

Debreceni Egyetem  
Informatikai Kar

# Webes UI fejlesztése a NAV Online Számla rendszeréhez

**Témavezető:** Dr. Pánovics János  
Beosztása: adjunktus

**Készítette:** Kovács Tamás  
Szak: gazdaságinformatikus BSc

Debrecen,  
2020

# Tartalomjegyzék

1. Bevezetés .....	4
2. Online Számla alapok .....	6
2. 1. Jogszabályi háttér .....	6
2. 2. Történeti áttekintés .....	8
2. 3. Regisztráció és felhasználótípusok .....	9
2. 3. 1. Elsődleges felhasználó .....	9
2. 3. 2. Másodlagos felhasználó .....	10
2. 3. 3. Technikai felhasználó .....	10
2. 4. Operációk leírása és használatuk HTTP kéréseken keresztül .....	10
2. 4. 1. <i>BasicRequestType</i> és <i>BasicResultType</i> .....	11
2. 4. 2. A <i>requestSignature</i> elem .....	11
2. 4. 3. /manageAnnulment .....	12
2. 4. 4. /manageInvoice .....	12
2. 4. 5. /queryInvoiceCheck .....	13
2. 4. 6. /queryInvoiceData .....	13
2. 4. 7. /queryInvoiceDigest .....	13
2. 4. 8. /queryTransactionStatus .....	14
2. 4. 9. /queryTaxpayer .....	15
2. 4. 10. /tokenExchange .....	15
3. A felhasznált technológiákról .....	16
3. 1. MongoDB .....	16
3. 2. Node.js .....	17
3. 2. 1. express és express-session .....	17
3. 2. 2. body-parser .....	17
3. 2. 3. method-override .....	17
3. 2. 4. mongoose .....	18
3. 2. 5. request .....	18
3. 2. 6. xml-js .....	18
3. 2. 7. full-icu .....	18
3. 2. 8. Embedded JavaScript (EJS) .....	18
3. 2. 9. connect-flash .....	19
3. 2. 10. js-sha512 és js-sha3 .....	19

3. 2. 11. js-base64 .....	19
3. 2. 12. aes-ecb .....	19
3. 3. Bootstrap.....	20
3. 4. jQuery .....	20
3. 4. 1. BootstrapValidator.....	20
4. A kódtárház elemei .....	21
4. 1. envvar.sh.....	21
4. 2. app.js.....	21
4. 3. public könyvtár .....	22
4. 4. models könyvtár .....	22
4. 5. request könyvtár .....	22
4. 6. routes könyvtár .....	23
4. 7. middleware könyvtár .....	23
4. 8. views könyvtár.....	24
5. Adatsémák .....	25
5. 1. User.....	25
5. 2. Transaction .....	26
5. 3. Line.....	28
6. Az alkalmazás bemutatása.....	29
7. További fejlesztési lehetőségek .....	38
8. Összefoglalás .....	40
9. Irodalomjegyzék .....	41

# 1. Bevezetés

Magyarországon az elmúlt évtizedben komoly átalakításokon ment keresztül a közigazgatási rendszer, fő stratégiai célként kitűzve, hogy olyan „szolgáltató állam” jöjjön létre, melynek alapvető ismérvei a szervezettség, a költséghatékonyság és a professzionalizmus. Az átalakítások között megemlíthetjük többek között egyes közigazgatási intézmények összevonását, az „egyablakos ügyintézési rendszer” kialakítását vagy a Nemzeti Közzolgálati Egyetem létrehozását. A legfontosabb intézkedésnek viszont az elektronikus közigazgatási szolgáltatások fejlesztése mondható, mely szolgáltatásoknak magja az állami Ügyfélkapu rendszer. Napjainkban elektronikus úton van lehetőség egyéni vállalkozás indítására, felsőoktatási felvételi jelentkezésre, bírságokkal kapcsolatos ügybetekintésre, szja-bevallás beküldésére és számos más ügytípussal kapcsolatos ügyintézésre is.

Az a tény, hogy az elektronikus úton való ügyintézés jelentősen megkönnyíti és meggyorsítja az adatok küldését, valamint feldolgozását, lehetőséget nyitott új gazdaságfehérítő intézkedésekre is. Ezen intézkedések lényege, hogy a vállalkozások rendszeres adatszolgáltatások teljesítésére lettek kötelezettek a Nemzeti Adó- és Vámhivatal (a továbbiakban: NAV) felé, az adatok szolgáltatására pedig kizárólag elektronikus úton van lehetőség. Első lépésként a pénztárgépeken rögzített adatok továbbítását írta elő jogszabály (48/2013. NGM rendelet), majd a szállításokhoz kapcsolódó adatok fuvarozás előtti továbbítását is (5/2015. NGM rendelet). Utóbbi kapcsán elindításra került az Elektronikus Közúti Áruforgalom Ellenőrző Rendszer (EKÁER). A szállítási adatok rögzítésére lehetőség van az [ekaer.nav.gov.hu](http://ekaer.nav.gov.hu) oldalon ügyfélkapus azonosítást követően, de API-n keresztül is, így olyan cégek esetén, amelyek nagyszámú fuvart indítanak napi szinten, az adatszolgáltatás meggyorsítható egy, a vállalatirányítási rendszerükhöz kapcsolódó egyedi fejlesztés segítségével.

A NAV Online Számla rendszere 2018 nyarán indult, felületet biztosítva a vállalkozások immár újabb kötelezettségének teljesítésére, mely a kibocsátott számlákkal kapcsolatos adatok szolgáltatására vonatkozik. Hasonlóan az EKÁER-hez, az adatok továbbítása itt is történhet böngészős felületen az [onlineszamla.nav.gov.hu](http://onlineszamla.nav.gov.hu) oldalon és REST API kérések révén is, melyek esetén a vállalkozás által igényelhető, úgynevezett *technikai felhasználók* segítségével történik az azonosítás. Az adatszolgáltatási kötelezettség érvénybe lépéséhez igazodva a számlázó programok fejlesztői is kiadták a megfelelő frissítéseket szoftverjeikhez, amelyek lehetővé

teszik, hogy a technikai felhasználók hitelesítési adatainak megadását követően a programban elmentett számlák automatikusan beküldésre kerüljenek a NAV-nak.

Dolgozatom és a hozzá kapcsolódó fejlesztés célja egy olyan felhasználóbarát webes alkalmazás készítése és dokumentálása, mely képes az Online Számla rendszer operációinak REST API-n keresztül történő végrehajtására. Az alkalmazás további alapvető funkciói a felhasználókezelés, testre szabható felhasználói profilok és a kérés előzmények visszanezethetősége. Ezek megvalósításán túl kísérletet teszek egy-két kényelmi funkció hozzáadására is.

Témaválasztásomat elsősorban a probléma aktualitása indokolja. Jelenleg is készül szerte az országban számos, éles környezetben használni kívánt, ehhez hasonló interfész, legyen az web alapú vagy valamilyen könyvelési program, esetleg ERP rendszer kiegészítése. Ezen alkalmazás fejlesztése révén lehetőségem van egy nagy gyakorlati jelentőséggel bíró kormányzati informatikai rendszer működési mechanizmusainak megismerésére, valamint arra, hogy tapasztalatot nyerjek API hívások és titkosítási módszerek implementálásában, mivel az Online Számla rendszer ezek közül a legkorszerűbbeket használja.

A webes megvalósítás mellett szól elsősorban annak egyszerűsége és platformfüggetlensége, ugyanakkor egy lehetőség is arra, hogy elméleti és gyakorlati ismereteket egyaránt szerezzek és elmélyítsek az olyan korszerű webalkalmazás-fejlesztési technológiák terén, mint a Node.js.

Természetesen nem célom – elsősorban kapacitáshiány miatt – egy üzleti felhasználásra is alkalmas szolgáltatás előállítása, viszont a készített keretrendszer további fejlesztésekkel akár piacképes is formálható lehet. A dolgozatom végén összegzem az alkalmazás gyenge pontjait, melyek feltétlenül javítást igényelnének az éles használathoz, valamint javaslatokat fogok tenni további lehetséges fejlesztési irányokra is.

## 2. Online Számla alapok

Az alábbiakban ismertetni fogom az Online Számla rendszerrel összefüggésben levő jogszabályokat, bemutatom a rendszer fejlődésének főbb mérföldköveit, valamint röviden összefoglalom az API specifikációját és annak helyes használatát.

### 2. 1. Jogszabályi háttér

A számlázással kapcsolatos olyan alapvető rendelkezéseket, mint a számlakibocsátási kötelezettség vagy a számlákra vonatkozó tartalmi és formai követelmények, az általános forgalmi adóról szóló 2007. évi CXXVII. törvény foglalja magában. Az online adatszolgáltatási kötelezettségről a törvény 10. számú melléklete rendelkezik, mely 2018. július 1-től hatályos. Ennek 5-6. pontjai a jelenleg érvényes értékhatárokat határozzák meg, amelyeket hogyha elér az áthárított adó összege, akkor az adott számla vonatkozásában az adatszolgáltatás kötelező.

---

*5. Az adóalany termék értékesítése, szolgáltatás nyújtása esetén azon nyomdai úton előállított nyomtatvány használatával kibocsátott számlákról, amelyekben egy másik, belföldön nyilvántartásba vett adóalanyra áthárított adó összege*

*a) a 100 000 forintot eléri vagy meghaladja, de az 500 000 forintot nem éri el, a számla kibocsátását követő öt naptári napon belül,*

*b) az 500 000 forintot eléri vagy meghaladja, a számla kibocsátását követő naptári napon belül számlánként köteles a számla 169. § szerinti tartalmáról adatot szolgáltatni.*

*6. Az adóalany külön jogszabályban meghatározott elektronikus módon számlánként adatszolgáltatást teljesít az állami adó- és vámhatóság részére azon számlázási funkcióval rendelkező programmal kiállított számlák külön jogszabályban meghatározott adattartalmáról, amelyekben egy másik, belföldön nyilvántartásba vett adóalanyra áthárított adó összege a 100 000 forintot eléri vagy meghaladja. Az adóalany ezen számlákat érintő módosításról vagy érvénytelenítésről is külön jogszabályban meghatározott módon elektronikus adatszolgáltatást teljesít. Szintén külön jogszabályban meghatározott módon kell elektronikus adatszolgáltatást teljesíteni azon módosításokról, amikor a módosítást követően éri el vagy haladja meg a 100 000 forintot a számlában áthárított adó.*

---

A 7-8. pontok azzal kapcsolatban pontosítanak, hogy az adatszolgáltatást akkor is el kell végezni, ha egy számlában történt módosítás következtében kerülnek elérésre ezen értékhatárok, illetve ha egy olyan számla kerül érvénytelenítésre vagy módosításra, amelyik kapcsán már történt korábbi adatszolgáltatás. Maga az Online Számla rendszer, mint felület, a 9. pontban kerül említésre:

---

*9. Az 5-8. pont szerinti adatszolgáltatást az állami adó- és vámhatóság által erre a célra biztosított elektronikus felületen kell teljesíteni. Az elektronikus felület az adóalany egyedi azonosítására szolgáló adatok igénylését követően használható. Az azonosító adatokat az adóalany vagy annak Air. szerinti állandó meghatalmazottja igényli.*

---

Természetesen a számlákról az adatszolgáltatást akkor is el lehet végezni, ha nem kerülnek elérésre a meghatározott értékhatárok (13. pont értelmében), azonban ez jelenleg még nem kötelező.

A pénzügyminiszter 2/2018. (VI. 1.) PM rendelete módosította a számla és a nyugta adóigazgatási azonosításáról, valamint az elektronikus formában megőrzött számlák adóhatósági ellenőrzéséről szóló 23/2014 (VI. 30.) NGM rendeletet is. Ennek a módosításnak értelmében a számlázó programokkal szemben támasztott új követelmény, hogy

---

*biztosítsa a kiállított számla, valamint számlával egy tekintet alá eső okirat legalább Áfa tv. szerinti kötelező adattartalmának az állami adó- és vámhatóság részére történő, 13/A-13/B. § szerinti elektronikus úton történő továbbítását.*

---

A rendeletet kiegészítő 13/A-13/B paragrafusokban kerül meghatározásra az adatszolgáltatást leíró adatszerkezet, valamint az, hogy mi a teendő esetleges hibás adatszolgáltatás vagy az Online Számla rendszer üzemzavara esetén.

A NAV által kiadott 3006/2018. útmutató 56. és 69. pontjának értelmében elmulasztott vagy nem megfelelő adatszolgáltatás esetén számlánként akár 500 000 forintos bírságra is számíthatnak az adózók.

## **2. 2. Történeti áttekintés**

**2015. november** – Elkészült az első előterjesztés a számlázó programok online adatszolgáltatásáról. A bevezetés céldátuma ekkor 2017. január 1. volt.

**2017. április 29.** – A Nemzetgazdasági Minisztérium (NGM) bejelenti, hogy jövő év július 1-jétől lesz kötelező a számlázó programok esetén az online adatszolgáltatás.

**2017. június 27.** – Megjelent az NGM oldalán a jogszabály-módosítási tervezet és az adatszolgáltatási rendszer fejlesztésének hatásvizsgálata.

**2018. január** – Megjelent az Online Számla interfész specifikációja és elindult a tesztüzem, lehetőséget adva a számlázó programok fejlesztőinek a rendszer kipróbálásához.

**2018. június 18.** – Megnyílt a regisztráció az éles rendszerbe.

**2018. július 1.** – Elindult az éles rendszer.

**2018. december** – Bejelentésre került az 1.1-es interfész. Fontosabb változások: XML séma módosítások, új és módosított hibaüzenetek, új blokkoló validációk.

**2019. június 4.** – Ettől a naptól kivezetésre került az 1.0-ás verzió, az adatszolgáltatás már csak az 1.1-es verzió szerint lehetséges.

**2019. augusztus** – Megjelent a 2.0-ás interfész specifikációja. Újdonságai közé tartozik az aláírás számításához használt algoritmus lecserélése SHA-512-ről SHA3-512-re, új operációk és a módosító, valamint érvénytelenítő számlák beküldésével kapcsolatos új szabályok bevezetése.

**2020. április 1.** – A 2.0-ás interfészre való kötelező átállás eredeti céldátuma. A koronavírus járvány okozta rendkívüli helyzetre való tekintettel ezt a határidőt utólag 2020. július 1-jére módosították.

**2020. július 1.** – Várhatóan eltörlésre kerül az áthárított adó értékhatára, tehát minden belföldi áfaalany részére kiállított számla adatait kötelező lesz az Online Számla rendszerben feladni.

**2021. január 1.** – Kötelező lesz várhatóan a magánszemélyek részére kiállított számlákról is az adatszolgáltatás. A cél, hogy 2021 tavaszától a NAV elkészíthesse a vállalkozások ÁFA bevallási tervezetét.



## **2. 3. Regisztráció és felhasználótípusok**

Fontos mindenekelőtt megjegyezni, hogy az éles Online Számla rendszer mellett (melynek webcíme <https://onlineszamla.nav.gov.hu/>) továbbra is elérhető egy tesztrendszer (<https://onlineszamla-test.nav.gov.hu/> címen), melyben lehetőség van az operációk működésének kipróbálására. Ez elsősorban az új számlázóprogramok fejlesztőinek hasznos, ugyanis az itt beküldött számlaadatok nem minősülnek hatósági adatszolgáltatás részének, így azoknak nem kell valósnak lenniük. A két rendszerbe való regisztráció egymástól függetlenül történik. Az egyes felhasználók regisztrálásának és a jogosultságok részletes leírása megtalálható a rendszer felhasználói kézikönyvében (ld. Irodalomjegyzék [11]).

### **2. 3. 1. Elsődleges felhasználó**

Elsődleges felhasználó regisztrálására az adózó, az adózó törvényes képviselője vagy állandó meghatalmazottja jogosult. A regisztráció elvégzésének feltétele, hogy rendelkezzen a Központi Ügyfél-regisztrációs Nyilvántartásban (a köznyelvben Ügyfélkapuként ismert) tárhellyel. A főoldalon a „Regisztráció” majd az „Adatszolgáltatásra kötelezett adózói regisztráció” gombokra való kattintást követően az oldal átirányít az Ügyfélkapus azonosítás oldalára, ahol az azonosítást követően a regisztrációt végző személy meg kell adja az adóazonosító jelét. Ezt követően választani kell egy tetszőleges felhasználónevet, majd beírni az egyéni vállalkozás vagy képviselt gazdálkodó szervezet adószámát. Miután ezeket megadtuk, a rendszer ellenőrzi az adózó érvényességét, valamint azt, hogy a regisztrációt végző felhasználó jogosult-e a megadott adóalany képviselőjére. Ha ezek teljesülnek, meg kell adni még egy belépési jelszót és az adózói adatokat a regisztráció befejezéséhez.

Egy regisztrált felhasználóhoz több elsődleges felhasználó is hozzárendelhető, amennyiben több adózó képviselőjére is jogosult. Ugyanakkor arra is lehetőség van, hogy minden adózó képviselője külön felhasználó alatt történjen.

Az elsődleges felhasználók az Online Számla webes felületén teljes körű jogosultsággal rendelkeznek: rögzíthetnek számlaadatokat, ezeket módosíthatják vagy érvényteleníthetik, lekérdezhetnek adatokat kimenő vevői vagy beérkező szállítói számlákról, megtekinthetik az interfész naplót, adózói név- és címadatokat lekérdezhetnek adószám alapján, jóváhagyhatnak technikai érvénytelenítéseket és létrehozhatnak az adózóhoz kapcsolt másodlagos és technikai felhasználókat is.

### 2. 3. 2. Másodlagos felhasználó

Adott adózóhoz kapcsolódó másodlagos felhasználókat az elsődleges felhasználó tud létrehozni belépést követően a „felhasználók” menüpont alatt. A szükséges adatok: felhasználónév, felhasználó e-mail címe, jelszó (és annak megerősítése), valamint opcionálisan a kapcsolattartás telefonszáma és nyelve. Minden másodlagos felhasználó esetében egyénileg testre szabható, hogy milyen jogosultságokkal rendelkezzen a webes felületen.

### 2. 3. 3. Technikai felhasználó

A technikai felhasználók nem alkalmasak a webes felületen történő bejelentkezésre. Arra szolgálnak, hogy HTTP kéréseken keresztül kezdeményezett operációk keretén belül azonosítsák az adózót. Alkalmazásuk elsősorban számlázó programokban jellemző. Létrehozásuk a másodlagos felhasználókhoz hasonló módon történik. Itt csak egy jelszót kell megadni, a felhasználónév automatikusan generálódik, akárcsak az XML aláíró kulcs és az XML cserekulcs. (Ezek szerepéről a későbbiekben szót ejtünk.) A megadható jogosultságok: számlák kezelése, számlák lekérdezése és saját számlák lekérdezése. Utóbbi kettő közül csak az egyik adható meg.

## 2. 4. Operációk leírása és használatuk HTTP kéréseken keresztül

Az alábbiakban röviden bemutatom a rendszerhez kapcsolódó REST API kérések általános szabályait, illetve egyes operációk elemeit a jelenlegi legfrissebb (2.0) specifikáció szerint. További részletek a hivatalos interfészleírásban találhatók (ld. Irodalomjegyzék [12]).

A számlabejelentő interfész hivatkozása <https://api.onlineszamla.nav.gov.hu/invoiceService/v2> az éles rendszer és <https://api-test.onlineszamla.nav.gov.hu/invoiceService/v2> a tesztrendszer esetén. A kéréseket ezekre kell címezni, természetesen a hivatkozás mögé írva a megfelelő REST operáció nevét. A kérések típusa minden esetben POST legyen, a fejlécükben pedig feltétlenül szerepeljen a *content-type* és az *accept* attribútum *application/xml* értékkel. A törzs minden esetben XML formátumú kell legyen, az egyes operációk pontos sémaleírásai és minta XML-ek megtalálhatók a <https://onlineszamla.nav.gov.hu/dokumentaciok> címen. A válaszok szintén XML formátumban érkeznek.

### 2. 4. 1. *BasicRequestType* és *BasicResultType*

Minden REST operáció esetében a kérés legelején szerepelnie kell néhány alapvető adatnak. Ezek alkotják az úgynevezett *BasicRequestType* elemet, melynek a felépítése a következő:

- *header* (a kérés tranzakciós adatai)
  - *requestId* (egyedi azonosító)
  - *timestamp* (a kérés kliensoldali időpontja UTC-ben)
  - *requestVersion* (a kérés verziószáma, a mi esetünkben ez 2.0)
  - *headerVersion* (a header verziószáma, nem kötelező mező)
- *user* (a kérés hitelesítési adatai)
  - *login* (a technikai felhasználó azonosítója)
  - *passwordHash* (a technikai felhasználó jelszavának SHA-512 hash értéke)
  - *taxNumber* (igénybe vevő adószámának első 8 számjegye)
  - *requestSignature* (számításának módja alább)
- *software* (a számlázóprogram adatai): 8 gyerekelemet tartalmaz, őket nem részletezem.

A beküldött POST kérésekre kapott válaszoknak is vannak alapvető adatai, melyek a *BasicResponseType* elembe foglalhatók össze. Felépítése:

- *header* (*BasicRequestType header* elemével megegyező)
- *result* (a kérés végrehajtásának eredménye)
  - *funcCode* (OK vagy ERROR)
  - *errorCode* (ERROR válasz esetén a feldolgozás hibakódja)
  - *message* (a hibakódot magyarázó kísérő üzenet)
- *software* (*BasicRequestType software* elemével megegyező)

### 2. 4. 2. A *requestSignature* elem

A *requestSignature* kiszámításának lépései a következők:

1. Kiszámítjuk az úgynevezett *parciális hitelesítést* a következők összefűzéséből:
  - a. A *requestId* elem értéke
  - b. A *timestamp* elem értéke YYYYMMDDhhmmss maszkkal
  - c. Az igénybe vevő technikai felhasználó XML aláíró kulcsa
2. Amennyiben a végrehajtandó operáció */manageInvoice* vagy */manageAnnulment*, ki kell számítani minden egyes művelet esetén az úgynevezett *index hash* értékeket: az adott

művelet típusának (*invoiceOperation* vagy *annulmentOperation* elem) és a művelet részletezésének BASE64-ben kódolt tartalmának (*invoiceData* vagy *invoiceAnnulment* elem) összefűzését SHA3-512 algoritmussal titkosítjuk. Az egyes műveletek *index hash* értékeiket összefűzzük.

3. Az első pontban kapott karaktersorozathoz hozzáfűzzük a második pontban kapott karaktersorozatot, amennyiben a második lépés alkalmazható. Az így kapott karakterláncnak SHA3-512 hash értéke lesz a *requestSignature* elem értéke.

### 2. 4. 3. /manageAnnulment

A */manageAnnulment* operáció már befejezett adatszolgáltatás technikai érvénytelenítésére szolgál. Itt fontos megjegyezni, hogy a technikai érvénytelenítés **csak akkor használható, ha a korábbi adatszolgáltatás technikai hiba miatt hibás adatokkal valósult meg**. Ha a számla a gazdasági eseményt helyesen írta le, viszont utólag módosítva vagy érvénytelenítve lett, akkor a */manageInvoice* operációval kell adatot szolgáltatni a módosító vagy stornó számláról. Technikai érvénytelenítéseket minden esetben jóvá kell hagyjon egy elsődleges vagy erre jogosult másodlagos felhasználó az Online Számla webes felületén.

Egy kérés legfeljebb 100 érvénytelenítő műveletet tartalmazhat. A kérésnek a *BasicRequestType* adatain felül tartalmaznia kell egy */tokenExchange* operációval igényelt érvényes tokent és az egyes érvénytelenítő műveletek adatait: művelet indexe, művelettípus (*annulmentOperation* elem, értéke minden esetben *ANNUL*) és a további adatok BASE64 kódolásban (*invoiceAnnulment* elem). A további adatok közé tartozik az érvényteleníteni kívánt okirat sorszáma, az érvénytelenítés időbélyege és oka.

A válasz a *BasicResponseType* elemein felül tartalmazza a kérés tranzakció azonosítóját.

### 2. 4. 4. /manageInvoice

A */manageInvoice* segítségével lehet számlaadatokat a NAV felé beküldeni.

A */manageAnnulment*-hez hasonlóan akár 100 műveletet is tartalmazhat egy kérés, és ugyanúgy tartalmaz a *BasicRequestType* adatain felül egy érvényes tokent és a következő műveleti adatokat: művelet indexe, művelettípus (*invoiceOperation* elem, értéke *CREATE*, *MODIFY* vagy *STORNO* – a számla jellegének függvényében) és a további adatok BASE64 kódolásban (*invoiceData* elem). A további adatok: számla sorszáma (módosító számla esetén az eredeti

számla sorszáma is), számla kelte, vevő és szállító adatai, a számlán szereplő tételek adatai és az összesítő adatok.

A válasz a *BasicResponseType* elemein felül tartalmazza a kérés tranzakció azonosítóját.

#### **2. 4. 5. /queryInvoiceCheck**

A */queryInvoiceCheck* operáció segítségével ellenőrizhetjük adott számlaszámhoz tartozó adatszolgáltatás létezését a rendszerben. Csak olyan számlákra kereshetünk, amelyeknek mi vagyunk a kibocsátói vagy befogadói.

A kérésnek a *BasicRequestType* adatain felül tartalmaznia kell a keresett számla számát, a számla irányát (vevői számla esetén *OUTBOUND*, szállítói számla esetén *INBOUND*), valamint opcionálisan a módosító okirat sorsszámát (köteget módosítás esetén) és a kiállító adószámát (amennyiben szállítói számlára keresünk).

A válasz a *BasicResponseType* elemein felül tartalmazza az ellenőrzés logikai értékét.

#### **2. 4. 6. /queryInvoiceData**

A */queryInvoiceData* operáció a */queryInvoiceCheck*-hez hasonló, azzal a különbséggel, hogy itt a megadott számlaszámhoz tartozó számla teljes adattartalma lekérdezhető.

A kérés formátuma a */queryInvoiceCheck* operációhoz tartozó kérés formátumával megegyező.

A válasz a *BasicResponseType* elemein felül tartalmazza a számla adatait BASE64 kódolással, a számla audit adatait és azt, hogy a BASE64-ben kódolt adatok dekódolást követően ki kell-e még tömöríteni az olvasáshoz. Az audit adatok a következők: az adatszolgáltatás időpontja, forrása és verziószáma, a beküldő technikai felhasználó, valamint opcionálisan az adatszolgáltatás kérésének tranzakció azonosítója, a számla kérésen belüli indexe és a módosító okirat kötegen belüli sorszáma (amennyiben alkalmazható).

#### **2. 4. 7. /queryInvoiceDigest**

A */queryInvoiceDigest* operáción keresztül adott keresési kritériumoknak megfelelő vevői vagy szállítói számlákat tartalmazó „lapozható” lista kérhető le.

Egy érvényes kérés a *BasicRequestType* adatai mellett tartalmaz keresési kritériumokat, a számlairányt (vevői vagy szállítói) és a lekérdezni kívánt lap sorsszámát. A lapok mérete és a

rendezési feltételek kliensoldalon nem befolyásolhatók, ezekről minden esetben a szerver dönt. A megadható keresési feltételek a következő csoportokba sorolhatók:

- **Kötelező paraméterek**, ezek közül legalább egyet kötelezően meg kell adni: számla kiállításának dátumtartománya, számláról való adatszolgáltatás feldolgozásának dátumtartománya és számla száma. A megadott dátumtartományok nem lehetnek 35 napnál hosszabbak. Számlaszám megadása esetén egy számla lánc kérhető le, amely tartalmazza az alapszámlát és minden alapszámlát hivatkozó módosító okiratot.
- **Kiegészítő paraméterek**: ügyfél adószáma, ügyfél neve, számla típusa, fizetési mód, számla megjelenési formája, adatszolgáltatás forrása, számla pénzneme.
- **Relációs paraméterek**: teljesítés dátuma, fizetési határidő, nettó fizetendő összeg és ÁFA összege. Ezek használata esetén egy kereső értéket és kereső operátort kell megadni. Egyenlőségre való keresés esetén az *EQ* kereső operátor használandó. Intervallumra való keresés esetén két paramétert kell megadni: az egyik az alsó korlát, melyet *GT* vagy *GTE* operátor kísér, a másik a felső korlát, melyet *LT* vagy *LTE* operátor kísér, annak függvényében, hogy az intervallum nyílt vagy zárt. Alulról vagy felülről nem korlátos intervallum megadására nincs lehetőség.
- **Tranzakciós paraméterek**: az adatszolgáltatás tranzakciós azonosítója, a tranzakción belüli index és számlaművelet típusa.

A válaszban a *BasicResponseType* adatain túl megjelenik a lekérdezett lap sorszáma, az adott feltételeknek megfelelő találatokat tartalmazó összesen elérhető lapok száma és az egyes számlák adatainak egy szűkített listája. Ha a szűrési feltételeknek megfelelő számlák további adatait, illetve tételeit is szeretnénk megtekinteni, a visszakapott számlák számait adjuk át a */queryInvoiceData* operációnak.

#### 2. 4. 8. */queryTransactionStatus*

A */queryTransactionStatus* operáció arra szolgál, hogy ellenőrizhessük már elvégzett adatszolgáltatások feldolgozottsági állapotát.

A kérdésben szerepelnie kell a *BasicRequestType* alapvető adatainak, az ellenőrizni kívánt tranzakció azonosítójának, valamint annak jelzésének, hogy az adott tranzakción belül beküldött számlák adatait is le kívánjuk-e kérdezni.

A válaszüzenetben a *BasicResponseType* adatok mellett megjelenik a kérés verziószáma, a technikai érvénytelenítés eredménye (amennyiben volt), a kért tranzakcióban beküldött egyes számlák feldolgozási eredménye és validációs hiba esetén a megfelelő technikai hibaüzenet, blokkoló hibaüzenet vagy figyelmeztetés. A lehetséges feldolgozottsági értékek: *RECEIVED*, *PROCESSING*, *SAVED*, *DONE* és *ABORTED*. Amennyiben erre vonatkozó igényünket a kérésben jeleztük, a tranzakción belüli számlák adatait BASE64 kódolásban kapjuk meg.

#### **2. 4. 9. /queryTaxpayer**

A */queryTaxpayer* operáció segítségével ellenőrizhető, hogy létezik-e adott adószámmal rendelkező adóalany az adóhivatal nyilvántartásában. Fontos, hogy csak magyar adószámokra lehet keresni, közösségi adószámokra nem.

Az operációra vonatkozó kérésben csupán a *BasicRequestType* elemeit kell kitölteni, valamint az ellenőrizni kívánt adószám első 8 számjegyét megadni.

A válaszüzenetben megjelennek a *BasicResponseType* adatai, az adószám érvényességének logikai értéke és igaz érték esetén az adószámhoz tartozó adózó neve, címe és esetleges ÁFA csoport tagsága.

#### **2. 4. 10. /tokenExchange**

A */tokenExchange* operáció révén van lehetőség adatszolgáltatási token igénylésére, mely jelen szabályok szerint 5 percre érvényes. Az igényelt token AES-128 ECB szimmetrikus titkosítással kerül kódolásra. Ahhoz, hogy ezt használni tudjuk egy */manageAnnulment* vagy */manageInvoice* típusú kérés beküldésére, dekódolnunk kell a technikai felhasználóhoz tartozó XML cserekulcs segítségével.

A kérés tartalma mindössze a *BasicRequestType*-ből áll.

A válasz a *BasicResponseType*-on túl magában foglalja az igényelt token XML cserekulccsal elkódolt formáját, valamint érvényességének kezdetét és végét jelző időbélyegeket.

### 3. A felhasznált technológiákról

A bemutatott operációk alkalmazására jött létre az *InvoiceBucket* névre hallgató projekt. Az alkalmazás jelen formájában a következő feladatokat képes ellátni egy érvényes technikai felhasználóval rendelkező számára:

1. Egyszerűsített számlákról történő adatszolgáltatások, valamint ezek technikai érvénytelenítése (tranzakciók)
2. Korábban beküldött tranzakciók állapotának ellenőrzése
3. A megadott technikai felhasználóhoz tartozó adóalany számára vagy az adóalany által kiállított számlák adatainak letöltése
4. Az adóhatóság nyilvántartásában szereplő adóalanyok adatainak lekérdezése adószám alapján.

Az elkészítéshez az alábbi fejlesztési technológiákat és eszközöket használtam.

#### 3. 1. MongoDB

Az adatbázis megvalósításának tekintetében a MongoDB-re esett a választás, ami egy NoSQL adatbázis-kezelő rendszer. A választás mögött levő szempontok:

- Az alkalmazás adatsémájának komplexitása és a potenciális adatmennyiség nem indokolja relációs adatbázis használatát.
- Az adatokat rugalmas módon, JSON formátumú dokumentumokban kezeli, könnyű átjárhatóságot képezve a JavaScript nyelv objektum típusú változói és a tárolt adatok között. Az adatok mentését és feldolgozását így jelentősen megkönnyíti.
- Egyre nagyobb népszerűségnek örvend, 2019-ben az 5. legnépszerűbb adatbázis-kezelő rendszer volt, a NoSQL kategóriában viszont egyértelmű fölénye van. Olyan szervezetek fejlesztő csapatai is használják, mint a Facebook, az Electronic Arts vagy az Egyesült Királyság kormánya.
- A rendszer használata ingyenes, így helyi gépen való futtatás esetén nem kell plusz költségekkel számolni. Ha esetleg mégis kényelmesebb megoldás lenne, felhőalapú adatbázisokat is kínál MongoDB Atlas néven a három nagy felhőszolgáltató (Amazon Web Services, Microsoft Azure és Google Cloud Platform).



## 3. 2. Node.js

A Node.js egy 2009-ben létrehozott nyílt forráskódú futtató környezet, mely segítségével JavaScript nyelven lehet szerveroldali kódot írni. A Stack Overflow 2019-es felmérésén az *egyéb keretrendszerek, könyvtárak és eszközök* kategóriában a legnépszerűbb technológiának bizonyult. Mellette szól még a népszerűsége és korszerűsége mellett a változók típusrugalmassága, valamint az is, hogy egy elég nagy közösséggel rendelkezik, melynek tagjai folyamatosan fejlesztenek újabb kiegészítőket. Ezen kiegészítők egyszerű módon, a Node Package Manager (NPM) segítségével hozzáadhatók egy projekthez. Az általam használt kiegészítő csomagok felsorolása alább látható.

### 3. 2. 1. express és express-session

Az express egy viszonylag egyszerű, de ugyanakkor gyors, hatékony és népszerű webfejlesztő keretrendszer Node.js alapú projektekhez. Számomra az útvonalválasztó (routing), a köztes szoftverek beiktatását lehetővé tevő (middleware) és a sablonokat megjelenítő (template engine) funkciói miatt van rá szükség. Egy expressre közvetlenül épülő csomag az express-session, melyet az alkalmazásban a munkamenet alapú felhasználói hitelesítéshez használtam.

### 3. 2. 2. body-parser

Ahhoz, hogy expressben a HTTP POST és PUT típusú kéréseket kezelni tudjuk, egy body-parser nevű middleware szükséges. A bejövő kérés törzse a *req.body* objektum típusú változóban került tárolásra, mely változó a kérést kezelő callback függvény hatókörében hivatkozható. Egykor az express csomag része volt, jelenleg külön kell telepíteni.

### 3. 2. 3. method-override

Sajnos a HTML űrlapoknak van egy olyan rossz tulajdonságuk, hogy nem nyújtható be rajtuk keresztül HTTP PUT és DELETE kérés. Erre a problémára egy kerülőutat nyújt a method-override kiegészítő, mellyel lehetőségünk van a *method* attribútumban szereplő kéréstípust felülbírálni. Ezt úgy tehetjük meg, hogy a HTML űrlap *method* attribútumának értéke legyen *POST*, az *action* attribútumban pedig a hivatkozásunkat követően írjuk be, hogy *?\_method=PUT* vagy *?\_method=DELETE*, annak függvényében, hogy melyik kéréstípussal szeretnénk hogy megtörténjen a felülbírálás.

### 3. 2. 4. mongoose

A mongoose csomag segítségével létre tudunk hozni a MongoDB alapú adatbázisunkhoz adatsémákat JavaScript kódok szintjén. Emellett olyan aszinkron függvényeket is definiál az adatsémákra, amelyekkel lényegesen könnyebb az adatok lekérdezése és manipulációja, mint az alap lekérdező nyelvvel. Néhány példa ilyen függvényre, amit az *InvoiceBucket* is használ: create, findOne, findOneAndUpdate, findById, findByIdAndUpdate, findByIdAndRemove.

### 3. 2. 5. request

Az Online Számla interfészével való kommunikációra a request nevű csomagot használtam. Lényege, hogy egy egyszerű aszinkron függvénnyel tudunk HTTP hívásokat kezdeményezni.

### 3. 2. 6. xml-js

A beküldött kéréseinkre az Online Számla XML formátumú választ ad. Ahhoz, hogy ezeket egyszerűen tudhassuk értelmezni és tárolni, az xml-js nevű kiegészítőt használtam, mely magában foglal többek között egy olyan függvényt, amelynek paraméterként átadva egy érvényes XML szöveget (biztosított, hogy a hatóságtól érvényes XML válasz érkezik), azt átalakítja egy JavaScript objektum típusúvá.

### 3. 2. 7. full-icu

A Node.js alapsomagjába nem lett áthozva az összes regionális formátumokat tartalmazó beállítás (locale), így a magyar dátumformátum megjelenítésére a full-icu kiegészítő csomagot telepítettem. Az alkalmazhatóságához viszont szükséges egy megfelelő környezeti változó beállítása is (erről később).

### 3. 2. 8. Embedded JavaScript (EJS)

Az EJS nevű kiegészítő – ahogy a neve is sugallja – lehetőséget ad arra, hogy a HTML dokumentumainkba JavaScript kódokat ágyazzunk be *script* tagek használata nélkül. Ez jelentősen megkönnyíti a dolgunkat a kliensoldali kódok vonatkozásában. Használata:

- Az express nézetmotorjának beállítjuk az *ejs* értéket: `app.set('view engine', 'ejs')`
- A sablonjainknak *.ejs* kiterjesztést adjunk *.html* helyett.
- Amikor szerver oldalról kérjük egy ilyen sablon renderelését, megadhatjuk milyen szerveroldali változókat szeretnénk felhasználni a sablonunk megjelenítéséhez. Példa:  

```
res.render("sablon_neve",{valtozo1_neve_kliensoldalon: valtozo1_neve_szerveroldalon, valtozo2_neve_kliensoldalon: valtozo2_neve_szerveroldalon [stb]})
```

- Ha szeretnénk egy JavaScript kódrészletet beágyazni a dokumentumunkba, akkor azt `<% és %>` tagek közé tesszük. Példa egy paragrafus megjelenítésének feltételhez kötésére:  

```
<% if(feltetel){ %>
<p>Ez egy paragrafus</p>
<% } %>
```
- Ha egy változónak az értékét szeretnénk a dokumentumban megjeleníteni, akkor azt `<%= és %>` tagek közé tesszük: `<%= változo %>`

### 3. 2. 9. connect-flash

Ez a csomag egy flash middleware-t biztosít, mely része volt az expressnek a 2.x verzióig, azóta külön telepítendő. Ezen middleware segítségével történik a felhasználókkal egyes üzenetek közlése az oldal tetején felvillanó paneleken (például: „Sikeres bejelentkezés” vagy „Hibás jelszó”). Az üzeneteket a *req.flash()* függvény segítségével jeleníthetjük meg az egyes kéréseket kezelő callback függvényeken belül.

### 3. 2. 10. js-sha512 és js-sha3

Az *InvoiceBucket* alkalmazás a regisztráló felhasználók jelszavait azok SHA-512 hash formájában tárolja, mely kiszámítása a js-sha512 kiegészítő egyik függvénye segítségével történik. Az Online Számla a 2.0-s interfésztől kezdve a hitelesítő adatok SHA3-512 hash formáját várja kérések feladásakor. Ezen hash kiszámítására nem tartalmaz függvényt a js-sha512, ezért annak egy rokon csomagját, a js-sha3 nevű csomagot is telepíteni kellett.

### 3. 2. 11. js-base64

Az Online Számla rendszer a beérkező számlaadatokat BASE64 kódolásban fogadja, ennek előállítására használok a js-base64 kiegészítőt, mely rendelkezik egy szöveget BASE64 karaktersorozattá kódoló és egy BASE64 karaktersorozatot olvasható szöveggé dekódoló függvénnyel.

### 3. 2. 12. aes-ecb

Az aes-ecb csomag a vele azonos nevű szimmetrikus titkosítási algoritmust implementálja. 16, 24 és 32 bájtos kulcsokat egyaránt támogat. Egyrészt az igényelt adatszolgáltatási tokenek dekódolására, másrészt az applikáció felhasználóihoz kapcsolt technikai felhasználók adatainak titkosítására használjuk.

### 3. 3. Bootstrap

A Bootstrap egy keretrendszer, mely segítségével lényegesen reszponzívvá és felhasználóbarátabbá varázsolhatjuk a weboldalunk front-endjét anélkül, hogy nagyon bonyolult CSS osztályokat íránk. Népszerűsége megkérdőjelezhetetlen, 2016. márciusáig a legnépszerűbb projekt volt Githubon, jelenleg a 7. helyen áll, több mint 140 ezer csillaggal. Az általam használt verzió a 3.3.7-es, és a következő kliensoldali komponensek hozzáadása terén könnyítette meg jelentősen a dolgom:

- Navigációs sávok (navbar)
- Ikonok (glyphicons)
- Oldalak fejlécei (jumbotron)
- Táblázatok, űrlapok és a hozzájuk tartozó gombok (bár ezek nem Bootstrap-specifikus elemek, jelentősen jobb kinézetűvé és reszponzívvá tehetők a Bootstrap kódjában definiált CSS osztályok hivatkozásával)

### 3. 4. jQuery

A jQuery egy nyílt forráskódú JavaScript könyvtár, mely lehetővé teszi többek között a HTML dokumentumok objektummodelljének (DOM) egyszerű manipulációját. Bár az új weboldalak készítése esetén mára már átveszik a szerepét az olyan keretrendszerek, mint az AngularJS, a React vagy a Vue.js, a 10 millió legnépszerűbb weboldal 73%-a jelenleg is használja valamilyen formában.

#### 3. 4. 1. BootstrapValidator

Bár a dokumentumok kliensoldali manipulálását a korábban említett EJS használata feleslegessé teheti, egy dolog miatt mégis szükség volt rá az *InvoiceBucket* vonatkozásában, ez pedig az űrlapokon való azonnali visszajelzés adása. Erre használtam ezt a jQuery alapú kiegészítőt, amellyel lehetőségünk van űrlapjaink mezőinek érvényességi kritériumokat megadni (pl. kitöltés kötelezővé tétele, minimális karakterhossz vagy reguláris kifejezés). A felhasználó valós időben láthatja, hogy egy adott mezőbe beírt tartalom megfelel-e a támasztott kritériumoknak, és addig nem lehetséges egy űrlap beküldése, amíg az összes mező tartalma érvényes nem lesz.

## 4. A kódtárház elemei

Az *InvoiceBucket* alkalmazás forráskódja nyilvános, bárki által megtekinthető, felhasználható és továbbfejleszhető. A projekt kódtárháza (code repository) a következő linken érhető el: <https://github.com/tamassky/InvoiceBucket/>. Az alábbiakban röviden áttekintjük a fontosabb elemeknek funkcióit.

### 4. 1. envvar.sh

Ebben az állományban találjuk az alkalmazás megfelelő működéséhez szükséges 4 környezeti változó alapbeállítását:

- **DATABASEURL**: a MongoDB adatbázisunk hivatkozása. Amennyiben helyi adatbázist használunk, nem szükséges módosítani, felhős adatbázis használata esetén át kell írni a címet és portot a szolgáltató útmutatásának megfelelően.
- **PORT**: a portnak a száma, amelyen a webszerverünket futtatni szeretnénk.
- **APIURL**: az Online Számla interfész hivatkozása. Tesztrendszerbe történő feltöltés esetén ez <https://api-test.onlineszamla.nav.gov.hu/invoiceService/v2/>, éles rendszer esetén pedig <https://api.onlineszamla.nav.gov.hu/invoiceService/v2/>.
- **NODE\_ICU\_DATA**: értéke konstans 'node\_modules/full-icu' legyen. A magyar dátumformátumok megfelelő megjelenítéséhez kell beállítani.

A webszerver indítása előtt a fenti környezeti változókat állítsuk be, ennek legegyszerűbb módja a `source envvar.sh` parancs terminálban való kiadása.

### 4. 2. app.js

Az **app.js** futtatásával indul el a webszerverünk. Tartalmazza a mongoose felparaméterezését és adatbázishoz való csatlakoztatását, az express, body-parser, ejs, method-override és connect-flash konfigurálását, a munkamenetek beállításait (alapbeállítás szerint egy munkamenet 30 percig / 1800000 milliszekundumig érvényes), a három fő útvonal (index, transaction, lines) valamint a szerverhez fűződő port megadását.

### 4. 3. public könyvtár

Ez a könyvtár mindössze egy képállományt tartalmaz (ingyenes, szabadon használható stockfotó), valamint egy CSS állományt, amely globálisan beállítja ezt a fotót háttérképnek.

### 4. 4. models könyvtár

Itt találhatók az alkalmazásunk 3 mongoose adatbázissémáját tartalmazó állományok:

- **user.js**: felhasználó típusú adatbáziselem modellje
- **transaction.js**: tranzakció típusú adatbáziselem modellje
- **line.js**: sor típusú adatbáziselem modellje

Az egyes adatsémákra a következő fejezetben részletesen kitérek.

### 4. 5. request könyvtár

Az itteni állományok az Online Számla rendszerrel való kommunikációt valósítják meg:

- **request.js**: három függvényt tartalmaz:
  - **setRequest**: aszinkron függvény, minden Online Számla felé irányuló kérés először ide fut be három paraméter révén (kérés típusa, kérést kezdeményező technikai felhasználó adatai és magának a kérésnek az adatai). Előállítja a kérés azonosítót egy ütközést garantáltan elkerülő logika alapján, majd a megfelelő kéréstörzset felépítő függvényt meghívja (*manageInvoice* és *manageAnnulment* esetén annak végrehajtása előtt egy *tokenExchange* alkérés generálódik, hogy az adatszolgáltatáshoz egy érvényes token rendelkezésre állhasson). Miután a kérés törzse felépül, meghívja a *sendRequest* függvényt, melynek visszatérő értéke alapján megtörténik a hibakezelés vagy a válasz feldolgozása a *processResponse* függvény meghívásával. Végül a feldolgozott választ visszatéríti.
  - **sendRequest**: a kérés típusa, az azonosító adatok és a kérés törzse alapján elküldi a kérést az Online Számla rendszer felé. Amennyiben a válasz státuszkódja 200, visszatéríti a válasz törzsének JavaScript objektummá alakított változatát. Ha a státuszkód nem 200, a visszatérési érték „error”, a kapcsolatfelvétel megghiúsulása esetén pedig „server\_error” lesz.

- **processResponse**: az Online Számla rendszer által küldött válaszokból kiszűri a számunkra fontos adatokat és azokat értelmezhető alakra hozza.
- **BasicRequestType.js**: a kérési azonosító és a hitelesítési adatok alapján előállítja a *requestSignature* karaktersorozatot, majd ez alapján felépíti és visszatéríti a *BasicRequestType* XML elemet. Mindegyik kéréstörzsét felépítő függvény meghívja.
- **manageInvoice.js**: *manageInvoice* típusú kérések törzsét építi fel.
- **InvoiceDataSimplifiedCreate.js**: egyszerűsített számlák esetén építi fel az *InvoiceData* XML elemet, majd azt BASE64 kódolásban visszatéríti.
- **manageAnnulment.js**: *manageAnnulment* típusú kérések törzsét építi fel.
- **AnnulmentData.js**: az *AnnulmentData* XML elemet építi fel, majd azt BASE64 kódolásban visszatéríti.
- **queryInvoiceCheck.js**: *queryInvoiceCheck* típusú kérések törzsét építi fel.
- **queryInvoiceData.js**: *queryInvoiceData* típusú kérések törzsét építi fel.
- **queryTaxpayer.js**: *queryTaxpayer* típusú kérések törzsét építi fel.
- **queryTransactionStatus.js**: *queryTransactionStatus* típusú kérések törzsét építi fel.
- **tokenExchange.js**: *tokenExchange* típusú kérések törzsét építi fel.

## 4. 6. routes könyvtár

A routes könyvtár állományai a 3 fő belső útvonal GET, POST, PUT és DELETE kéréseinek kezelőfüggvényeit tartalmazzák:

- **line.js**: a line típusú (vagyis „/transaction/[azonosító]/line” kezdetű URL-lel rendelkező) HTTP kérések callback függvényeit foglalja magában.
- **transaction.js**: a transaction típusú (vagyis „/transaction” kezdetű URL-lel rendelkező, de nem line típusú) HTTP kérések callback függvényeit foglalja magában.
- **index.js**: az index típusú (vagyis az előző két kategória egyikébe sem sorolható) HTTP kérések callback függvényeit foglalja magában.

## 4. 7. middleware könyvtár

Itt az egyetlen található állomány az **index.js**, amely egy tranzakció megnyitási vagy karbantartási kísérletekor kerül meghívásra. Először megkeresi a megtekinteni vagy szerkeszteni kívánt tranzakciót, amennyiben nem találja, hibaüzenetet küld a felhasználónak.

Ezt követően azt ellenőrzi, hogy van-e bejelentkezett felhasználó, és ha igen, akkor az tulajdonosa-e a szóban forgó tranzakciónak. Ha ez nem teljesül, megtagadja a hozzáférést és flash hibaüzenetet küld.

## 4. 8. views könyvtár

Ebben a könyvtárban találhatók az egyes EJS sablonok, melyek egy-egy GET kérés segítségével jeleníthetők meg:

- **landing.ejs:** főoldal
- **register.ejs:** regisztráció
- **login.ejs:** bejelentkezés
- **techdata.ejs:** technikai felhasználó adatainak rögzítése
- **pass.ejs:** jelszóváltoztatás
- **dashboard.ejs:** irányítópult, itt létrehozható új tranzakció, áttekinthetők a meglevő tranzakciók és azok állapota frissíthető
- **arinvoiceData.ejs:** vevői számlák adatainak lekérdezése
- **apinvoiceData.ejs:** szállítói számlák adatainak lekérdezése
- **taxpayerData.ejs:** adózó lekérdezése
- **transactionCS.ejs:** egyszerűsített számlához kapcsolódó tranzakció táblázatos áttekintése
- **basicDataCS.ejs:** egyszerűsített számla alapadatainak karbantartása
- **supplierCS.ejs:** egyszerűsített számla kibocsátójának karbantartása
- **customerCS.ejs:** egyszerűsített számla vevőjének karbantartása
- **linenewCS.ejs:** új sor rögzítése egyszerűsített számlához
- **lineditCS.ejs:** egyszerűsített számla meglevő sorának karbantartása
- **summaryCS.ejs:** egyszerűsített számla összegző adatainak karbantartása



## 5. Adatsémák

Ahogy azt már korábban említettem, az alkalmazásunknak három adatsémája van.

### 5.1. User

A User adatséma írja le az webalkalmazásunkba regisztrált felhasználók alapvető adatait.

Attribútum neve	Típus	Leírás
username	String	Regisztrációkor megadott felhasználónév (egyedi)
password	String	Regisztrációkor megadott jelszó SHA-512 hashje
navUsername	String	A kapcsolt technikai felhasználó titkosított azonosítója
navPassword	String	A kapcsolt technikai felhasználó titkosított jelszava
xmlsign	String	A kapcsolt technikai felhasználó titkosított XML aláíró kulcsa
xmlexchange	String	A kapcsolt technikai felhasználó titkosított XML cserekulcsa
taxnumber	String	A kapcsolt technikai felhasználót kiadó szervezet adószámának első 8 számjegye (törzsszám)
taxpayerName	String	A kapcsolt technikai felhasználót kiadó szervezet neve
creationDate	Dátum	A felhasználó létrehozásának (regisztrálásának) UTC dátuma és időpontja (ISO 8601 szabvány szerint)
lastUpdateDate	Dátum	A felhasználó adataiban történt utolsó módosítás UTC dátuma és időpontja (ISO 8601 szabvány szerint)

1. táblázat. A User adatséma attribútumai

**Megjegyzés:** a kapcsolt technikai felhasználók adatai minden esetben AES-ECB algoritmussal kerülnek titkosításra, melyhez a használt 16 bájtos kulcs a következőképp kerül megállapításra:

- A felhasználó jelszavának első 16 karaktere, ha az legalább 16 karakter hosszú
- A felhasználó jelszava kiegészítve „X” karakterekkel, ha az rövidebb 16 karakternél

Mivel ez a kulcs nem kerül sosem tárolásra, a felhasználó minden Online Számla rendszerrel való kapcsolatfelvétel alkalmával a kérés adataihoz mellékelnie kell az *InvoiceBucket* jelszavát, a megfelelő azonosító adatok dekódolásához.

## 5. 2. Transaction

A Transaction adatséma a létrehozott tranzakciók adatait írja le. Jelenleg csak egyszerűsített számlákkal összefüggésben levő tranzakciókat támogat.

Attribútum neve	Típus	Leírás
number	Szám	Automatikusan generált egyedi azonosító (szekvencia)
owner	String	A tranzakciót létrehozó felhasználó azonosítója (hivatkozás)
invoiceOperation	String	Számlaművelet (pl. „CREATE”)
invoiceNumber	String	Számla sorszáma
invoiceIssueDate	String	Számla kelte (YYYY-MM-DD formátumban)
invoiceCategory	String	Számla típusa (pl. „SIMPLIFIED”)
invoiceDeliveryDate	String	Teljesítés dátuma (YYYY-MM-DD formátumban)
paymentMethod	String	Fizetési mód (pl. „CASH”)
invoiceAppearance	String	Megjelenési forma (pl. „PAPER”)
supplierTaxpayerId	String	Szállító törzsszáma
supplierVatCode	String	Szállító áfakódja
supplierCountyCode	String	Szállító megyekódja
supplierName	String	Szállító neve
supplierPostalCode	String	Szállító székhelyének irányítószáma
supplierCity	String	Kibocsátó székhelyének települése
supplierAdditionalAddressDetail	String	Kibocsátó székhelyének további címadatai
customerTaxpayerId	String	Vevő törzsszáma
customerVatCode	String	Vevő áfakódja
customerCountyCode	String	Vevő megyekódja
customerName	String	Vevő neve
customerPostalCode	String	Vevő székhelyének irányítószáma

customerCity	String	Vevő székhelyének települése
customerAdditionalAddressDetail	String	Vevő székhelyének további címadatai
lines	ObjectId-kat tartalmazó tömb	A tranzakció tárgyát képező számla sorainak azonosítói (hivatkozások Line típusú rekordokra)
vatContent	String	Egyszerűsített számla adótartalma
invoiceGrossAmount	String	Bruttó fizetendő összeg
transactionId	String	Az adatszolgáltatási kérelemről kapott azonosító
transactionStatus	String	Tranzakció állapota, lehetséges értékei: „Piszkozat”, „Aláírva”, „Elfogadva”, „Elutasítva”, „Elfogadva figyelmeztetésekkel” és „Érvénytelenítve”.
transactionValidation	Stringeket tartalmazó tömb	Adatszolgáltatást követően visszakapott hibaüzenetek és figyelmeztetések
annuled	Logikai	Érvénytelenített-e a tranzakció
annulmentId	String	Érvénytelenítési kérelemről kapott azonosító
annulmentCode	String	Érvénytelenítés kódja
annulmentReason	String	Érvénytelenítés indoka
creationDate	Dátum	A tranzakció létrehozásának UTC dátuma és időpontja (ISO 8601 szabvány szerint)
lastUpdateDate	Dátum	A tranzakcióban történt utolsó módosítás UTC dátuma és időpontja (ISO 8601 szabvány szerint)
disabled	Logikai	Törölve lett-e a tranzakció

2. táblázat. A Transaction adatséma attribútumai

## 5. 3. Line

A Line adatséma a számlasorok adatait leíró séma. Csak egyszerűsített számlák sorait tudja tárolni jelen formájában.

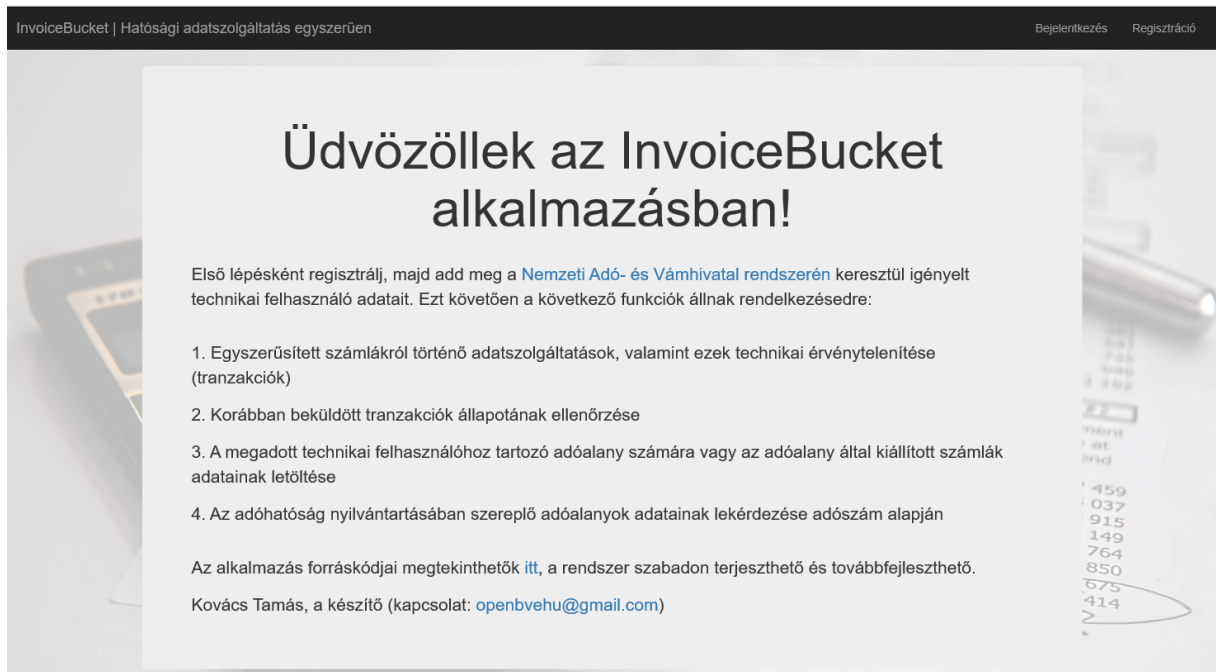
Attribútum neve	Típus	Leírás
lineDescription	String	Termék vagy szolgáltatás megnevezése
unitOfMeasure	String	Mértékegység
unitOfMeasureOwn	String	Egyedi mértékegység
quantity	String	Mennyiség
unitPrice	String	Egységár
lineGrossAmountSimplified	String	Bruttó érték

3. táblázat. A Line adatséma attribútumai

Jogosan tevődhet fel a kérdés, hogy miként hivatkozzuk a Transaction típusú dokumentumokban a *line* attribútumok tömbjeiben az egyes sorokat, ha azoknak nem adtunk egyedi azonosítót. Erre a válasz a mongoose által automatikusan hozzárendelt ObjectId, mely minden esetben hozzáadódik a rekordokhoz, a sémákban való explicit deklaráció nélkül is.

## 6. Az alkalmazás bemutatása

Az alkalmazásunk főoldalán néhány gyors információ, felül pedig egy navigációs sáv fogad bennünket, egy „Bejelentkezés” és egy „Regisztráció” gombbal.



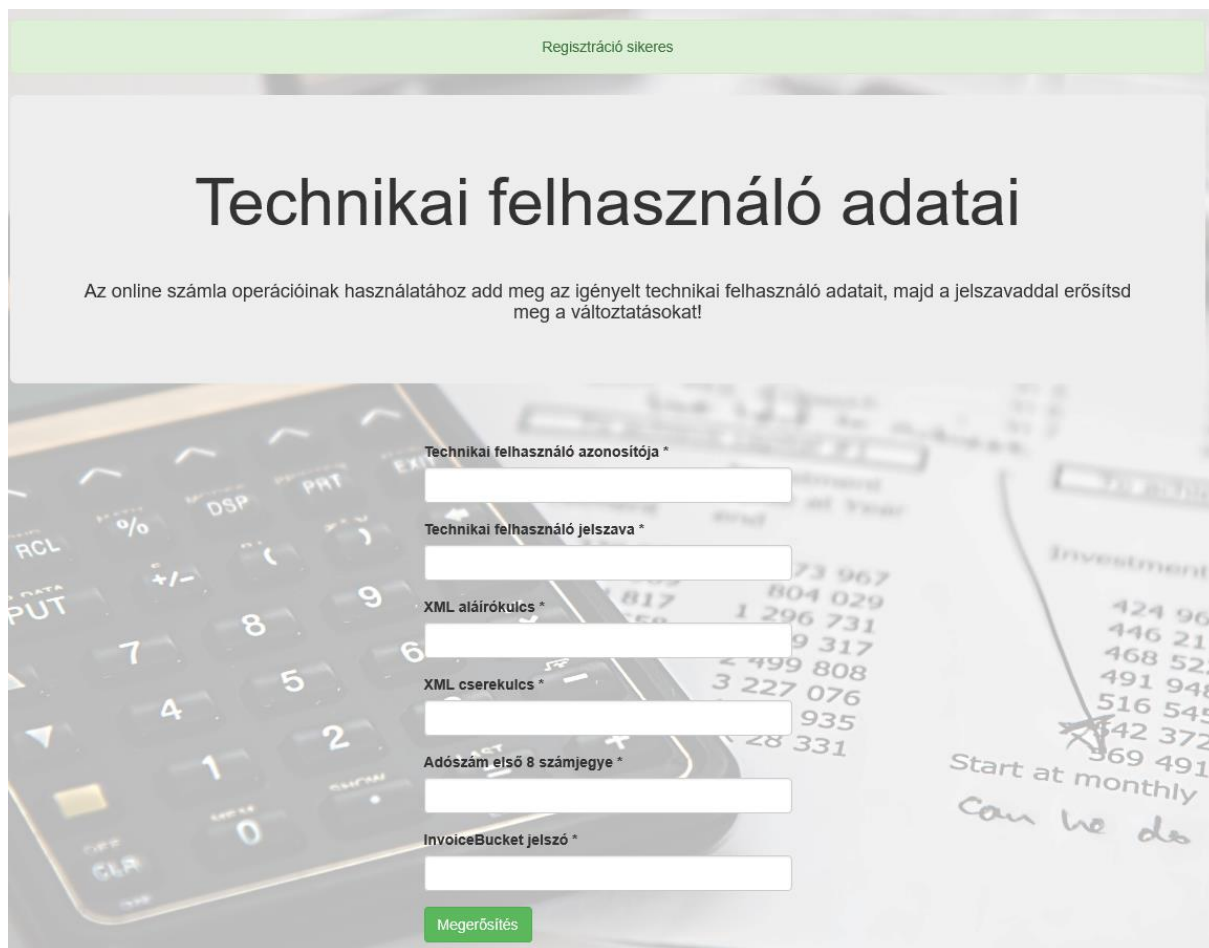
1. ábra. Főoldal

A regisztrációs űrlapon olvashatók a jelszóval szemben támasztott követelmények. Ezek teljesüléséről az űrlap valós idejű visszajelzést ad.

The screenshot shows the registration form titled "Regisztráció". It includes a sub-header: "A regisztráláshoz adj meg egy felhasználói azonosítót és egy megfelelően erős jelszót! (Legalább 8 karakter hosszú legyen, valamint tartalmazzon kisbetűt, nagybetűt és számot egyaránt!)". The form has three input fields: "Felhasználónév" (containing "cure\_for\_covid19" with a green checkmark), "Jelszó" (containing "\*\*\*\*\*" with a red X and the text "Túl gyenge jelszó"), and "Jelszó megerősítése" (empty). At the bottom, there are two buttons: "Regisztrálok" (green) and "Vissza" (blue).

2. ábra. Regisztráció

Regisztrációt követően annak a technikai felhasználónak az adatait kell megadni, amellyel szeretnénk az adatszolgáltatásokat végezni. Mivel ezek kritikus adatok, egészen addig ide leszünk visszairányítva, míg érvényes adatokat nem rögzítünk.

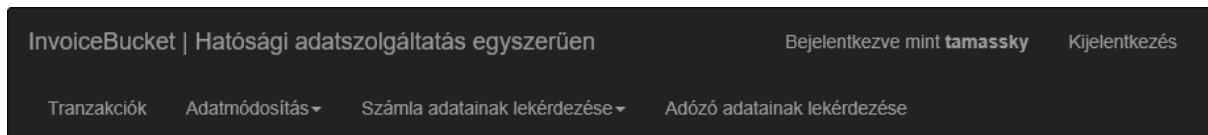


3. ábra. Technikai felhasználó adatainak bekérése

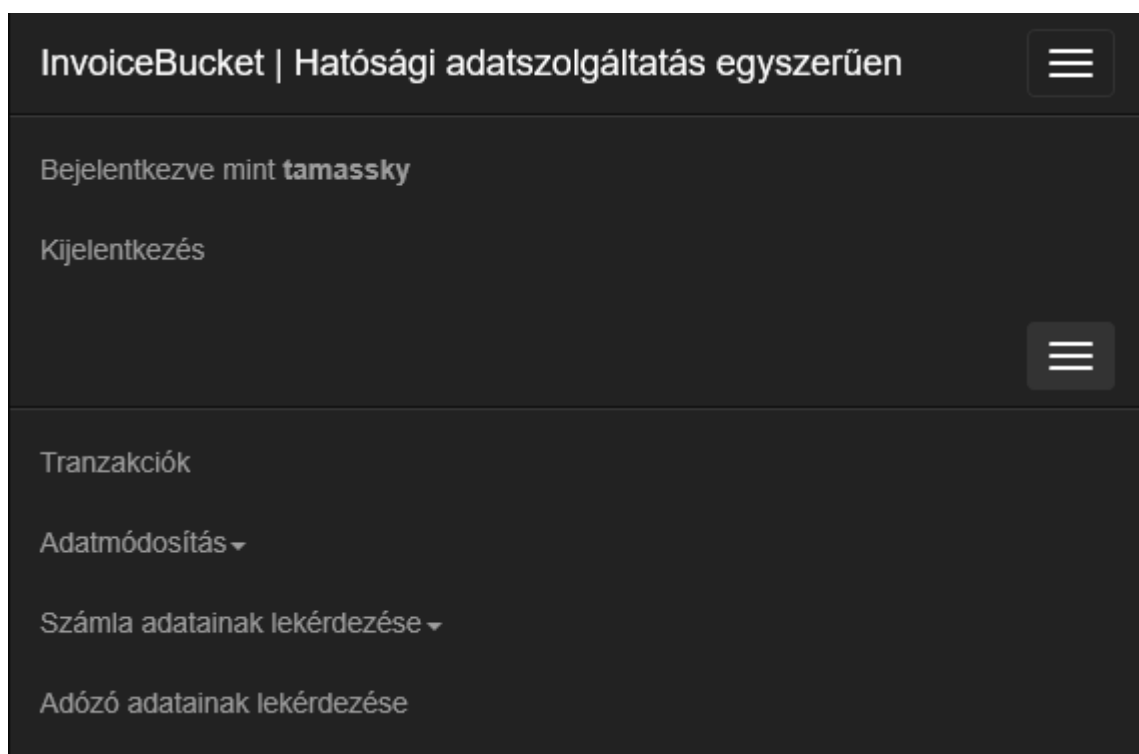
Megerősítést követően egy tesztkéres kerül feladásra az Online Számlának az adatok ellenőrzésére. Így hibás vagy lejárt adatok mentése nem lehetséges, kivéve az XML cserekulcs, amely ezen a ponton még nem kerül ellenőrzésre. A megadott adatok – titkosak lévén – a jelszavunkkal titkosítva kerülnek lementésre. Az adatok bármikor megváltoztathatók az „Adatmódosítás” fülön, természetesen minden módosítás alkalmával validálásra kerülnek. Mivel az alkalmazás lehetővé teszi a jelszómódosítást is, annak alkalmával először a régi jelszó megadásával visszafejtésre kerülnek a technikai adatok majd az új jelszóval kerülnek ismét titkosításra.

Ha van felhasználó bejelentkezve, akkor az oldal tetején két navigációs sáv is megjelenik. Egyiken láthatjuk a felhasználónevünket és egy „Kijelentkezés” gombot, a másikon pedig

böngészhetünk az alkalmazás funkciói között. Ha megfelelően kicsire csökkentjük a böngészőnk ablakát, ez a két sáv összezsugorodik és „hamburger ikonokkal” lesz függőlegesen lenyitható.



4. ábra. Navigációs sávok nagyobb böngészőablak esetén



5. ábra. Navigációs sávok kisebb böngészőablak esetén

Bejelentkezés után a felhasználó a dashboardjára lesz irányítva, ahol áttekintheti korábbi tranzakcióit, valamint újakat kezdeményezhet. Jelenleg csak egyszerűsített számlák adatszolgáltatásához vehetők fel új tranzakciók. Ez az „Új” legördülő menü megnyitásával, majd az „Egyszerűsített számla” gomb lenyomásával lehetséges. Meglévő tranzakció részleteinek áttekintéséhez a sorszáma rá kattintva van lehetőség. Törölni kizárólag „Piszkozat” státusz esetén lehet, és nem történik tulajdonképpeni törlés sem, így a törölt tranzakciók egy webadmin által bármikor visszaállíthatók.

# Tranzakciók

Az adatszolgáltatások a következő szervezet nevében fognak történni:  
DANIELLA KERESKEDELMI KORLÁTOLT FELELŐSSÉGŰ TÁRSASÁG

InvoiceBucket jelszó \*

Aláírt tranzakciók frissítése

Új

#	Tranzakció típusa	Számlaszám	Kapott azonosító	Állapot	Utolsó módosítás (UTC)	Törítés
7	Egyszerűsített számla			Piszkozat	2020. 05. 02. 12:59:05	
6	Egyszerűsített számla	2020/30002430		Piszkozat	2020. 05. 02. 13:10:27	
5	Egyszerűsített számla	2020/30003033	2YBCE16MBD065MTT	Aláírva	2020. 05. 02. 12:41:03	
4	Egyszerűsített számla	2020/102	2YBC8UN2Q64VILUX	Elfogadva	2020. 05. 02. 12:37:20	
3	Egyszerűsített számla	2020/102	2YBC3UAQAM04M8XE	Elutasítva	2020. 05. 02. 12:37:20	
2	Egyszerűsített számla	2020/101	2YBBC89JLVD6PO8E	Elfogadva	2020. 05. 02. 12:11:48	
1	Egyszerűsített számla	2020/101	2YBAWZI2CFG9TOF	Érvénytelenítve	2020. 05. 02. 12:03:15	

6. ábra. Dashboard

## Egyszerűsített számla

### Tranzakció #6 (Piszkozat)

#### Számla alapadatai

Sorszám	2020/30002430
Számla kelte	2020-05-02
Teljesítés dátuma	2020-05-02
Fizetési mód	Készpénz
Megjelenési forma	Papír alapú számla

#### Számlakibocsátó adatai

Adószám	10683424209
Név	Daniella Kereskedelmi Kft
Irányítószám	4031
Település	Debrecen
További cím adatok	Köntösgát sor 1-3

#### Vevő adatai

Adószám	15916511111
Név	Kapperino Zrt
Irányítószám	1035
Település	Budapest
További cím adatok	Szentendrei út 354.

7. ábra. Tranzakció áttekintés (1)



## Sorok +

Tétel megnevezése	Mértékegység	Egyéni mértékegység	Mennyiség	Egységár	Érték	
A tétel	Darab		1	2000	4000	
B tétel	Perc		5	500	2500	
C tétel	Egyéni	Szerződéskötés	5	20000	100000	
D tétel	Kilogramm		10.55	450	4747.5	

## Összegző adatok

Áfataralom	21.26%
Fizetendő	111247.5

## Tranzakció aláírása

InvoiceBucket jelszó \*

8. ábra. Tranzakció áttekintés (2)

A tranzakció részenként szerkeszthető, ahogy azt a fenti képernyőképek is mutatják. Az egyes részek csak addig szerkeszthetők, illetve új sorok is csak addig vehetők fel és törölhetők, amíg a tranzakció aláírásra nem kerül.

## Sor szerkesztése

Tétel megnevezése \*

 ✓

Mértékegység \*

 ▼

Egyéni mértékegység megnevezése

Mennyiség \*

 ✓

Egységár \*

 ✓

Érték \*

 ✓

9. ábra. Tranzakció részének szerkesztése

Az aláíráshoz a jelszó beírása szükséges. Érvényes jelszó esetén ellenőrzésre kerül, hogy minden rész megfelelően ki lett-e töltve és az, hogy van-e legalább egy felvett sor. Ha ezek teljesülnek, az adatszolgáltatás megtörténik, a tranzakció új státusza „Aláírva” lesz, és egy alábbihoz hasonló visszaigazoló üzenettel jelenik meg egy flash.

**Sikeres adatszolgáltatás! Kapott azonosító: 2YEJY8NJAVL7DEFS**

*10. ábra.* Flash üzenet a sikeres adatszolgáltatásról

Ha a kapcsolt technikai felhasználó XML cserekulcsa hibás, az itt fog kiderülni egy ezt közlő hibaüzenet révén. Ilyenkor újra meg kell adni a technikai felhasználó adatait, mielőtt tranzakciókat aláírhatnánk. Előfordulhat az is, hogy az Online Számla rendszer nem elérhető (pl. karbantartás miatt). Ilyenkor egy felvillanó üzenet erre fogja felhívni a figyelmünk.

A már aláírt tranzakcióink státuszának frissítése az irányítópultunkon található miniürlap segítségével lehetséges. Csupán be kell írni a jelszót, majd megnyomni az „Aláírt tranzakciók frissítése” gombot. Érvényes jelszó esetén a következő flash üzenet fogad minket.

**Tranzakciók állapotának lekérdezése megtörtént. Frissítsd az oldalt!**

*11. ábra.* Flash üzenet a tranzakciók állapotának sikeres frissítéséről

Amennyiben az új státusz „Elfogadva” lesz, nincs további teendő. Elutasított és figyelmeztetésekkel elfogadott tranzakciók esetén az áttekintő lapon megjelenítésre kerülnek a hiba- vagy figyelmeztető üzenetek.

## Egyszerűsített számla

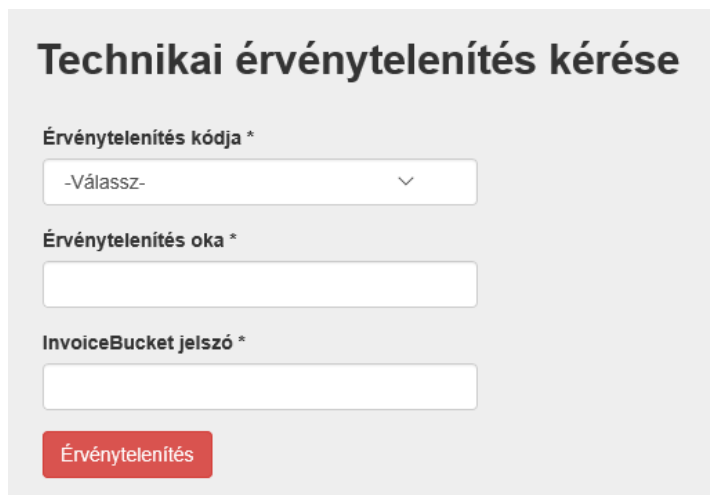
### Tranzakció #6 (Elfogadva figyelmeztetésekkel)

Figyelmeztetés: Egyszerűsített számla tételsor mennyiség és egységár szorzata a kedvezmény adatait figyelembe véve (ha releváns) eltér a tételsor bruttó értékétől.

Figyelmeztetés: Vevő adószáma nem létezik.

*12. ábra.* Figyelmeztető üzenetek egy tranzakció áttekintő lapján

Ha egy tranzakció státusza „Elfogadva” vagy „Elfogadva figyelmeztetésekkel”, az áttekintő lapjuk alján megjelenik egy űrlap, melynek kitöltésével kezdeményezhető a tranzakció technikai érvénytelenítése.



13. ábra. Érvénytelenítő űrlap

Ennek beküldése esetén az új státusz „Érvénytelenítve” lesz, valamint az érvénytelenítési kérelem továbbítva lesz az Online Számla felé. A válasz megérkezését követően az áttekintő lapra felkerül az érvénytelenítés azonosítója is.



14. ábra. Áttekintő lap teteje érvénytelenítést követően

Fontos megjegyezni azonban, hogy ezzel még a tulajdonképpeni technikai érvénytelenítés nem történik meg, csupán kezdeményezve lesz. Az érvénytelenítés véglegesítéséhez egy erre jogosult felhasználónak be kell lépnie az Online Számla webes felületére és jóvá kell hagyja a kérést, amint erre egy flash üzenet is felhívja a figyelmünk.

Az alkalmazás arra is lehetőséget ad, hogy a kapcsolt technikai felhasználó tulajdonosa által kiállított számlák és a számára kiállított számlák hatósági adatait letöltsük. Ezt a „Számla

adatainak lekérdezése” menüpontban tehetjük meg. Vevői számla lekérdezése esetén csupán a számlának a sorszámát kell megadni, szállítói számla esetén pedig a szállító törzsszámát és a számla sorszámát. A kérést ugyancsak a jelszavunkkal kell megerősíteni. Érvényes jelszó megadása esetén először egy */queryInvoiceCheck* típusú kérés történik, amely a számla meglétét kérdezi le. Ha az Online Számla rendszerben nem található számla a megadott paraméterekkel, erről egy felvillanó üzenetet kapunk.

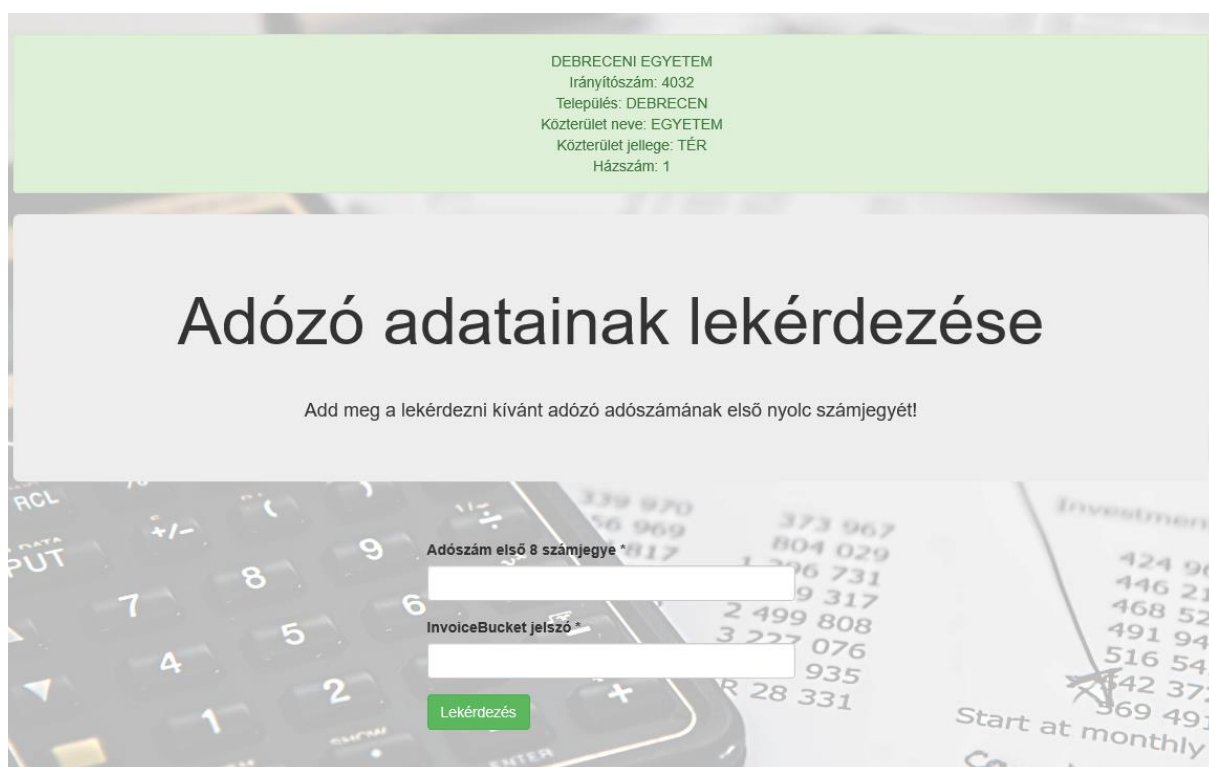
Nem létező számla

15. ábra. Flash üzenet érvénytelen számlaparaméterek esetén

Ha a */queryInvoiceCheck* igaz értékkel tér vissza, akkor ugyanezen adatokra megtörténik egy */queryInvoiceData* művelet is, ami már visszaadja az eredeti adatszolgáltatás adott számlához kapcsolódó *InvoiceData* XML elemét. Ezt követően az XML dokumentum tartalma egy ideiglenes fájlba íródik, majd letöltésre kerül.

16. ábra. Számlaadatok letöltése

Az *InvoiceBucket* utolsó funkciója az adózói adatok azonnali lekérdezése a */queryTaxpayer* művelet révén. Ehhez csupán a lekérdezni kívánt adóalany törzsszámát kell megadni és a jelszavunkkal megerősíteni a kérést. Érvényes adószám esetén az adózói adatok felvillanó üzenet formájában jelennek meg. (Nem számítanak érvényesnek a már visszavont adószámok, például felszámolás vagy végelszámolás miatt.)



DEBRECENI EGYETEM  
Irányítószám: 4032  
Település: DEBRECEN  
Községi név: EGYETEM  
Községi jellege: TÉR  
Házszám: 1

## Adózó adatainak lekérdezése

Add meg a lekérdezni kívánt adózó adószámának első nyolc számjegyét!

Adószám első 8 számjegye

InvoiceBucket jelszó \*

17. ábra. Adózói adatok sikeres lekérdezése

## 7. További fejlesztési lehetőségek

Az *InvoiceBucket* egyik fő hiányossága, hogy a regisztrálásnak nem feltétele egy email cím megadása és megerősítése, és ebből kifolyólag elfelejtett jelszavakat sincs lehetőség helyreállítani. Ezen továbbfejlesztések implementálásához a *nodemailer* nevű emailküldő npm-es csomag használatát javaslom. Éles használathoz egy adatvédelmi tájékoztató megírása is szükséges lesz.

Ahogy korábban már említettem, az egyszerűség kedvéért csak az egyszerűsített számlákról történhet adatszolgáltatás az alkalmazás jelenlegi állapotában. Komoly fejlesztési munkát igényel, ugyanakkor egy fontos továbbfejlesztés lenne a normál- és gyűjtőszámlák, valamint az egyes számlakategóriákon belül a módosító és storno számlák támogatása. Ehhez szükséges minden típusú számlára egyedi EJS sablonok és *InvoiceData* törzset generáló függvények létrehozása, a megfelelő útvonalak logikáinak beállítása, sőt talán a teljes felhasználó felület újragondolása is, mivel a normál- és gyűjtőszámlák esetén sokkal több mezőt kell kitölteni.

Kényelmi funkcióként lehetne egy „tranzakció másolása” lehetőséget létrehozni az irányítópulton arra az esetre, ha valaki több olyan számláról is szeretne adatot szolgáltatni, amelyek hasonlóak, csupán néhány pontban térnek el egymástól. Ugyancsak jó ötlet lehet egy új adatséma létrehozása az ügyféladatok számára, amelyek így minden mentés alkalmával egy külön adatbázis kollekcióba kerülnek tárolásra. A korábban megadott ügyfelek pár gombnyomásra való előhívása így lehetővé válna a vevői adatok kitöltésekor.

Ha korábban kiállított számlák adatait kérdezzük vissza, azokat jelenleg egy XML állományban kapjuk meg, ami nem biztos, hogy minden felhasználó számára egy kényelmes megoldás. Ennek a problémának egy lehetséges orvoslása PDF formátumú számlasablonok létrehozása, amelyeknek kinézete hasonlít a való életben használatos papír alapú számlákhoz. PDF-ek készítéséhez a *pdfkit* vagy a *pdfmake* kiegészítők egyikének használatát érdemes megfontolni.

Bár a technikai felhasználók adatainak tárolása viszonylag biztonságosnak mondható, ennek a biztonságának az ára az, hogy minden Online Számlával való kapcsolatfelvételhez szükséges a jelszó beírása. Bár erre lehet egy megoldás, ha a felhasználó megjegyezteti a jelszavát a böngészővel, ez mégsem teljesen kézenfekvő. Ha sikerülne egy másik, külső inputot nem igénylő titkosítási módszert kitalálni a technikai adatok biztonságos tárolására, egy olyan

ütemezett feladat is létrehozható lenne, ami időközönként végigmegy az összes „Aláírva” státuszú tranzakción és automatikusan lekérdezi azok feldolgozottságát.

Az előbbihez kapcsolódóan megjegyezném, hogy az *InvoiceBucket* nem használja ki a */queryInvoiceDigest* operáció nyújtotta lehetőségeket. Ha sikerülne az előbb említett módon automatikusan hozzáférhetővé tenni a technikai felhasználók adatait, akkor ezen operációt kihasználva egy automatizált folyamat időközönként leellenőrizhetné, hogy a legutolsó keresés óta került-e kiállításra számla a kapcsolt technikai felhasználót kiadó számára. Amennyiben igen, erről lehetne egy email értesítést küldeni egy felhasználó által megadott email címre.

## 8. Összefoglalás

Úgy gondolom, hogy az előző fejezetben említett hiányosságoktól eltekintve sikerült megvalósítanom a dolgozatom elején kitűzött célokat. Az *InvoiceBucket* webes alkalmazás egy kivételével az Online Számla mindegyik műveletét képes végrehajtani REST API hívások segítségével, a felhasználói profilokat illetően sikerült beépíteni a technikai felhasználók kapcsolásának és jelszó módosításának lehetőségét. Az alkalmazásban korábban létrehozott tranzakciók visszanezethetők, piszkozatként menthetők és feldolgozottságuk helyileg ellenőrizhető. Javaslatokat is adtam, ami a későbbi lehetséges fejlesztési irányokat illeti.

A projekt megvalósítása szakmailag sokat nyújtott számomra, e téren beváltotta a hozzá fűzött reményeket. Kezdődött egy átfogó webfejlesztési kurzus elvégzésével, amely során megismerkedhettem a HTML, CSS, Bootstrap, JavaScript, jQuery, Node.js, Express, EJS és MongoDB technológiák alapjaival, valamint olyan módszerekkel, amikkel az egyes webes technológiákat összerakva egy komplex, dinamikus és forráskódok szempontjából könnyen áttekinthető webalkalmazást tudunk fejleszteni. Ezt követte az Online Számla rendszer működésének megértése, amely hosszú dokumentáció tanulmányozást és sok kísérletezést igényelt. Ekkor szembesültem a következő akadállyal, ami a Node.js-nek azon tulajdonsága, hogy számos függvényének végrehajtása aszinkron módon történik. Megértve, hogy itt nem működik az eddig tanult lineáris fejlesztési látásmód, elkezdtem tanulmányozni és gyakorolni a callback függvények és promise-ok helyes használatát.

Az Online Számla API interfészének működését, a kérések felépítését is sikerült jobban megértsem az XML törzseket felépítő függvények megírása és a próbahívások futtatása alatt. Ezen ismereteknek a gyakorlatban is hasznát veszem, jelenleg munkahelyi környezetben is dolgozok hasonló, Online Számla rendszerrel összefüggő projekteken.

Remélem, hogy dolgozatom mindazok számára hasznosnak bizonyul, akik szeretnének egy gyors betekintést nyerni a NAV-nak történő hatósági adatszolgáltatások, valamint a webes alkalmazások fejlesztésének világába. Az *InvoiceBucket* alkalmazás forráskódja, amint azt korábban is említettem, bárki által megtekinthető, felhasználható és továbbfejleszthető.



## 9. Irodalomjegyzék

[1] *Neten a Hivatal* projekt: A közigazgatási rendszer bemutatása

<http://kozigazgatas.netenahivatal.gov.hu/a-kozigazgatasi-rendszer-bemutatasa>

(hozzáférés dátuma: 2020.02.04.)

[2] Online Számla hivatalos közlemények oldala

<https://onlineszamla.nav.gov.hu/home>

(hozzáférés dátuma: 2020.02.04.)

[3] 2007. évi CXXVII. törvény az általános forgalmi adóról

<https://net.jogtar.hu/jogszabaly?docid=a0700127.tv>

(hozzáférés dátuma: 2020.02.04.)

[4] 23/2014. (VI. 30.) NGM rendelet a számla és a nyugta adóigazgatási azonosításáról, valamint az elektronikus formában megőrzött számlák adóhatósági ellenőrzéséről

[http://njt.hu/cgi\\_bin/njt\\_doc.cgi?docid=170220](http://njt.hu/cgi_bin/njt_doc.cgi?docid=170220)

(hozzáférés dátuma: 2020.02.04.)

[5] A Nemzeti Adó- és Vámhivatal által kiadott 3006/2018. útmutató az állami adó- és vámhatóságnak az adózás rendjéről szóló 2017. évi CL. törvény, az adóigazgatási rendtartásról szóló 2017. évi CLI. törvény, valamint az adóhatóság által fogantatosítandó végrehajtási eljárásokról szóló 2017. évi CLIII. törvény alapján történő bírságolási gyakorlatáról

[https://www.nav.gov.hu/data/cms479361/3006\\_2018.pdf](https://www.nav.gov.hu/data/cms479361/3006_2018.pdf)

(hozzáférés dátuma: 2020.02.04.)

[6] Tamásné Szabó Zsuzsanna: 2017-től jön a NAV-hoz bekötött kötelező online számlázás – 24.hu, 2015.11.27.

<https://24.hu/fn/gazdasag/2015/11/27/2017-tol-jon-a-nav-hoz-bekotott-kotelezo-online-szamlazas/>

(hozzáférés dátuma: 2020.02.04.)

[7] Egy év, és jön az online számla – MTI, 2017.04.29.

<https://hirado.hu/2017/04/29/egy-ev-es-jon-az-online-szamla/>

(hozzáférés dátuma: 2020.02.04.)

[8] Csiki Gergely: Jön a NAV-testvér, megjelent az NGM fontos rendelete – Portfolio, 2017.06.28.

<https://www.portfolio.hu/gazdasag/20170628/jon-a-nav-testver-megjelent-az-ngm-fontos-rendelete-254987>

(hozzáférés dátuma: 2020.02.04.)

[9] Így adózunk 2020-ban: olyan adóváltozás jön, amelyen még soha – Portfolio, 2020.01.01.

<https://www.portfolio.hu/gazdasag/20200101/igy-adozunk-2020-ban-olyan-adovaltozas-jon-amilyen-meg-soha-411279>

(hozzáférés dátuma: 2020.02.04.)

[10] Online Számla regisztrációs tájékoztató

[https://onlineszamla.nav.gov.hu/tajekoztatas\\_a\\_regisztraciorol](https://onlineszamla.nav.gov.hu/tajekoztatas_a_regisztraciorol)

(hozzáférés dátuma: 2020.02.04.)

[11] Nemzeti Adó- és Vámhivatal: Online Számla Rendszer – Felhasználói kézikönyv

[https://onlineszamla.nav.gov.hu/api/files/container/download/Online%20Sz%C3%A1mla%20Rendszer\\_felhaszn%C3%A1l%C3%B3i\\_k%C3%A9zik%C3%B6nyv\\_v1.8.pdf](https://onlineszamla.nav.gov.hu/api/files/container/download/Online%20Sz%C3%A1mla%20Rendszer_felhaszn%C3%A1l%C3%B3i_k%C3%A9zik%C3%B6nyv_v1.8.pdf)

(hozzáférés dátuma: 2020.02.04.)

[12] Nemzeti Adó- és Vámhivatal: NAV Online Számla Rendszer – Számlaadat-szolgáltatás REST API interfészleírás és fejlesztői dokumentáció

[https://onlineszamla-test.nav.gov.hu/api/files/container/download/Online%20Szamla\\_interfesz%20specifik%C3%A1ci%C3%B3\\_HU\\_v2.0.pdf](https://onlineszamla-test.nav.gov.hu/api/files/container/download/Online%20Szamla_interfesz%20specifik%C3%A1ci%C3%B3_HU_v2.0.pdf)

(hozzáférés dátuma: 2020.02.04.)

[13] The Web Developer Bootcamp – online webfejlesztői kurzus, oktató: Colt Steele.

Megvásárolható: <https://www.udemy.com/course/the-web-developer-bootcamp/>

(hozzáférés dátuma: 2020.02.05.)

[14] Mayowa Adegbola: How to build a simple session-based authentication system with NodeJS from scratch – Codementor, 2017.04.07.

<https://www.codementor.io/@mayowa.a/how-to-build-a-simple-session-based-authentication-system-with-nodejs-from-scratch-6vn67mcy3>

(hozzáférés dátuma: 2020.05.02.)

- [15] Tyler McGinnis: Async JavaScript. From Callbacks, to Promises, to Async/Await  
<https://tylermcginnis.com/async-javascript-from-callbacks-to-promises-to-async-await/>  
(hozzáférés dátuma: 2020.05.02.)
- [16] What is MongoDB? – Hivatalos MongoDB bemutató anyag  
<https://www.mongodb.com/what-is-mongodb>  
(hozzáférés dátuma: 2020.05.02.)
- [17] Node.js – Wikipedia  
<https://en.wikipedia.org/wiki/Node.js>  
(hozzáférés dátuma: 2020.05.02.)
- [18] Stack Overflow: Developer Survey Results 2019  
<https://insights.stackoverflow.com/survey/2019>  
(hozzáférés dátuma: 2020.05.02.)
- [19] Express hivatalos dokumentáció  
<http://expressjs.com/>  
(hozzáférés dátuma: 2020.05.02.)
- [20] Express-session kódtárház és használati útmutató  
<https://github.com/expressjs/session>  
(hozzáférés dátuma: 2020.05.02.)
- [21] Body-parser útmutató - NPM  
<https://www.npmjs.com/package/body-parser>  
(hozzáférés dátuma: 2020.05.02.)
- [22] Method-override útmutató - NPM  
<https://www.npmjs.com/package/method-override>  
(hozzáférés dátuma: 2020.05.02.)
- [23] Mongoose hivatalos dokumentáció  
<https://mongoosejs.com/docs/>  
(hozzáférés dátuma: 2020.05.02.)

[24] Request útmutató - NPM

<https://www.npmjs.com/package/request>

(hozzáférés dátuma: 2020.05.02.)

[25] Xml-js útmutató - NPM

<https://www.npmjs.com/package/xml-js>

(hozzáférés dátuma: 2020.05.03.)

[26] Full-icu útmutató - NPM

<https://www.npmjs.com/package/full-icu>

(hozzáférés dátuma: 2020.05.03.)

[27] EJS hivatalos dokumentáció

<https://ejs.co/>

(hozzáférés dátuma: 2020.05.03.)

[28] Connect-flash útmutató - NPM

<https://www.npmjs.com/package/connect-flash>

(hozzáférés dátuma: 2020.05.03.)

[29] Js-sha512 útmutató - NPM

<https://www.npmjs.com/package/js-sha512>

(hozzáférés dátuma: 2020.05.03.)

[30] Js-sha3 útmutató - NPM

<https://www.npmjs.com/package/js-sha3>

(hozzáférés dátuma: 2020.05.03.)

[31] Js-base64 útmutató - NPM

<https://www.npmjs.com/package/js-base64>

(hozzáférés dátuma: 2020.05.03.)

[32] Aes-ecb útmutató - NPM

<https://www.npmjs.com/package/js-base64>

(hozzáférés dátuma: 2020.05.03.)

[33] Bootstrap 3.3 hivatalos dokumentáció

<https://getbootstrap.com/docs/3.3/>

(hozzáférés dátuma: 2020.05.03.)

[34] CodersRank: Most Popular Repositories | 2012-2019 – Youtube, 2019.11.14.

<https://www.youtube.com/watch?v=2vAk8h7nZGs>

(hozzáférés dátuma: 2020.05.03.)

[35] Gitstar Ranking

<https://gitstar-ranking.com/>

(hozzáférés dátuma: 2020.05.03.)

[36] jQuery – Wikipedia

<https://en.wikipedia.org/wiki/JQuery>

(hozzáférés dátuma: 2020.05.03.)

[37] BootstrapValidator hivatalos dokumentáció

<http://bootstrapvalidator.votintsev.ru/>

(hozzáférés dátuma: 2020.05.03.)