



17 DE JUNIO DE 2021

# SERVICIOS Y APLICACIONES DE TYPEME

DOCUMENTACIÓN

JUÁREZ GARCÍA ÁNGEL DE JESÚS | GUADARRAMA CHÁVEZ SAMMY  
DESARROLLO DE SISTEMAS EN RED Y DESARROLLO DE APLICACIONES  
BAEZ HERRERA SAMUEL | LÓPEZ HERRERA JUAN LUIS

# Tabla de contenido

<b>INTRODUCCIÓN</b>	<b>4</b>
<b>REQUERIMIENTOS</b>	<b>5</b>
<b>Contexto</b>	<b>5</b>
<b>Clases de usuario</b>	<b>6</b>
Administrador	6
Typer	6
<b>Casos de uso (análisis)</b>	<b>6</b>
<b>Prototipo de UI</b>	<b>7</b>
<b>Requerimientos funcionales</b>	<b>12</b>
<b>Requerimientos no funcionales</b>	<b>13</b>
<b>DISEÑO</b>	<b>14</b>
<b>Diseño arquitectónico</b>	<b>14</b>
Vistas de casos de uso	14
Vista de implementación	16
Vista de procesos	19
Vista de despliegue	22
Modelo de datos	23
Descripciones de casos de uso	26
Administrador	26
Typer	30
<b>CONSTRUCCIÓN</b>	<b>44</b>
<b>Selección justificada pila tecnológica</b>	<b>44</b>
Aplicación de cliente rico	44
Aplicación de servicios (API)	45
<b>Estándares de codificación</b>	<b>47</b>
Convención de nombres	47
Patrones de escritura de código	47
<b>Reportes de análisis estático de código</b>	<b>50</b>
<b>Prácticas de construcción realizadas</b>	<b>54</b>
<b>PRUEBAS</b>	<b>56</b>

<b>Plan de pruebas para la Aplicación de servicios (API)</b>	<b>56</b>
Proyecto	56
Elementos de prueba	56
Alcance de la prueba	57
Requisitos de datos de prueba	58
Requisitos de entorno de prueba	58
Roles, actividades y responsabilidades	59
<b>Procedimiento de prueba</b>	<b>60</b>
<b>Casos de prueba</b>	<b>61</b>
Métodos GET	61
<b>Métodos POST</b>	68
Métodos PUT	84
<b>Resultados</b>	<b>91</b>
Resultados métodos GET	91
Resultados métodos POST	97
Resultados métodos PUT	113
<b>ESTRATEGIA DE DESPLIEGUE</b>	<b>120</b>
<b>CONCLUSIONES</b>	<b>122</b>
<b>REFERENCIAS</b>	<b>123</b>

## Introducción

En este documento se encuentra la documentación acerca del sistema TypeMe la cual es una aplicación de chat utilizando un enfoque de sistema distribuido. Esta aplicación esta realizada en lenguajes como C#, JavaScript y Python. En esta aplicación los usuarios o Typers son capaces de comunicarse a través de los distintos clientes que existen de una manera asíncrona y todos sus datos son guardados en las múltiples bases de datos que se manejan.

El proyecto hace uso de una arquitectura basada en microservicios la cual ofrece la ventaja del encapsulamiento de los comportamientos en distintos módulos para de esta manera cumplir con una alta cohesión y un bajo acoplamiento.

El documento comienza presentando los requerimientos recolectados en el proceso de elicitación, aquí se presentan las clases de usuario, funciones básicas y primeros prototipos de la aplicación con los cuales se forman los requerimientos funcionales y no funcionales que son la base de la aplicación.

Seguido de esto se comienza la sección de diseño, donde se realiza una vista de los casos de uso identificados hasta el momento y que clase de usuarios van a hacer uso de estas, a esto le sigue una vista de despliegue mostrando la manera en la que la aplicación va a funcionar y por ultimo los diseños de las bases de datos que se realizaron para almacenar la información.

La siguiente sección es la de construcción en la cual se enlistan todas las tecnologías usadas a detalle y se justifica el uso que se le dio a a cada una de estas, además de mostrar las distintas prácticas y recursos que se utilizaron para llevar a cabo el desarrollo de la aplicación y mostrando por último análisis estáticos de los distintos módulos realizados verificando la calidad de construcción de estos.

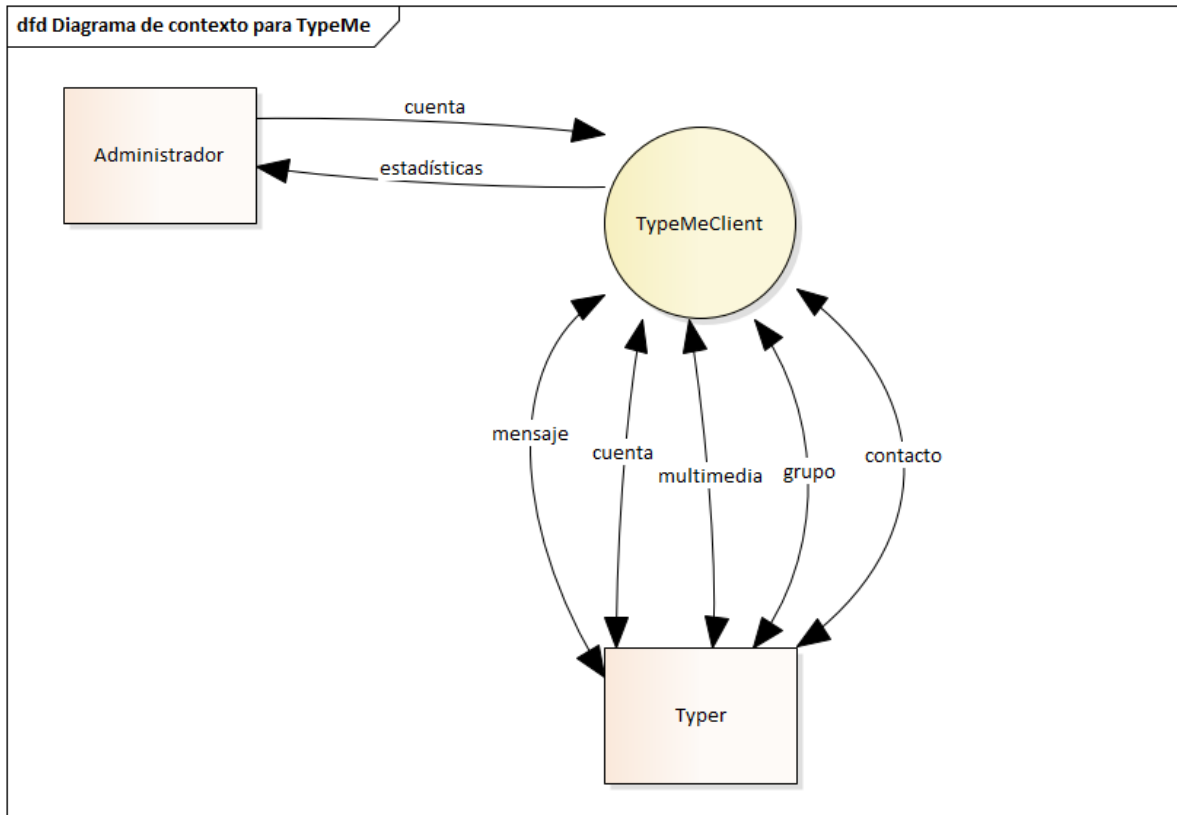
Por último, se incluye una sección de pruebas en la que se muestra el proceso y practicas seguidas para realizar las pruebas sobre el software creado, en esta sección se enlistan los múltiples casos de prueba diseñados y sus condiciones para ejecutarse para después mostrar el resultado obtenido por cada uno de estos casos.

Para cerrar el documento se especifica como se conforma el funcionamiento de la aplicación durante la ejecución y el cómo cada uno de los módulos están conectados entre si. La documentación pretende ser una guía sobre el estado del sistema y todo lo que se tuvo en cuenta para llegar a ese estado, de esta manera los lectores obtendrán familiaridad con la aplicación TypeMe y podrán realizar modificaciones de manera segura sabiendo el objetivo y funcionamiento de cada uno de los módulos creados.

# Requerimientos

## Contexto

En la siguiente Figura se muestra el diagrama de contexto para la aplicación TypeMe. Donde en el centro podemos ver TypeMeClient, haciendo referencia a todos los clientes que se desarrollaron para TypeMe, y a los extremos, los principales actores y los datos que proporcionan a la aplicación y los datos que ella proporciona a ellos.



## Clases de usuario

Durante el análisis de los requerimientos para una aplicación de chat se han determinado las siguientes clases de usuario:

### Administrador

El administrador es la clase de usuario que está encargada de monitorear la actividad que sucede en la aplicación ya que con esto tiene acceso a todas las estadísticas necesarias para saber el estado de la aplicación en cuanto a distintas métricas de interés.

### Typer

El *Typer* es la clase de usuario la cual interactúa con el sistema en todos sus casos de uso principales, este puede enviar mensajes y manejar la lista de sus contactos, el *Typer* tendrá acceso a todas las funcionalidades de la aplicación por medio de uno de los clientes que se programen.

## Casos de uso (análisis)

Después de haber analizado los requerimientos el equipo llego como conclusión a una lista de casos de uso que deberían estar dentro de la aplicación los cuales son:

### Administrador

- CU-A-01: Ver estadísticas de usuarios registrados
- CU-A-02: Bloquear usuario

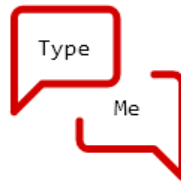
### Typer

- CU-T-01: Registrar cuenta
- CU-T-02: Enviar mensaje
- CU-T-03: Enviar archivos multimedia
- CU-T-04: Agregar contacto
- CU-T-05: Editar perfil
- CU-T-06: Eliminar historial de mensajes
- CU-T-07: Crear grupo
- CU-T-08: Eliminar contacto
- CU-T-09: Bloquear contacto

## Prototipo de UI

En esta sección se presentan los prototipos que se tomaron como base para la realización de la interfaz gráfica de los clientes.

A continuación, se muestra el prototipo de la ventana login de la aplicación



Usuario/Correo electrónico

Contraseña

Iniciar sesión

Crear cuenta

A continuación, se muestra el prototipo de la ventana de registro de la aplicación

## Registra tu cuenta

Ingresa los datos para crear una nueva cuenta

Usuario/Correo electrónico

Contraseña

Conforma tu contraseña

Crear cuenta

Cancelar



A continuación, se muestra el prototipo de la ventana para agregar un contacto en la aplicación

## Agregar contacto

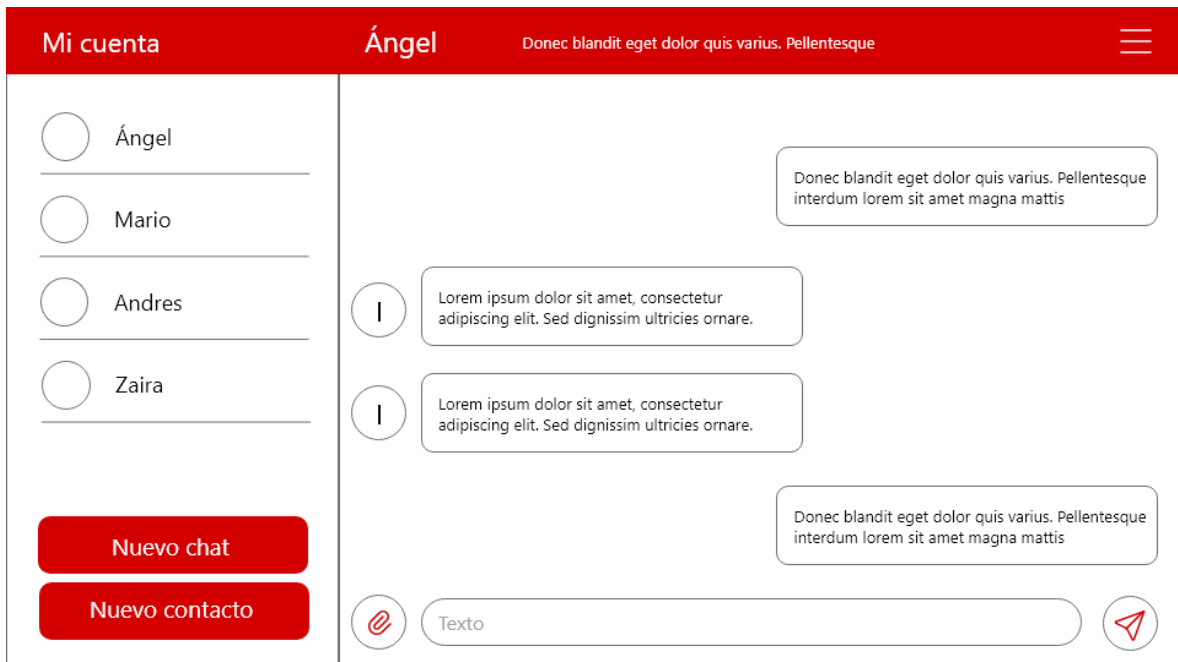
Ingresa el nombre del usuario o correo de la persona que quieres agregar

Usuario/Correo electrónico

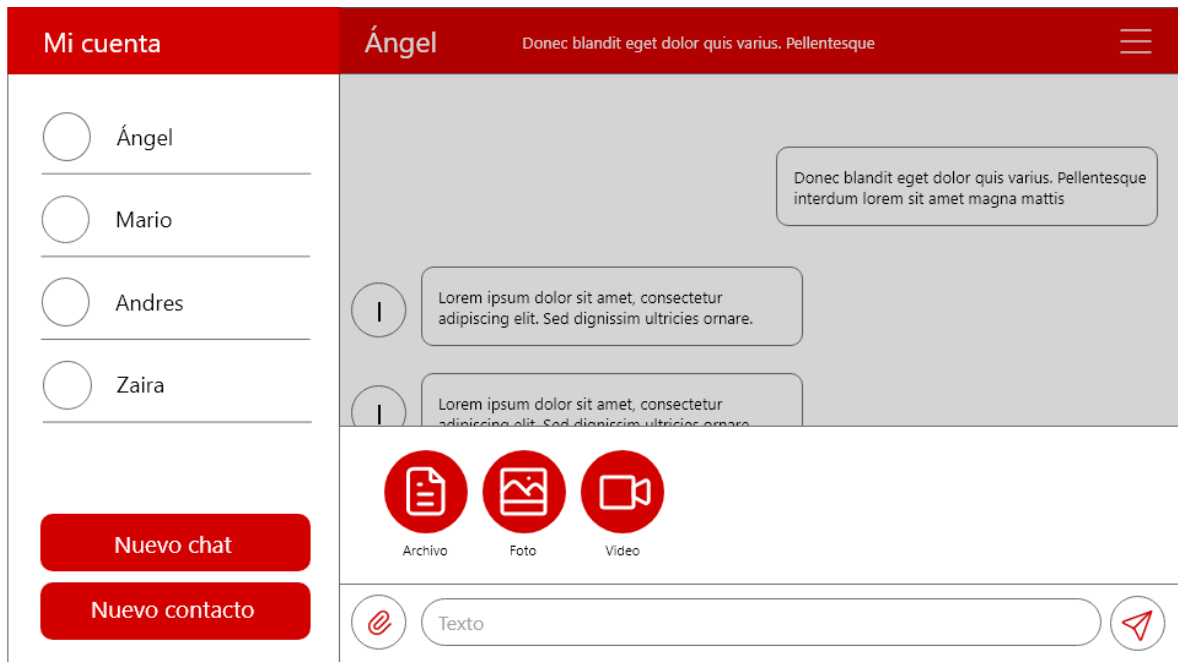
Agregar contacto

Cancelar

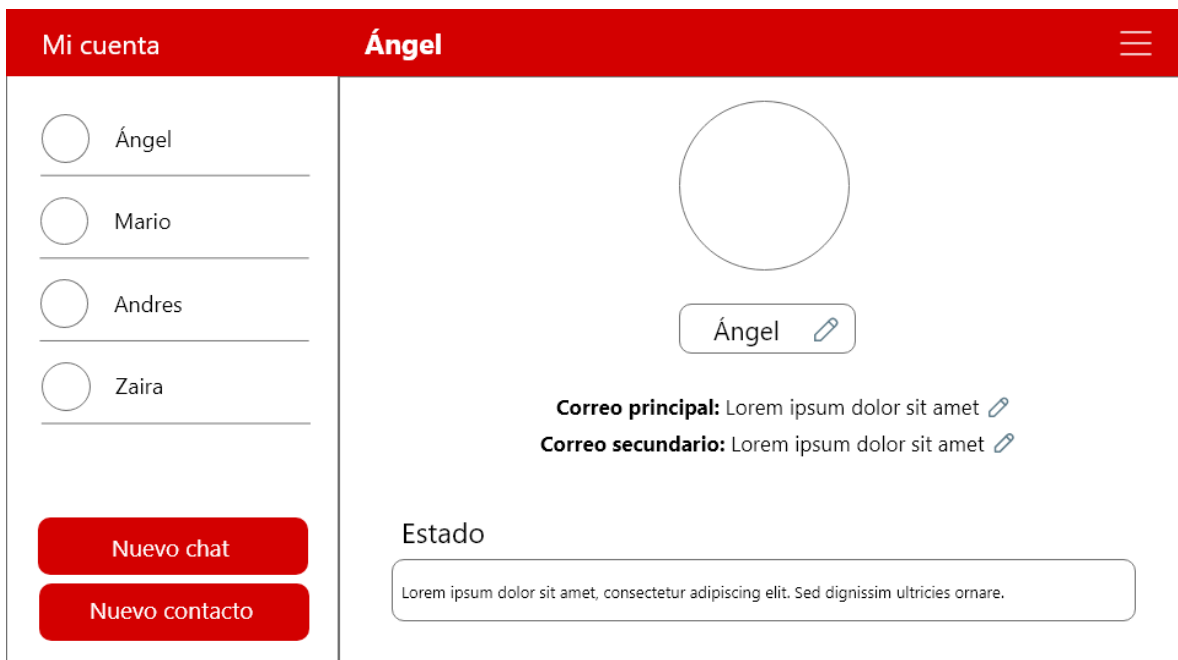
A continuación, se muestra el prototipo de la ventana principal de la aplicación



A continuación, se muestra el prototipo del envío de multimedia dentro de la aplicación



A continuación, se muestra el prototipo de la ventana "Mi perfil" dentro de la aplicación



## Requerimientos funcionales

A continuación, se muestra el listado de los requerimientos funcionales (RF) identificados para la aplicación TypeMe.

ID	Requerimiento
RF-01	La aplicación TypeMe debe de mostrar las estadísticas correspondientes a los usuarios que hay en el sistema, numero de mensajes enviados, usuarios bloqueados.
RF-02	La aplicación TypeMe debe permitir al Administrador bloquear usuarios de la aplicación.
RF-03	La aplicación TypeMe debe de realizar una autenticación de cuenta por medio de correo electrónico al realizar un nuevo registro.
RF-04	La aplicación TypeMe debe mostrar las conversaciones más recientes en la lista de chats al entrar a la aplicación.
RF-05	La aplicación TypeMe debe permitir el envío de mensajes entre contactos agregados.
RF-06	La aplicación TypeMe debe de permitir el envío de imagenes.
RF-07	La aplicación TypeMe debe permitir al Typer editar su perfil de usuario.
RF-08	La aplicación TypeMe debe permitir que el Typer agregue nuevos contactos a través del nombre de usuario de ellos.
RF-09	La aplicación TypeMe debe de permitir que el Typer elimine contactos agregados.
RF-10	La aplicación TypeMe debe de permitirle al Typer bloquear a un contacto en específico.
RF-11	La aplicación TypeMe debe de permitirle al Typer iniciar una conversación con un contacto agregado.
RF-12	Si ocurre un problema de conexión, un error de datos o una acción es realizada de manera correcta, la aplicación TypeMe debe mostrar una ventana notificando sobre el problema o sobre la acción realizada.
RF-13	La aplicación no debe permitir el registro de una nueva cuenta cuando el username coincida con una cuenta ya registrada.

## Requerimientos no funcionales

Ahora, se muestra el listado de los requerimientos no funcionales (RNF) identificados.

ID	Requerimiento
RNF-01	La aplicación TypeMe debe de estar disponible las 24 horas del día de toda la semana.
RNF-02	La aplicación TypeMe debe de ofrecer una fiabilidad de 90% de su funcionamiento.
RNF-03	La aplicación TypeMe debe de solicitar una contraseña para acceder a la cuenta del Typer.
RNF-04	La aplicación TypeMe debe encriptar la contraseña ingresada cuando el Typer crea una nueva cuenta.
RNF-05	El Typer debe de poder navegar por la aplicación en los primeros 5 minutos de entrar en ella.
RNF-06	La aplicación Typme debe de mostrar claramente las opciones de envío en la pantalla.
RNF-07	La aplicación TypeMe debe mostrar una apariencia agradable para el usuario.
RNF-08	La aplicación TypeMe debe de estar construido siguiendo un estándar de programación para su fácil mantenimiento.
RNF-09	La aplicación TypeMe debe de estar dividido en componentes que permitan su fácil integración y mantenimiento.
RNF-10	Los componentes visuales deben de poder ser reutilizados en distintos lugares de la aplicación TypeMe.

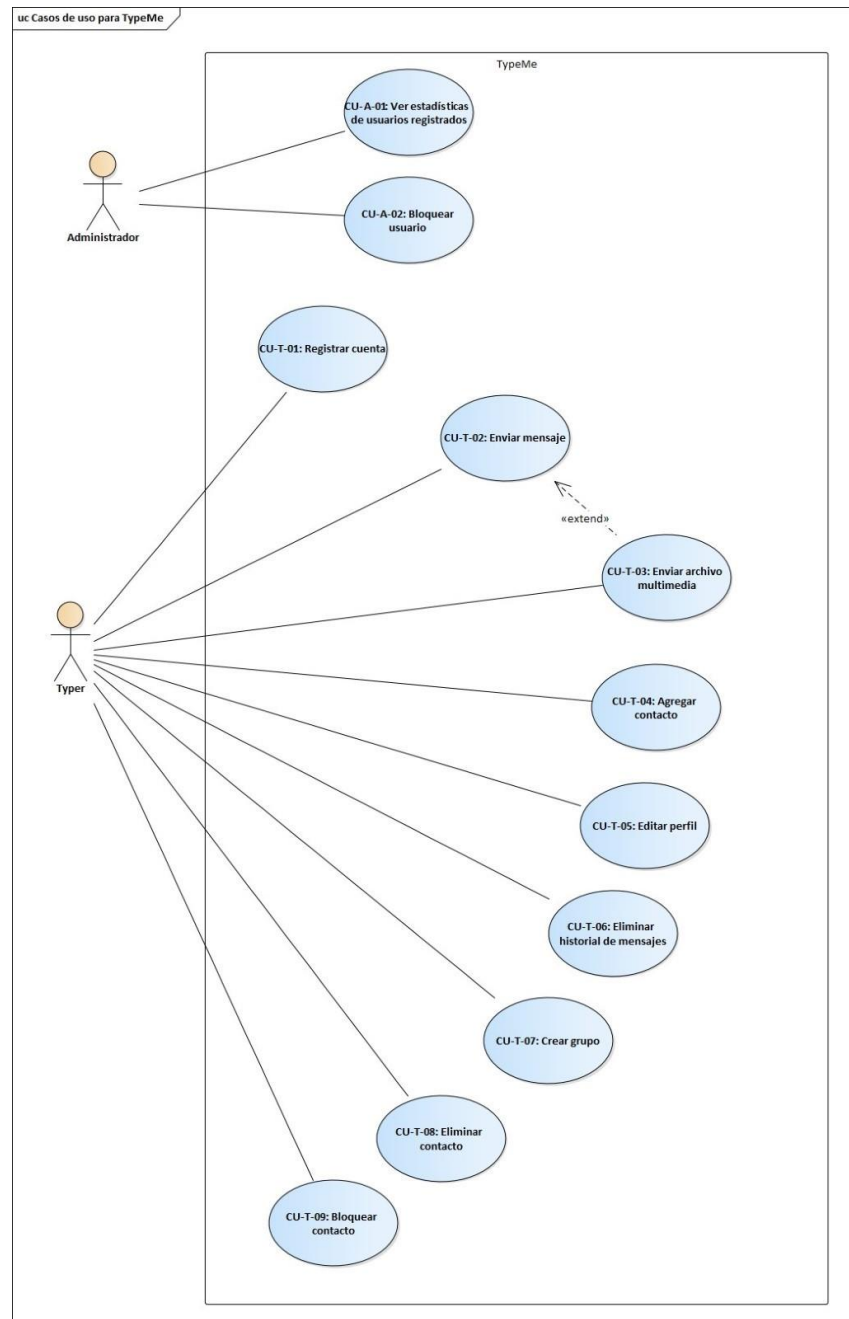
## Diseño

### Diseño arquitectónico

En esta sección se presentan los principales diagramas para definir el diseño de la arquitectura de la aplicación de TypeMe, tanto por el lado de los microservicios, como del lado del cliente.

### Vistas de casos de uso

En la siguiente Figura se muestra el diagrama de casos de uso, los cuales representan las principales funcionalidades de TypeMe en todos sus clientes.

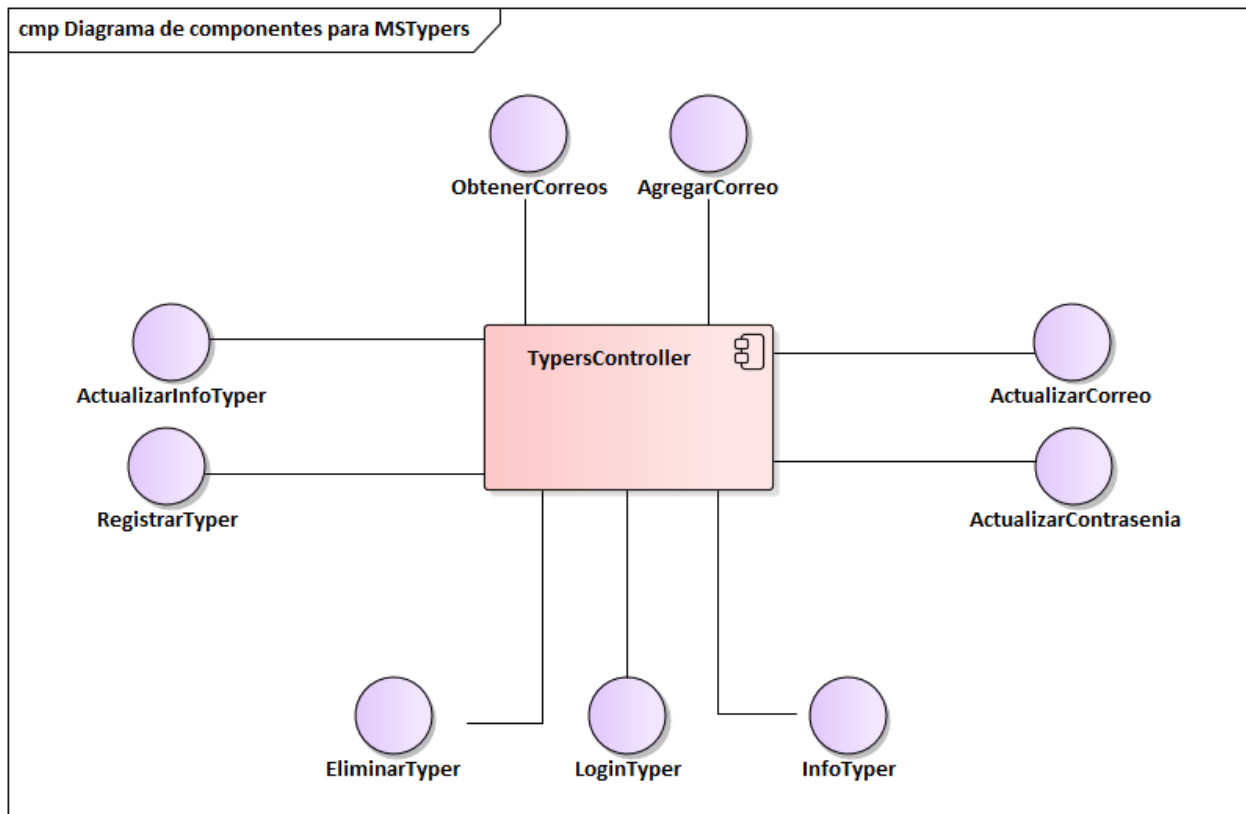




### Vista de implementación

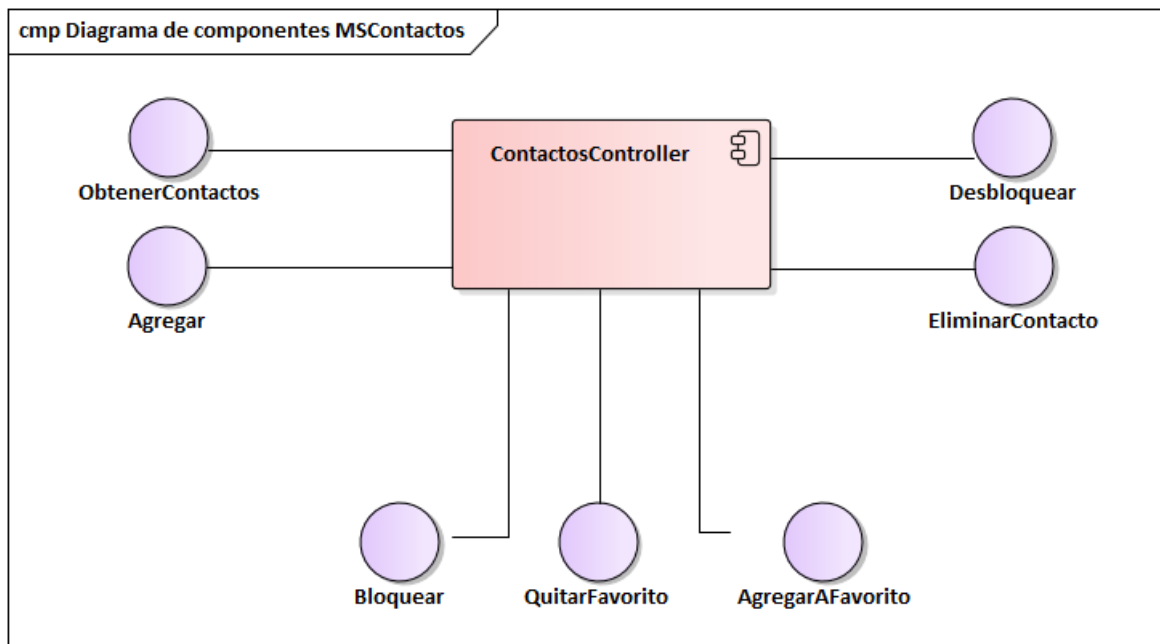
A continuación, se presentarán los diagramas de componentes, los cuales representan la vista de implementación. En este caso, se realizó un diagrama de componente por cada servicio que ofrecemos, desde los microservicios, hasta la API Gateway. Cada diagrama presenta los componentes de cada servicio y las interfaces que ofrece para la comunicación con ellos.

En primer lugar, en la siguiente Figura se muestra el diagrama de componentes para el Microservicio “Typers”. Donde se puede apreciar las interfaces que ofrece para poder comunicarse con él.

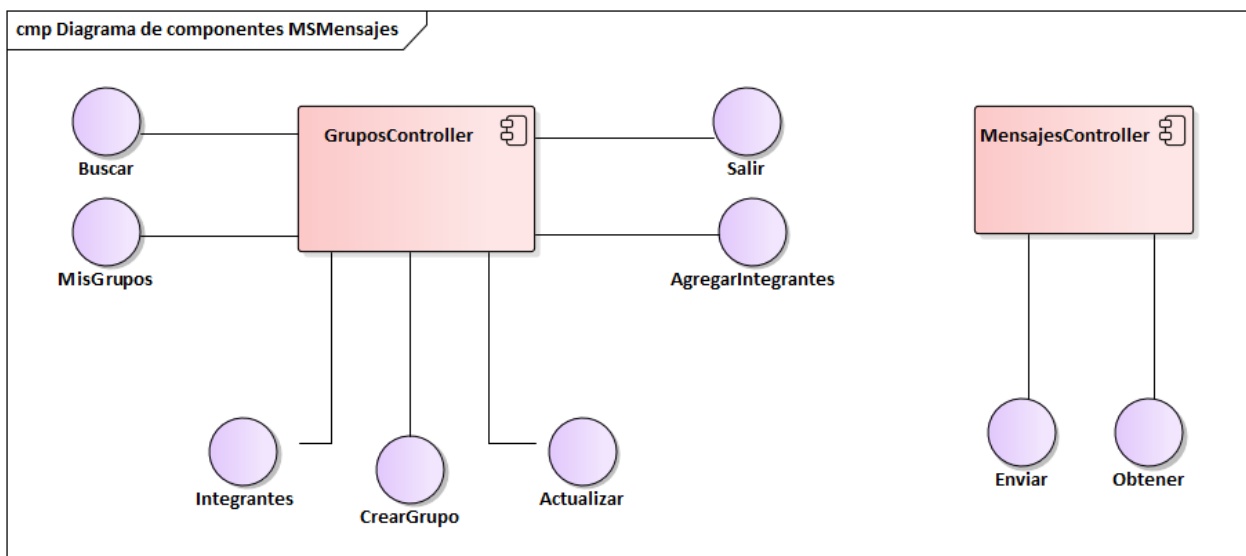




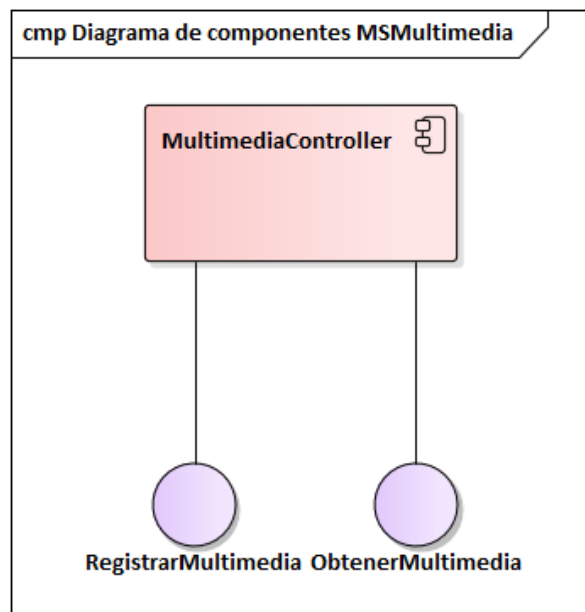
Ahora, en la siguiente Figura se muestra el diagrama de componente para el microservicio de “Contactos”.



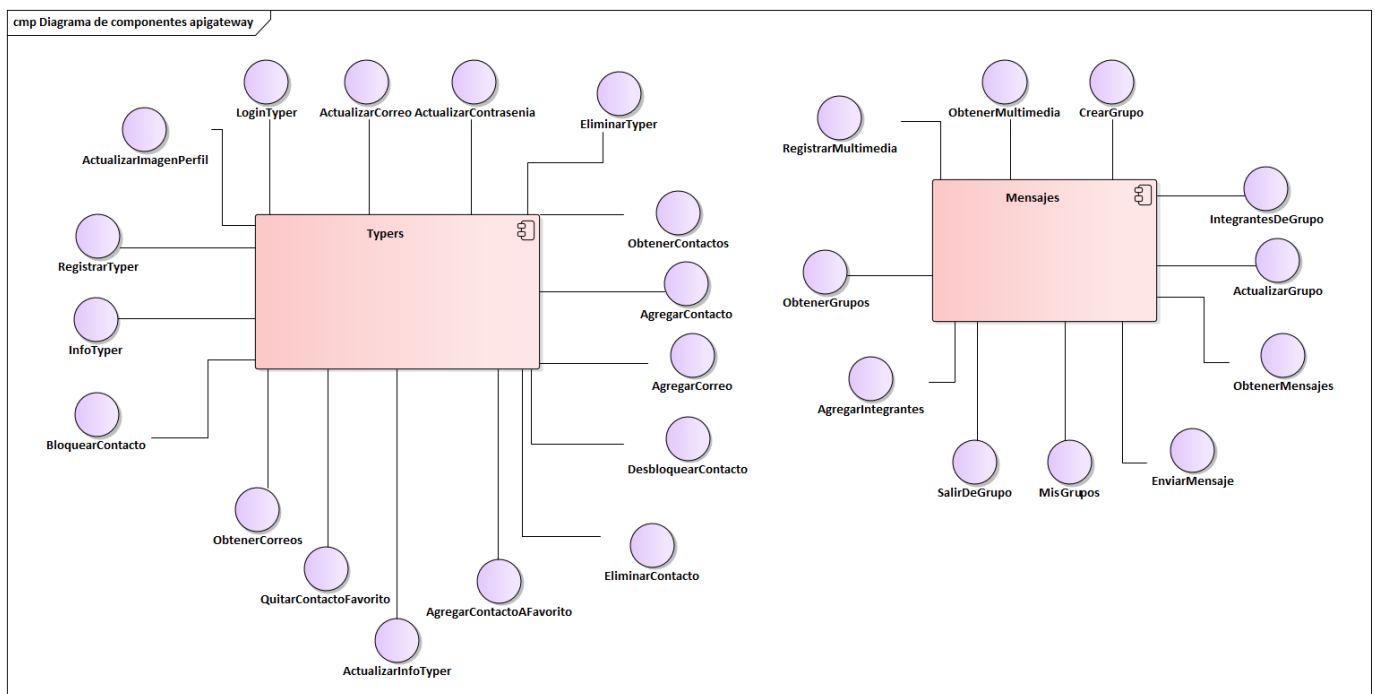
Siguiendo ahora, la Figura que se muestra abajo, presenta el diagrama de componentes para el microservicio de “Mensajes”.



Ahora, en la siguiente Figura, se muestra el diagrama de componentes para el microservicio de Multimedia.



Finalmente, en la siguiente Figura se muestra el diagrama de componentes para la API Gateway.

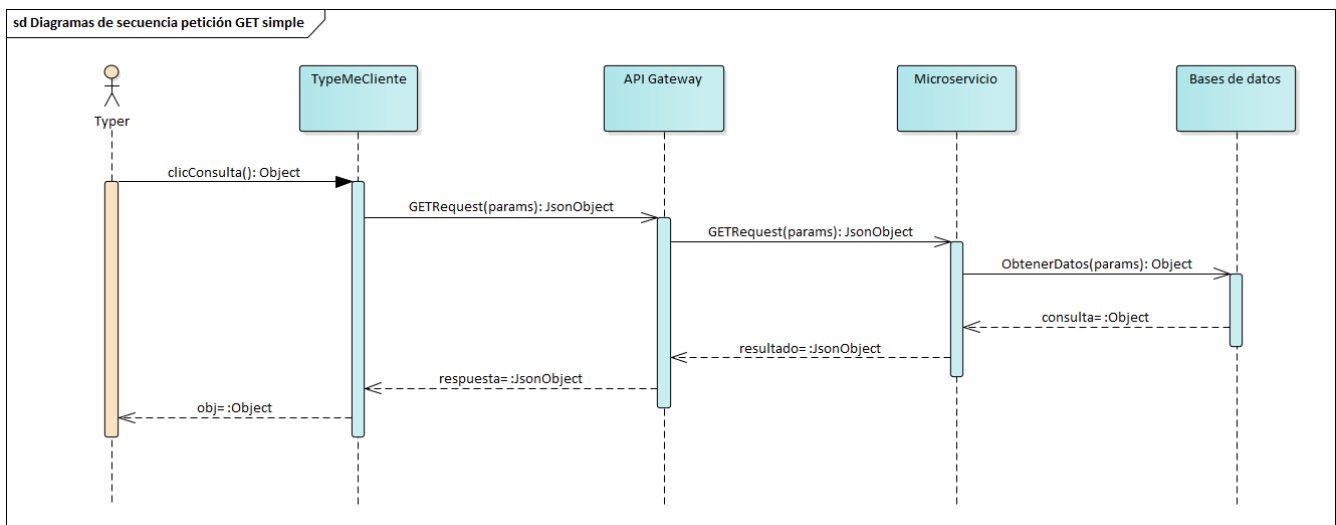


## Vista de procesos

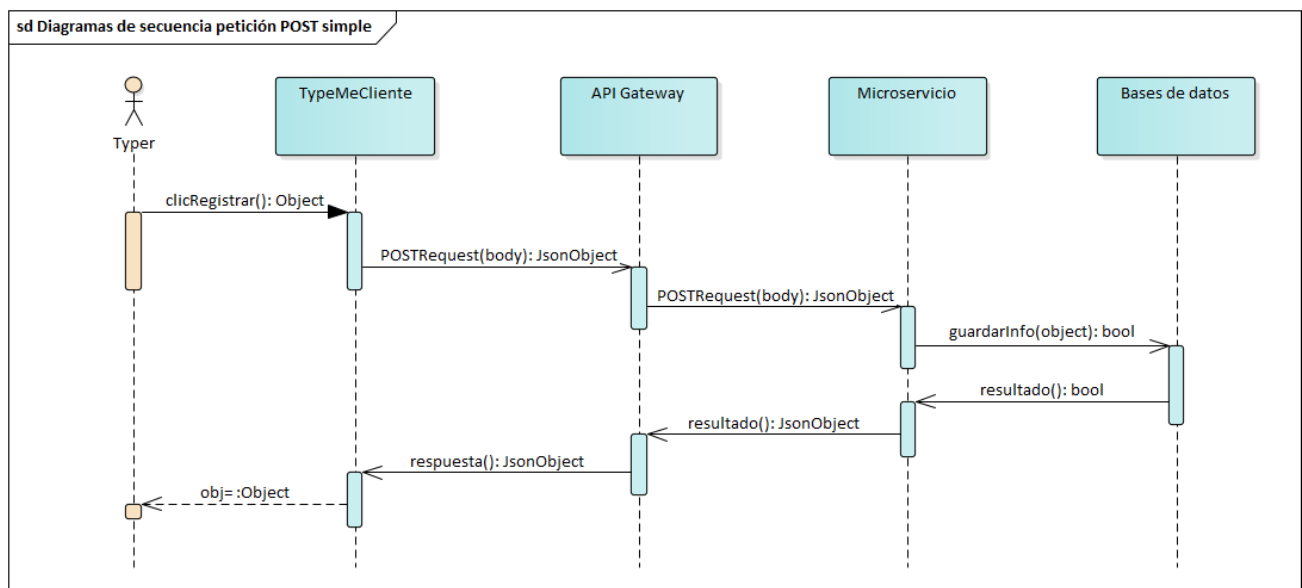
En esta sección se presentan los diagramas de secuencia para mostrar el proceso de las llamadas a los métodos de la API y cómo esta, se comunica con los microservicios correspondientes.

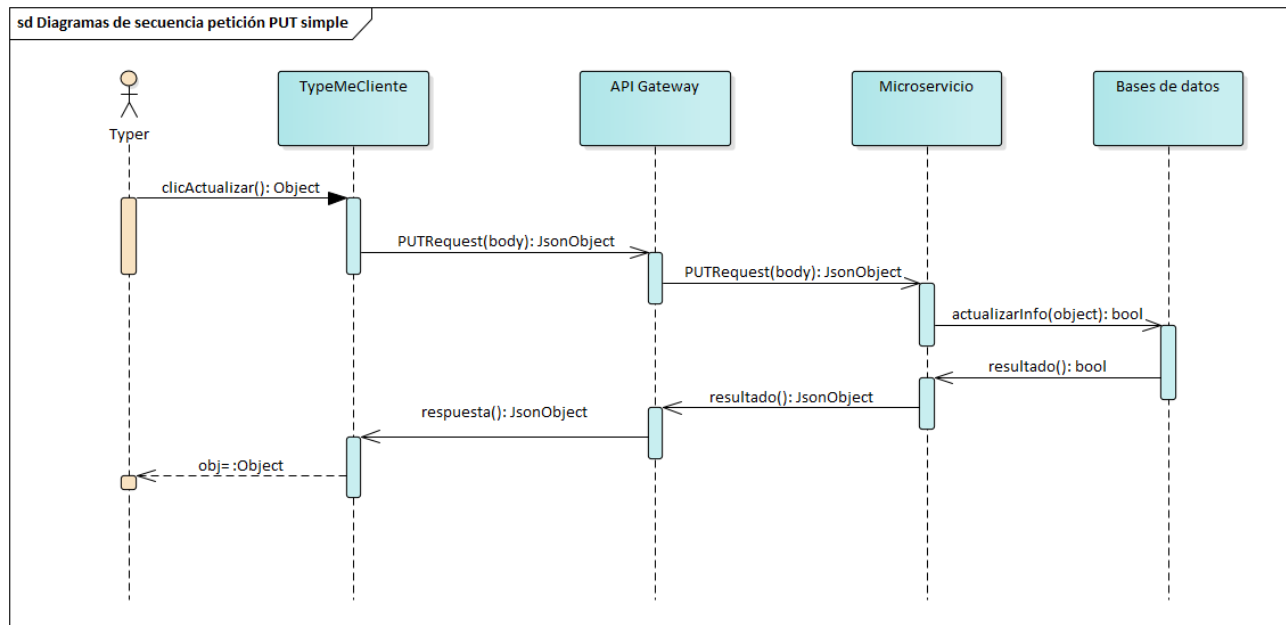
Esta vista de procesos se dividió por los métodos HTTP que los microservicios manejan (GET; POST, PUT) y se generalizaron estos procesos al darnos cuenta de que en la mayoría son similares los comportamientos de las llamadas. A diferencia de algunos, donde hacen más llamadas o realizan otro tipo de consulta a la base de datos.

En el siguiente diagrama de secuencia, se muestra el proceso para las llamadas al método GET simple.

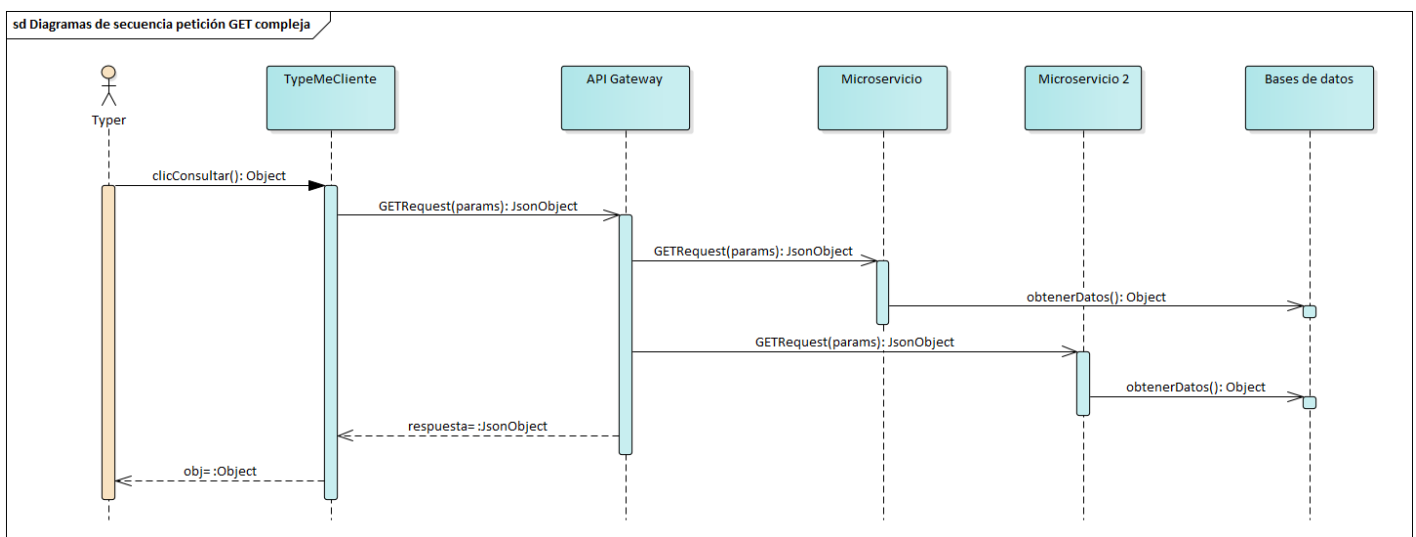


Ahora, siguiendo con el método POST, también se presenta el diagrama de secuencia para una petición simple.



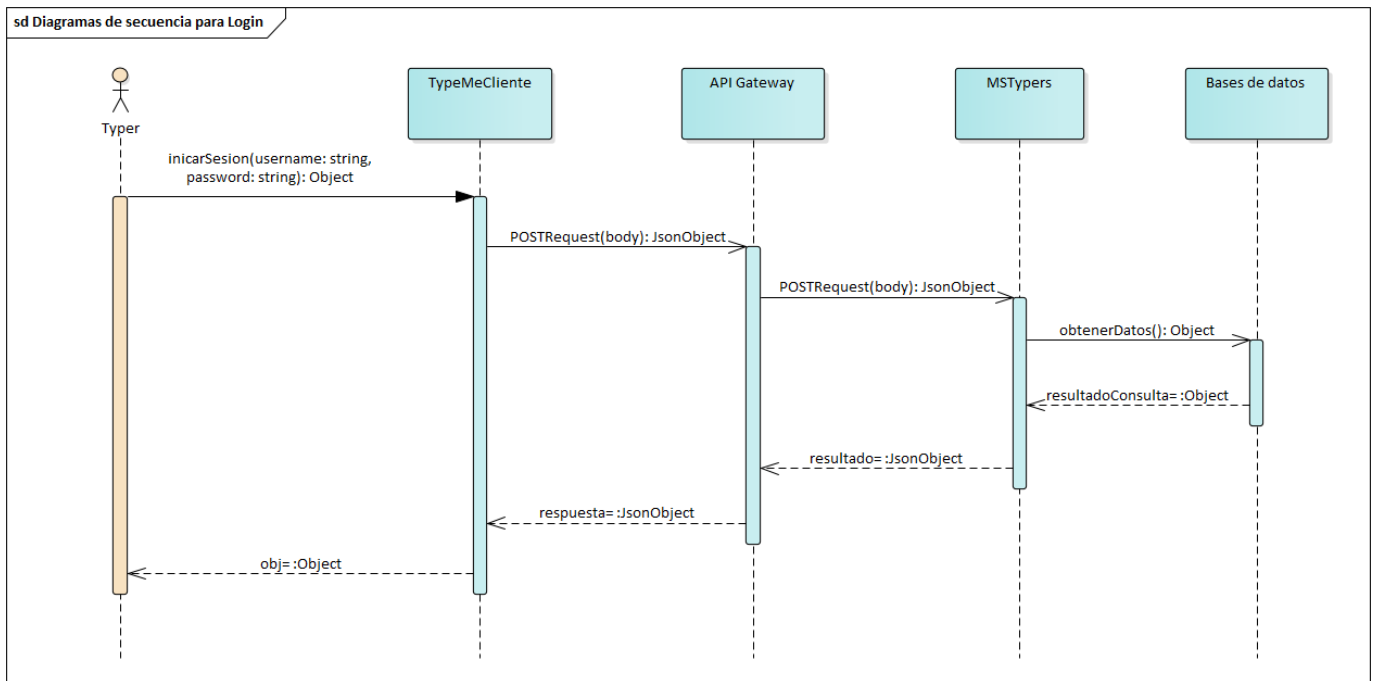


En la siguiente Figura se muestra el diagrama de secuencia para la llamada PUT simple.



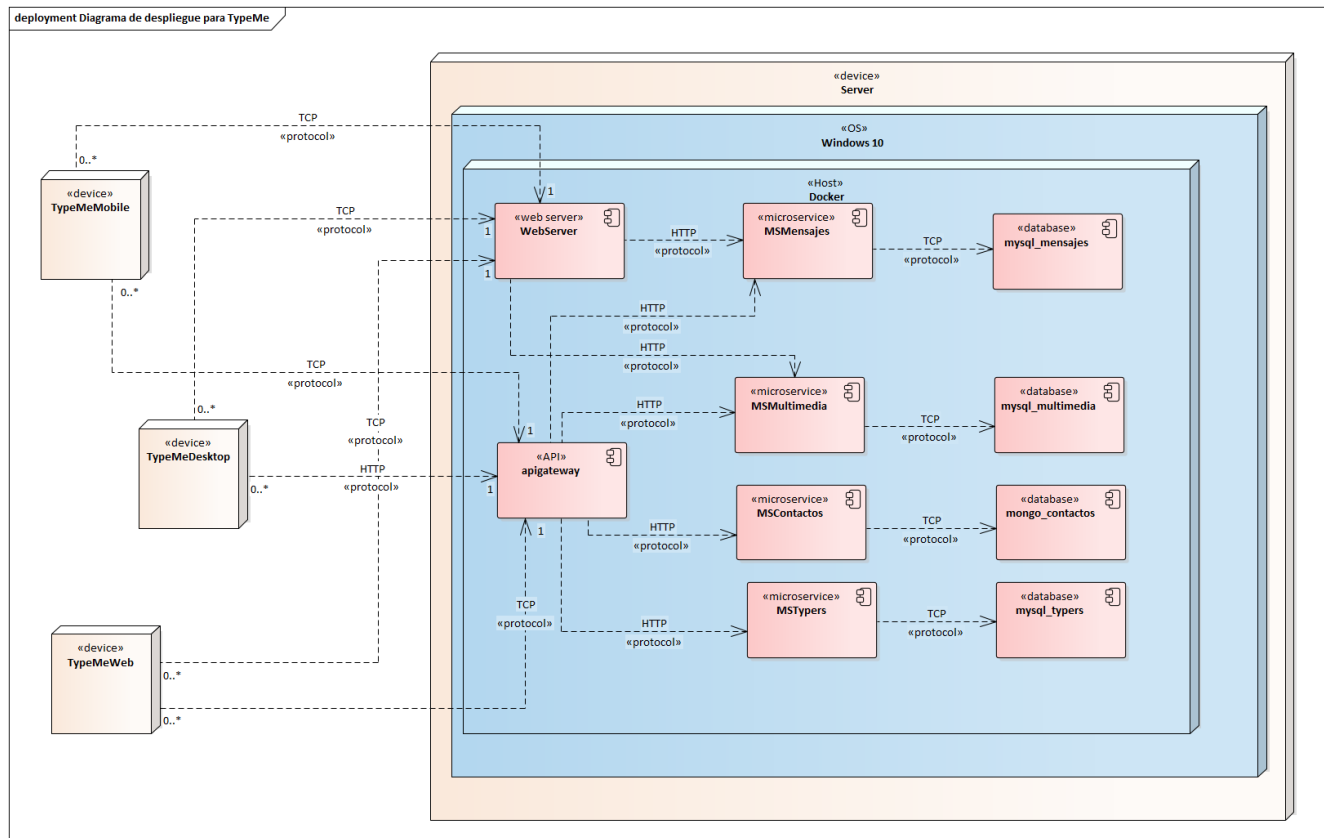
En las peticiones GET existen escenarios donde la API Gateway consulta los datos a más de un microservicio. En la siguiente Figura se muestra el diagrama de secuencia del proceso que se realiza para esto.

Existe una petición POST que cambia un poco el proceso, esto es cuando un usuario desea iniciar sesión y en lugar de crear un registro en la base de datos, solo se consulta la información para verificar que exista el Typers.



## Vista de despliegue

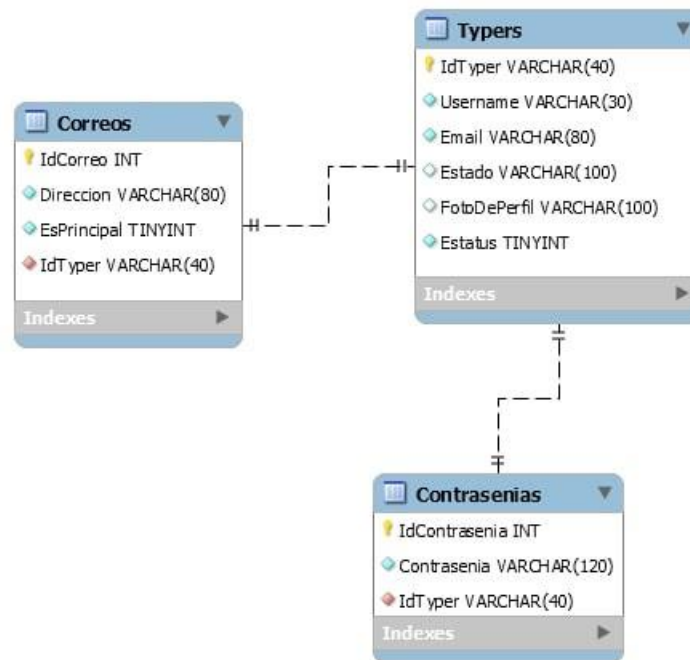
A continuación, se muestra la vista de despliegue de la aplicación TypeMe, donde se muestran los microservicios que fueron desarrollados para la parte del back-end, cómo es que son desplegados con la tecnología de Docker y sus contenedores, además de cómo se comunica la API Gateway con ellos y los clientes a ella.



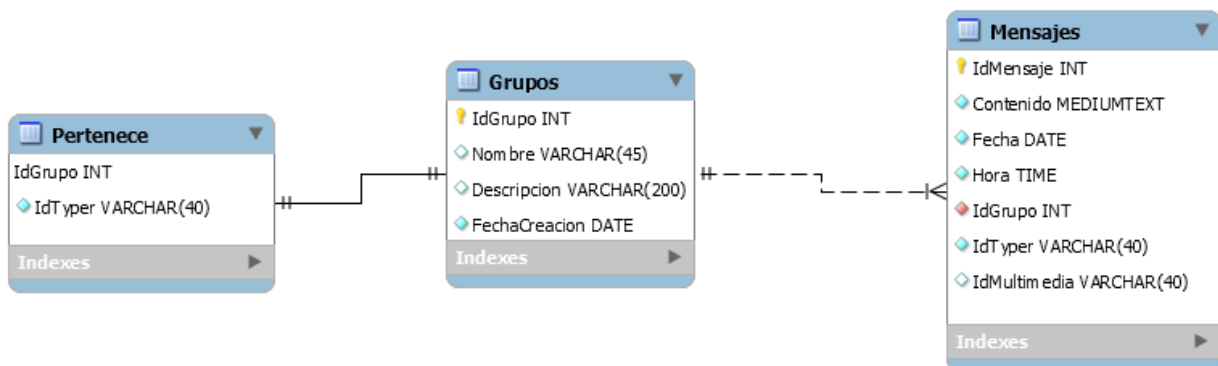
## Modelo de datos

En esta sección se muestran los diferentes diagramas relacionales de cada uno de los microservicios que se han tomado en cuenta para el desarrollo de la aplicación.

A continuación, se muestra el diagrama relacional para el microservicio Typers encargado de las cuentas de usuario.



A continuación, se muestra el diagrama relacional para el microservicio Mensajes encargado de gestionar el envío y almacén de mensajes.



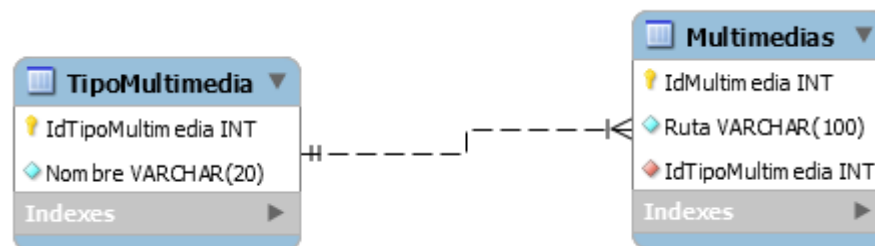




Como para el microservicio de Contactos, se utilizó una base de datos no relacional con MongoDB, no existe un diagrama relacional. Sin embargo, se muestra la estructura que se siguió para llevar el registro de los contactos en un objeto JSON.

```
{
  "idTyper": "identificador del typer",
  "contactos": [
    {
      "idTyper": "identificador del contacto",
      "bloqueado": [0, 1],
      "esFavorito": [0, 1]
    },
    ...
  ]
}
```

A continuación, se muestra el diagrama relacional para la base de datos del microservicio Multimedia encargado de administrar los archivos que serán enviados a través de la aplicación.



## Descripciones de casos de uso

En esta sección se presentan las descripciones de casos de uso, correspondientes al diagrama presentado la vista de casos de uso.

### Administrador

En la siguiente tabla se muestra la descripción para el caso de uso CU-A-01.

<b>ID:</b>	CU-A-01
<b>Nombre:</b>	Ver estadísticas de usuarios registrados
<b>Autor(es):</b>	Sammy Guadarrama Chávez
<b>Fecha de creación:</b>	10 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El Administrador es capaz de analizar las estadísticas de los usuarios registrados en la aplicación, como un promedio del tiempo que están conectados, el horario en que más usan la aplicación, entre otros, para la toma de decisiones.
<b>Actor(es):</b>	<b>Administrador</b>
<b>Frecuencia de uso:</b>	Aproximadamente 10 veces por semana
<b>Disparador:</b>	El <b>Administrador</b> selecciona el botón “Ver estadísticas” dentro del menú principal.
<b>Precondiciones:</b>	PRE-01: Debe existir al menos un usuario registrado en la aplicación.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"><li>1. La aplicación TypeMe consulta a los usuarios registrados junto con su registro de actividad dentro de la aplicación y muestra la ventana “Estadísticas” con la información consultada previamente, y el botón “Regresar”. (Ver EX1).</li><li>2. El <b>Administrador</b> selecciona el botón “Regresar”.</li><li>3. La aplicación TypeMe cierra la ventana “Estadística” y muestra el menú principal.</li><li>4. Fin del caso de uso.</li></ol>
<b>Flujos Alternos:</b>	No identificados
<b>Excepciones:</b>	<p>EX1 – Error de conexión a la base de datos para mostrar la información.</p> <ol style="list-style-type: none"><li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Ocurrió un error al mostrar la información. Por favor intente más tarde” y el botón “Aceptar”.</li><li>2. El <b>Administrador</b> selecciona el botón “Aceptar”.</li><li>3. La aplicación TypeMe muestra el menú principal.</li></ol>
<b>Postcondiciones:</b>	POST-01: La aplicación TypeMe debe mostrar las estadísticas del registro de actividad de los usuarios registrados en la aplicación.
<b>Incluye (relación Include):</b>	No aplica.

<b>Extiende (relación Extend):</b>	No aplica.
--	------------

<b>Prioridad:</b>	Alta
-------------------	------

En la siguiente tabla se muestra la descripción al caso de uso CU-A-02.

<b>ID:</b>	CU-A-02
<b>Nombre:</b>	Bloquear usuario
<b>Autor(es):</b>	Sammy Guadarrama Chávez
<b>Fecha de creación:</b>	10 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El Administrador es capaz de controlar a los usuarios que sean reportados por otros según las razones que se especifiquen. Para esto puede realizar un bloqueo o eliminación completa de la cuenta reportada y así evitar problemas en la comunidad de TypeMe.
<b>Actor(es):</b>	<b>Administrador</b>
<b>Frecuencia de uso:</b>	Aproximadamente 5 veces por semana
<b>Disparador:</b>	El <b>Administrador</b> selecciona el botón “Ver solicitudes de bloqueo” dentro del menú principal.
<b>Precondiciones:</b>	PRE-01: Debe existir al menos un usuario registrado en la aplicación. PRE-02: Debe existir al menos una solicitud de bloqueo sobre un usuario.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Solicitudes bloqueo” con las solicitudes de bloqueo agrupadas por usuario en una lista, en cada elemento de ella se muestra el botón “Detalles”. Al final, muestra el botón “Regresar”. (Ver EX1).</li> <li>2. El <b>Administrador</b> selecciona el botón “Detalles” en un usuario con solicitud de bloqueo. (Ver FA 2.1)</li> <li>3. La aplicación TypeMe muestra la ventana “Solicitud bloqueo”, con la información de las solicitudes para el usuario. Además, muestra los botones “Cancelar”, “Descartar” y “Bloquear”.</li> <li>4. El <b>Administrador</b> selecciona el botón “Bloquear”. (Ver FA 4.1 y FA 4.2).</li> <li>5. La aplicación TypeMe bloquea la cuenta del usuario seleccionado por 5 días y muestra la ventana “Éxito” con el mensaje “El usuario fue bloqueado exitosamente” y el botón “Aceptar”. (Ver EX2)</li> <li>6. El <b>Administrador</b> selecciona el botón “Aceptar”.</li> <li>7. Fin del caso de uso.</li> </ol>
<b>Flujos Alternos:</b>	FA 2.1 – Salir de las solicitudes de bloqueo. <ol style="list-style-type: none"> <li>1. El <b>Administrador</b> selecciona el botón “Regresar”.</li> </ol>

	<ol style="list-style-type: none"> <li>2. La aplicación TypeMe muestra el menú principal.</li> <li>3. Termina caso de uso.</li> </ol> <p>FA 4.1 – Cancelación del bloqueo de una cuenta.</p> <ol style="list-style-type: none"> <li>1. El <b>Administrador</b> selecciona el botón “Cancelar”.</li> <li>2. La aplicación TypeMe regresa al paso 1 del flujo normal.</li> </ol> <p>FA 4.2 – Descarte de la solicitud de bloqueo de un usuario.</p> <ol style="list-style-type: none"> <li>1. El <b>Administrador</b> selecciona el botón “Descartar”.</li> <li>2. La aplicación TypeMe elimina todas solicitudes de bloqueo relacionadas con el usuario y regresa al paso 1 del flujo normal.</li> </ol>
<b>Excepciones:</b>	<p>EX1 – Error de conexión para mostrar las solicitudes de bloqueo.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Ocurrió un error al mostrar las solicitudes. Por favor intente más tarde” y el botón “Aceptar”.</li> <li>2. El <b>Administrador</b> selecciona el botón “Aceptar”.</li> <li>3. La aplicación TypeMe muestra el menú principal.</li> </ol> <p>EX2 – Error de conexión para bloquear un usuario.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Ocurrió un error al intentar bloquear la cuenta. Por favor intente más tarde” y el botón “Aceptar”.</li> <li>2. El <b>Administrador</b> selecciona el botón “Aceptar”.</li> <li>3. La aplicación TypeMe regresa al paso 3 del flujo normal.</li> </ol>
<b>Postcondiciones:</b>	<p>POST-01: La aplicación TypeMe bloquea la cuenta del usuario seleccionado por el Administrador.</p> <p>POST-02: La aplicación TypeMe descarta las solicitudes de bloqueo de una cuenta seleccionada por el Administrador.</p>
<b>Incluye (relación Include):</b>	No aplica.
<b>Extiende (relación Extend):</b>	No aplica.
<b>Prioridad:</b>	Media

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-01.

<b>ID:</b>	CU-T-01
<b>Nombre:</b>	Registrar cuenta
<b>Autor(es):</b>	Sammy Guadarrama Chávez
<b>Fecha de creación:</b>	10 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El registro de una cuenta dentro de la aplicación TypeMe permite a las personas acceder a las funcionalidades principales para mantenerse en contacto con las personas que también estén registrados y sean agregados por el usuario.
<b>Actor(es):</b>	<b>Typer</b>
<b>Frecuencia de uso:</b>	Aproximadamente 40 veces por semana
<b>Disparador:</b>	El <b>Typer</b> selecciona el botón “Registrar cuenta” dentro la ventana “Iniciar sesión”.
<b>Precondiciones:</b>	No identificadas
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Registrar cuenta” con los campos necesarios para el registro, además muestra los botones “Cancelar” y “Registrar”.</li> <li>2. El <b>Typer</b> ingresa los datos solicitados y selecciona el botón “Registrar”. (Ver FA 2.1).</li> <li>3. La aplicación TypeMe verifica que los campos estén completos y que el “username” ingresado no pertenezca a otro usuario. Después muestra la ventana “Éxito” con el mensaje “Se ha registrado exitosamente” y el botón “Aceptar”. (Ver FA 3.1, FA 3.2 y EX1).</li> <li>4. El <b>Typer</b> selecciona el botón “Aceptar”.</li> <li>5. La aplicación TypeMe muestra la ventana “Iniciar sesión”.</li> <li>6. Termina caso de uso.</li> </ol>
<b>Flujos Alternos:</b>	<p>FA 2.1 – El Typer cancela el registro de una cuenta.</p> <ol style="list-style-type: none"> <li>1. El <b>Typer</b> selecciona el botón “Cancelar”.</li> <li>2. La aplicación TypeMe cierra la ventana “Registrar cuenta” y muestra la ventana “Iniciar sesión”.</li> <li>3. Termina caso de uso.</li> </ol> <p>FA 3.1 – Existen campos incompletos para el registro.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Hay campos vacíos, por favor ingresa toda la información solicitada e intente de nuevo.” Y el botón “Aceptar”.</li> <li>2. El <b>Typer</b> selecciona el botón “Aceptar”.</li> </ol>

	<p>3. La aplicación TypeMe regresa al paso 2 del flujo normal.</p> <p>FA 3.2 – El username ingresado ya existe.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “El username ingresa ya se encuentra registrado. Por favor ingrese otro diferente.” Y el botón “Aceptar”:</li> <li>2. El <b>Typier</b> selecciona el botón “Aceptar”.</li> <li>3. La aplicación TypeMe regresa al paso 2 del flujo normal.</li> </ol>
<b>Excepciones:</b>	<p>EX1 – Error de conexión a la base de datos para registrar una nueva cuenta.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Ocurrió un error al intentar registrar la cuenta. Por favor intente más tarde” y el botón “Aceptar”.</li> <li>2. El <b>Typier</b> selecciona el botón “Aceptar”.</li> <li>3. La aplicación TypeMe regresa al paso 2 del flujo normal.</li> </ol>
<b>Postcondiciones:</b>	POST-01: La aplicación TypeMe crea una nueva cuenta con la información ingresada por el usuario.
<b>Incluye (relación Include):</b>	No aplica.
<b>Extiende (relación Extend):</b>	No aplica.
<b>Prioridad:</b>	Alta

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-02:  
Enviar mensaje.

<b>ID:</b>	CU-T-02
<b>Nombre:</b>	Enviar mensaje
<b>Autor(es):</b>	Ángel de Jesús Juárez García
<b>Fecha de creación:</b>	11 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El <b>Typer</b> envía un mensaje a un contacto de su agenda
<b>Actor(es):</b>	<b>Typer</b>
<b>Frecuencia de uso:</b>	Aproximadamente 30 veces al día
<b>Disparador:</b>	El <b>Typer</b> selecciona a un contacto de su agenda
<b>Precondiciones:</b>	PRE-01: El contacto debe de estar registrado en la agenda PRE-02: El contacto no debe de estar bloqueado
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. TypeMe actualiza la ventana mostrando la bandeja de mensajes mostrando los mensajes más recientes entre el <b>Typer</b> y el contacto, además en la parte inferior se muestra un cuadro de texto y un botón para enviar un mensaje</li> <li>2. El <b>Typer</b> ingresa el texto que desea enviar en el cuadro de texto mostrado y pulsa el botón enviar.(Ver FA 2.1)</li> <li>3. TypeMe verifica que exista texto en el mensaje y captura lo ingresado en el cuadro de texto y lo envía a los servidores para que sea enviado al contacto del <b>Typer</b> y actualiza el historial de mensajes mostrando el mensaje más nuevo en la parte inferior.(Ver EX 01) (Ver FA 3.1)</li> <li>4. Fin del caso de uso</li> </ol>
<b>Flujos Alternos:</b>	<p>FA 2.1 – El <b>Typer</b> desea enviar un archivo multimedia</p> <ol style="list-style-type: none"> <li>1. El <b>Typer</b> ingresa el texto que desee enviar y pulsa el botón para enviar multimedia</li> <li>2. El Caso de Uso extiende a CU-T-05: Enviar archivo multimedia</li> </ol> <p>FA 3.1 – Existen campos incompletos para la agregación.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Hay campos vacíos, por favor ingresa toda la información solicitada e intente de nuevo.” Y el botón “Aceptar”.</li> <li>2. El <b>Typer</b> selecciona el botón “Aceptar”.</li> <li>3. La aplicación TypeMe regresa al paso 2 del flujo normal.</li> </ol>



<b>Excepciones:</b>	EX 01 – TypeMe no es capaz de enviar el mensaje <ol style="list-style-type: none"> <li>1. TypeMe no logra realizar una conexión con el servidor y muestra un mensaje de error “Error al enviar el mensaje” junto con la opción “Ok”</li> <li>2. El <b>Typer</b> selecciona el botón “Ok”</li> <li>3. TypeMe muestra la bandeja de mensajes entre en contacto y el <b>Typer</b></li> <li>4. TypeMe regresa al paso 4 del caso de uso</li> </ol>
<b>Postcondiciones:</b>	POST-01: La bandeja de mensajes cuenta con un nuevo mensaje por parte del <b>Typer</b>
<b>Incluye (relación Include):</b>	No incluye
<b>Extiende (relación Extend):</b>	No extiende
<b>Prioridad:</b>	Alta

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-03:  
Enviar archivos multimedia.

<b>ID:</b>	CU-T-03
<b>Nombre:</b>	Enviar archivo multimedia
<b>Autor(es):</b>	Ángel de Jesús Juárez García
<b>Fecha de creación:</b>	11 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El <b>Typers</b> envía un archivo (PDF, imagen o video) a un contacto de su agenda
<b>Actor(es):</b>	<b>Typers</b>
<b>Frecuencia de uso:</b>	Aproximadamente 10 veces por semana
<b>Disparador:</b>	El <b>Typers</b> selecciona el botón de multimedia en el chat
<b>Precondiciones:</b>	PRE-01: El contacto debe de estar registrado en la agenda del <b>Typers</b>
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. TypeMe muestra un menú en el que se muestra los tipos de archivo que el <b>Typers</b> puede enviar junto con un botón "Cancelar"</li> <li>2. El <b>Typers</b> selecciona un tipo de archivo a enviar (Ver FA 2.1)</li> <li>3. TypeMe abre el explorador de archivos para que el <b>Typers</b> seleccione uno</li> <li>4. El <b>Typers</b> selecciona un archivo que desea enviar</li> <li>5. TypeMe muestra el nombre del archivo a enviar (Ver FA 2.1)</li> <li>6. El <b>Typers</b> pulsa el botón enviar en la parte inferior derecha</li> <li>7. TypeMe obtiene el archivo seleccionado por el <b>Typers</b> y lo envía al contacto seleccionado, además guarda una copia del archivo en la base de datos y actualiza la interfaz mostrando el archivo enviado. (Ver EX01)</li> <li>8. Fin del caso de uso</li> </ol>
<b>Flujos Alternos:</b>	FA 2.1 – El <b>Typers</b> desea cancelar la operación <ol style="list-style-type: none"> <li>1. El <b>Typers</b> selecciona el botón "Cancelar"</li> <li>2. TypeMe elimina la ventana emergente de selección</li> <li>3. Fin del caso de uso</li> </ol>
<b>Excepciones:</b>	EX 01 – TypeMe no es capaz de enviar el mensaje <ol style="list-style-type: none"> <li>1. TypeMe no logra realizar una conexión con el servidor y muestra un mensaje de error "Error al enviar el archivo" junto con la opción "Ok"</li> <li>2. El <b>Typers</b> selecciona el botón "Ok"</li> <li>3. TypeMe muestra la bandeja de mensajes entre en contacto y el <b>Typers</b></li> </ol>

	4. TypeMe regresa al paso 1 del caso de uso
<b>Postcondiciones:</b>	POST-01: La bandeja de mensajes cuenta con un nuevo mensaje de tipo archivo por parte del <b>Typer</b> POST-02: El archivo enviado se encuentra almacenado en la base de datos
<b>Incluye (relación Include):</b>	No incluye
<b>Extiende (relación Extend):</b>	Extiende de CU-T-04
<b>Prioridad:</b>	Alta

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-04:  
Agregar contacto.

<b>ID:</b>	CU-T-04
<b>Nombre:</b>	Agregar contacto
<b>Autor(es):</b>	Ángel de Jesús Juárez García
<b>Fecha de creación:</b>	10 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	Permite al <b>Typer</b> agregar un nuevo contacto a su agenda para poder intercambiar mensajes entre las dos cuentas
<b>Actor(es):</b>	<b>Typer</b>
<b>Frecuencia de uso:</b>	Aproximadamente 10 veces por semana
<b>Disparador:</b>	El <b>Typer</b> selecciona el botón “Agregar contacto” dentro la ventana “Principal”.
<b>Precondiciones:</b>	No identificadas
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Agregar contacto” con los campos “Usuario o correo” y los botones “Agregar” y “Cancelar”.</li> <li>2. El <b>Typer</b> ingresa la información en el campo solicitado y selecciona la opción “Agregar”. (Ver FA 2.1)</li> <li>3. La aplicación TypeMe verifica que los campos estén completos y busca el contacto en la base de datos y lo agrega como contacto al perfil del <b>Typer</b>, por ultimo la aplicación TypeMe muestra una ventana emergente con el mensaje “Contacto agregado exitosamente” con la opción “Ok” (Ver FA 3.1) (Ver EX01)</li> <li>4. El <b>Typer</b> selecciona la opción “Ok”</li> <li>5. La aplicación TypeMe muestra la ventana “Principal” actualizando los contactos mostrando el más reciente</li> <li>6. Fin del caso de uso</li> </ol>

<b>Flujos Alternos:</b>	<p>FA 2.1 – El <b>Typer</b> cancela el registro de una cuenta.</p> <ol style="list-style-type: none"> <li>1. El <b>Typer</b> selecciona el botón “Cancelar”.</li> <li>2. La aplicación TypeMe cierra la ventana “Agregarcontacto” y muestra la ventana “Principal”.</li> <li>3. Termina caso de uso.</li> </ol> <p>FA 3.1 – Existen campos incompletos para la agregación.</p> <ol style="list-style-type: none"> <li>4. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Hay campos vacíos, por favor ingresa toda la información solicitada e intente de nuevo.” Y el botón “Aceptar”.</li> <li>5. El <b>Typer</b> selecciona el botón “Aceptar”.</li> <li>6. La aplicación TypeMe regresa al paso 2 del flujonormal.</li> </ol>
<b>Excepciones:</b>	<p>EX1 – Error de conexión a la base de datos para agregar un contacto</p> <ol style="list-style-type: none"> <li>4. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Ocurrió un error al intentar agregar el contacto. Por favor intente más tarde” y el botón “Aceptar”.</li> <li>5. El <b>Typer</b> selecciona el botón “Aceptar”.</li> <li>6. La aplicación TypeMe regresa al paso 2 del flujo normal.</li> </ol>
<b>Postcondiciones:</b>	<p>POST-01: El <b>Typer</b> cuenta con un nuevo contacto en su lista de contactos</p> <p>POST-02: Existe un registro de contacto en la base de datos</p>
<b>Incluye (relación Include):</b>	No aplica.
<b>Extiende (relación Extend):</b>	No aplica.
<b>Prioridad:</b>	Alta

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-05: Editar perfil.

<b>ID:</b>	CU-T-05
<b>Nombre:</b>	Editar perfil
<b>Autor(es):</b>	Ángel de Jesús Juárez García
<b>Fecha de creación:</b>	11 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El <b>Typers</b> actualiza su información de perfil que es mostradaa otros <b>Typers</b>
<b>Actor(es):</b>	<b>Typers</b>
<b>Frecuencia de uso:</b>	Aproximadamente 2 veces por semana
<b>Disparador:</b>	<b>Typers</b> selecciona la opción “Editar” en la ventana “Mi perfil”
<b>Precondiciones:</b>	No identificadas
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. TypeMe muestra la ventana “Mi perfil” mostrando campos para que el usuario ingrese la información que desea actualizar junto con las opciones “Guardar” y “Cancelar”.</li> <li>2. El <b>Typers</b> ingresa la información que desea cambiar en su perfil y pulsa el botón “Guardar”. (Ver FA 2.1)</li> <li>3. TypeMe verifica que los campos estén completos y captura la información ingresada por el usuario y manda la información al servidor para ser actualizada, mostrando una ventana con el mensaje “Información actualizada correctamente” con la opción “Ok”. (Ver EX01) (Ver FA 3.1)</li> <li>4. El <b>Typers</b> selecciona la opción “Ok”</li> <li>5. TypeMe regresa a la ventana “Mi perfil” reflejando los cambios hechos</li> <li>6. Fin del caso de uso</li> </ol>

<b>Flujos Alternos:</b>	<p>FA 2.1 – El <b>Typser</b> desea cancelar la operación de editar</p> <ol style="list-style-type: none"> <li>1. El <b>Typser</b> selecciona el botón “Cancelar”</li> <li>2. TypeMe recarga la pantalla “Mi perfil” quitando los campos para la edición de la información</li> <li>3. Fin del caso de uso</li> </ol> <p>FA 3.1 – Existen campos incompletos para la actualización.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Hay campos vacíos, por favor ingresa toda la información solicitada e intente de nuevo.” Y el botón “Aceptar”.</li> <li>2. El <b>Typser</b> selecciona el botón “Aceptar”.</li> <li>3. La aplicación TypeMe regresa al paso 2 del flujo normal.</li> </ol>
<b>Excepciones:</b>	<p>EX 01 – TypeMe no es capaz de conectarse con el servidor</p> <ol style="list-style-type: none"> <li>1. TypeMe no logra conectarse con el servidor y muestra una ventana con el mensaje “No se pudo actualizar la información, intente más tarde” junto con la opción “Ok”</li> <li>2. El <b>Typser</b> selecciona la opción “Ok”</li> <li>3. TypeMe regresa al paso 1 del flujo normal</li> </ol>
<b>Postcondiciones:</b>	POST-01: El <b>Typser</b> cuenta con nueva información en superfil
<b>Incluye (relación Include):</b>	No incluye
<b>Extiende (relación Extend):</b>	No extiende
<b>Prioridad:</b>	Alta

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-06:  
 Editar perfil.

<b>ID:</b>	CU-T-06
<b>Nombre:</b>	Eliminar historial de mensajes
<b>Autor(es):</b>	Sammy Guadarrama Chávez
<b>Fecha de creación:</b>	10 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	La eliminación del historial de mensajes permite que los mensajes dentro de una conversación seleccionada se eliminen definitivamente, además de los archivos multimedia enviados.
<b>Actor(es):</b>	<b>Typer</b>
<b>Frecuencia de uso:</b>	Aproximadamente 6 veces por semana
<b>Disparador:</b>	El <b>Typer</b> selecciona el botón “Eliminar historial” dentro el menú de opciones de un chat.
<b>Precondiciones:</b>	PRE-01: El Typer debe tener al menos una conversación iniciada con otro Typer. PRE-02: La conversación iniciada debe tener al menos un mensaje en su historial.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Confirmación” con el mensaje “¿Seguro desea eliminar el historial de mensajes de esta conversación?, y los botones “Cancelar” y “Eliminar”.</li> <li>2. El <b>Typer</b> selecciona el botón “Eliminar”. (Ver FA 2.1).</li> <li>3. La aplicación TypeMe elimina todos los mensajes y archivos multimedia enviados y muestra la descripción en el chat “Aún no hay mensajes en esta conversación”. (Ver EX1).</li> <li>4. Termina caso de uso.</li> </ol>
<b>Flujos Alternos:</b>	FA 2.1 – El Typer cancela la eliminación del historial de mensajes. <ol style="list-style-type: none"> <li>1. El <b>Typer</b> selecciona el botón “Cancelar”.</li> <li>2. La aplicación TypeMe cierra la ventana “Confirmación”.</li> <li>3. Termina caso de uso.</li> </ol>
<b>Excepciones:</b>	EX1 – Error de conexión para eliminar el historial de mensajes. <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Ocurrió un error al intentar eliminar el historial de mensajes. Por favor intente más tarde” y el botón “Aceptar”.</li> <li>2. El <b>Typer</b> selecciona el botón “Aceptar”.</li> <li>3. Termina caso de uso.</li> </ol>

<b>Postcondiciones:</b>	POST-01: La aplicación TypeMe elimina el historial de mensajes de la conversación seleccionada por el Typer.
<b>Incluye (relación Include):</b>	No aplica.
<b>Extiende (relación Extend):</b>	No aplica.
<b>Prioridad:</b>	Alta

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-07: Creargrupo.

<b>ID:</b>	CU-T-07
<b>Nombre:</b>	Crear grupo
<b>Autor(es):</b>	Ángel de Jesús Juárez García
<b>Fecha de creación:</b>	11 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El <b>Typer</b> es capaz de seleccionar uno o más contactos de su lista y crear un grupo de conversación al cual se pueden unir aún más contactos.
<b>Actor(es):</b>	<b>Typer</b>
<b>Frecuencia de uso:</b>	Aproximadamente 10 veces al mes
<b>Disparador:</b>	El <b>Typer</b> selecciona la opción “Crear grupo” en la pantalla principal
<b>Precondiciones:</b>	PRE-01: El contacto a agregar debe de haber sido registrado por el <b>Typer</b> PRE-02: El contacto seleccionado no debe de estar bloqueado
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. TypeMe muestra la ventana “Crear conversación” mostrando un campo para colocar el nombre del nuevo grupo y la lista de los contactos que puede agregar el <b>Typer</b> junto con los botones “Crear conversación” y “Cancelar”.</li> <li>2. El <b>Typer</b> selecciona a él o los contactos que desea agregar a la conversación, coloca un nombre al grupo que se va a crear y selecciona la opción “Crear conversación”. (Ver FA 2.1, Ver FA 2.2)</li> <li>3. TypeMe recolecta los contactos seleccionados, verifica que los campos estén completos y crea un nuevo grupo con todos los contactos, TypeMe envía el grupo al servidor para su registro y actualiza interfaz mostrando la bandeja de mensajes del grupo creado.</li> <li>4. Fin del caso de uso.</li> </ol>



<b>Flujos Alternos:</b>	<p>FA 2.1 – El <b>Typer</b> busca un contacto de la agenda</p> <ol style="list-style-type: none"> <li>1. El <b>Typer</b> selecciona la barra de búsqueda e ingresa el nombre de algún contacto registrado</li> <li>2. TypeMe realiza una búsqueda entre la agenda del <b>Typer</b> y se actualizan los contactos mostrados en la barra izquierda con las coincidencias de la búsqueda</li> <li>3. El <b>Typer</b> selecciona el usuario mostrado</li> <li>4. Se regresa al paso 1 del flujo normal</li> </ol> <p>FA 2.2 – El <b>Typer</b> desea cancelar la creación del grupo</p> <ol style="list-style-type: none"> <li>1. El <b>Typer</b> selecciona la opción “Cancelar”</li> <li>2. TypeMe cierra la ventana de creación de grupo</li> <li>3. Termina el caso de uso</li> </ol> <p>FA 3.1 – Existen campos incompletos para la creación del grupo.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Hay campos vacíos, por favor ingresa toda la información solicitada e intente de nuevo.” Y el botón “Aceptar”.</li> <li>2. El <b>Typer</b> selecciona el botón “Aceptar”.</li> <li>3. La aplicación TypeMe regresa al paso 2 del flujo normal.</li> </ol>
<b>Excepciones:</b>	
<b>Postcondiciones:</b>	POST-01: TypeMe muestra en la zona derecha una bandeja de mensajes entre el contacto y el <b>Typer</b>
<b>Incluye (relación Include):</b>	No incluye
<b>Extiende (relación Extend):</b>	No extiende
<b>Prioridad:</b>	Alta

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-08.

<b>ID:</b>	CU-T-10
<b>Nombre:</b>	Eliminar contacto
<b>Autor(es):</b>	Sammy Guadarrama Chávez
<b>Fecha de creación:</b>	10 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El Typer tiene la posibilidad de eliminar un contacto desea de su lista de contactos para ya no iniciar una conversación con dicho contacto.
<b>Actor(es):</b>	<b>Typer</b>
<b>Frecuencia de uso:</b>	Aproximadamente 3 veces por mes
<b>Disparador:</b>	El <b>Typer</b> selecciona el botón “Eliminar contacto” del menú de “Opciones” en la ventana “Contactos” en un contacto deseado.
<b>Precondiciones:</b>	PRE-01: El Typer debe tener al menos un contacto registrado en su lista de contactos.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Confirmación” con el mensaje “¿Seguro desea eliminar a NOMBRE_CONTACTO de tu lista de contactos?, y los botones “Cancelar” y “Eliminar”.</li> <li>2. El <b>Typer</b> selecciona el botón “Eliminar”. (Ver FA 2.1).</li> <li>3. La aplicación TypeMe elimina al contacto seleccionado de la lista de contactos. (Ver EX1).</li> <li>4. Termina caso de uso.</li> </ol>
<b>Flujos Alternos:</b>	<p>FA 2.1 – El Typer cancela la eliminación de un contacto.</p> <ol style="list-style-type: none"> <li>1. El <b>Typer</b> selecciona el botón “Cancelar”.</li> <li>2. La aplicación TypeMe cierra la ventana “Confirmación”.</li> <li>3. Termina caso de uso.</li> </ol>
<b>Excepciones:</b>	<p>EX1 – Error de conexión al eliminar un contacto.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Ocurrió un error al intentar eliminar el contacto seleccionado. Por favor intente más tarde” y el botón “Aceptar”.</li> <li>2. El <b>Typer</b> selecciona el botón “Aceptar”.</li> <li>3. Termina caso de uso.</li> </ol>
<b>Postcondiciones:</b>	POST-01: La aplicación TypeMe elimina el contacto seleccionado de la lista de contactos del Typer.
<b>Incluye (relación Include):</b>	No aplica.
<b>Extiende (relación Extend):</b>	No aplica.
<b>Prioridad:</b>	Baja

En la siguiente tabla se muestra la descripción para el caso de uso CU-T-09.

<b>ID:</b>	CU-T-11
<b>Nombre:</b>	Bloquear contacto
<b>Autor(es):</b>	Sammy Guadarrama Chávez
<b>Fecha de creación:</b>	10 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	Cuando se desea que un contacto no pueda enviarte mensajes o que visuales nuevos mensajes, o que el Typer envíe mensajes a ese contacto, se realiza el bloqueo de un contacto.
<b>Actor(es):</b>	<b>Typer</b>
<b>Frecuencia de uso:</b>	Aproximadamente 3 veces por mes
<b>Disparador:</b>	El <b>Typer</b> selecciona el botón “Bloquear contacto” del menú de “Opciones” en la ventana “Contactos” en un contacto deseado.
<b>Precondiciones:</b>	PRE-01: El Typer debe tener al menos un contacto registrado en su lista de contactos.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Confirmación” con el mensaje “¿Seguro desea bloquear a NOMBRE_CONTACTO?, y los botones “Cancelar” y “Bloquear”.</li> <li>2. El <b>Typer</b> selecciona el botón “Bloquear”. (Ver FA 2.1).</li> <li>3. La aplicación TypeMe bloquea al contacto seleccionado de la lista de contactos. (Ver EX1).</li> <li>4. Termina caso de uso.</li> </ol>
<b>Flujos Alternos:</b>	<p>FA 2.1 – El Typer cancela el bloqueo de un contacto.</p> <ol style="list-style-type: none"> <li>1. El <b>Typer</b> selecciona el botón “Cancelar”.</li> <li>2. La aplicación TypeMe cierra la ventana “Confirmación”.</li> <li>3. Termina caso de uso.</li> </ol>
<b>Excepciones:</b>	<p>EX1 – Error de conexión al bloquear un contacto.</p> <ol style="list-style-type: none"> <li>1. La aplicación TypeMe muestra la ventana “Error” con el mensaje “Ocurrió un error al intentar bloquear el contacto. Por favor intente más tarde” y el botón “Aceptar”.</li> <li>2. El <b>Typer</b> selecciona el botón “Aceptar”.</li> <li>3. Termina caso de uso.</li> </ol>
<b>Postcondiciones:</b>	POST-01: La aplicación TypeMe bloquea al contacto seleccionado por el Typer.
<b>Incluye (relación Include):</b>	No aplica.
<b>Extiende (relación Extend):</b>	No aplica.
<b>Prioridad:</b>	Baja

## Construcción

### Selección justificada pila tecnológica

Para el desarrollo de esta aplicación se escogió un conjunto de tecnologías que ofrecen características deseables para un sistema distribuido como lo es la separación de las funcionalidades utilizando un enfoque en microservicios con el cual cada funcionalidad del sistema esta manejada por una tecnología específica, en este caso se utilizaron las siguientes herramientas para desarrollar la aplicación:

#### Aplicación de cliente rico

##### **WPF**

Windows Presentation Foundation es una herramienta la cual nos permite crear aplicaciones de escritorio enfocadas en el sistema operativo de Windows. Esto nos permite crear una aplicación de escritorio utilizando los lenguajes XAML para crear estilos gráficos de las interfaces de usuario y C# para programar el comportamiento que deben de tener cada uno de los elementos en pantalla, uniendo estos lenguajes obtenemos la posibilidad de consumir servicios y con los resultados obtenidos realizar operaciones en el lado del cliente. Es por eso por lo que se tomó como tecnología para el desarrollo del cliente de escritorio y por la familiaridad que tienen los desarrolladores con ella, esto redujo la curva de aprendizaje para la construcción.

##### **Razor Pages**

Razor Pages es un modelo complementario a MVC que nos ofrece .NET y nos brinda la ventaja de que la codificación de escenarios centrados en páginas sea más fácil y productiva que el uso de controladores y vistas. Debido a que está basado en página, ya no es necesario de un Controlador que nos retorne una vista, sino que ahora cuenta con un Modelo por páginas que se encarga de las peticiones y los datos que podemos necesitar. Gracias a estas ventajas y el contacto aún con C# para la lógica y de igual manera con Javascript, es que se tomó para el desarrollo del cliente web, facilitando la comunicación con la API Gateway porque, por una parte, desde código C# se utilizó el WebClient y por el otro, con Javascript se tuvo contacto con AJAX de JQuery. De igual manera, las vistas parciales que nos brinda esta tecnología facilitan la recarga dinámica de solo algunas partes de una página, evitando llamadas pesadas al servidor.

##### **Kotlin**

Kotlin es un lenguaje de programación desarrollado por el equipo de JetBrains y debido a que compila a bytecode JVM es muy útil para el desarrollo de aplicaciones Android, además de que se puede compilar con JavaScript. Es totalmente compatible con Java y el código de Kotlin puede ser simplemente convertido a código Java y viceversa. Esto significa que Kotlin puede usar cualquier marco, biblioteca, etc., escrito en Java. En consecuencia, esta tecnología fue tomada para

el desarrollo del cliente móvil de TypeMe, gracias a que las corrutinas de Kotlin optimizan la programación asíncrona y simplifican y agilizan las tareas comunes, idealmente para las llamadas a nuestra API y Microservicios.

## Aplicación de servicios (API)

### **C#**

El lenguaje C# es un lenguaje que trabaja bajo el paradigma orientado a objetos con el cual se nos facilita realizar la programación debido a la separación de funciones, con este lenguaje se han programado los múltiples microservicios utilizados en la aplicación, con esto logramos una homogeneidad entre estos y tanto el mantenimiento como la actualización de cada uno no requerirá que el desarrollador tenga que aprender múltiples tecnologías, además de tener un control total con las llamadas asíncronas gracias a la facilidad que el lenguaje nos brinda.

### **MySQL**

MySQL es un sistema gestor de bases de datos en la cual se pueden realizar múltiples registros de información, estas bases de datos son del tipo relacional la cual hace uso de tablas de información relacionadas por medio de llaves primarias y foráneas, en este caso se usaron múltiples bases de datos conectadas a los microservicios, terminando así con una base de datos para cada uno de los microservicios favoreciendo así a la separación de responsabilidades y a futuro la mantenibilidad de la aplicación.

### **MongoDB**

MongoDB es un sistema de base de datos NoSQL, orientado a documentos y de código abierto. En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. Es por eso que se decidió utilizarlo para una de las bases de datos de nuestros Microservicios, agilizando las consultas y con esquema flexible para el control de los datos.

### **Docker**

Docker es una tecnología que nos permite simular un tipo de máquinas virtuales dentro de un sistema, esto se logra por medio de los llamados contenedores, los cuales son sistemas aislados en los cuales se ejecutan programas establecidos y bajo cierto ambiente, esto nos permite que durante el desarrollo todos los involucrados trabajen bajo las mismas condiciones. El uso que se les dio durante el desarrollo de esta aplicación fue el de encapsular cada uno de los microservicios y sus funcionalidades es una pieza de software específica, de esta manera cada uno de los servicios funciona como un bloque que expone una funcionalidad concreta y se conecta a otros contenedores para hacer uso de su funcionalidad y en conjunto crear resultados más complejos.

## ***NodeJS***

Node es un ambiente de ejecución para el lenguaje JavaScript el cual nos permite llevar este lenguaje fuera de la web y poder hacer uso de el en un sistema local y tener acceso al sistema de archivos y múltiples funcionalidades no permitidas en internet por el riesgo de seguridad que representan. Durante el desarrollo de esta aplicación, NodeJS y Express fueron las tecnologías utilizadas para construir una API que centraliza todos los microservicios y la cual sirve como puerta de entrada para todas las operaciones que se realizan en la aplicación.

## ***Flask***

Flask es auto denominado como un microframework de Python para crear aplicaciones web, como páginas web dinamicas, APIs, entro otros. Aquí se trabaja con un lienzo en blanco, donde nos apoyaremos de sus módulos por default, y, sí así lo deseamos de bibliotecas de terceros. Gracias a que somos nosotros quienes creamos cada uno de los módulos y archivos, fue utilizado para la creación de uno de nuestros Microservicios, trabajando en conjunto con la base de datos de MongoDB, facilita y agiliza las consultas de los datos, además de los registros de ellos.

## ***SignalR:***

Esta es una tecnología que cuenta con las herramientas para crear una comunicación bidireccional entre el cliente y un servidor, esto lo hace por medio de los llamados “hubs” que establecen una serie de métodos los cuales pueden ser llamados desde el cliente y el servidor realiza una llamada de método a todos los clientes conectados a ese hub, de esta manera el envío de mensajes de nuestra aplicación se realiza en tiempo real, sin necesidad de implementar algún otro tipo de lógica mediante TCP, ya que SignalR nos asegura esta comunicación optimizada y en tiempo real.

## Estándares de codificación

En primer lugar, la mayoría de nuestros microservicios fueron desarrollados con .NET 5 por lo que el lenguaje principal fue C#. A la hora de codificar se trató de seguir el estándar de codificación que se recomienda para este lenguaje. De igual manera el IDE utilizado, Visual Studio fue de gran ayuda por las reglas que tiene definidas apegadas al estándar, como consecuencia, nos facilitó un poco más el apego al estándar. Además, como se comentó anteriormente, la API Gateway y parte del cliente web se utilizó el lenguaje de JavaScript, con el cual también se siguió un estándar ya definido para él.

A continuación, se presentan los criterios principales o las reglas que se tomaron en cuenta.

### Convención de nombres

Los nombres para las clases, paquetes, métodos, variables y constantes deben tener un nombre descriptivo con respecto al propósito de estas. Además, dicho nombre no debe de superar más de cuatro palabras. Evitar uso de adverbios y artículos.

Por ejemplo:

Si se desea calcular el área de un círculo, los nombres aceptables son:

- Clase o Interfaces: Circle
- Método: CalculateCircleArea
- Variable: radius
- Constante: PI

Los nombres **NO** aceptados son:

- Clase o Interfaces: GeometricFigure
- Método: cArea
- Variable: r
- Constante: CONST1

### Patrones de escritura de código

En esta sección se abordan las declaraciones de variables, clases e interfaces del lenguaje de C#.

1. Ponga declaraciones solo al comienzo de cada bloque. (Un bloque es cualquier código rodeado de llaves "{" y "}"). No espere para declarar variables hasta su primer uso.
2. Intente inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algún cálculo que ocurra primero.
3. Se debe de hacer una declaración por línea sin importar que sean del mismo tipo.
4. Deben seguir el estilo camelCase: Este estilo define que la primera letra de cada palabra a excepción de la primera debe ser mayúscula.

5. No incluir espacios entre el nombre de un método y los paréntesis donde se encuentran los parámetros del método.
6. El nombramiento de las clases o interfaces debe seguir el estilo PascalCase: Esta nomenclatura o estilo define que la primera letra de cada palabra debe ser mayúscula. Ejemplo: *IrregularPolygon*.
7. Se debe de hacer una declaración de variable local por línea sin importar que sean del mismo tipo.
8. En las clases e interfaces, la llave de apertura debe aparecer una línea abajo de la declaración.
9. En las clases, la llave de cierre debe comenzar en una línea nueva, alineada verticalmente con la declaración, excepto cuando es una declaración nula, el `"}` debe aparecer inmediatamente después del `"{"`.
10. El uso de estos comentarios NO se debe utilizar al menos que sea totalmente necesario. Simplemente para aclarar el porqué de la línea de código.
11. Los comentarios de línea deben de ir antes de la expresión que se desea aclarar.
12. La longitud del comentario no debe exceder la longitud del código que se comenta.
13. Evita asignar a múltiples variables el mismo valor en una sola línea.
14. No usar el operador de asignación (`=`) en un lugar donde puede ser fácilmente confundido con un operador de comparación.
15. Usar paréntesis en expresiones que contengan operadores mezclados para evitar los problemas de precedencia.

Ahora, para el lenguaje de JavaScript, se tomaron en cuenta los siguientes criterios.

1. Los nombres de funciones deben ser escritos con notación camelCase.
2. Las llaves de las funciones deben empezar en la misma línea que se declara la función y debe haber un espacio entre el paréntesis de cierre de argumentos y la llave de inicio de cuerpo de función.
3. Para mejorar la legibilidad del código los argumentos de funciones deben ser nombres con notación camelCase. Si son varios argumentos deben estar separados por coma (,) y un espacio. Si únicamente hay un argumento no hay espacios a ningún lado.
4. Todas las operaciones deben tener espacios entre los operadores y los operandos para mejorar la legibilidad del código. Si se están usando paréntesis no es necesario usar espacio entre los paréntesis y los operandos.
5. Deben usarse los comparadores estrictos (aquellos que comparan valor y tipo (`===`, `!==`) para todas las operaciones lógicas de comparación.
6. Deben ser usadas comillas simples ( `'` ) puesto de ésta manera se reducen los problemas a la hora de escapar las comillas dobles ( `"` ) usadas cuando se genera HTML y mejora la legibilidad.



7. No declarar funciones dentro de bloques (condicionales, bucles...). Si se requiere crear funciones dentro de bloques, usarlas como expresiones que se asignan a una variable.
8. Usa siempre indentado para aquello que está dentro de un bloque. Por ejemplo, el código dentro de un bucle *for* estará desplazado hacia la derecha. Usa siempre el mismo indentado.

## Reportes de análisis estático de código

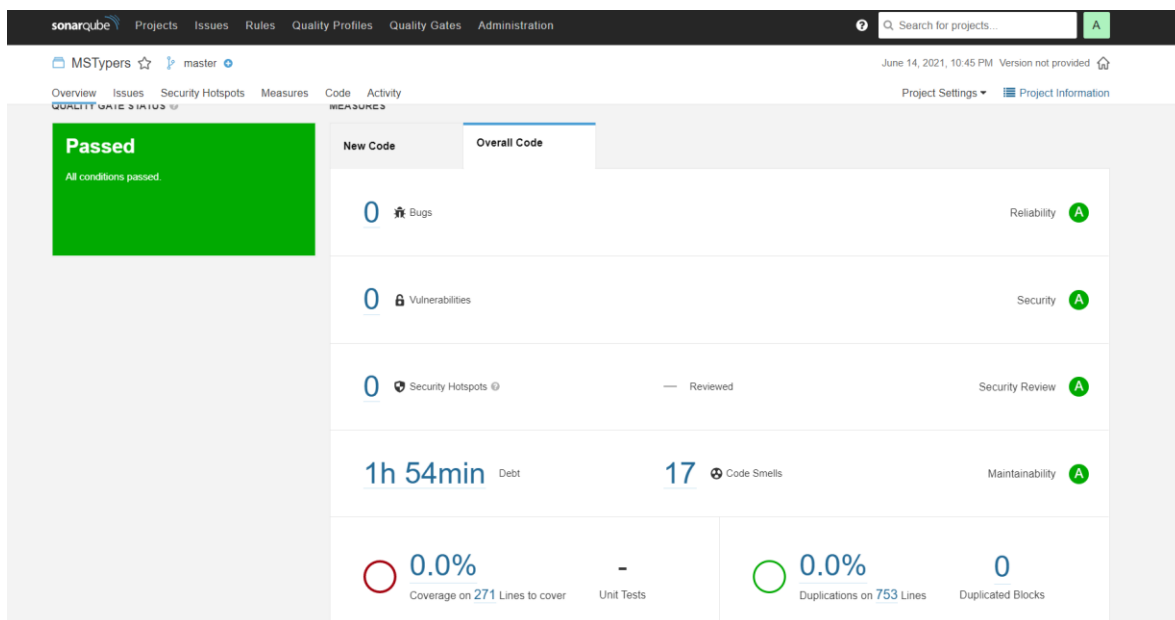
Para la realización de un análisis estático de código se utilizó la herramienta SonarQube, la cual nos permite realizar estos análisis para diferentes lenguajes de programación, entre ellos C#, Python y Javascript, los cuales fueron utilizados para nuestro desarrollo.

Este tipo de análisis es importante para el desarrollo de todo proyecto, ya que nos permite encontrar Code Smells, vulnerabilidades, bugs y hasta temas de seguridad que podrían afectar a nuestra aplicación. Además, nos proporciona el tiempo de deuda técnica para resolver dichos problemas.

A continuación, se presentan los reportes de los análisis estáticos para cada uno de los Microservicios y la API Gateway.

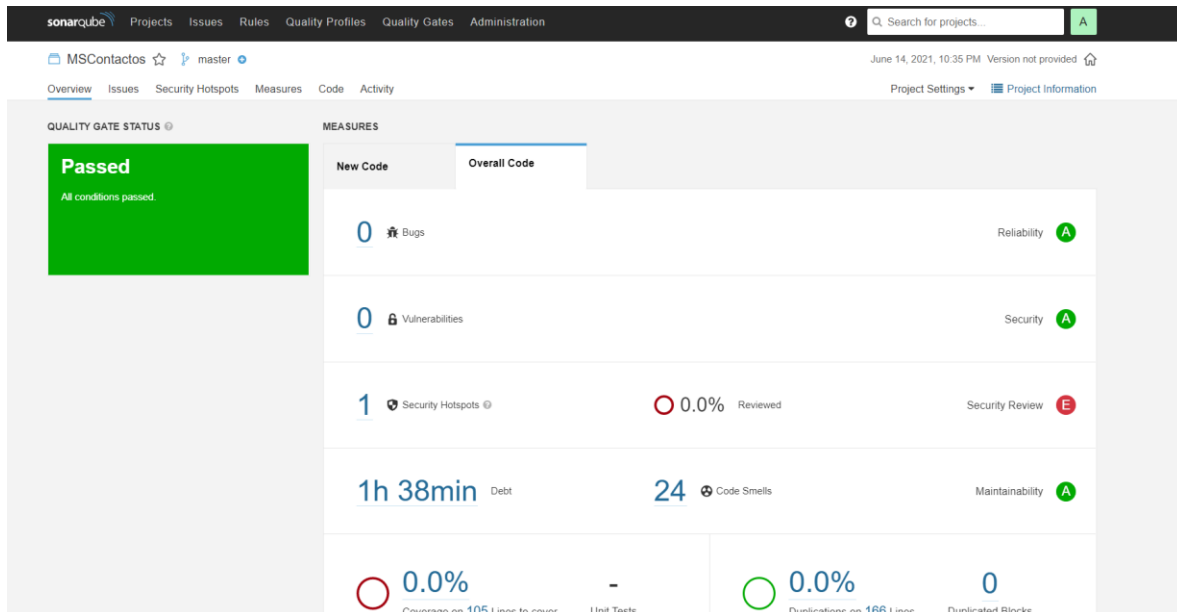
### ***MSTypers***

En la siguiente Figura se muestra el reporte final del análisis estático para el Microservicio de Typers.



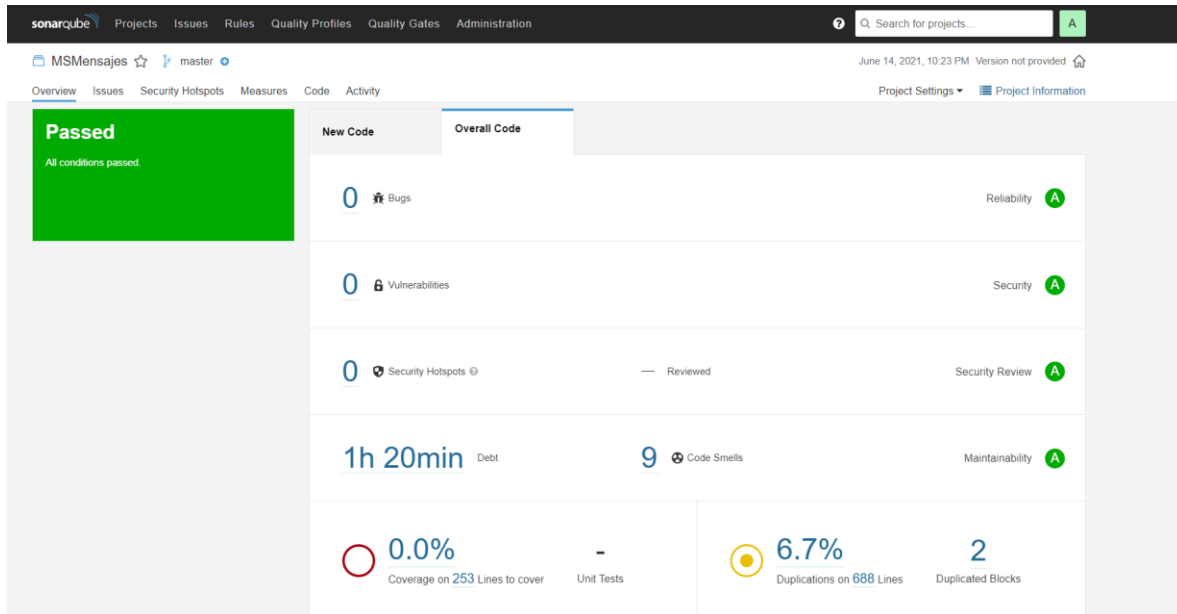
## MSMensajes

En la siguiente Figura se muestra el reporte del análisis estático para el Microservicio de MSMensajes.



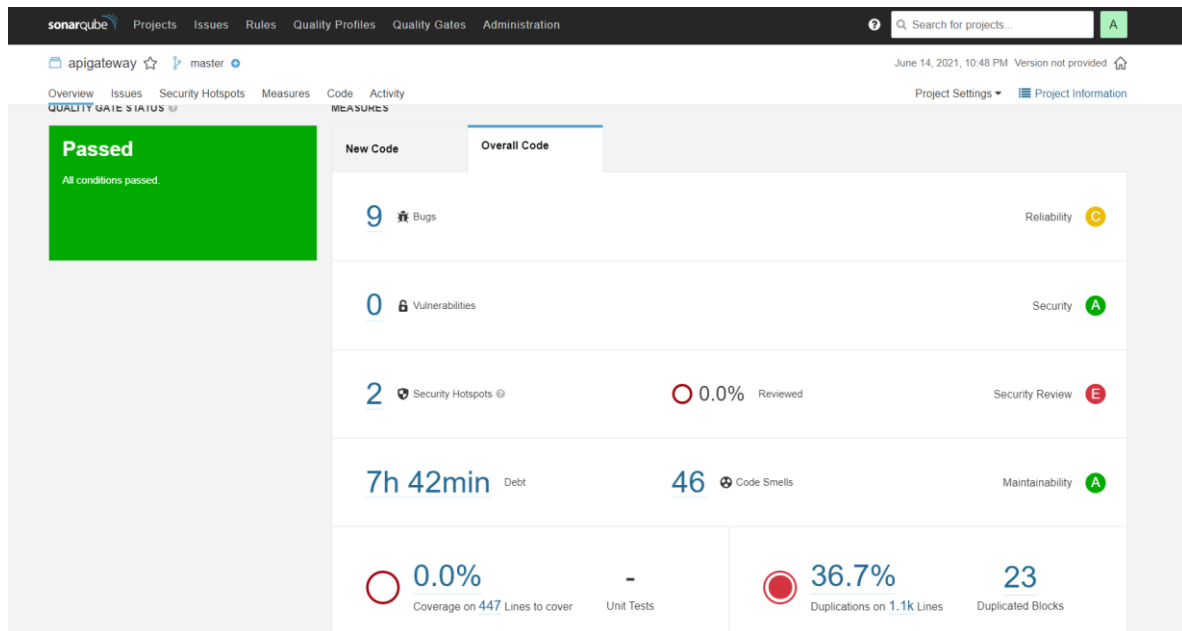
## MSContactos

En la siguiente Figura se muestra el reporte del análisis estático para el Microservicio de Contactos.



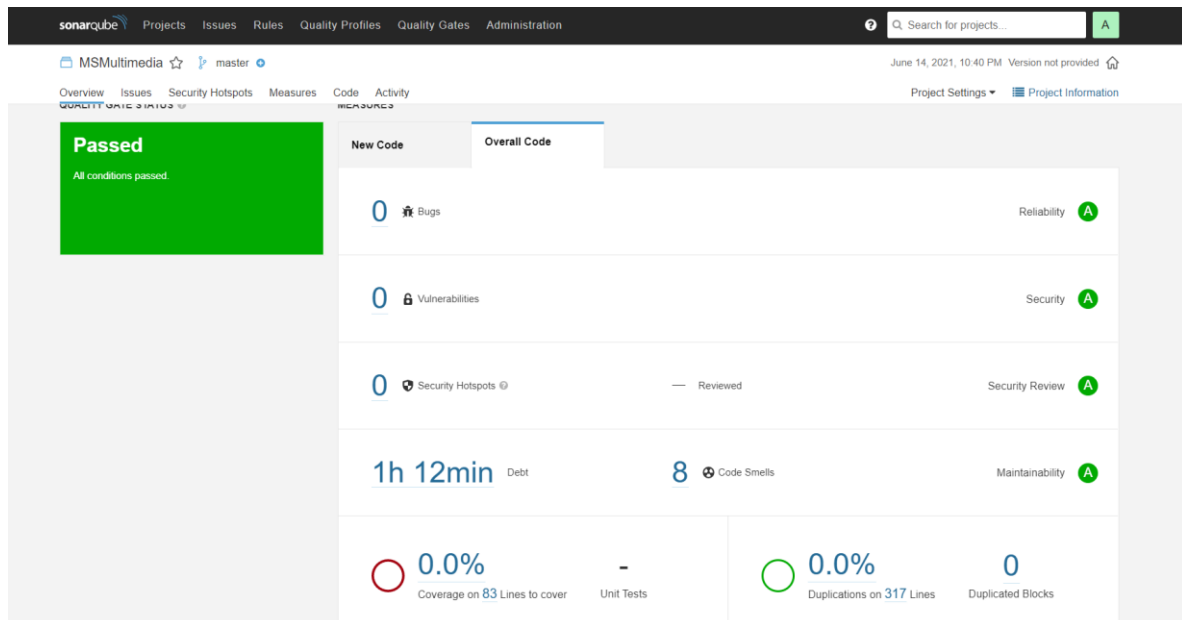
## MSMultimedia

En la siguiente Figura se muestra el reporte del análisis estático para el Microservicio de Multimedia.



## API Gateway

En la siguiente Figura se muestra el reporte final del análisis estático para la API Gateway de la aplicación TypeMe.



Como se mostró en las Figuras anteriores, el reporte de cada proyecto presentó un estatus de “Pasó”, lo cual nos indica que los problemas presentados en cada uno son menores y pueden ser resueltos sin mayor problema.

En promedio, los Code Smells que presentó cada proyecto fue de 9 y una deuda técnica de aproximadamente 1 hora y media. Solo en el proyecto de la API Gateway se presentaron problemas de seguridad que no tienen prioridad alta y está relacionado con los CORS. Sin embargo, esto se habilitó con fines de aprendizaje y el desarrollo de una aplicación para proyectos escolares.

## Prácticas de construcción realizadas

Como primer acercamiento a las peticiones web y el desarrollo de una API RESTful, se tomaron en consideración los métodos HTTP básicos que nos permitieron el acceso y manejo de los datos que nos ofrecen los Microservicios a través de la API.

HTTP define un conjunto de métodos de petición para indicar cierta acción que se desea realizar para un recurso determinado que nosotros pongamos a disposición. Aunque estos también pueden ser sustantivos, también son llamados *HTTP verbs*. Los métodos que fueron utilizados para la construcción del proyecto fueron los siguientes:

### **GET**

El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos. A diferencia de los otros métodos, a este no se le especifica un cuerpo para mandar datos. En caso de que se necesite mandar un dato, como un identificador para obtener la información de un solo recurso, se le especifica mediante variables dentro de la misma *query* o consulta. Por ejemplo, **GET /computadoras?id=344**.

De igual manera, existen escenarios en los cuales se pasaban esos datos mediante la ruta de la API, por ejemplo, **GET /computadoras/344**.

### **POST**

El método POST se utiliza para enviar una entidad a un recurso en específico, causando generalmente un cambio en el estado o efectos secundarios en el servidor.

En este método o petición siempre fue consumido enviándole a la API la información necesaria especificada en el *body* de la petición. Por lo general, dentro de la cabecera Content-Type se le especificó el tipo de cuerpo que es *application/json*, a excepción de la petición para enviar una multimedia a la API, en esta ocasión fue manejada con *multipart/form-data*.

Este método es siempre recomendado cuando de enviar datos se trata, por su seguridad al enviarlos. Es por eso por lo que se tomó en cuenta para el registro de un Typer, el envío de los mensajes y creación de los grupos, entre otros.

### **PUT**

El modo PUT reemplaza todas las representaciones actuales del recurso de destino con los datos de la petición. La diferencia entre el método PUT y el método POST es que PUT es un método idempotente: llamarlo una o más veces de forma sucesiva tiene el mismo efecto (sin efectos secundarios), mientras que una sucesión de peticiones POST idénticas pueden tener efectos adicionales, como enviar una orden varias veces.

Este método fue utilizado con el mismo fin que este tiene, actualizar la información de un recurso, como por ejemplo el Typer y la información de un grupo. Al igual que el método POST, la información correspondiente se le envía mediante el *body* de la petición y con el Content-Type de *application/json*.

Por otro lado, la finalidad de este proyecto fue el acercamiento a los microservicios y dada su naturaleza o finalidad, las funcionalidades fueron distribuidas de manera que se desarrollaron cuatro: un microservicio para la administración del Typer, otro para los Contactos, uno para los Mensajes y finalmente otro para la Multimedia. Cada uno de ellos tienen sus bases de datos y la manera en que se relacionan es guardando una referencia mediante un identificador en las tablas correspondientes. Esto nos ayuda cuando, mediante la API Gateway, se consultan los datos y son necesarios más de un microservicio, ya que, la información de uno es utilizada para obtener la información de otro. De esta manera, estamos juntando o relacionando diferentes bases de datos que conviven por sí solas y enviamos dicha información al cliente, ya con el dominio correspondiente.

Como se comentó en la pila de tecnología utilizada, el uso de Docker fue fundamental para el despliegue y con esto, se utilizaron prácticas recomendadas por la documentación, como la definición de las variables de entorno, las dependencias entre los contenedores para que, si uno aún no está activo, el que dependa de él tampoco se active. La construcción del contenedor mediante el Dockerfile ubicados en el repositorio y las ramas correspondientes a los microservicios. También, en los contenedores donde se encuentran las bases de datos, se utilizaron los volúmenes para la persistencia de los datos cuando dichos contenedores sean reiniciados.

## Pruebas

En esta sección se presenta el plan de Pruebas para la API Gateway, al igual que los casos de pruebas y resultados obtenidos.

### Plan de pruebas para la Aplicación de servicios (API)

#### Proyecto

La aplicación TypeMe, en todos sus clientes, tiene la finalidad de mantener y facilitar la comunicación entre dos o más personas que formen parte de la aplicación. Para esto, TypeMe permite el envío de mensajes a través de la creación de Grupos de al menos dos integrantes, el envío de imágenes, añadir y consultar los contactos, esto con el username que cada *Typer* haya definido. Además, la aplicación permite la actualización de los datos de la cuenta.

La aplicación está pensada para ser usada desde un navegador web, como Chrome, Microsoft Edge y Mozilla Firefox en sus versiones más recientes. También podrá ser utilizada en computadoras laptops o de escritorios con sistema operativo de Windows, a partir de la versión 7 en adelante. De igual manera, podrá ser utilizada en dispositivos móviles con sistema operativo Android en sus versiones más recientes.

Para que las aplicaciones clientes de TypeMe puedan funcionar, fue necesario el desarrollo de un *back-end* donde se diseñaron cuatro Microservicios y una API Gateway para centralizar la información de ellos. Es en este último donde los clientes tienen comunicación directa a través del protocolo HTTP.

#### Elementos de prueba

Las pruebas realizadas se enfocan a las siguientes funcionalidades específicas:

- Registro de usuarios
  - Estas pruebas buscan verificar que el registro de los usuarios se realice correctamente ya sea en caso de que la información es correcta o que no se pueda realizar un registro en caso de que la información no esté completa.
- Administración de perfil
  - Las pruebas realizadas estarán enfocadas en el múltiple cambio de información que un usuario puede realizar sobre su perfil como lo es el cambio de nombre de usuario, cambio de imagen de perfil o cambios de correo.
- Creación de grupos
  - Las pruebas realizadas para los grupos están enfocadas en la adición de nuevos miembros a un grupo, así como la notificación de los nuevos mensajes a todos sus integrantes.
- Envío de imágenes



- Estas pruebas ayudarán a verificar si el envío de imágenes a través del chat es óptimo pudiendo mantener el registro de estas dentro del microservicio y obteniéndolas por medio de la API.

#### Alcance de la prueba

Para realizar las pruebas necesarias sobre los elementos anteriormente descritos se ha tomado como alcance y enfoque la API, debido a que es el elemento que centraliza todo el comportamiento de la aplicación utilizando los múltiples microservicios programados.

Dentro de la API se someterá a prueba los métodos creados para manejar las peticiones de tipo GET, POST y PUT las cuales son las más usadas dentro del sistema y con esto garantizamos que el comportamiento esperado de la aplicación será probado bajo condiciones correctas.

Los métodos que serán sometidos a pruebas serán los siguientes:

#### **GET**

/obtenerMultimedia

/obtenerMensajes/:idGrupo

/misGrupos/:idTyper

/obtenerContactos/:idTyper

#### **POST**

/registrarMultimedia

/crearGrupo

/agregarIntegrantes/:idGrupo

/enviarMensaje

/registrarTyper

/loginTyper

/infoTyper

/ActualizarImagenPerfil

/agregarContacto

#### **PUT**

/actualizarInfoTyper

/actualizarGrupo/:idGrupo

/salirDeGrupo

/actualizarContraseña

#### Requisitos de datos de prueba

En la siguiente tabla se muestran los requisitos de datos de prueba que se definieron para la ejecución del plan de pruebas.

ID	Descripción
RDP-01	Los datos deben ser lo más cercano al formato real que se maneja en las aplicaciones de chat.
RDP-02	Los datos pueden violar su formato siempre y cuando los casos de prueba así lo ameriten.
RDP-03	Los datos utilizados NO deben ser iguales a los datos usados en sistemas actuales para proteger la privacidad. Pero siguiendo el RDP-01.
RDP-04	Los datos utilizados en las pruebas serán almacenados en una base de datos de prueba siguiendo el esquema original.

#### Requisitos de entorno de prueba

Para la ejecución de las pruebas se deben tomar dos factores importantes con respecto a los recursos necesarios: el hardware y software. En la siguiente tabla se muestran los requisitos de hardware (REH) necesarios para llevar a cabo las pruebas.

ID	Descripción
REH-01	Contar con una computadora de escritorio o laptop.
REH-02	Procesador de mínimo 1.8 GHz de velocidad de procesamiento.
REH-03	Memoria RAM mínima de 2 GB (8 GB recomendados)
REH-04	Almacenamiento disponible en disco duro de al menos 5 GB

En la siguiente tabla se muestran los requisitos de software (RES) necesarios para llevar a cabo las pruebas.

ID	Descripción
RES-01	Contar con alguno de los siguientes sistemas operativos (recomendado 64 bit): <ul style="list-style-type: none"><li>Windows 10 versión 1703 o posterior: Home, Professional, Education y Enterprise (LTSC y S no son compatibles)</li><li>Windows Server 2019: Estándar y centro de datos</li><li>Windows Server 2016: Estándar y centro de datos</li></ul>

	<ul style="list-style-type: none"> <li>• Windows 8.1 (con actualización 2919355): Core, Professional y Enterprise</li> <li>• Windows Server 2012 R2 (con actualización 2919355): Essentials, Standard, Datacenter</li> <li>• Windows 7 SP1 (con las últimas actualizaciones de Windows): Home Premium, Professional, Enterprise, Ultimate</li> </ul>
RES-02	Contar con el software Postman para crear consultas hacia los servicios de la API
RES-03	Contar con el software MySql donde se almacenarán los datos para las pruebas
RES-04	Contar con el software Docker para ejecutar los microservicios a probar

#### Roles, actividades y responsabilidades

El equipo está integrado por Ángel de Jesús Juárez García y Sammy Guadarrama Sánchez, ambos han colaborado en la construcción del software por lo que cuentan con un conocimiento general de la estructura del proyecto y de esta manera se les facilitara realizar las pruebas diseñadas.

Los roles que se manejaran en el equipo son los de "Diseñador de pruebas" y Tester y estarán repartidos como se muestran en la siguiente tabla:

Miembro del equipo	Rol
Ángel de Jesús Juárez García	Diseñador de pruebas y Tester
Sammy Guadarrama Sánchez	Diseñador de pruebas y Tester

Como se mostró en la tabla anterior, los dos integrantes del equipo cumplen con los roles debido a la carga de trabajo y el tiempo limitado debido al plan de clases de las Experiencias Educativas.

### Procedimiento de prueba

Para la realización de los casos de prueba se tomarán en cuenta el siguiente formato:

ID	Descripción	Precondiciones
[1]	[2]	[3]
<b>Entrada</b>	<b>Resultado esperado</b>	<b>Resultado obtenido</b>
[4]	[5]	[6]
<b>Estado</b>	<b>Fecha</b>	<b>Responsable</b>
[7]	[8]	[9]

A continuación, se describe cada uno de los puntos de la tabla:

[1] Identificador único para el caso de prueba.

[2] Descripción lo suficientemente detallada para definir los objetivos del caso de prueba.

[3] Las precondiciones que se deben cumplir para llevar a cabo el caso de prueba.

[4] Las entradas necesarias para llevar a cabo el caso de prueba.

[5] Los resultados esperados para el caso de prueba.

[6] El resultado obtenido por la ejecución del caso de prueba.

[7] Estado de la prueba, los cuales pueden ser: Pasó, No pasó, Sin ejecutar.

[8] Fecha en que se realizó el caso de prueba.

[9] Nombre del responsable del caso de prueba.

Para el desarrollo de los casos de prueba, se tomarán los métodos definidos en el alcance y se probarán los caminos que pueden tomar las llamadas a esos métodos. Para esto, se utilizará la herramienta *Postman*, que es una plataforma de colaboración para el desarrollo de API.

Con todo esto, se harán las llamadas a los métodos con la información necesaria según sea el caso de prueba, finalmente se realizará una captura de pantalla sobre la respuesta que, de la API, dicho resultado está especificado en la sección de Resultados del plan de pruebas.

## Casos de prueba

A continuación, se presentan los casos de prueba (CP) para los métodos definidos en nuestro alcance de las pruebas.

### Métodos GET

*GET obtenerMensajes/:idGrupo*

#### CP\_obtenerMensajes\_01

ID	Descripción	Precondiciones
CP_obtenerMensajes_01	Obtener los mensajes existentes de un grupo, con idGrupo existente y correcto	Debe existir el grupo con id=1 Deben existir mensajes relacionados al grupo con id=1
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/obtenerMensajes/1	Status = true, Message = "Mensajes encontrados", result = arreglo de objetos de mensajes	Status = true, Message = "Mensajes encontrados", result = arreglo de objetos de mensajes
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

#### CP\_obtenerMensajes\_02

ID	Descripción	Precondiciones
CP_obtenerMensajes_02	Consultar los mensajes de un grupo existente, pero no cuenta con mensajes relacionados a él	Debe existir el grupo No deben existir mensajes relacionados al grupo
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/obtenerMensajes/2	status = false, data con message = "No hay mensajes en el grupo", result = null	status = false, data con message = "No hay mensajes en el grupo", result = null
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

### CP\_obtenerMensajes\_03

ID	Descripción	Precondiciones
CP_obtenerMensajes_03	Consultar los mensajes de un grupo no existente con id incorrecto	Pueden no existir grupos registrados
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/obtenerMensajes/-2	status = false, data con message = "Id del grupo incorrecto", result = null	status = false, data con message = "Id del grupo incorrecto", result = null
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

### CP\_obtenerMensajes\_04

ID	Descripción	Precondiciones
CP_obtenerMensajes_04	Consultar los mensajes de un grupo con la query incorrecta. No se especifica el id	Pueden no existir grupos registrados
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/obtenerMensajes/	"success": false, "msg": "This route does not exist"	"success": false, "msg": "This route does not exist"
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

GET /misGrupos/:idTyper

CP\_misGrupos\_01

ID	Descripción	Precondiciones
CP_misGrupos_01	Obtener los grupos de un Typer existente, con id correcto	Debe existir un typer registrado Deben existir grupos relacionados al Typer
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/misGrupos/8121d387-f063-454f-a764-d0b29385741f	{ "status": true, "message": "Grupos encontrados", "result": arreglo de objetos de grupos }	{ "status": true, "message": "Grupos encontrados", "result": arreglo de objetos de grupos }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

CP\_misGrupos\_02

ID	Descripción	Precondiciones
CP_misGrupos_02	Obtener los grupos de un Typer no existente	No debe existir un Typer con el id de la prueba Pueden no existir grupos
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/misGrupos/8121d387-f063-454f-a764-d0b29385741f2	{ "status": false, "message": "Grupo(s) no encontrado(s) o inexistente(s)", "result": null }	{ "status": false, "message": "Grupo(s) no encontrado(s) o inexistente(s)", "result": null }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

### CP\_misGrupos\_03

ID	Descripción	Precondiciones
CP_misGrupos_03	Consultar los grupos de un Typer existente, pero sin grupos relacionados a él	Debe existir un Typer registrado, podría tener el id de la prueba, sino, especificar el id. No deben existir grupos donde pertenezca el Typer
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/misGrupos/b202eeb5-5f4e-4afa-b241-e3fb0d767997	{ "status": false, "message": "Grupo(s) no encontrado(s) o inexistente(s)", "result": null }	{ "status": false, "message": "Grupo(s) no encontrado(s) o inexistente(s)", "result": null }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

*GET /obtenerContactos/:idTyper*

### CP\_obtenerContactos\_01

ID	Descripción	Precondiciones
CP_obtenerContactos_01	Obtener los contactos de un Typer existente	Debe existir un Typer registrado, podría tener el id de la prueba, sino, especificar el id. Debe tener contactos el Typer.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /typers/obtenerContactos/8121d387-f063-454f-a764-d0b29385741f	{ "status": true, "message": "Contactos encontrados", "result": arreglo de objetos de contactos }	{ "status": true, "message": "Contactos encontrados", "result": arreglo de objetos de contactos }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez



# CP\_obtenerContactos\_02

ID	Descripción	Precondiciones
CP_obtenerContactos_02	Obtener los contactos de un Typer existente pero el typer no tiene contactos	Debe existir un Typer registrado, podría tener el id de la prueba, sino, especificar el id. No debe tener contactos el Typer.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /typers/obtenerContactos/ff2e5ee8-0712-48ec-a1dd-6825008977ec	{ "status": false, "message": "Contactos no encontrados", "result": [] }	{ "status": false, "message": "Contactos no encontrados", "result": [] }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

# CP\_obtenerContactos\_03

ID	Descripción	Precondiciones
CP_obtenerContactos_03	Obtener los contactos de un Typer con id incorrecto	Puede no existir un Typer registrado, podría tener el id de la prueba, sino, especificar el id.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /typers/obtenerContactos/ff2e5ee8-0712-48ec-a1dd-6825008977ec0	{ "status": false, "message": "Contactos no encontrados", "result": [] }	{ "status": false, "message": "Contactos no encontrados", "result": [] }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

GET /obtenerMultimedia

CP\_ObtenerMultimedia\_01

ID	Descripción	Precondiciones
CP_obtenerMultimedia_01	Se obtiene una ruta de imagen del microservicio para ser consultada	La consulta debe de contar con idValido
Entrada	Resultado esperado	Resultado obtenido
Consulta: http://localhost:4000/mensajes/obtenerMultimedia/?idMMultimedia=4ab911f2-6f57-4c02-8d61-b37699cf0ddf	Objeto con la url de la multimedia	{ "urlMultimedia": "http://localhost:3325/multimedia/obtenerMultimedia?idMultimedia=undefined" }
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

CP\_ObtenerMultimedia\_02

ID	Descripción	Precondiciones
CP_obtenerMultimedia_02	Se obtiene un error debido a	La consulta debe de contar con id invalido

	que el id de la imagen buscada no existe	
<b>Entrada</b>	<b>Resultado esperado</b>	<b>Resultado obtenido</b>
Consulta: http://localhost:4000/mensajes/obtenerMultimedia/?idMMultimedia=4ab911f2	Status = false	Status = false
<b>Estado</b>	<b>Fecha</b>	<b>Responsable</b>
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

## Métodos POST

POST /registrarMultimedia

CP\_registrarMultimedia \_01

ID	Descripción	Precondiciones
CP_registrarMultimedia_01	Se registra un archivo multimedia en el microservicio	La imagen registrada debe de estar en formato jpg  La consulta debe de contener un form data con la imagen a guardar
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/mensajes/registrarMultimedia?idTyper=ac8794cd-6ffe-46dc-b2b6-836e03fc0de5">http://localhost:4000/mensajes/registrarMultimedia?idTyper=ac8794cd-6ffe-46dc-b2b6-836e03fc0de5</a>	Status = true,	Status = true
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

## POST /crearGrupo

### CP\_crearGrupo\_01

ID	Descripción	Precondiciones
CP_crearGrupo_01	Crear grupo con información correcta	Deben existir los Typers a agregar al grupo
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/crearGrupo Body: { "nombre": "Grupo prueba unitaria", "descripcion": "Otro grupo de prueba", "pertenece": [ { "idTyper": "8121d387-f063-454f-a764-d0b29385741f" }, { "idTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275" } ] }	{ "status": true, "message": "Nuevo grupo creado", "result": grupo creado }	{ "status": true, "message": "Nuevo grupo creado", "result": grupo creado }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

### Problemas detectados

El método del microservicio no verifica que sean al menos dos integrantes al momento de registrar el grupo.

## CP\_crearGrupo\_02

ID	Descripción	Precondiciones
CP_crearGrupo_02	Crear grupo con información incorrecta, no se especifica la descripción y el nombre	Puede no existir los Typers a agregar al grupo
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/crearGrupo Body: { "nombre": "", "descripcion": "" }	{ "status": true, "message": "Nuevo grupo creado", "result": grupo creado }	{ "status": true, "message": "Nuevo grupo creado", "result": grupo creado }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

## CP\_crearGrupo\_03

ID	Descripción	Precondiciones
CP_crearGrupo_03	Crear grupo con información incorrecta, no se especifica el body de la petición	Puede no existir los Typers a agregar al grupo
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/crearGrupo	{ "status": true, "message": "Nuevo grupo creado", "result": grupo creado }	{ "status": true, "message": "Nuevo grupo creado", "result": grupo creado }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

POST /agregarIntegrantes/:idGrupo

CP\_agregarIntegrantes\_01

ID	Descripción	Precondiciones
CP_agregarIntegrantes_01	Agregar integrantes al grupo con Typers existentes y grupo existente	Debe existir un grupo registrado Deben existir los Typers registrados. No deben pertenecer al grupo.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/agregarIntegrantes/5 Body [ { "idTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f02" } ]	{ "status": true, "message": "Integrante(s) agregado(s)", "result": arreglo de la información de los nuevos integrantes }	{ "status": true, "message": "Integrante(s) agregado(s)", "result": arreglo de la información de los nuevos integrantes }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

CP\_agregarIntegrantes\_02

ID	Descripción	Precondiciones
CP_agregarIntegrantes_02	Agregar integrantes al grupo con Typers existentes y grupo NO existente	No debe existir el grupo al cual se le quiere agregar integrantes Deben existir los Typers registrados.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/agregarIntegrantes/50 Body [ { "idTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f02" } ]	{ "status": false, "message": "No existe el grupo con el id especificado.", "result": null }	{ "status": false, "message": "No existe el grupo con el id especificado.", "result": null }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

### CP\_agregarIntegrantes\_03

ID	Descripción	Precondiciones
CP_agregarIntegrantes_03	Agregar integrantes al grupo con Typers NO existentes y grupo existente	Debe existir el grupo al cual se le quiere agregar integrantes No deben existir los Typer registrados.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/agregarIntegrantes/5 Body <pre>[   {     "idTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f020"   } ]</pre>	<pre>{   "status": false,   "message": "No existe el Typer especificado",   "result": null }</pre>	<pre>{   "status": true,   "message": "Integrante(s) agregado(s)",   "result": [     {       "IdGrupo": 5,       "IdTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f020",       "IdGrupoNavigation": null     }   ] }</pre>
Estado	Fecha	Responsable
No pasó	15/06/21	Sammy Guadarrama Chávez

#### Problemas detectados:

No se verifica que los Typers se encuentren registrados previamente. Esto es porque nunca se hace una validación del idTyper al MSTypers.



#### CP\_agregarIntegrantes\_04

ID	Descripción	Precondiciones
CP_agregarIntegrantes_04	Agregar integrantes al grupo con Typers existentes y grupo existente, pero sin especificar el idTyper	Debe existir el grupo al cual se le quiere agregar integrantes Pueden no existir los Typer registrados.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/agregarIntegrantes/5 Body [ ] ]	{ "status": false, "message": "No se puede agregar el nuevo integrante", "result": null }	{ "status": false, "message": "No se puede agregar el nuevo integrante", "result": null }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

#### POST /enviarMensaje

#### CP\_enviarMensaje\_01

ID	Descripción	Precondiciones
CP_enviarMensaje_01	Enviar nuevo mensaje con información correcta, sin multimedia, a un grupo existente.	Debe existir el grupo al cual se le quiere enviar el mensaje Debe existir el Typer que enviará el mensaje y debe pertenecer al grupo.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/enviarMensaje Body { "contenido": "Mensaje pruebas manuales", "idGrupo": 5, "idTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275", "idMultimedia": "" }	{ "status": true, "message": "Mensaje enviado", "result": objeto del mensaje enviado }	{ "status": true, "message": "Mensaje enviado", "result": objeto del mensaje enviado }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

### Problemas detectados

No se verifica si el Typer, con el idTyper, pertenece al grupo.

#### CP\_enviarMensaje\_02

ID	Descripción	Precondiciones
CP_enviarMensaje_02	Enviar nuevo mensaje con información correcta con multimedia, a un grupo existente.	Debe existir el grupo al cual se le quiere enviar el mensaje Debe existir el Typer que enviará el mensaje y debe pertenecer al grupo. Se debe haber registrado una multimedia previamente.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/enviarMensaje Body { "contenido": "Mensaje pruebas manuales", "idGrupo": 5, "idTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275", "idMultimedia": "346d6a15-6347-4ac7-91bf-d9d1da2c28ff" }	{ "status": true, "message": "Mensaje enviado", "result": objeto del mensaje enviado, con el id multimedia }	{ "status": true, "message": "Mensaje enviado", "result": objeto del mensaje enviado, con el id multimedia }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

## CP\_enviarMensaje\_03

ID	Descripción	Precondiciones
CP_enviarMensaje_03	Enviar nuevo mensaje con información correcta, a un id grupo no existente.	No debe existir el grupo al cual se le quiere enviar el mensaje Debe existir el Typer que enviará el mensaje y debe pertenecer al grupo.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/enviarMensaje Body <pre>{   "contenido": "Mensaje pruebas manuales",   "idGrupo": 50,   "idTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275",   "idMultimedia": "" }</pre>	<pre>{   "status": false,   "message": "No existe ningún grupo con el id",   "result": null }</pre>	<pre>{}</pre>
Estado	Fecha	Responsable
No pasó	15/06/21	Sammy Guadarrama Chávez

### Problemas detectados:

No se realiza la respuesta correcta desde la API cuando el Microservicio le responde que no se puede registrar el mensaje porque no existe el grupo.

POST /registrarTyper

CP\_registrarTyper\_01

ID	Descripción	Precondiciones
CP_registrarTyper_01	Se registra un nuevo Typer en el sistema correctamente	Los datos ingresados cuentan con la información correcta  No existe un Typer con el mismo nombre de usuario
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/registrarTyper">http://localhost:4000/typers/registrarTyper</a>  { "Username": "Nuevo miembro", "Estado": "Estado generico", "FotoDePerfil": "", "Estatus": 1, "Contrasenia": [ { "Contrasenia1": "admin" } ], "Correos": [ { "direccion": "miCorreo@hotmail.com", "esPrincipal": "1" }, { "direccion": "miCorreo2@hotmail.com", "esPrincipal": "0" } ] }	Status = true,	Status = true
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

# CP\_ registrarTyper \_02

ID	Descripción	Precondiciones
CP_registrarTyper_02	No se logra registrar el nuevo Typer debido a la existencia de otro con el mismo nombre	Los datos ingresados cuentan con la información correcta  Ya existe un Typer con el mismo nombre de usuario
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/registrarTyper">http://localhost:4000/typers/registrarTyper</a>  <pre> {   "Username": "Nuevo miembro",   "Estado": "Estado generico",   "FotoDePerfil": "",   "Estatus": 1,   "Contrasenia": [     {       "Contrasenia1": "admin"     }   ],   "Correos": [     {       "direccion": "miCorreo@hotmail.com",       "esPrincipal": "1"     },     {       "direccion": "miCorreo2@hotmail.com",       "esPrincipal": "0"     }   ] }</pre>	Status = false	Status = false
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

POST /loginTyper

CP\_loginTyper \_01

ID	Descripción	Precondiciones
CP_loginTyper _01	Se realiza la autenticación del usuario para iniciar sesión	La contraseña es correcta  El nombre de usuario existe
Entrada	Resultado esperado	Resultado obtenido
Consulta: http://localhost:4000/typers/loginTyper { "identificadorTyper": "Nuevo miembro", "InformacionComplementaria": "admin", }	Status = true	Status = true
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

CP\_loginTyper \_02

ID	Descripción	Precondiciones
CP_loginTyper _02	Se falla el login debido a la información incorrecta	La contraseña es incorrecta
Entrada	Resultado esperado	Resultado obtenido
Consulta: http://localhost:4000/typers/loginTyper { "identificadorTyper": "Nuevo miembro", "InformacionComplementaria": "adminDos", }	Status = false	Status = false
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

### CP\_loginTyper \_03

ID	Descripción	Precondiciones
CP_loginTyper _03	Se falla el login debido a la información incorrecta	El nombre de usuario no existe
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/loginTyper">http://localhost:4000/typers/loginTyper</a> { "identificadorTyper": "Nuevo miembro dos", "InformacionComplementaria": "admin", }	Status = false	Status = false
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

### POST /infoTyper

### CP\_infoTyper \_01

ID	Descripción	Precondiciones
CP_infoTyper _01	Se obtiene la información de un typer por medio de su nombre de usuario	El typer debe de existir
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/infoTyper">http://localhost:4000/typers/infoTyper</a> { "identificadorTyper": "Nuevo miembro", "modificadorDeMetodo": "usuario" }	Status = true	Status = true
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

### CP\_ infoTyper \_02

ID	Descripción	Precondiciones
CP_ infoTyper _02	Se obtiene la información de un typer por medio de su nombre de su id	El id del typer debe de existir
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/infoTyper">http://localhost:4000/typers/infoTyper</a>  { "identificadorTyper": " ac8794cd-6ffe-46dc-b2b6-836e03fc0de5", "modificadorDeMetodo": "id" }	Status = true	Status = true
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

### CP\_ infoTyper \_03

ID	Descripción	Precondiciones
CP_ infoTyper _03	No se obtiene la información del typer debido a que no existe	El usuario del typer no existe en los registros
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/infoTyper">http://localhost:4000/typers/infoTyper</a>  { "identificadorTyper": "1234", "modificadorDeMetodo": "usuario" }	Status = false	Status = false
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García



# CP\_ infoTyper \_04

ID	Descripción	Precondiciones
CP_ infoTyper _04	No se obtiene la información del typer debido a que no existe	El id del typer no existe en los registros
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/infoTyper">http://localhost:4000/typers/infoTyper</a>  { "identificadorTyper": "1234", "modificadorDeMetodo": "id" }	Status = false	Status = false
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

## POST /actualizarImagenDePerfil

# CP\_ actualizarImagenDePerfil\_01

ID	Descripción	Precondiciones
CP_ actualizarImagenDePerfil_01	Se actualiza la imagen de perfil de un typer	El id del typer existe en los registros
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/actualizarImagenPerfil?idTyper=04e402b1-70fe-4285-926b-27d00a8a7de9">http://localhost:4000/typers/actualizarImagenPerfil?idTyper=04e402b1-70fe-4285-926b-27d00a8a7de9</a>	Status = true	Status = true
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

## CP\_ actualizarImagenDePerfil\_02

ID	Descripción	Precondiciones
CP_ actualizarImagenDePerfil_02	No se logra a actualizar la imagen de perfil debido a que el id no existe	El id del typer no existe en los registros
Entrada	Resultado esperado	Resultado obtenido
Consulta: http://localhost:4000/typers/actualizarImagenPerfil?idTyper=1234	Status = false	Status = false
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

## POST /agregarContacto

### CP\_agregarContacto\_01

ID	Descripción	Precondiciones
CP_agregarContacto_01	Agregar un contacto con username existente, al igual que el idTyper del que va a registrar el contacto es existente	Deben existir al menos dos Typer registrados
Entrada	Resultado esperado	Resultado obtenido
Consulta: /typers/agregarContacto Body { "idTyper": "8121d387-f063-454f-a764-d0b29385741f", "contacto": "user_5" }	{ "status": true, "message": "Contacto agregado", "result": objeto del contacto creado }	{ "status": true, "message": "Contacto agregado", "result": objeto del contacto creado }
Estado	Fecha	Responsable

Pasó	15/06/21	Sammy Guadarrama Chávez
------	----------	----------------------------

#### CP\_agregarContacto\_02

ID	Descripción	Precondiciones
CP_agregarContacto_02	Agregar un contacto con username existente pero que ya es contacto del Typer	Deben existir al menos dos Typer registrados. El contacto a agregar ya debe ser contacto del typer
Entrada	Resultado esperado	Resultado obtenido
Consulta: /typers/agregarContacto Body { "idTyper": "8121d387-f063-454f-a764-d0b29385741f", "contacto": "user_5" }	{ "status": false, "message": "El contacto ya se encuentra agregado", "result": null }	{ "status": false, "message": "El contacto ya se encuentra agregado", "result": null }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

#### CP\_agregarContacto\_03

ID	Descripción	Precondiciones
CP_agregarContacto_03	Agregar un contacto con username NO existente	No debe existir el Typer a agregar.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /typers/agregarContacto Body { "idTyper": "8121d387-f063-454f-a764-d0b29385741f", "contacto": "user_50" }	{ "status": false, "message": "No se encuentra el usuario buscado", "result": null }	{ "status": false, "message": "No se encuentra el usuario encontrado", "result": null }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

#### Problemas detectados

No se verifica si el Typer que va a agregar el contacto, ya se encuentra previamente registrado.

Métodos PUT

*PUT /actualizarInfoTyper*

CP\_ actualizarInfoTyper\_01

ID	Descripción	Precondiciones
CP_ actualizarInfoTyper_01	Se logra actualizar el nombre de usuario de un typer	El id del typer existe en los registros
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/actualizarInfoTyper">http://localhost:4000/typers/actualizarInfoTyper</a> { "IdentificadorTyper": "37ac87e0-2e63-40c0-9635-339264ec0802", "InformacionActualizada": "Nombre actualizado", "ModificadorDeMetodo": "usuario" }	Status = true	Status = true
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

CP\_ actualizarInfoTyper\_02

ID	Descripción	Precondiciones
CP_ actualizarInfoTyper_02	Se logra actualizar el estado de usuario de un typer	El id del typer existe en los registros
Entrada	Resultado esperado	Resultado obtenido
Consulta: <a href="http://localhost:4000/typers/actualizarInfoTyper">http://localhost:4000/typers/actualizarInfoTyper</a> { "IdentificadorTyper": "37ac87e0-2e63-40c0-9635-339264ec0802", "InformacionActualizada": "Nuevo estado de usuario", "ModificadorDeMetodo": "estado" }	Status = true	Status = true

<b>Estado</b>	<b>Fecha</b>	<b>Responsable</b>
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

#### CP\_ actualizarInfoTyper\_03

<b>ID</b>	<b>Descripción</b>	<b>Precondiciones</b>
CP_ actualizarInfoTyper_03	No se logra actualizar la información del usuario debido a que no existe en el sistema	El id del typer no existe en los registros
<b>Entrada</b>	<b>Resultado esperado</b>	<b>Resultado obtenido</b>
Consulta: <a href="http://localhost:4000/typers/actualizarInfoTyper">http://localhost:4000/typers/actualizarInfoTyper</a> { "IdentificadorTyper": "1234", "InformacionActualizada": "Nuevo estado de usuario", "ModificadorDeMetodo": "estado" }	Status = false	Status = false
<b>Estado</b>	<b>Fecha</b>	<b>Responsable</b>
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

*PUT /actualizarGrupo*

CP\_actualizarGrupo\_01

ID	Descripción	Precondiciones
CP_actualizarGrupo_01	Actualizar la descripción y el nombre del grupo con información correcta	Debe existir el grupo
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/actualizarGrupo/5 Body { "idGrupo": 5, "nombre": "Grupo de pruebas manuales actualizados", "descripcion": "Una nueva descripcion" }	{ "status": true, "message": "Grupo actualizado", "result": objeto del grupo actualizado }	{ "status": true, "message": "Grupo actualizado", "result": objeto del grupo actualizado }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

CP\_actualizarGrupo\_02

ID	Descripción	Precondiciones
CP_actualizarGrupo_02	Actualizar la descripción del grupo por una descripción vacía	Debe existir el grupo
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/actualizarGrupo/5 Body { "idGrupo": 5, "nombre": "Grupo de pruebas manuales actualizados", "descripcion": "" }	{ "status": false, "message": "La información está incorrecta", "result": null }	{ "status": true, "message": "Grupo actualizado", "result": objeto del grupo actualizado }
Estado	Fecha	Responsable
No pasó	15/06/21	Sammy Guadarrama Chávez

### CP\_actualizarGrupo\_03

ID	Descripción	Precondiciones
CP_actualizarGrupo_03	Actualizar la información de un grupo con el Id no existente	Puede no existir el grupo
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/actualizarGrupo/50 Body { "idGrupo": 50, "nombre": "Grupo de pruebas manuales actualizados", "descripcion": "Nueva descripcion" }	{ "status": false, "message": "Grupo no encontrado", "result": null }	{ "status": false, "message": "Grupo no encontrado", "result": null }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

### PUT /salirDeGrupo

#### CP\_salirDeGrupo\_01

ID	Descripción	Precondiciones
CP_salirDeGrupo_01	Salir de grupo con idTyper existente y idGrupo existente	Debe existir al menos un Typer registrado Debe existir el grupo El Typer debe ser integrante del grupo.
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/salirDeGrupo Parámetros: idGrupo=5 idTyper=" 22ae38b7-9f0a-4256-887a-a3f5d9d00275"	{ "status": true, "message": "Typer eliminado del grupo", "result": objeto con la información del typer eliminado del grupo }	{ "status": true, "message": "Typer eliminado del grupo", "result": objeto con la información del typer eliminado del grupo }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

## CP\_salirDeGrupo\_02

ID	Descripción	Precondiciones
CP_salirDeGrupo_02	Salir de grupo con idTyper No existente y idGrupo existente	No debe existir el Typer a eliminar
Entrada	Resultado esperado	Resultado obtenido
Consulta: /mensajes/salirDeGrupo Parámetros: idGrupo=5 idTyper=" 22ae38b7-9f0a-4256-887a-a3f5d9d00275b"	{ "status": false, "message": "El id del grupo y/o id del Typer no está registrado", "result": null }	{ "status": false, "message": "El id del grupo y/o id del Typer no está registrado", "result": null }
Estado	Fecha	Responsable
Pasó	15/06/21	Sammy Guadarrama Chávez

### Problemas detectados

No se realiza una validación para comprobar que la información proporcionada este correcta para poder ser actualizada.



PUT /actualizarContrasenia

CP\_ actualizarContrasenia\_01

ID	Descripción	Precondiciones
CP_ actualizarContrasenia_01	Se logra cambiar la contraseña de la cuenta del usuario	El id del typer existe en los registros
Entrada	Resultado esperado	Resultado obtenido
Consulta: http://localhost:4000/typers/actualizarContrasenia { "IdentificadorTyper": "37ac87e0-2e63-40c0-9635-339264ec0802", "InformacionActualizada": "admin2" }	Status = true	Status = true
Estado	Fecha	Responsable
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

CP\_ actualizarContrasenia\_02

ID	Descripción	Precondiciones
CP_ actualizarContrasenia_02	No se logra cambiar la contraseña de la cuenta del usuario debido a que no existe el id registrado	El id del typer no existe en los registros
Entrada	Resultado esperado	Resultado obtenido
Consulta: http://localhost:4000/typers/actualizarContrasenia { "IdentificadorTyper": "1234",	Status = false	Status = false

"InformacionActualizada": "admin2" }		
<b>Estado</b>	<b>Fecha</b>	<b>Responsable</b>
Prueba exitosa	15/06/21	Ángel de Jesús Juárez García

## Resultados

Resultados métodos GET

CP\_obtenerMensajes\_01

apigateway / <http://localhost:4000/mensajes/obtenerMensajes/1> Save

GET <http://localhost:4000/mensajes/obtenerMensajes/1>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 11.67 s Size: 778 KB

Pretty Raw Preview Visualize JSON

```
1  {
2    "status": true,
3    "message": "Mensajes encontrados",
4    "result": [
5      {
6        "idMensaje": 1,
7        "contenido": "Hola como están?",
8        "fecha": "2021-05-19T05:04:32",
9        "hora": "05:04:00",
10       "idGrupo": 1,
11       "typer": {
12         "IdTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275",
13         "Username": "SammyGuayaba",
14         "Estado": "gaseoso y sólido xdd",
15         "FotoDePerfil": "RUTA DE IMAGEN",
16         "Estatus": 1,
17         "Contraseña": [],
18         "Correos": []
19       }
20     ]
21   }
```

CP\_obtenerMensajes\_02

apigateway / <http://localhost:4000/mensajes/obtenerMensajes/1> Save

GET <http://localhost:4000/mensajes/obtenerMensajes/2>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 17 ms Size: 346 B

Pretty Raw Preview Visualize JSON

```
1  {
2    "status": false,
3    "data": {
4      "message": "No hay mensajes en el grupo",
5      "result": null
6    }
7  }
```

### CP\_obtenerMensajes\_03

GET ▼ http://localhost:4000/mensajes/obtenerMensajes/2

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK Time: 30

Pretty Raw Preview Visualize JSON ▼ 🔧

```
1 {
2   "status": false,
3   "data": {
4     "message": "No hay mensajes en el grupo",
5     "result": null
6   }
7 }
```

### CP\_obtenerMensajes\_04

GET ▼ http://localhost:4000/mensajes/obtenerMensajes/

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results 🌐 Status: 404 Not Found Time: 30

Pretty Raw Preview Visualize JSON ▼ 🔧

```
1 {
2   "success": false,
3   "msg": "This route does not exist"
4 }
```

## CP\_misGrupos\_01

GET <http://localhost:4000/mensajes/misGrupos/8121d387-f063-454f-a764-d0b29385741f>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": true,
3   "message": "Grupos encontrados",
4   "result": [
5     {
6       "IdGrupo": 1,
7       "Nombre": "Chat chido",
8       "Descripcion": "Creado desde web java",
9       "FechaCreacion": "2021-05-19T00:00:00",
10      "Mensajes": [],
11      "Pertenece": []
12    },
13    {
14      "IdGrupo": 4,
```

## CP\_misGrupos\_02

GET <http://localhost:4000/mensajes/misGrupos/8121d387-f063-454f-a764-d0b29385741f2>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {}
2   "status": false,
3   "message": "Grupo(s) no encontrado(s) o inexistente(s)",
4   "result": null
5 }
```

## CP\_misGrupos\_03

GET ▼ http://localhost:4000/mensajes/misGrupos/b202eeb5-5f4e-4afa-b241-e3fb0d767997

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
	Key	Value	Description

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK

Pretty Raw Preview Visualize JSON ▼ 🔧

```
1 [
2   "status": false,
3   "message": "Grupo(s) no encontrado(s) o inexistente(s)",
4   "result": null
5 ]
```

## CP\_obtenerContactos\_01

GET ▼ http://localhost:4000/typers/obtenerContactos/8121d387-f063-454f-a764-d0b29385741f

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
	Key	Value	Description

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK Time: 7.89 s

Pretty Raw Preview Visualize JSON ▼ 🔧

```
1 [
2   "status": true,
3   "message": "Contactos encontrados",
4   "result": [
5     {
6       "bloqueado": false,
7       "esFavorito": false,
8       "contacto": {
9         "IdTypers": "9148c367-2388-465b-9e04-fc9f56bbab6a",
10        "Username": "frews",
11        "Estado": "Viva Belice",
12        "FotoDePerfil": "RUTA DE IMAGEN",
13        "Estatus": 1,
14        "Contraseña": [],
15        "Correos": []
16      }
17    },
18    ...
19  ]
20 ]
```

## CP\_obtenerContactos\_02

GET ▼ http://localhost:4000/typers/obtenerContactos/ff2e5ee8-0712-48ec-a1dd-6825008977ec

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK Time

Pretty Raw Preview Visualize JSON ▼ 🔍

```
1 {
2   "status": false,
3   "message": "Contactos no encontrados",
4   "result": []
5 }
```

## CP\_obtenerContactos\_03

GET ▼ http://localhost:4000/typers/obtenerContactos/ff2e5ee8-0712-48ec-a1dd-6825008977ec0

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK Time

Pretty Raw Preview Visualize JSON ▼ 🔍

```
1 {
2   "status": false,
3   "message": "Contactos no encontrados",
4   "result": []
5 }
```

## CP\_ObtenerMultimedia\_01

The screenshot displays a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:4000/mensajes/obtenerMultimedia?idMMultimedia=4ab911f2-6f57-4c02-8d61-b37699cf0ddf`
- Params:** A table with one entry: 

KEY	VALUE	DESCRIPTION
idMMultimedia	4ab911f2-6f57-4c02-8d61-b37699cf0ddf	
- Body:** The response is shown in JSON format: 

```
{  "urlMultimedia": "http://localhost:3325/multimedia/obtenerMultimedia?idMultimedia=undefined"}
```
- Status:** 200 OK
- Time:** 12 ms



## CP\_ObtenerMultimedia\_02

GET <http://localhost:4000/mensajes/obtenerMultimedia?idMMultimedia=4ab911f2>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	idMMultimedia	4ab911f2	
	Key	Value	Description

Body Cookies Headers (4) Test Results Status: 400 Bad Request Time: 755 ms

Pretty Raw Preview Visualize JSON

```
1 [
2   "status": false,
3   "data": {
4     "message": "No se encontro el archivo buscado",
5     "result": null
6   }
7 ]
```

## Resultados métodos POST

### CP\_registrarMultimedia \_01

POST <http://localhost:4000/mensajes/registrarMultimedia?idTyper=ac8794cd-6ffe-46dc-b2b6-836e03fc0de5>

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	file	<input type="text" value="compu.jpg"/>	
	Key	Value	Description

Body Cookies Headers (8) Test Results Status: 200 OK Time: 2.46 s

Pretty Raw Preview Visualize JSON

```
1 [
2   "status": true,
3   "message": "El archivo multimedia se registro correctamente",
4   "result": {
5     "IdMultimedia": "f7f3282d-3a9e-4bf8-a081-2640c2969636",
6     "Ruta": "/app/imagenes/ac8794cd-6ffe-46dc-b2b6-836e03fc0de5compu.jpg",
7     "IdTipoMultimedia": 1,
8     "IdTipoMultimediaNavigation": null
9   }
10 ]
```

## CP\_crearGrupo\_01

The screenshot displays a REST client interface with a POST request to `http://localhost:4000/mensajes/crearGrupo`. The request body is a JSON object with the following structure:

```
1 {
2   "nombre": "Grupo prueba unitaria",
3   "descripcion": "Otro grupo de prueba",
4   "pertenece": []
5 }
6 "idTvoer": "8121d387-f063-454f-a764-d0b29385741f"
```

The response status is 200 OK. The response body, shown in 'Pretty' JSON format, is as follows:

```
1 {
2   "status": true,
3   "message": "Nuevo grupo creado",
4   "result": {
5     "IdGrupo": 5,
6     "Nombre": "Grupo prueba unitaria",
7     "Descripcion": "Otro grupo de prueba",
8     "FechaCreacion": "2021-06-16T01:22:53.5313188+00:00",
9     "Mensajes": [],
10    "Pertenece": [
11      {
12        "IdGrupo": 5,
13        "IdTyper": "8121d387-f063-454f-a764-d0b29385741f"
14      },
15      {
16        "IdGrupo": 5,
```

## CP\_crearGrupo\_02

The screenshot shows a REST client interface with a POST request to `http://localhost:4000/mensajes/crearGrupo`. The request body is in JSON format and contains the following data:

```
{
  "nombre": "",
  "descripcion": ""
}
```

The response body is also in JSON format and shows an error status:

```
{
  "status": false,
  "message": "El grupo no tiene los datos necesarios",
  "result": null
}
```

The status bar at the bottom right indicates a status of 200.

## CP\_crearGrupo\_03

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:4000/mensajes/crearGrupo
- Body Type:** raw (selected)
- Response Body (JSON):**

```
1 {  
2   "status": false,  
3   "message": "El grupo no tiene los datos necesarios",  
4   "result": null  
5 }
```

## CP\_agregarIntegrantes\_01

POST ▼ http://localhost:4000/mensajes/agregarIntegrantes/5

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 [
2   {
3     "idTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f02"
4   }
5 ]
```

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK

Pretty Raw Preview Visualize **JSON** ▼ 🔍

```
1 {
2   "status": true,
3   "message": "Integrante(s) agregado(s)",
4   "result": [
5     {
6       "IdGrupo": 5,
7       "IdTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f02",
8       "IdGrupoNavigation": null
9     }
10  ]
11 }
```

## CP\_agregarIntegrantes\_02

POST ▼ http://localhost:4000/mensajes/agregarIntegrantes/50

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 [
2   {
3     "idTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f02"
4   }
5 ]
```

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK Time: 91ms

Pretty Raw Preview Visualize **JSON** ▼ 🔍

```
1 {
2   "status": false,
3   "message": "No existe el grupo con el id especificado.",
4   "result": null
5 }
```

### CP\_agregarIntegrantes\_03

The screenshot shows a REST client interface with a POST request to `http://localhost:4000/mensajes/agregarIntegrantes/5`. The request body is a JSON array with one object containing `"idType": "f21d4482-e56b-46c6-aaff-1e3ac7e68f020"`. The response status is `200 OK`. The response body, shown in Pretty JSON, is:

```
{
  "status": true,
  "message": "Integrante(s) agregado(s)",
  "result": [
    {
      "IdGrupo": 5,
      "IdType": "f21d4482-e56b-46c6-aaff-1e3ac7e68f020",
      "IdGrupoNavigation": null
    }
  ]
}
```

### CP\_agregarIntegrantes\_04

The screenshot shows a REST client interface with a POST request to `http://localhost:4000/mensajes/agregarIntegrantes/5`. The request body is an empty JSON array `[]`. The response status is `200 OK`. The response body, shown in Pretty JSON, is:

```
{
  "status": false,
  "message": "No se pudo agregar el nuevo integrante",
  "result": null
}
```

## CP\_enviarMensaje\_01

The screenshot displays a REST client interface with a POST request to `http://localhost:4000/mensajes/enviarMensaje`. The request body is a JSON object with the following fields:

```
{
  "contenido": "Mensaje pruebas manuales",
  "idGrupo": 5,
  "idTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275",
  "idMultimedia": ""
}
```

The response status is **200 OK** with a time of **1326** ms. The response body is a JSON object with the following fields:

```
{
  "status": true,
  "message": "Mensaje enviado",
  "result": {
    "idMensaje": 60,
    "contenido": "Mensaje pruebas manuales",
    "fecha": "2021-06-16T05:07:04.5673861+00:00",
    "hora": "05:07:00",
    "idGrupo": 5,
    "typer": {
      "IdTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275",
      "Username": "SammyGuayaba",
      "Estado": "gaseoso y sólido xdd",
      "FotoDePerfil": "RUTA DE IMAGEN",
      "Estatus": 1,
      "Contrasenia": [],
    }
  }
}
```

## CP\_enviarMensaje\_02

The screenshot displays a REST client interface for a POST request to `http://localhost:4000/mensajes/enviarMensaje`. The request body is a JSON object with the following fields:

```
{  "contenido": "Mensaje pruebas manuales",  "idGrupo": 5,  "idTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275",  "idMultimedia": "346d6a15-6347-4ac7-91bf-d9d1da2c28ff"}
```

The response status is **200 OK** with a response time of **304 ms**. The response body is a JSON object with the following fields:

```
{  "contenido": "Mensaje pruebas manuales",  "fecha": "2021-06-16T05:12:03.0165721+00:00",  "hora": "05:12:00",  "idGrupo": 5,  "typer": {    "IdTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275",    "Username": "SammyGuayaba",    "Estado": "gaseoso y sólido xdd",    "FotoDePerfil": "RUTA DE IMAGEN",    "Estatus": 1,    "Contrasenia": [],    "Correos": []  },  "idMultimedia": "http://localhost:3325/multimedia/obtenerMultimedia?idMultimedia=346d6a15-6347-4ac7-91bf-d9d1da2c28ff"}
```



## CP\_registrarTyper\_01

POST http://localhost:4000/typers/registrarTyper

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

Body Cookies Headers (8) Test Results Status: 200 OK Time: 713 ms

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": true,
3   "message": "El usuario se registro correctamente",
4   "result": {
5     "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802",
6     "Username": "Nuevo miembro",
7     "Estado": "Estado generico",
8     "FotoDePerfil": "",
9     "Estatus": 1,
10    "Contrasenia": [
11      {
12        "IdContrasenia": 4,
13        "Contrasenia1": "8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918",
14        "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802"
15      }
16    ],
17    "Correos": [
18      {
19        "IdCorreo": 7,
20        "Direccion": "miCorreo@hotmail.com",
21        "EsPrincipal": 1,
22        "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802"
23      },
24      {
25        "IdCorreo": 8,
```

## CP\_registrarTyper\_02

POST http://localhost:4000/typers/registrarTyper

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

Body Cookies Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": false,
3   "message": "El nombre de usuario ya existe",
4   "result": null
5 }
```

## CP\_loginTyper\_01

```
POST http://localhost:4000/typers/loginTyper

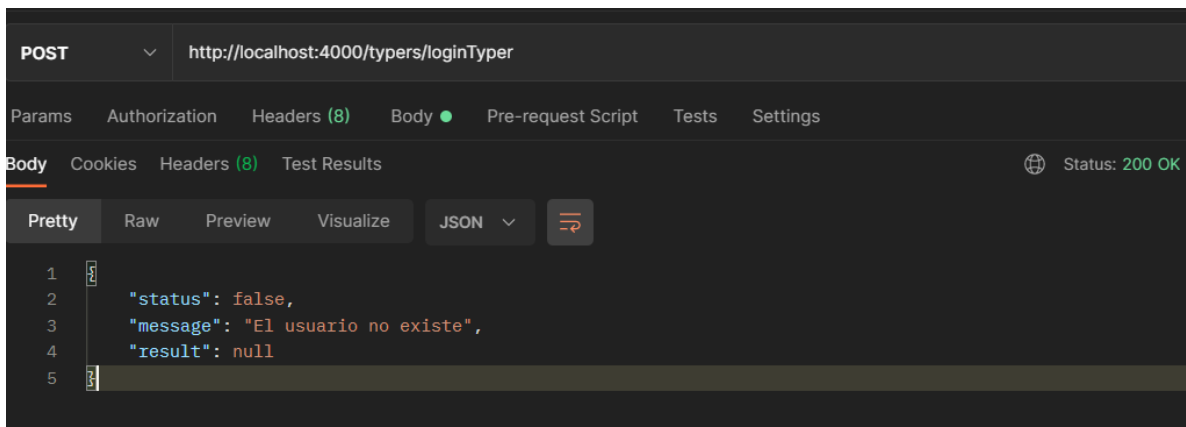
Params Authorization Headers (8) Body ● Pre-request Script Tests Settings
Body Cookies Headers (8) Test Results Status: 200 OK
Pretty Raw Preview Visualize JSON ↗
1 {
2   "status": true,
3   "message": "Inicio de sesión exitoso",
4   "result": [
5     {
6       "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802",
7       "Username": "Nuevo miembro",
8       "Estado": "Estado generico",
9       "FotoDePerfil": "",
10      "Estatus": 1,
11      "Contrasenia": [
12        {
13          "IdContrasenia": 4,
14          "Contrasenia1": "8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918",
15          "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802"
16        }
17      ],
18      "Correos": [
19        {
20          "IdCorreo": 7,
21          "Direccion": "miCorreo@hotmail.com",
22          "EsPrincipal": 1,
23          "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802"
24        }
25      ]
26    }
27  ]
28 }
```

CP\_loginTyper\_02

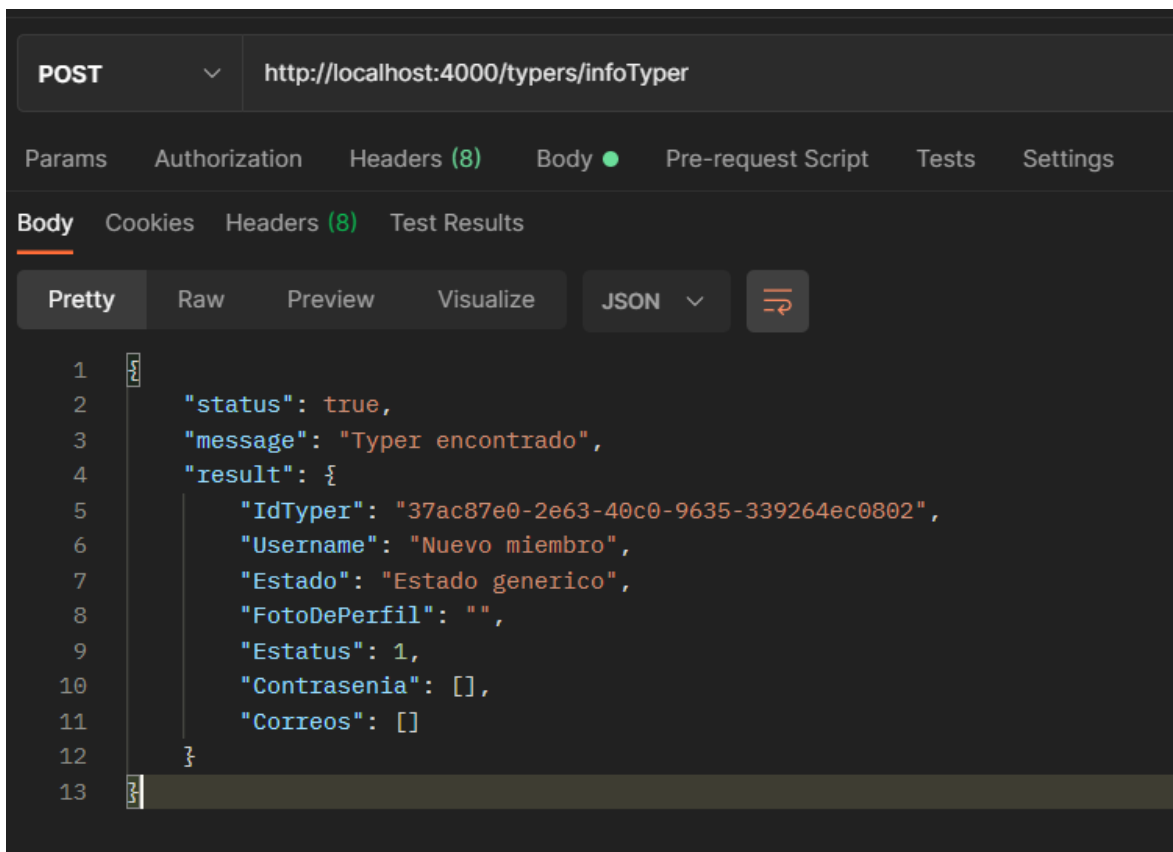
```
POST http://localhost:4000/typers/loginTyper

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings
Body Cookies Headers (8) Test Results Status: 200 OK
Pretty Raw Preview Visualize JSON ↗
1 {
2   "status": false,
3   "message": "Contraseña incorrecta",
4   "result": null
5 }
```

### CP\_loginTyper \_03



### CP\_infoTyper \_01



## CP\_infoTyper \_02

The screenshot shows a REST client interface with a POST request to `http://localhost:4000/typers/infoTyper`. The response body is displayed in JSON format, showing a successful status and user information.

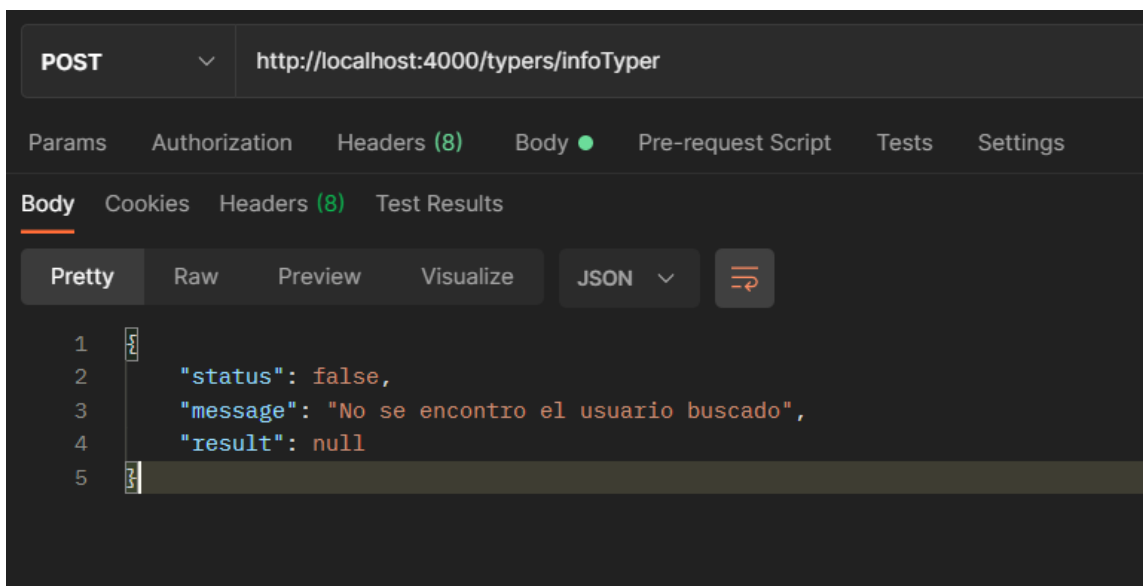
```
1 {
2   "status": true,
3   "message": "Typer encontrado",
4   "result": {
5     "IdTyper": "ac8794cd-6ffe-46dc-b2b6-836e03fc0de5",
6     "Username": "admin",
7     "Estado": null,
8     "FotoDePerfil": "http://localhost:3325/multimedia/obtenerMultimedia",
9     "Estatus": 1,
10    "Contrasenia": [],
11    "Correos": []
12  }
13 }
```

## CP\_infoTyper \_03

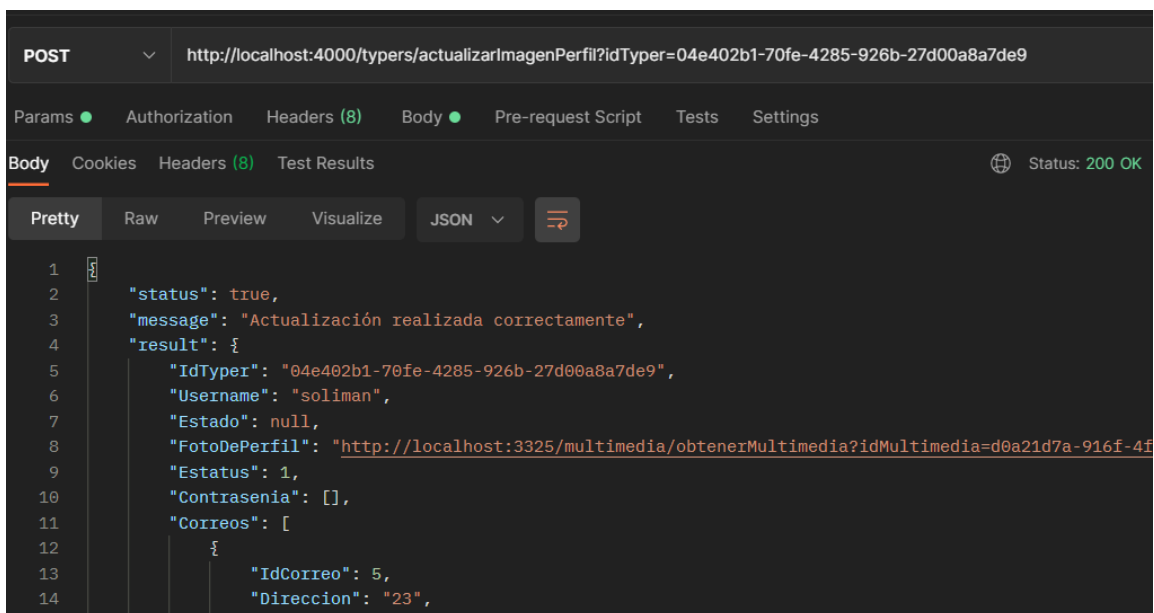
The screenshot shows a REST client interface with a POST request to `http://localhost:4000/typers/infoTyper`. The response body is displayed in JSON format, showing a failed status and a message indicating the user was not found.

```
1 {
2   "status": false,
3   "message": "No se encontro el usuario buscado",
4   "result": null
5 }
```

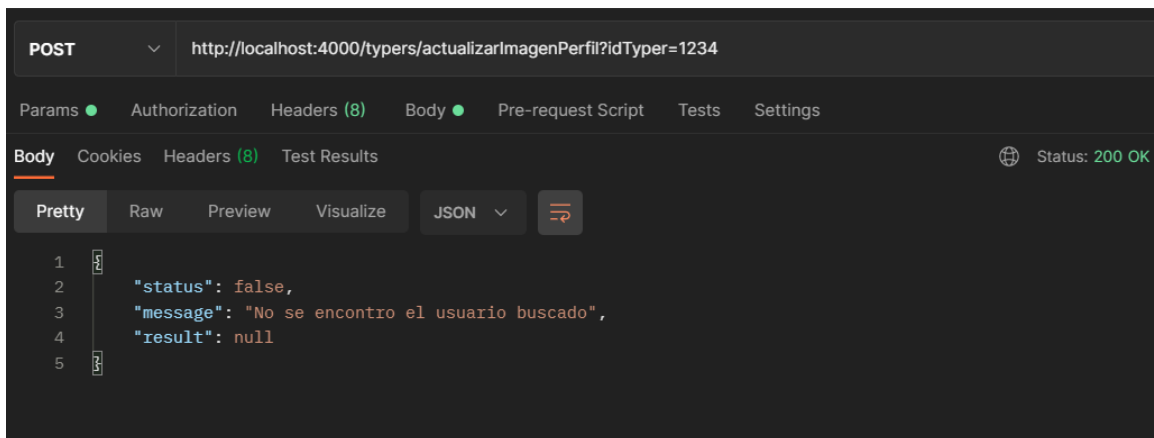
## CP\_infoTyper \_04



## CP\_actualizarImagenDePerfil\_01



## CP\_ actualizarImagenDePerfil\_02



## CP\_agregarContacto\_01

**POST** `http://localhost:4000/typers/agregarContacto`

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   ... "idTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f02",
3   "contacto": "user_5"
4 }
```

**Body** Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 {
2   "status": true,
3   "message": "Contacto agregado",
4   "result": {
5     "bloqueado": false,
6     "esFavorito": false,
7     "contacto": {
8       "IdTyper": "ff2e5ee8-0712-48ec-a1dd-6825008977ec",
9       "Username": "user_5",
10      "Estado": "Holaa",
11      "FotoDePerfil": ""
12    }
13   }
14 }
```

## CP\_agregarContacto\_02

**POST** `http://localhost:4000/typers/agregarContacto`

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   ... "idTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f02",
3   "contacto": "user_5"
4 }
```

**Body** Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 {
2   "status": false,
3   "message": "El contacto ya se encuentra agigeado",
4   "result": {}
5 }
```

## CP\_agregarContacto\_03

The screenshot displays a REST client interface for a POST request to `http://localhost:4000/typers/agregarContacto`. The request body is a JSON object with the following structure:

```
{
  "idTyper": "f21d4482-e56b-46c6-aaff-1e3ac7e68f02",
  "contacto": "user_50"
}
```

The response body, shown in the 'Body' tab, is a JSON object indicating a failed operation:

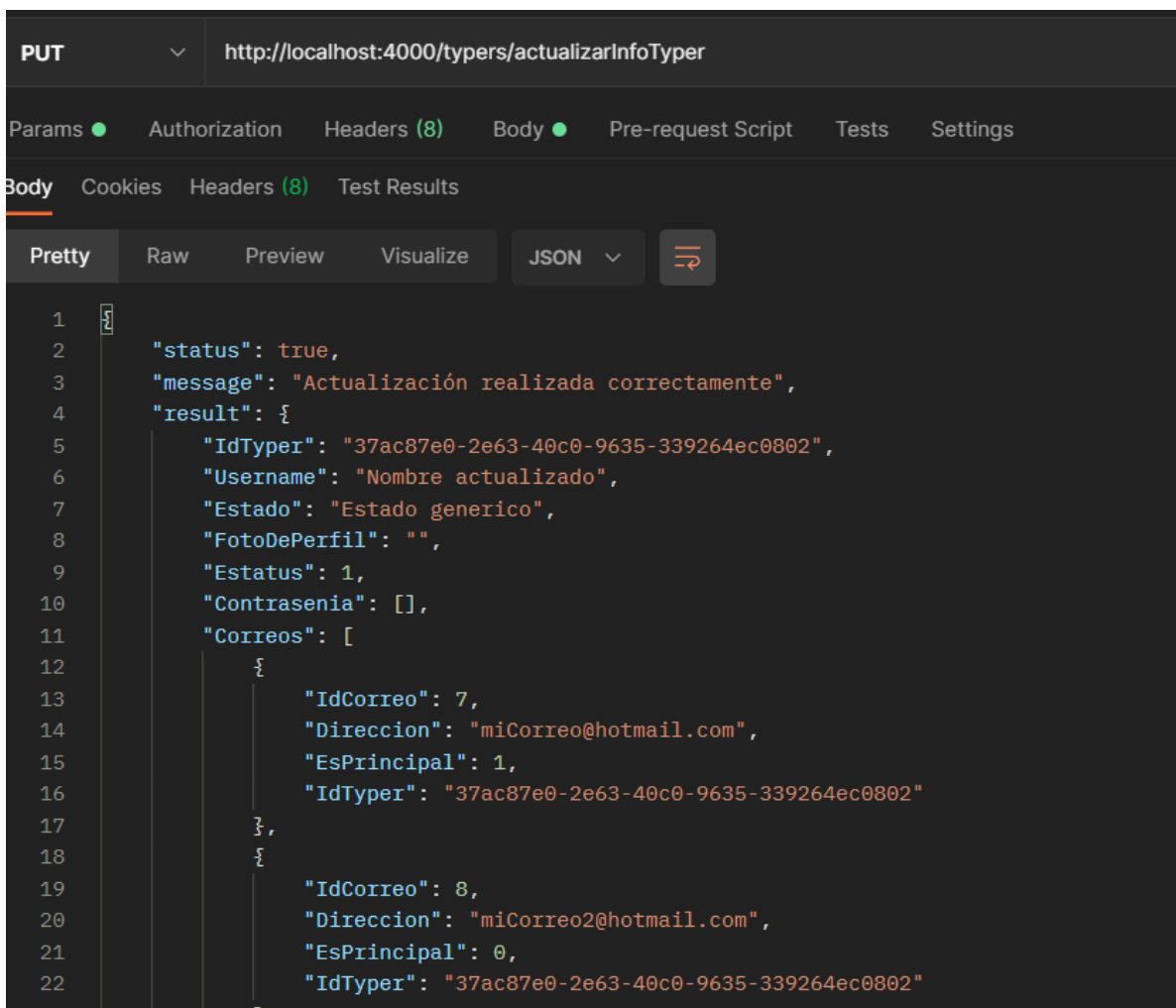
```
{
  "status": false,
  "message": "No se encontro el usuario buscado",
  "result": null
}
```

The interface includes tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The Body tab is active, showing the response in a 'Pretty' JSON format. The response status is 200, and the content type is application/json.



## Resultados métodos PUT

### CP\_ actualizarInfoTyper\_01



```
PUT http://localhost:4000/typers/actualizarInfoTyper

Body
  Cookies
  Headers (8)
  Test Results

Pretty
Raw
Preview
Visualize
JSON

1  {
2    "status": true,
3    "message": "Actualización realizada correctamente",
4    "result": {
5      "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802",
6      "Username": "Nombre actualizado",
7      "Estado": "Estado generico",
8      "FotoDePerfil": "",
9      "Estatus": 1,
10     "Contrasenia": [],
11     "Correos": [
12       {
13         "IdCorreo": 7,
14         "Direccion": "miCorreo@hotmail.com",
15         "EsPrincipal": 1,
16         "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802"
17       },
18       {
19         "IdCorreo": 8,
20         "Direccion": "miCorreo2@hotmail.com",
21         "EsPrincipal": 0,
22         "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802"
23       }
24     ]
25   }
26 }
```

## CP\_ actualizarInfoTyper\_02

PUT ▼ http://localhost:4000/typers/actualizarInfoTyper

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK

Pretty Raw Preview Visualize JSON ▼ ↔

```
1  {
2    "status": true,
3    "message": "Actualización realizada correctamente",
4    "result": {
5      "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802",
6      "Username": "Nombre actualizado",
7      "Estado": "Nuevo estado de usuario",
8      "FotoDePerfil": "",
9      "Estatus": 1,
10     "Contrasenia": [],
11     "Correos": [
12       {
13         "IdCorreo": 7,
14         "Direccion": "miCorreo@hotmail.com",
15         "EsPrincipal": 1,
16         "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802"
17       }
18     ]
19   }
20 }
```

## CP\_ actualizarInfoTyper\_03

PUT ▼ http://localhost:4000/typers/actualizarInfoTyper

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings

Body Cookies Headers (8) Test Results 🌐 Status: 200 OK

Pretty Raw Preview Visualize JSON ▼ ↔

```
1  {
2    "status": false,
3    "message": "No se encontro el usuario buscado",
4    "result": null
5  }
```

## CP\_actualizarGrupo\_01

The screenshot displays a REST client interface with a PUT request to `http://localhost:4000/mensajes/actualizarGrupo/5`. The 'Body' tab is selected, showing a JSON payload. Below the request, the 'Body' section shows the response in 'Pretty' format, which is a JSON object containing status, message, and result details.

**Request:**

```
1 {
2   "idGrupo": 5,
3   "nombre": "Grupo de pruebas manuales actualizados",
4   "descripcion": "Una nueva descripcion"
5 }
```

**Response:**

```
1 {
2   "status": true,
3   "message": "Grupo actualizado",
4   "result": {
5     "IdGrupo": 5,
6     "Nombre": "Grupo de pruebas manuales actualizados",
7     "Descripcion": "Una nueva descripcion",
8     "FechaCreacion": "2021-06-16T00:00:00",
9     "Mensajes": [],
10    "Participantes": []
11  }
12 }
```

## CP\_actualizarGrupo\_02

The screenshot displays a REST client interface with a PUT request to `http://localhost:4000/mensajes/actualizarGrupo/5`. The request body is a JSON object with the following structure:

```
1 {
2   "idGrupo": 5,
3   "nombre": "Grupo de pruebas manuales actualizados",
4   "descripcion": ""
5 }
```

The response body is also in JSON format, showing the updated group details and a success message:

```
1 {
2   "status": true,
3   "message": "Grupo actualizado",
4   "result": {
5     "IdGrupo": 5,
6     "Nombre": "Grupo de pruebas manuales actualizados",
7     "Descripcion": "",
8     "FechaCreacion": "2021-06-16T00:00:00",
9     "Mensajes": [],
10    "Pertenece": []
  }
```

## CP\_actualizarGrupo\_03

The screenshot shows a REST client interface with a PUT request to `http://localhost:4000/mensajes/actualizarGrupo/50`. The 'Body' tab is selected, and the request body is a JSON object:

```
{
  "idGrupo": 50,
  "nombre": "Grupo de pruebas manuales actualizados",
  "descripcion": "Nueva descripcion"
}
```

Below the request, the 'Body' tab shows the response in 'Pretty' JSON format:

```
{
  "status": false,
  "message": "Grupo no encontrado",
  "result": null
}
```

## CP\_salirDeGrupo\_01

The screenshot shows a REST client interface with a PUT request to `http://localhost:4000/mensajes/salirDeGrupo?idGrupo=5&idTyper=22ae38b7-9f0a-4256-887a-a3f5d9d00275`. The 'Params' tab is selected, showing query parameters:

KEY	VALUE	DES
<input checked="" type="checkbox"/> idGrupo	5	
<input checked="" type="checkbox"/> idTyper	22ae38b7-9f0a-4256-887a-a3f5d9d00275	
Key	Value	Des

Below the params, the 'Body' tab shows the response in 'Pretty' JSON format:

```
{
  "status": true,
  "message": "Typer eliminado del grupo",
  "result": {
    "IdGrupo": 5,
    "IdTyper": "22ae38b7-9f0a-4256-887a-a3f5d9d00275",
    "IdGrupoNavigation": null
  }
}
```

## CP\_salirDeGrupo\_02

**PUT** ⌵ <http://localhost:4000/mensajes/salirDeGrupo?idGrupo=5&idTyper=22ae38b7-9f0a-4256-887a-a3f5d9d00275b>

**Params** ● Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	idGrupo	5	
<input checked="" type="checkbox"/>	idTyper	22ae38b7-9f0a-4256-887a-a3f5d9d00275b	
	Key	Value	Description

**Body** Cookies Headers (8) Test Results 🌐 Status: 200 OK

**Pretty** Raw Preview Visualize JSON ⌵ ⌵

```
1  {
2    "status": false,
3    "message": "El id del grupo y/o id del Typer no está registrado",
4    "result": null
5  }
```

## CP\_ actualizarContrasenia\_01

The screenshot shows a REST client interface with a PUT request to `http://localhost:4000/typers/actualizarContrasenia`. The response is displayed in the 'Body' tab, showing a JSON object with a status of true, a success message, and user details.

```
PUT http://localhost:4000/typers/actualizarContrasenia

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

1 {
2   "status": true,
3   "message": "Contraseña actualizada correctamente",
4   "result": {
5     "IdContrasenia": 4,
6     "Contrasenia1": "1c142b2d01aa34e9a36bde480645a57fd69e14155dacfab5a3f9257b77fdc8d8",
7     "IdTyper": "37ac87e0-2e63-40c0-9635-339264ec0802",
8     "IdTyperNavigation": null
9   }
10 }
```

## CP\_ actualizarContrasenia\_02

The screenshot shows a REST client interface with a PUT request to `http://localhost:4000/typers/actualizarContrasenia`. The response is displayed in the 'Body' tab, showing a JSON object with a status of false, an error message, and a null result.

```
PUT http://localhost:4000/typers/actualizarContrasenia

Params Authorization Headers (8) Body Pre-request Script Tests

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

1 {
2   "status": false,
3   "message": "El usuario no se encontro",
4   "result": null
5 }
```

## Estrategia de despliegue

Se recomienda que se analice la sección “Vista de despliegue” para conocer todas las partes de las que consta la aplicación y el cómo se comunican.

La aplicación TypeMe al ser enfocada a la comunicación por medio de un chat debe de estar disponible en múltiples plataformas para así ofrecer la flexibilidad suficiente para su uso cotidiano. En este caso la aplicación está separada en 3 tipos de cliente y un lado del servidor. En el lado del servidor se cuenta con 10 servicios de los 4 son bases de datos, que se muestran en la siguiente tabla:

Servicio	Nombre del contenedor	Puerto
mysql_tipers	db_tipers	3320
mongo_contactos	db_contactos	33021
mysql_mensajes	db_mensajes	3322
mysql_multimedia	db_multimedia	3323

Los otros 4 son microservicios que procesan la información individual de cada base de datos:

Servicio	Depende de	Puerto
ms_tipers	mysql_tipers	3324
ms_contactos	mongo_contactos	3327
ms_mensajes	mysql_mensajes	3326
ms_multimedia	mysql_multimedia	3325

Luego, se cuenta con la API donde se integran todas las operaciones de los microservicios, esta se encuentra en el puerto 4000. Y, por último, 1 servidor web que nos permite establecer comunicación entre los distintos clientes que estén ocupando la aplicación en tiempo real y el servidor para el cliente web.

Por el lado del cliente se cuenta con 3 tipos, una es el cliente de escritorio para el sistema operativo en Windows, otro es un cliente web que se ejecuta en línea el cual está disponible a través de internet y por último un cliente como aplicación para el sistema operativo Android.

La funcionalidad del sistema se lleva a cabo de la siguiente manera:

Al iniciar cualquiera de los clientes se muestra la pantalla de login, en este caso el usuario ingresa sus datos y el cliente envía dichos datos a través de una petición con el protocolo http a la API gateway ubicado en el servidor, esta a su vez realiza las operaciones necesarias para iniciar sesión utilizando los microservicios pertinentes, y devolver la información de inicio de sesión.

De esta manera cualquiera de los clientes puede realizar peticiones a la API gateway sin importar quien lo hace ya que la petición enviada utilizada en los 3



clientes es la misma y la parte encargada de procesar la información y devolver el resultado es la API.

En el caso específico del chat se debe de tener en cuenta que, en caso de recibir un mensaje, el servidor debe notificarlo al o los destinatarios en ese momento, por lo que se creó un servidor el cual se encuentra a la escucha de estos envíos de información y notifica a los usuarios conectados, esto se realiza de la siguiente manera:

Al iniciar sesión en alguno de los clientes se crea una conexión desde el cliente hasta el servidor por medio de SignalR y se une a un canal de comunicación en tiempo real, posteriormente al enviar un mensaje, este se envía al servidor y se notifica a todos los clientes conectados en el momento de dicho envío por lo que los integrantes de un chat reciben dicho mensaje en el momento y se les notifica.

## Conclusiones

Durante la realización de la aplicación TypeMe existieron retos los cuales no se tuvieron en cuenta al plantear la aplicación y su entorno operativo, el primero y más importante fue el de la creación de microservicios ya que al ser una aplicación pensada con un enfoque distribuido el principal problema es administrar cada uno de los servicios ofrecidos, esto se logró a través del análisis de las funciones que se tendrían dentro de la aplicación y el cómo aislaríamos cada una de estas en una pieza que funcionara autónomamente para así poder ejecutar cada una de sus funciones en un espacio diferente sin afectar el funcionamiento general del sistema, gracias a las herramientas como *Docker* este proceso de aislamiento y distribución se facilitó de gran manera.

De igual manera, la creación de una aplicación de mayor tamaño a las hechas anteriormente nos ha dado más experiencia en áreas diferentes a la programación como la importancia de la organización en un equipo de trabajo, así como la resolución de distintos tipos de problemas que pueden ocurrir en estos proyectos.

Al momento de creación de este documento consideramos que la aplicación realizada cumple con las funcionalidades necesarias para poder ser utilizada sin problema por una persona y que con lo que llega a ofrecer puede considerarse una aplicación de chat construida correctamente.

## Referencias

(2016, 13 junio). *Ogranada/js-coding-standards*. GitHub. <https://github.com/Ogranada/js-coding-standards>

*JavaScript / MDN*. (2021, 13 junio). MDN Web Docs.

<https://developer.mozilla.org/es/docs/Web/JavaScript>

Bc, F. (2012). *Estandar de Codificación C#*. Scribd.

<https://es.scribd.com/document/106455151/Estandar-de-COdificacion-C>