

REST

i JWT Sigurnost

Interoperabilnost
informacijskih sustava

Algebra Bernays University · 2025./2026.

Teme kolegija

- 01 Povijest i podrijetlo REST-a
- 02 Temeljni pojmovi i arhitektura
- 03 REST u usporedbi s drugim tehnologijama
- 04 Statistike i raširenost REST-a
- 05 CRUD akcije i HTTP metode
- 06 Integracija javnih REST API-ja
- 07 Integracija AI API-ja (Gemini / OpenAI)
- 08 JWT sigurnost i dobre prakse
- 09 Postavke kolačića i pohrana tokena
- 10 Demo: Spring Boot aplikacija

Povijest i podrijetlo REST-a

Slajd 02

1989

Tim Berners-Lee izumljuje World Wide Web na CERN-u. HTTP protokol i hiperveze čine prvi RESTful sustav.

1994

Roy Fielding sudjeluje u izradi HTTP/1.0 specifikacije. Prepoznaje arhitekturne prednosti weba koje će postati REST načela.

2000

Fielding u doktorskoj disertaciji na UC Irvine formalno definira REST (Representational State Transfer). REST nije arhitektura — stil je.

2002

Amazon lansira AWS REST API — prva velika komercijalna primjena. eBay iste godine slijedi s vlastitim REST API-jem.

2006

Twitter objavljuje REST API. Flickr i del.icio.us slijede. REST počinje dominirati nad SOAP-om u web servisima.

2010+

REST postaje de-facto standard. GitHub, Stripe, Twilio, Google, Facebook — svi grade REST API-je. RESTful je norma.

2020+

REST koegzistira s GraphQL-om i gRPC-om, ali i dalje pokreće >83% javnih API-ja širom svijeta.

Temeljni pojmovi i arhitektura REST-a

Klijent-Poslužitelj

Odvajanje korisničkog sučelja od pohrane podataka. Klijent i poslužitelj razvijaju se neovisno.

Bez stanja (Stateless)

Svaki zahtjev sadrži SVE potrebne informacije. Nema sesijskog stanja na poslužitelju.

Podrška za predmemoriju

Odgovori moraju biti označeni kao cacheable ili non-cacheable. Povećava performanse.

Jedinstveno sučelje

Standardizirani način interakcije: URI-ji, HTTP glagoli, reprezentacije, HATEOAS.

Slojeviti sustav

Klijent ne može znati je li izravno spojen na poslužitelj ili na posrednika (proxy).

Kod na zahtjev

(Opcionalno) Poslužitelj može proširiti funkcionalnost klijenta slanjem izvršnog koda.

REST vs SOAP vs GraphQL vs gRPC

Značajka	REST	SOAP	GraphQL	gRPC
Protokol	HTTP/HTTPS	Bilo koji (HTTP)	HTTP/HTTPS	HTTP/2
Format podataka	JSON/XML/ostalo	Samo XML	JSON	Protobuf (binarni)
Krivulja učenja	Niska ✓	Visoka ✗	Srednja	Srednja-visoka
Performanse	Dobre	Sporije (XML)	Dobre	Najbrže ⚡
Fleksibilnost	Visoka	Kruta	Vrlo visoka	Umjerena
Predmemorija	Ugrađena ✓	Ručna ✗	Složena	Ograničena
Tipna sigurnost	Ne (opcionalno)	Da (WSDL)	Schema	Jaka ✓
Idealno za	Javne API-je	Enterprise/Legacy	Složene upite	Mikroservise
Zrelost	Vrlo zrela ✓	Zrela ✓	Raste	Raste
Podrška brow.	Nativna ✓	Ograničena	Da	grpc-web

REST: Brojke koje govore

83%

javnih API-ja
koristi REST

25B+

REST API poziva
dnevno (2024.)

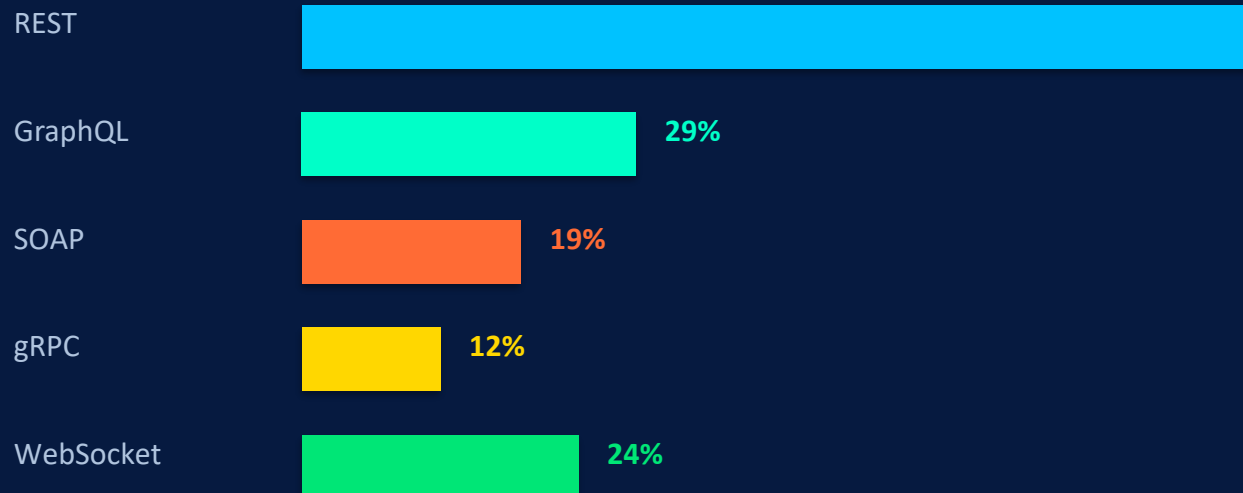
2,1T\$

vrijednost API
ekonomije do 2027.

10×

rast korištenja
REST API-ja 2018–24.

Zastupljenost arhitektura API-ja (2024.)



Gdje REST pokreće svijet?

-  Web aplikacije (React, Vue, Angular)
-  Mobilne aplikacije (iOS, Android)
-  AI i ML API-ji (OpenAI, Gemini, Anthropic)
-  Platni sustavi (Stripe, PayPal)
-  Cloud usluge (AWS, Azure, GCP)
-  IoT komunikacija uređaja
-  E-commerce platforme (Shopify, WooCommerce)
-  Platforme za podatke i analitiku

▶ [YouTube: API Economy explained](#)

CRUD Akcije i HTTP Metode

CRUD	HTTP	SQL	Primjer URI-ja	Kod uspjeha	Siguran?	Idempotent?
STVORI	POST	INSERT	/studenti	201 Created	Ne	Ne
ČITAJ	GET	SELECT	/studenti/{id}	200 OK	Da	Da
AŽURIRAJ	PUT	UPDATE	/studenti/{id}	200 OK	Ne	Da
ZAKRPI	PATCH	UPDATE	/studenti/{id}	200 OK	Ne	Da
OBRIŠI	DELETE	DELETE	/studenti/{id}	204 No Content	Ne	Da

Primjer: GET /api/studenti/42

```
GET /api/studenti/42 HTTP/1.1
Host: api.algebra.hr
Accept: application/json
Authorization: Bearer eyJhbGc...

← HTTP/1.1 200 OK
Content-Type: application/json
{"id":42,"ime":"Ana","dob":23}
```

Ključni HTTP statusni kodovi

- 200 OK — Uspjeh

400 Bad Request — Neispravan unos

403 Forbidden — Pristup odbijen

429 Too Many Req. — Prekoračen limit
- 201 Created — Resurs stvoren

401 Unauthorized — Autentikacija potrebna

404 Not Found — Resurs ne postoji

500 Server Error — Greška poslužitelja

Integracija javnih REST API-ja

Arhitektura integracije

1

Klijentska aplikacija

Vaša Spring Boot / frontend aplikacija

2

API Pristupnik

Ograničavanje brzine, autentikacija, usmjeravanje

3

REST Klijent

RestTemplate / WebClient / Feign

4

Javni API

GitHub / Stripe / Gemini / i dr.

5

Odgovor

JSON parsiran → domenski objekt

Popularni REST API-ji za integraciju

GitHub | Stripe | Twilio | OpenWeatherMap | Google Maps | Spotify

Spring Boot: Pozivanje GitHub API-ja

```
@Service
public class GitHubService {
    private final WebClient webClient;

    public GitHubService(WebClient.Builder builder) {
        this.webClient = builder
            .baseUrl("https://api.github.com")
            .defaultHeader("Authorization",
                "Bearer " + apiToken)
            .build();
    }

    public Mono<GitHubUser> getUser(String username) {
        return webClient.get()
            .uri("/users/{u}", username)
            .retrieve()
            .onStatus(HttpStatus::is4xxClientError,
                resp -> Mono.error(new ApiException()))
            .bodyToMono(GitHubUser.class);
    }
}
```

Integracija AI API-ja: Gemini i OpenAI

Google Gemini API (REST)

```
POST https://generativelanguage.googleapis.com
/v1beta/models/gemini-pro:generateContent
?key=VAŠ_API_KLJUČ

Tijelo zahtjeva (JSON):
{
  "contents": [{
    "parts": [{
      "text": "Objasni REST u 3 rečenice"
    }]
  }],
  "generationConfig": {
    "temperature": 0.7,
    "maxOutputTokens": 256
  }
}
```

Spring Boot Gemini servis

```
@Value("${gemini.api.key}")
private String apiKey;

public String pitajGemini(String prompt) {
    var tijelo = Map.of("contents", List.of(
        Map.of("parts", List.of(
            Map.of("text", prompt)))));

    return webClient.post()
        .uri("/v1beta/models/gemini-pro:generateContent?key="+apiKey)
        .bodyValue(tijelo)
        .retrieve()
        .bodyToMono(GeminiOdgovor.class)
        .map(r -> r.candidates().get(0)
            .content().parts().get(0).text())
        .block();
}
```

OpenAI Chat Completions API

```
POST https://api.openai.com/v1/chat/completions
Authorization: Bearer sk-...
Content-Type: application/json

Tijelo:
{
  "model": "gpt-4o",
  "messages": [
    {"role": "system",
     "content": "Ti si REST stručnjak."},
    {"role": "user",
     "content": "Objasni JWT tokene."}
  ],
  "temperature": 0.7,
  "max_tokens": 256
}
```

Usporedba AI API-ja

Davatelj	Model	Autentikacija	Besplatno
Gemini	gemini-pro/flash	API ključ	✔ Da
OpenAI	gpt-4o / gpt-4	Bearer Token	✘ Ne
Anthropic	claude-3.5-sonnet	x-api-key zaglavlje	✘ Ne
Mistral	mistral-large	Bearer Token	✘ Ne

JWT: JSON Web Token

Struktura JWT-a: zaglavlje.korisni_teret.potpis

ZAGLAVLJE	KORISNI TERET	POTPIS
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...	eyJzdWIiOiIxMjM0NTY3ODkwIn0...	Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQ...
<pre>{ "alg": "HS256", "typ": "JWT"}</pre>	<pre>{ "sub": "korisnik123", "name": "Ana Horvat", "role": "STUDENT", "iat": 1700000000, "exp": 1700003600}</pre>	<pre>HMACSHA256(base64(zaglavlje) + "." + base64(korisni_teret), tajniKljuc)</pre>

Standardni JWT zahtjevi (RFC 7519)

iss	Izdavatelj — Tko je izdao token	sub	Subjekt — Koga token predstavlja
aud	Publika — Tko smije primiti token	exp	Istjecanje — Kada token ističe (Unix vrijeme)
iat	Izdano u — Kada je token izdan	jti	JWT ID — Jedinstveni identifikator tokena

Access Token vs Refresh Token

ACCESS TOKEN

Svrha:	Omogućuje pristup zaštićenim resursima
Trajanje:	Kratko: 5–15 minuta (tipično)
Veličina:	Može biti veći (sadrži zahtjeve/claims)
Pohrana:	Memorija (JS varijabla) — NE localStorage
Šalje se:	Authorization: Bearer zaglavlje
Opoziv?:	✗ Nije lako (bez stanja / stateless)
Sadrži:	ID korisnika, uloge, ovlasti, exp
Po isteku:	Klijent mora koristiti Refresh Token
Sigurnost:	Obavezno HTTPS. Kratko TTL je ključno.

REFRESH TOKEN

Svrha:	Izdaje nove Access Tokenne bez ponovne prijave
Trajanje:	Dugo: 7–30 dana (ili klizno)
Veličina:	Neproziran (opaque) ili JWT
Pohrana:	HttpOnly Secure kolačić ☒
Šalje se:	Kolačić (automatski) ili POST tijelo
Opoziv?:	☒ Da — pohrana hash-a u bazi/Redisu
Rotacija:	Novi refresh token pri svakoj upotrebi
Po isteku:	Korisnik se mora prijaviti ponovo
Sigurnost:	HttpOnly, Secure, SameSite=Strict

Pohrana JWT-a i sigurnost kolačića

Usporedba mjesta pohrane

Lokacija	XSS rizik	CSRF rizik	JS pristup	Trajno?	Preporučeno za
Memorija (JS var.)	✓ Sigurno	✓ Sigurno	✓ Da	✗ Ne	Access Token ✓
localStorage	✗ VISOK	✓ Sigurno	✓ Da	✓ Da	✗ NIKAD za JWT
sessionStorage	✗ VISOK	✓ Sigurno	✓ Da	✗ Ne	⚠ Izbjegavaj
HttpOnly kolačić	✓ Sigurno	⚠ Srednji	✗ Ne	✓ Da	Refresh Token ✓
HttpOnly + SameSite	✓ Sigurno	✓ Sigurno	✗ Ne	✓ Da	Najbolja praksa ✓✓

Atributi sigurnosti kolačića

HttpOnly	Kolačić nedostupan JavaScriptu. Sprječava krađu XSS napadom.	✓ OBAVEZNO	SameSite=None	Kolačić na svim cross-site zahtjevima. Zahtijeva Secure zastavicu.	✗ Rizično
Secure	Kolačić se šalje samo putem HTTPS-a. Nikad plain HTTP.	✓ OBAVEZNO	Domain	Kontrolira koje domene primaju kolačić. Ograniči opseg!	⚠ Pažljivo
SameSite=Strict	Kolačić se NE šalje na cross-site zahtjeve. Najbolja CSRF zaštita.	✓ Optimalno	Path=/api	Ograničava kolačić na određenu putanju. Smanjuje izloženost.	✓ Dobro
SameSite=Lax	Kolačić se šalje na top-level GET navigacije. Dobra ravnoteža.	⚠ Dobro	Max-Age	Trajanje kolačića u sekundama. Postavi jednako TTL-u refresh tokena.	✓ Postavi

Životni ciklus JWT-a, rotacija i dobre prakse

Tijek životnog ciklusa tokena

► [YouTube: Hacking HTTP Only, Same Site, Secure cookies with XSS](#)

Prijava

POST /auth/login
{korisnik, lozinka}

Provjera

Provjera podataka
Generiranje tokena

Izdavanje

Access Token (15 min)
Refresh Token (7d)

Zahtjev

GET /api/podaci
Bearer: {access_token}

Istekao?

Access token
istekao → 401

Osvježavanje

POST /auth/refresh
Kolačić: refresh_token

Rotacija

Opozovi stari refresh
Izdaj novi par

Odjava

DELETE /auth/logout
Briši + opozovi

Sigurnosne dobre prakse

✓ Koristi RS256 (asimetrično) umjesto HS256 za višeservisne arhitekture

✓ Rotacija refresh tokena: novi refresh + opoziv starog pri svakoj upotrebi

✗ Nikad ne pohranjuj osjetljive podatke u JWT payload (base64, nije kriptirano!)

✗ Nikad ne pohranjuj JWT u localStorage — XSS ga može tiho ukrasti

✓ Uvijek validiraj: exp, iss, aud, algoritam, potpis

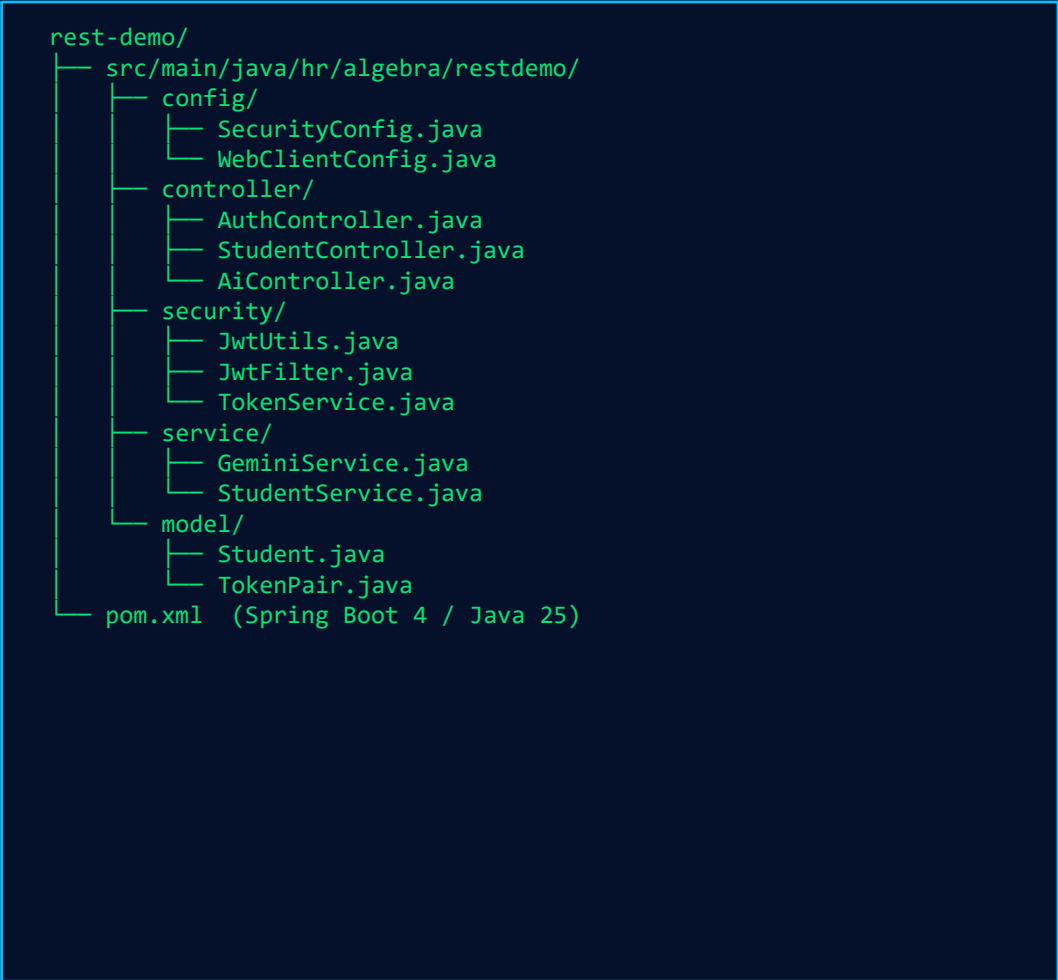
✓ Vodi listu opoziva refresh tokena (Redis/DB) radi podrške odjave

✗ Nikad ne koristi algorithm=none — uvijek eksplicitno bijeli popis algoritama

⚠ Koristi kratkoživuće access tokene + klizne sesije s refresh tokenima

Demo: Spring Boot 4 + JWT + AI API-ji

Arhitektura projekta



REST Krajnje točke (Endpoints)

POST	/api/auth/login	Autentikacija, dohvat para tokena
POST	/api/auth/refresh	Rotacija refresh tokena
DELETE	/api/auth/logout	Opoziv tokena, brisanje kolačića
GET	/api/studenti	Popis studenata [ROLE_STUDENT+]
GET	/api/studenti/{id}	Dohvat studenta po ID-u
POST	/api/studenti	Kreiranje studenta [ROLE_ADMIN]
PUT	/api/studenti/{id}	Ažuriranje studenta [ROLE_ADMIN]
DELETE	/api/studenti/{id}	Brisanje studenta [ROLE_ADMIN]
POST	/api/ai/pitaj	Slanje prompta Geminiju
GET	/api/github/{korisnik}	Dohvat GitHub profila korisnika

Tehnološki stack

Java 25 · Spring Boot 4 · Spring Security 7 · JJWT 0.12 · WebFlux · Redis

► [YouTube: JWT Authentication with Spring Security](#)

Sažetak i ključni zaključci

REST

Arhitekturni stil bez stanja, jedinstven, skalabilan. Pokreće 83% javnih API-ja.

HTTP metode

GET (čitaj), POST (stvari), PUT/PATCH (ažuriraj), DELETE (obriši)

Access Token

Kratkoživući JWT u memoriji. Nikad u localStorage. Šalje se kao Bearer.

Sigurn. kolačića

HttpOnly + Secure + SameSite=Strict je zlatni standard

Resursi

Sve je resurs dostupan putem URI-ja. Imenice, ne glagoli u putanjama!

AI API-ji

REST omogućuje jednostavnu AI integraciju: Gemini, OpenAI — ista načela

Refresh Token

Dugoživući. Pohrani u HttpOnly + Secure + SameSite kolačić. Rotiraj!

Nikad ne radi

Ne pohranjuj JWT u localStorage, ne koristi alg:none, nema tajni u payloadu

Hvala! Imate li pitanja?

▶ [YouTube: Top 12 Tips For API Security](#)