Representational State Transfer

Nem egy specifikáció, inkább egy tervezési minta. A cél a standardizálás.

# Megkötések

1. Kliens-szerver architektúra: a kliens csak az UI-ért felel, a szerver pedig csak az adattárolást végzi.
2. Állapotmentesség: két kérés között a szerver nem tárol el állapotot. A kliens maga felel az állapotáért, minden egyes kérés pedig önmagában is értelmezhető és teljes, független a többi kéréstől.
3. Cachelhetőség: minden REST válaszról egyértelműen ki kell derüljön, hogy cachelhető-e, vagy sem.
4. Rétegelt architektúra: a kliens nem tudja, hogy a REST kérést mi szolgálja ki. Maga a szerver? Proxy? Mi fut a szerveren? Lényegtelen.
5. Code on demand: a szerver küldhet vissza pl. JS kódot feldolgozásra. Nem gyakori.
6. Egységes interface
   1. Erőforrás megjelölése a kérésben: a kérésben legyen megjelölve, hogy milyen választ várunk a szervertől.
   2. Erőforrás manipuláció: a kliens azt kezd a kapott válasszal, amit jónak lát. Megfelelő jogosultsággal ezt a szervernek is visszaadhatja, a szerver pedig le fogja tárolni az állapot megváltozását.
   3. Self descriptive message: ha pl. JSON választ kapunk, a szerver a http válasz fejlécében köteles jelezni.
   4. Hypermedia as the engine of application: elviekben a szerver válasza kiterjed arra, hogy a klines mi mindent tud csinálni a szerveren (milyen módszerekkel, milyen útvonalakat ér el pl.)[[1]](#footnote-1)

**RESTful API**: http-n keresztül elérhető REST API.

# ReqRes.in

Ez egy jó példa a REST API-k működésére, hogy mit is kellene implementálnom, illetve a dokumentálásra is, amire ugye szintén szükségem lesz.

# REST response

Head request csak ezt tölti le. Alapvetően sima http header. Lényeges, hogy a Content-Type legyen kitöltve, ezt a szabvány megköveteli. Amivel a kliensnek szintén mindenképpen foglalkoznia kell, az a válaszban küldött státusz.

A státuszokkal is nagyon szépen el lehet szöszölni. Ami új, az a 405 lehet: a kliens olyan method-ot használ, amit mi nem támogatunk. Ezt is érdemes visszajelezni. 5xx-et akkor küldhetünk, ha pl. az adatbázist nem érjük el.

# Authorization

Legelterjedtebb az authorization header használata. De mint tanultam, sütikkel is dolgozhatunk. JWT, OAuth2, stb.

Megj.: DELETE reqest-tel nem kell törölnünk. A mi dolgunk, mit csinálunk vele. Pl. csak a státuszt állítjuk át nem látható-ra. (Ennek azért még utána kell nézni.)

1. LinkedIn Learning: Learning REST APIs – Request – Discovery vodeóban van egy példa. Nice to have, de ne ez legyen az elsődleges fókusz. [↑](#footnote-ref-1)