STRUKTURE ZA RAZMENU PODATAKA

JSON • XML • API razmena • validacija • čuvanje u bazama



ZAŠTO SU POTREBNI FORMATI ZA RAZMENU PODATAKA?

Različite aplikacije moraju da razumeju isto značenje podataka.

WEB APP
JavaScript

MOBILE
Android / iOS

SERVIS
Python / Java

BAZA
SQL / NoSQL

ZAJEDNIČKI JEZIK PODATAKA = JSON / XML

Interoperabilnost
Različite tehnologije razmenjuju iste informacije.

Nezavisnost
Format nije vezan za jedan programski jezik.

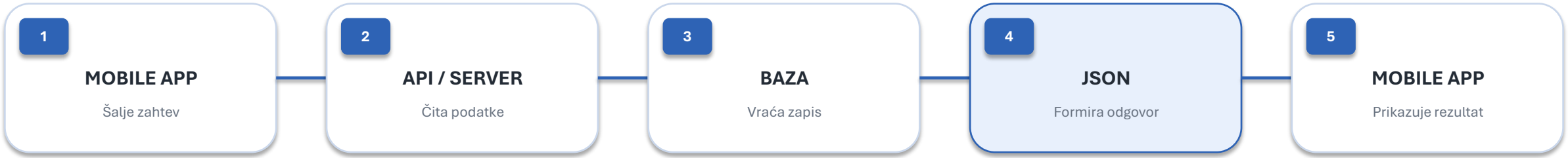
Održavanje
Lakše je menjati jednu aplikaciju bez rušenja drugih.

Automatizacija
Podaci se mogu parsirati i obraditi bez ručnog rada.

Student treba da zapamti: format je dogovor kako će podatak izgledati tokom prenosa između sistema.

KAKO IZGLEDA RAZMENA PODATAKA U PRAKSI?

Primer: mobilna aplikacija traži podatke o studentu.



ODGOVOR

```
{
  "id": 50,
  "ime": "Jelena",
  "smer": "Informacione tehnologije"
}
```

JSON ili XML su sadržaj poruke; API je mehanizam preko kog se poruka šalje između aplikacija.

JSON I XML — DVA NAČINA DA OPIŠEMO ISTI PODATAK

Isti sadržaj, različita sintaksa.

JSON

```
{
  "student": {
    "ime": "Marko",
    "godina": 17,
    "budzet": true
  }
}
```

ISTO ZNAČENJE

XML

```
<student>
  <ime>Marko</ime>
  <godina>17</godina>
  <budzet>true</budzet>
</student>
```

JSON je obično kompaktniji; XML je eksplicitniji i ima bogatiji sistem tagova, atributa i šema.

JSON OBJEKAT — KLJUČ + VREDNOST

Objekat je skup parova koji opisuju jedan entitet.

JSON OBJEKAT

```
{
  "ime": "Marko",
  "prezime": "Marković",
  "godina": 17,
  "budzet": true
}
```

1. { }
Početak i kraj objekta

2. "ime"
Ključ — naziv podatka

3. :
Razdvaja ključ od vrednosti

4. "Marko"
Vrednost podatka

5. ,
Razdvaja parove ključ–vrednost

Najvažnije pravilo: ključ je u navodnicima, zatim ide dvotačka, pa vrednost; stavke se razdvajaju zarezom.

JSON TIPOVI PODATAKA

Vrednost nije uvek tekst — JSON razlikuje šest osnovnih tipova.

TEKST

```
"Marko"
```

Primer: ime

BROJ

```
17 / 4.5
```

Primer: godina / ocena

LOGIČKA

```
true / false
```

Primer: budžet

NULL

```
null
```

Primer: telefon nije poznat

NIZ

```
[8, 9, 10]
```

Primer: ocene

OBJEKAT

```
{"grad":"Niš"}
```

Primer: adresa

Tip podatka određuje kako aplikacija može da obradi vrednost — npr. broj sabira, tekst prikazuje, niz ponavlja.

JSON NIZOVI I UGNEŽDAVANJE

Jedan objekat može sadržati listu objekata.

PRIMER

```
{
  "student": "Ana",
  "ocene": [8, 9, 10],
  "predmeti": [
    {"naziv":"Baze", "espb":6},
    {"naziv":"Mreže", "espb":6}
  ]
}
```

Niz [ ]

Više vrednosti istog ili različitog tipa, u definisanom redosledu.

Ugnežden objekat { }

Objekat se može nalaziti unutar drugog objekta ili niza.

Zašto je korisno?

Prirodno opisuje realne strukture: student → više predmeta.

Niz odgovara listi, a ugnežđeni objekat složenom delu entiteta.

PRIMER 1 — PROFIL NA DRUŠTVENOJ MREŽI

Od ekrana aplikacije do JSON dokumenta.

MM

Marko Marković

@marko_2007 • Beograd

342	89	2
prati	pratilaca	fotografije

JSON

```
{
  "korisnicko_ime": "marko_2007",
  "ime": "Marko Marković",
  "grad": "Beograd",
  "prati": 342,
  "pratioci": 89,
  "fotografije": [
    {"naslov":"Na planini", "likes":45},
    {"naslov":"Sa prijateljima", "likes":67}
  ]
}
```

Ekran koji korisnik vidi može da bude direktno popunjen podacima iz JSON odgovora.

PRIMER 1 — KAKO SE PROFIL ČUVA U TABELAMA?

JSON dokument može da se dobije spajanjem podataka iz više povezanih tabela.

Tabela KORISNIK

KorisnikID	KorisničkoIme	Ime	Grad	Prati	Pratilaca
101	marko_2007	Marko Marković	Beograd	342	89

Tabela FOTOGRAFIJA

FotoID	KorisnikID (FK)	Naslov	URL	Likes
1	101	Na planini	foto1.jpg	45
2	101	Sa prijateljima	foto2.jpg	67

Veza 1 : N

KORISNIK ——— **FOTOGRAFIJA**

Jedan korisnik može imati više fotografija.
KorisnikID u tabeli FOTOGRAFIJA je strani ključ.

API može ove redove da sastavi u jedan JSON profil sa nizom fotografija.

U JSON-u fotografije su niz, a u relacionoj bazi redovi posebne tabele povezani stranim ključem.

PRIMER 2 — VREMENSKA PROGNOZA

Aplikacija pozove servis i dobije strukturirani odgovor.

NIŠ

22°C

Sunčano

Vlaga: 65% Vetar: 12 km/h

API ODGOVOR

```
{
  "grad": "Niš",
  "temperatura": 22,
  "vlaga": 65,
  "opis": "Sunčano",
  "vetar": {
    "brzina": 12,
    "smer": "severozapad"
  }
}
```

Aplikacija ne mora da zna kako servis interno čuva podatke — dovoljno je da zna strukturu JSON odgovora.

PRIMER 2 — KAKO SE VREMENSKI PODACI ČUVAJU?

Svaki red predstavlja jedno merenje u određenom trenutku.

Tabela MERENJE_VREMENA

MerenjeID	Grad	Vreme	Temperatura	Vlaga	Opis	BrzinaVetra	SmerVetra
501	Niš	12:00	22	65	Sunčano	12	SZ
502	Niš	13:00	23	61	Sunčano	10	Z
503	Beograd	12:00	20	68	Oblačno	8	S

JSON objekat „vetar“

```
{
  "brzina": 12,
  "smer": "SZ"
}
```

→ u tabeli mogu biti dve kolone

Za istoriju je potreban datum/vreme merenja — bez toga ne znamo kada je vrednost nastala.

Ako vetar postane složen entitet sa više podataka, može se izdvojiti u posebnu tabelu.

Ugnežden JSON objekat se u tabelarnom modelu često „izravna“ u kolone radi jednostavnijeg pretraživanja.

PRIMER 3 — ONLINE PORUDŽBINA U JSON-U

Složen poslovni objekat: kupac + niz proizvoda + ukupna cena.

PORUDŽBINA #12345

Kupac: Marko • Niš

Patike 4.500 RSD x1

Čarape 200 RSD x3

UKUPNO: 5.100 RSD

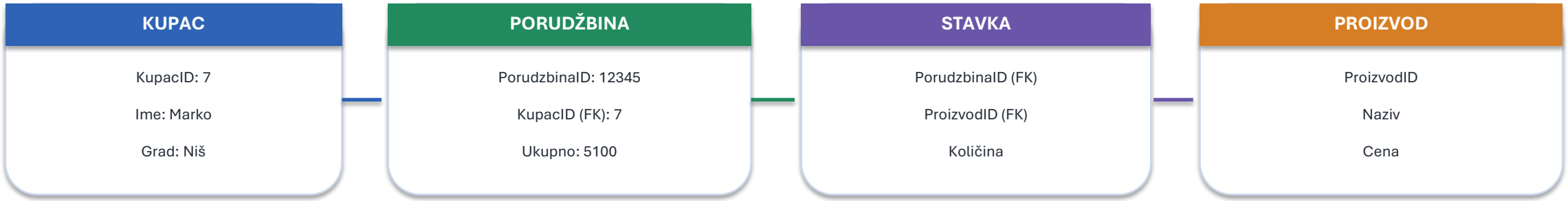
JSON

```
{
  "id_porudzbine": 12345,
  "kupac": {"ime":"Marko", "grad":"Niš"},
  "proizvodi": [
    {"naziv":"Patike", "cena":4500, "kolicina":1},
    {"naziv":"Čarape", "cena":200, "kolicina":3}
  ],
  "ukupno": 5100
}
```

U jednom JSON dokumentu možemo da prenesemo ceo poslovni događaj sa svim povezanim podacima.

PRIMER 3 — ONLINE PORUDŽBINA U RELACIONOJ BAZI

Jedan JSON dokument se u relacionoj bazi prirodno razlaže na više tabela.



1 kupac → više porudžbina → više stavki → svaka stavka pokazuje na proizvod

STAVKA_PORUDŽBINE

PorudzbinaID	ProizvodID	Količina
12345	301	1
12345	412	3

PROIZVOD

ProizvodID	Naziv	Cena
301	Patike	4500
412	Čarape	200

Niz proizvoda iz JSON-a postaje skup redova u tabeli STAVKA_PORUDŽBINE.

XML — ELEMENTI I TAGOVI

XML opisuje podatke kroz hijerarhiju otvarajućih i zatvarajućih oznaka.

XML DOKUMENT

```
<student>
  <ime>Marko</ime>
  <godina>17</godina>
  <budzet>true</budzet>
</student>
```

Koreni element

<student> ... </student>

Element

<ime>Marko</ime>

Otvarajući tag

<ime>

Sadržaj

Marko

Zatvarajući tag

</ime>

XML dokument mora imati jedan koreni element koji obuhvata celu strukturu.

XML ČVOROVI, ATRIBUTI I KOMENTARI

XML ima više vrsta gradivnih elemenata od JSON-a.

ELEMENT

```
<student>...</student>
```

Glavni nosilac strukture

TEKST

```
Marko
```

Sadržaj elementa

ATRIBUT

```
godina="2026"
```

Dodatni podatak u tagu

KOMENTAR

```
<!-- napomena -->
```

Ne obrađuje se kao podatak

CDATA

```
<![CDATA[...]]>
```

Sadržaj koji parser ne tumači kao XML

NULL / NIL

```
xsi:nil="true"
```

Eksplicitno odsustvo vrednosti

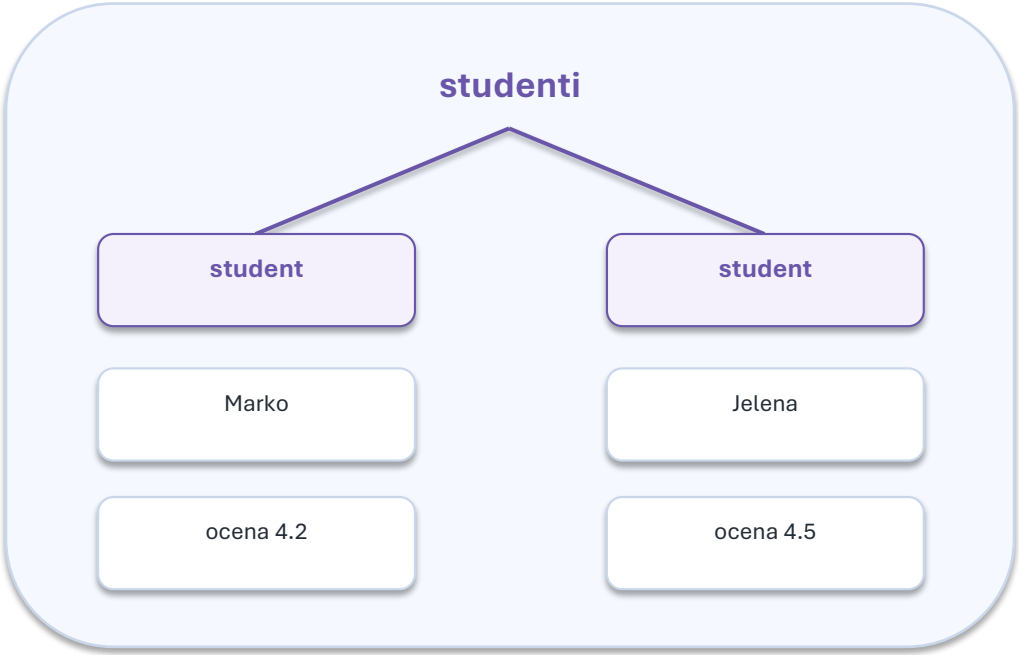
XML je opisniji i pogodniji kada su važni tagovi, atributi, validacija i formalna struktura dokumenta.

KAKO XML PREDSTAVLJA LISTE?

Lista se najčešće zapisuje ponavljanjem elemenata istog tipa.

XML LISTA

```
<studenti>
  <student>
    <ime>Marko</ime>
    <ocena>4.2</ocena>
  </student>
  <student>
    <ime>Jelena</ime>
    <ocena>4.5</ocena>
  </student>
</studenti>
```



JSON koristi niz [ ], dok XML istu ideju najčešće predstavlja ponavljanjem istih elemenata.

ISTA PORUDŽBINA — JSON I XML

Uporedite strukturu, ne učite oba primera napamet.

JSON

```
{
  "id_porudzbine": 105,
  "kupac": {
    "ime":"Ana Petrović",
    "grad":"Niš"
  },
  "proizvodi": [
    {"naziv":"Laptop", "kolicina":1}
  ],
  "ukupno": 85000
}
```

XML

```
<porudzbina>
  <id_porudzbine>105</id_porudzbine>
  <kupac>
    <ime>Ana Petrović</ime>
    <grad>Niš</grad>
  </kupac>
  <proizvodi>
    <proizvod>
      <naziv>Laptop</naziv>
      <kolicina>1</kolicina>
    </proizvod>
  </proizvodi>
  <ukupno>85000</ukupno>
</porudzbina>
```

Pitajte: gde je kupac, gde je proizvod, gde je ukupno? Struktura je ista, zapis je drugačiji.

JSON ILI XML — KADA KOJI FORMAT?

Ne postoji univerzalno bolji format; izbor zavisi od sistema i zahteva.

JSON

Čitljivost i veličina

Kraći zapis, manje oznaka.

JSON

Web / REST API

Danas je najčešći izbor za web servise.

XML

Atributi i formalni dokumenti

Prirodna podrška za tagove i atribute.

XML

Validacija složenih dokumenata

XSD omogućava vrlo formalna pravila.

JSON

Rad u JavaScript-u

Direktno se mapira na objekte.

XML

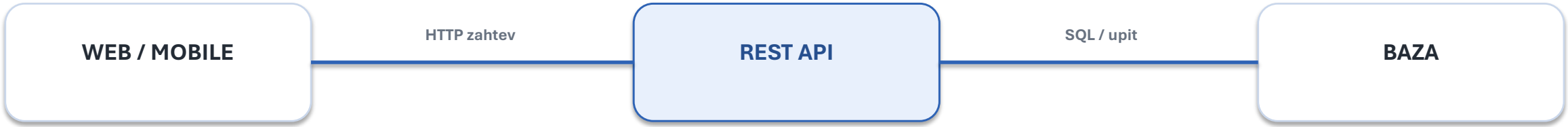
Legacy / enterprise integracije

Čest u SOAP i starijim poslovnim sistemima.

Za moderne REST API-je najčešće ćete videti JSON; XML je i dalje važan u standardizovanim i enterprise integracijama.

JSON U REST API-JU

Klijent šalje zahtev, server vraća odgovor — najčešće u JSON formatu.



ZAHTEV

```
GET /api/studenti/50
Accept: application/json
```

ODGOVOR

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id":50, "ime":"Jelena"
}
```

JSON je format tela poruke; HTTP definiše kako zahtev i odgovor putuju između klijenta i servera.

VALIDACIJA — KAKO ZNAMO DA JE DOKUMENT ISPRAVAN?

Pored sintakse, sistemi često proveravaju i pravila strukture.

SINTAKSNO ISPRAVAN, ALI POGREŠNI TIPOVI

```
{
  "ime": "Ana",
  "godina": "sedamnaest",
  "email": 123
}
```

VALIDAN PREMA PRAVILIMA

```
{
  "ime": "Ana",
  "godina": 17,
  "email": "ana@example.com"
}
```

JSON Schema

Definiše obavezna polja, tipove, opsege i format vrednosti.

XSD za XML

Definiše dozvoljene elemente, atribute, redosled i tipove.

„Može da se parsira“ nije isto što i „podaci su ispravni“ — validacija proverava poslovna i strukturna pravila.

JSON I XML U RELACIONIM BAZAMA

Savremeni RDBMS može da kombinuje klasične kolone i polustrukturirane dokumente.

Tabela: Porudzbine

ID	Kupac	Ukupno	Details (JSON/XML)
105	Ana	88.000	`{"grad":"Niš","kanal":"web"}`
106	Marko	5.100	`{"grad":"Niš","kanal":"mobile"}`
107	Jelena	12.500	`<dostava><grad>BG</grad></dostava>`

Hibridni model

Relacione kolone + JSON/XML u istom redu.

Pretraga iz SQL-a

Funkcije mogu čitati polja unutar dokumenta.

Indeksiranje

Često je moguće indeksirati polja iz JSON-a.

Relacione i dokumentne ideje danas često rade zajedno — ne moramo birati samo jednu paradigmu.

PODRŠKA U RDBMS SISTEMIMA

Primeri mogućnosti koje se sreću u savremenim relacionim bazama.

Oracle

JSON + XML

JSON funkcije; XMLType / XML podrška

SQL + dokumenti

SQL Server

JSON + XML

JSON funkcije; xml tip i metode

SQL + dokumenti

PostgreSQL

JSON / JSONB

Napredna obrada i indeksiranje JSON-a

SQL + dokumenti

MySQL

JSON

JSON tip i ugrađene JSON funkcije

SQL + dokumenti

Različiti DBMS proizvodi nude različite detalje implementacije, ali trend je jasan: JSON je standardna potreba modernih aplikacija.

PROVERA RAZUMEVANJA — OD EKRANA DO DOKUMENTA

Kratka vežba kojom studenti povezuju strukturu, format i primenu.

SCENARIO: ONLINE PRODAVNICA

- Kupac: Ana Petrović, Niš
- Laptop — 1 kom — 85.000 RSD
- Miš — 2 kom — 1.500 RSD
- Ukupno: 88.000 RSD

1 **JSON** — Napiši dokument sa kupcem i nizom proizvoda.

2 **XML** — Prevedi isti sadržaj u XML hijerarhiju.

3 **POREĐENJE** — Označi objekat/niz i odgovarajuće XML elemente.

4 **API** — Objasni kako bi ovaj dokument stigao do mobilne aplikacije.

Ako student ume da prevede isti realni primer u JSON i XML i objasni put kroz API — razume temu.

PITALICA 1/5

Izaberite jedan tačan odgovor.

JSON profil korisnika sadrži niz fotografija. Koji relacioni model najbolje predstavlja taj podatak?

A Jedna tabela KORISNIK sa kolonama Foto1, Foto2, Foto3...

B Tabela KORISNIK i tabela FOTOGRAFIJA povezana stranim ključem KorisnikID.

C Jedna tabela FOTOGRAFIJA u kojoj se ponavljaju svi podaci o korisniku.

D JSON se ne može predstaviti u relacionoj bazi.

Razmišljajte o strukturi podataka, relacijama i načinu na koji će se podaci pretraživati.

PITALICA 2/5

Izaberite jedan tačan odgovor.

API vraća {"grad":"Niš","vetar":{"brzina":12,"smer":"SZ"}}. Baza čuva istoriju merenja. Koji model je najpraktičniji?

A Jedan red po gradu bez vremenske oznake, jer se čuva samo poslednje stanje.

B Tabela MERENJE sa datumom/vremenom i kolonama BrzinaVetra i SmerVetra.

C Svako merenje čuvati kao XML fajl u posebnom folderu.

D Kolonu Vetar čuvati kao tekst „12-SZ“ bez odvojenih podataka.

E Za svaki sat napraviti novu tabelu.

Razmišljajte o strukturi podataka, relacijama i načinu na koji će se podaci pretraživati.

PITALICA 3/5

Izaberite jedan tačan odgovor.

Zašto se niz proizvoda iz JSON porudžbine najčešće ne čuva kao tekst „Patike x1, Čarape x3“ u jednoj koloni?

A Zato što SQL baze ne podržavaju tekstualne kolone.

B Zato što bi bilo teško pretraživati, povezivati i menjati pojedinačne proizvode i količine.

C Zato što JSON uvek mora biti sačuvan u XML formatu.

D Zato što svaki proizvod mora imati svoju bazu podataka.

Razmišljajte o strukturi podataka, relacijama i načinu na koji će se podaci pretraživati.

PITALICA 4/5

Izaberite jedan tačan odgovor.

Koja tvrdnja najbolje opisuje odnos JSON/XML formata i načina čuvanja podataka u relacionoj bazi?

A Ako API koristi JSON, baza mora da bude dokumentna baza.

B Ako baza koristi tabele, API mora da vraća CSV.

C Format razmene i model skladištenja mogu biti različiti; backend prevodi između njih.

D XML i JSON se koriste samo za čuvanje, ne za razmenu podataka.

E Relaciona baza ne može da generiše JSON odgovor.

Razmišljajte o strukturi podataka, relacijama i načinu na koji će se podaci pretraživati.

PITALICA 5/5

Izaberite jedan tačan odgovor.

Baza ima tabele PORUDŽBINA, STAVKA_PORUDŽBINE i PROIZVOD. Šta backend najverovatnije radi kada aplikacija traži porudžbinu kao JSON?

A Vraća samo prvi red iz tabele PORUDŽBINA.

B JOIN-uje povezane tabele i formira JSON objekat sa nizom stavki.

C Pretvara celu bazu u jedan veliki JSON dokument.

D Briše strane ključeve jer JSON nema relacije.

E Pravi novu tabelu za svaki API zahtev.

Razmišljajte o strukturi podataka, relacijama i načinu na koji će se podaci pretraživati.