

Tangram REST API

1	Technické údaje.....	2
1.1	Verzia.....	2
1.2	Autor.....	2
1.3	Online dokumentácia.....	2
1.4	Úvod.....	2
1.5	Podporované dátové typy.....	2
1.6	URL/URI.....	2
1.7	Request headers.....	2
1.8	Response OK (HTTP Status 200).....	3
1.9	Response BROWSER (HTTP Status 200 pre zobrazenie tabuľkových dát).....	3
1.10	Response ERROR (HTTP Status 400 Bad Request).....	3
1.11	Response SYSTEM ERROR (HTTP Status 500 a podobné).....	3
2	Security – protokol HTTPS (šifrovanie).....	4
3	Autentifikácia (meno/heslo).....	5
3.1	Nastavenie systému (Používatelia, Roly, Práva).....	5
3.2	Autentifikácia Basic (meno a heslo).....	5
3.3	Autentifikácia Bearer (token).....	6
4	Prístupové práva.....	7
4.1	Použitie vo web-aplikáciách.....	7
5	BROWSER.....	8
5.1	Úvod.....	8
5.2	Request.....	8
5.3	Response.....	9
6	VENDING (predajné automaty).....	11
6.1	Úvod.....	11
6.2	Autentifikácia.....	11
6.3	Vyhľadanie karty.....	11
6.4	Zápis transakcie.....	11

1 Technické údaje

1.1 Verzia

Táto dokumentácia sa vzťahuje na verziu API **23.02**

1.2 Autor

Tangram SW s.r.o., Chalupkova 14, Presov, Slovakia, www.tangram.sk

Ivan Fedorko, fedorko@tangram.sk

Posledný update: **04.04.2023**

1.3 Online dokumentácia

Dokumentácia k API je dostupná cez web-browser po nainštalovaní a spustení API servera na adrese:

`http://localhost:port/swagger-ui.html`

(adresu servera a číslo portu doplňte podľa skutočnosti)

V browseri sa objaví štandardné rozhranie Swagger s podrobným popisom requestov. Okrem zobrazenia dokumentácie umožňuje aj testovať requesty nad reálnymi dátami v pripojenej databáze.

1.4 Úvod

Tangram API je realizované ako RESTful-API cez HTTP alebo HTTPS. Použitý framework **Spring Boot** a HTTP server **Apache Tomcat** (embedded).

Import z API je vždy cez príkaz GET, dáta sú vrátené ako Response Body v JSON štruktúre. Prázdne hodnoty sa nemusia exportovať, klient by to nemal ohlásiť ako chybu.

Export do API je vždy cez príkaz POST, dáta sú v Request Body v **JSON** štruktúre. Prázdne hodnoty sa nemusia exportovať, API to nebude hlásiť ako chybu. Povinné položky sa musia vždy vyplniť neprázdnu hodnotou.

Character set pre dáta je vždy **UTF-8**.

Úspešne vykonaný request vracia HTTP-status OK (200). Telo response vtedy obsahuje odpoveď v JSON formáte.

Chybný request vracia HTTP status 4xx pre chyby klienta a 500 pre systémové chyby na strane servera (podrobnejší popis ďalej).

1.5 Podporované dátové typy

String	"text" (text v úvodzovkách)
Boolean	true/false (bez úvodzoviek)
Number	číslo so znamienkom a desatinnou bodkou (bez úvodzoviek)
Date	dátum v ISO tvare „YYYY-MM-DD“ (v úvodzovkách)
Timestamp	dátum a čas v ISO tvare „2022-10-21T18:21:49“ (v úvodzovkách)

1.6 URL/URI

`http://adresa:port/request_name?parameters`

- IP-adresa a Port kde beží API server
- Request_name, názov requestu, napríklad „eshop/pricelist“
- Parameters, nepovinné parametre requestu napríklad „id=abcd&date_from=2018-01-01“

1.7 Request headers

Pre správne fungovanie v JSON je potrebné v každom requeste nastaviť HTTP header:

Content-Type: `application/json`

1.8 Response OK (HTTP Status 200)

Úspešne vykonaný request **GET** vracia dáta naformátované v JSON. Pre každý request sú data inak formátované.

Úspešne vykonaný request **POST**, **PUT** alebo **DELETE** vracajú štandardizovanú odpoveď v tomto tvare:

```
{
  "timestamp" : "2022-03-31T19:26:16.3074307",
  "status" : 200,
  "message" : "xxxxxx",
  "path" : ""
}
```

1.9 Response BROWSER (HTTP Status 200 pre zobrazenie tabuľkových dát)

Requesty typu Browser umožňujú zobrazit' dáta v tabuľkovej forme, s podporou filtrovania, usporiadania a ďalších nastavení. Podrobnejší popis je v samostatnej kapitole.

1.10 Response ERROR (HTTP Status 400 Bad Request)

Soft-chyby, napríklad chybné parametre requestu, vracajú štandardizovanú odpoveď v JSON. Oproti Response OK obsahujú navyše polia "error" a "exception". Pole "error" obsahuje kód chyby (rozdielne pre každý typ requestu). Pole "exception" je vždy prázdne (nedochádza k výnimke). Pole "message" obsahuje podrobnejší popis chyby.

```
{
  "timestamp" : "2022-03-31T19:29:37.1425269",
  "status" : 400,
  "error" : "EXIST",
  "exception" : "",
  "message" : "Zákazník s kódom TEST123 už existuje v databáze",
  "path" : ""
}
```

1.11 Response SYSTEM ERROR (HTTP Status 500 a podobné)

Systémové chyby (výnimky) vracajú odpoveď v rovnakom tvare, ako soft-chyby. Pole "error" obsahuje stručný názov chyby, pole "exception" obsahuje názov (triedu) výnimky, pole "message" obsahuje podrobný popis chyby. Pole "path" obsahuje URL requestu, ktorý vyvolal výnimku.

```
{
  "timestamp" : "2022-03-31T17:51:27.283+00:00",
  "status" : 500,
  "error" : "Internal Server Error",
  "exception" : "org.springframework.jdbc CannotGetJdbcConnectionException",
  "message" : "Failed to obtain JDBC Connection; java.sql.SQLException: ...",
  "path" : "/api/customer"
}
```

2 Security – protokol HTTPS (šifrovanie)

Tangram API umožňuje zapnúť šifrovaný prenos (HTTPS protokol). Nadväzovanie spojenia je pomalšie, ako v prípade nešifrovaného prenosu, ale celá komunikácia (vrátane autentifikácie klienta menom a heslom) je šifrovaná.

HTTPS sa doporučuje zapnúť, ak klienti pristupujú k serveru z vonkajšej nezabezpečenej siete.

HTTPS sa nastavuje v konfiguračnom súbore „application.properties“. Ak súbor neobsahuje príslušné riadky, tak ich doplňte. Znak # na začiatku riadka znamená, že celý riadok je „komentár“ (vypnutý). Po zmene v konfiguračnom súbore je nutné reštartovať API, aby sa zmeny prejavili.

```
# SECURITY
server.ssl.enabled=true
server.ssl.key-store: file:./tangram.p12
server.ssl.key-store-password: tangram
server.ssl.key-store-type: pkcs12
server.ssl.key-alias: tangram.sk
```

Vysvetlivky:

enabled = true zapína šifrovanie

key-store obsahuje názov súboru s SSL certifikátom. Certifikát musí byť uložený v tzv. „key store“ vo formáte PKCS12. Na prípone súboru nezáleží. TangramAPI už obsahuje v inštalácii predpripravený self-signed certifikát – certifikát nebol vydaný Certifikačnou autoritou a je to tzv. „neoverený certifikát“. Šifrovanie s týmto certifikátom funguje tak ako má, ale browser (a niektoré knižnice na strane klienta) hlásia varovanie že ide o neoverený certifikát (po odkliknutí komunikácia pokračuje ďalej).

key-store-password je heslo zadané pri generovaní key-store súboru, serveru slúži na otvorenie súboru a načítanie certifikátu

key-store-type je formát key-store súboru, predvolený je štandard PKCS12 (iné formáty neboli testované)

key-alias je názov domény, pre ktorú bol certifikát vygenerovaný

Použitie vlastného certifikátu

Pri inštalácii API na verejne dostupný server (napríklad pre spustenie CRM modulu vo vonkajšej sieti) by sa mal použiť oficiálne vygenerovaný certifikát, viazaný na príslušnú doménu servera.

Certifikáty vydávajú tzv. certifikačné authority za poplatok (certifikát obvykle platí 2 roky).

Oficiálny (overený) certifikát je možné získať aj zdarma (napríklad <https://zerossll.com>, <https://letsencrypt.org>, <https://www.sslforfree.com> a ďalšie). Tieto certifikáty majú obvykle obmedzenú platnosť na maximálne 90 dní. Väčšina free služieb umožňuje generovať certifikáty automaticky v pravidelných intervaloch (hlavne LetsEncrypt).

Pre použitie vlastného certifikátu je nutné ho uložiť do key-store súboru a súbor nahrať do adresára, kde je nainštalovaný TangramAPI server. Tiež je nutné upraviť konfiguračný súbor a reštartovať server.

3 Autentifikácia (meno/heslo)

Autentifikácia sa zapína v konfiguračnom súbore „application.properties“, viď Nastavenie systému.

Tangram REST API podporuje dva druhy autentifikácie: Basic (meno + heslo) a Bearer (token). Obidva spôsoby sú zapnuté naraz a môžu sa kombinovať. Basic sa obvykle používa pre napojenie cudzích systémov (e-shopy a pod.), Bearer používa webová aplikácia Tangram CRM.

Chybné meno, heslo alebo token vracia status **401 Unauthorized**. Request body obsahuje podrobnejší popis chyby (príklad):

```
{
  "timestamp" : "2023-04-11T20:46:28.1176956",
  "error" : "AUTHENTICATION FAILED",
  "message" : "Basic Authentication - unknown user [Andrej]"
}
```

Nepovolený prístup k requestu vracia status **403 Forbidden** (viď nastavenie práv v ďalšom texte).

Vytvorenie používateľov pre API:

- v application.properties vložte riadok „**tangram.admin.password=xxxxx**“, ktorý nastaví heslo pre zabudovaný (nezmazateľný) login „admin“, reštartujte službu API. Bez admin hesla sa nedajú spustiť ďalšie funkcie.
- v TangramEOS spustíte **Číselníky – Používateľ systému – Zamestnanci a prístupové práva**,

3.1 Nastavenie systému (Používatelia, Roly, Práva)

V adresári TangramServer je konfiguračný súbor „application.properties“, nastavte riadky:

- **tangram.authentication=true**, zapne kontrolu
- **tangram.admin.password=xxxxx**, nastaví heslo pre zabudovaný (nezmazateľný) login „admin“

Heslo pre administrátora je potrebné na nastavenie prístupových práv (viď ďalej). Po zmene v súbore reštartujte službu API.

V programe TangramEOS vytvorte používateľov a prístupové práva:

- v **Systém – Konfigurácia – Konfigurácia programu**, v bloku Tangram Server API, zadajte adresu a port servera. Zadajte aj admin heslo, ktoré ste vytvorili v súbore application.properties
- funkciou **Systém – API – Zoznam rolí TangramServer** vytvorte zoznam rolí, ktoré sa budú prideľovať jednotlivým používateľom API
- funkciou **Systém – API – Prístupové práva TangramServer** prideľte práva k vytvoreným roliam, na spustenie tejto funkcie musí bežať API služba
- funkciou **Systém – API – Zoznam používateľov TangramServer** vytvorte jedného alebo viac používateľov

Zoznam používateľov je integrovaný priamo do zoznamu zamestnancov kvôli prístupu k modulu CRM. Na vytvorenie loginu pre externý systém vytvorte nového „zamestnanca“. Musí sa zadať Osobné číslo (ľubovoľný unique kód), Meno (napríklad názov externého systému) a Prístupové údaje pre TangramServer (login a heslo). Zo zoznamu rolí vyberte predtým nadefinovanú rolu.

Ak zmeníte údaje v Roliach alebo Zamestnancoch, doporučujeme reštartovať TangramServer, pretože tieto údaje sú držané v cache a je nutné zmenené dáta načítať z databázy.

Zoznam používateľov je možné vytvoriť/aktualizovať aj vo web module TangramCRM, pokiaľ je nainštalovaný.

3.2 Autentifikácia Basic (meno a heslo)

Pri tomto spôsobe sa zasiela meno a heslo v každom requeste. Je vhodný pre klientov, ktorí môžu mať meno a heslo bezpečne uložené.

Každý request musí obsahovať HTTP header **Authorization: Basic xxxxxx**, kde reťazec „xxxxxx“ obsahuje jednoducho zakódované meno a heslo. Medzi „Basic“ a „xxxxxx“ musí byť jedna medzera.

Spôsob šifrovania mena a hesla:

- meno a heslo sa spojí do reťazca „**meno:heslo**“ (oddeľovač dvojbodka, žiadne medzery)
- reťazec sa zakóduje algoritmom **Base64**

Príklad (uvodzovky sú ilustračné, do HTTP headera sa nedávajú):

- meno „Andrej“
- heslo „andrej123“
- reťazec „Andrej:andrej123“
- zakódované do Base64 „QW5kcmlVqOmFuZlJlajEyMw==“

Meno aj heslo sú case-sensitive. Meno nesmie obsahovať znak „.“ (dvojbodka).

Chyby (status 401):

- „HTTP header [Authorization] not found“ ak request vôbec neobsahuje HTTP header „Authorization“
- „Illegal base64 character“ ak je reťazec „meno:heslo“ chybné zakódovaný (status 500)
- „wrong format“ ak zakódovaný reťazec neobsahuje dvojbodku a nedá sa rozdeliť na meno a heslo
- „unknown user“ ak sa meno nenašlo v zozname používateľov
- „account is locked“ ak je používateľ v stave „neaktívny“
- „invalid password“ ak nesúhlasí heslo

3.3 Autentifikácia Bearer (token)

Pri tomto spôsobe sa najprv vyšle request **GET /api/login**, ktorý musí obsahovať Basic autentifikáciu (viď predošlý odstavec) s menom a heslom. V odpovedi sa vygeneruje token (textový reťazec), ktorý sa použije na autentifikáciu všetkých nasledujúcich requestov. Token je platný 24 hodín a môže sa predčasne zneplatniť requestom **GET /api/logout**. Vygenerované tokeny sa držia len v pamäti a pri reštarte servera sa vymažú.

Autentifikácia tokenom je vhodná pre webové aplikácie, kde user zadá meno a heslo. API vygeneruje token, ktorý sa uloží na strane frontendu (napríklad do cookies). Dôležité je, že sa do cookies neukladá meno a heslo a nie je možné ho odhaliť.

Login – vygenerovanie tokenu:

- odoslať request GET /api/login, ktorý v hlavičke obsahuje Basic autentifikáciu (meno a heslo)
- v prípade neúspešnej autentifikácie sa vráti status 401 s príslušnou chybou (viď Basic Auth)
- príklad úspešnej odpovede (request body):

```
{
  "user" : {
    "id" : "Andrej",
    "psw" : "",
    "role" : "MANAGER",
    "active" : true,
    "meno" : "Andrej Kokoška",
    "funkcia" : "OBCHOD",
    "pracovisko" : "Obchod",
    "telefon" : "+421999666777",
    "email" : "andrej.kokoska@test.sk",
    "poznamka" : "Poznámka na viacej riadkov\r\nndruhý riadok\r\ntretí riadok",
    "obchzast" : "ABCDEFGHJIJ"
  },
  "tokenId" : "9ce5dd24-0587-4174-9551-bb885e15bac2",
  "tokenValidity" : "2023-04-12T21:36:07.4127835"
}
```

Pole „user“ obsahuje údaje o užívateľovi (heslo je vždy vymazané)

Pole „tokenId“ obsahuje vygenerovaný token

Pole „tokenValidity“ obsahuje dátum a čas, dokedy je token platný

Ak sa pošle login pre toho istého užívateľa znova, vygeneruje sa nový token a predošlý sa vymaže. Z toho vyplýva, že jeden používateľ môže byť prihlásený len z jedného zariadenia naraz.

Request – použitie tokenu

Všetky nasledujúce requesty musia obsahovať HTTP header **Authorization: Bearer xxxxxx**, kde reťazec „xxxxxx“ je vygenerovaný token. Medzi „Bearer“ a „xxxxxx“ musí byť jedna medzera.

Chyby (status 401):

- „Token unknown“ ak sa zadaný token vôbec nenašiel v pamäti
- „Token expired“ ak uplynula platnosť tokenu

Logout – zrušenie tokenu

Request **GET /api/logout** vymaže token z pamäte. Tento request musí byť autentifikovaný, buď Basic alebo Bearer. Ak je token už neplatný alebo vymazaný, nevráti sa chyba, vždy sa vracia status 200.

4 Prístupové práva

Kontrola prístupových práv k API sa vykonáva, len ak je v konfigurácii zapnutá autentifikácia. Ak je autentifikácia vypnutá (napr. API spustené v testovacej prevádzke), klient má prístup k všetkým endpointom bez obmedzenia.

Prístupové práva sú realizované pomocou Rolí (Role) a Prístupov (Permission).

Nastavenie (editácia) rolí sa môže vykonávať pomocou requestov **/api/admin/roles**, **/api/admin/permissions** a **/api/admin/users**. Všetky vyžadujú právo ADMIN. Podrobnosti v online dokumentácii.

Okrem toho sa používatelia/roly/práva nastavujú aj v programe TangramEOS, menu Systém.

Prístupy (Permissions) je pevne daný zoznam, definovaný vnútri API servera. Zoznam všetkých Prístupov sa dá načítať requestom **GET api/admin/permissions/all** (vyžaduje ADMIN permission). Zoznam má tvar:

```
[
  { "id": "ADMIN", "label": "Administrátor (všetky admin funkcie)" },
  { "id": "CUSTOMER_READ", "label": "Zákazníci - prezerať" },
  { "id": "CUSTOMER_CREATE", "label": "Zákazníci - vytvoriť" },
  { "id": "CUSTOMER_UPDATE", "label": "Zákazníci - opraviť" },
  { "id": "CUSTOMER_DELETE", "label": "Zákazníci - vymazať z databázy" },
  ...
]
```

Roly (Roles) sú definovateľné používateľom a ukladajú sa v databáze. Každý používateľ musí mať priradenú jednu rolu (ak nemá žiadnu rolu, nemá žiadne práva). K roli sa priradzujú prístupové práva zo zoznamu Permissions. Zoznam prístupov k roli sa dá načítať requestom **GET api/admin/permissions?role_id=MANAGER** v tvare:

```
[ "ADMIN", "CUSTOMER_READ", "CUSTOMER_UPDATE", ... ]
```

Každý request je chránený niektorým z Prístupov, okrem requestov zo sekcie Login (login, logout, check, auth a permissions) a Číselníky (/api/list/xxx).

Pri volaní requestu, ku ktorému nemá user prístup sa vráti status 403 Forbidden a response body obsahuje podrobnosti v tvare:

```
{
  "timestamp" : "2023-04-19T10:40:38.8571505",
  "error" : "NOACCESS",
  "message" : "User [ANDREJ], role [ROLE01], no permission [ORDER_READ]"
}
```

Pole „message“ obsahuje informáciu o autentifikovanom userovi, jeho priradenej roli a vyžadovanom Prístupovom práve k volanému requestu.

4.1 Použitie vo web-aplikáciách

Aplikácia by mala korektne ošetriť response 403 a zobraziť používateľovi rozumné chybové hlásenie aj s obsahom poľa „message“. Toto pomôže správcovi rýchlejšie sa zorientovať, ktoré Prístupy je nutné priradiť príslušnej roli.

Na skrytie nedostupných funkcií (menu, tlačítka, ...) slúži request **/api/permissions**, ktorý vráti zoznam Prístupov pre aktuálne prihláseného používateľa v tvare:

```
[ "ADMIN", "CUSTOMER_READ", "CUSTOMER_UPDATE", ... ]
```

Tento request vyžaduje autentifikáciu (prihlásenie používateľa) ale nevyžaduje žiadne prístupové práva. Podľa vráteného zoznamu by mala aplikácia zablokovať alebo skryť nedostupné funkcie.

5 BROWSER

5.1 Úvod

Requesty typu Browser umožňujú zobrazit' dáta v tabuľkovej forme, s podporou filtrovania, usporiadania a ďalších nastavení. Na rozdiel od štandardných requestov, ktoré vracajú len čisté dáta, browser vracia aj popis stĺpcov a ďalšie meta-dáta.

5.2 Request

Request pre browser je vždy metóda **POST** a môže obsahovať parametre v request-body. Pri prvom volaní sa parametre obvykle nezadávajú a server vygeneruje odpoveď s default hodnotami pre filter aj order.

Štruktúra request body je štandardizovaná:

```
{
  "search": "xxx":
  "filter":
  [
    { "id": "id",      "operator": "IN",      "values": ["0004", "0006"] },
    { "id": "cislo",  "operator": "BETWEEN", "values": [100, 199] },
    { "id": "active", "operator": "EQ",      "values": [true] },
    { "id": "datum", "operator": "BETWEEN", "values": ["2022-01-01", "2022-12-31"] }
  ],
  "order": { ... upresniť štruktúru ... }
}
```

Search

Hodnota hľadaná ako podreťazec vo viacerých stĺpcoch naraz (full-text search). Hľadá sa v stĺpcoch s príznakom „search“=true. Hľadanie je case-insensitive a accent-insensitive (ignoruje diakritiku).

Filter

Zoznam podmienok, ktoré sa použijú na vytvorenie filtra (SQL WHERE) pri čítaní z databázy. Ak je zadaných viac podmienok, musia platiť všetky súčasne. Ak nie je zadaná žiadna podmienka, server použije default hodnoty (pre každý browser iné). Názvy stĺpcov sú case-sensitive, operátory sú vždy uppercase. Číselné a boolean hodnoty môžu byť zadané v úvodzovkách alebo bez nich, stringy a dátumy sú vždy v úvodzovkách (JSON pravidlá).

Jedna podmienka má štruktúru { "id": "stĺpec", "operator": "kód", "values": [hodnota1, hodnota2, ...] }

Podporované operátory:

Operátor	Hodnoty	Popis
EQ	1	presná zhoda podporované všetky typy (STRING NUMBER DATE DATETIME a BOOLEAN) stringy sa porovnávajú case-sensitive
SUBSTR	1	hľadanie podreťazca len pre STRING ignoruje sa case aj diakritické znamienka
BETWEEN	1 alebo 2	hľadanie od-do (vrátane hraníc) len pre NUMBER a DATE, dá sa použiť aj na hľadanie >= ("values":["2022-01-01", null]) alebo <= ("values":[null, "2022-12-31"])
IN	1..999	hľadanie v zozname hodnôt len pre hodnoty STRING podmienka je splnená ak sa hodnota stĺpca zhoduje s niektorou zo zadaných hodnôt hľadá sa presná zhoda

Order

Ešte nie je dokončené ...

5.3 Response

Štandardizovaná odpoveď pre načítanie dát vo forme tabuľky (riadky a stĺpce). Obsahuje definíciu stĺpcov s názvami a dátami vo forme tabuľky (dvojmerné pole).

Základná štruktúra odpovede (príklad tabuľky, 3 stĺpce, 4 riadky):

```
{
  "columnCount" : 3,
  "dataCount" : 4,
  "totalCount" : 150,
  "limitCount" : 4,
  "columns" :
  [ { "id":"id",
      "type":"STRING",
      "decimals":0,
      "style":"BOLD",
      "clickable":true,
      "uppercase":true,
      "title":"Login",
      "tooltip":"prihlasovacie meno (unique ID)"
    },
    { "id":"active",
      "type":"BOOLEAN",
      "decimals":0,
      "style":"NONE",
      "clickable":false,
      "title":"Aktívny",
      "tooltip":null
    },
    { "id":"meno",
      "type":"STRING",
      "decimals":0,
      "style":"NONE",
      "clickable":false,
      "title":"Meno",
      "tooltip" : "Plné meno používateľa"
    }
  ],
  "data" :
  [ [ "001", true, "Stredisko Bratislava"],
    [ "002", false, "Stredisko Žilina"],
    [ "003", true, "Stredisko Košice"],
    [ "004", true, "Stredisko Banská Bystrica"]
  ],
  "filter" : { kópia filtra z requestu }
}
```

Pole **columnCount** obsahuje počet stĺpcov v poli **columns**.

Pole **dataCount** obsahuje počet riadkov v tabuľke **data**.

Pole **totalCount** obsahuje počet všetkých riadkov z databázy, ktoré vyhoveli filtru

Pole **limitCount** obsahuje nastavený limit (maximálny počet riadkov v tabuľke **data**)

Pole **columns** obsahuje zoznam stĺpcov tabuľky. Každý stĺpec obsahuje atribúty:

- **id** = unikátne ID stĺpca v rámci tabuľky (nezobrazuje sa)
- **type** = typ hodnoty STRING, NUMBER, DATE, DATETIME alebo BOOLEAN
- **decimals** = počet desatinných miest (len pre typ NUMBER)
- **style** = formátovanie obsahu stĺpca (**NONE**, **BOLD** tučný font, **STRIKE** preškrtnutý font, **HIDDEN** skrytý stĺpec)
- **clickable** = true (povolený click na hodnotu v riadku, ktorý otvorí "formulár"), false = žiaden click
- **uppercase** = true (hodnoty sú uppercased, filter typu EQ sa musí zadávať uppercased)
- **title** = nadpis stĺpca pri zobrazení tabuľky na strane frontendu
- **tooltip** = podrobnejší popis stĺpca (text do bubliny)

Pole **data** obsahuje hodnoty usporiadané po riadkoch. Každý riadok obsahuje pole hodnôt. Poradie hodnôt v riadku zodpovedá poradiu stĺpcov definícii columns.

Hodnoty sú formátované podľa JSON štandardu:

STRING: "text v dvojitéch úvodzovkách"

NUMBER: 9999.9999 (číslo s desatinnou bodkou, bez úvodzoviek)

DATE: "2022-07-14" (dátum v ISO tvare, v úvodzovkách)

DATETIME: "2022-07-14T11:16:44.0" (časová pečiatka v ISO tvare)

BOOLEAN: true/false (bez úvodzoviek)

6 VENDING (predajné automaty)

6.1 Úvod

Táto časť API je určená na pripojenie predajných automatov (nápoje, jedlá, ...).

Predpokladá sa nasledujúci postup predaja cez automat:

- zákazník priloží kartu k snímaču
- automat odošle na API request „Vyhľadanie karty“
- API vyhľadá kartu v databáze a vráti údaje o karte a zákazníkovi, vrátane zostávajúceho kreditu na karte
- pokiaľ je dostatočný kredit, automat vykoná predaj
- automat odošle na API request „Zápis transakcie“
- API zapíše transakciu do databázy a ihneď aktualizuje zostávajúci kredit na karte

6.2 Autentifikácia

API umožňuje dva spôsoby autentifikácie, BasicAuth a Token, podrobnosti v prvej kapitole.

6.3 Vyhľadanie karty

Vyhľadanie údajov o karte, zákazníkovi a zostávajúcom kredite

[GET] [api/vending/cards?id=8700C033E6](#)

Povinný parameter [id] obsahuje kód karty (číslo čipu v prípade RFID kariet)

Odpoveď OK (statut 200)

```
{
  "id" : "8700C033E6",
  "printedId" : "1025",
  "customerId" : "00101",
  "popis" : "Poznámka pre kartu Jozka Mrkvicku",
  "meno" : "Mrkvíčka Jozef Ing.",
  "stredisko" : "PSA.001",
  "publicMsg" : "Poznámka pre zákazníka, zobrazuje sa na termináloch",
  "kredit" : 12.40
}
```

id = kód karty (číslo čipu)

printedId = evidenčné číslo karty, obvykle vytlačené na karte

customerId = kód zákazníka (napr. osobné číslo zamestnanca)

popis = interná poznámka ku karte

meno = meno a priezvisko zákazníka

stredisko = stredisko/oddelenie

publicMsg = správa určená pre zákazníka, doporučujeme zobrazit' na displayi automatu, ak je to možné

kredit = zostávajúci kredit na karte

Odpoveď ERROR (status 400)

nenájdená alebo neplatná karta (nie je priradená zákazníkovi, platnosť karty skončila a pod.)

```
{
  "timestamp" : "2023-04-04T11:57:16.9979481",
  "error" : "NODATA",
  "message" : "Card [8700C033E6] not found"
}
```

6.4 Zápis transakcie

Zápis transakcie do databázy po úspešnom predaji na automate. Pri zápise transakcie sa už zostávajúci kredit neoveruje, túto kontrolu musí vykonať automat ešte pred predajom. Pri zápise transakcie sa nemôže vyhodit' chyba 400. Ak sa vyhodí chyba 5xx, znamená to systémovú chybu na strane servera.

[POST] [api/vending/transactions](#)

```
{
  "cardId": "8700C033E6",
  "amount": 2.40,
  "device": "KOD-AUTOMATU-123456",
}
```

```
    "comment": "Poznámka od predajného automatu"  
}
```

cardId = kód karty

amount = suma nákupu (celá suma, vrátane DPH)

device = kód automatu, ktorý uskutočnil predaj

comment = ľubovoľná poznámka, napr. číslo slotu alebo názov produktu

Odpoveď OK (status 200)

```
{  
  "timestamp" : "2023-04-04T12:13:29.4579135",  
  "message" : "0000000013"  
}
```

Pole „message“ obsahuje kód vytvorenej transakcie, doporučujeme logovať na strane automatu, pokiaľ je to možné.