Technical University of Cluj-Napoca

Faculty of Electronics, Telecommunications and Information Technology

Proiectare avansata in retelele de calculatoare

Analiza servicii REST-API oferite de Cisco Spark

Coordonator: Conf.dr.ing. Daniel ZINCA

Student: Andreica Maria Antonia

INTRODUCERE

1. REST APIs

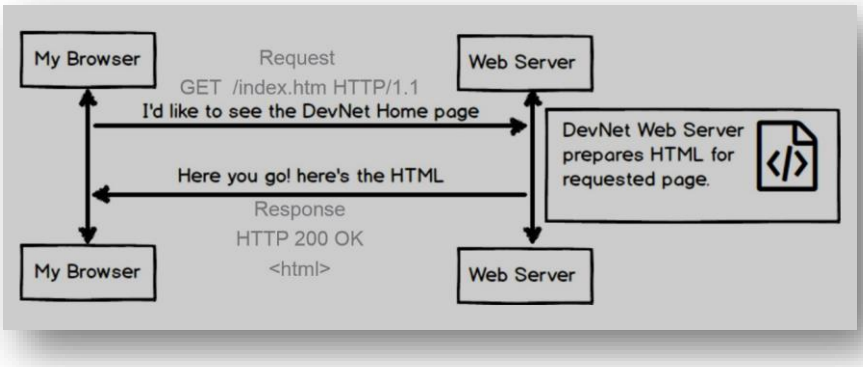
Application Programming Interface reprezintă un set de definiții de sub-programe, protocoale si unelte pentru programarea de aplicații si software. Un API poate fi folosit pentru un sistem web, sistem de operare, sistem de baze de date, hardware sau biblioteci software. În termeni generali, este vorba despre un set de metode de comunicare între diferitele componente software.

REST(Representational State Transfer) este un stil de arhitectură de aplicație care, în loc să impună decizii asupra tehnologiei, preferă să definească un set de constrângeri la care sistemul să adere. În felul acesta, detaliile de implementare se pot schimba ulterior, dar să se păstreze avantajele care decurg din abordarea RESTful.

REST a fost descris în mod formal de Roy J. Fielding în teza sa de doctorat. El pleacă de la un sistem care nu are delimitări clare între componente, și aplică cinci constrângeri obligatorii și una opțională asupra elementelor care compun arhitectura, precum client-server, fără stare, cache, interfață uniforma, sistem-stratificat și opțional cod-la -cerere.

Deși nu constituie o constrângere în sine, mecanismele de comunicare oferite de HTTP sunt alegerea celor mai mulți dezvoltatori care implementează REST.

2. MODUL DE FUNCȚIONARE REST WEB SERVICES



Această diagramă arată modul în care un browser preia paginile web. În mod normal, după ce un utilizator solicită o anumită resursă într-un browser, serverul web răspunde cu HTML-ul corespunzător pentru a afișa pagina către browser-ul client. Totodată, HTTP folosește operațiile CRUD (Create, Read, Update, Delete) pentru cereri de date. Mai sus, browser-ul trimite o cerere GET pentru a citi pagina web asociată, iar serverul web returnează datele asociate și un răspuns HTTP la browser-ul client.

Autentificarea este utilizată pentru a controla accesul și drepturile de access la API-urile REST. De exemplu, unii utilizatori ar putea avea access doar la citire, ceea ce înseamnă că pot utiliza doar părțile din API care citesc date.

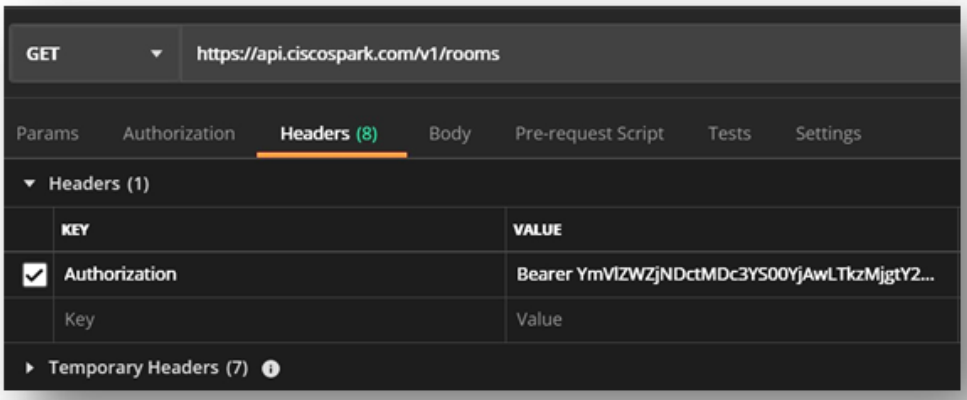
Alți utilizatori ar putea avea acces la citire și scriere. Acest lucru înseamnă că pot utiliza API-ul pentru a efectua operațiuni care nu doar citesc date, ci și adaugă, editează și

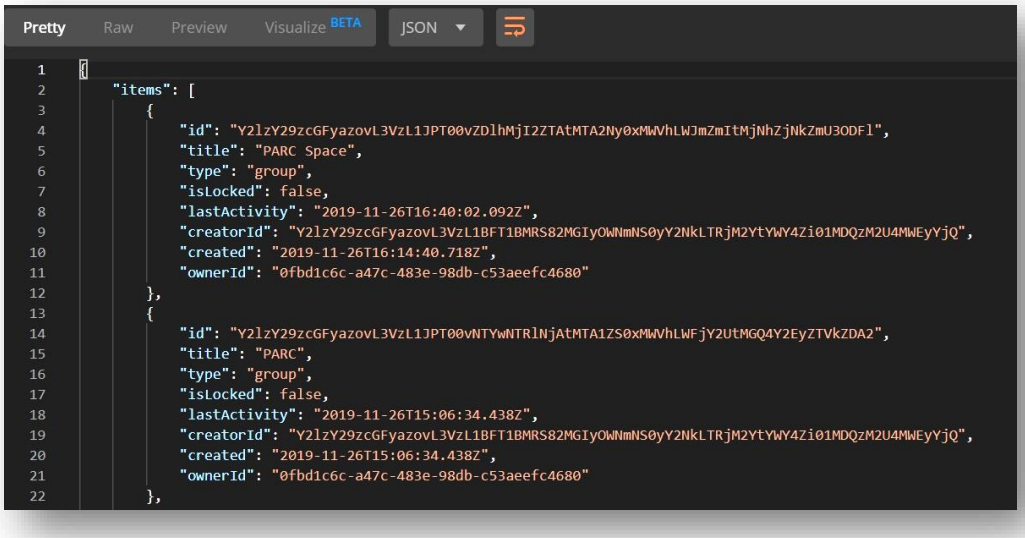
șterg date. Aceste drepturi de acces se bazează, de regulă, pe roluri atribuite de utilizator, cum ar fi rolul de ADMINISTRATOR, unde un utilizator are drepturi de a schimba datele. De exemplu, un rol de utilizator simplu poate avea drepturi de acces numai în citire.

3. Utilizarea lui Postman pentru a apela API-ul Webex Teams

Postman oferă o interfață ușor de utilizat pentru învățarea și interacțiunea cu API-urile REST. Utilizatorii pot trimite apeluri API și primi răspunsuri în aceeași fereastră. Această aplicație poate fi folosită și pentru a genera cod nativ pentru diferite limbaje de programare, cum ar fi Python.

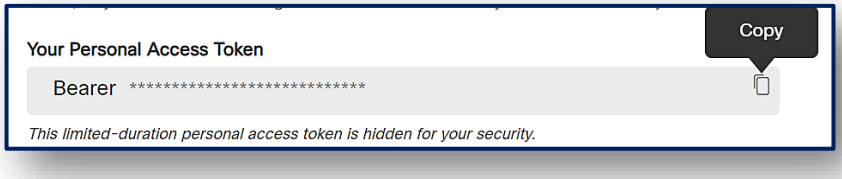
API-ul Webex Teams necesită un Token de acces pentru a efectua apeluri funcționale. Din acest motiv, am adăugat cheia denumită Authorization în lista anteturilor. Am introdus valoarea Bearer {access_token}. Tasta Content-Type specifică ce tip de conținut formatat este trimis serverului HTTP. În acest caz, nu este trimis un conținut, fiind doar o solicitare GET (numai în citire). Rezultatul constă în lista camerelor create. Datele sunt în format JSON.

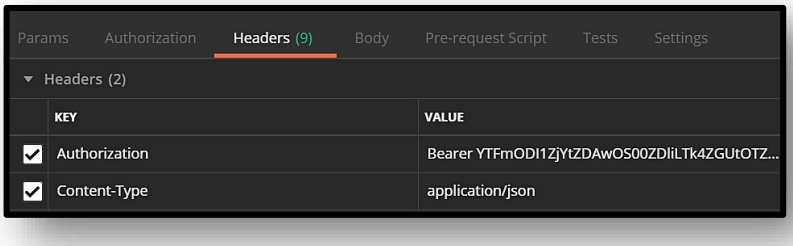




4. Webex Teams Apeluri REST API

Jetonul de access este furnizat numai în scopuri de testare, nu se folosește niciodată în producție. Expiră în 12 ore după ce a fost generat, sau după deconectarea din Webex Developers. Pentru a genera codul, trebuie obținută o listă de camere asociate cu utilizatorul care face solicitarea. Prin urmare, este folosită metoda GET, iar pentru URL adresa https://api.ciscospark.com/v1/rooms.





5. COLLECTIONS

În acest proiect am realizat un grup de solicitări GET, POST și DEL, pentru a arăta modul în care sunt create apelurile REST API. Metoda HTTP GET este folosită pentru a solicita date dintr-o resursă specificată, iar pentru orice API HTTP GET, dacă resursa se găsește pe server, atunci trebuie să returneze codul de răspuns HTTP 200 (OK) - împreună cu corpul de răspuns, care este de obicei conținut XML sau JSON. În cazul acestei metode, cererile nu vor fi niciodată stocate în cache sau în instoricul browserului, neavând restricții privind lungimea datelor.. Metodele HTTP DEL, sunt utilizate pentru a șterge resurse. Un răspuns cu succes al solicitărilor DELETE ar trebui să fie codul de răspuns HTTP 200 (OK) dacă răspunsul include o entitate care descrie starea, 202 (acceptat) dacă acțiunea a fost pusă în coadă, sau 204 (fără conținut) dacă acțiunea a fost efectuată, dar răspunsul nu include o entitate.

6. CONCLUSIONS

În concluzie, REST este un tip de transfer de date care se bazează pe arhitectura protocolului HTTP și care permite transmisia și recuperarea cu ușurință a datelor dintre două servicii diferite folosind XML sau JSON.

Pentru o aplicație care necesită un front - end Javascript sau integrarea cu o aplicație pentru smartphone, utilizarea unei arhitecturi REST devine o necesitate, deoarece permite transferul de date de la API la client. Adesea este o practică bună de a planifica colecțiile și resursele cele mai bune de la bun început..