Mérési jegyzőkönyv – Szoftver Laboratórium 5

4. mérés: SOA

|  |  |
| --- | --- |
| Név: | **Szabó Bence Farkas** |
| Neptun kód: | **RF57V5** |
| Feladat kódja: | **26 – SZALL** |
| Mérésvezető neve: | **Böjti Bence** |
| Mérés időpontja: | **2017-04-06 16:15** |
| Mérés helyszíne: | **R4K** |
| A működő alkalmazás elérhetősége: |  |
| Megoldott feladatok: | **1,2,3,4,5,6,** |
| Elérhető pontszám (plusz pontok nélkül): | **40p** |

1. Mérési feladatok megoldása
   1. feladat: kötelező REST szolgáltatás készítése

A megoldáshoz használt SQL utasítás

*Ha a feladat megoldásához szükség volt SQL utasításra, azt itt tüntessük fel, a Kód stílus segítségével!*

1. lekérdezés:

SELECT order\_id, description, quantity, deadline\_date FROM orders

2. lekérdezés:

’SELECT description, vehicle\_type, quantity, origin, destination,

order\_date, deadline\_date, comment\_text

FROM orders

WHERE order\_id = :order\_id’, order\_id=order\_id

Magyarázat

Az első feladatban két egyszerű Python kódot kellett írni, amelyeknek a „/orders.json” és a „/orders/<order\_id>.json” címeknél kellett a megfelelő adatokat kilistázniuk, JSON formátumban. A megoldáshoz a „@app.route(„/orders.json”)” és az „@app.route(„/orders/<order\_id>.json”)” kezdetű metódusok tartoznak.

Elsőként a megrendeléseket lekérdező metódust valósítottam meg. Az adatok eléréséhez a fentebb láthadó 1. lekérdezéssel jelzett SQL parancsot használtam.

Ennek az SQL lekérdezésnek a segítségével és a cur.execute() metódussal megkapjuk az eredménytáblát, amelynek a sorait egy JSON tömbbe másoljuk, melynek sorai szótárak. Ezt a tömböt adjuk vissza. A dátum tárolásához először sima, időpont nélküli dátummá konvertáljuk a date() metódussal, majd ezt az isoformat()-tal a kellő formátumra hozzuk.

Az adott order\_id-jű megrendelés adatait lekérdező metódust a 2. lekérdezéssel valósítottam meg. Ha ennek eredménytáblája üres (None) akkor a felhasználónak abort(404) üzenetet adunk. Ha nem üres, a lekért adatokat az előző metódusnál látott módon JSON formátumra hozzuk. Itt nem kell tömböt visszaadni, mivel csak egyetlen megrendelés adatait adjuk vissza.

Tesztelés menete

/orders.json tesztelése:

$ curl localhost:20918/orders.json

{

"orders": [

{

"deadline\_date": "2017-05-10",

"description": "bútor",

"order\_id": "2017/000001",

"quantity": 10000

},

{

"deadline\_date": "2017-04-11",

"description": "építőanyag",

"order\_id": "2017/000002",

"quantity": 8000

},

{

"deadline\_date": "2017-03-21",

"description": "építőanyag",

"order\_id": "2017/000003",

"quantity": 20000

},

{

"deadline\_date": "2017-04-12",

"description": "élelmiszer <tartós>",

"order\_id": "2017/000004",

"quantity": 15000

},

{

"deadline\_date": "2017-02-22",

"description": "bútoralapanyag",

"order\_id": "2017/000005",

"quantity": 30000

},

{

"deadline\_date": "2017-05-01",

"description": "élelmiszer <ital>",

"order\_id": "2017/000006",

"quantity": 20000

},

{

"deadline\_date": "2017-02-10",

"description": "műszaki cikk",

"order\_id": "2017/000011",

"quantity": 20000

},

{

"deadline\_date": "2017-03-08",

"description": "papíráru, <nyers>",

"order\_id": "2017/000012",

"quantity": 34000

},

{

"deadline\_date": "2017-04-10",

"description": "vegyi áru (festék&higító)",

"order\_id": "2017/000013",

"quantity": 12000

},

{

"deadline\_date": "2017-03-16",

"description": "üzemanyag",

"order\_id": "2017/000021",

"quantity": 16000

},

{

"deadline\_date": "2017-03-12",

"description": "üzemanyag",

"order\_id": "2017/000022",

"quantity": 42000

},

{

"deadline\_date": "2017-04-12",

"description": "baromfi",

"order\_id": "2017/000031",

"quantity": 16000

},

{

"deadline\_date": "2017-02-28",

"description": "versenyló",

"order\_id": "2017/000032",

"quantity": 4000

},

{

"deadline\_date": "2017-04-13",

"description": "sertés",

"order\_id": "2017/000033",

"quantity": 20000

},

{

"deadline\_date": "2017-04-24",

"description": "használt autó",

"order\_id": "2017/000041",

"quantity": 10

},

{

"deadline\_date": "2017-03-12",

"description": "hatóságilag lefoglalt járművek",

"order\_id": "2017/000042",

"quantity": 15

}

]

}

/orders/<order\_id>.json tesztelése:

$ curl localhost:20918/orders/2017/000001.json

{

"comment\_text": null,

"deadline\_date": "2017-05-10",

"description": "bútor",

"destination": "Budapest (HU)",

"destination\_currency": "HUF",

"order\_date": "2017-01-02",

"origin": "Berlin (DE)",

"origin\_currency": "EUR",

"quantity": 10000,

"vehicle\_type": "D"

}

* 1. feladat: kötelező REST fogyasztó készítése
     1. A megoldáshoz használt SQL utasítás
     2. Magyarázat

1. A második feladatnál a Request távoli szolgáltatás-elérő funkciót kellett használni. Az adott ország valutáját az alábbi linken tudtam lekérdezni: http://rapid.eik.bme.hu:9080/currency\_ws/currencies/<country>.json, ahol a „country” az adott ország hivatalos országkódja. Ezt az előző feladat második metódusából, a lekérdezés eredményeként kapott response változójából kapjuk meg, melyből kiolvassuk a forrás és cél nevét. Ez tartalmazza az ország azonosítóját is, melyet kivágunk a string-ből. Ezt adjuk hozzá a GET metódus által használt url-hez. Így már lekérdezhető az országok valutája, melyet az előző feladat második metódusának kiegészítéseként adunk hozzá a visszaadott eredményhez.

Tesztelés menete

1. $ curl http://rapid.eik.bme.hu:9080/currency\_ws/currencies/hu.json
2. {
3. "country": "en"
4. "currency": "HUF"
5. }
6. Példaadatok
7. *Ha a feladat teszteléséhez szükség van példaadatokra, azt itt tüntessük fel*

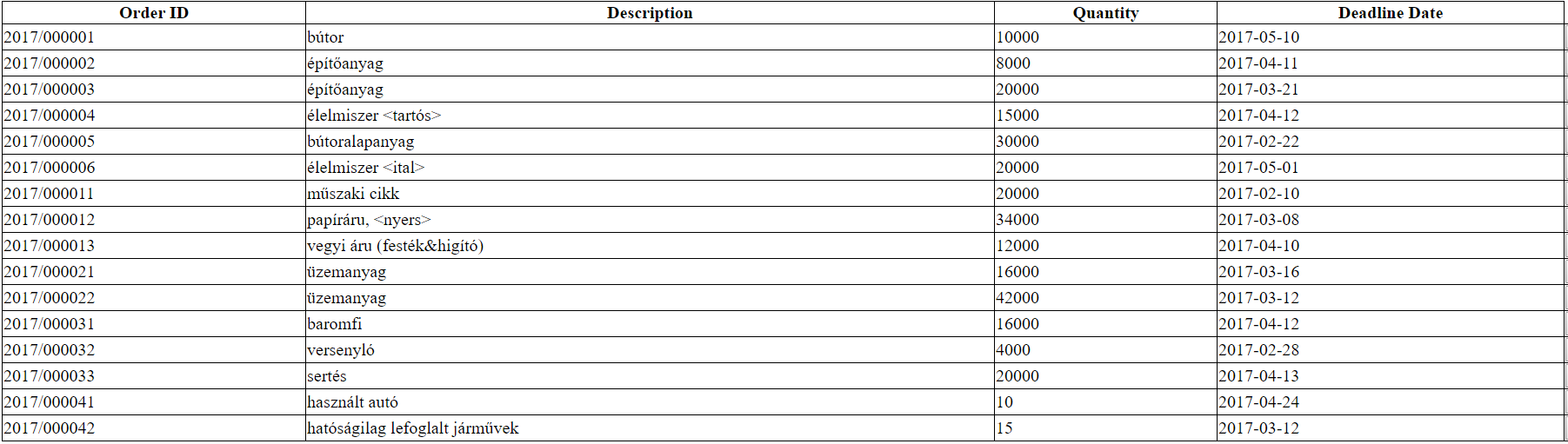
3. feladat: kötelező AJAX megoldás készítése

* + 1. Magyarázat

A harmadik feladatnál egy statikus .html weblapot kellett készíteni, amely tartalmazta a segédlet összes $-al kapcsolatos funkcióját. A weblap betöltődésekor a loadData() metódus hívódik meg, mely egy $.ajax aszinkron hívással a „../orders.json” címről lekérdezi a táblázatba az adatokat, majd az orders\_into\_table() metódussal lehet ezeket az adatokat a táblázatba felvenni. Ezek mind <script> és a </script> közé kerültek, amelyet a <body></body>-ba tettem. Ezen kívül még felvettem az oldalra a táblázatot, „orders” ID-vel, hogy hivatkozni lehessen rá, és felvettem a táblázat fejlécét. A táblázatot kicsit kiszépítettem, hogy vonalakkal legyenek a cellák elválasztva.

Tesztelés menete

<http://localhost:20918/static/orders.html>



4. feladat: választható REST szolgáltatás bővítés

* + 1. A megoldáshoz használt SQL utasítás

1. lekérdezés

"SELECT order\_id, description, quantity, deadline\_date FROM orders

WHERE description LIKE :description ESCAPE '@'", description=description"

2. lekérdezés

SELECT order\_id, description, quantity, deadline\_date FROM orders WHERE vehicle\_type

LIKE :vehicle\_type ESCAPE '@'", vehicle\_type=vehicle\_type

* + 1. Magyarázat

A negyedik feladathoz két Python kódot írtam meg, melyek a „@app.route("/orders/by\_description/<desc\_pfx>.json")” és az „@app.route("/orders/by\_type/<type>.json")” kezdetű metódusokhoz tartoznak. A kettő igencsak hasonló volt, mivel mind a kettő egy-egy paraméteres SQL lekérdezést hajtottak végre, és ez a paraméter a keresendő szó volt. A megadott paramétert ahhoz, hogy a speciális karakterek kereshetőségét megteremtsem, a Python replace() string-metódusával változtattam meg, minden speciális karakter elé egy „@” jel került, majd az SQL lekérdezést annyival módosítottam, hogy a „@” jelet „escape-elje”, vagyis a mögötte lévő karakter karakterként értelmeződjön. Így a fent látható paraméteres SQL lekérdezéseket használtam. Ahhoz, hogy a csonkolt szavak is kereshetők legyen, a keresendő szavakat kiegészítettem: az elejére és végére is egy-egy „%” karaktert került, így már bármelyik olyan szót, amely tartalmazza a keresett kifejezést, visszaadja a kereső. Ha az SQL lekérdezés által adott eredménytábla üres (None), akkor 404-es kódú hibaüzenetet adunk a felhasználónak, amiből tudja, hogy a keresett szöveg nem szerepel a táblázat adott oszlopában. A választ az első feladathoz hasonló JSON formátumban adjuk vissza.

Tesztelés menete

Láthatjuk, hogy a ’b’-re keresve kilistázza az összes megrendelést, melynek leírásában szerepel b-betű:

1. $ curl <http://localhost:20918/orders/by_description/b.json>
2. {
3. "orders": [
4. {
5. "deadline\_date": "2017-05-10",
6. "description": "bútor",
7. "order\_id": "2017/000001",
8. "quantity": 10000
9. },
10. {
11. "deadline\_date": "2017-02-22",
12. "description": "bútoralapanyag",
13. "order\_id": "2017/000005",
14. "quantity": 30000
15. },
16. {
17. "deadline\_date": "2017-04-12",
18. "description": "baromfi",
19. "order\_id": "2017/000031",
20. "quantity": 16000
21. }
22. ]
23. }

Hasonlóan működik ez a jármű típusánál is:

1. & curl <http://localhost:20918/orders/by_type/D.json>
2. {
3. "orders": [
4. {
5. "deadline\_date": "2017-05-10",
6. "description": "bútor",
7. "order\_id": "2017/000001",
8. "quantity": 10000
9. },
10. {
11. "deadline\_date": "2017-04-11",
12. "description": "építőanyag",
13. "order\_id": "2017/000002",
14. "quantity": 8000
15. },
16. {
17. "deadline\_date": "2017-03-21",
18. "description": "építőanyag",
19. "order\_id": "2017/000003",
20. "quantity": 20000
21. },
22. {
23. "deadline\_date": "2017-04-12",
24. "description": "élelmiszer <tartós>",
25. "order\_id": "2017/000004",
26. "quantity": 15000
27. },
28. {
29. "deadline\_date": "2017-02-22",
30. "description": "bútoralapanyag",
31. "order\_id": "2017/000005",
32. "quantity": 30000
33. },
34. {
35. "deadline\_date": "2017-05-01",
36. "description": "élelmiszer <ital>",
37. "order\_id": "2017/000006",
38. "quantity": 20000
39. }
40. ]
41. }

Illetve a % karakterre való kereséssel nem listáz ki mindent a kérés:

1. & curl [http://localhost:20918/orders/by\_type/%.json](http://localhost:20918/orders/by_type/%25.json)
2. {
3. "orders": []
4. }

5. feladat: választható REST szolgáltatás bővítés

* + 1. A megoldáshoz használt SQL utasítás
    2. Magyarázat

1. Az ötödik feladatot a DELETE verb megírásával kezdtem. A service.py –ba vettem fel egy új metódust, amely akkor hívódik meg, hogyha a „/orders/<order\_id>” címen szeretnének valamilyen műveletet végrehajtani, a megengedett műveletek: DELETE és PUT.
2. A egy vizsgálattal kezdem, mellyel megnézem hogy létezik-e az adott order\_id-val rendelkező megrendelés az adatbázisban. Ha nem akkor 404-es hibával jelezzük a felhasználónak, ha pedig igen, akkor a megadott utasítástól függően hajtsa végre a DELETE vagy PUT szálat. A DELETE szál SQL utasítása:

"DELETE FROM orders WHERE order\_id = :order\_id", order\_id=order\_id

Ha nem sikerült a törlés, akkor 500-as kódú hibát dob a program.

A PUT művelethez az alábbi SQL kódot használtam fel:

"UPDATE orders

SET description = :description,

vehicle\_type = :vehicle\_type,

quantity = :quantity,

origin = :origin,

destination = :destination,

order\_date = TO\_DATE(:order\_date, 'yyyy-MM-dd'),

deadline\_date = TO\_DATE(:deadline\_date, 'yyyy-MM-dd'),

comment\_text = :comment\_text

WHERE order\_id = :order\_id",

description=data['description'],

vehicle\_type=data['vehicle\_type'],

quantity=data['quantity'],

origin=data['origin'],

destination=data['destination'],

order\_date=data['order\_date'],

deadline\_date=data['deadline\_date'],

comment\_text=data['comment\_text'],

order\_id=order\_id)

Ha nem sikerült a beillesztés, 500-as kódú hibát dob a program.

A POST művelethez egy külön metódust írtam meg, amely az „/orders.json” hivatkozásakor elérhető. A metódus részletes magyarázata megtalálható a service.py file kommentjei között, itt csak a lényegesebb dolgokra térek ki. Ahhoz, hogy az újonnan felvett eszközhöz azonosítót tudjon rendelni a függvény, generálni kell egy ilyen azonosítót. Ez a generate\_order\_id() metódus feladata, amely végigiterál az azonosítókon, és a legnagyobbhoz hozzáadva egyet, megteremti az új, egyedi azonosítót. Miután ez megvan, már csak végre kell hajtani az alábbi SQL utasítást:

"INSERT INTO orders

VALUES

(:order\_id,

:description,

:vehicle\_type,

:quantity, :origin,

:destination,

TO\_DATE(:order\_date, 'yyyy-MM-dd')),

TO\_DATE(:deadline\_date, 'yyyy-MM-dd')),

:comment\_text",

order\_id=order\_id,

description=data['description'],

vehicle\_type=data['vehicle\_type'],

quantity=data['quantity'],

origin=data['origin'],

destination=data['destination'],

order\_date=data['order\_date'],

deadline\_date=data['deadline\_date'],

comment\_text=data['comment\_text'])

Tesztelés menete

1. $ curl –request DELETE –header „Content-Type: application/json”localhost:20918/orders.json

Példaadatok

*Ha a feladat teszteléséhez szükség van példaadatokra, azt itt tüntessük fel*

6. feladat: választható AJAX megoldás bővítés

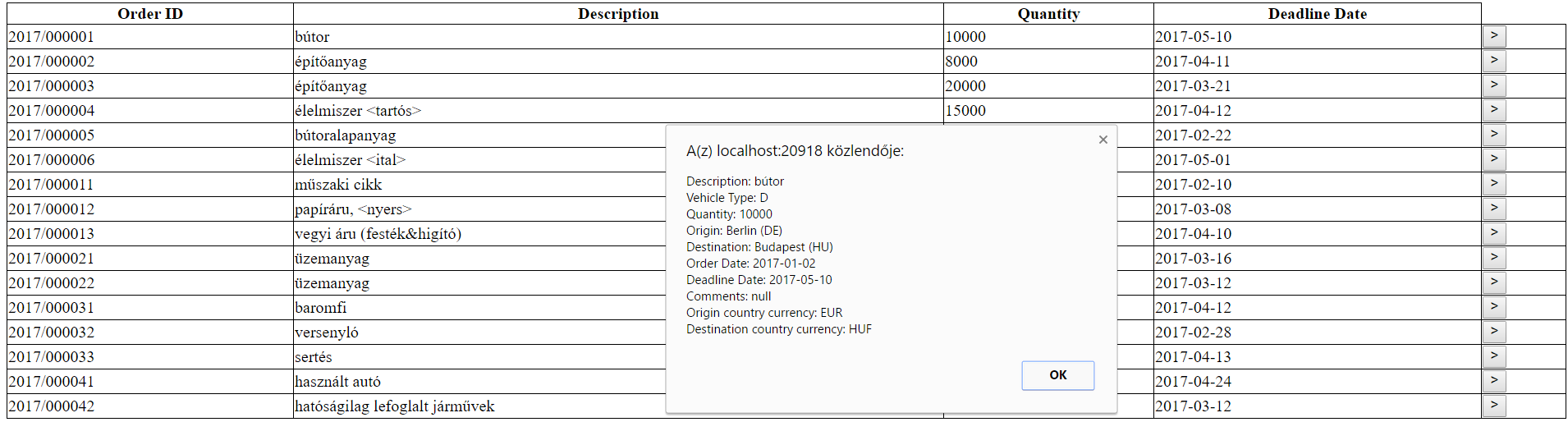
* + 1. Magyarázat

1. A hatodik feladathoz az orders.html oldalt kellett kiegészíteni gombokkal, valamint az adatok frissítésével. Először a gombokat vettem fel a táblába, az ajaxdemo.html file mintájára, kiegészítve az ID megadásával a táblafeltöltő-ciklusban. A gombok nevének a sorhoz tartozó megrendelés order\_id mezőjét adtam meg, hogy arra később könnyedén hivatkozhassak. A gomb clicked() metódusát a úgy írtam meg, hogy egy $.ajax() lekérdezéssel a korábban már megírt „orders/order\_id.json” paraméteres oldalról kérje le az adatokat, és ha ez sikeres volt, akkor azokat egy alert ablakban jelenítse meg. A $.ajax()-ban úgy adtam meg a megrendelés azonosítóját, hogy lekérdeztem a megnyomott button nevét.
2. Ez után következett az adatok frissítése. Ehhez a window.setInterval() metódust használtam, ami a paraméterként megadott függvényt a (másik) paraméterként megadott időközönként meghívja. Ez a metódus ismét a loadData() volt, mivel ez tölti be az adatokat. A success() függvényét még ki kellett egészíteni, hogy a már meglévő sorokat törölje ki, azokra már nem lesz szükség. Ezt egy for ciklussal oldottam meg. Ahhoz hogy az eredeti, még frissítés előtti, sorszámot eltároljam, felvettem egy globális tableData változót.
3. A változás jelzéséhez összehasonlítottam a tábla frissítés előtti és frissítés utáni változatát a compare() függvénnyel. Ez a függvény ellenőrzi, hogy a különbözik e a tábláknak bármely cellája. Ha valahol eltérést találunk, akkor az eltérésnek megfelelően alert()-et küldünk a felhasználónak a változásokról.

Tesztelés menete

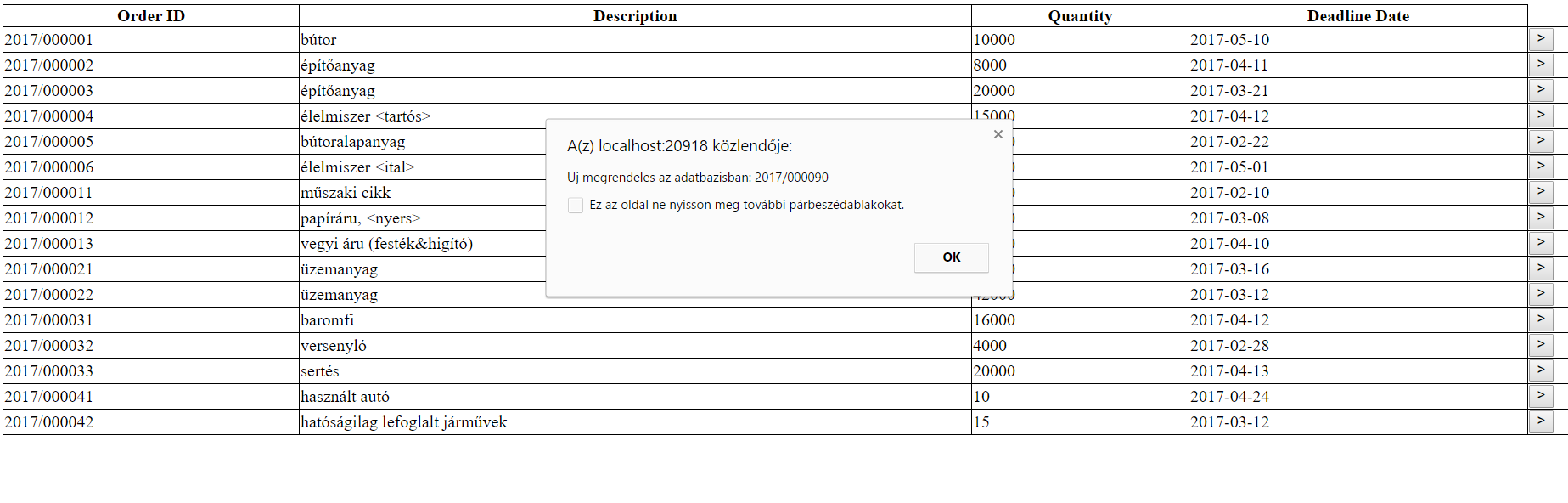
Az weblapon lévő gombok működése az első képen, míg az oldal frissítését végző metódusok üzenetei a további képeken láthatók.

[*http://localhost:20918/static/orders.html*](http://localhost:20918/static/orders.html)



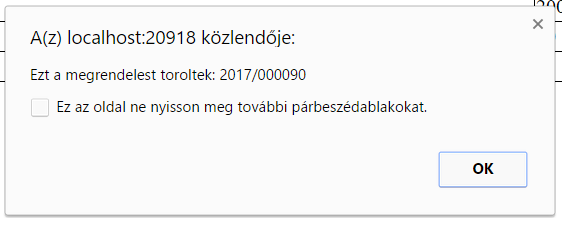
*Az oldal egy gomb megnyomása után.*

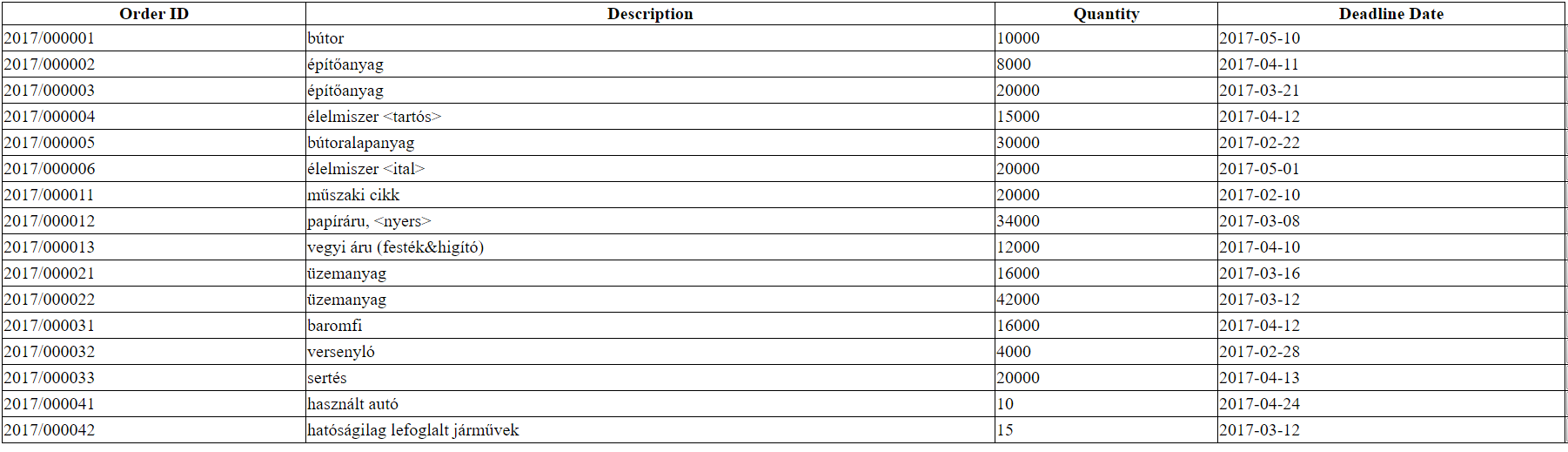
A teszteléshez az SQL developer-ben módosítottam az adatbázis tartalmát, melynek hatására az alábbi eredmények láthatók.

**

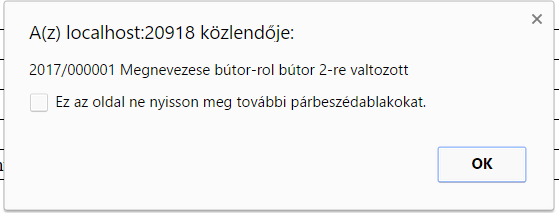
**

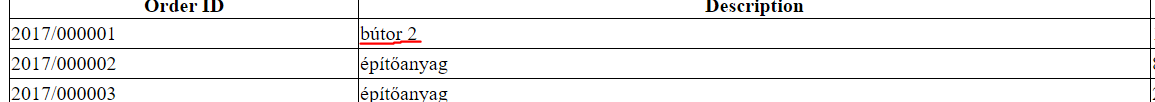
*Az első képen az oldal üzenete látható insert után, a második képen pedig az új megrendelés a táblázatban.*

**

**

*Az első képen az oldal üzenete delete után, a másodikon a már megváltozott táblázat látható.*

**

**

*Az első képen az oldal üzenete update után, a másodikon a megváltoztatott adatok látszódnak.*

1. Vélemény(ek) a mérésről

*Vélemény, építő jellegű kritika.*