Lokális Energiamenedzsment Platform szoftver minimális műszaki követelményeinek leírása

valamint a Közintézményi Energiamenedzsment Platform interfész specifikációja

Készült: 2024.12.20.

Tartalomjegyzék

1 Bevezetés

A FEAK Független Energetikai Adatközpont Zrt. egy olyan rendszert fejleszt és üzemeltet, amely a közszférába tartozó épületek, intézmények energiafogyasztási, termelési és tárolási adatait országosan gyűjti, és a gyűjtött adatokon alapulva adatelemzési szolgáltatásokat nyújt a felhasználók számára. Ezen kívül a rendszer képes lesz az épületeket üzemeltető, kezelő intézmények számára energiamenedzsment szolgáltatásokat nyújtani, hogy az energiafogyasztásukat és termelésüket optimalizálhassák, az energiavásárlási és értékesítési potenciáljaikat jobban kihasználhassák. A rendszer neve Közintézményi Energiamenedzsment Platform (KEP). A fejlesztés Magyarország Kormánya és az Európai Unió támogatásával, a DIMOP_Plusz-2.1.3-23-2023-00001 konstrukció keretében zajlik.

A FEAK célja, hogy a közintézmények tulajdonában, használatában, fenntartásában, üzemeltetésében álló épületek (a továbbiakban: épületek) energiahasználattal kapcsolatos adatai, valamint a kapcsolódó energiafogyasztási és -termelési adatok rendszeres, automatizált, mért adatokon alapuló adatszolgáltatás keretében minél szélesebb körben kerüljenek be a fejlesztés alatt álló és a FEAK által üzemeltetendő Közintézményi Energiamenedzsment Platformba. A közintézmények alatt az energiahatékonyságról szóló 2015. Évi LVII. Törvényben meghatározott fogalmat kell érteni. A beérkezett adatokon alapulva a KEP automatikus és adatbányászaton alapuló adatelemzési szolgáltatásokat végez az épületeket használó, üzemeltető intézményi, fenntartói, szakpolitikai, energetikai szakpolitikai célcsoportok számára döntéselőkészítési célokkal.

A KEP rendszerhez kapcsolódó jelen interfész specifikáció részletesen leírja az API-kat, azok működését, a követelményeket, az adatstruktúrákat, valamint a kommunikáció pontos módját. Ezen felül a folyamatos KEP-LEMP kapcsolat miatt az érintett LEMP-ek tekintetében felsoroljuk a legfontosabb funkcionális és nem funkcionális követelményeket, amit a szoftvernek teljesítenie kell.

2 Fogalmak és rövidítések

Fogalom	Magyarázat
KEP	Közintézményi Energiamenedzsment Platform, a FEAK Zrt. által a DIMOP Plusz-2.1.3-23-2023-00001 projektben, Magyarország Kormánya és az Európai Unió finanszírozásában fejlesztett, helyi és központi képességekkel rendelkező energiamenedzsment szoftver és adatelemző platform.
LEMP	Lokális Energiamenedzsment Platform – azon piaci szoftvernek a projektben történő megnevezése, mely a közintézményekben kerül telepítésre, és aminek van saját adatbázisa, üzleti logikája és felhasználói felülete. Decentralizált LEMP alatt olyan kiépítést értünk, amikor az épület szintű lokális energiamenedzsment funkciókat valamely piaci LEMP szoftver látja el, és ez a szoftver szolgáltat adatokat a KEP felé. Ebben a dokumentumban a LEMP szó vagy a „decentralizált LEMP” szókapcsolat ezt a kiépítést takarja.
REST	A REST olyan irányelvek összessége, amelyek segítségével a szoftverek az interneten keresztül kommunikálhatnak, hogy az integrációkat egyszerűvé és méretezhetővé tegyék. A REST API (más néven „RESTful” API) az API egy speciális típusa, amely követi ezeket az irányelveket. A REST a Representational State Transfer rövidítése. Ez azt jelenti, hogy amikor egy kliens erőforrást kér a REST API használatával, a kiszolgáló szabványos megjelenítésben adja vissza az erőforrás aktuális állapotát. A REST az integrált fejlesztési környezetben elterjedt kommunikációs architektúra. Ez egy felhasználóbarát interfész-architektúra, amely egy általánosan ismert internet-protokollt használ. Az adatátvitel a Hyper Text Transfer Protocol (HTTP) használatával történik.
HTTPS	A https egy URI-séma, amely biztonságos http kapcsolatot jelöl. Szintaktikailag megegyezik a http sémával, amelyet a HTTP protokollnál használnak, de a https nem önálló protokoll, hanem csak egy URI séma, mely azt jelzi, hogy a HTTP protokollt kell használni a szerver 443-as TCP portján a HTTP és a TCP szintek közé titkosító/autentikáló SSL vagy TLS réteg beiktatásával. (A titkosítatlan HTTP rendszerint a 80-as TCP portot használja.)

Fogalom	Magyarázat
API	Az alkalmazásprogramozási interfész (API) egy számítógépek vagy számítógépes programok közötti kapcsolat. Egyfajta szoftver interfész, amely szolgáltatást nyújt más szoftverek számára. Az ilyen kapcsolatok vagy interfészek létrehozásának módját leíró dokumentumot vagy szabványt API-specifikációnak nevezzük. Egy olyan számítógépes rendszerről, amely megfelel ennek a szabványnak, azt mondjuk, hogy API-t valósít meg vagy tesz közzé. Az API kifejezés utalhat akár a specifikációra, akár a megvalósításra.
Ügyfél	Az API-t használó személy vagy program. Az ügyfél kéréseket intéz az API-hoz bizonyos információk lekérése vagy az alkalmazáson belüli módosítások érdekében. Az Ön webböngészője egy kliens – interakcióba lép a különböző webhelyekkel rendelkező API-kkal, hogy oldaltartalmat kapjon tőlük. A kért adatok visszaküldésre kerülnek a böngészőjébe, és megjelennek a képernyőn.
Erőforrás	Minden olyan információ, amelyet az API tud biztosítani az ügyfél számára. Például a Facebook API-jában egy erőforrás lehet egy felhasználó, egy oldal, egy fénykép vagy egy bejegyzés. Minden erőforrásnak egyedi neve van, amelyet erőforrásazonosítónak neveznek.
Kiszolgáló	Az az alkalmazás használ, amely fogadja az ügyfél kéréseit, és tartalmazza az ügyfél által igényelt erőforrásokat. A szerver rendelkezik egy API-val, amellyel interakcióba léphet az ügyfelekkel anélkül, hogy közvetlen hozzáférést biztosítana számukra az adatbázisában tárolt tartalomhoz.
JSON	A JSON (JavaScript Object Notation) egy könnyű adatcsere formátum, amelyet az adatstruktúrák és objektumok szöveges ábrázolására használnak. A JSON egyszerű, ember által olvasható, és széles körben támogatott adatcsere formátum, amelyet gyakran használnak webalkalmazásokban az adatok szerver és kliens közötti átvitelére.
XML (Extensible Markup Language)	Az XML egy általános célú, szabványosított jelölőnyelv, amelyet adatok tárolására és szállítására használnak. Strukturált, ember által olvasható és géppel feldolgozható formátumot biztosít. Jellemzően olyan alkalmazásokban használják, ahol az adatok cseréje különböző rendszerek között fontos.
SOAP (Simple Object Access Protocol)	A SOAP egy protokoll, amely az XML-t használja az adatok strukturált csomagokban történő továbbítására. Az egyszerű objektum-hozzáférési protokoll segítségével webszolgáltatások közötti kommunikáció valósul meg.

Fogalom	Magyarázat
WSDL (Web Services Description Language)	A WSDL egy XML-alapú szabvány, amelyet webszolgáltatások leírására használnak. Segítségével a szolgáltatás fogyasztói (kliensek) megérthetik, hogyan kell kommunikálni egy adott webszolgáltatással.
OpenAPI	Az OpenAPI (korábban Swagger) egy szabvány, amely RESTful API-k dokumentálására szolgál. JSON vagy YAML formátumot használ, hogy leírja az API-k működését és struktúráját.
OAuth2	Azonosítási és engedélyezési protokoll, amelyet széles körben alkalmaznak webes és mobilalkalmazásokban az erőforrások biztonságos hozzáféréséhez.
HMAC (Hash-Based Message Authentication Code)	Egy kriptográfiai eljárás, amelyet üzenetek hitelesítésére és integritásának ellenőrzésére használnak.

3 Minimális funkcionális és nem funkcionális elvárások

Mivel a decentralizált LEMP folyamatos adat kapcsolatban van a KEP rendszerrel a KEP-LEMP interfészen keresztül, így vonatkozik rá a 2013.évi L. törvény szerinti 4-es osztálynak történő informatikai megfelelés, emiatt többek között a következőkben felsorolt funkcionális és nem funkcionális követelményeket szükséges teljesítenie. A besorolásról és a megvalósítandó intézkedésekről a következő linken lehet tájékozódni: https://nki.gov.hu/wp-content/uploads/2019/03/ovi460.xltm

A szoftvernek fejlett jogosultság kezelő rendszerrel kell rendelkeznie, amiben lehetséges a különböző felhasználói körök szerinti adathozzáférés biztosítása, a felhasználói körök későbbi bővítésének a lehetősége
A szoftvernek a felhasználói autentikáció tekintetében a megfelelő felhasználók esetében az autentikációs applikációval biztosított 2 faktoros bejelentkezést kell megvalósítania.
A LEMP szoftvernek titkosított adatbázisban kell tárolnia az adatokat.
A LEMP szoftvernek az fent hivatkozott törvényben meghatározott naplózási, mentési és védelmi rendszerekkel kell rendelkeznie.
A LEMP szoftver az adatok tárolására csak olyan felhő infrastruktúrát használhat, mely bizonyíthatóan csak Magyarországon belül tárolja az adatokat.
A LEMP szoftvernek a 7-es pontban felsorolt törzsadatokat kezelnie kell a KEP-LEMP interfésznek megfelelő hierarchiában.
A LEMP szoftvernek az almérők (nagyfogyasztói almérők) által mért fázisonkénti hatásos és meddő fogyasztásokat 15 perces felbontásban tárolnia kell és a KEP-LEMP interfészek keresztül át kell tudnia adni a KEP rendszernek. Azonban a saját üzleti logika indokán ugyanezen adatokat 5 perces felbontásban is tárolnia kell.
A LEMP-nek a hozzá kapcsolódó Iot szenzorok adatait olyan formában és gyakorisági felbontásban kell tárolnia, hogy a KEP-LEMP interfész elvárása szerint át tudja adni a hasznos információkat a KEP rendszer számára.

A LEMP szoftvernek rendelkeznie kell olyan bővíthető REST API modullal, ahová új IoT eszközök és szenzorok beköthetők oly módon, hogy amennyiben szükséges, az adatok a KEP-LEMP interfészen keresztül átadhatók legyenek a KEP rendszernek.
A LEMP szoftvernek rendelkeznie kell egy olyan kimeneti REST API modullal, amin keresztül a KEP-től érkező vezérlési utasításokat a hatásos beavatkozáshoz szükséges idő alatt elvégzi.
A LEMP-nek rendelkeznie kell olyan fejlett analitikai és villamos ipari menetrend tervező modullal, ami gépi tanulás segítségével az intézmény számára optimális villamos menetrendet tud előállítani mind a fogyasztás, mind a villamos energia termelés tekintetében, azonban ezen menetrendek nem írhatják felül a KEP-től érkező vezérlési utasításokat. A menetrendeket tervező algoritmusoknak figyelembe kell tudni vennie nemcsak villamos paramétereket, hanem egyéb környezeti paramétereket is, amelyek az adott menetrendet befolyásolhatják, pl. napelem esetén a besugárzási szög, vagy felhőzöttség.
A LEMP szoftvernek rendelkeznie kell egy riport készítő modullal, ahol egyedi és rendszeres riportokat tudnak készíteni a felhasználók a szoftverben lévő adatokról és azt diagrammos és táblázatos formában is le tudják menteni.
A LEMP szoftvernek rendelkeznie kell egy grafikus felülettel, ahol a rendszerben tárolt adatok tetszőleges időszakra, tetszőleges felbontással (az értelmezhetőséget is figyelembe véve) megjeleníthetők mind diagrammos, mind táblázatos formában.

4 Integrációra használt technológia

Az információ technológia változásával a fejlődés felgyorsulása, nem csak az eszközökre, hanem a megoldásokra is hatással volt. A korábban bonyolult és nehézkes, valamint lassú kommunikációs módszereket egyre inkább felváltja egy könnyen érthető, egyszerű és a weben natív megoldásnak számító megoldás.

A REST olyan irányelvek összessége, amelyek segítségével a szoftverek az interneten keresztül kommunikálhatnak, hogy az integrációkat egyszerűvé és méretezhetővé tegyék. A REST API (más néven „RESTful" API) az API egy speciális típusa, amely követi ezeket az irányelveket.

A REST a Representational State Transfer rövidítése. Ez azt jelenti, hogy amikor egy kliens erőforrást kér a REST API használatával, a kiszolgáló szabványos megjelenítésben adja vissza az erőforrás aktuális állapotát.

Tehát a REST API-k úgy működnek, hogy kéréseket küldenek egy erőforrásra, és visszaadják az erőforrásra vonatkozó összes releváns információt, olyan formátumba lefordítva, amelyet az ügyfelek könnyen értelmezhetnek (ezt a formátumot a kéréseket fogadó API határozza meg). Az ügyfelek a kiszolgálón lévő elemeket is módosíthatják, és akár új elemeket is hozzáadhatnak a szerverhez a REST API-n keresztül.

A REST szoftverarchitektúra specifikációja a https://www.w3.org/2001/sw/wiki/REST címen érhető el.

4.1 Rövid áttekintés

A 2000-es évek elején kezdett el teret hódítani az XML. Addig nem volt elterjedt egységes adatszerkezet, amelyet ember és gép egyszerűen olvashatott volna. Az XML felett megjelentek olyan rétegek (pl.: a WSDL, a SOAP, stb.) amelyek próbálták formalizálni a szoftverek egymás közötti kommunikációját. A megoldások is erre épültek, a vállalatok, kormányzatok belső rendszereikben használták.

Az internet széleskörű elterjedése azonban változást hozott, az alkalmazásoknak meg kellett nyílniuk a külvilág felé. A nyitás folyamán a belső, megszokott módszereket vitték át a webes platformra, amely gyorsan felszínre hozta a megoldás hátrányát. A korábbi tranzakció számok megnövekedtek, ezáltal a nyomon követés fontos lett a minőségi szolgáltatások biztosításában.

Az XML technológiához kapcsolódó megoldások erőforrásigényesek, nehézkesek, helyigényesek voltak. A böngészőkön natív adatszerkezet a JSON, az XML idegen számára. A mobil eszközökön sem volt mindegy, hogy 50 kbyte XML vagy 15 kbyte JSON adatot kell továbbítani, mert ennek feldolgozása több ideig tartott és több energiába került. A szervereken például a bankok és a kormányzatok naplózást végeznek, amelyek az XML esetén 2-3x több tárterületet igényelnek. A feldolgozáshoz kétszer annyi szerver és energia kell.

A REST-hez hasonló protokoll követelményt azonban semmilyen szabvány nem biztosított, így kézenfekvő volt bevezetése. Ez a döntés hatalmas lökést adott az alkalmazás integrációnak, mert az alkalmazások egymás között kommunikálni tudtak oly módon, hogy nem egy interfészre szabott protokollt használtak, hanem általánosan elterjedt, kipróbált eljárásokat, kódokat alkalmazhattak, ugyanis nincsenek SOAP üzenetkódok, de a REST API a HTTP státusz kódokat definiálta, mint használandó üzenet kód[1].

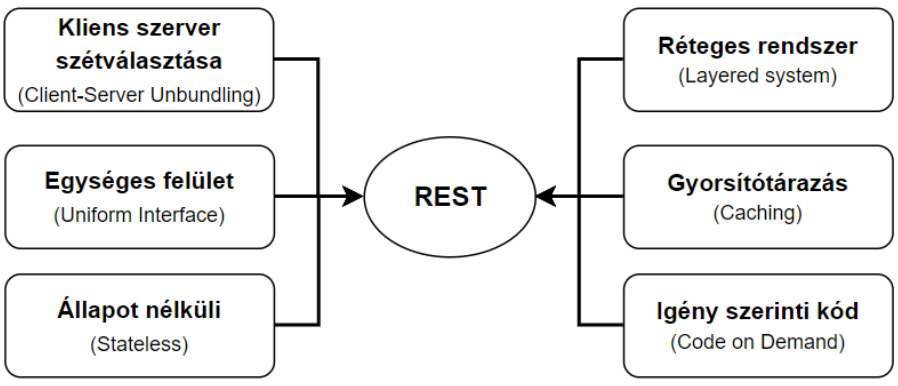
A REST keretrendszert Roy Fielding informatikus vezette be 2000-ben, és ma már ez határozza meg, hogyan tekintjük meg, módosítjuk és továbbítjuk a tartalmakat az interneten. A legnépszerűbb web- és felhőcégek közül sok REST API-t használ alkalmazásaihoz, beleértve a Facebookot, a YouTube-ot, a Twittert és a Google-t.

Alapvetően ez egy kiváló rendszer webes alkalmazásokhoz. Az ilyen típusú API fő előnyei:

- A REST API-k rugalmasak. Sokféle kérést tudnak kezelni, és sokféle formátumban küldhetnek adatokat.
- A REST API-k méretezhetők. Bármely két szoftver közötti kommunikációra készültek, mérettől vagy képességtől függetlenül. Ahogy egy webalkalmazás növekszik és több erőforrást ad hozzá, REST API-ja képes lesz gyorsan kezelni a növekvő mennyiségű és változatos kéréseket.
- A REST API-k a meglévő webtechnológiákat tartalmazzák, így viszonylag könnyen összeállíthatók és használhatók. Ha REST API-n keresztül szeretne erőforrást kérni, csak meg kell adnia annak URL-címét.

4.2 A REST API-k működési szabályai

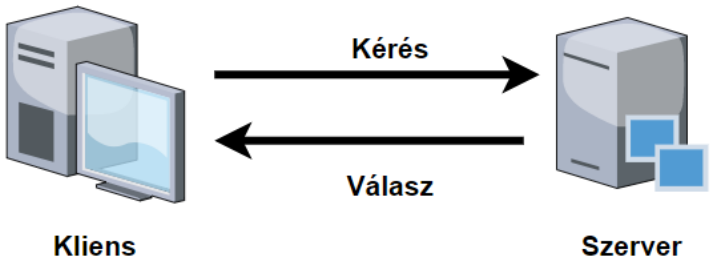
A REST által nyújtott funkciók teljes kihasználásához az API-knak hat követelménynek kell megfelelniük. Mindegyik követelmény megalapozza a gyors és sokoldalú API-t.



1. ábra: REST API korlátozások

1) Kliens-szerver szétválasztás:

REST architektúra esetén a kliens és a szerver csak egyféleképpen tud együttműködni: A kliens kérést küld a szervernek, majd a szerver választ küld vissza a kliensnek. A kiszolgálók nem tudnak kéréseket küldeni, az ügyfelek pedig nem tudnak válaszolni – minden interakciót az ügyfél kezdeményez.

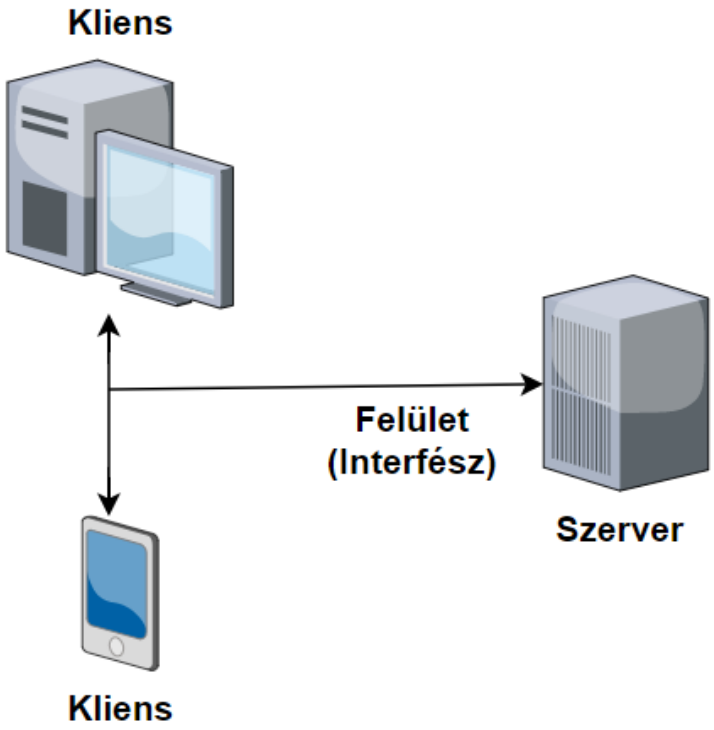


2. ábra: Kliens-szerver szétválasztásának folyamat a REST-en belül

2) Egységes felület:

Ez az irányelv kimondja, hogy minden kérésnek és minden válasznak egy közös protokollt vagy az üzenetek formázási módját kell követnie. Az alkalmazások és a szerverek különféle nyelveken készülnek, amelyek közvetítő nélkül nem működnek együtt. Az egységes felület minden kliens közös nyelve bármely REST API-val való kommunikációhoz.

Szabványosított kommunikáció nélkül a kérések és válaszok szoftverek közötti fordítása teljes káosz lenne. A kisebb eltérések az információk összezavarodását és elvesztését okoznák, és az alkalmazásoknak frissíteniük kellene kérési folyamataikat, amikor az API-k frissítik a sajátjukat. Az egységes interfész ezt a lehetőséget kiküszöböli.



3. ábra: Egységes felület felépítése REST API-n belül

A legtöbb REST API esetében ez a közös nyelv a HTTP vagy a Hyper-Text Transfer Protocol. A HTTP nem kifejezetten a REST számára készült. A REST inkább ezt a kommunikációs protokollt fogadta el szabványként az azt használó alkalmazások számára.

Mint minden REST API-hoz intézett kérés, ez a kérelem is két információt tartalmaz. A GET a HTTP módszer.

Ez határozza meg az ügyfél által az erőforráson végrehajtani kívánt műveletet. Négy alapvető HTTP-kérelem van, amelyet az ügyfél tehet:

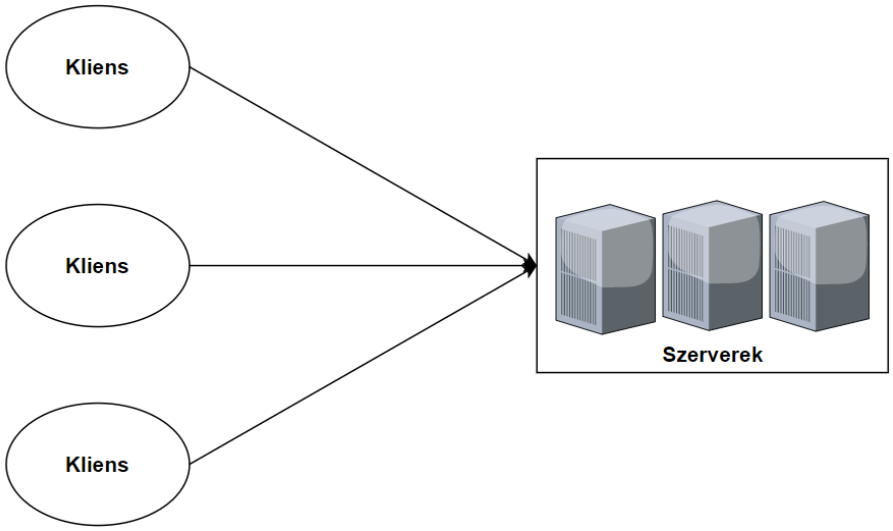
- GET: Ez a kérelem a megadott erőforrás ábrázolását kéri. A kérések csak adatokat, erőforrásokat kérhetnek le.
- POST: Új erőforrás létrehozása, amely során a metódus elküld egy entitást a megadott erőforráshoz, ami gyakran állapotváltozást okoz.
- PUT: Meglévő erőforrás szerkesztése vagy frissítése, vagyis ez a módszer lecseréli a célerőforrás összes jelenlegi reprezentációját a kérés hasznos adatára.
- TÖRLÉS: Ez a kérelem törli a megadott erőforrást.
- https://… az URL. Az URL tartalmazza az egységes erőforrás-azonosítót vagy URI-t, amely meghatározza a célerőforrást.

Ebben az esetben az URL-t végpontnak is nevezik, mert ez az a hely, ahol az API ténylegesen interakcióba lép az ügyféllel.

A kérés fogadása és érvényesítése után a gazdagép információkat ad vissza a célerőforrásról. Általában az információkat JSON nevű formátumban küldik vissza.

3) Állapot nélküli:

Minden REST API-val rendelkező hívásnak állapot nélkülinek kell lennie. Ez azt jelenti, hogy minden interakció független, és minden kérés és válasz megadja az interakció befejezéséhez szükséges összes információt. Az ügyfél minden kérését a szerver teljesen új kérésként értelmezi – a szerver nem emlékszik semmire a korábbi kérésekről.



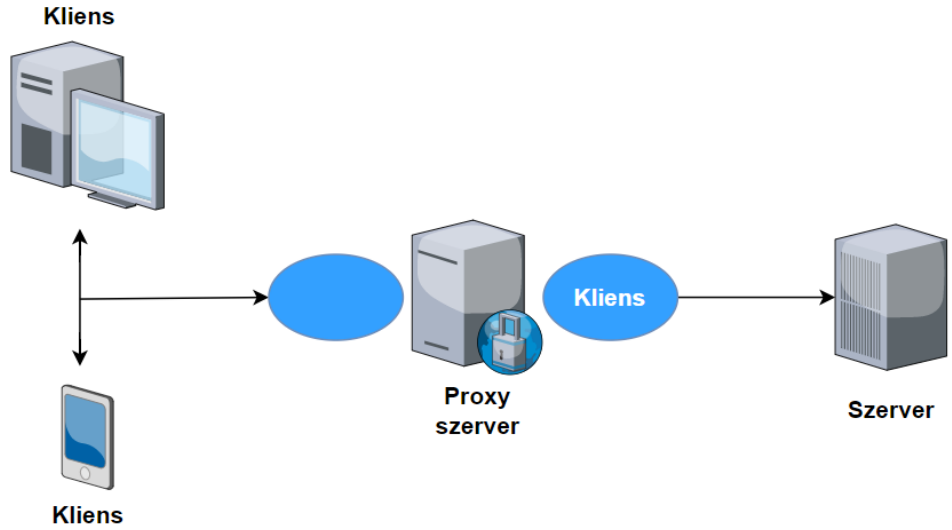
4. ábra: Állapot nélküli REST API-k működési elve

4) Réteges rendszer:

Ez az elv megköveteli, hogy a kliens és a célkiszolgáló közötti üzeneteket mindig azonos módon kell formázni és feldolgozni, függetlenül a közöttük lévő rétegektől. A további rétegek nem befolyásolhatják az ügyfél-szerver interakciókat.

A réteges architektúra miatt proxyt vagy terheléselosztót helyezhet el a kliens és a szerver között, és így javíthatja a méretezhetőséget. A biztonság külön rétegként történő hozzáadása növeli a rendszer biztonságát. Bár ezek a szolgáltatások részt vesznek a válasz létrehozásában, az ügyfélnek nem kell aggódnia amiatt, hogy mi van a felület mögött.

Ha a fejlesztők követik ezt az irányelvet, a kiszolgálórendszerek átrendezhetők, frissíthetők vagy más módon módosíthatók anélkül, hogy ez befolyásolná az alapvető kérés-választ.

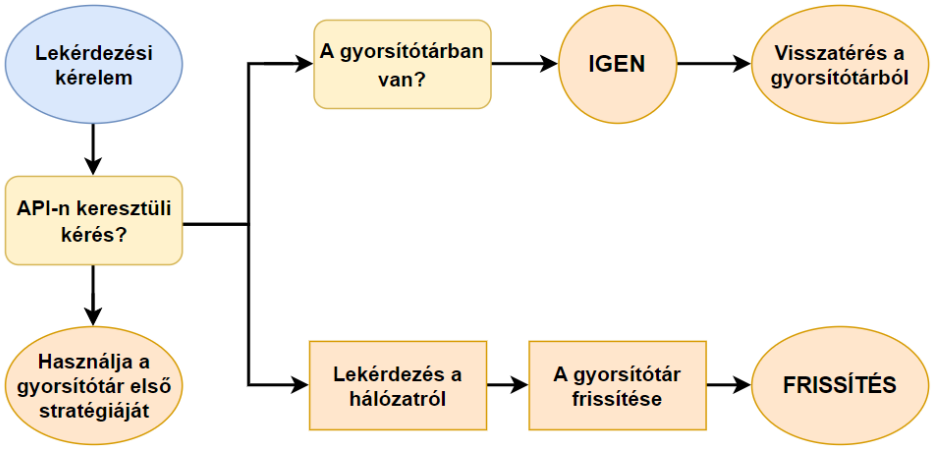


5. ábra: Réteges rendszer felépítése a REST API-n belül.

5) Gyorsítótárazható:

A REST API-k az adatok gyorsítótárazását szem előtt tartva jönnek létre. Amikor a szerver elküldi a választ egy ügyfélnek, a válasznak jeleznie kell, hogy a megadott erőforrás gyorsítótárazható-e, és mennyi ideig.

A gyorsítótárazás megtakarítja a szerver erőforrásait és a sávszélességet, miközben csökkenti az oldalbetöltési időt, ezért a legtöbb nagy webhely ezt teszi.



6. ábra: A gyorsítótárazás folyamata a REST-en belül.

6) Igény szerinti kód (Opcionális):

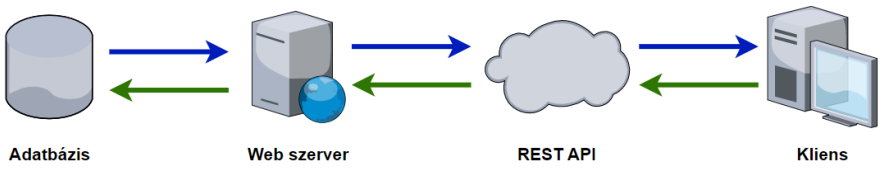
A végső REST elv nem kötelező. Kívánt esetben az API válaszában számítógépes kódot küldhet az ügyfeleknek. Ez felhatalmazza az ügyfelet arra, hogy a kódot a saját háttérprogramjában futtassa.

Mindaddig, amíg egy API betartja ezt a szabálykészletet, RESTteljesnek minősül. Ezek a szabályok azonban bőséges teret hagynak a fejlesztők számára API-juk funkcióinak testre szabására. Ez a rugalmasság különbözteti meg a REST API-kat egy másik elterjedt webes API-módszertől, a Simple Object Access Protocol-tól (SOAP).

4.3 A REST API használata

A nyilvánosan elérhető API-kkal rendelkező webes alkalmazások dokumentációja elérhető lesz a webhelyük „fejlesztők" részében. Itt talál útmutatást az API eléréséhez és használatához saját szoftverével együtt. Ha az API REST elvekkel épül fel, a dokumentáció valószínűleg ezt jelzi.

Sok API használatához API-kulcs szükséges. Az API-kulcs egy egyedi karakterlánc, amelyet az API-szolgáltató ad a fejlesztőnek, hogy engedélyezze az API-jához való hozzáférést. Az API-kulcsokat gyakran elküldik az ügyfél kéréseivel, hogy azonosítsák a klienst a kiszolgálónak.



7. ábra: REST API működése ábrán bemutatva

4.3.1 REST Standardok alkalmazása

Az alábbi REST standardok alkalmazása rendszer szinten szükséges alkalmazni:

Fejléc adatok (HEADERS):

- **Authorization** – Szükséges oAuth2 authentikáció megvalósításakor. Az engedélyezés annak ellenőrzése, hogy a csatlakozási kísérlet engedélyezett-e. Az engedélyezés a sikeres hitelesítés után történik.
- **Date** – Szükséges az oAuth2 és HMAC authentikáció típusoknál. Tartalmazza továbbá az üzenet keletkezésének dátumát és időpontját.
- **Accept** – application/json támogatott. A kérés fejléc felé jelzi, hogy a kliens mely tartalomtípusokat, MIME-típusokban „Accept" kifejezve képes megérteni. A szerver tartalomegyeztetés segítségével választja ki az egyik javaslatot, és a válaszfejlécben tájékoztatja az ügyfelet a választásról.

Metódusok (Methods):

- **GET** - Reading (Lekérdezés, olvasás): Ez a kérelem a megadott erőforrás ábrázolását kéri. A kérések csak adatokat, erőforrásokat kérhetnek le. A metódus képes lekérni a szerveren tárolt bármilyen adatot, legyen szó képekről vagy teljes HTML dokumentumokról.

- **PUT** – Updating (Frissítés): Meglévő erőforrás szerkesztése vagy frissítése, vagyis ez a módszer lecseréli a célerőforrás összes jelenlegi reprezentációját a kérés hasznos adatára. Az egyetlen dolgot, amit szem előtt kell tartani a metódussal kapcsolatban, hogy kevesebb elemet kell szerkeszteni, hanem inkább le kell cserélni.
- **POST** – Creating (Létrehozás): Új erőforrás létrehozása, amely során a metódus elküld egy entitást a megadott erőforráshoz, ami gyakran állapotváltozást okoz. Ez általában az útvonalparaméterek használatával történik, amelyek lehetővé teszik a felhasználónak, hogy a feladási kérelmet adatokkal töltsék fel.
- **DELETE** – Removing (Törlés): Ez a kérelem törli a megadott erőforrást.
- **OPTIONS** – All the options that call has (A hívás valamennyi opciójának visszaadása)

A rendszer szolgáltatásainak dokumentálása OpenAPI specifikáció (korábbi Swagger) alapján szükséges definiálni. Az OpenAPI struktúra szerint szükséges definiálni és leírni a REST Interfészeket, amelyet a rendszeren belül egy Swagger végponton szükséges regisztrálni.

A rendszer működése szempontjából elkülönítünk a rendszeren belüli és rendszeren kívüli szolgáltatásokat. A rendszeren belüli szolgáltatások esetében a modulok egymás közötti hívásait, szolgáltatásait értjük.

A külső szolgáltatások az Energiamenedzsment rendszer határain kívüli szolgáltatások hívását és kiszolgálását biztosítják.

A rendszer fejlesztés során minden REST hívás csak authentikált módon történhet, a szolgáltatások hívása naplózott esemény.

4.3.2 REST válaszstruktúra

Válaszul a szerver nem magát a keresett erőforrást küldi el, hanem annak reprezentációját – az aktuális állapot géppel olvasható leírását. Ugyanaz az erőforrás különböző formátumokban jeleníthető meg, de a legnépszerűbbek az XML és a JSON.

Ha releváns, a kiszolgáló a válaszban olyan hiperhivatkozásokat vagy hipermédiát is tartalmaz, amely más kapcsolódó erőforrásokra hivatkozik. Így a szerver utasításokat ad arra vonatkozóan, hogy mit tehet a kliens ezután, és milyen további kéréseket tehet[i].

A sikeres webhely egyik legfontosabb része az, hogy a webhely hogyan válaszol a kérésekre. Előfordulhat, hogy minden jól néz ki a böngészőben, de ha nem tudja a felhasználó, hogy az oldalak milyen válaszkódot küldenek vissza, nem számít, mennyire jól néz ki.

A REST erőforrásoktól érkező üzenetek az erőforráspéldánytól a válaszhoz kerülnek továbbításra. A HTTP-válasz állapotkódjai jelzik, hogy egy adott HTTP-kérés befejeződött-e. A válaszok öt osztályba vannak csoportosítva:

1. táblázat: REST API válaszkódok

Megnevezés	Értéktartomány
Információ	100-199
Sikeresség	200-299
Átirányítás	300-399
Ügyfél oldali hiba	400-499
Szerver oldali hiba	500-599

A válaszkódok mindig háromjegyű számok, amelyek csoportokra vannak osztva. Minden csoportnak van jelentése, az alábbiak szerint[ii]:

1xx: Tájékoztató: Kérés beérkezett, a folyamat folytatódik.

2xx: Siker: A műveletet sikeresen fogadták, megértették és elfogadták.

3xx: Átirányítás: A kérés teljesítéséhez további lépéseket kell tenni.

4xx: Felhasználó – Ügyfél hiba: A kérés rossz szintaxist tartalmaz, vagy nem teljesíthető.

5XX: Szerverhiba: A szerver nem tudott teljesíteni egy látszólag érvényes kérést.

Az interfész válaszokra az alábbi leggyakoribb válaszkódokat szükséges használni[iii]:

- **200** – OK: A kérés sikeres volt. A „siker” eredmény jelentése a HTTP-módszertől függ, ami lehet GET, PUT, POST, HEAD vagy DELETE.
- **201** – Created (Létrehozva): A kérés sikeres volt, és a kérelem alapján valami új jött létre
- **204** – No Content (Nincs tartalom): A kérés hatására a szerver sikeresen feldolgozta a kérést, de nincs mit megjeleníteni.
- **301** – Moved Permanently (Véglegesen áthelyezve): A kért dokumentum véglegesen át lett helyezve, és a válasz tartalmazza az új hely URL-jét.
- **400** - Bad Request (Hibás kérés): A kiszolgáló nem tudja vagy nem fogja feldolgozni a kérelmet valami miatt, amelyet ügyfélhibaként észlelnek
- **401** – Unauthorized (Jogosulatlan kérés): Az Ügyfelet a hozzáférés előtt engedélyeztetni kell, jellemzően valamilyen bejelentkezés útján.
- **403** – Forbidden (Tiltott hívás): Az ügyfélnek nincs engedélye a kért dokumentum elérésére. Az ügyfélnek nincs hozzáférési joga a tartalomhoz (pl.: a kért dokumentumhoz), azaz jogosulatlan, így a szerver nem hajlandó megadni a kért erőforrást.
- **404** – Not Found (Nincs találat): A kért dokumentum nem található a megadott URL-címen, és nincs megadva új hely a dokumentumhoz. Nem jelenti azt, hogy a dokumentum véglegesen hiányzik az adott URL-ről.
- **405** – Method Not Allowed (Nem engedélyezett metódus): A kérési módszer nem engedélyezett a megadott dokumentumhoz (pl.: hogy egy API nem teszi lehetővé a DELETE meghívását egy erőforrás eltávolításához).

- **406** – Not Acceptable (Nem elfogadható): A kért dokumentum nem küldhető el úgy, hogy az ügyfél megértse.
- **408** – Request Timeout (Kérés időtúllépése): A kérés időtartama meghaladta azt az időtartamot, ameddig a szerver be van állítva, hogy várjon egy kérésre.
- **409** – Conflict (Konfliktus): A kért dokumentumot a kérés ütközése miatt nem lehetett elküldeni.
- **415** – Unsopported Media Type (Nem támogatott médiatípus): Azt jelzi, hogy a kérelem legalább egy részének formátuma nem támogatott.
- **429** – Too Many Request (Túl sok kérés): A felhasználó túl sok kérést küldött egy adott időn belül („sebességkorlátozás”).
- **500** – Internal Server Error (Kiszolgálóhiba): Általános hibaüzenet, ami azt jelenti, hogy valami elromlott, de semmi konkrétabbat nem lehet küldeni.
- **502** – Bad Gateway (Rossz átjáró): Átjáróként vagy proxyként működő kiszolgáló érvénytelennek ítélt választ kapott egy upstream szervertől.
- **503** – Service Unavailable (Szolgáltatás nem elérhető): A szerver jelenleg nem érhető el nagy terhelés, karbantartás vagy egyéb átmeneti helyzet miatt.
- **504** – Gateway Timeout (Átjáró időtúllépés): Az átjáróként vagy proxyként működő kiszolgáló nem kapott választ azon időn belül, ameddig a kiszolgáló válaszvárásra van beállítva.

5 Interfész tulajdonságok

5.1 Interfész azonosító

Egyedi azonosító, amely megkülönbözteti az interfészt a rendszer többi kapcsolatától.

5.2 Interfész típusa

Az interfész működésének jellege:

- Adatküldő: Csak adatok továbbítása.
- Adatfogadó: Csak adatok fogadása.
- Kétirányú: Adatok kölcsönös továbbítása és fogadása.
- Szinkron/aszinkron: Az adatátvitel időzítésének jellemzője. Szinkron interfész esetén a folyamat azonnali választ vár, míg aszinkronnál az adatfeldolgozás később történik.

5.3 Adatok tulajdonsága

Az interfész által továbbított adatok jellemzői:

- Adattípusok (pl. szöveg, szám, bináris).
- Formátum (pl. JSON, XML, CSV).
- Adatmennyiség (pl. egyszeri rekordok vagy tömeges adatcsomagok).
- Adatok érzékenysége (pl. személyes adatok, titkosított információk).

5.3.1 Adatok forráskomponense

Az a rendszer vagy modul, amely az adatokat előállítja, kezeli vagy küldi az interfésznek. Példa: egy szenzor adatokat gyűjtő alkalmazás.

5.3.2 Adatok célkomponense

Az a rendszer vagy modul, amely az interfész által küldött adatokat fogadja és feldolgozza. Példa: központi adattároló vagy elemző rendszer.

5.3.3 Adatátadás leírása

Részletezi az adatávitel folyamatát:

- Az adatátadás mechanizmusa (pl. API hívás, fájlátvitel).
- Az adatok validációja (pl. sémaellenőrzés).
- Az átviteli protokollok (pl. HTTPS, MQTT).

5.3.4 Adatátadás gyakorisága

Meghatározza, hogy milyen időközönként történik az adatátadás:

- Valós idejű: Az adatok folyamatosan frissülnek.

- Időzített: Meghatározott időközönként (pl. napi, óránkénti).
- Eseményvezérelt: Csak bizonyos események bekövetkezésekor.

5.3.5 Adatátadás kezdeményezője

Az az entitás, amely az adatátadást elindítja.

- Forráskomponens: Az adatok küldését az adatok forrása indítja
- Célkomponens: Az adatokat a fogadó rendszer kérdezi le.

6 Kialakítandó interfészek

Az interfészek jelentősége a szoftverfejlesztésben alapvető, mivel azok biztosítják a különböző rendszerek, modulok és alkalmazások közötti hatékony és strukturált kommunikációt. Az interfészek nemcsak technikai szempontból fontosak, hanem a szoftver rugalmasságának, újrahasznosíthatóságának és karbantarthatóságának növelésében is kulcsszerepet játszanak.

6.1 Adat interfész

Az adatinterfész az alkalmazások vagy modulok közötti adatkommunikáció módját határozza meg. Ez az interfész szabályokat és struktúrákat ír elő az adatok kezelésére, cseréjére és tárolására, biztosítva, hogy a különböző komponensek egységesen értsék meg az adatokat.

6.1.1 Almérő törzsadat (fogyasztói) interfész

Interfész azonosító	**1**
Interfész típusa	Adat interfész
Adatok forráskomponense	LEMP
Adatok célkomponense	KEP
Adatátadás leírása	a felszerelt nagyfogyasztói almérők törzsadatainak lekérdezése
Adatátadás gyakorisága	ütemezetten
Adatátadás kezdeményezője	KEP
Döntési pontok és tisztázandó kérdések	N/A
Alkalmazott technológia	REST API, https, …
Biztonsági követelmények	https protokol, titkosított adatátadás tanúsítványok segítségével, fogadó oldalon adat validálás
Alkalmazott szabvány (ha értelmezett)	-
Adattartalom	fogyasztói almérő törzsadat

6.1.2 Almérő mérési adat (fogyasztói) interfész

Interfész azonosító	**2**
Interfész típusa	Adat interfész

Adatok forráskomponense	LEMP
Adatok célkomponense	KEP
Adatátadás leírása	a felszerelt nagyfogyasztói almérők mérési adatainak lekérdezése
Adatátadás gyakorisága	ütemezetten
Adatátadás kezdeményezője	KEP
Döntési pontok és tisztázandó kérdések	N/A
Alkalmazott technológia	REST API
Biztonsági követelmények	https protokol, titkosított adatátadás tanúsítványok segítségével, fogadó oldalon adat validálás
Alkalmazott szabvány (ha értelmezett)	REST API, https, …
Adattartalom	fogyasztói almérő mérési adat

6.1.3 Szenzor törzsadat interfész

Interfész azonosító	**3**
Interfész típusa	Adat interfész
Adatok forráskomponense	LEMP
Adatok célkomponense	KEP
Adatátadás leírása	szenzor törzsadatainak lekérdezése
Adatátadás gyakorisága	ütemezetten
Adatátadás kezdeményezője	KEP
Döntési pontok és tisztázandó kérdések	N/A
Alkalmazott technológia	REST API
Biztonsági követelmények	https protokol, titkosított adatátadás tanúsítványok segítségével, fogadó oldalon adat validálás
Alkalmazott szabvány (ha értelmezett)	-

Adattartalom	szenzor törzsadat

6.1.4 Szenzorok mérési adatainak lekérés interfész

Interfész azonosító	**4**
Interfész típusa	Adat interfész
Adatok forráskomponense	LEMP
Adatok célkomponense	KEP
Adatátadás leírása	szenzor mérési adatainak lekérdezése
Adatátadás gyakorisága	ütemezetten
Adatátadás kezdeményezője	KEP
Döntési pontok és tisztázandó kérdések	N/A
Alkalmazott technológia	REST API
Biztonsági követelmények	https protokol, titkosított adatátadás tanúsítványok segítségével, fogadó oldalon adat validálás
Alkalmazott szabvány (ha értelmezett)	-
Adattartalom	szenzor mérési adat

6.1.5 Közintézményi törzsadatok lekérdezés interfész

Interfész azonosító	**5**
Interfész típusa	Adat interfész
Adatok forráskomponense	LEMP
Adatok célkomponense	KEP
Adatátadás leírása	közintézményi törzsadatainak lekérdezése

Adatátadás gyakorisága	ütemezetten
Adatátadás kezdeményezője	KEP
Döntési pontok és tisztázandó kérdések	N/A
Alkalmazott technológia	REST API
Biztonsági követelmények	https protokol, titkosított adatátadás tanúsítványok segítségével, fogadó oldalon adat validálás
Alkalmazott szabvány (ha értelmezett)	-
Adattartalom	közintézmény törzsadat

6.2 Szolgáltatás interfész

A szolgáltatásinterfész olyan mechanizmus, amely lehetővé teszi különböző szolgáltatások vagy alkalmazások közötti funkcionalitás megosztását. Ez az interfész a szolgáltatások által nyújtott funkciókat és ezek elérésének módját határozza meg, például API-kon keresztül.

6.2.1 Inverter típusú eszköz vezérlés interfész

Interfész azonosító	**6**
Interfész típusa	Adat interfész
Adatok forráskomponense	LEMP
Adatok célkomponense	KEP
Adatátadás leírása	Inverter típusú eszközök vezérlése
Adatátadás gyakorisága	ütemezetten
Adatátadás kezdeményezője	KEP
Döntési pontok és tisztázandó kérdések	N/A
Alkalmazott technológia	REST API
Biztonsági követelmények	https protokol, titkosított adatátadás tanúsítványok segítségével, fogadó oldalon adat validálás
Alkalmazott szabvány (ha értelmezett)	-

Adattartalom	Inverter vezérlés

6.2.2 Ki/be típusú eszköz vezérlés interfész

Interfész azonosító	**7**
Interfész típusa	Adat interfész
Adatok forráskomponense	LEMP
Adatok célkomponense	KEP
Adatátadás leírása	Ki/Be típusú eszköz vezérlés
Adatátadás gyakorisága	ütemezetten
Adatátadás kezdeményezője	KEP
Döntési pontok és tisztázandó kérdések	N/A
Alkalmazott technológia	REST API
Biztonsági követelmények	https protokol, titkosított adatátadás tanúsítványok segítségével, fogadó oldalon adat validálás
Alkalmazott szabvány (ha értelmezett)	-
Adattartalom	Állítható vezérlés

6.3 Összegző táblázat

Interfész kapcsolat leírás	**Adatkör leírás az Excelben**	**Interfész típus**	**Interfész szolgáltató**	**Interfész használó**
A felszerelt almérők törzsadatainak lekérése	almérők törzsadatai	adat interfész	LEMP	KEP
A felszerelt almérők mérési adatai	almérők mérési adatai	adat interfész	LEMP	KEP
Szenzorok törzsadatainak lekérése (pl.: hőmérő	IoT szenzor definíciós példa	adat interfész	KEP	LEMP

szenzor, épületen belüli lokáció)				
Szenzorok mérési adatainak lekérése	IoT szenzor adatfolyam példa	adat interfész	KEP	LEMP
Közintézményi törzsadatok lekérdezése	Közintézményi törzsadatok adat struktúrája	adat interfész	LEMP	KEP
Inverter típusú eszköz vezérlése	Inverter típusú eszköz vezérlési példa	szolgáltatás interfész	LEMP	KEP
Ki/be típusú eszköz vezérlése	KI/BE típusú eszköz vezérlési példa:	szolgáltatás interfész	LEMP	KEP

7 Adattartalom és adatkörök leírása

Az interfész egyértelmű és biztonságos kommunikációt tesz lehetővé a LEMP és KEP között, támogatva az energiamenedzsment optimalizálását.

- **LEMP → KEP által küldött adatok**:
    - Fogyasztási adatok (időbélyeg, mérési hely azonosító, értékek).
    - Termelési adatok (pl. napelemek kimeneti teljesítménye).
    - Eszközmetaadatok (pl. típus, helyszín).
- **KEP → LEMP által küldött adatok**:
    - Aggregált fogyasztási minták.
    - Termelési előrejelzések.
    - Konfigurációs visszacsatolások.

7.1 Adattartalmak

7.1.1 Közintézményi törzsadatok adat struktúrája

- Üzleti partnerszám (ÜP)
- Szerződéses folyószámla szám (SZFSZ)
- Telephelyek
    - épületek
        - eszközök: almérők, szenzorok
        - érvényességi dátum kezdete
        - érvényességi dátum vége
    - GPS koordináta
        - hosszúság
        - szélesség
        - elszámolási hely-e flag
- Telephely üzleti azonosítója
- Telephely megnevezése
- Telephely cím
- Kapcsolattartó adatok:
    - Név
    - Telefon
    - e-mail cím

7.1.2 Almérők törzsadatai

- almérő azonosító
- almérő helye (telephely, épület)
- almérő GPS koordinátája

7.1.3 Almérők mérési adatai

1.fázis hatásos fogyasztás 15 perces felbontásban

1.fázis meddő fogyasztás 15 perces felbontásban

1.fázis áramerősség

2.fázis hatásos fogyasztás 15 perces felbontásban

2.fázis meddő fogyasztás 15 perces felbontásban

2.fázis áramerősség

3.fázis hatásos fogyasztás 15 perces felbontásban

3.fázis meddő fogyasztás 15 perces felbontásban

3.fázis áramerősség

7.1.4 IoT szenzor definíciós példa

```
{
  "deviceId": "sensor123",
    "location": "GPS",
    "sensorType": "temperature",
  "properties": {
    "desired": {
      "measurements": {
        "temperature": {
          "unit": "Celsius",
          "minValue": -40,
          "maxValue": 125
        },
        "humidity": {
```

```
            "unit": "%",
            "minValue": 0,
            "maxValue": 100
          }
        }
      }
    }
  }
}
```

7.1.5 IoT szenzor adatfolyam példa

```
{
  "deviceId": "sensor123",
  "temperature": 22.5,
  "unit": "Celsius",
  "timestamp": "2024-11-30T13:43:00Z"
}
```

7.1.6 Inverter típusú eszköz vezérlési példa

```
DEVICE_ID = "YourInverterDeviceId"
registry_manager = IoTHubRegistryManager(CONNECTION_STRING)
@app.route('/control-inverter', methods=['POST'])
def control_inverter():
    data = request.json
    method_name = data.get('method_name')
    payload = data.get('payload', {})
    timeout = data.get('timeout', 30)
    try:
        # Direct method hívása az eszközön
        response = registry_manager.invoke_device_method(DEVICE_ID, method_name, json.dumps(payload), timeout)
        return jsonify({"response": response}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

7.1.7 KI/BE típusú eszköz vezérlési példa:

```
DEVICE_ID = "YourDeviceId"
registry_manager = IoTHubRegistryManager(CONNECTION_STRING)
@app.route('/control-device', methods=['POST'])
def control_device():
    data = request.json
    state = data.get('state')
    if state not in ['on', 'off']:
        return jsonify({"error": "Invalid state. Use 'on' or 'off'."}), 400
    method_name = "setDeviceState"
    payload = {"state": state}
    timeout = 30
    try:
        # Direct method hívása az eszközön
        response = registry_manager.invoke_device_method(DEVICE_ID, method_name, json.dumps(payload), timeout)
        return jsonify({"response": response}), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

8 Biztonsági követelmények

A REST API-k használata széles körben elterjedt, de mivel adatokat továbbítanak és szolgáltatásokat nyújtanak, megfelelő biztonsági intézkedések nélkül sérülékenyek lehetnek.

A szükséges biztonsági követelményeket a FEAK Zrt. a szakvéleményezési tevékenysége körében egyedi szoftver- illetve eszközvizsgálattal határozza meg. A vonatkozó feltételekből az alábbi fejezetek kivonatot tartalmaznak, a konkrét, egyedi szoftverek bekötése során ezek pontosításra kerülnek.

8.1 Főbb biztonsági követelmények

8.1.1 Hozzáférés -szabályozás és hitelesítés

- OAuth 2.0 vagy OpenID Connect: Javasolt szabványos protokollok használata a felhasználók hitelesítésére és az engedélyek kezelésére.
- API kulcsok: Minden API híváshoz egyedi, csak meghatározott funkciókra érvényes API kulcsok használata ajánlott.
- Token-alapú autentikáció: JWT (JSON Web Token) vagy más tokenek használata az egyszerűsített és biztonságos hitelesítéshez.
- Korlátozott hozzáférési jogok (RBAC): Szerepkör-alapú hozzáférés-szabályozás alkalmazása a funkciók és az adatok megfelelő védelmére.

8.1.2 Adatátvitel és integritás biztonsága

- HTTPS használata: A kommunikáció titkosításához a TLS protokoll (HTTPS) kötelező az adatok lehallgatásának megakadályozására.
- Titkosítás: Érzékeny adatok (pl. hitelesítési adatok, személyes információk) továbbítása titkosítva.
- CORS szabályok beállítása: Csak meghatározott domain-ek engedélyezése az API hívásokhoz.
- Digitális aláírások vagy Checksum: Az adatok sérülésének vagy manipulációjának felismerésére.
- Időbélyegek használata: A replay-támadások elkerülésére.

8.1.3 Adathalászat és visszaélések elleni védelem

- Rate Limiting: A kérésenkénti és felhasználónkénti hívások számának korlátozása a túlterheléses támadások ellen.

- IP cím alapú korlátozások: Csak megbízható IP-címekről érkező kérések engedélyezése.

8.1.4 Adat biztonsági osztályozás és naplózása

- Érzékeny adatok védelme: Személyes és bizalmas információk (pl. jelszavak) hash-elése vagy titkosítása az adatbázisokban.
- Audit és naplózás: Minden API hívás naplózása, amely tartalmazza a kérések és válaszok részleteit, időbélyegekkel. A naplókat védeni kell a jogosulatlan hozzáféréstől.

8.1.5 Verziókezelés, hibakezelés és szabványok

- Verziókezelés: Az API különböző verzióinak szétválasztása a kompatibilitási problémák elkerülése érdekében.
- Hibakezelés: A hibaüzenetek nem tartalmazhatnak érzékeny információkat, amelyek segíthetnek egy támadónak.
- Az API-k biztonságának biztosításához ajánlott szabványos eszközök és keretrendszerek használata (pl. OWASP API Security Top 10 iránymutatások).

Ezek az intézkedések megfelelő alapot nyújtanak egy REST API biztonságának növelésére, miközben biztosítják az adatok és a szolgáltatások integritását és megbízhatóságát.

8.2 API Hitelesítés példa

Azonosító:	**ApiAuthenticate/GetSecOperationMatrix**		
Név:	Biztonsági műveleti mátrix lekérdezése		
Hívó:	Keretrendszer	**Kiszolgáló:**	Keretrendszer
Leírás:	Az API Hitelesítés során a biztonsági műveleti mátrix lekérdezése.		
Gyakoriság:	Közepes		
Hívásparaméterek:	Nincsenek paraméterek		
Válasz:	- Egyszerű szöveg		

Azonosító:	**ApiAuthenticate**		
Név:	API Hitelesítés		
Hívó:	Keretrendszer	**Kiszolgáló:**	Keretrendszer

Leírás:	Az API Hitelesítésének folyamata.
Gyakoriság:	Közepes
Hívásparaméterek:	Nincsenek paraméterek
Válasz:	- Egyszerű szöveg

Azonosító:	**ApiAuthenticate/GetSecOperationMatrix**		
Név:	Biztonsági műveleti mátrix lekérdezése		
Hívó:	Keretrendszer	**Kiszolgáló:**	Keretrendszer
Leírás:	A felület visszaadja a biztonsági műveleti mátrix lekérdezések folyamat során elérhető hitelesítéseket.		
Gyakoriság:	Közepes		
Hívásparaméterek:	Nincsenek paraméterek		
Válasz:	- Egyszerű szöveg		