Rest APIs

API

Application Programming Interface

API היא דרך של תוכנה מסויימת, לאפשר ל**תוכנה** אחרת להשתמש בה. הכוונה היא כזו: נהוג לחשוב על תוכנה כמשהו שמשתמש אנושי משתמש בו. API מאפשר לתוכנה להשתמש בתוכנה אחרת כאילו הייתה משתמש אנושי.

חשוב לדעת: כמעט כל אפליקציה ראויה, משתמשת ב-API של עצמה גם באופן פנימי.

למה הכוונה?

גיטהאב למשל, מאפשרת ליצור רפוזיטורי חדש בשתי דרכים:

1. דרך טופס באתר (נועד למשתמש קצה)
2. דרך קריאת api (נועד לתוכנה)

שימו לב, שבפועל, הטופס באתר, ישתמש ב-API כדי לעבוד.

בשאלה מה קדם למה, ממשק משתמש או API, התשובה היא בד"כ, API.

כמעט לכל מותג, חברה, מוצר אינטרנט ידוע בדורנו יש API:

1. פייסבוק - יש (נקרא גרף)
2. גוגל - יש
3. Ynet – יש (RSS)
4. אינסטגרם
5. טוויטר
6. ולכולם....
7. יותר מזה, יש לא מעט סטארטאפים שכל מה שהם מציעים זה API. למשל twilio

REST API

פעם, כל API היה ממציא שפב וסגנון משלו.

בוא נגיד שאני בונה את גיטהאב ובונה את ה-API שלו

אני צריך להמציא רשימת url-ים אליהם יוכלו הצרכנים שלי לפנות. אז אני ממציא:

Api.github.com/new-repository

Api.github.com/new-comment

Api.github.com/create-commit

...

...

...

עם הזמן, ועם התפתחות התעשייה, נהיה סלט.

כל חברה המציאה את הסגנון והסטנדרטים שלה.

כצרכן API היה קשה מאד לעבור בין API שונים וכל אחד היה צריך ללמוד מחדש.

REST ה

תפתח מתוך התעשייה כפתרון.

REST – representational state transfer

מאד מפתיע לגלות, אבל REST על אף הפופולריות שלו, אינו מושג שמוגדר כהלכה.

אבל מה REST נותן:

* לכל API יהיו מספר endpoint. מקומות בהם יש אפשרות לייצר מגע עם ה-API. כל endpoint מייצג יישות אחרת ב-API. מה זה ישות? למשל רפוזיטורי בגיטהאב. למשל תגובה בפייסבוק. למשל לייק באינסטגרם.
* לכל endpoint ניתן לפנות במספר פקודות http.   
  GET, POST, DELETE, PUT, PATCH. זוכרים את מושג ה-CRUD? , create retrieve update delete. אלה הן 4 הפעולות שניתן לעשות על נתונים. מסתבר שיש מתאם בין פקודות http לבין crud

|  |  |
| --- | --- |
| * CRUD | * REST |
| * Create | * POST |
| * Retrieve | * GET |
| * Update | * PUT, PATCH |
| * Delete | * Delete |

מה היה פעם?

Api.github.com/delete-repository POST

Api.github.com/create-repository POST  
api.github.com/update-repository POST

Api.github.com/get-repository GET

מה יש במקום ב-REST?

יש רק resource אחד שמיוצג על ידי האנדפוינט, למשל api.github.com/repository ואנחנו יכולים לקרוא לו באמצעות 5 פקודות http שונות.

* כל הפעלה של פקודת rest (למעט get) עושה שינוי בנתונים. אחרי כל פעולה, ה-API יחזיר את תצוגת ה-resource העדכנית.

בדרך כלל אפשר יהיה לפנות לרשימת משאבים, למשל:

Api.github.com/repositories – תביא לי את כל הרפוזיטורים

או, לפנות למשאב אחד מתוכם

Api.github.com/repositories/{id}

למשל

Api.github.com/repositories/23 – תביא לי את הרפוזיטורי שהאיידי שלו 23

מה ההבדל בין Patch ל – Put

שניהם נועדו לעדכון משאב קיים.

Put מעביר רשומה שלמה, ומחליף אותה.

Patch מעביר שדה אחד או יותר

נניח שזה מייצג רפוזיטורי:

{

Id: 23,

Name: 'repo',

Desc: 'a sample repo',

}

אם נשתמש ב-PUT כדי לעדכן את האובייקט, נצטרך להעביר מידע שלם. למשל:

{

Id: 23,

Name: 'repo',

Desc: 'a sample repo blah blah',

}

אם נרצה לעדכן רק שדה אחד מהמשאב, נשתמש ב-patch

{

Name: 'repo2'

}