**REST**

**Ez a lap egy ellenőrzött változatarészletek megjelenítése/elrejtése**

A Wikipédiából, a szabad enciklopédiából

A **REST** (**Representational State Transfer**) egy szoftverarchitektúra típus [elosztott](https://hu.wikipedia.org/wiki/Elosztott_sz%C3%A1m%C3%ADt%C3%A1sok) [hipermédia](https://hu.wikipedia.org/wiki/Hiperm%C3%A9dia) rendszerek számára, amilyen például a [világháló](https://hu.wikipedia.org/wiki/Vil%C3%A1gh%C3%A1l%C3%B3). A *Representational State Transfer* kifejezést Roy Fielding vezette be és definiálta 2000-ben a doktori disszertációjában.[[1]](https://hu.wikipedia.org/wiki/REST#cite_note-Fielding-Ch5-1)[[2]](https://hu.wikipedia.org/wiki/REST#cite_note-2) Fielding egyike a [HTTP](https://hu.wikipedia.org/wiki/HTTP) (HyperText Transfer Protocol) specifikáció szerkesztőinek.[[3]](https://hu.wikipedia.org/wiki/REST#cite_note-3)[[4]](https://hu.wikipedia.org/wiki/REST#cite_note-4)

Azokat a rendszereket, amelyek eleget tesznek a REST megszorításainak, "RESTful"-nak nevezik.[[5]](https://hu.wikipedia.org/wiki/REST#cite_note-5)

**Tartalomjegyzék**

* [1 Történet](https://hu.wikipedia.org/wiki/REST#T.C3.B6rt.C3.A9net)
* [2 Koncepció](https://hu.wikipedia.org/wiki/REST#Koncepci.C3.B3)
  + [2.1 A HTTP példája](https://hu.wikipedia.org/wiki/REST#A_HTTP_p.C3.A9ld.C3.A1ja)
  + [2.2 A SOAP példája](https://hu.wikipedia.org/wiki/REST#A_SOAP_p.C3.A9ld.C3.A1ja)
* [3 Megszorítások](https://hu.wikipedia.org/wiki/REST#Megszor.C3.ADt.C3.A1sok)
* [4 Az interfész vezérelvei](https://hu.wikipedia.org/wiki/REST#Az_interf.C3.A9sz_vez.C3.A9relvei)
* [5 Források](https://hu.wikipedia.org/wiki/REST#Forr.C3.A1sok)
* [6 Lásd még](https://hu.wikipedia.org/wiki/REST#L.C3.A1sd_m.C3.A9g)
* [7 Külső referencia](https://hu.wikipedia.org/wiki/REST#K.C3.BCls.C5.91_referencia)
* [8 Külső hivatkozások](https://hu.wikipedia.org/wiki/REST#K.C3.BCls.C5.91_hivatkoz.C3.A1sok)

**Történet**

A REST architektúra típust párhuzamosan fejlesztették a [HTTP](https://hu.wikipedia.org/wiki/HTTP) specifikáció 1.1-es változatával, a már meglévő HTTP 1.0-s specifikáció dizájnjára alapozva.[[6]](https://hu.wikipedia.org/wiki/REST#cite_note-6) A legnagyobb olyan rendszer, amely eleget tesz a REST szoftverarchitektúra típus követelményeinek a [világháló](https://hu.wikipedia.org/wiki/Vil%C3%A1gh%C3%A1l%C3%B3). A REST szemlélteti a világháló architektúráját azzal, hogy leírja és megköti a világháló négy komponensének ([kiszolgálók](https://hu.wikipedia.org/wiki/Kiszolg%C3%A1l%C3%B3), átjárók, [proxyk](https://hu.wikipedia.org/wiki/Proxy) és [kliensek](https://hu.wikipedia.org/wiki/Kliens)) magas szintű kölcsönhatásait.

**Koncepció**

Egy REST típusú architektúra kliensekből és szerverekből áll. A kliensek kéréseket indítanak a szerverek felé; a szerverek kéréseket dolgoznak fel és a megfelelő választ küldik vissza. A kérések és a válaszok erőforrás-reprezentációk szállítása köré épülnek. Az erőforrás lényegében bármilyen koherens és értelmesen címezhető koncepció lehet. Egy erőforrás-reprezentáció általában egy dokumentum, mely rögzíti az erőforrás jelenlegi vagy kívánt állapotát.

Bármely adott pillanatban egy kliens vagy állapotok közötti átmenetben van, vagy "nyugalmi" állapotban. A nyugalmi állapotban lévő kliens képes interakcióra a felhasználójával, de nem hoz létre terhelést és nem fogyaszt tárolót a szervereken vagy a hálózaton.

Ha a kliens készen áll az átmenetre egy új állapotba, akkor elkezdi küldeni a kéréseit a szerverekhez. Míg legalább egy olyan kérés van, amelyre nem érkezett válasz, a kliens átmeneti állapotban marad. Egyes erőforrás-reprezentációk hivatkozásokat tartalmaznak további erőforrásokra, amelyeket a kliens felhasználhat új állapotba történő átmenetkor.[[7]](https://hu.wikipedia.org/wiki/REST#cite_note-7)

A REST eredetileg a [HTTP](https://hu.wikipedia.org/wiki/HTTP) keretein belül lett leírva, de nem korlátozódik erre a protokollra. Egy "RESTful" architektúra más [alkalmazási rétegbeli](https://hu.wikipedia.org/wiki/Alkalmaz%C3%A1si_r%C3%A9teg) protokollra is épülhet, amennyiben az már rendelkezik értelmes erőforrás-reprezentáció átvitelhez szükséges gazdag és egységes szókinccsel. A "RESTful" alkalmazások maximálisan kihasználják a választott hálózati protokoll már létező, jól kialakított interfészeit és egyéb beépített képességeit, és minimalizálják új alkalmazás-specifikus jellemzők bevezetését.

**A HTTP példája**

A HTTP nagyon gazdag szókinccsel rendelkezik igék (vagy "metódusok"), [URI](https://hu.wikipedia.org/wiki/URI)-k, média típusok, kérés- és feleletkódok stb. szempontjából. Egy REST alkalmazás a HTTP protokoll meglévő tulajdonságait használja és így lehetővé teszi a proxyknak és az átjáróknak, hogy együttműködjenek az alkalmazással (például gyorsítótárazás vagy biztonsági funkciók formájában).

**A SOAP példája**

A fentiekkel szemben a [SOAP](https://hu.wikipedia.org/wiki/SOAP) [RPC](https://hu.wikipedia.org/wiki/Remote_procedure_call) protokoll arra ösztönzi az alkalmazásfejlesztőket, hogy új és önkényes főnév és ige szókincset definiáljanak (például getUsers(), savePurchaseOrder(...)) és csak a POST HTTP-igét használják. Emiatt sok létező HTTP képesség nincs kihasználva (pl. gyorsítótárazás, autentikáció).[[8]](https://hu.wikipedia.org/wiki/REST#cite_note-Scribner_and_Seely-8)

**Megszorítások**

Egy REST architektúra a következő hat megszorításnak kell megfelelnie, miközben az egyes komponensek implementációit hagyja szabadon tervezni:

[Kliens-szerver architektúra](https://hu.wikipedia.org/wiki/Kliens-szerver_architekt%C3%BAra)

A kliensek el vannak különítve a szerverektől egy egységes interfész által. Az érdekeltségek ilyen nemű szétválasztása azt jelenti, például, hogy a kliensek nem foglalkoznak adattárolással, ami a szerver belső ügye marad, és így a kliens kód hordozhatósága megnő. A szerverek nem foglalkoznak a felhasználói felülettel vagy a kliens állapotával, így a szerverek egyszerűbbek és még skálázhatóbbak lehetnek. A szerverek és kliensek áthelyezhetőek és fejleszthetőek külön-külön is, egészen addig amíg az interfész nem változik meg.

Állapotmentesség

A kliens-szerver kommunikáció tovább korlátozott az által, hogy a szerveren nem tárolják a kliens állapotát a kérések között. Minden egyes kérés bármelyik klienstől tartalmazza az összes szükséges információt a kérés kiszolgálásához, és minden állapotot a kliens tárol. A szerver lehet állapottartó; ez a korlátozás csupán azt követeli meg, hogy a szerver oldali erőforrás-állapotok URL által címezhetőek legyenek. Ez nem csak a szerver felügyeletét teszi lehetővé, de megbízhatóbbá teszi őket a hálózati meghibásodásokkal szemben, valamit tovább fokozza a skálázhatóságot.

Gyorsítótárazhatóság

Mint ahogy a világhálón, a kliensek itt is képesek gyorsító-tárazni a válaszokat. A válaszoknak ezért impliciten vagy expliciten tartalmazniuk kell, hogy gyorsítótárazhatóak-e vagy sem. Így elkerülhető, hogy a kliens téves vagy elavult adatokat használjon fel újra. Egy jól menedzselt gyorsítótár lehetővé teszi, hogy teljesen megkerüljünk egyes kliens-szerver interakciókat, továbbá megnöveli a rendszer skálázhatóságát és a teljesítményét.

Réteges felépítés

Egy kliens általában nem tudja megmondani, hogy direkt csatlakozott-e a végpont szerverhez, vagy közvetítő segítségével. A közvetítő szerverek megnövelhetik a rendszer skálázhatóságát terheléseloszlás kiegyenlítéssel és megosztott gyorsítótárak használatával.

[Igényelt kód](https://hu.wikipedia.org/wiki/Ig%C3%A9nyelt_k%C3%B3d) (opcionális)

A szerverek képesek időlegesen kiterjeszteni vagy testre szabni egy kliens funkcionalitását, programrészek átadásával, amelyeket a kliens futtatni képes. Ide tartoznak az előre fordított komponensek (pl. Java appletek) és a kliensoldali szkriptek (pl. [JavaScript](https://hu.wikipedia.org/wiki/JavaScript)).

Egységes interfész

Az egységes interfész kliens és szerver között egyszerűsíti és kettéválasztja az architektúrát, és lehetővé teszi, hogy egymástól függetlenül fejlődjenek az egyes részek. Az interfész négy irányadó elve alább kerül részletezésre.

A REST architektúra egyetlen opcionális megszorítása az igényelt kód. Ha egy szolgáltatás sért bármely más megszorítást, azt nem lehet feltétlenül "RESTful"-nak nevezni.

Ha az architektúra teljesíti ezeket a korlátozásokat, akkor rendelkezni fog az elosztott hipermédia rendszerek kiemelkedő tulajdonságaival, mint például a teljesítmény, skálázhatóság, egyszerűség, módosíthatóság, láthatóság, hordozhatóság és megbízhatóság.

**Az interfész vezérelvei**

Az egységes interfész, melyet minden REST interfésznek biztosítania kell, alapvetőnek tekinthető minden REST szolgáltatás tervezésekor.[[8]](https://hu.wikipedia.org/wiki/REST#cite_note-Scribner_and_Seely-8)

Erőforrások azonosítása

Egyéni erőforrások azonosítása a kérésekben történik, például [URI](https://hu.wikipedia.org/wiki/URI)-k használatával HTTP-alapú REST rendszereknél. A források maguk koncepcionálisan elkülönítettek a reprezentációktól, melyeket a kliens kap. Például a szerver nem küldi el az adatbázisát, hanem néhány [HTML](https://hu.wikipedia.org/wiki/HTML), [XML](https://hu.wikipedia.org/wiki/XML) vagy [JSON](https://hu.wikipedia.org/wiki/JSON) dokumentumot, melyek az adatbázis néhány rekordját reprezentálják, [UTF-8](https://hu.wikipedia.org/wiki/UTF-8)-ban kódolva, a kérés adataitól és a szerver implementációjától függően.

Erőforrások manipulációja ezeken a reprezentációkon keresztül

Ha egy kliens rendelkezik egy erőforrás-reprezentációval, beleértve minden csatolt [metaadatot](https://hu.wikipedia.org/wiki/Metaadat), akkor elegendő információja van az erőforrás módosításához vagy törléséhez a szerverről, feltéve, ha van engedélye hozzá.

Önleíró üzenetek

Minden egyes üzenet elegendő információt tartalmaz az üzenet feldolgozásához. Például a média típusát, hogy a kliens tudja, hogyan jelenítse meg az erőforrást.[[1]](https://hu.wikipedia.org/wiki/REST#cite_note-Fielding-Ch5-1)

Hipermédia, mint az alkalmazásállapot motorja

A kliensek csakis azokon az állapotokon mehetnek át, amelyeket a szerver által küldött hipermédia tartalmaz hivatkozások alakjában. Pár egyszerű belépési pont kivételével a kliens nem feltételezi egyik művelet meglétét sem.