Build your own Microsoft Teams using Microsoft Graph Toolkit and .NET Core API

Sohil Bhalla

Microsoft 365 Expert

Codeless Technology B.V.

sbhalla@codeless.com

LinkedIn: https://www.linkedin.com/in/sohil-bhalla-46121434/

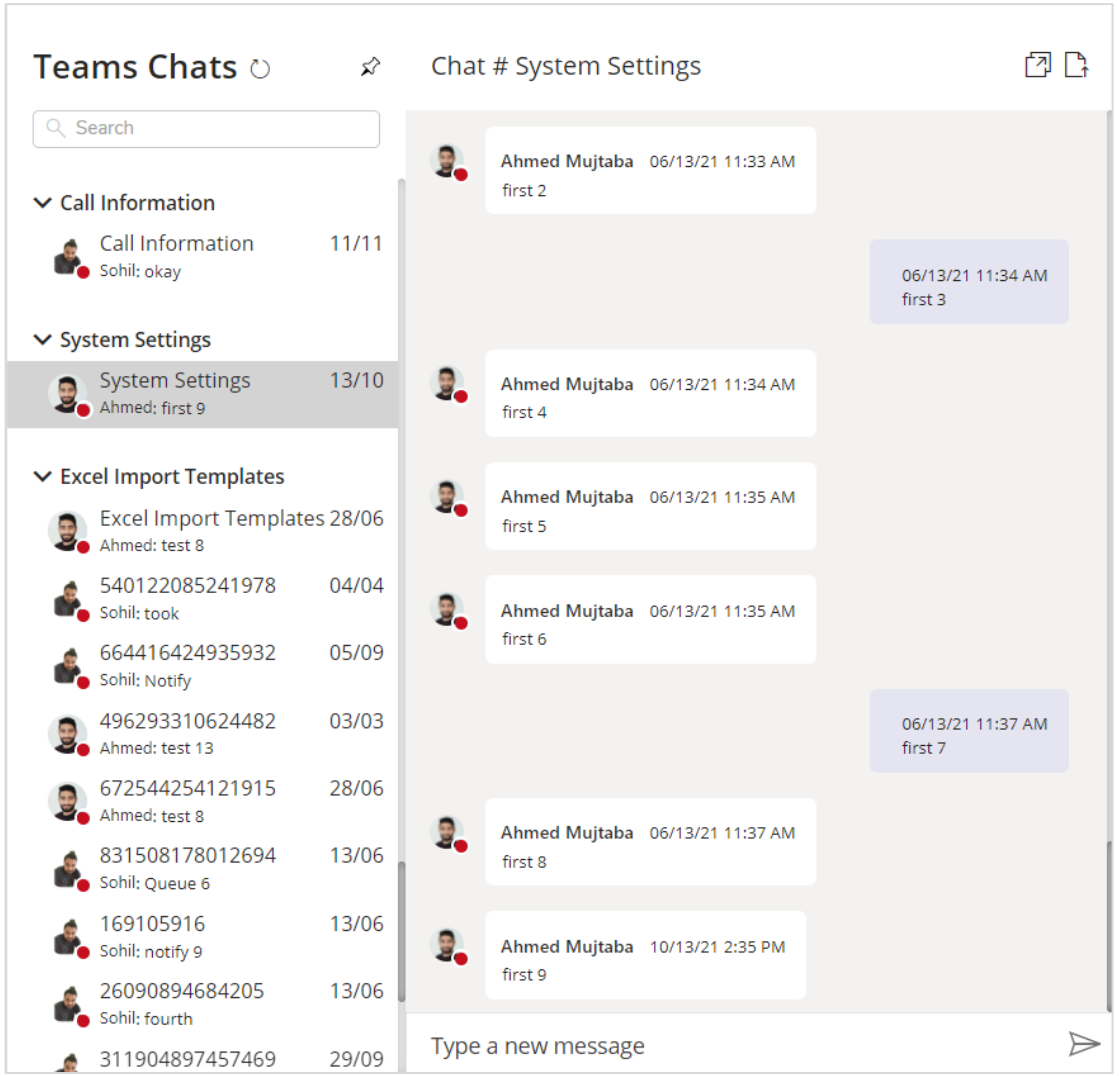
PnP Blog: https://pnp.github.io/blog/post/build-teams-using-graph-toolkit/

Business Case

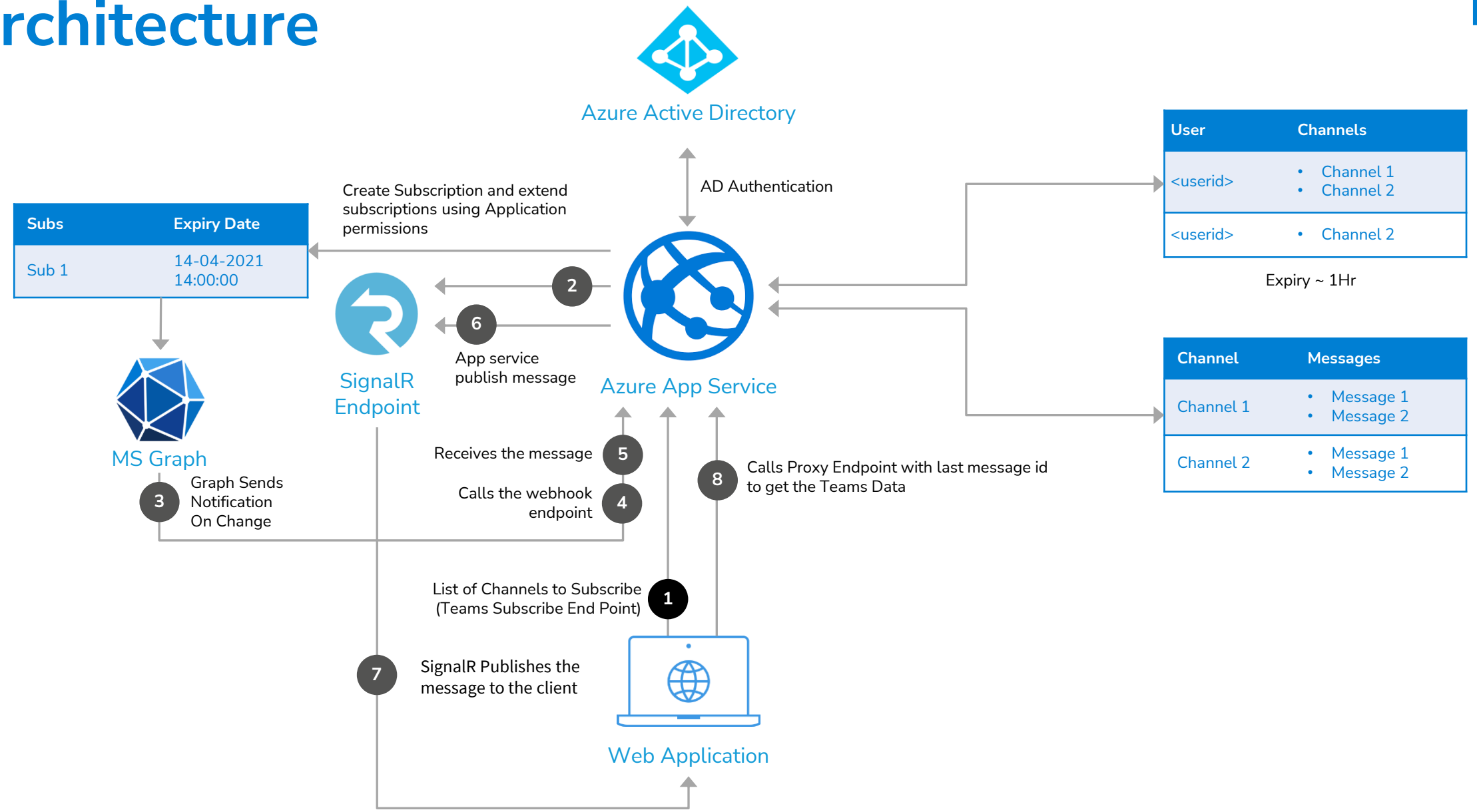
Build Microsoft Teams in React Typescript and integrated with a web application using web components.

The Teams functionality should include:

- ✓ **Sending messages**
- ✓ **Initiate chat**
- ✓ **Receiving messages**
- ✓ **Get notified**
- ✓ **Show emojis**



Architecture

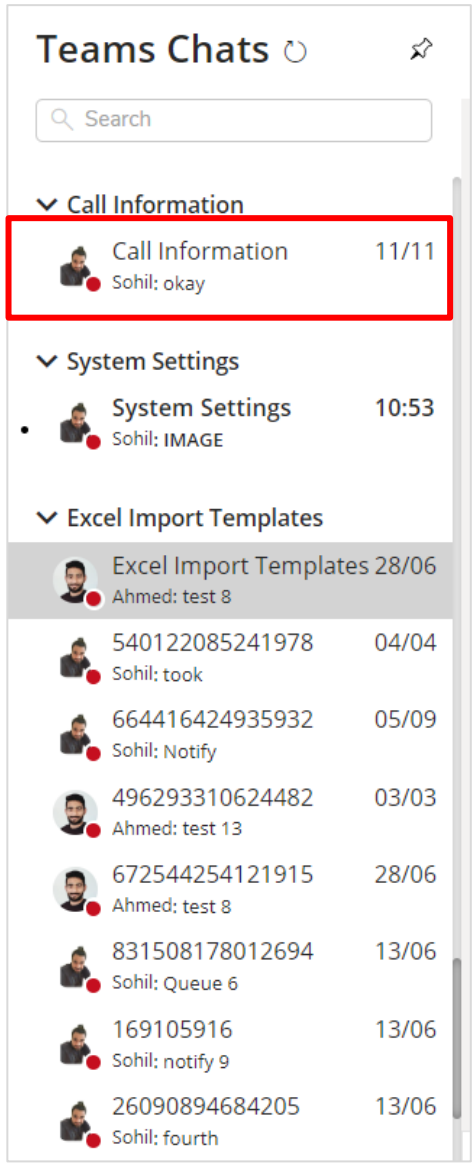


Graph Toolkit Provider

Proxy provider that will redirect all the request from the Graph toolkit components to my custom .NET Core API

```
Providers.globalProvider = new ProxyProvider(`${notificationAPIURL}`,
 async () => {
          return {
            "Content-Type" : "application/json",
            "Authorization" : "Bearer <token>"+
          };
        }
        );
```

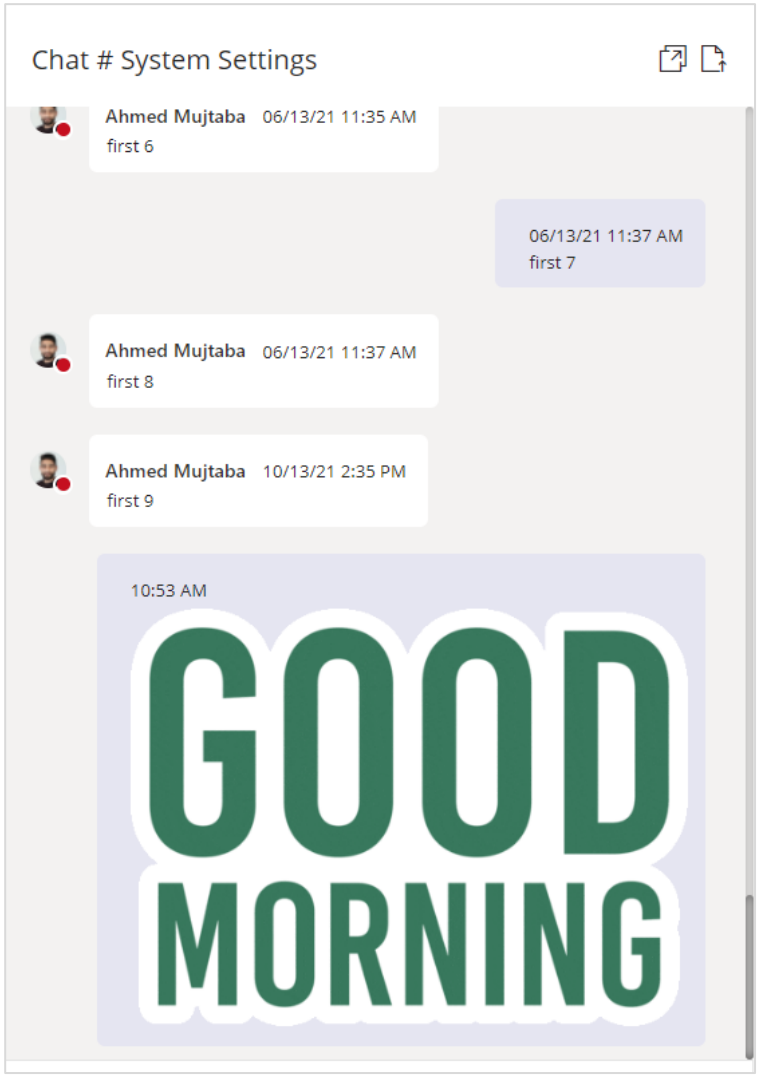
Left Side Screen Component




```
export const LastMessageItem = (props: any) => {

    return (
    <div className="teams-chat-last-message-container">
        <div className="teams-chat-last-message-container-block">
            <div className="teams-chat-conversation-message">
                <Person line1Property={"givenName"} fetchImage={true}
                        userId={props.dataContext.from.user.id}
                        showPresence={true} personCardInteraction={1}
                        view={PersonViewType.oneline}>
                </Person>
                <div data-testid="content" className={`teams-chat-last-message-
content>
                    {content}
                </div>
            </div>
        </div>
        <div className={`teams-chat-last-message-time`}>
            <created date time>
        </div>
    </div>);
}
```

Right Side Screen Component




```
<mgt-get cache-enabled={true}
    cache-invalidation-period={36000}
     id="messagesGet"
    version="beta"
    resource={"teams/<TeamID>/channels/<ChannelID>/messages/<MessageID>/replies"}>
    <template data-type="value"
        <Inside the template based on the graph response , rendering can happen like
        emojis , logged in user message UI and other users message UI as shown above
    </template>
 </mgt-get>
```

Back End Implementation (.NET 5)

Involves development of .NET core API and manages:

- ✓ **User authentication**
- ✓ **Graph change subscription**
- ✓ **SignalR management with groups**
- ✓ **Respond to all the graph calls**
  (acting as a provider in the Microsoft Graph toolkit proxy provider)

Graph Change Subscription

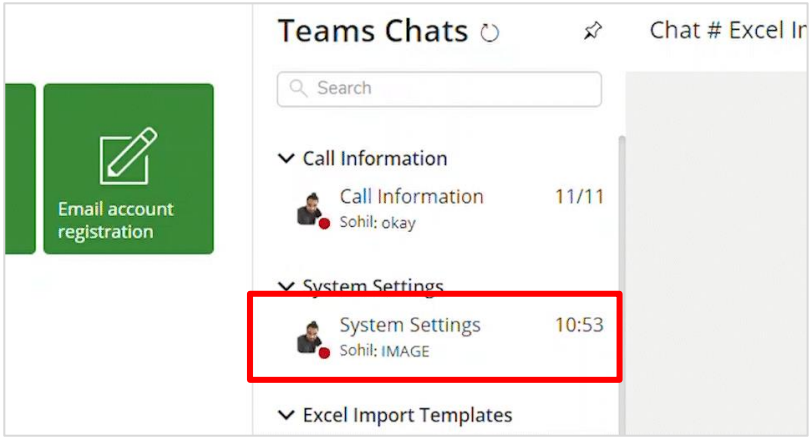
A subscription allows a client app to receive change notifications about the changes in data in Microsoft Graph.

Sample request to Graph:

```
https://graph.microsoft.com/v1.0/subscriptions
{
    "changeType": "created,updated,deleted",
    "notificationUrl": "<API Endpoint URL>",
    "resource": "teams/<teamid>/channels/<channelid>",
    "expirationDateTime": "2021-06-24T12:45:45.9356913Z",
    "clientState": "Y29kZWxlc3M=",
    "latestSupportedTlsVersion": "v1_2"
}
```

SignalR Implementation

SignalR is used to send notifications to the client for any new messages posted on the MS Teams channel




```
public void AddClientToSignalRGroup(string groupName)
{
    if (hub.Context != null)
    {
        string connectionID = hub.Context.ConnectionId;
        hubContext.Groups.AddToGroupAsync(connectionID, groupName).Wait();
    }
}

public void SendSignalRMessage(string groupName,Notification signalRMessage)
{
    hubContext.Clients.Group(groupName).BroadcastMessage(signalRMessage);
}
```

Client Side Changes to Get latest messages from Teams

- ✓ The backend API is now ready to serve the client requests.
- ✓ Once the client receives a message from signalR, the MGT Get component will refresh the data by calling the refresh method of the MGT component.

```
newConnection.on('BroadcastMessage', (message:Notification) => {
          if(message.resource.includes("/replies("))
          {
              //refresh the MGT Get components to pull the latest changes
               var mgtElement = document.getElementById("<mgt parent
identifier").getElementsByTagName('mgt-get') as MgtGet
      mgtElement.refresh(false);
          }
          else
          {
              //Ignore the Notification
          }
```

Respond to All Graph Calls (Proxy Provider)

The backend API will expose an endpoint that responds to all the Graph calls from the client.

This includes:

- ✓ **Get new messages**
- ✓ **Profile information** (Including profile card)
- ✓ **Presence indicator**

```
[HttpGet]
        [Route("
{version}/teams/{teamId}/channels/{channelId}/messages/{conversationId}/replies")]
        public async Task<IActionResult> GetAsync(string version, string teamId,
                                                  string channelId, string
conversationId)
        {
```

Final Demo