

Microsoft Graph

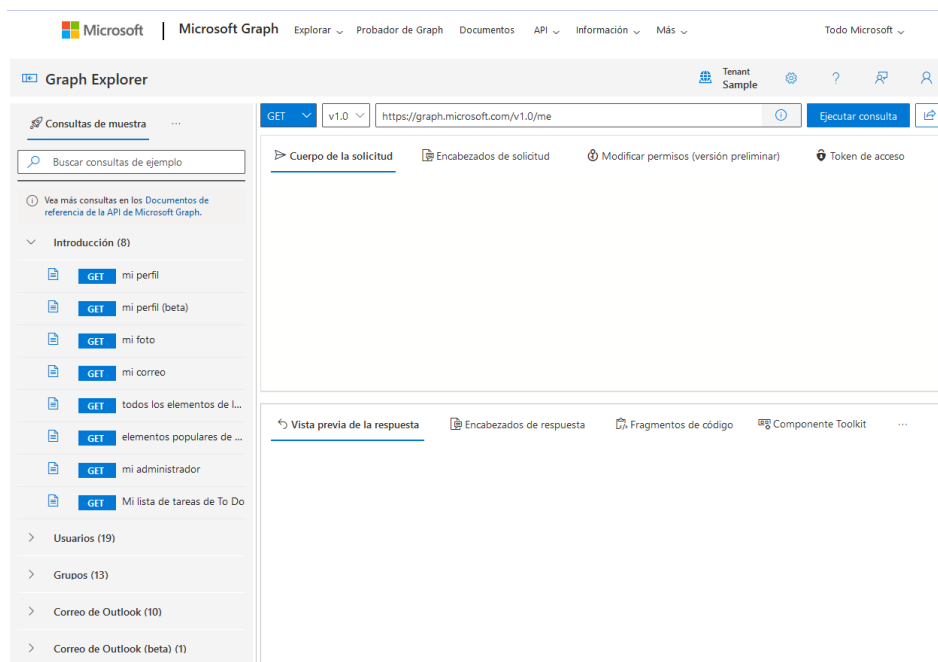
Contenido

Utilización del Probador de Microsoft Graph	2
Configuración de Cuenta.....	3
Modificadores de Acceso de Correo	6
Configuración en Postman	9
Configuración en Karate.....	13
Implementación de Graph en un proyecto.....	22
Creamos la clase donde hacemos la tarea para las funciones del graph.....	23
Creación de nuestra tarea para tomar el código del mensaje.....	25
Configuramos la clase StepDefinition	26

Microsoft Graph

Utilización del Probador de Microsoft Graph

Como primera medida es necesario ingresar al probador de Microsoft Graph para realizar las configuraciones y pruebas respectivas que nos permitirán entender el uso de esta herramienta, nos dirigimos al siguiente enlace <https://developer.microsoft.com/es-es/graph/graph-explorer>, aquí podremos ver una interfaz con un aspecto familiar.



En la parte superior se encuentra la barra para probar las diferentes URI de la API, con los diferentes métodos REST conocidos, es importante tener en cuenta que para utilizarla siempre debe comenzar con <https://graph.microsoft.com/v1.0> o <https://graph.microsoft.com/beta> para la versión beta, en este caso utilizaremos la versión 1.0; en la parte izquierda se encuentran todas las URI de ejemplo que nos proporciona la herramienta, puedes revisar dando clic en una de ellas, automáticamente se añade al campo de texto en la barra superior y damos clic en “Ejecuta consulta”, en este caso obtuvimos la imagen del perfil de cuenta por defecto que nos presenta la herramienta. Si quieres revisar más a detalle otros ejemplos puedes visitar el siguiente enlace <https://www.postman.com/microsoftgraph/workspace/microsoft-graph/overview>

Microsoft Graph

Graph Explorer

Consultas de muestra

Buscar consultas de ejemplo

Veá más consultas en los Documentos de referencia de la API de Microsoft Graph.

Introducción (8)

- GET mi perfil
- GET mi perfil (beta)
- GET mi foto**
- GET mi correo
- GET todos los elementos de l...
- GET elementos populares de ...
- GET mi administrador
- GET Mi lista de tareas de To Do

Usuarios (19)

Grupos (13)

Correo de Outlook (10)

GET v1.0 https://graph.microsoft.com/v1.0/me/photo/\$value

Possible error found in URL near: \$value

Ejecutar consulta

Cuerpo de la solicitud Encabezados de solicitud Modificar permisos (versión preliminar) Token de acceso

OK - 200 - 392ms

Vista previa de la respuesta Encabezados de respuesta Fragmentos de código Componente Toolkit

Actualmente utiliza una cuenta de muestra. Inicie sesión para tener acceso a sus propios datos:

Vista previa de la respuesta

Configuración de Cuenta

Para el uso de la herramienta es necesario utilizar un correo que pertenezca a Microsoft, ya sea Hotmail, Outlook, etc., nos dirigimos a la barra superior y damos clic en el ícono de usuario e ingresamos el correo que usaremos para leer la información de nuestro correo electrónico, nos pedirá permiso para acceder a nuestra información y damos en "Sí".

Tenant Sample

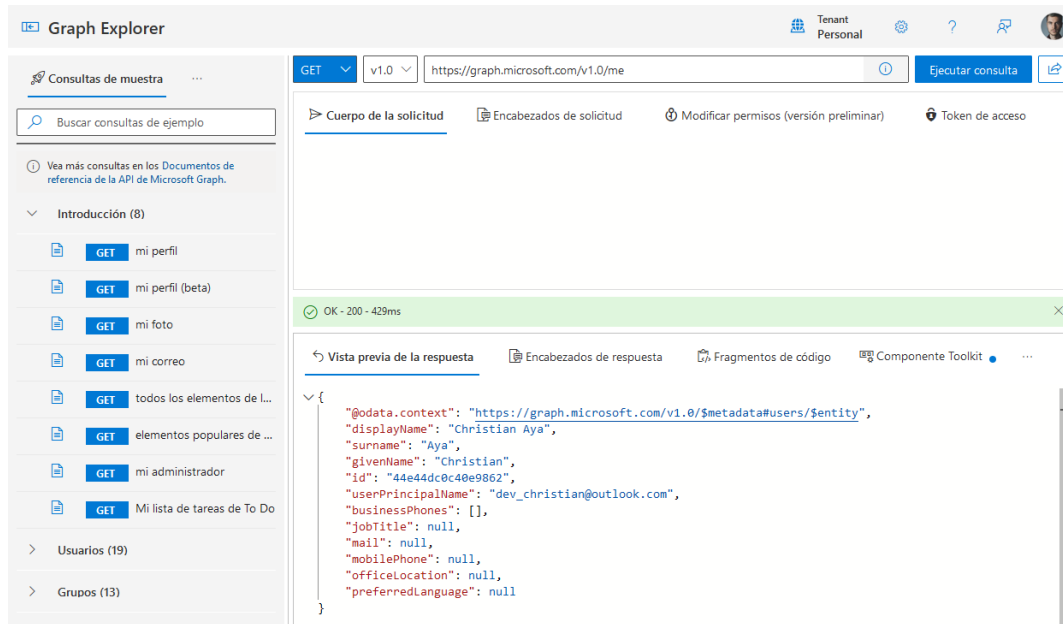
GET v1.0 https://graph.microsoft.com/v1.0/me

Ejecutar consulta

Cuerpo de la solicitud Encabezados de solicitud Modificar permisos (versión preliminar) Token de acceso

Microsoft Graph

Como podemos ver ya estamos usando la herramienta con nuestro usuario, para verificarlo ejecutamos la consulta de perfil dando clic en la opción “mi perfil” ubicada en el menú lateral y los datos de nuestra cuenta se presentarán en el campo de respuesta de la herramienta.



De la información resultante hay un campo importante que es el id, este nos permitirá realizar las consultas correspondientes a la cuenta asociada y si te has dado cuenta en la URI, el path asociado es “/me” que podría traducirse como “/users/{id}” puedes hacer la prueba, ambos te darán el mismo resultado; si quieres ser más específico puedes usarlo con tu identificador, pero el path “/me” tiene el mismo resultado.

Microsoft Graph

GET
v1.0
https://graph.microsoft.com/v1.0/users/44e44dc0c40e9862
Ejecutar consulta

Possible error found in URL near 44e44dc0c40e9862

Cuerpo de la solicitud
Encabezados de solicitud
Modificar permisos (versión preliminar)
Token de acceso

OK - 200 - 672ms

Vista previa de la respuesta
Encabezados de respuesta
Fragmentos de código
Componente Toolkit

```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity",
  "displayName": "Christian Aya",
  "surname": "Aya",
  "givenName": "Christian",
  "id": "44e44dc0c40e9862",
  "userPrincipalName": "dev_christian@outlook.com",
  "businessPhones": [],
  "jobTitle": null,
  "mail": null,
  "mobilePhone": null,
  "officeLocation": null,
  "preferredLanguage": null
}
```

Otro aspecto importante que debemos tener en cuenta es el token de acceso, este es fundamental para poder realizar todas la consultas, **IMPORTANTE el token de acceso tiene un tiempo de expiración de una hora**, por consiguiente, es necesario verificar su uso cada vez que realicemos una prueba u obtendremos un error de autenticación, para verlo nos dirigimos a la pestaña “Token de acceso” y lo copiaremos cada vez que queramos realizar una prueba ya sea en Postman o en Karate.

GET
v1.0
https://graph.microsoft.com/v1.0/me
Ejecutar consulta

Cuerpo de la solicitud
Encabezados de solicitud
Modificar permisos (versión preliminar)
Token de acceso

Token de acceso

```
EwBwA16BAALk3jnuJYtVthA+Hogk+HE1PbQc04AAa+htyq2HfhtjmfK821LQFCzqSeqeE7J55F5v1ffont58a15IgggPAV6F05dnwPq+V1XRakR0LZeb/6H1YIAIUEvcbGISpooY8AUFpCwLUZ0p6zdttekiwdhyps3JRipuk8008sCY
8w5f7yctYP2IBCB85N4+UJZ58HA12povou4Bwfrw2A/NEJv803mc1C8KjfhFcJahqA7SUJ2rE1AmuUfmezIqk0bn1FH
/US6wWYDfsw0XEn8d1FmZfrFgltg1Z0U9NpY2qK0F6cJ6j5u6gULR0L0K2mragz11jegk0bthK5C4fzr6VZen1u115voIDdjuRvEEDgAACFcvH5fbac59QWuaYTKBLEang84qHQFL1AvaVz505ckcAe5KA6onikng2xcZsnD
G'noorUuJot53QZLpLp58F/cbVtq8BoYttryjy0rVdHLCj7Ggz2P2P575ZLn1Y3UoVveFF28dN1b87g+q17bLunZvskimvFSPQdqlbawR0L8hz
/Q4prY2GI5xb8ZuUg0xvd7n3z3J5HsRbgybF2EysIOxCLVdamIqhcRyev38qE7NvACq8hfAV576J%un331DpT8t1F+Xie1edwdz3oCK9SyAA0Ry7NAONLVH+Umr6KnH8Bbkztu8G6ac62pAUthrIGuDLcnY1
/bZFPZ7h7Boq0csynonH0FuRm80q0wrv803uPerrtz1nIOzpdN4ZjKdLmpGT1p6FQpScue38gjbvS1s6aEynPGqkxpbLHRO2j28e3/5ENgh3/4/6jZ8v1SR1+g+gKQE7x90R8+GuuFdkhzUj45+uo
/9L1LeFk0421+fCUE8ccok+ux74GhFncU7xib801BVzif8vnb64z5n+7G2efyEnqhgW9G8E8Y1Phr0804aQ7hD32vT131IhpsuYzV8ro531y86bkhmYxunk+nK5GCMoCZHEm8gu3i0080A98VUP1nghtimFzAhf8D054IdFvptbtvias
7mC7dz7vC5x+fV8mDET21fTOYk1bZ8n8p4UPX9UFLBC0uwr9P/L3Y7xdiXk9vZLDgNYu4519N1dVub777ktm9Doy1z86zP45T/6np3CtbnH0Sg==
```

Microsoft Graph

Modificadores de Acceso de Correo

Para lograr obtener información de nuestro correo electrónico es necesario habilitar los permisos suficientes de acuerdo con las pruebas que queramos realizar, en este caso solo vamos a leer los mensajes (para saber más sobre los permisos visite este enlace <https://learn.microsoft.com/es-es/graph/permissions-reference>), para el ejemplo debemos habilitar el permiso Mail.Read, si ejecutamos la consulta para revisar los mensajes de nuestro correo nos responderá con un error de acceso denegado.

The screenshot shows the Microsoft Graph Explorer interface. At the top, there is a search bar with the URL `https://graph.microsoft.com/v1.0/me/messages` and a dropdown menu set to `GET`. Below the search bar, there are tabs for `Cuerpo de la solicitud`, `Encabezados de solicitud`, `Modificar permisos (versión preliminar)`, and `Token de acceso`. The `Modificar permisos (versión preliminar)` tab is selected. Below the tabs, there is a red error message: `Forbidden - 403 - 465ms. El usuario que ha iniciado sesión no tiene privilegios suficientes o usted debe dar su consentimiento a uno de los permisos en el Modificar permisos (versión preliminar) pestaña`. Below the error message, there are tabs for `Vista previa de la respuesta`, `Encabezados de respuesta`, `Fragmentos de código`, and `Componente Toolkit`. The `Vista previa de la respuesta` tab is selected. Below the tabs, there is a JSON response showing an error: `{ "error": { "code": "ErrorAccessDenied", "message": "Access is denied. Check credentials and try again." } }`.

Nos dirigimos a la pestaña “Modificar permisos” y daremos clic en el botón “Consentimiento” para el permiso Mail.Read, la herramienta nos desplegará una ventana en la que nos pide acceso a la información de lectura de nuestro correo y le damos que “Sí”, es importante que ejecute la consulta de mensajes para visualizar los permisos usados para este, de lo contrario podría aparecer una información diferente.

Microsoft Graph

GET

v1.0

https://graph.microsoft.com/v1.0/me/messages

Ejecutar consulta

Cuerpo de la solicitud

Encabezados de solicitud

Modificar permisos (versión preliminar)

Token de acceso

Permisos

Se requiere uno de los siguientes permisos para ejecutar la consulta. Seleccione un permiso y haga clic en Consentimiento

Permission	Descripción	Se requiere el consentimiento	Estado	Consent type
Mail.Read	Permite que la aplicación lea el correo electrónico de su buzón.	No	Consentimiento	
Mail.ReadBasic	Permite a la aplicación leer el correo en el buzón del usuario que haya iniciado sesión a excepción del cuerpo, previewBody, los datos adjuntos y las propiedades extendidas.	No	Consentimiento	
Mail.ReadWrite	Permite que la aplicación lea, actualice, cree y elimine correos electrónicos en su buzón. No incluye el permiso para enviar correos.	No	Consentimiento	

Microsoft

dev_christian@outlook.com

¿Permites el acceso a tu información?
Microsoft

Graph Explorer necesita tu permiso para:

Leer tu correo
Graph Explorer podrá leer el correo electrónico de tu buzón.

La aceptación de estos permisos implica que permite a esta aplicación usar sus datos, tal y como se especifica en las [condiciones de servicio](#) y la [declaración de privacidad](#). Estos permisos se pueden cambiar en <https://microsoft.com/consent>. [Mostrar detalles](#)

No

Sí

Verificamos nuevamente ejecutando la consulta de mensajes y nos podemos dar cuenta que ahora todos los mensajes con su metadata han sido leídos y se presentan en formato JSON.

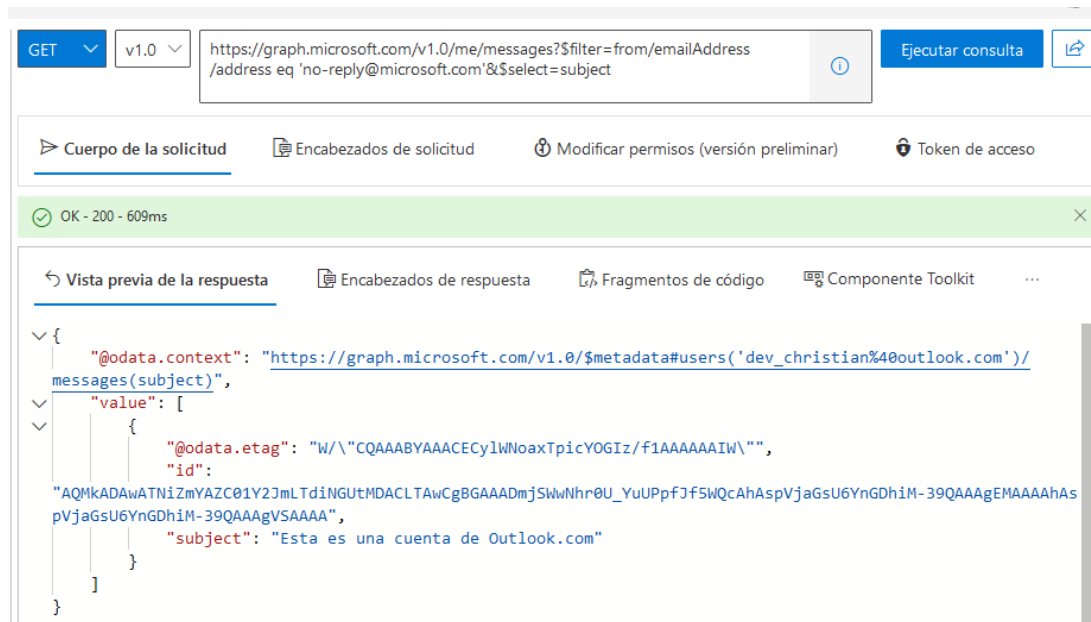
Microsoft Graph

The screenshot displays the Choucair Testing REST client interface. At the top, the method is set to GET, the version to v1.0, and the URL to https://graph.microsoft.com/v1.0/me/messages. Below the URL bar, there are tabs for 'Cuerpo de la solicitud', 'Encabezados de solicitud', 'Modificar permisos (versión preliminar)', and 'Token de acceso'. A green status bar indicates 'OK - 200 - 1090ms'. The main area shows the 'Vista previa de la respuesta' (Preview of the response) tab, displaying a JSON object representing a message. The JSON includes fields like '@odata.context', '@odata.etag', 'id', 'createdDateTime', 'lastModifiedDateTime', 'changeKey', 'categories', 'receivedDateTime', 'sentDateTime', 'hasAttachments', 'internetMessageId', 'subject', 'bodyPreview', 'importance', 'parentFolderId', 'conversationId', 'conversationIndex', 'isDeliveryReceiptRequested', and 'isReadReceiptRequested'. The 'bodyPreview' field contains a text message about Microsoft applications connecting to the account.

```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users('dev_christian%40outlook.com')/messages",
  "value": [
    {
      "@odata.etag": "W/\\\"CQAAABYAAACECylWNoaxTpicYOGIz/f1AAABNjOe\\\"\"",
      "id": "AQMKADAwATNiZmYAZC01Y2JmLTdiNGUtMDACLTAwCgBGAAADmJSWwNhr0U_YuUPpfJf5WQcAhAspVjaGsU6YnGDHiM-39QAAAgEMAAAHAspVjaGsU6YnGDHiM-39QAAAE3KfkAAAA=",
      "createdDateTime": "2023-02-01T15:58:19Z",
      "lastModifiedDateTime": "2023-02-01T15:58:21Z",
      "changeKey": "CQAAABYAAACECylWNoaxTpicYOGIz/f1AAABNjOe",
      "categories": [],
      "receivedDateTime": "2023-02-01T15:58:20Z",
      "sentDateTime": "2023-02-01T15:58:16Z",
      "hasAttachments": false,
      "internetMessageId": "<30LDRJ5J2JU4.DNSY0I33030N2@BL02EPF00001957>",
      "subject": "Nuevas aplicaciones conectadas a la cuenta de Microsoft",
      "bodyPreview": "Cuenta de MicrosoftNuevas aplicaciones con acceso a los datosGraph Explorer se ha conectado a la cuenta de Microsoft de**n@outlook.com.Si no se ha concedido este acceso, se deben eliminar las aplicaciones de la cuenta.Administrar las aplicaciones",
      "importance": "normal",
      "parentFolderId": "AQMKADAwATNiZmYAZC01Y2JmLTdiNGUtMDACLTAwCgAAADmJSWwNhr0U_YuUPpfJf5WQcAhAspVjaGsU6YnGDHiM-39QAAAgEMAAAA",
      "conversationId": "AQKADAwATNiZmYAZC01Y2JmLTdiNGUtMDACLTAwCgAQAD0JZv_XPjpKjTw4kw81gSQ=",
      "conversationIndex": "AQHZNlYAM41m/5c+OkqNPDirBzWB3A==",
      "isDeliveryReceiptRequested": null,
      "isReadReceiptRequested": false,
      "isForwarded": false
    }
  ]
}
```

Con esto ya podemos utilizar los parámetros de consulta que deseemos para personalizar la respuesta (para saber más sobre los parámetros de consulta visite el siguiente enlace <https://learn.microsoft.com/es-es/graph/query-parameters?tabs=http>), ya sea filtrar, contar, seleccionar, etc. y sus diferentes combinaciones. Por ejemplo, filtrar los correos de un remitente en específico y que se visualice únicamente el asunto.

Microsoft Graph



Configuración en Postman

Con la configuración de permisos realizada anteriormente ya podemos utilizar nuestras URI en otras aplicaciones, para este ejemplo veremos la configuración para utilizarlo en la herramienta Postman.

Crearemos una colección que contendrá la variable de entorno de nuestra URL y el token de acceso que se heredará a nuestras solicitudes. Para esto nos dirigimos a la pestaña de Collections y creamos una nueva colección, en la pestaña Authorization seleccionamos el tipo Bearer Token y pegamos el token que nos proporciona la API de Microsoft Graph.

Microsoft Graph

MicrosoftGraph Share Fork | 0 ▶ 💾 ⋮

Authorization • Pre-request Script Tests Variables Runs

This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

Type Bearer Token ▼

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#) ↗

Token EwBwA8l6BAAUkj1NuJYtTVha+Mog...

Ahora en la pestaña de Variables crearemos una variable que apuntará a la URL de la API, mencionada anteriormente <https://graph.microsoft.com/v1.0>, fíjese que no colocamos el path `/me` a pesar de que es necesario para realizar la consulta de nuestros mensajes, ya que debemos tener en cuenta que hay otros path que se asocian a consultas diferentes, sí lo desea puede adicionarlo, pero recuerde que para probar casos como users o groups debe quitarlo.

MicrosoftGraph Share Fork | 0 ▶ 💾 ⋮

Authorization • Pre-request Script Tests Variables • Runs

These variables are specific to this collection and its requests. Learn more about [collection variables](#) ↗

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	⋮	Persist All	Reset All
<input checked="" type="checkbox"/>	urlBase	https://graph.microsoft.com/v1.0	https://graph.microsoft.com/v1.0			

[Add a new variable](#)

Con nuestras configuraciones realizadas ya podemos guardar nuestra colección y realizar la prueba realizando la consulta de nuestro perfil. Recuerde que para utilizar los datos de

Microsoft Graph

configuración debemos guardar nuestras consultas en la colección que acabamos de crear.

The screenshot shows the Choucair Testing interface. At the top, the URL is `{{urlBase}}/me`. A red box highlights the 'Save' button. Below the URL bar, the method is 'GET' and the URL is `{{urlBase}}/me`. A 'Send' button is visible. The 'Params' tab is selected, showing a table with columns: KEY, VALUE, DESCRIPTION, and Bulk Edit. The table has one row with 'Key' and 'Value'. Below the table, the 'SAVE REQUEST' dialog is open. The 'Request name' field contains 'GetProfile'. The 'Add description' link is visible. The 'Save to' dropdown is set to 'Ejemplos / MicrosoftGraph', which is highlighted with a red box.

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

SAVE REQUEST

Request name

GetProfile

[Add description](#)

Save to Ejemplos / **MicrosoftGraph**

Sí todo ha sido correcto podremos visualizar la información de nuestro perfil tal cual como lo hicimos en el probador de Microsoft Graph.

The screenshot shows the Choucair Testing interface with the 'Body' tab selected. The response status is '200 OK' with a response time of '799 ms' and a size of '920 B'. The response body is displayed in a code editor with a 'Pretty' format. The JSON response contains user profile information.

```
1  {
2    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity",
3    "displayName": "Christian Aya",
4    "surname": "Aya",
5    "givenName": "Christian",
6    "id": "44e44dc0c40e9862",
7    "userPrincipalName": "dev_christian@outlook.com",
8    "businessPhones": [],
9    "jobTitle": null,
10   "mail": null,
11   "mobilePhone": null,
12   "officeLocation": null,
13   "preferredLanguage": null
14 }
```

Microsoft Graph

Ahora veremos la consulta que realizamos filtrando los correos de un remitente en específico y que se visualice únicamente el asunto. Como podemos ver con Postman podemos optimizar el ingreso de los parámetros de consulta.

The screenshot shows a Postman interface for a GET request to the Microsoft Graph endpoint: `{{urlBase}}/me/messages?$filter=from/emailAddress/address eq 'account-security-n`. The request is configured with the following query parameters:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> \$filter	from/emailAddress/address eq 'acc...	
<input checked="" type="checkbox"/> \$select	subject	

The response is a 200 OK status with a 177 ms response time and 1.21 KB of data. The response body is displayed in JSON format, showing a list of messages with their subjects and IDs.

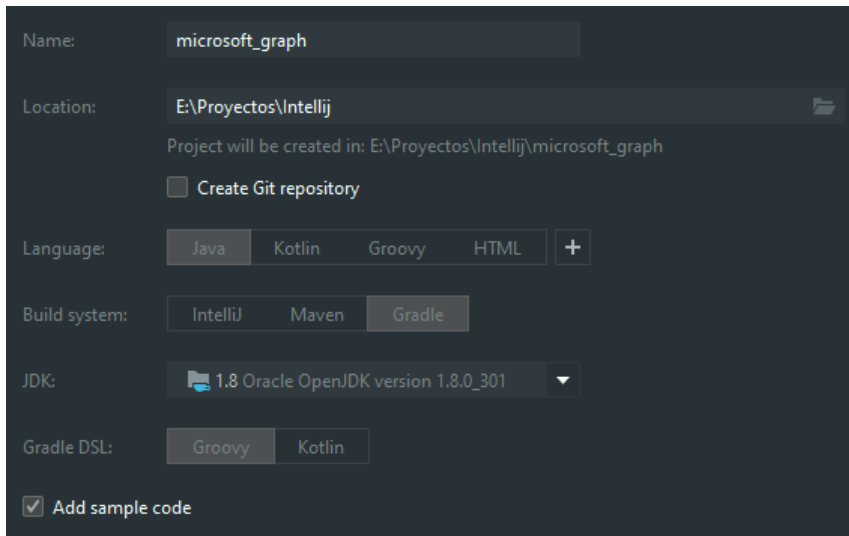
```
6      "id":  
7      "AQMKADAwATNiZmYAZC01Y2JmLTdiNGUtMDACLTAwCgBGAAADmjSkwNhr0U_YuUPpfJf5WQcAhAspVjaGsU  
8      6YnGDhiM-39QAAAgEMAAAAhAspVjaGsU6YnGDhiM-39QAAAAE3KfgAAAA=",  
9      "subject": "Nuevas aplicaciones conectadas a la cuenta de Microsoft"  
10     },  
11     {  
12       "@odata.etag": "W/\\\"CQAAABYAAACECy1wNoaxTpicYOGIz/f1AAABNj0e\\\"",  
13       "id":  
14       "AQMKADAwATNiZmYAZC01Y2JmLTdiNGUtMDACLTAwCgBGAAADmjSkwNhr0U_YuUPpfJf5WQcAhAspVjaGsU  
15       6YnGDhiM-39QAAAgEMAAAAhAspVjaGsU6YnGDhiM-39QAAAAE3KfgAAAA=",  
16       "subject": "Nuevas aplicaciones conectadas a la cuenta de Microsoft"  
17     }  
18   ]  
19 }
```

Recuerde que el token tiene un tiempo de vigencia, por consiguiente, es necesario modificarlo cada vez que expire en la configuración de autorización que realizamos para la colección.


Microsoft Graph

Configuración en Karate

Creemos un nuevo proyecto



Name:



Location: 

Project will be created in: E:\Proyectos\IntelliJ\microsoft_graph

☐ Create Git repository

Language:

Build system:

JDK:  1.8 Oracle OpenJDK version 1.8.0_301 

Gradle DSL:

☒ Add sample code

Buscamos el archivo build.gradle y añadimos la siguiente información

```
plugins {  
    id 'java'  
}  
  
compileJava {  
    sourceCompatibility = 1.8  
    targetCompatibility = 1.8  
}  
  
ext {  
    karateVersion = '1.3.1'  
}  
  
dependencies {  
  
    testImplementation "com.intuit.karate:karate-junit5:${karateVersion}"  
    testImplementation 'net.masterthought:cucumber-reporting:5.7.4'  
}
```

Microsoft Graph

```
tasks.withType(JavaCompile) {
    options.encoding = 'UTF-8'
}

sourceSets {
    test {
        resources {
            srcDir file('src/test/java')
            exclude '**/*.java'
        }
    }
}

test {
    useJUnitPlatform() // junit5
    systemProperty "karate.env", System.properties.getProperty("karate.env")
    systemProperties = System.properties
    outputs.upToDateWhen { false }
}

repositories {
    mavenCentral()
}

gradle.startParameter.refreshDependencies = true
```

Como podemos observar en las dependencias hemos implementado la librería necesaria para utilizar la API de Microsoft Graph y con esta, la librería que nos permite autenticarnos por medio del token.

```
implementation 'com.microsoft.graph:microsoft-graph:5.45.0'
implementation 'com.azure:azure-identity:1.2.5'
```

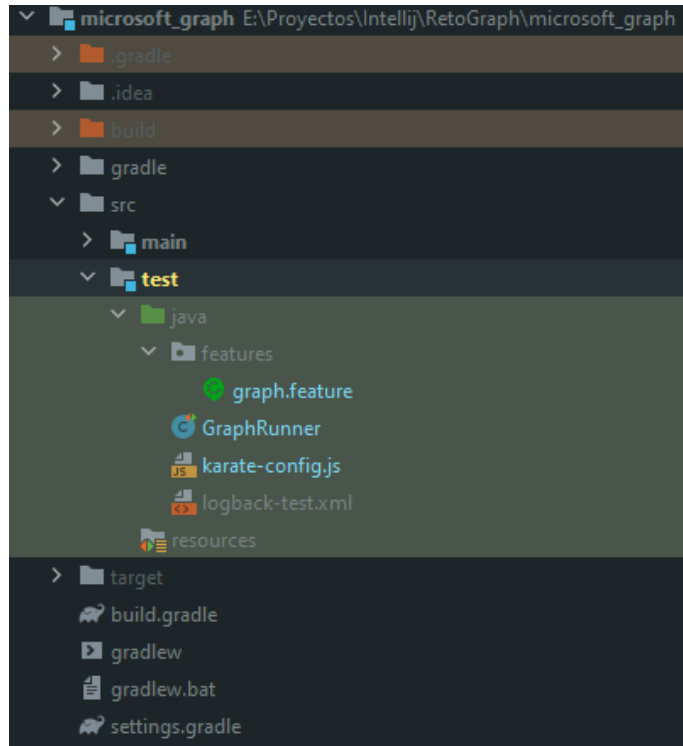
Dentro de la carpeta test/java crearemos los siguientes archivos

- Un paquete con el nombre features y dentro de este nuestra feature, para el ejemplo tendrá el nombre de graph.feature
- Una clase java para ejecutar nuestro runner, para el ejemplo GraphRunner
- Un archivo js llamado karate-config.js

Microsoft Graph

- Un archivo logback-test.xml

Nuestro árbol de proyecto debería verse de la siguiente manera



Dentro del archivo logback-test.xml ingresaremos el siguiente código:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} -
%msg%n</pattern>
    </encoder>
  </appender>
  <logger name="com.intuit.karate" level="DEBUG"/>
  <root level="info">
    <appender-ref ref="STDOUT" />
  </root>
</configuration>
```

Microsoft Graph

Para el archivo karate-config.js ingresaremos:

```
function fn() {  
  karate.configure('connectTimeout', 10000);  
  karate.configure('readTimeout', 10000);  
  karate.configure('ssl', true);  
  var env = karate.env; // get java system property 'karate.env' in build.gradle  
  var account = env == 'dev' ? '360735510274' : '278078741213';  
  karate.log('karate.env system property was:', env);  
  
  return {  
    graph: {  
      token: "  
    }  
  };  
}
```

Tenga en cuenta que dentro las comillas simples irá el token que le proporciona el probador de Microsoft Graph, este campo no puede ir vacío.

Dentro de nuestro archivo GraphRunner ingresaremos:

```
import com.intuit.karate.Results;  
import com.intuit.karate.Runner;  
import net.masterthought.cucumber.Configuration;  
import net.masterthought.cucumber.ReportBuilder;  
import org.apache.commons.io.FileUtils;  
import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.Test;  
  
import java.io.File;  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.List;  
  
public class GraphRunner {  
  @Test  
  void testParallel() {  
    Results results = Runner.path("classpath:features/graph.feature")  
      .outputCucumberJson(true)  
      .tags("")  
      .parallel(4);  
  }  
}
```


Microsoft Graph

```
generateReport(results.getReportDir());
Assertions.assertEquals(0, results.getFailCount(), results.getErrorMessages());
}

public static void generateReport(String karateOutputPath) {
    Collection<File> jsonFiles = FileUtils.listFiles(new File(karateOutputPath), new
String[]{"json"}, true);
    List<String> jsonPaths = new ArrayList<String>(jsonFiles.size());
    jsonFiles.forEach(file -> jsonPaths.add(file.getAbsolutePath()));
    Configuration config = new Configuration(new File("target"), "Microsoft Graph Test");
    ReportBuilder reportBuilder = new ReportBuilder(jsonPaths, config);
    reportBuilder.generateReports();
}
```

Por el momento el campo que corresponde al tag del escenario que queremos probar está vacío, más adelante lo llenaremos con los valores correspondientes.

Ahora crearemos la Feature con los escenarios de acuerdo con los ejemplos que hemos visto anteriormente, la comprobación de nuestro perfil y el filtro de nuestros mensajes.

Como primer paso añadiremos el Background de nuestra feature en la cual irá la información para acceder a la API, estos datos son los mismos que hemos usado anteriormente, la URL base y el token de acceso.

Feature: Microsoft Graph request messages email

Background:

```
* url "https://graph.microsoft.com/v1.0"
* header Authorization = 'Bearer '+ graph.token
```

Como podemos ver, el dato que contiene la información del token es el que añadimos en nuestro archivo karate-config.js y lo invocamos como graph.token. Es importante que cada vez que expire el token lo modifiquemos en el archivo.

Ahora creamos el escenario para obtener la información de nuestro perfil, como se dijo anteriormente, en nuestra urlBase podemos añadir el path /me o lo podemos adicionar dentro del escenario.

Microsoft Graph

```
@GetProfile
Scenario: Get user profile
  Given path "/me"
  When method GET
  Then status 200
  * print response.displayName
```

Ahora vamos a comprobar que la información sea correcta. Añadimos el tag dentro del campo correspondiente de nuestra clase GraphRunner y ejecutamos.

```
@Test
void testParallel() {
    Results results = Runner.path( ...paths: "classpath:features/graph.feature")
        .outputCucumberJson( value: true)
        .tags("@GetProfile")
        .parallel( threadCount: 4);
    generateReport(results.getReportDir());
    Assertions.assertEquals( expected: 0, results.getFailCount(), results.getErrorMessages())
}
```

Como podemos ver obtenemos la información de nuestro perfil de la misma manera que hemos visto anteriormente y la impresión del dato de nuestro nombre.

```
1 < Date: Wed, 01 Feb 2023 18:49:59 GMT
{"odata.context":"https://graph.microsoft.com/v1.0/$metadata#users/$entity","displayName":"Christian Aya",
 "surname":"Aya","givenName":"Christian","id":"44e44dc0c40e9862","userPrincipalName":"dev_christian@outlook.com",
 "businessPhones":[],"jobTitle":null,"mail":null,"mobilePhone":null,"officeLocation":null,"preferredLanguage":null}

13:49:59.348 [pool-1-thread-1] INFO  com.intuit.karate - [print] Christian Aya
```

Sí queremos visualizar el reporte de manera más ordenada, podemos abrir el que proporciona cucumber y nos daremos cuenta de que obtenemos la misma información, para ello nos dirigimos a la carpeta target/cucumber-html-reports/ y abrimos el archivo overview-features.html en el navegador de nuestra preferencia

Microsoft Graph

Cucumber Report

FeaturesTagsStepsFailures

Project	Date
Microsoft Graph Test	01 feb 2023, 13:50

Feature Report

Feature	Steps						Scenarios			Features	
	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
features/graph.feature	6	0	0	0	0	6	1	0	1	1.250	Passed

Feature features/graph.feature

Microsoft Graph request messages email

Background > 0.035

Tags: @GetProfile

Scenario Get user profile 1.215

Steps >

Given path "/me"0.001

When method GET1.206

Doc string

Then status 2000.000

* print response.displayName0.008

Doc string

13:49:59.348 [print] Christian Aya

Ahora crearemos nuestro escenario para filtrar los mensajes y seleccionar el asunto. Podemos crear una variable que almacene el valor del correo del destinatario y el path contendrá la ruta que llamará nuestros mensajes, añadiremos los parámetros de filter y select por medio de la palabra param asignándoles los valores necesarios para realizar la consulta.

```
@GetEmails
Scenario: Get list messages
* def fromAddress = 'account-security-noreply@accountprotection.microsoft.com'
Given path "/me/messages"
And param filter = "(from/emailAddress/address) eq '" + fromAddress + "'"
And param select = "subject"
When method GET
Then status 200
* print response
```

Ahora vamos a comprobar que la información sea correcta. Añadimos el tag dentro del campo correspondiente de nuestra clase GraphRunner y ejecutamos.

Microsoft Graph

```
@Test
void testParallel() {
    Results results = Runner.path( ...paths: "classpath:features/graph.feature")
        .outputCucumberJson( value: true)
        .tags("@GetEmails")
        .parallel( threadCount: 4);
    generateReport(results.getReportDir());
    Assertions.assertEquals( expected: 0, results.getFailCount(), results.getErrorMessage());
}
```

Como podemos ver, obtenemos la información de nuestros mensajes con los parámetros de consulta indicados de la misma manera que hemos visto anteriormente.

```
14:02:40.038 [pool-1-thread-1] INFO com.intuit.karate - [print] {
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users('dev_christian%40outlook.com')/messages",
  "value": [
    {
      "@odata.etag": "W/\"CQAAABYAAACECyLWNoaxTpicYOGIz/f1AAABNj0D\\\"",
      "id": "AQMKADAwATNiZmYAZC01Y2JmLTdiNGUtMDACLTAwCgBGAAADmjSWwNhrOU_YuUPpfJf5WQcAhAspVjaGsU6YnGDhiM-39QAAAgEMAAAHhAspVjaGsU6YnGDhiM-39QAAAAE3KfgAAAA=",
      "subject": "Nuevas aplicaciones conectadas a la cuenta de Microsoft"
    },
    {
      "@odata.etag": "W/\"CQAAABYAAACECyLWNoaxTpicYOGIz/f1AAABNj0e\\\"",
      "id": "AQMKADAwATNiZmYAZC01Y2JmLTdiNGUtMDACLTAwCgBGAAADmjSWwNhrOU_YuUPpfJf5WQcAhAspVjaGsU6YnGDhiM-39QAAAgEMAAAHhAspVjaGsU6YnGDhiM-39QAAAAE3KfkAAAA=",
      "subject": "Nuevas aplicaciones conectadas a la cuenta de Microsoft"
    }
  ]
}
```

Sí queremos visualizar el reporte de manera más ordenada podemos abrir el que nos proporciona cucumber y nos daremos cuenta de que obtenemos la misma información.

Microsoft Graph

Feature features/graph.feature	
Microsoft Graph request messages email	
Background >	0.020
Tags: @GetEmails	
Scenario Get list messages v	0.979
Steps v	
* def fromAddress = 'account-security-noreply@accountprotection.microsoft.com'	0.001
Given path "/me/messages"	0.000
And param filter = "(from/emailAddress/address) eq '" + fromAddress + "'"	0.001
And param select = "subject"	0.000
When method GET	0.967
Doc string	
Then status 200	0.000
* print response	0.007
Doc string	
<pre>14:02:40.041 [print] { "@odata.context": "https://graph.microsoft.com/v1.0/\$metadata#users('dev_christian%40outlook.com')/messages(subject)", "value": [{ "@odata.etag": "W/\"CQAAABYAAACECy1wNoaxTpicYOGIz/f1AAABNj00\"\"", "id": "AQHkADAwATNIZmYAZC01Y2JmLTdINGUtMDACLTAwCgBGAAADmJ5wMhR0U_YuUPpfJf5WQcAhAspVjaGsU6YnGDhiM-39QAAAgEMAAAAhAspVjaGsU6Y", "subject": "Nuevas aplicaciones conectadas a la cuenta de Microsoft" }, { "@odata.etag": "W/\"CQAAABYAAACECy1wNoaxTpicYOGIz/f1AAABNj0e\"\"", "id": "AQHkADAwATNIZmYAZC01Y2JmLTdINGUtMDACLTAwCgBGAAADmJ5wMhR0U_YuUPpfJf5WQcAhAspVjaGsU6YnGDhiM-39QAAAgEMAAAAhAspVjaGsU6Y", "subject": "Nuevas aplicaciones conectadas a la cuenta de Microsoft" }] }</pre>	

De esta manera ya somos capaces de acceder a la API de Microsoft Graph, visualizar nuestra información con la herramienta de Postman y realizar nuestras respectivas pruebas con el framework de Karate.

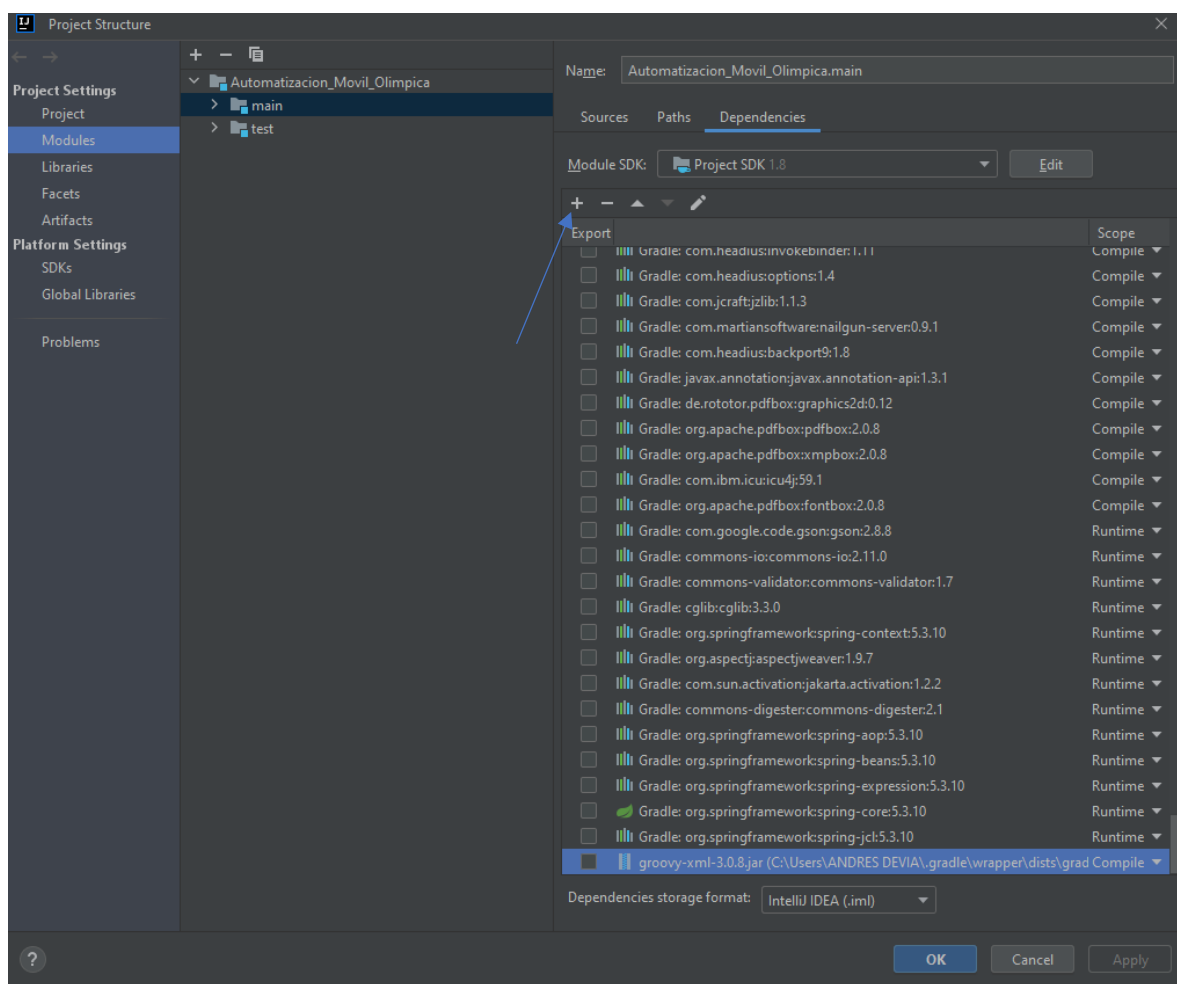
Microsoft Graph

Implementación de Graph en un proyecto

Descargamos groovy-xml JAR 3.0.8 y groovy JAR 3.0.8 de la siguiente pagina <https://jar-download.com/artifacts/org.codehaus.groovy/groovy-xml/3.0.8/source-code>

Nota: en el archivo zip vienen los dos groovy.

Luego lo añadimos a nuestro proyecto, dando clic en File luego Project Structure, y por último añadimos los dos archivos en la carpeta main y test.



Microsoft Graph

Luego en la clase build.gradle agregamos la dependencia de Serenity

```
implementation 'net.serenity-bdd:serenity-screenplay-rest:3.1.16'
```

por último, en configuration.all la forzamos

```
force 'net.serenity-bdd:serenity-screenplay-rest:2.6.0'
```

Creamos la clase donde hacemos la tarea para las funciones del graph

En esta clase definimos el token extraído de nuestro probador de Microsoft Graph;

Definimos el dominio del correo del cual esperamos que nos envíen el mensaje con el código.

Creamos nuestro método estático y definimos las funciones del actor donde por medio del método Get.resource le pasamos el path de nuestra url;

Luego le pasamos la autorización mas el token de acceso;

En la siguiente línea le decimos que solo nos traiga el primer correo;

Con la línea select subject le indicamos que solo nos seleccione el asunto de dicho mensaje;

Luego le indicamos que nos filtre solo los correos cuyo remitente sea del dominio definido en la variable correo;

Luego le indicamos que nos organice los correos del más actual al más viejo;

Por último, retornamos esa variable request que contendría todo el resultado de el proceso que acabamos de hacer.

Microsoft Graph

Nuestra clase debería quedar de esta manera:

```
Graph.java
1 package co.com.choucair.certification.automatizacionmovilolimpica.tasks;
2
3 import net.serenitybdd.screenplay.Actor;
4 import net.serenitybdd.screenplay.Task;
5 import net.serenitybdd.screenplay.Tasks;
6 import net.serenitybdd.screenplay.rest.interactions.Get;
7
8 // 22PIPE
9 public class Graph implements Task {
10
11     1 usage
12     private final String TOKEN = "EmBoA8l6BAAUA0yDv0L6PcCVuB9kmzvqZmkWABkAAZnN8YuuIfc8eof2cw+Vd3H6j8oyI+mZafktjLWDYpZQK13+canYLGLOt0tJ";
13     1 usage
14     private final String CORREO = "noreply@vtexcommerce.com.br";
15     1 usage // 22PIPE
16     public static Graph paginaGraph() { return Tasks.instrumented(Graph.class); }
17
18     // 22PIPE
19     @Override
20     public <T extends Actor> void performAs(T actor) {
21
22         actor.attemptsTo(
23             Get.resource("/me/messages")
24                 .with(request -> {
25                     request.header( headerName: "Authorization", headerValue: "Bearer " + TOKEN);
26                     request.param( parameterName: "$top", ...parameterValues: "1");
27                     request.param( parameterName: "$select", ...parameterValues: "subject");
28                     request.param( parameterName: "$filter", ...parameterValues: "(from/emailAddress/address) eq '" + CORREO + "'");
29                     request.param( parameterName: "$orderby", ...parameterValues: "(from/emailAddress/address) desc");
30                     return request;
31                 })
32         );
33     }
34 }
```


Microsoft Graph

Creación de nuestra tarea para tomar el código del mensaje

Creamos nuestra clase para definir la tarea para tomar el mensaje con el código.

Empezamos definiendo las variables donde vamos a almacenar nuestro código y nuestra contraseña para completar la automatización.

Creamos el constructor e inicializamos dichas variables;

Creamos nuestro método estático;

Definimos las acciones de nuestro actor;

Ya teniendo mapeados los campos donde vamos a introducir el código y la contraseña, empezamos ingresando el código traído por la API de Microsoft Graph en su respectivo campo, luego le damos clic al botón para enviar el código, de ser aprobado este código proseguimos ingresando nuestra contraseña y por último dando clic en el botón para registrar nuestra contraseña y finalizar con la automatización la cual consistió en el registro de un nuevo usuario en la app Olímpica.

Nuestra clase debería quedarnos de la siguiente manera:

Microsoft Graph

```
11 import static co.com.choucair.certification.automatizacionmovilolimpica.userinterface.TextoConfirmacionRegistro.TEXTO_CONFIRMACION_REGISTRO;
12 import static net.serenitybdd.screenplay.matchers.WebElementStateMatchers.isCurrentlyVisible;
13
14 3 usages 22PIPE *
15 public class LlenarCodigo implements Task {
16
17 2 usages
18 private String codigo;
19 3 usages
20 private String contraseña;
21
22 new *
23 public LlenarCodigo(String codigo, String contraseña) {
24     this.codigo = codigo;
25     this.contraseña = contraseña;
26 }
27
28 1 usage 22PIPE *
29 public static LlenarCodigo llenarCodigo(String codigo, String contraseña) {
30     return Tasks.instrumented(LlenarCodigo.class, codigo, contraseña);
31 }
32
33 22PIPE *
34 @Override
35 public <T extends Actor> void performAs(T actor) {
36
37     actor.attemptsTo(
38         WaitUntil.the(LABEL_CODIGO, isCurrentlyVisible()).forNoMoreThan( amount: 15).seconds(),
39         Enter.theValue(codigo).into(LABEL_CODIGO),
40         Click.on(BOTON_ENVIAR_CODIGO),
41         WaitUntil.the(LABEL_CONTRASEÑA, isCurrentlyVisible()).forNoMoreThan( amount: 15).seconds(),
42         Enter.theValue(contraseña).into(LABEL_CONTRASEÑA),
43         Enter.theValue(contraseña).into(LABEL_CONFIRMAR_CONTRASEÑA),
44         Click.on(BOTON_CONTRASEÑA),
45         WaitUntil.the(TEXTO_CONFIRMACION_REGISTRO, isCurrentlyVisible()).forNoMoreThan( amount: 15).seconds()
46     );
47 }
48 }
```

Configuramos la clase StepDefinition

Empezamos definiendo una variable de tipo String donde guardaremos la URL de la API de Microsoft;

Luego en nuestro método And donde recibimos el código, le decimos que va a recibir un String el cual será la contraseña, llamamos la URL y por último le decimos que vaya a la página de Graph.

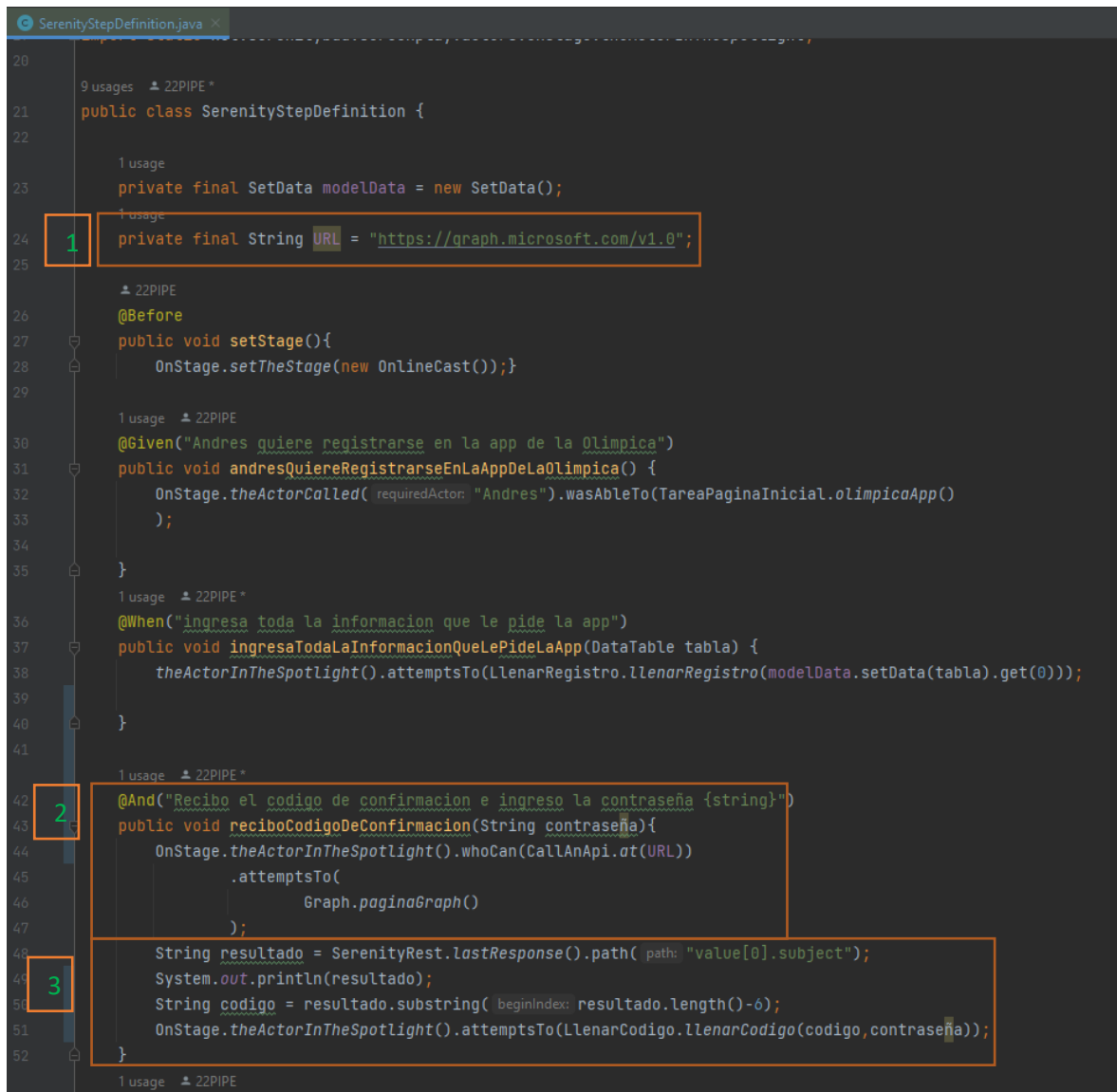
Luego creamos una variable de tipo String, donde vamos a almacenar la respuesta de nuestra petición y le decimos que solo nos traiga el asunto del correo.

Continuamos creando una variable de tipo String donde guardaremos el código que nos traiga la API, le decimos que nos traiga los últimos seis caracteres que contenga el asunto de nuestro mensaje, esos últimos 6 caracteres serán el código;

Por último, llamamos la clase que contiene la tarea de llenar el código y la contraseña.

Microsoft Graph

Nuestro método debería quedarnos de la siguiente manera:



```
20
21 9 usages 22PIPE *
22  public class SerenityStepDefinition {
23      1 usage
24      private final SetData modelData = new SetData();
25      1 usage
26      private final String URL = "https://graph.microsoft.com/v1.0";
27
28      22PIPE
29      @Before
30      public void setStage(){
31          OnStage.setTheStage(new OnLineCast());}
32
33      1 usage 22PIPE
34      @Given("Andres quiere registrarse en la app de la Olimpica")
35      public void andresQuiereRegistrarseEnLaAppDeLaOlimpica() {
36          OnStage.theActorCalled( requiredActor: "Andres").wasAbleTo(TareaPaginaInicial.olimpicaApp()
37          );
38
39      }
40
41      1 usage 22PIPE *
42      @When("ingresa toda la informacion que le pide la app")
43      public void ingresaTodaLaInformacionQueLePideLaApp(DataTable tabla) {
44          theActorInTheSpotlight().attemptsTo(LlenarRegistro.llenarRegistro(modelData.setData(tabla).get(0)));
45
46      }
47
48      1 usage 22PIPE *
49      @And("Recibo el codigo de confirmacion e ingreso la contraseña {string}")
50      public void reciboCodigoDeConfirmacion(String contraseña){
51          OnStage.theActorInTheSpotlight().whoCan(CallAnApi.at(URL))
52          .attemptsTo(
53              Graph.paginaGraph()
54          );
55
56      String resultado = SerenityRest.lastResponse().path( path: "value[0].subject");
57      System.out.println(resultado);
58      String codigo = resultado.substring( beginIndex: resultado.length()-6);
59      OnStage.theActorInTheSpotlight().attemptsTo(LlenarCodigo.llenarCodigo(codigo, contraseña));
60
61      }
62
63      1 usage 22PIPE
```

The screenshot shows a Java code editor with the file name 'SerenityStepDefinition.java'. The code defines a class 'SerenityStepDefinition' with several methods. Three sections are highlighted with orange boxes and numbered 1, 2, and 3. Section 1 highlights the URL constant. Section 2 highlights the 'reciboCodigoDeConfirmacion' method. Section 3 highlights the final part of the 'ingresaTodaLaInformacionQueLePideLaApp' method, including the extraction of the confirmation code and the final attempt to fill the code.