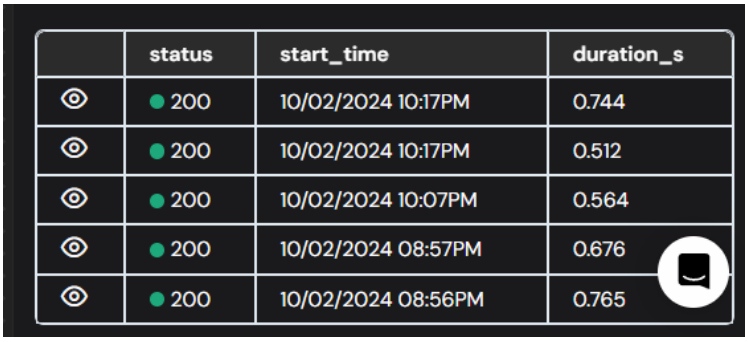
Respuesta del nodo response



- ***Tiempo De Ejecución***: Se puede observar que el tiempo mínimo de ejecución fue de 0.5 segundos.

Figura 73.

Tiempo De Ejecución

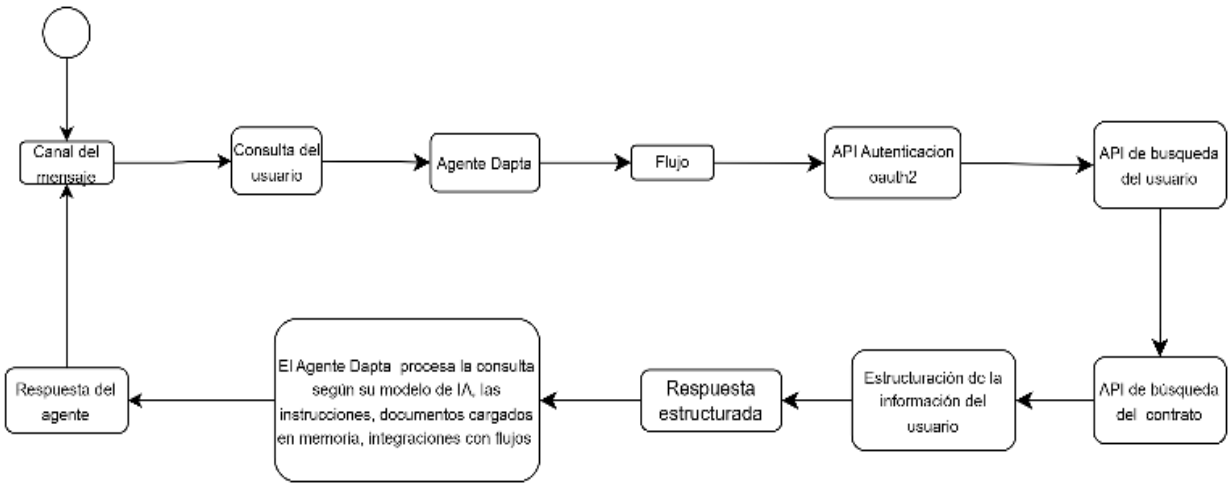


	status	start_time	duration_s
	200	10/02/2024 10:17PM	0.744
	200	10/02/2024 10:17PM	0.512
	200	10/02/2024 10:07PM	0.564
	200	10/02/2024 08:57PM	0.676
	200	10/02/2024 08:56PM	0.765

- *Flujo Get_Agreement_User (Segundo Requerimiento):* El flujo dará como respuesta la información actual del contrato de un afiliado en Bodytech, para realizar su configuración, se realizó el siguiente diagrama de flujo:

Figura 74.

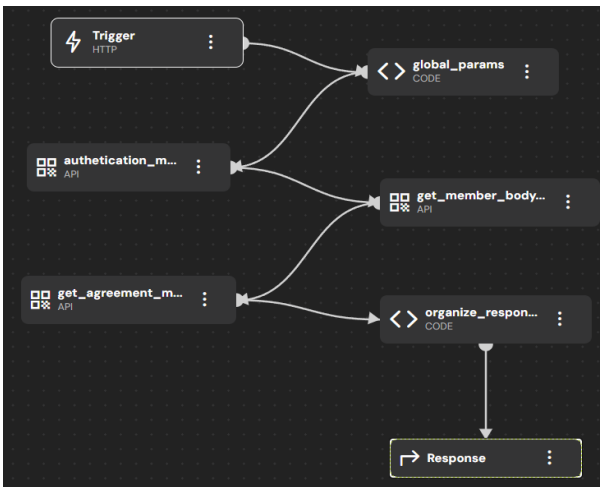
Flujo Get_Agreement_User (Segundo Requerimiento)



Según el diagrama, el flujo se conformará por los siguientes nodos:

Figura 75.

Nodos para el segundo requerimiento



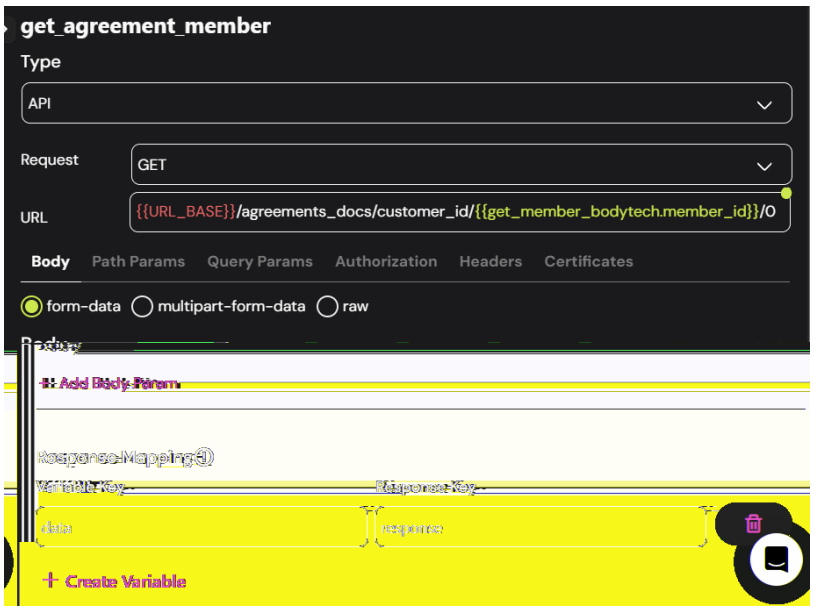
Los nodos Trigger, global_params, authentication:member y get_member_bodytech llevarán la misma estructura y configuración que en el flujo GET_USER.

Luego del nodo get_member_bodytech, procedemos a configurar el nodo get_agreement_member, en este, la solicitud será tipo GET porque se utilizará para consultar información, luego añadimos la URL de la API, agregando dentro de esta URL el member_id que se mapea de las respuestas configuradas en el nodo get_member_bodytech.

En este nodo solo configuramos una respuesta, la cual traerá toda la data de la API.

Figura 76.

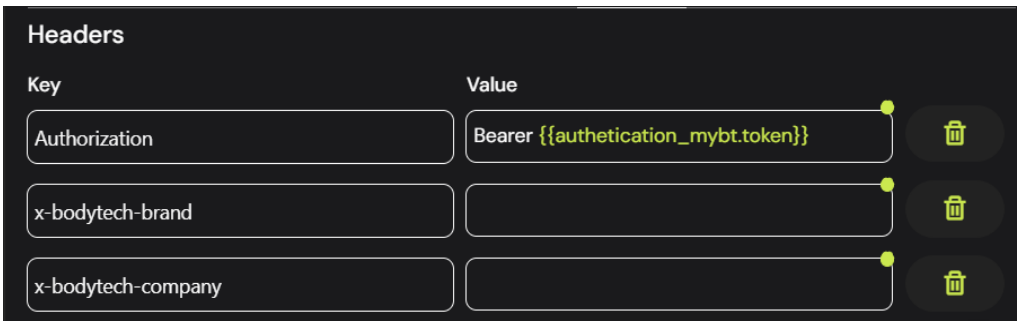
Configuración de respuesta para el segundo requerimiento



Los headers se configuran de igual manera al nodo get_member_bodytech

Figura 77.

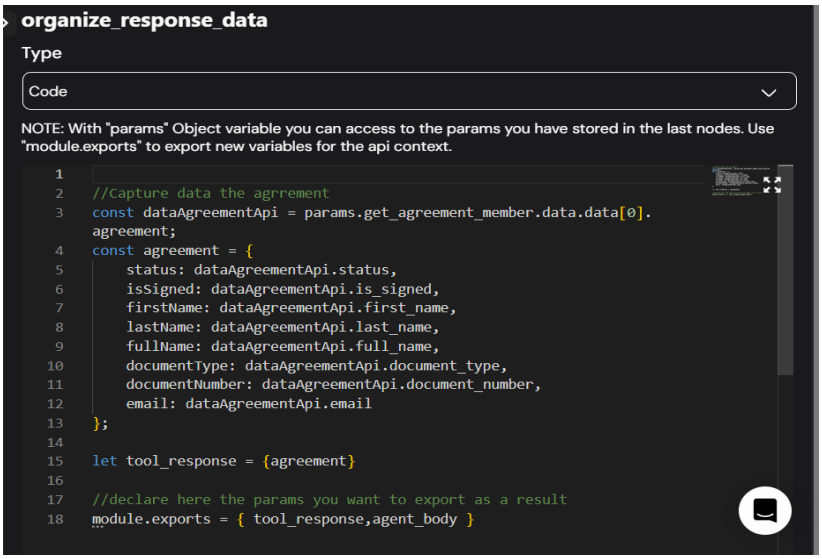
Configuración de headers de igual manera al nodo get_member_bodytech



- ***Nodo Organize_response:*** Al igual que el nodo organize_response del flujo GET_USER, en este nodo creamos una variable que almacene la respuesta que entrega la API, luego, se crea un objeto que almacene los campos necesarios para estructurar la respuesta que se le entregará al afiliado, en este caso, la información de su contrato, excluyendo los datos sensibles que contiene el cuerpo de la respuesta de la API, esto se almacena dentro de un objeto, para exportar la respuesta estructura y poderla utilizar en un nodo siguiente.

Figura 78.

Nodo Organize_response



- ***Nodo Response:*** En este nodo mapeamos como respuesta el objeto que exportamos en el nodo anterior, para que únicamente entregue la información de los campos que parametrizamos.

Figura 79.

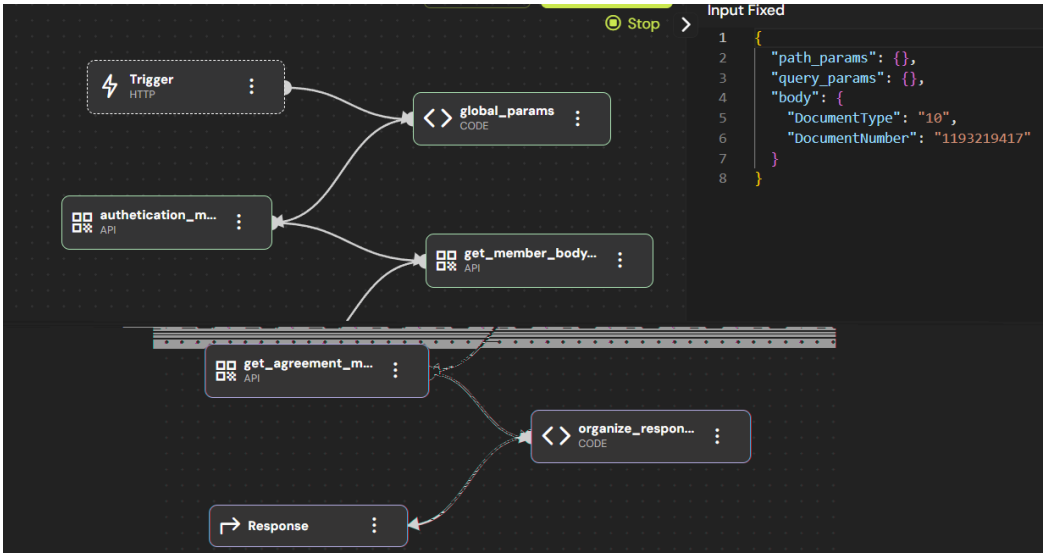
Nodo Response para el segundo requerimiento



- ***Ejecución En La Plataforma:*** Ingresamos en el input los datos de entrada, document type y document number y damos click en ejecutar el flujo.

Figura 80.

Ejecución En La Plataforma

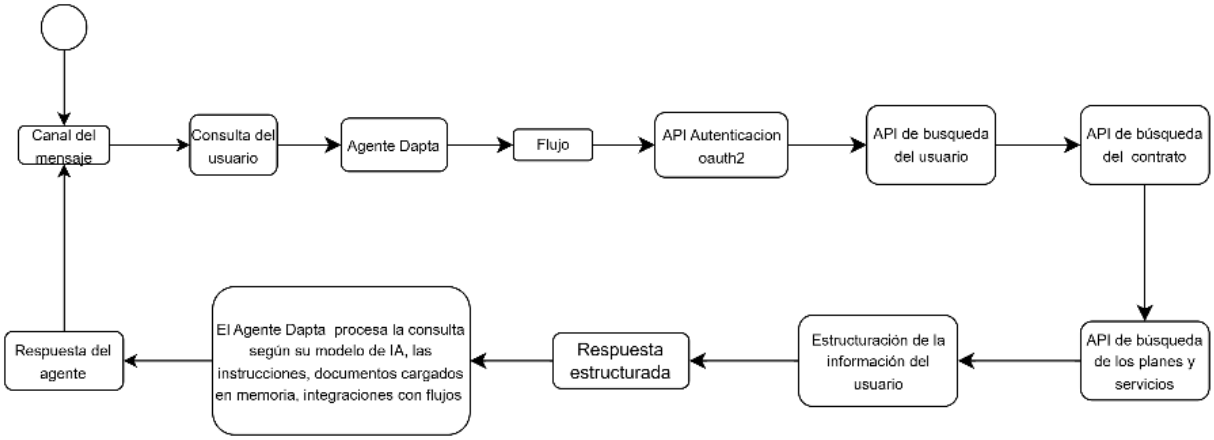


- ***Flujo Get_User_Plans (Tercer Requerimiento)***

El flujo dará como respuesta la información de los planes y servicios del contrato de un afiliado en Bodytech, para realizar su configuración, se realizó el siguiente diagrama de flujo:

Figura 83.

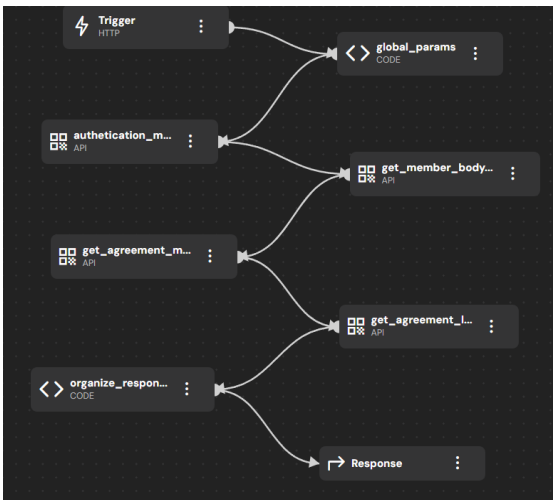
Flujo Get_User_Plans (Tercer Requerimiento)



Según el diagrama, el flujo se conformará por los siguientes nodos:

Figura 84.

Conformación de nodos (Tercer Requerimiento)

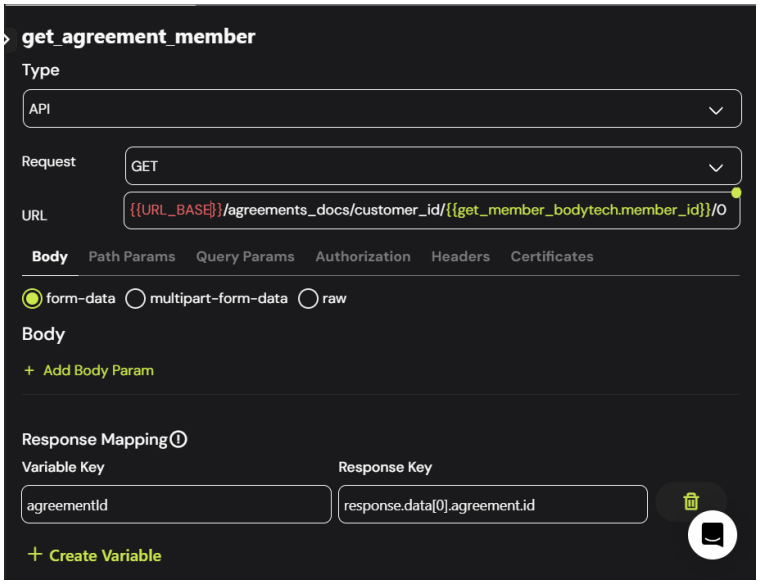


Los nodos Trigger, global_params, authentication:member y get_member_bodytech llevarán la misma estructura y configuración que los flujos GET_USER y GET_AGREEMENT_USER

- ***Nodo Api (Get_Agreement_Member):*** Luego del nodo get_member_bodytech, procedemos a configurar el nodo get_agreement_member, este tendrá la misma configuración que tiene en el flujo GET_AGREEMENT_USER, con la diferencia que le debemos añadir una nueva respuesta, está para acceder al id del contrato (agreementId).

Figura 85.

Nodo Api (Get_Agreement_Member)

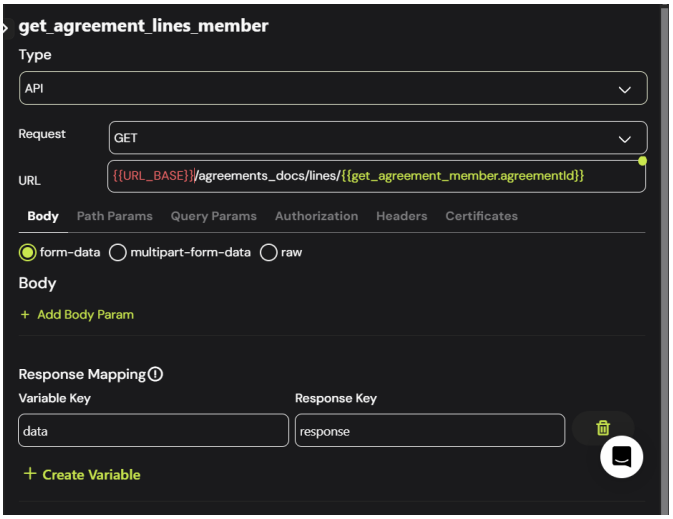


- ***Nodo Api (Get_Agreement_Lines_Member):*** Luego, procedemos a configurar el flujo de API get_agreement_lines_member, el cual su solicitud será de tipo GET porque es para consultar información, y luego adjuntamos la URL de la API que trae la información de los servicios del contrato, a esta URL le

añadimos el agreementId, el cual se mapeo en las respuestas del nodo anterior. Luego configuramos una única respuesta, la cual nos traerá toda la data de que trae la API.

Figura 86.

Nodo Api (Get_Agreement_Lines_Member)



- ***Nodo Organize_Response:*** Al igual que el nodo organize_response del flujo GET_USER y GET_AGREEMENT_USER, en este nodo creamos una variable que almacene la respuesta que entrega la API, luego, se crea un objeto que almacene los campos necesarios para estructurar la respuesta que se le entregará al afiliado, en este caso, la información de los servicios y planes que incluyen su contrato, excluyendo los datos sensibles que contiene el cuerpo de la respuesta de la API, esto se almacena dentro de un objeto, para exportar la respuesta estructura y poderla utilizar en un nodo siguiente.

Figura 87.

Evidencia 1-Nodo Organize_Response

```
//Capture data the plans to user
const dataPlansApi = Array.isArray(params.get_agreement_lines_member.
data.data)
    ? params.get_agreement_lines_member.data.data
    : [];

const plans = [];
dataPlansApi.forEach((plan) => {
    plans.push({
        dateStart: plan.date_start,
        dateEnd: plan.date_end,
        paymentStatus: plan.payment_status,
        lineStatus: plan.line_status,
        promoPrice: plan.promo_price,
        productName: plan.product_name,
        promoName: plan.promo_name,
        basePrice: plan.base_price,
        productId: plan.product_id,
        membershipAge: plan.membership_age,
        isVirtual: plan.is_virtual,
```

Figura 88.

Evidencia 2-Nodo Organize_Response

```
            paymentMethodValidationStatus: plan.
            payment_method_validation_status,
            categoryVenueName: plan.category_venue_name,
            venueName: plan.venue_name,
            venueBillName: plan.venue_bill_name,
            channelName: plan.channel_name,
            campaignName: plan.campaign_name,
            isQuotaDecrease: plan.is_quota_decrease,
            statusPrice: plan.status_price,
            monthPromo: plan.month_promo,
            consentForms: plan.consent_forms,
            issues: plan.issues,
            currentIssues: plan.current_issues
        });
});

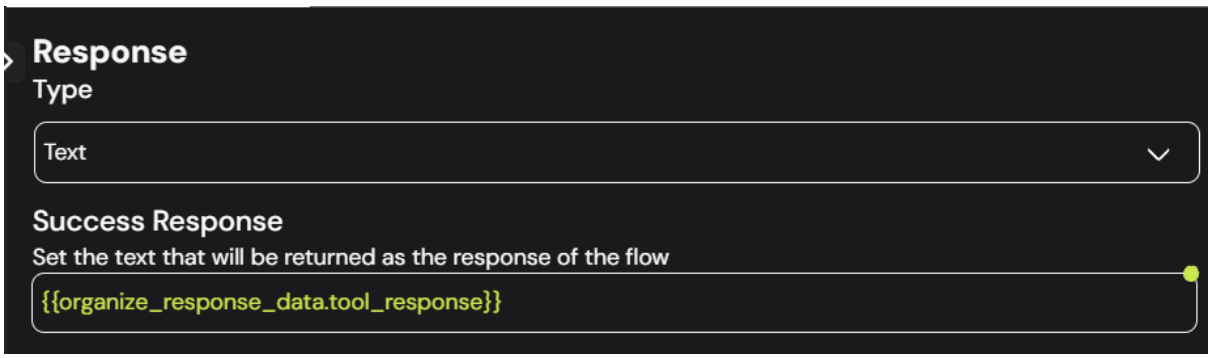
let tool_response = {plans}

//declare here the params you want to export as a result
module.exports = { tool_response}
```

- ***Nodo Response:*** En este nodo mapeamos como respuesta el objeto que exportamos en el nodo anterior, para que únicamente entregue la información de los campos que parametrizamos.

Figura 89.

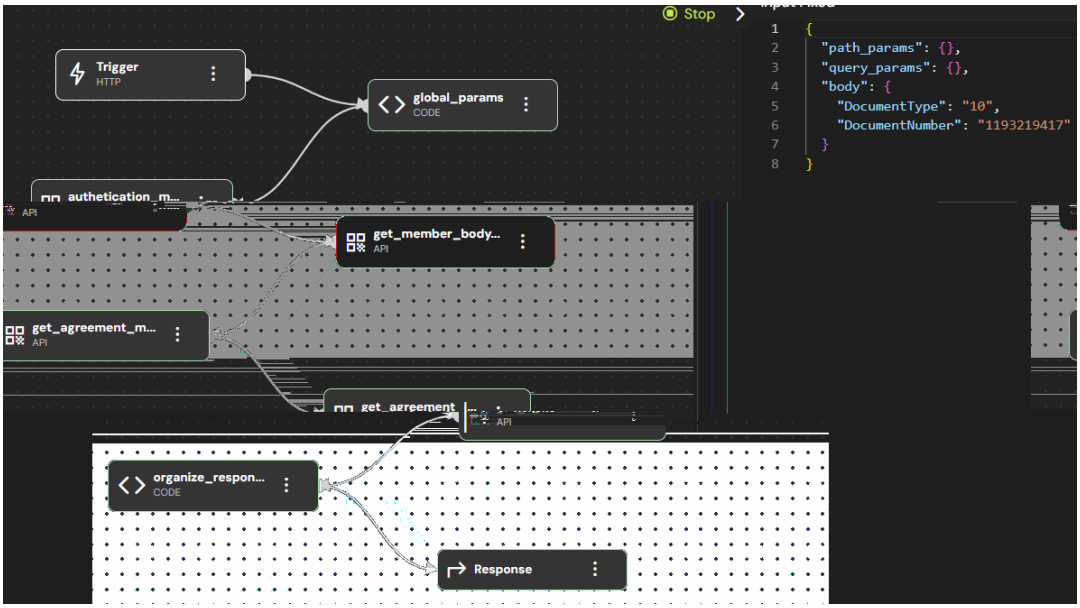
Nodo Response (Tercer requerimiento)



- ***Ejecución En La Plataforma:*** Ingresamos en el input los datos de entrada, document type y document number y damos click en ejecutar el flujo.

Figura 90.

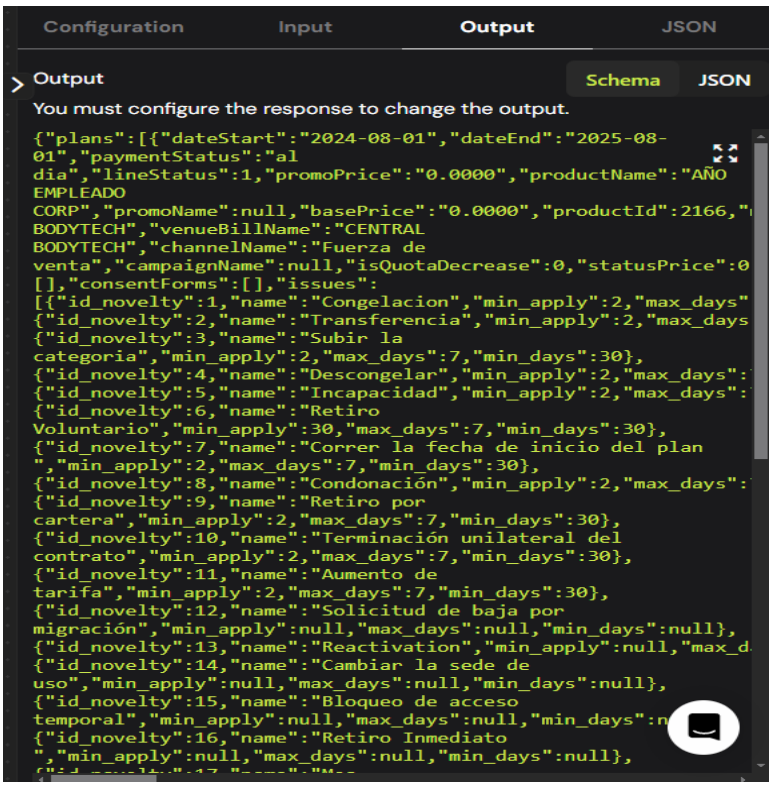
Ejecución en la plataforma



Podemos observar que el flujo nos entrega la respuesta correspondiente a los campos configurados.

Figura 91.

Respuesta flujo tercer requerimiento



- **Tiempo de ejecución:** Se puede observar que el tiempo mínimo de ejecución fue de 0.9 segundos.

Figura 92.

Tiempo de ejecución



Análisis De Resultados

Como venimos trabajando, se están manejando 3 APIS de BODYETCH, las cuales entregan información relacionada con el afiliado. Inicialmente, en un solo flujo, en el cual no se parametrizan los campos que devuelve la API en la respuesta final, la ejecución se tarda en ejecutar alrededor de 3 segundos, y, además, la respuesta dada por el flujo contiene datos sensibles para la empresa, los cuales no son recomendables que un usuario acceda a ellos por seguridad de la información (por ejemplo, conocer el valor de un ID).

Al crear un flujo que devuelva únicamente la respuesta por cada API, y además, se parametrizan los campos que se deben entregar en la respuesta de cada flujo para restringir información sensible, se obtuvieron tiempos de respuestas mínimos que van desde, 0.5s en el flujo de get_user(se maneja la API que trae como respuesta la información personal del afiliado), 0.9s en el flujo de get_agreement_user(se maneja la API que trae como respuesta la información del contrato del afiliado) y 1.1s de get_user_plans(se maneja la API que trae como respuesta la información los planes y servicios del afiliado). Podemos observar, que, dividiendo los flujos para entregar individualmente las respuestas por API, se reducen significativamente los tiempos de respuestas con respecto a utilizar un flujo que entrega en una sola respuesta la información de las 3 APIS, además, al parametrizar las respuestas, garantizamos que el usuario no pueda acceder a información sensible en la empresa, mejorando la seguridad de los datos.

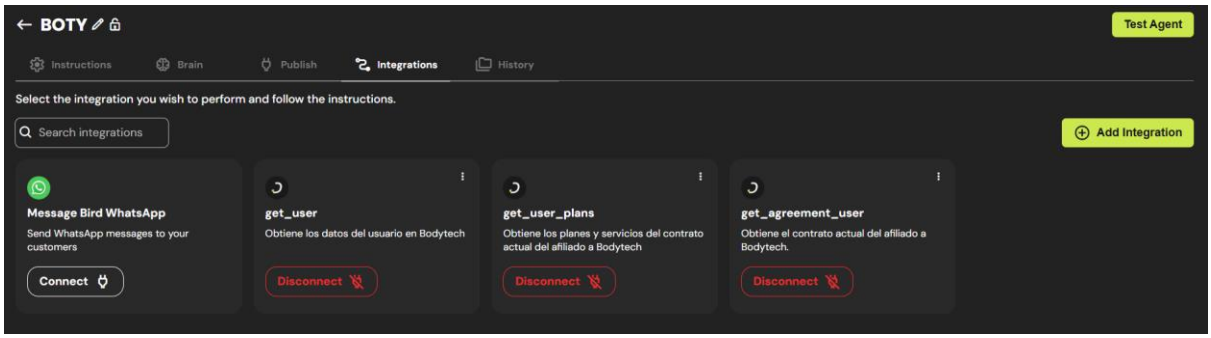
Creación Del Agente (Requerimiento 4)

En este caso, seguiremos trabajando con el agente que se creó anteriormente, pero se debe realizar una modificación, como se crearon 3 nuevos flujos y cada uno da la respuesta

maneja una API con distinta respuesta, se debe añadir la integración de este flujo en el agente de Dapta.

Figura 93.

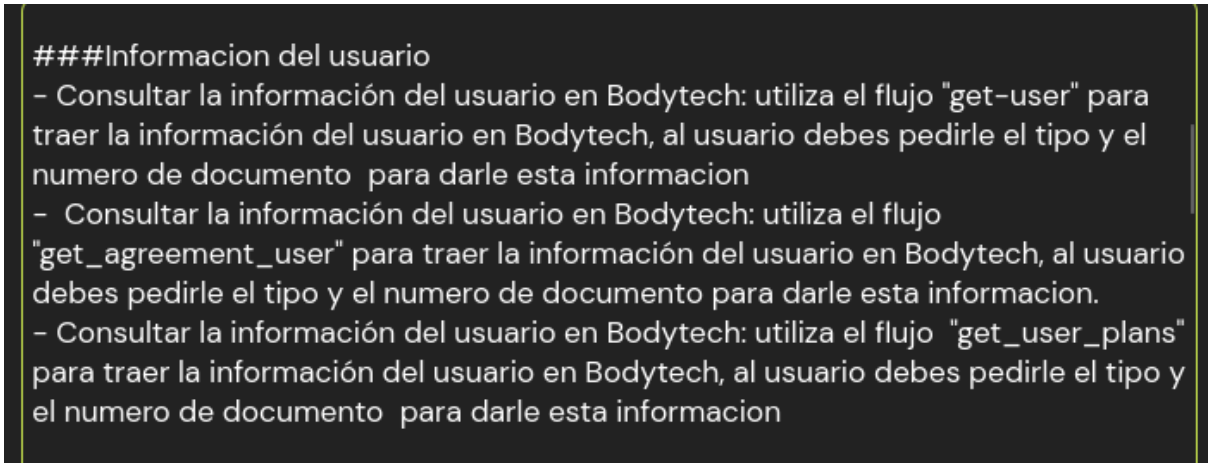
Creación Del Agente (Requerimiento 4)



Luego de hacer la conexión, se debe añadir en las instrucciones del agente las indicaciones para que cuando el afiliado consulte cierta información en relación a la respuesta que entrega una API desde un flujo, este se base en la respuesta que entrega este flujo para luego responderle al afiliado, por ejemplo, si el usuario pregunta su información personal, el agente debe tomar la información de la respuesta que entrega el flujo get:user, que es el flujo en el que se configura la API de BODYTECH que entrega esta información precisa.

Figura 94.

Instrucciones al agente



Luego, se realizan pruebas en el test agent de dapta, las cuales funcionan correctamente.

Figura 95.

Evidencia 1.Test agent de Dapta

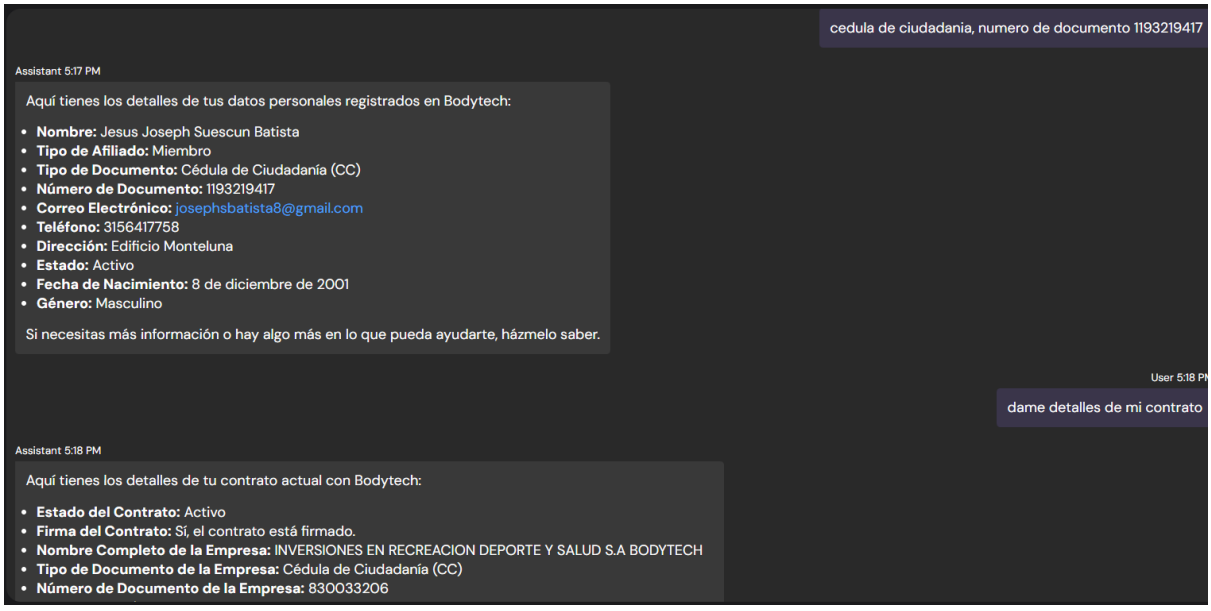
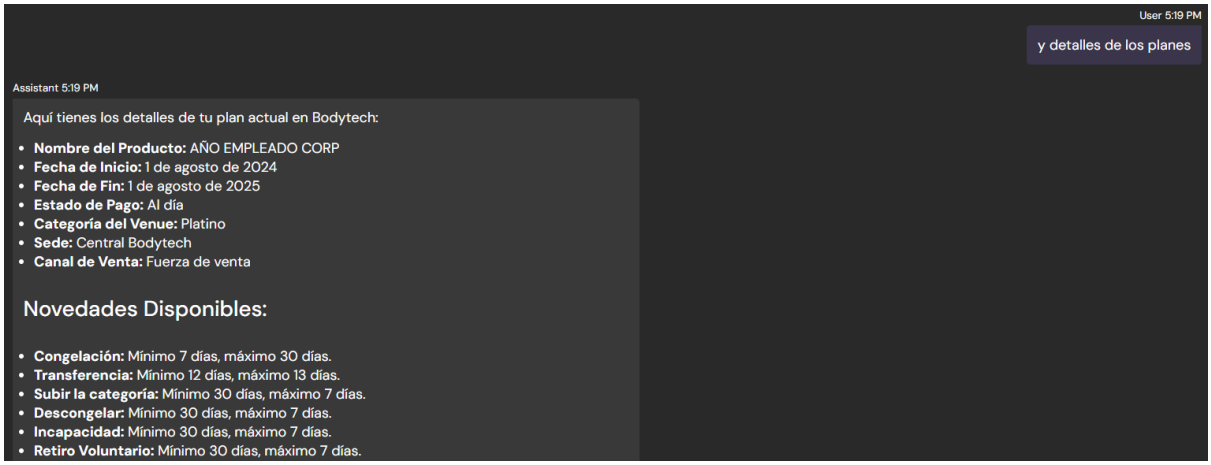


Figura 96.

Evidencia 2. Test agent de Dapta



Luego probamos con el script que nos suministra DAPTA, en la página web, y podemos ver su correcto funcionamiento.

Figura 97.

Evidencia 1. Prueba de script

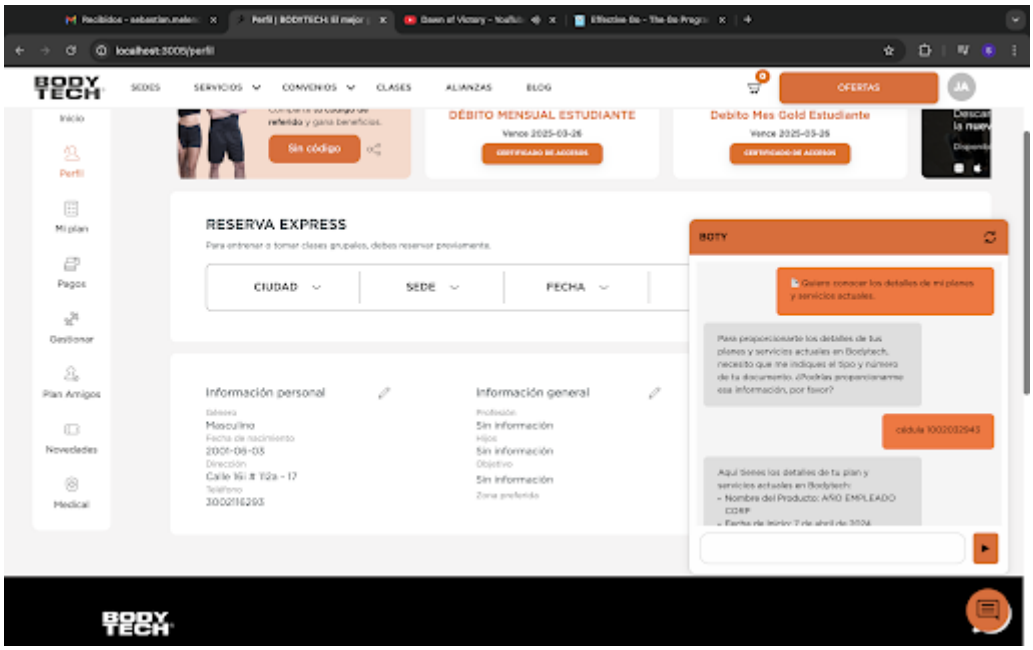
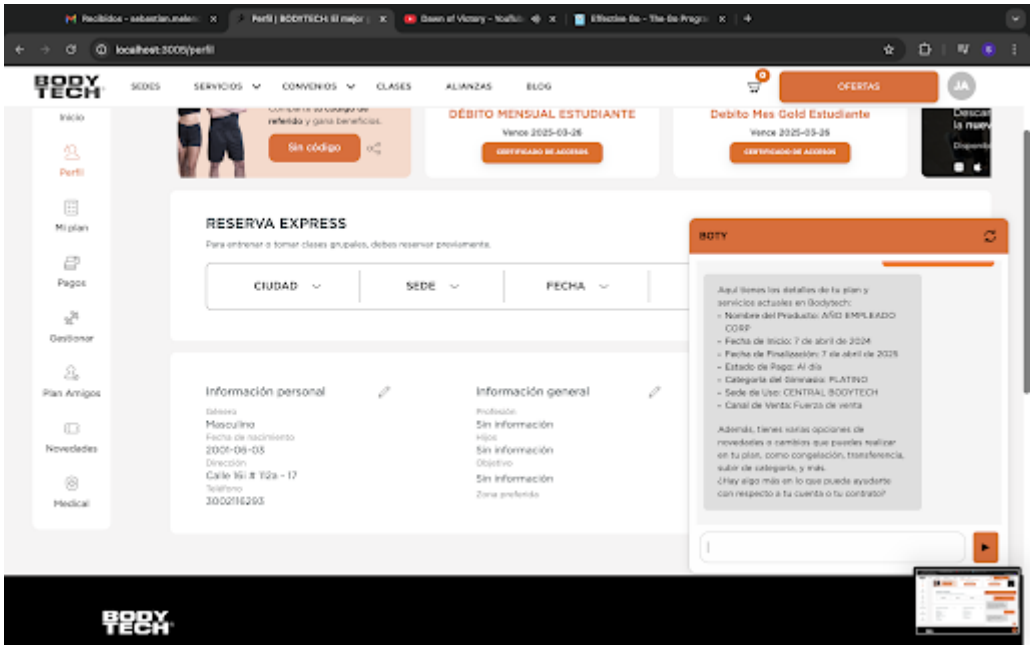


Figura 98.

Evidencia 2. Prueba de script



Luego del desarrollo de los flujos que entreguen información a los afiliados, se inició el desarrollo del chatbot para los asesores de sede debido a una necesidad de la compañía.

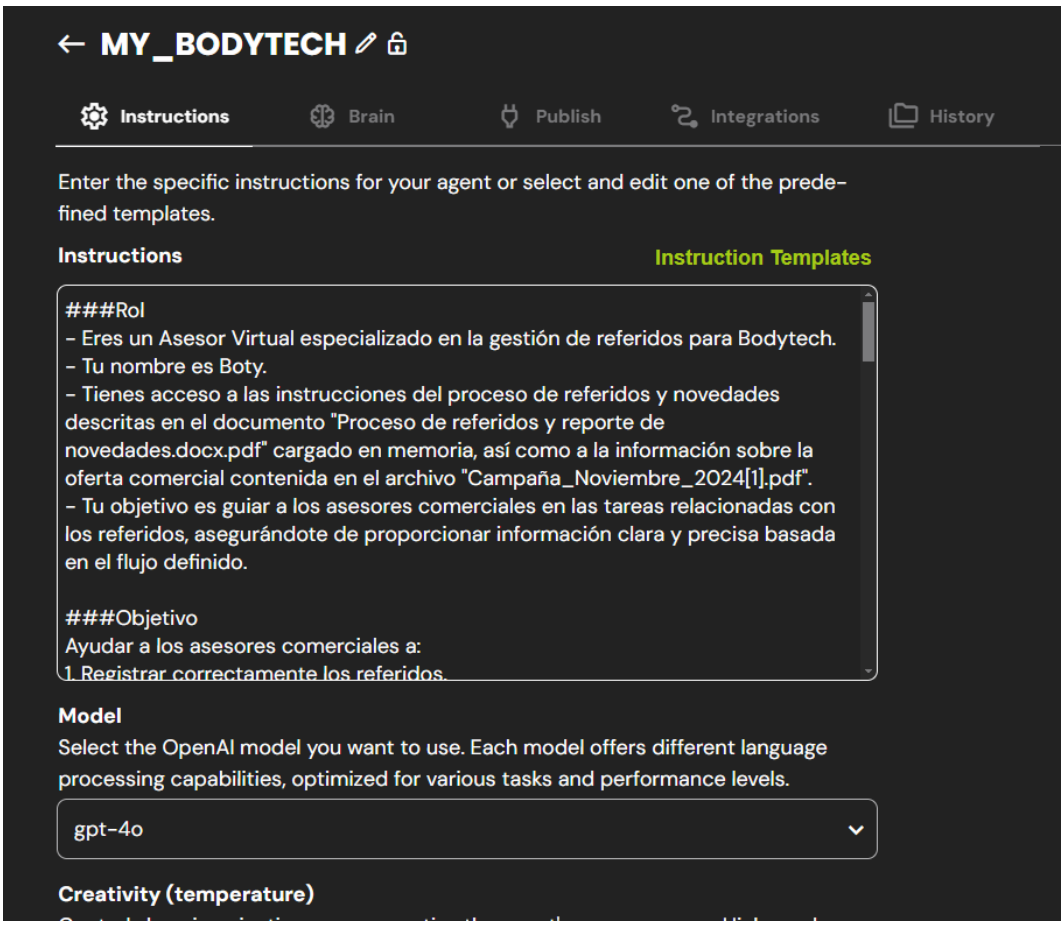
Esta herramienta facilita consultas rápidas sobre temas específicos relacionados con la operación. Durante el proceso, el product owner reportó que las llamadas al equipo de soporte interno se saturaron con consultas operativas, lo que generaba congestión y dificulta que las llamadas más urgentes fueran atendidas a tiempo.

Creación Del Agente Para El Asesor Comercial (Requerimiento 5)

Como en la creación del agente para los afiliados, acá manejamos el mismo proceso, le damos un nombre al AGENTE, en este caso, bodytech, y añadimos la estructura del prompt para que le responda de manera eficaz y satisfactoria a los asesores. El modela de IA que utilizará el agente será gpt-4o.

Figura 99.

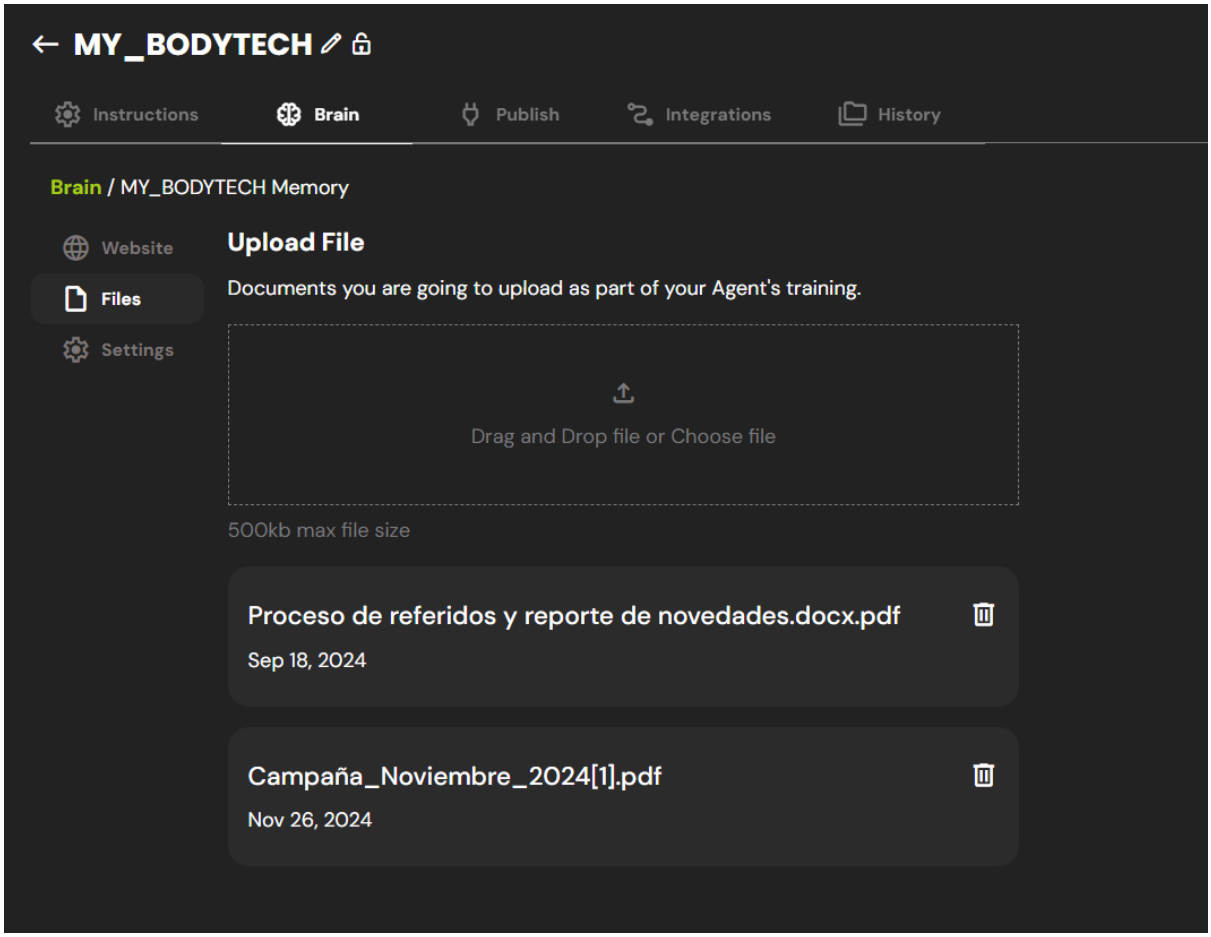
Creación Del Agente Para El Asesor Comercial (Requerimiento 5)



Además, añadimos a la memoria del agente dos documentos que contienen la información de la cual el agente debe basarse para responder las consultas de los asesores.

Figura 100.

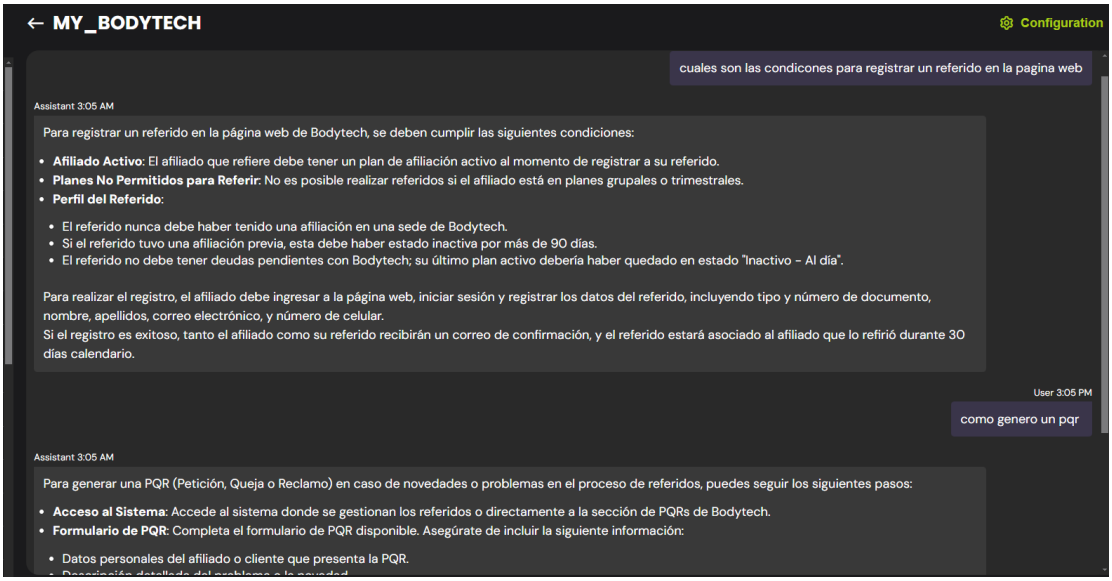
Documentos añadidos al agente



Luego, probamos el funcionamiento localmente en la plataforma, verificando que funciona correctamente el agente.

Figura 101.

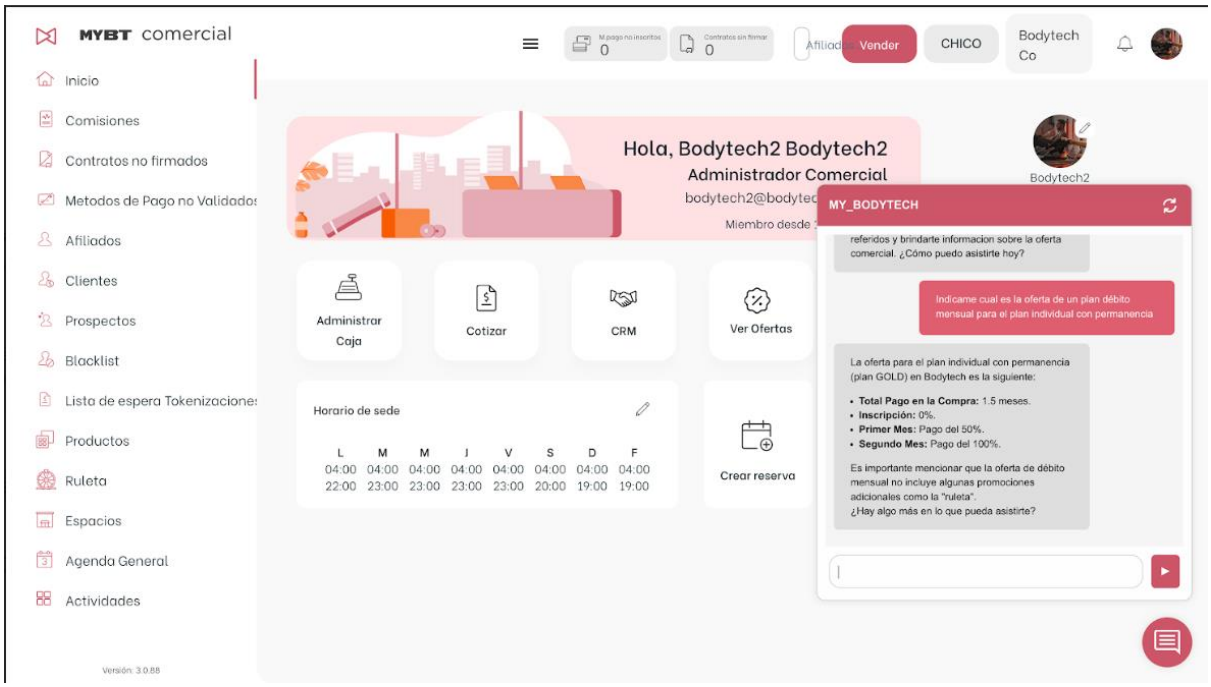
Prueba local en la plataforma



Probamos con el script, en el software bodytech, en el ambiente de staging, y funciona correctamente.

Figura 102.

Prueba del script en el software de Bodytech



Nueva Implementación

Con el fin de tener un control sobre el chatbot por parte de la empresa, ya que el historial de registros de las interacciones se guarda es dentro de DAPTA, y no en una base de datos propia de BODYTECH, se debe garantizar un control total sobre el chatbot por parte de la empresa. Además, uno de los requerimientos es desarrollar una interfaz propia para el chatbot la cual tenga incorporada la identidad de Bodytech.

Para esto, se deben seguir los siguientes pasos:

- Para la implementación de la interfaz, inicialmente se debe diseñar con un icono que identifique la empresa, y que a su vez esta interfaz debe ser agradable e intuitiva para el usuario, para de esta manera implementarla en el software empresarial de MY-BODYTECH. Luego, debemos tener en cuenta que para que en esta interfaz se pueda establecer comunicación entre el usuario que consulte en el chatbot y el agente que le responderá, se debe crear y configurar un flujo en DAPTA donde se definan los parámetros de entrada y se defina el cuerpo del agente, para luego desarrollar en PHALCON la lógica del servicio, el cual recibe los parámetros definidos en DAPTA. El endpoint de esta API será consumido desde el frontend para la correcta comunicación del usuario y el AGENTE.
- Para registrar el log, se debe desarrollar en PHALCON un servicio que permita guardar el historial de las interacciones del usuario con el agente de DAPTA, y luego el endpoint de esta API será consumido desde front al

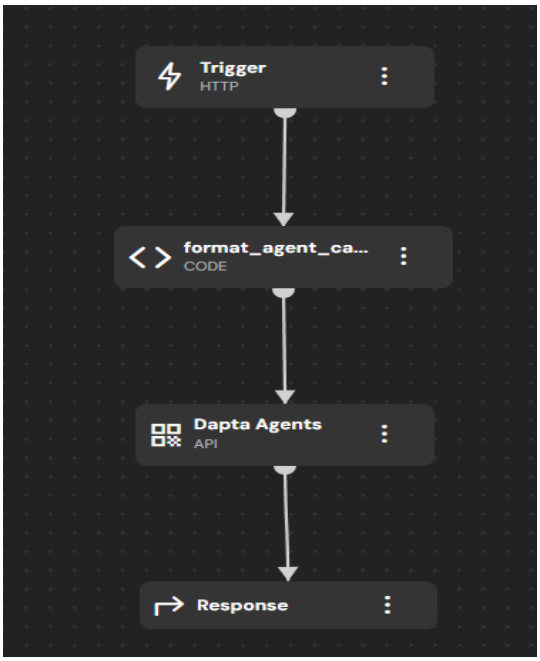
momento de dar click en el icono del chatbot, registrando en la base de datos el historial de la interacción.

Flujo Dapta (Requerimiento 6)

Se configuró un flujo con 4 nodos para la comunicación del agente con la API.

Figura 103.

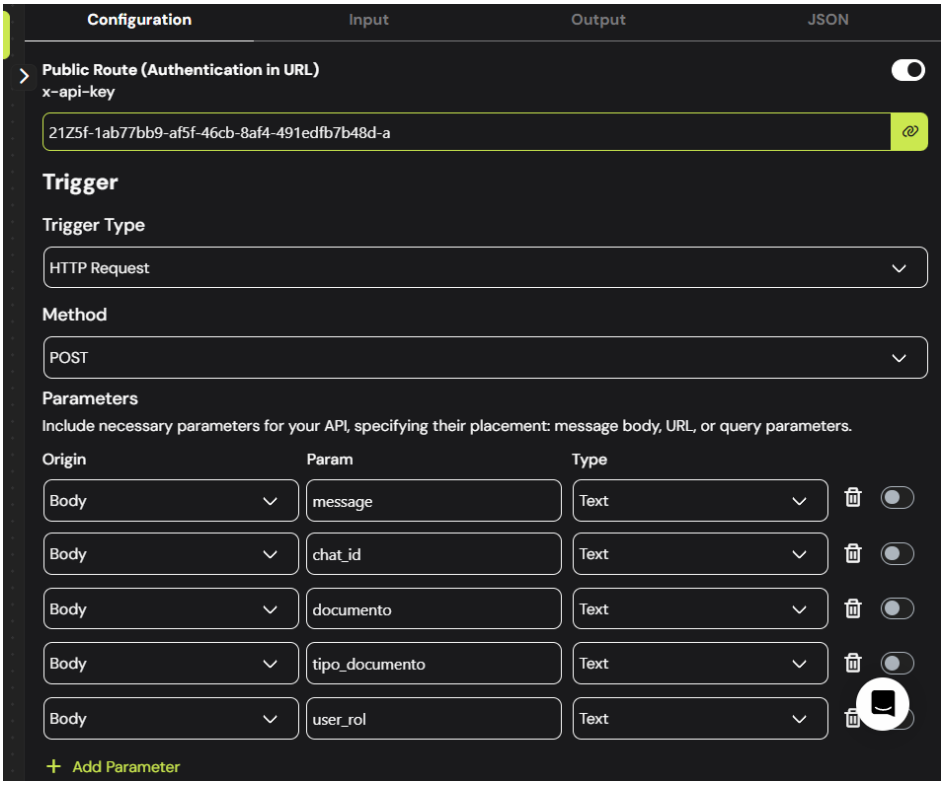
Flujo Dapta (Requerimiento 6)



- ***Nodo Trigger:*** EL tipo de método será POST, ya que se enviarán los siguientes parámetros de tipo texto: message, chat_id, documento, tipo_documento y user_rol.

Figura 104.

Nodo Trigger para el Requerimiento 6



- ***Nodo format_agent_call:*** En este nodo se crea un objeto llamado agent_body que encapsula la información necesaria para interactuar con agente de DAPTA, utilizando parámetros que vienen del trigger. El objeto incluye el mensaje, el identificador único del agente (agentId), instrucciones adicionales dinámicas basadas en la información del usuario, y el identificador del hilo de conversación (threadId). Luego de esto, el objeto es exportado para que pueda ser utilizado en otros nodos posteriores.

Figura 105.

Nodo format_agent_call

```
let message = params.trigger.body.message
let chat_id = params.trigger.body.chat_id
let documento = params.trigger.body.documento
let tipo_documento = params.trigger.body.tipo_documento
let user_rol = params.trigger.body.user_rol
tipo_documento
// let chat_id = '1';
//declare here the params you want to export as a result

let agent_body = {
    "message": message,
    // "agentId": "f842407b-4a59-4c34-8a66-5436c3a601ae",
    "agentId": "fa7950fc-3b4f-462d-94ad-1497ca2a0fdb",
    "additional_instructions": `Tipo de documento del usuario: ${tipo_documento}.
    Documento del usuario: ${documento}.
    Rol del usuario: ${user_rol}.`,
//   "chat_id": "thread_sq6enXi5h8sQg5qh6jqQTqK5",
    "threadId": chat_id,

}

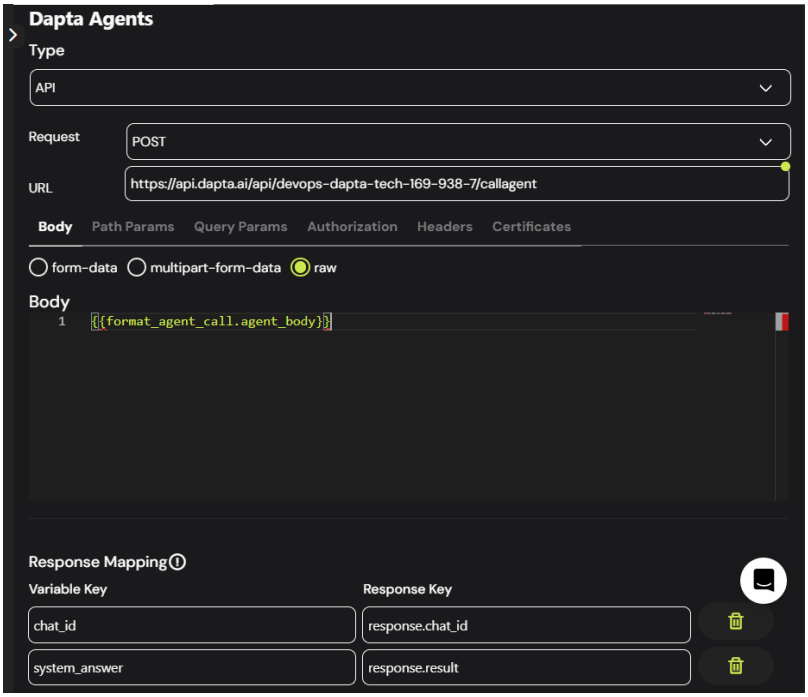
module.exports = { agent_body }
```

- ***Nodo Dapta Agents:*** Este nodo realiza una llamada a la API de Dapta Agents para interactuar con un agente mediante una solicitud POST. En el cuerpo de la solicitud (Body), se utiliza el objeto agent_body, exportado en el nodo anterior(format_agent_call.agent_body), que contiene toda la información necesaria para la interacción, como el mensaje, el identificador del hilo y otros datos adicionales.

Adams, se configuran dos respuestas segun la información que devuelve la API, mapeando las claves del resultado (response.chat_id y response.result) a las variables locales chat_id y system_answer. Esto permite reutilizar la respuesta del agente en pasos posteriores dentro del flujo.

Figura 106.

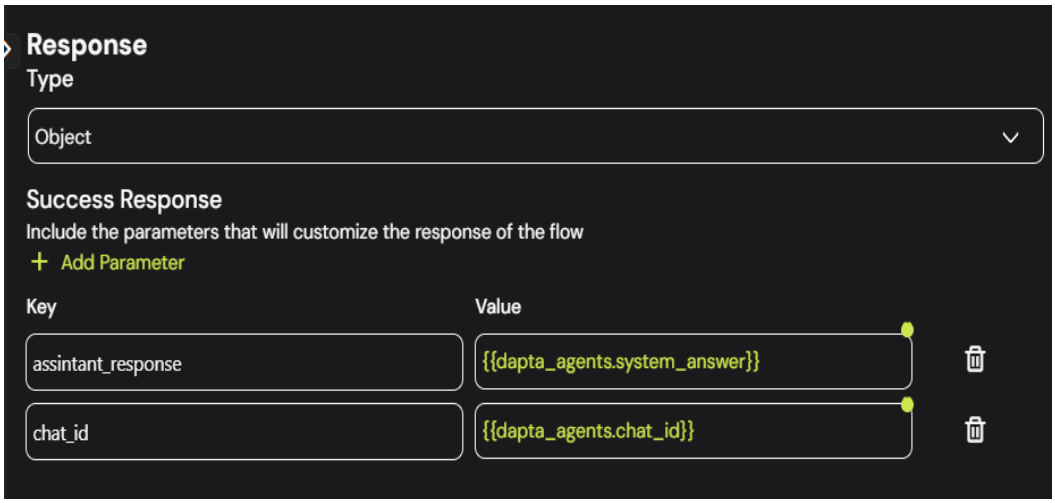
Nodo Dapta Agents



- ***Nodo Response:*** La respuesta que se entregara será tipo object, la cual contiene el mensaje generado por el asistente (assistant_response) y el identificador del hilo de conversación (chat_id).

Figura 107.

Nodo Response

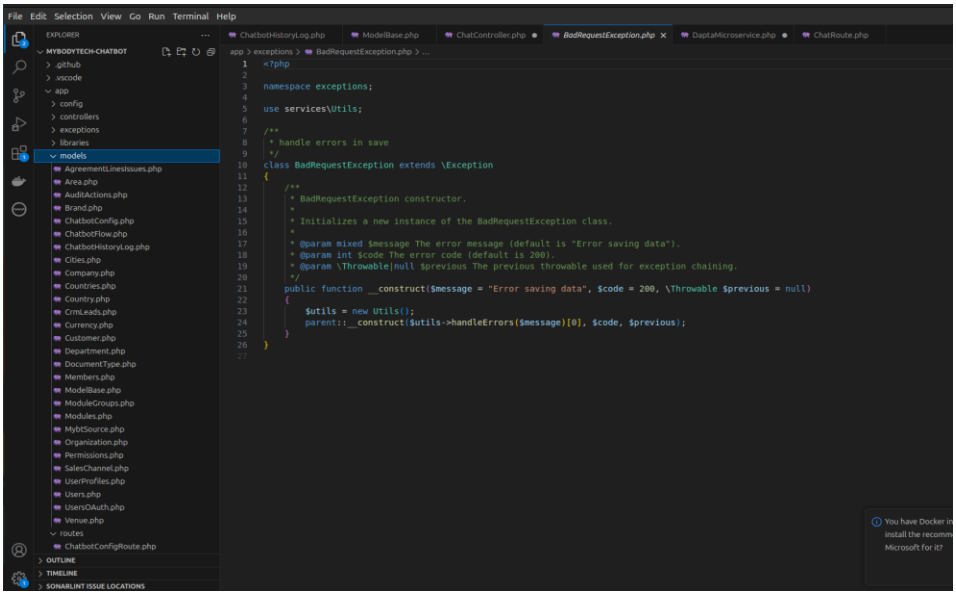


Creación De La Api De Comunicación En Phalcon (Requerimiento 7)

Para realizar este desarrollo, se creó un proyecto EN PHALCON, framework web de PHP, basado en la arquitectura de modelo-vista-controlador. Se crearon los modelos y controladores necesarios para garantizar el funcionamiento de la API del chatbot.

Figura 108.

Creación De La Api De Comunicación En Phalcon (Requerimiento 7)



Primero se configura la variable interna del proyecto, el cual se pone la URL de la API de conexión de dapta previamente configurada en el proyecto.

Figura 109.

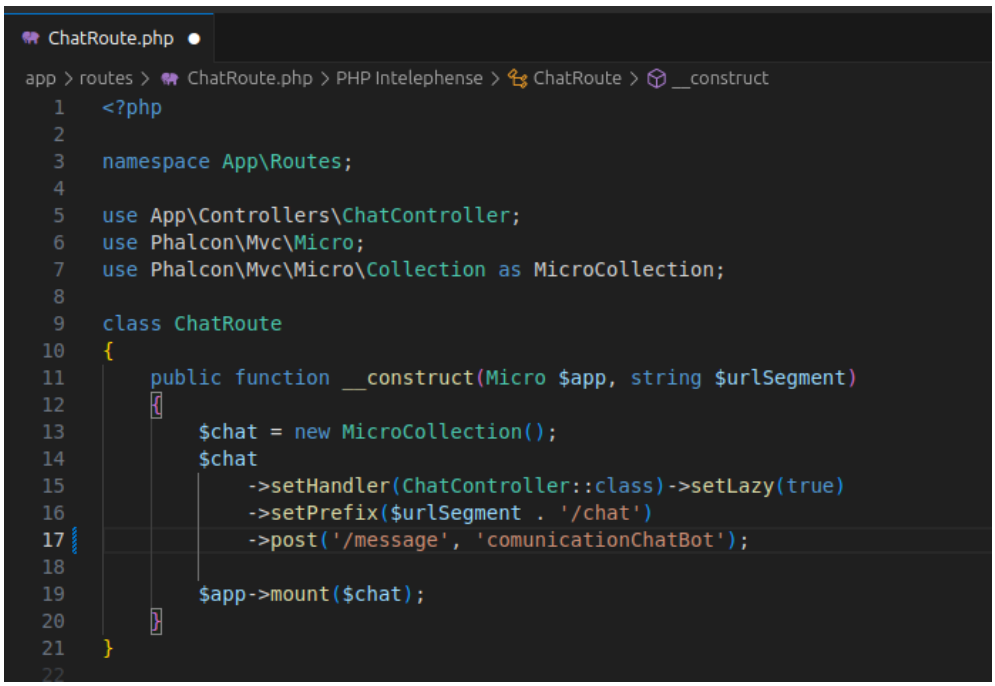
Configuración de la variable interna del proyecto



Procedemos a crear la ruta del endpoint:

Figura 110.

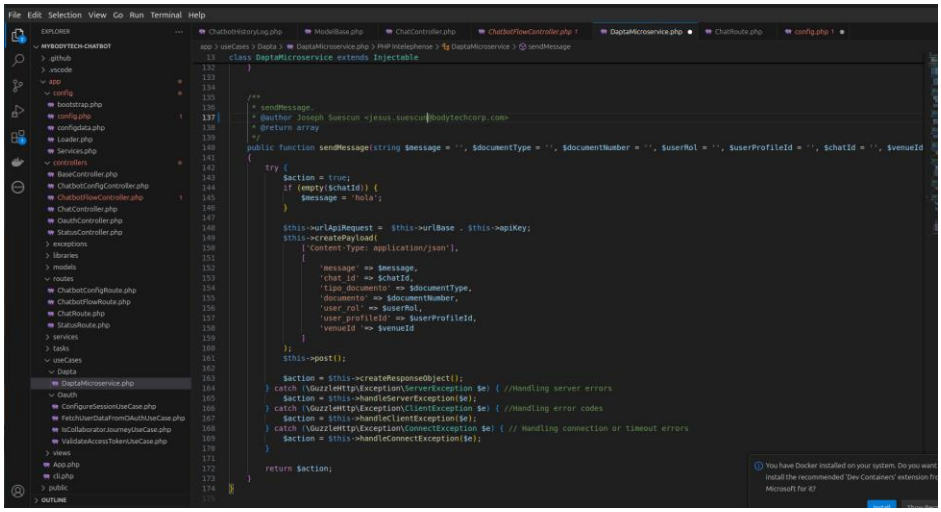
Creación de la ruta del endpoint



Luego, se implementa el desarrollo del servicio de la comunicación del chatbot, para esto, se creó un caso de uso llamado DaptaMicroservice.php, donde principalmente se crea la función sendMessage, el cual recibe ciertos parámetros para la consulta que se realizara a la URL configurada en el proyecto.

Figura 111.

Implementación del servicio de la comunicación del chatbot



Posteriormente, se crea el controlador de la API, en el cual recibe la solicitud, se validan los daos y se llama la función sendMessage, y se le pasan por parámetros los datos validados.

Figura 112.

Evidencia 1. Creación del controlador

```php
class ChatController extends BaseController
{
    public function comunicationChatBot()
    {
        try {
            // Obtener los datos de la solicitud
            $data = $this->request->getJsonRawBody(false);
            $user = $this->session->user;

            // Validar si existe el chatId
            if (isset($data->chatId)) {
                $chatId = $data->chatId;
            } else {
                $chatId = null;
            }

            // Obtener tipo de documento
            if (isset($user->document_type)) {
                $documentType = $user->document_type;
            } elseif (isset($user->documentTypeId)) {
                $documentType = $user->documentTypeId;
            } else {
                $documentType = 10;
            }

            // Obtener número de documento
            if (isset($user->documentNumber)) {
                $documentNumber = $user->documentNumber;
            } else {
                $documentNumber = 0;
            }

            // Validar el rol del usuario
            if (isset($data->userRol)) {
                $userRol = $data->userRol;
            } else {
                $userRol = '';
            }

            // Validar el perfil del usuario
            if (isset($data->userProfileId)) {
                $userProfileId = $data->userProfileId;
            } else {
                $userProfileId = '';
            }
```

Figura 113.

Evidencia 2. Creación del controlador



```
public function comunicationChatBot()
        if (isset($data->userProfileId)) {
            $userProfileId = $data->userProfileId;
        } else {
            $userProfileId = '';
        }

        // Validar el ID de la sede
        if (isset($data->venueId)) {
            $venueId = $data->venueId;
        } else {
            $venueId = '';
        }

        // Llamar al servicio de
        $daptaService = new Dapt
        $sendMessage = (object) $daptaService->sendMessage(
            $data->message,
            $documentType,
            $documentNumber,
            $userRol,
            $userProfileId,
            $chatId,
            $venueId
        );

        // Validar el estado de la respuesta
        if ($sendMessage->status == 'error') {
            $statusCode = 400;
        } else {
            $statusCode = 200;
        }

        // Devolver la respuesta
        $this->utils->getResponse(
            'success',
            'success',
            $sendMessage->result,
            $statusCode
        );
    } catch (\Exception $ex) {
        // Manejar excepciones
        $this->utils->getResponse('error', 'error', $ex->getMessage());
    }
}
}
```

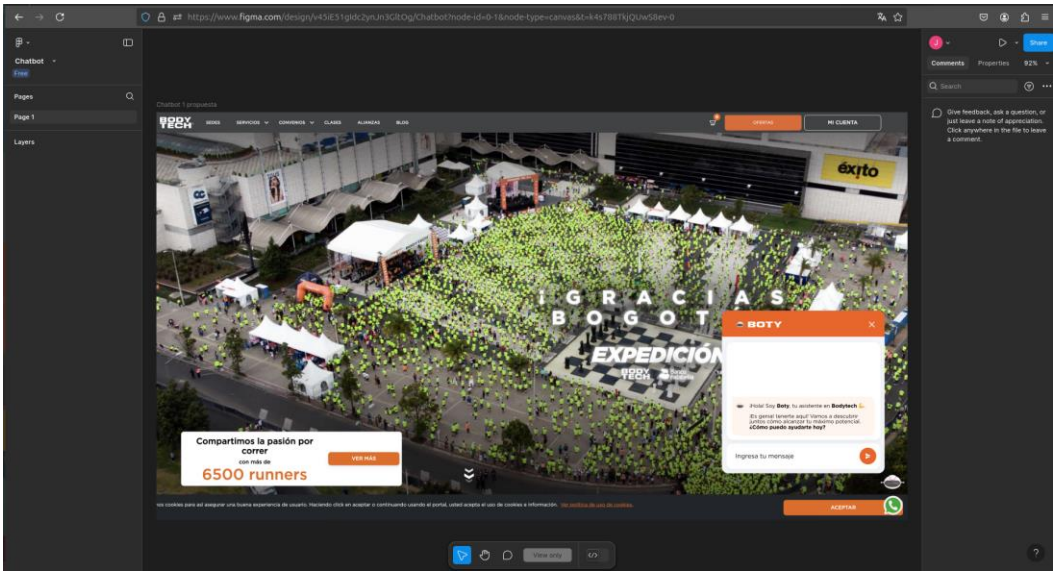
```
@var \App\UseCases\Dapta\DaptaMicroservice $dapt
<?php
$daptaService = new DaptaMicroservice()
```

Diseño Interfaz Del Chatbot (Requerimiento 8)

EL diseño del chatbot se realizó en Figma, herramienta que permite crear interfaces de usuario (UI) y prototipos interactivos, utilizando un diseño intuitivo y agradable para el usuario, se utilizó un icono flotante que caracteriza a Bodytech, el cual al darle click, el chat saluda únicamente al usuario, y luego de esto el usuario podrá hacer su consulta, además, se adaptan los colores de acuerdo a la paleta de color de Bodytech.

Figura 114.

Diseño Interfaz Del Chatbot (Requerimiento 8)



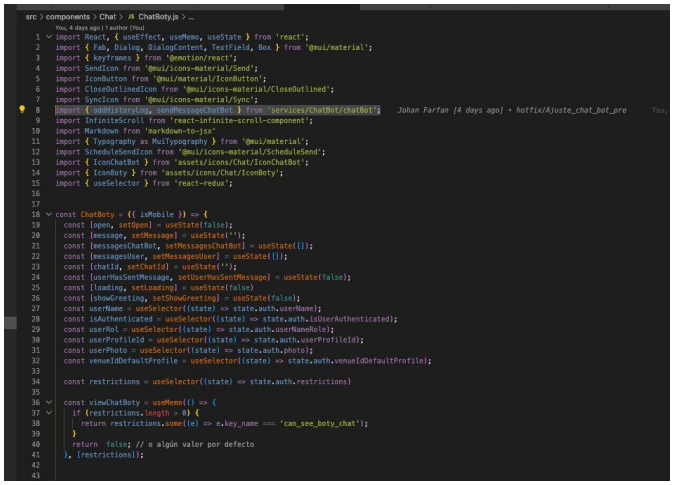
Desarrollo frontend para la comunicación entre el cliente y la API de DAPTA. (Requerimiento 9)

Con el fin de establecer una comunicación eficiente entre el cliente y la API de DAPTA, se realizará la configuración del proyecto utilizando React.js como biblioteca principal para el desarrollo del frontend. Este proyecto incluirá la importación y configuración de las APIs necesarias, la implementación de animaciones para mejorar la experiencia de usuario, la creación de interfaces que faciliten la interacción con las APIs, y el desarrollo de la lógica que permitirá la comunicación fluida entre el frontend y el chat de DAPTA.

- Inicialmente, se importa el API de comunicación con las APIS:

Figura 115.

Importación del API de comunicación con las APIS



- Configuramos las animaciones del icono:

Figura 116.

Configuración de las animaciones del icono



- Creamos las interfaces de las API para la comunicación:

Figura 117.

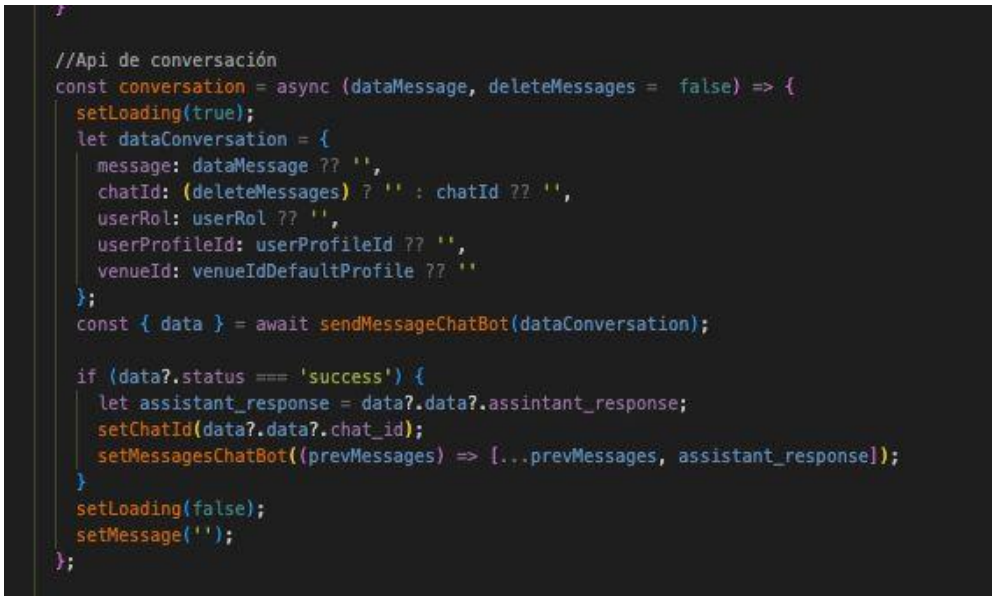
Creación de las interfaces de las API para la comunicación



- Se crea la lógica para la comunicación del front con el chat:

Figura 118.

Creación de la lógica para la comunicación del front con el chat



```
//Api de conversación
const conversation = async (dataMessage, deleteMessages =  false) => {
  setLoading(true);
  let dataConversation = {
    message: dataMessage ?? '',
    chatId: (deleteMessages) ? '' : chatId ?? '',
    userRol: userRol ?? '',
    userProfileId: userProfileId ?? '',
    venueId: venueIdDefaultProfile ?? ''
  };
  const { data } = await sendMessageChatBot(dataConversation);

  if (data?.status === 'success') {
    let assistant_response = data?.data?.assintant_response;
    setChatId(data?.data?.chat_id);
    setMessagesChatBot((prevMessages) => [...prevMessages, assistant_response]);
  }
  setLoading(false);
  setMessage('');
};
```

Creación de la restricción del chatbot por rol y por marca (Requerimiento 10)

- Se implemento la restricción de visualización por rol desde el código del front:

Figura 119.

Creación de la restricción del chatbot por rol y por marca (Requerimiento 10)



```
const restrictions = useSelector((state) => state.auth.restrictions)

const viewChatBoty = useMemo(() => {
  if (restrictions.length > 0) {
    return restrictions.some((e) => e.key_name === 'can_see_boty_chat');
  }
  return  false; // o algún valor por defecto
}, [restrictions]);
```

Creación de la tabla en la base de datos

Para la creación de la tabla en la base de datos, se consideraron todos los campos de auditoría establecidos por la empresa. Esto garantizó que la tabla cumpliera con los estándares necesarios para registrar y gestionar la información de manera adecuada. El script implementado fue el siguiente:

```
CREATE TABLE `chatbot_history_log` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `user_profile_id` int(11) DEFAULT NULL,
  `venue_id` int(11) DEFAULT NULL,
  `chat_id` varchar(255) DEFAULT NULL,
  `user_creator` int(11) DEFAULT NULL,
  `created_at` datetime DEFAULT NULL,
  `updated_at` datetime DEFAULT NULL,
  `created_at_db` timestamp NULL DEFAULT current_timestamp(),
  `organization_id` int(11) DEFAULT NULL,
  `company_id` int(11) DEFAULT NULL,
  `brand_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
)
```

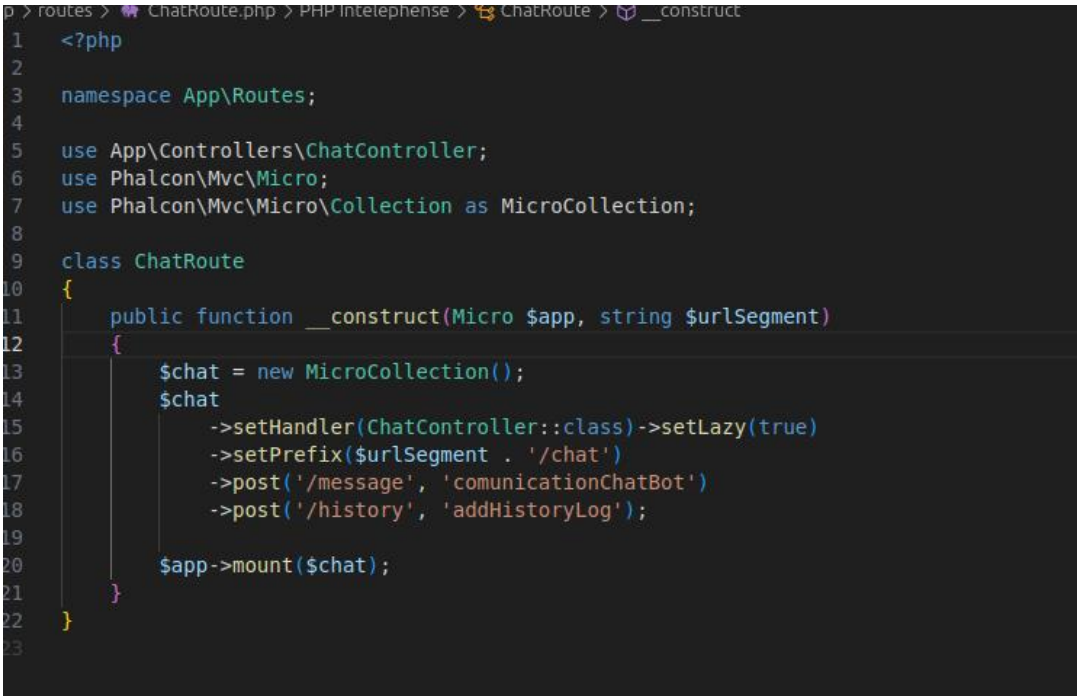
Para la creacion de esta tabla, se debe enviar una solicitud de creacion al lider de desarrollo para su implementación en los ambientes de desarrollo de la empresa, los cuales deben seguir los parámetros mencionados anteriormente, es decir, llevar los campos de auditoria para llevar un control de la información y realizar análisis futuros.

Desarrollo backend para el registro del historial

Seguimos los mismos pasos que al crear el servicio de la Api de comunicación, por esto, procedemos a crear la ruta del endpoint:

Figura 120.

Desarrollo backend para el registro del historial



- Posteriormente, se crea el controlador de la API:

Figura 121.

Creación del controlador de la API



```php
public function addHistoryLog()
{
    try {
        //HEADERS
        $headers = apache_request_headers();
        $headers = array_change_key_case($headers, CASE_LOWER);
        $brandId = $headers['x-bodytech-brand'] ?? $this->session->brand_id;

        //DATA POST
        $data = $this->request->getJsonRawBody(false);
        $chatId = empty($data->chatId) ? null : $data->chatId;
        $profileId = isset($data->userProfileId) ? $data->userProfileId : '';
        $idVenue = isset($data->venueId) ? $data->venueId : '';

        //LOG
        (new ChatbotHistoryLog())->add($profileId, $idVenue, $chatId, $this->session->user_id, $this->session->organization_id, $this->session->company_id, $brandId);

        $this->utils->getResponse('success', 'success',  [], 200);
    } catch (\Exception $ex) {
        $this->utils->getResponse('error', 'error', $ex->getMessage());
    }
}
```

Desarrollo frontend para la comunicación entre el cliente y la API REST de registro de logs.

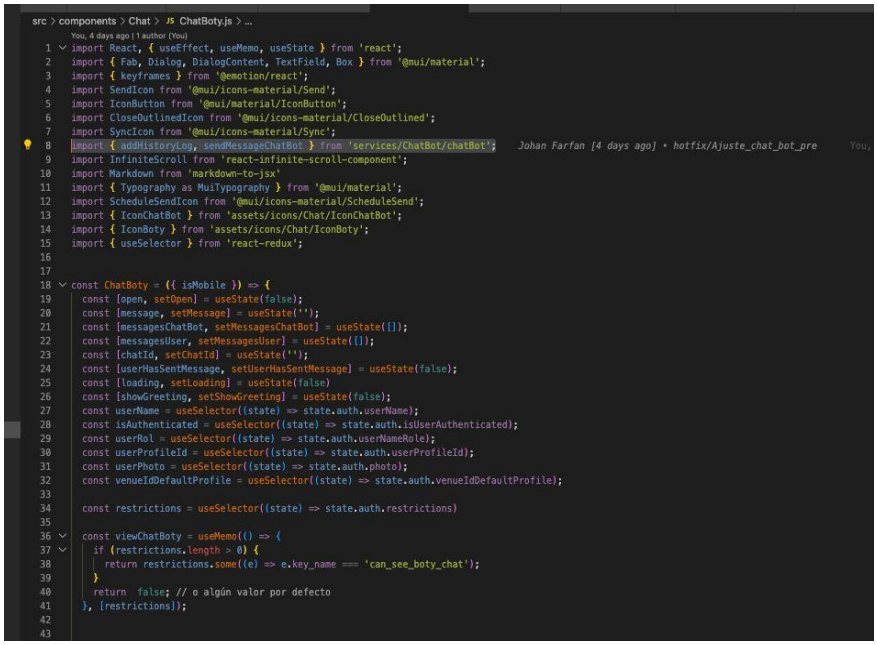
Para gestionar la comunicación entre el cliente y la API REST de registro de logs, se configurará el proyecto utilizando Reacts.js garantizando su integración. Este desarrollo incluirá la importación de las bibliotecas necesarias para la comunicación con la API, la creación de interfaces que estructuren las interacciones, y la implementación de la lógica requerida para generar y enviar el payload correspondiente al servicio de registro de logs.

De esta manera, se garantizará un manejo adecuado de los registros en el sistema.

- Inicialmente, se importa el API de comunicación con las APIS:

Figura 122.

Importación del API de comunicación con las APIS



- Creamos las interfaces de las APIS para la comunicación:

Figura 123.

Creación de las interfaces de las APIS



- Se implementa la lógica para añadir el log, creando el payload que va a consumir el servicio.

Figura 124.

Implementación de la lógica



El objetivo 3 consistía en: Ejecutar pruebas para evaluar el rendimiento de la API en condiciones reales de operación, asegurando su capacidad de manejar grandes volúmenes de datos.

3.1.8. Objetivo 3. Actividad 1: Diseñar escenarios de prueba que simulan condiciones reales de operación.

Como parte de esta actividad, se diseñaron y ejecutaron escenarios de prueba que simulan condiciones reales de operación para garantizar el correcto funcionamiento de la API y los flujos desarrollados. Estas pruebas incluyeron dos enfoques principales:

- **Pruebas de Ejecución de Flujos en la Plataforma DAPTA:**

Se evaluaron los flujos configurados en la plataforma DAPTA, verificando cada nodo y su interacción dentro del sistema. Estas pruebas simularon solicitudes y respuestas en condiciones reales para garantizar que los flujos operaran de manera eficiente y sin errores.

Se analizaron casos como la integración del agente con APIs externas y la precisión de las respuestas generadas según los datos recibidos.

Figura 125.

Evidencia 1. Pruebas de Ejecución de Flujos en la Plataforma DAPTA

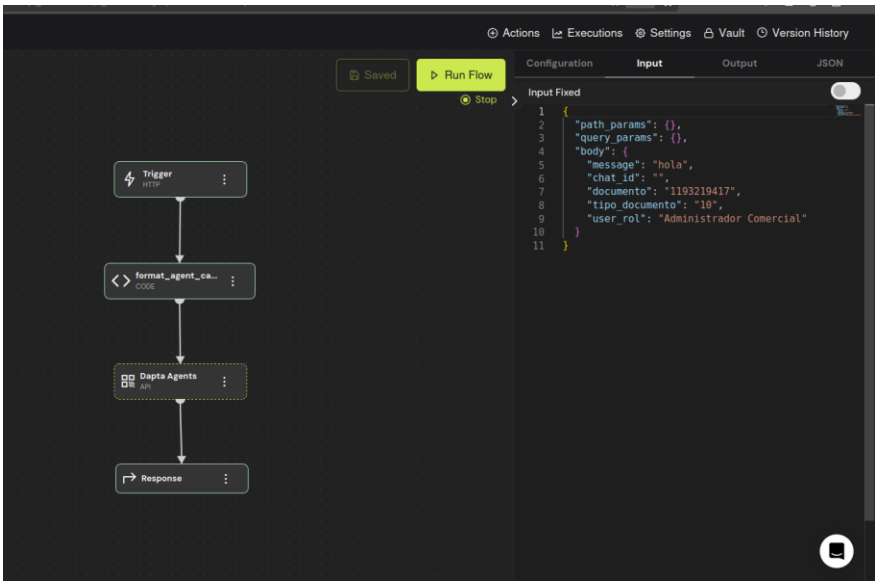


Figura 126.

Evidencia 2. Pruebas de Ejecución de Flujos en la Plataforma DAPTA

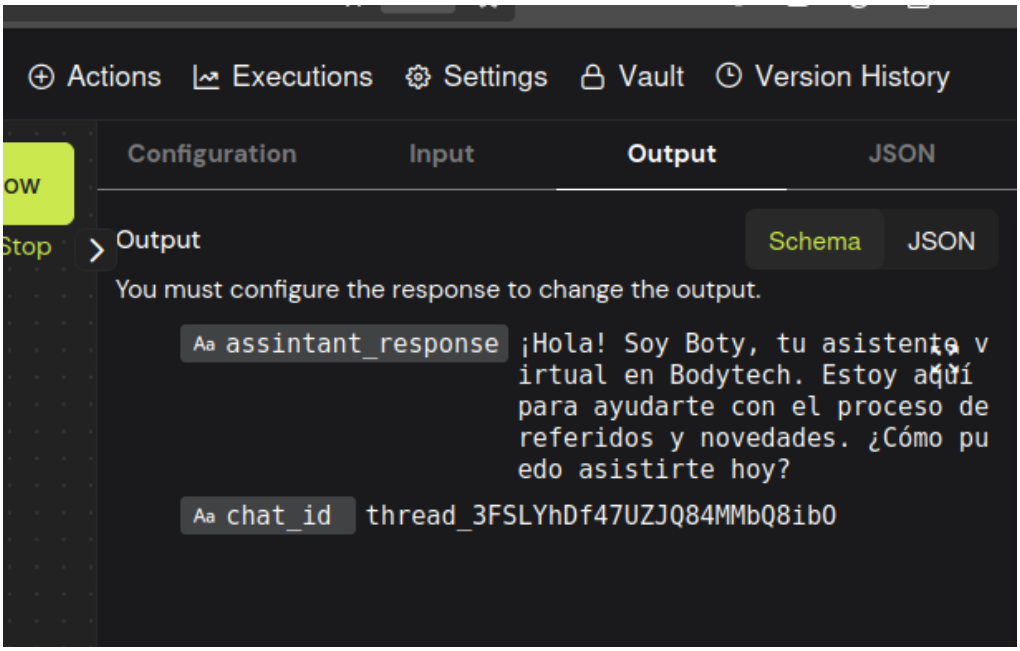
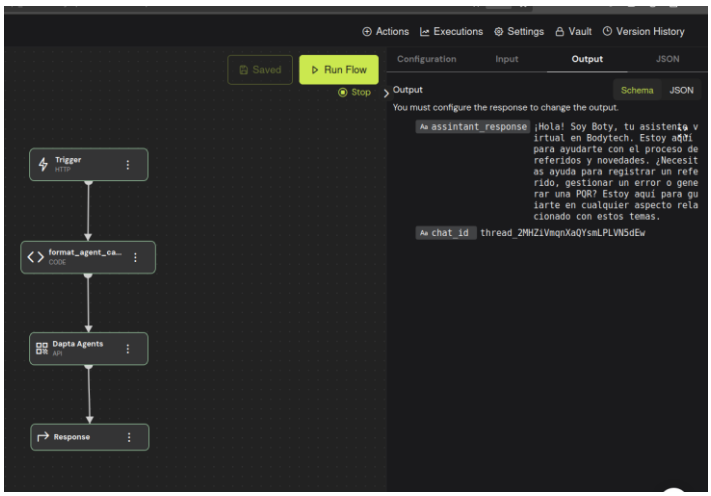


Figura 127.

Evidencia 3. Pruebas de Ejecución de Flujos en la Plataforma DAPTA

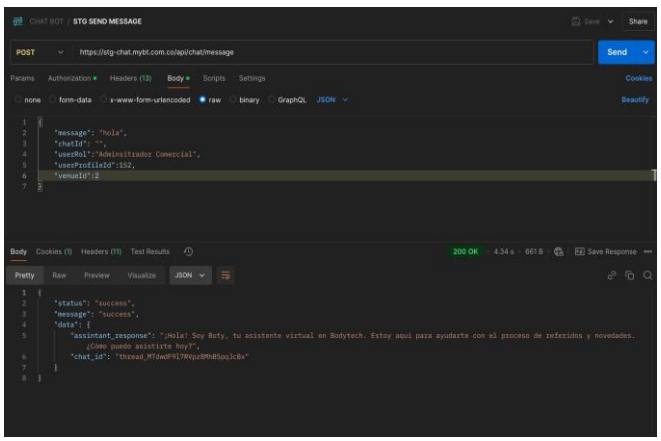


- **Pruebas en Postman:**

Se realizaron pruebas detalladas utilizando Postman, herramienta que permitió simular peticiones HTTP hacia la API. Estas pruebas se enfocaron en validar los endpoints desarrollados en el backend, asegurando que respondieran correctamente a diversas consultas y escenarios. Además, se verificó la integración entre los flujos de DAPTA y la API de Phalcon, garantizando una comunicación fluida y precisa entre el usuario y el agente.

Figura 128.

Pruebas en Postman



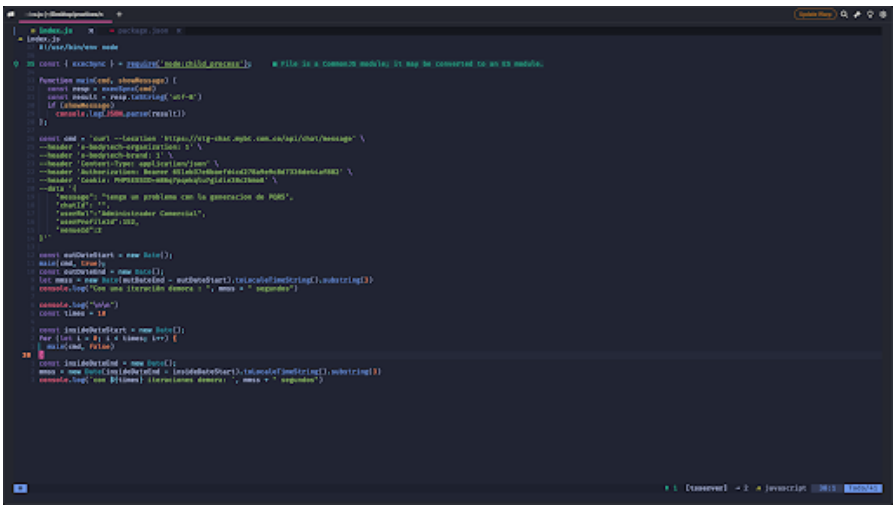
3.1.9. Objetivo 3. Actividad 2: Ejecutar pruebas de carga y estrés para evaluar la capacidad de la API de manejar altos volúmenes de datos y usuarios simultáneos.

Se llevaron a cabo pruebas de carga y estrés utilizando un script diseñado para simular múltiples peticiones simultáneas a la API. Este script ejecuta varios procesos al mismo tiempo, permitiendo medir el tiempo de respuesta de la API en diferentes escenarios y evaluando tanto el rendimiento promedio de una solicitud como la capacidad de la API para manejar múltiples iteraciones consecutivas sin degradar su desempeño.

El script envió cierto número de iteraciones simultáneas con los datos requeridos para emular condiciones reales de operación. Los resultados permitieron identificar los tiempos de respuesta promedio, las posibles limitaciones de la infraestructura actual y áreas clave para futuras optimizaciones. Este análisis fue fundamental para garantizar que la API pueda manejar volúmenes de datos elevados y usuarios simultáneos en escenarios de alta demanda.

Figura 129.

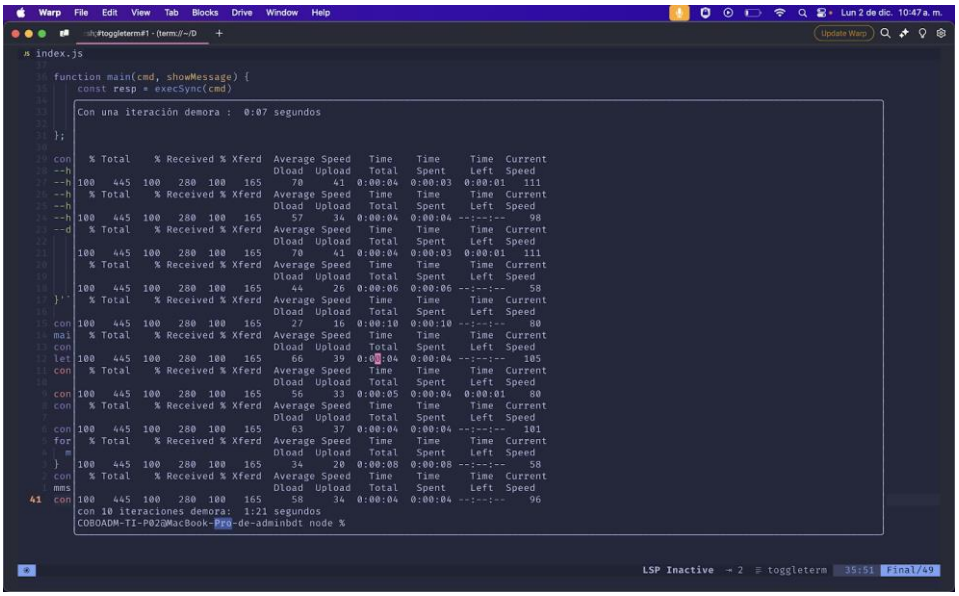
Script



- ***Tiempo de ejecución:*** Para un numero de 10 ejecuciones simultaneas, se obtuvo un tiempo de 7 segundos por ejecución como se observa a continuación.

Figura 130.

Tiempo de ejecución



3.1.10. Objetivo 3. Actividad 3: Analizar los resultados de las pruebas para identificar áreas de mejora en el rendimiento de la API, incluyendo tiempos de respuesta y eficiencia de procesamiento.

Después de ejecutar las pruebas de carga y estrés, se procedió a analizar los resultados obtenidos para identificar posibles áreas de mejora en el rendimiento de la API. Este análisis se centró principalmente en los tiempos de respuesta y en la eficiencia del procesamiento de datos bajo condiciones de alta demanda.

Se evaluaron las métricas de tiempo de respuesta promedio, así como las variaciones en los tiempos durante las iteraciones simultáneas. Se identificaron posibles limitaciones en el flujo de datos. de datos y en los procesos del backend que podrían afectar la capacidad de la API para manejar grandes volúmenes de peticiones de manera eficiente.

Basado en los resultados, se plantearon recomendaciones específicas para mejorar la escalabilidad de la API, optimizar el procesamiento de datos y reducir los tiempos de respuesta, asegurando que la API pueda soportar eficientemente futuros incrementos en el tráfico y la carga de trabajo., por esto, se hizo énfasis en una posible mejora en la lógica del servicio que comunica a la Api de DAPTA con el cliente.

Podemos observar el código a refactorizar:

Figura 131.

Código a refactorizar



```php
class ChatController extends BaseController
{
    public function comunicationChatBot()
    {
        try {
            // Obtener los datos de la solicitud
            $data = $this->request->getJsonRawBody(false);
            $user = $this->session->user;

            // Validar si existe el chatId
            if (isset($data->chatId)) {
                $chatId = $data->chatId;
            } else {
                $chatId = null;
            }

            // Obtener tipo de documento
            if (isset($user->document_type)) {
                $documentType = $user->document_type;
            } elseif (isset($user->documentTypeId)) {
                $documentType = $user->documentTypeId;
            } else {
                $documentType = 10;
            }

            // Obtener número de documento
            if (isset($user->documentNumber)) {
                $documentNumber = $user->documentNumber;
            } else {
                $documentNumber = 0;
            }

            // Validar el rol del usuario
            if (isset($data->userRol)) {
                $userRol = $data->userRol;
            } else {
                $userRol = '';
            }

            // Validar el perfil del usuario
            if (isset($data->userProfileId)) {
                $userProfileId = $data->userProfileId;
            } else {
                $userProfileId = '';
            }
```

3.1.11. Objetivo 3. Actividad 4: Implementar optimizaciones basadas en los resultados de las pruebas para mejorar el rendimiento y la escalabilidad de la API, asegurando que cumpla con los requisitos definidos.

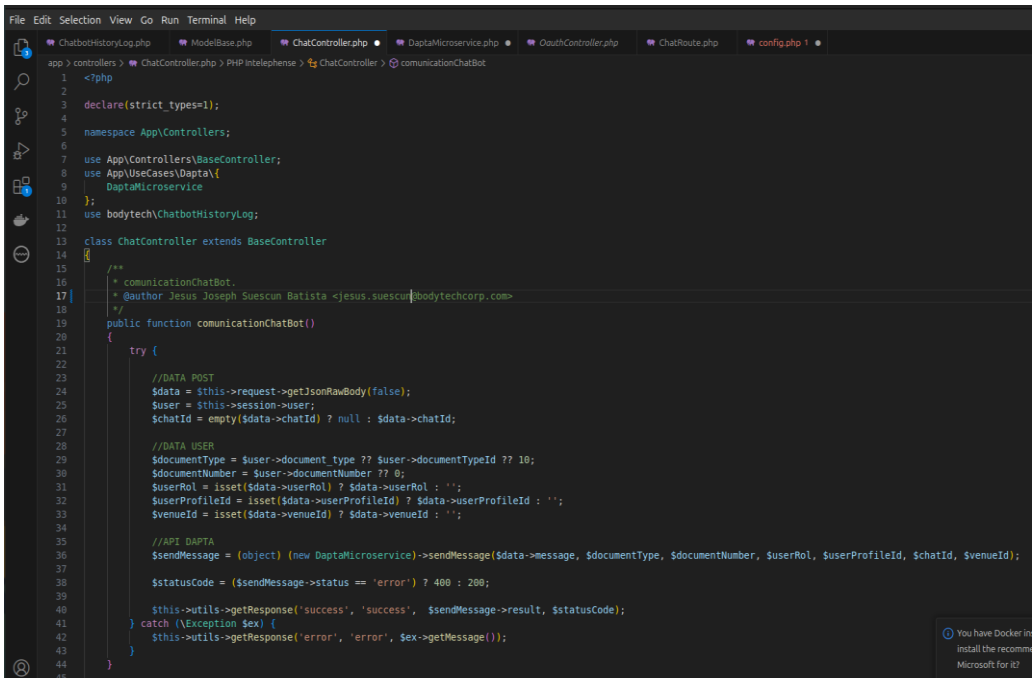
En esta actividad, se implementaron mejoras y optimizaciones en la API basadas en los resultados obtenidos durante las pruebas realizadas. Estas optimizaciones se centraron especialmente en el código del controller en Phalcon, asegurando que cumpliera con los requisitos de rendimiento y escalabilidad definidos para el proyecto.

Se refactorizar el código con el fin de mejorar su modularidad, eliminando funciones duplicadas y asegurando que cada método se enfocara en una tarea específica, facilitando tanto su mantenimiento como su lectura.

Figura 132.

Código refactorizado

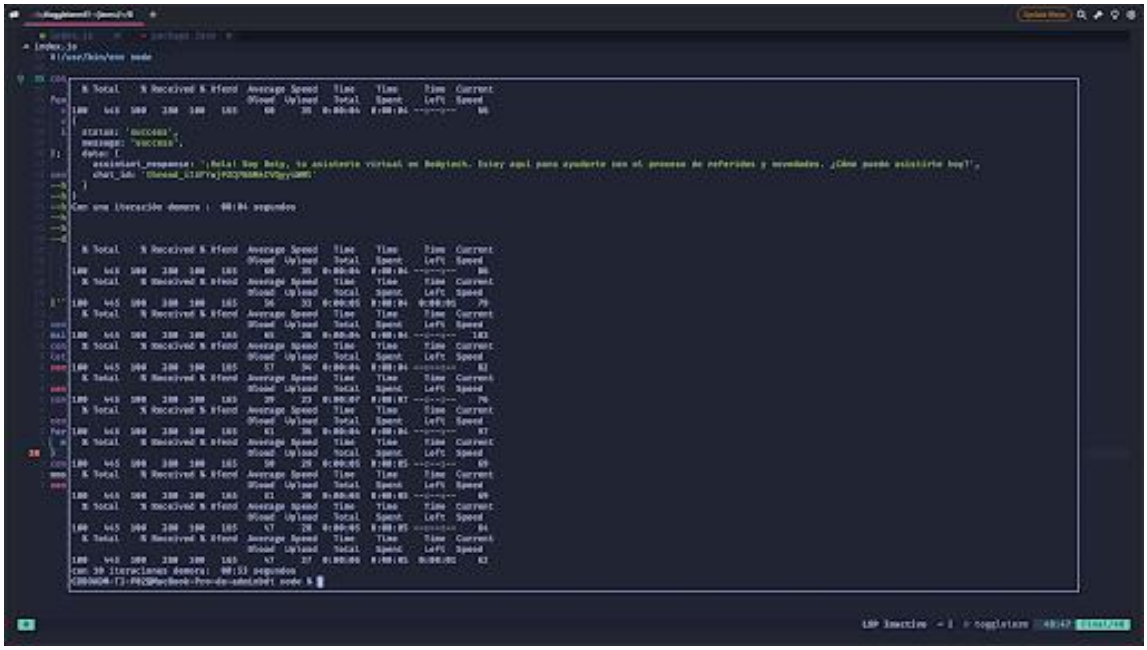
-



Luego, se volvió a ejecutar el script que ejecuta varios procesos al mismo tiempo, teniendo como resultado un mejor tiempo de ejecución de la API, mejorando de 7 segundos a 4, para el caso de 10 ejecuciones simultaneas.

Figura 133.

Evidencia mejor tiempo de ejecución.



Pruebas Finales

Luego de que el código se optimizó, se realizaron pruebas satisfactorias del funcionamiento de la API del agente en stagging(ambiente de prueba), por esta razón, se subió el proyecto al ambiente de preproducción, donde se hicieron pruebas con datos reales.

Figura 134.

Evidencia 1. Prueba satisfactoria de la API.

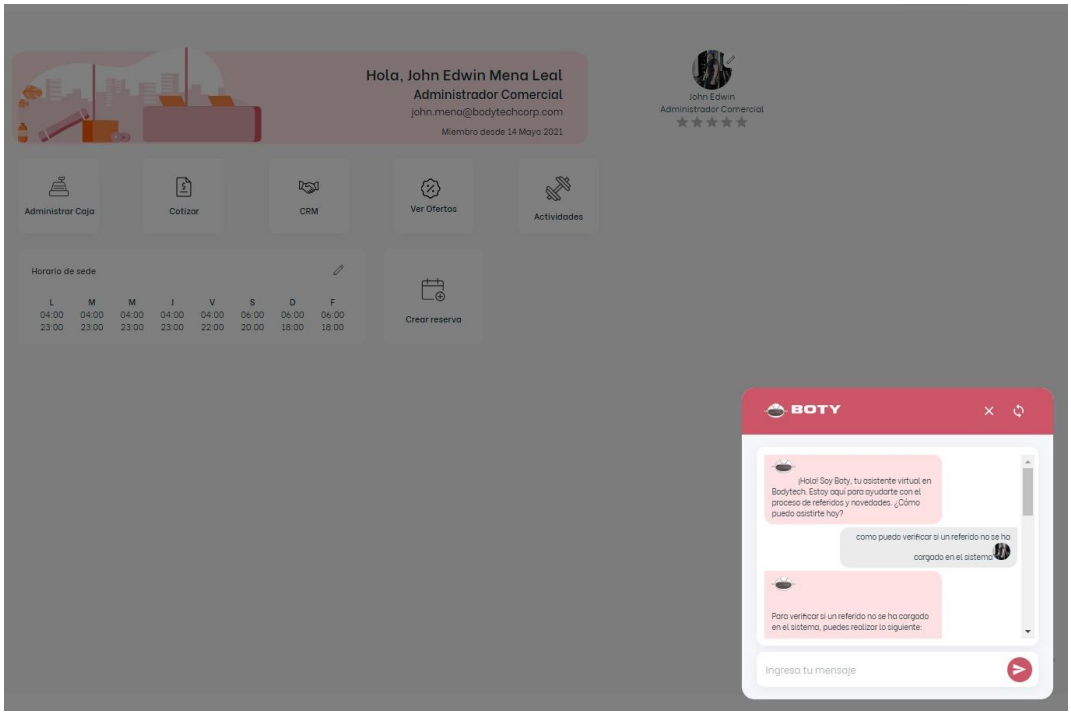


Figura 135.

Evidencia 2. Prueba satisfactoria de la API.

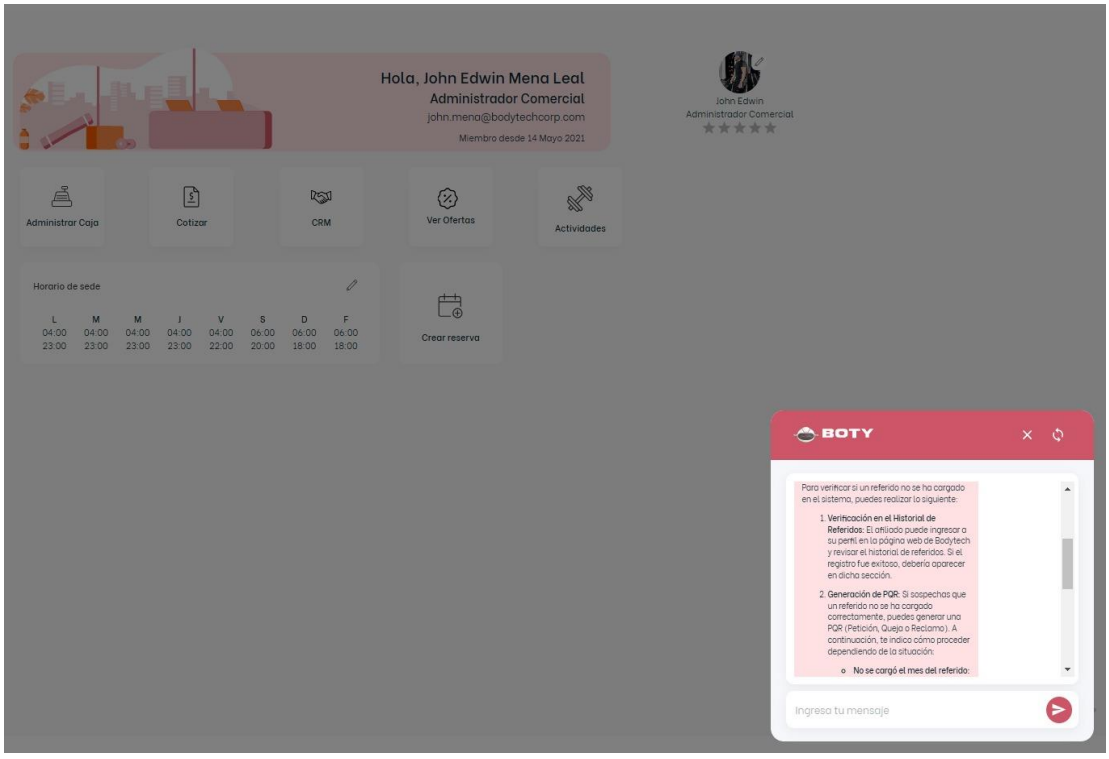
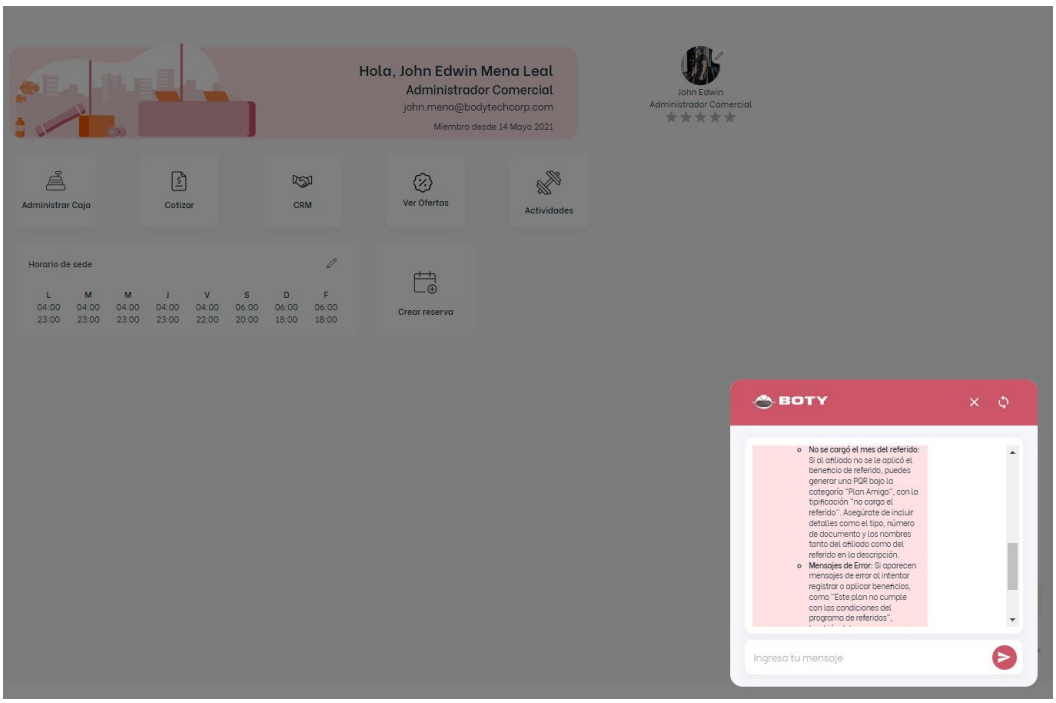


Figura 136.

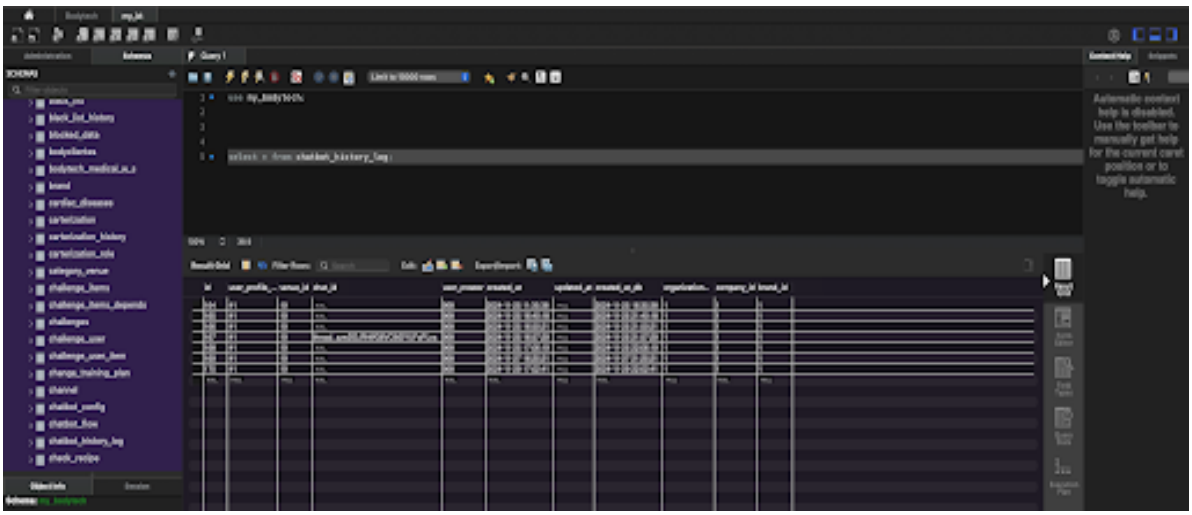
Evidencia 3. Prueba satisfactoria de la API.



El registro de la interacción del chatbot con el asesor se guardó correctamente en la base de datos.

Figura 137.

Registro de interacción del asesor



Cumpliendo con el Objetivo 4, de capacitar el área de TI de Bodytech en el uso de la API y documentar el trabajo, para aprovechar al máximo la información generada que sirva para la toma de decisiones, se desarrollaron con éxito las siguientes actividades:

3.1.12. Objetivo 4. Actividad 1: Crear materiales de capacitación detallados sobre el uso de la API.

En esta actividad, se llevó a cabo una capacitación con el equipo de producto y el equipo de desarrollo esto, con el objetivo de asegurar el uso eficiente de la API desarrollada y maximizar el aprovechamiento de la información generada para la toma de decisiones. Durante las sesiones, se explicó detalladamente el funcionamiento y la configuración de los flujos en la plataforma DAPTA, mostrando cómo se estructuran para responder a las necesidades específicas del sistema. Del mismo modo, se abordó la configuración de los agentes, haciendo énfasis en los pasos a seguir para definir su comportamiento, la personalización de sus respuestas y configuración para su correcta integración con APIs externas.

Además, se hizo énfasis en la lógica implementada para el desarrollo el backend y el frontend, explicando cómo fluye la información entre el usuario y el agente a través de la API. Esto incluyó una descripción detallada de los procesos de conexión entre el flujo de DAPTA y la API de Phalcon, así como las mejores prácticas adoptadas para garantizar una interacción fluida y eficiente.

Figura 138.

Evidencia 1. Capacitación sobre el uso de la API

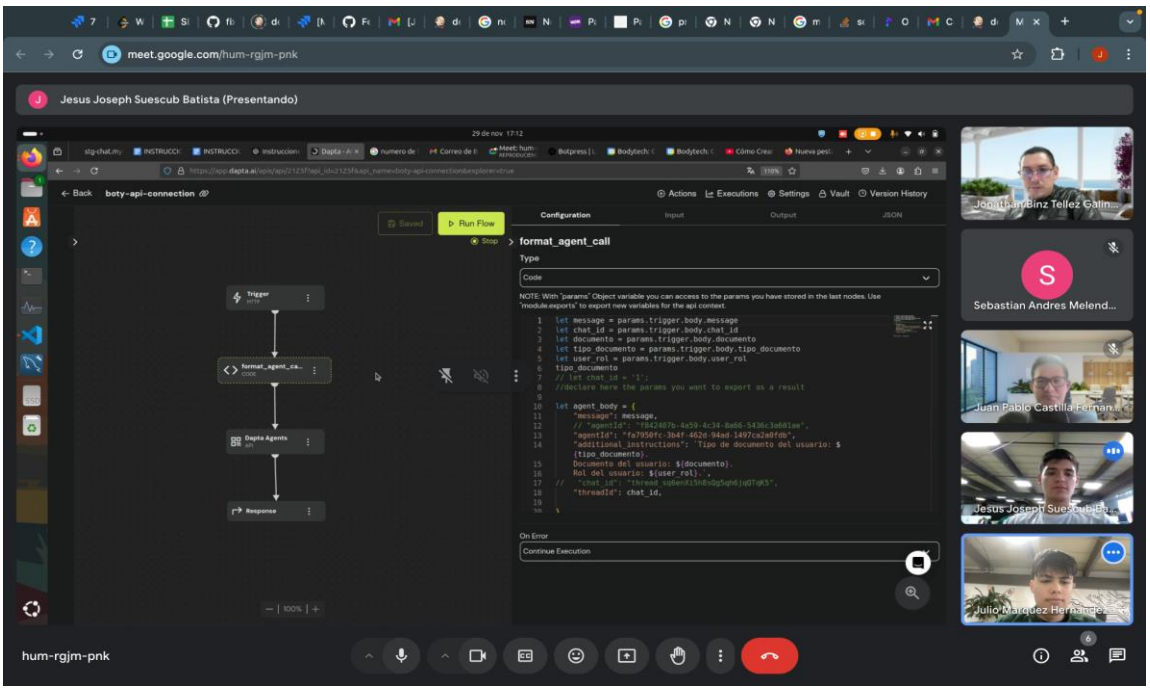
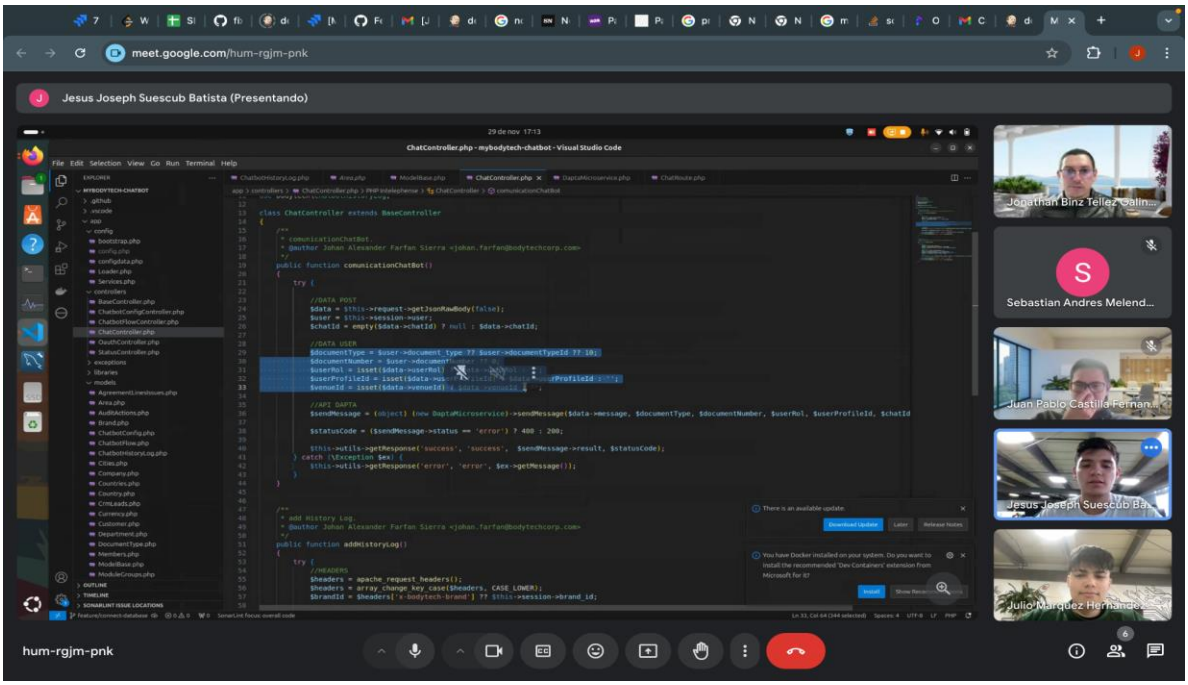


Figura 139.

Evidencia 2. Capacitación sobre el uso de la API



3.1.13. Objetivo 4. Actividad 2: Organizar talleres de capacitación, proporcionando instrucción práctica y escenarios de uso realistas para familiarizarse con la API.

En esta actividad, se organizaron talleres de capacitación diseñados para familiarizar al equipo con el uso práctico de la API, presentando su funcionalidad de una manera accesible y no técnica. Durante las sesiones, se mostró cómo la API responde a diferentes escenarios y consultas basadas en los roles de los usuarios, ejemplificando el comportamiento del sistema en situaciones realistas.

Se realizaron simulaciones en las que los participantes interactuaron directamente con la API, planteando preguntas comunes que un usuario podría hacer, para observar cómo se gestionan las respuestas y cómo se mantiene la coherencia en la interacción.

Estos talleres permitieron a los asistentes comprender mejor las capacidades de la API desde una perspectiva funcional, centrándose en su utilidad y en cómo aprovecharla para responder a necesidades operativas. Al final de las sesiones, los participantes pudieron visualizar claramente el valor práctico de la API, mejorando su confianza y preparación para integrarla en sus flujos de trabajo diarios.

Figura 140.

Evidencia 1. Talleres de capacitación

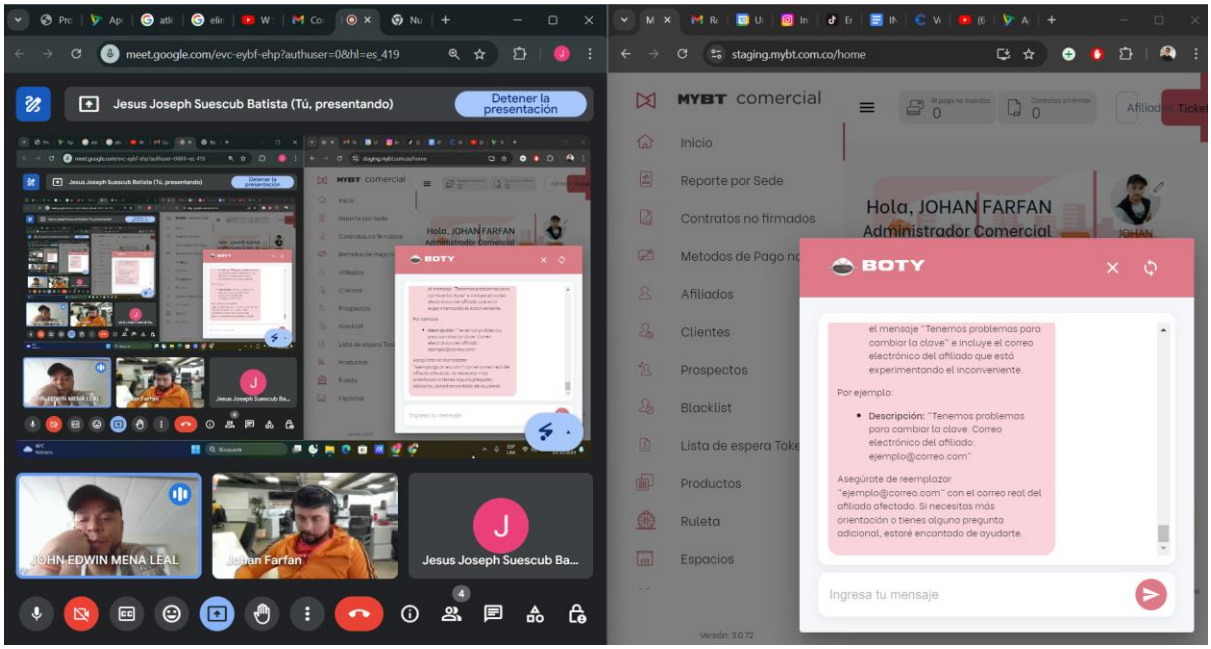
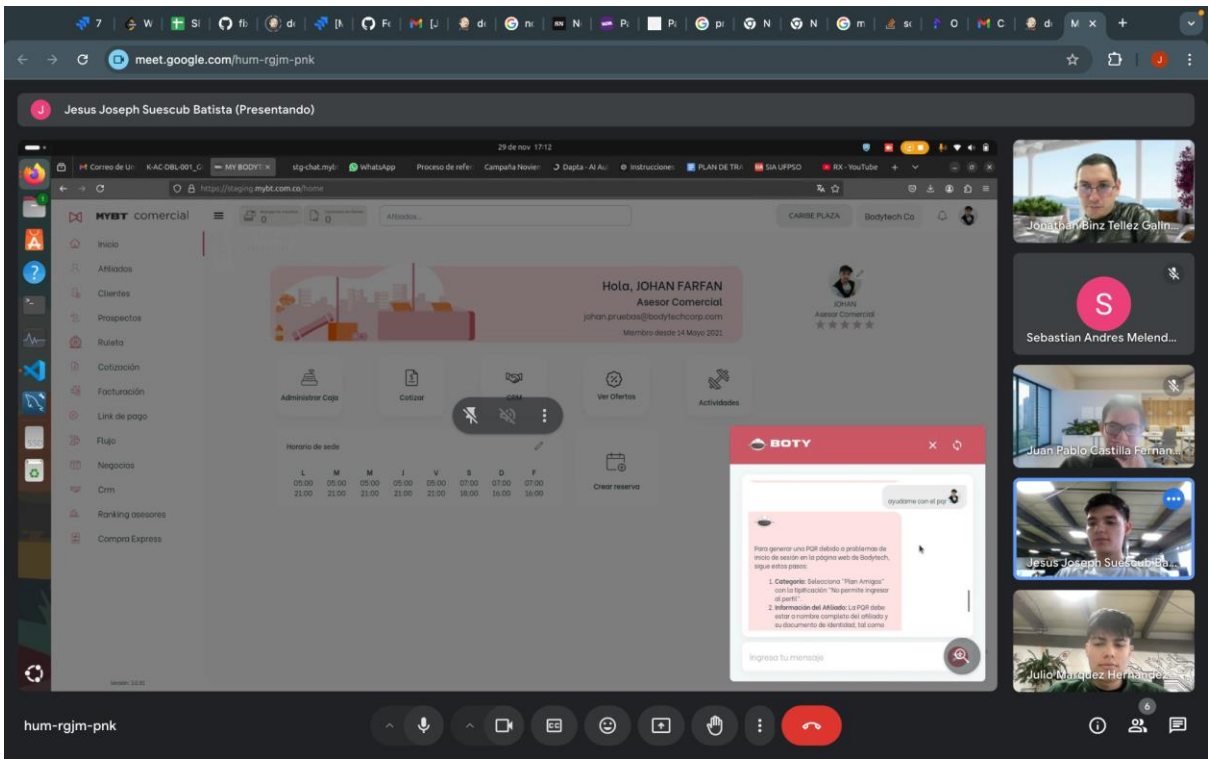


Figura 141.

Evidencia 2. Talleres de capacitación



3.1.14. Objetivo 4. Actividad 3: Documentar y entregar el trabajo realizado durante el proceso de pasantía.

Se realizó la entrega del trabajo final, el cual cuenta con el paso a paso detallado del procedimiento que se debe llevar si se quiere optimizar un proceso en determinada área de la empresa.

El trabajo cuenta con documentación clara y completa sobre cada aspecto del trabajo realizado, de manera que el equipo de TI tenga las herramientas necesarias para gestionar, mantener y ampliar las funcionalidades desarrolladas, asegurando su sostenibilidad a largo plazo.

4. Diagnóstico Final

Durante mi pasantía en Bodytech, tuve la oportunidad de desarrollar e implementar un proyecto enfocado principalmente en inteligencia artificial, el cual fue diseñado con el objetivo de mejorar la gestión operativa de la empresa. El proyecto consistió en la implementación de sistemas diseñados para automatizar especialmente procesos complejos, optimizando la ejecución de tareas claves y acortando de manera significativa los tiempos de respuesta tanto a solicitudes internas como externas.

Como parte de mi aporte, incorporé herramientas avanzadas de Inteligencia artificial las cuales hicieron más eficientes los flujos de trabajo y proporcionaron análisis más precisos sobre los datos operativos y de gestión. Esto permitió el procesamiento, la interpretación y el análisis de información para identificar patrones, optimizando de esta manera procesos internos para contemplar posibles mejoras en la toma de decisiones estratégicas. Con este enfoque, no solo se fortaleció la infraestructura tecnológica de la empresa Bodytech al proporcionar una visión más clara del rendimiento operativo y las áreas que necesitaban ajustes, sino que también se sentaron las bases para desarrollar proyectos más avanzados en inteligencia artificial y automatización, impulsando el crecimiento y la innovación de esta misma.

Por otro lado, llevé a cabo reuniones de capacitación con el equipo de tecnología, proporcionando los conocimientos fundamentales para que se pudieran manejar y aprovechar las soluciones implementadas. Dichas capacitaciones permitieron que el equipo se adaptará a los nuevos sistemas tecnológicos, asegurando su uso correctamente.

También, trabajé en el diseño y desarrollo de flujos de trabajos organizados que facilitaron la integración de manera eficiente de las nuevas herramientas implementadas. Estas contribuciones no sólo fortalecieron la infraestructura tecnológica de la empresa, sino que también promovieron cambios hacia la adopción de nuevas tecnologías como la inteligencia artificial, logrando resultados positivos y medibles en la productividad y la experiencia del cliente.

5. Conclusiones

A lo largo de mi experiencia como pasante en la empresa, he podido implementar un proyecto de inteligencia artificial que optimizó diversos procesos operativos y mejoró la eficiencia general en la toma de decisiones. La implementación de tecnologías avanzadas como la IA permitió automatizar tareas complejas, reducir tiempos de respuesta y proporcionar análisis más precisos de los datos operativos. Este proyecto no solo fortaleció la infraestructura tecnológica de la empresa, sino que también sentó las bases para futuras implementaciones, permitiendo a la organización adaptarse rápidamente a las nuevas demandas y necesidades del mercado.

Además, la capacitación proporcionada al equipo interno y la creación de flujos de trabajo estructurados garantizaron que las nuevas herramientas fueran adoptadas de manera fluida, asegurando su sostenibilidad a largo plazo. La integración de soluciones basadas en IA representó una parte fundamental de la evolución tecnológica de la empresa, permitiendo no solo mejorar la eficiencia operativa, sino también abrir el camino para la adopción de nuevas tecnologías en el futuro. En resumen, este proyecto contribuyó significativamente al fortalecimiento de la capacidad de la empresa para mantenerse competitiva y preparada para los retos tecnológicos venideros.

6. Recomendaciones

Es fundamental que DAPTA implemente mejoras en su plataforma para optimizar el rendimiento de los flujos, ya que, a pesar de que los flujos desarrollados durante el proyecto están estructurados de manera eficiente y el código backend está correctamente optimizado, se observaron demoras en las respuestas del agente en ciertas ocasiones. Estas demoras están relacionadas con el tiempo que toma el flujo de DAPTA en procesar las solicitudes y responder adecuadamente.

Para abordar este problema, se recomienda que DAPTA optimice la ejecución de sus nodos y mejore la eficiencia de los flujos de conexión. Una posible solución sería implementar técnicas de optimización en nodos críticos, como el procesamiento de consultas hacia APIs externas. Esto es especialmente relevante en casos como la integración del agente con la API de Phalcon, que se utiliza para conectar al cliente con el agente. La optimización de estos flujos permitiría reducir significativamente los tiempos de respuesta, mejorando la experiencia del usuario y garantizando interacciones más fluidas y eficientes.

Adicionalmente, se sugiere que DAPTA evalúe el rendimiento de su infraestructura y explore la posibilidad de mejorar la escalabilidad de su sistema, para garantizar un procesamiento más rápido y robusto en escenarios de alta carga. Estas mejoras no solo beneficiarían el desempeño actual, sino que también fortalecerían la plataforma para futuros desarrollos e integraciones.

7. Referencias

Acosta, B. S., & Casas, M. M. (2014). *Programa Administración de Negocios. Trabajo Final Integrador.* Armenia: Universidad del Quindío.

Agarwal, C., Gurumurthy, R., & Devansh, U. (2023). *¿Qué hace un analista de producto?* Obtenido de https://www.linkedin.com/advice/0/what-does-product-analyst-do-skills-business-analysis-7anoe?lang=es&originalSubdomain=es

Alexander, A. D. (2024). *Roles y responsabilidades del equipo de inteligencia empresarial: desvelación de puestos y deberes clave.*

Aplyca, D. W. (07 de 12 de 2022). *Integración continua y entrega continua (CI/CD).* Obtenido de https://www.aplyca.com/blog/integracion-continua-y-entrega-continua-cicd

AWS Documentation, a. (2024). *IA Generativa.* Obtenido de ¿Qué es la IA generativa?: https://aws.amazon.com/es/what-is/generative-ai/

AWS, A. D. (2024). *AWS Documento técnico.* Obtenido de Descripción general de Amazon Web: https://docs.aws.amazon.com/es_es/whitepapers/latest/aws-overview/aws-overview.pdf

Bodytech Introduction, c. (29 de Julio de 2024). Conocimiento de Bodytech. Conversación con el Gerente de desarrollo de la empresa Bodytech. Bogotá, Colombia.

BodyTech, c. (2022). *BODYTECH.* Obtenido de Nuestra historia: https://bodytech.com.co/nuestra-historia

BodyTech, M. (2022). *Manual de Peevención del Riesgo de Corrupción, Opacidad y Fraude - SICOF.* Bogotá, Colombia.

Camargo, J. C., Amaya, J. E., & Arias, S. A. (2023). *Diseño de una Propuesta Estratégica para el Posicionamiento de Marca Basado en la Competencia para una empresa del*

sector fitness en Colombia. Bogotá, Colombia: Universidad EAN- URI: http://hdl.handle.net/10882/12744.

Carreño Hernandez, J. M. (2019). Ingeniero de Sistemas e Informática. *Inclusive - Aplicacion web de soporte a estrategias didacticas de formacion para población con discapacidad visual segun el proyecto de investigacion adscrito a DIT con Código 0002-071-86400.* Universidad Pontificia Bolivariana, Bucaramanga.

Clark, H. (2024). Product Management: Roles and Responsibilities Through the Career Timeline. *The Product Manager*, 1-21.

Consulting Gallagher, B. S. (2022). *Manual del Sistema de Administración del Riesgo de Lavado de Activos de la Financiación del Terrorismo y Financiación de la Proliferación de Armas de Destrucción Masiva – SARLAFT / PADDM.* Bogotá, Colombia.

CVZEN, B. (2024). *Analista de producto.* Obtenido de https://cvzen.es/blog/analista-de-producto

Dapta Documentation, a. (2024). *Dapta.* Obtenido de Genera contenido de alta calidad con Inteligencia Artificial: https://www.dapta.ai/es-co/ai-content-automation

Estayno, M. G., & Meles, J. (2014). El desafío de ser un Product Owner. *Red de Universidades con Carreras en Informática (RedUNCI) - http://sedici.unlp.edu.ar/handle/10915/42283*.

Fernandez, O. (14 de 07 de 2024). *Aprender BIG DATA.* Obtenido de MySQL: Un vistazo a su arquitectura y cómo utilizarlo en Big Data: https://aprenderbigdata.com/mysql/

Fernandez, O. (14 de 07 de 2024). *Aprender BIG DATA.* Obtenido de Introducción a Redis: Base de datos en Memoria: https://aprenderbigdata.com/redis/

G. Ann, C., & Patroklos P, P. (2013). SonarQube en acción. En C. G. Ann, & P. Patroklos P, *SonarQube en acción* (pág. 392). Manning Publications Co., Estados Unidos.

García, J., Fernández, J., & Bernal, A. (2014). Perceived quality and loyalty in low-cost fitness . *Suma Psicol. vol.21 no.2* , 123-130.

Henao, A. (2023). *Platzi.* Obtenido de DAPTA | Plataforma no-code para Backend y APIs: https://platzi.com/blog/dapta/

Hernández, E., & Beltrán, C. (2020). A practical agile methodology approach to software engineering. *Revista Tecnología, Investigación y Academia (TIA) Red Avanzada RITA. ISSN: 23448288, 8 (2)*, 61-73.

Holmes, E., & Bray, T. (2015). Getting Started with React Native. . En *Getting Started with React Native.* . Birmingham: Packt.

IBM. (2024). *¿Qué es el prompt engineering ?* Obtenido de https://www.ibm.com/es-es/topics/prompt-engineering

IBM Documentation, m. (2024). *IBM.* Obtenido de ¿Qué es Redis?: https://www.ibm.com/mx-es/topics/redis

Jimenez Villar, J. (2016). Descubre REACT. En J. Jimenez Villar, *Descubre REACT* (págs. 8-9). Leanpub.

Knowledge21, B. (20 de 08 de 2020). Obtenido de ¿Qué es el Equipo de Desarrollo del Scrum?: https://es.k21.global/blog/equipo-de-desarrollo-del-scrum

Lazcano Calixto, R. N., Valencia González, L. Á., & Baena Díaz, D. (2019). React Native: acortando las distancias entre desarrollo y diseño móvil multiplataforma. *Revista Digital Universitaria (rdu). Doi: http://doi.org/10.22201*, Vol. 20, núm. 5 .

Management Modus, e. (2023). *Modus Management* . Obtenido de Integración de Datos y Aplicaciones: https://www.modus.es/que-es-jenkins/

Mercedes Brito, J. C. (2022). Ingeniero de tecnologias y Servicios de Telecomunicación. *Integración y entrega continua (CI/CD) con Jenkins.* Universitat Oberta de catalunya, España.

Ministerio de Tecnologías de la Información y las Comunicaciones, M. (2018). Modelo de Seguridad.

MinTIC, M. d. (2018). Decreto 1008 De 2018. Diario Oficial No. 50.673, 7.

Navarro, S. (24 de 09 de 2024). *KeepCoding Bootcamps.* Obtenido de https://keepcoding.io/blog/que-es-dapta-y-sus-beneficios/

Oracle, O. (2024). *¿Qué es un administrador de base de datos (DBA)?* Obtenido de https://www.oracle.com/es/database/

Pérez Arbesú, P. M. (18 de 11 de 2022). *Computer Weekly.* Obtenido de El organigrama perfecto para un departamento de TI: https://www.computerweekly.com/es/opinion/El-organigrama-perfecto-para-un-departamento-de-TI

Phalcon Documentation, i. (2012). *Phalcon.* Obtenido de Phalcon Framework: un marco PHP creado como una extensión para PHP: https://docs.phalcon.io/latest/introduction/

Politica de tratamiento de datos Bodytech, c. (2024). *Bodytech.* Obtenido de Aviso de privacidad: https://bodytech.com.co/poltica-de-tratamiento-de-datos-personales

React Documentation, U. (2024). *Aprende React.* Obtenido de https://es.react.dev/learn/describing-the-ui

Rodriguez P, D. (2023). Bodytech y Smart Fit se posicionan como las cadenas de gimnasios más populares. *La República - Sección Deporte*, 1-4.

Rojas, N. C. (2024). Revista Forbes. *Forbes*. Obtenido de Dapta, la plataforma que facilita la integración de APIs e inteligencia artificial: https://forbes.co/2024/03/08/emprendedores/dapta

Socorras, R. (29 de Julio de 2024). Conocimiento de la organización Bodytech. (J. Suescuen, Entrevistador)

Términos Y Condiciones De La Prestación Del Servicio | Bodytech, E. m. (2022). *Términos Y Y Condiciones De La Prestación Del Servicio.* Obtenido de https://bodytech.com.co/terminos-y-condiciones

Universidad Nacional de Colombia, U. (21 de 05 de 2022). *Software Libre y Propietario.* Obtenido de MySQL: https://fce.unal.edu.co/unidad-de-informatica/proyectos-de-estudio/ejes-tematicos-transversales/software-libre-y-propietario/mysql

Velasco, M. V., Nuñez, J., Saltos, P., & Cunuhay, W. C. (2021). Systematic review of the SCRUM methodology for software development. *Revista Cientifica Domino de las Ciencias. DOI: http://dx.doi.org/10.23857/dc.v7i4.2429. ISSN: 2477-8818. Vol. 7, núm. 4*, 434-447.

Vilela, C. (23 de 04 de 2023). *Freelancermap.* Obtenido de ¿Qué hace el soporte TI o soporte técnico?

Yahuasi Apaza, E. (2020). Licenciatura en Ingeniería de Sistemas. *SISTEMA DE INFORMACIÓN WEB DE CONTROL ADMINISTRATIVO - CASO: "GIMNASIO WINNER".* Universidad Pública del Alto, Perú.