Postojeća implementacija satelitske obrade zasnovana na Copernicus servisima

Tehnički opis

Sadržaj

[0. Svrha dokumenta i kontekst 2](#_Toc221553131)

[0.1 Zašto je ovaj dokument napisan 2](#_Toc221553132)

[0.2 Uloga dokumenta u okviru šireg projekta 2](#_Toc221553133)

[0.3 Status rešenja opisanog u dokumentu 2](#_Toc221553134)

[0.4 Ciljna publika 3](#_Toc221553135)

[0.5 Kako koristiti ovaj dokument 3](#_Toc221553136)

# 0. Svrha dokumenta i kontekst

## 0.1 Zašto je ovaj dokument napisan

Ovaj dokument opisuje **postojeću tehničku implementaciju obrade satelitskih snimaka**, razvijenu kao inicijalno rešenje zasnovano na Copernicus servisima. Njegova osnovna svrha je da pruži **transparentan i precizan uvid u trenutno stanje sistema**, kako bi svi učesnici u projektu – interni tim i eksterni saradnici – imali zajedničko razumevanje:

* šta je do sada implementirano,
* kako je rešenje tehnički organizovano,
* koje servise i alate koristi,
* i koja su njegova trenutna ograničenja.

Dokument nije zamišljen kao promocija rešenja, već kao **tehnička osnova za evaluaciju i donošenje odluka** u narednim fazama projekta.

## 0.2 Uloga dokumenta u okviru šireg projekta

Ovaj dokument je **tehnički prateći dokument** uz glavni, konceptualni dokument:

**„Satelitski i dron snimci u Smart Agriculture – koncepti, arhitektura i prvi koraci“**.

Dok glavni dokument objašnjava *zašto* i *na koji način* se satelitski i dron snimci koriste u Smart Agriculture sistemima, ovaj dokument se fokusira isključivo na:

* *kako je jedna konkretna implementacija realizovana*,
* *koje sutehničke odluke donete*,
* *šta trenutno funkcioniše, a šta ne u potpunosti*.

Na taj način se jasno razdvaja:

* **konceptualni nivo** (dugoročno stabilan),
* od **implementacionog nivoa** (podložnog promenama).

## 0.3 Status rešenja opisanog u dokumentu

Rešenje koje je opisano u ovom dokumentu ima status:

* **inicijalne / pilot implementacije**,
* razvijene eksterno,
* korišćene kao polazna osnova za dalje testiranje i razvoj.

Važno je naglasiti da:

* sistem **nije finalna verzija platforme**,
* pojedinekomponentesueksperimentalne,
* deo funkcionalnosti je pogođen promenama verzija alata i servisa,
* očekuju se korekcije i poboljšanja u narednom periodu.

Cilj dokumentovanja trenutnog stanja je da se izbegne „implicitno znanje“ i da se omogući **objektivna tehnička diskusija** o daljem razvoju.

## 0.4 Ciljna publika

Dokument je namenjen sledećim grupama:

* tehničkim članovima tima koji će raditi na instalaciji, testiranju i eventualnoj modifikaciji rešenja,
* arhitektama i donosiocima odluka koji treba da procene koje komponente zadržati, a koje zameniti,
* eksternim saradnicima uključenim u razvoj i održavanje sistema.

Dokument **nije namenjen krajnjim korisnicima** (poljoprivrednicima) niti predstavlja korisničku dokumentaciju.

## 0.5 Kako koristiti ovaj dokument

Dokument treba čitati:

1. zajedno sa glavnim konceptualnim dokumentom,
2. kao snimak trenutnog tehničkog stanja,
3. kao osnovu za:
   * instalaciju rešenja u internom okruženju,
   * testiranje funkcionalnosti,
   * identifikaciju tačaka za unapređenje ili refaktorisanje.

Svi budući razvojni koraci (zamena servisa, uvođenje sopstvenih modula, integracija dronova i senzora) **ne zavise direktno od ove konkretne implementacije**, već od zaključaka koji se iz nje izvuku.

# 1.Kratak funkcionalni opis sistema (šta sistem trenutno radi)

**Napomena:** Ovo poglavlje ima za cilj da opiše *funkcionalno ponašanje sistema iz ugla korisnika i operativnog toka*, bez ulaska u detalje implementacije. Tehnički detalji (servisi, API-ji, alati) biće obrađeni u narednim poglavljima.

## 1.1 Opšti opis funkcionalnosti

**Šablon (tekst koji se popunjava):**

Trenutno implementirano rešenje omogućava obradu satelitskih snimaka za definisane poljoprivredne parcele i generisanje osnovnih indikatora stanja useva. Sistem je namenjen za inicijalno testiranje koncepta i validaciju pristupa zasnovanog na Copernicus servisima.

Na visokom nivou, sistem obavlja sledeće funkcije:

* prihvata prostorne podatke o parcelama (AOI),
* preuzima ili obrađuje odgovarajuće satelitske snimke za izabrani vremenski period,
* izračunava definisane vegetacione indekse,
* generišeagregiranerezultate po parceli,
* omogućava osnovni uvid u rezultate kroz vizuelizaciju ili izvoz podataka.

**Pitanja za autora (Sinišu):**

* Kako bi ti u jednoj–dve rečenice opisao *šta sistem radi danas*?

Sistem automatski preuzima Sentinel-2 satelitske snimke iz Copernicus servisa, izračunava vegetacione indekse (NDVI, NDMI, NDRE) sa SCL filterom za maskiranje oblaka, i generiše zone za varijabilnu prihranu azota koje se vizualizuju na interaktivnoj web mapi sa mogućnošću klika na parcelu za detaljne vrednosti indeksa i preporuke za đubrenje.

* Da li sistem trenutno ima interfejs (UI), API, ili oba?

Oba.

**UI (Web Interface):**

Frontend: Leaflet mapa (http://localhost:8088/leaflet\_demo.html)

Funkcionalnost:

Izbor opštine i parcele

Prikaz WMS layera (NDVI, NDMI, NDRE, NDRE Zones)

Klik na mapu za prikaz vrednosti indeksa

Info balon sa agronomskim preporukama

CSV podaci i grafikoni vremenske serije

**API (HTTP):**

Parcel Server: http://localhost:5010

Endpoint-i:

/run?parcel={id}&layer={layer} - generiši NDVI raster

/ndmi?parcel={id}&layer={layer} - generiši NDMI raster

/ndre?parcel={id}&layer={layer} - generiši NDRE raster

/csv?parcel={id}&days={n}&cloud={%}&layer={layer} - generiši NDVI CSV

/ndmi\_csv?parcel={id}&days={n}&cloud={%}&layer={layer} - generiši NDMI CSV

/ndre\_csv?parcel={id}&days={n}&cloud={%}&layer={layer} - generiši NDRE CSV

**WMS/WFS:**

GeoServer: http://localhost:8083/geoserver

WMS: Služi raster layere (indeksi, zone)

WFS: Služi vektorske podatke (parcele)

* Da li se sistem koristi interaktivno ili isključivo kao batch obrada?

Oba načina korišćenja:

**Interaktivno (primarno):**

Frontend: Korisnik unosi ID parcele, klikne dugme za prikaz indeksa

On-demand generisanje: Kada korisnik zatraži NDRE za parcelu, backend poziva Copernicus API i generiše raster/CSV u realnom vremenu

Klik na mapu: GetFeatureInfo upiti vraćaju vrednosti u realnom vremenu

**Batch obrada (sekundarno):**

Docker kontejner: ndvi\_updater periodično ažurira globalne indekse

Ručne komande: Admin može pokrenuti generisanje za više parcela odjednom

docker exec -e PARCEL\_ID="1427/2" parcel\_server python /app/download\_ndre\_parcel.py

Scheduled updates: UPDATE\_INTERVAL\_MINUTES=1440 (dnevno)

**Tipičan workflow:**

Interaktivno: Korisnik zatraži parcelu → sistem generiše podatke → prikaz na mapi

Batch: Noću se ažuriraju globalni indeksi za sve parcele (opciono)

**Dodatne napomene za dokumentaciju:**

Tehnička arhitektura:

Docker Compose: 6 kontejnera (PostgreSQL, GeoServer, Nginx, Python backend, parcel server)

Backend: Python 3.13 (rasterio, shapely, numpy, requests)

Frontend: Leaflet.js + HTML/CSS/JavaScript

Data source: Copernicus DataSpace Ecosystem API (Sentinel-2 L2A)

Storage: GeoTIFF rasteri, CSV podaci, PostGIS baza

**Ključne funkcionalnosti:**

Cloud masking: SCL filter (filtrira oblake, saturated piksele)

Dual-layer pristup: RGB raster za vizualizaciju + FLOAT32 raster za tačne vrednosti

Zone mapping: 3 zone za varijabilnu prihranu azota (0.14, 0.19 pragovi)

Time series: CSV podaci sa statistikama za poslednje 30-120 dana

Testirana konfiguracija:

Opština: Kovin

Parcela: 1427/2

Indeksi: NDVI, NDMI, NDRE (+ NDRE Zones)

Period: 90 dana (fallback na 120 dana)

## 1.2 Ulazni podaci

**Šablon:**

Sistem kao ulazne podatke koristi sledeće informacije:

* prostorni opis parcela (format, način unosa),
* vremenski opseg obrade (jedan datum ili vremenski interval),
* dodatne parametre potrebne za pokretanje obrade (ako postoje).

Ulazni podaci se koriste kao osnova za izbor odgovarajućih satelitskih snimaka i dalju obradu.

**Pitanja za autora:**

* U kom formatu se unose parcele (GeoJSON, WKT, drugi)?

Format u bazi:

PostGIS baza: Parcele su skladištene kao PostGIS geometrije (POLYGON/MULTIPOLYGON)

GeoServer WFS: Parcele se izlažu kao GeoJSON preko WFS servisa

CRS: EPSG:4326 (WGS84 geografske koordinate)

Format razmene sa Copernicus API:

GeoJSON: Geometrija parcele se preuzima iz PostGIS-a i konvertuje u GeoJSON format

WKT (Well-Known Text): Interno se koristi za PostGIS upite

Bounding Box: Ekstrahuje se iz geometrije za Copernicus API zahteve

* Da li se parcele unose ručno ili se učitavaju iz postojećeg sistema?

Učitavaju se iz postojećeg sistema (PostGIS baze):

Trenutna implementacija:

Inicijalno učitavanje: Parcele su prethodno učitane u PostGIS bazu (jednokratno, batch import iz DKP-a. Učitane su opštine Vršac, Kovin i Pančevo)

Korisnikov unos: Korisnik unosi samo ID parcele (npr. "1427/2") u web interfejs

Automatsko preuzimanje geometrije: Sistem automatski dobavlja geometriju iz baze preko GeoServer WFS

* Kako se definiše vremenski period obrade?

Trenutna implementacija - 3 nivoa definisanja:

1. Environment Variable (Default)

2. Frontend (Hardkodovano)

3. API parametar (Dinamički)

## 1.3 Izlazni podaci i rezultati

**Šablon:**

Kao rezultat obrade, sistem proizvodi sledeće izlazne podatke:

* vegetacioneindekse (navesti koji),
* agregiranestatistike po parceli,
* eventualne vizuelne prikaze (mape, grafikone),
* mogućnost izvoza podataka u datoteke ili kroz API.

Izlazni podaci predstavljaju osnovu za dalju analizu i evaluaciju sistema.

**Pitanja za autora:**

* Koje indekse sistem trenutno računa (NDVI, NDRE, NDMI…)?

### 4 vegetaciona indeksa + zone mapping:

### 1. NDVI (Normalized Difference Vegetation Index)

* **Formula:** (NIR - Red) / (NIR + Red) = (B08 - B04) / (B08 + B04)
* **Opseg:** -1.0 do +1.0 (tipično 0.0 - 0.8 za vegetaciju)
* **Značenje:** Opšti pokazatelj zelene biomase i zdravlja useva
* **Interpretacija:**
* < 0.2: Gola zemlja / vrlo slaba vegetacija
* 0.2 - 0.4: Retka vegetacija / stres
* 0.4 - 0.6: Umerena vegetacija
* > 0.6: Zdrava, gusta vegetacija

### 2. NDMI (Normalized Difference Moisture Index)

* **Formula:** (NIR - SWIR1) / (NIR + SWIR1) = (B08 - B11) / (B08 + B11)
* **Opseg:** -1.0 do +1.0 (tipično -0.2 do +0.4 za vegetaciju)
* **Značenje:** Pokazatelj sadržaja vlage u biljkama
* **Interpretacija:**
* < -0.2: Vrlo suvo
* -0.2 - 0.0: Suvo
* 0.0 - 0.2: Umereno
* > 0.2: Vlažno

### 3. NDRE (Normalized Difference Red Edge)

* **Formula:** (NIR - Red Edge) / (NIR + Red Edge) = (B08 - B05) / (B08 + B05)
* **Opseg:** -1.0 do +1.0 (tipično 0.0 - 0.3 za vegetaciju)
* **Značenje:** Pokazatelj sadržaja azota / hlorofila u biljkama
* **Interpretacija:**
* < 0.1: Niska vegetacija / manjak azota
* 0.1 - 0.2: Umerena vegetacija / standardan azot
* > 0.2: Dobra vegetacija / visok azot

### 4. NDRE Zones (Nitrogen Management Zones)

* **Baziran na:** NDRE vrednostima
* **3 zone za varijabilnu prihranu azota:**
* **Crvena (< 0.14):** Više azota - može manje đubrenja
* **Žuta (0.14 - 0.19):** Standardna količina azota
* **Zelena (≥ 0.19):** Manje azota - može više đubrenja

### Cloud Masking (SVE indeksi):

* **SCL Filter:** Automatsko filtriranje oblaka, saturated piksela
* **Filtrirane SCL vrednosti:** 0 (no data), 1 (saturated), 8 (oblaci srednja verovatnoća), 9 (oblaci visoka verovatnoća)
* Da li se vraćaju rasteri, numeričke vrednosti ili oba?

### Oba - 5 tipova izlaznih podataka:

### A) RASTERI (GeoTIFF)

#### 1. RGB Vizualizacioni rasteri (3-band UINT8)

* **Namena:** Prikaz na mapi sa bojama
* **Output:** RGB boje sa kontrast stretching-om

#### 2. Value rasteri (1-band FLOAT32)

* **Namena:** Tačne numeričke vrednosti za GetFeatureInfo
* **Output:** Sirove indeks vrednosti bez procesiranja

### B) STATISTIKE (CSV)

Statistike po redu:

date: Datum snimka (ISO 8601)

min/max: Minimalna/maksimalna vrednost indeksa

mean: Prosečna vrednost

stDev: Standardna devijacija

sampleCount: Broj validnih piksela

noDataCount: Broj filtriranih piksela (oblaci, no data)

median: Medijana vrednosti

p10/p90: 10. i 90. percentil

cloudCoveragePercent: Procenat oblaka na sceni

### C) TAČKASTE VREDNOSTI (GetFeatureInfo)

### D) GEOMETRIJE (GeoJSON/WFS)

### E) VIZUELIZACIJE (WMS PNG)

* U kom formatu su izlazni podaci (JSON, CSV, GeoTIFF, PNG)?

| **Tip podatka** | **Format** | **Primer fajla / URL** | **Veličina** |
| --- | --- | --- | --- |
| **RASTERI**  **(RGB)** | GeoTIFF | ndre\_parcel\_1427\_2.tif | 1-5 KB |
| **RASTERI (Value)** | GeoTIFF | ndre\_value\_parcel\_1427\_2.tif | 1-5 KB |
| **Statistike** | CSV | parcela\_1427\_2\_NDRE.csv | 500-2000 bytes |
| **WMS Slike** | PNG | http://.../wms?...&FORMAT=image/png | 5-50 KB |
| **Parcele** | GeoJSON | http://.../wfs?...&outputFormat=application/json | 1-10 KB |
| **GetFeatureInfo** | JSON | http://.../wms?REQUEST=GetFeatureInfo&INFO\_FORMAT=application/json | < 1 KB |
| **Metadata** | JSON | latest\_metadata.json | < 1 KB |

## 1.4 Stepen automatizacije

**Šablon:**

Sistem je trenutno automatizovan na sledećem nivou:

* pokretanje obrade (ručno / polu-automatski / automatski),
* izbor snimaka (ručni ili automatski),
* ponavljanje obrade za više parcela ili datuma.

Određeni koraci i dalje mogu zahtevati manuelnu intervenciju, što je prihvatljivo u fazi pilot implementacije.

**Pitanja za autora:**

* Šta se danas pokreće ručno, a šta automatski?

## Rezime - Nivo automatizacije:

| **Proces** | **Ručno** | **Polu-Auto** | **Potpuno Auto** |
| --- | --- | --- | --- |
| **Izbor parcele** | ✅ (korisnik unosi) | ❌ | ❌ |
| **Izbor snimka** | ❌ | ❌ | ✅ (Copernicus API) |
| **Filtriranje oblaka** | ❌ | ❌ | ✅ (SCL filter) |
| **Izračunavanje indeksa** | ❌ | ❌ | ✅ (evalscript) |
| **Generisanje rastera** | ❌ | ❌ | ✅ (backend) |
| **Upload u GeoServer** | ❌ | ❌ | ✅ (backend) |
| **Periodično ažuriranje** | ❌ | ✅ (scheduler) | ❌ |
| **Detekcija novih snimaka** | ❌ | ✅ (periodično) | ❌ |
| **Batch obrada parcela** | ✅ (shell script) | ❌ | ❌ |
| **Notifikacije** | ❌ | ❌ | ❌ |

* Da li postoji scheduler ili batch proces?

Postoji scheduler, ali ograničen:

### Šta radi scheduler:

* ✅ Periodično ažurira globalne indekse (NDVI, NDMI, NDRE za celu oblast)
* ✅ Automatski restart nakon greške
* ✅ Konfigurabilan interval (minutima)

### Šta NE radi scheduler:

* ❌ Ne ažurira pojedinačne parcele automatski
* ❌ Ne proverava nove snimke proaktivno (samo u definisanim intervalima)
* ❌ Ne šalje notifikacije o novim snimcima
* ❌ Nema prioritizaciju parcela
* Da li sistem može sam da pronađe novi snimak kada se pojavi?
* ✅  **Sistem periodično proverava** (dnevno, nedeljno, mesečno) Copernicus snimke
* ✅  **Automatski uzima najnoviji snimak** u tom trenutku
* ⚠  **NE reaguje odmah** kada novi snimak postane dostupan
* ⚠  **Čeka interval** (npr. 24h) pre sledeće provere

### Scenario 3: Event-Driven (NIJE implementirano)

## 1.5 Tipičan tok rada (high-level)

**Šablon:**

Tipičan tok rada sistema može se opisati sledećim koracima:

1. Definisanje ili izbor parcele,
2. Definisanje vremenskog perioda,
3. Pokretanje obrade,
4. Prikupljanje i obrada satelitskih podataka,
5. Generisanje i prikaz rezultata.

Ovaj tok rada predstavlja osnovni scenario korišćenja sistema.

**Pitanja za autora:**

* Da li je ovaj tok rada realan u odnosu na trenutnu implementaciju?

DA - potpuno realan.

* Postoje li alternativni tokovi (npr. masovna obrada više parcela)?

DA - postoje 3 alternativna toka:

ALTERNATIVNI TOK 1: Batch obrada više parcela (Shell Script)

Scenario: Admin želi da obradi 10 parcela odjednom

ALTERNATIVNI TOK 2: Scheduled Update (ndvi\_updater)

Scenario: Sistem automatski ažurira globalnu oblast dnevno

ALTERNATIVNI TOK 3: API-Driven Workflow

Scenario: Eksterni sistem integrisan sa parcel\_server API-jem

## 1.6 Ograničenja trenutne funkcionalnosti

**Šablon:**

Trenutna funkcionalnost sistema ima sledeća poznata ograničenja:

* ograničen broj indeksa ili metrika,
* ograničenja u pogledu performansi ili stabilnosti,
* zavisnost od konkretnih verzija alata ili servisa.

Ova ograničenja su očekivana u fazi inicijalne implementacije i predstavljaju ulazne tačke za dalji razvoj.

**Pitanja za autora:**

* Koja ograničenja bi ti trenutno istakao kao najvažnija?

### A) FUNKCIONALNA OGRANIČENJA

#### 1. Ograničen broj indeksa i metrika

**Trenutno implementirano:**

* ✅ NDVI (Vegetation Index)
* ✅ NDMI (Moisture Index)
* ✅ NDRE (Red Edge / Nitrogen Index)
* ✅ NDRE Zones (3-zone mapping)

**Nije implementirano (ali bi bilo korisno):**

* ❌  **EVI** (Enhanced Vegetation Index) - bolje za guste useve
* ❌  **SAVI** (Soil Adjusted Vegetation Index) - za golu zemlju
* ❌  **LAI** (Leaf Area Index) - površina lišća
* ❌  **MSAVI** (Modified SAVI) - za raniji stadijum useva
* ❌ **NDWI** (Water Index) - za vodna tela
* ❌  **Biomass estimation** - procena biomase
* ❌  **Chlorophyll content** - direktna procena hlorofila
* ❌  **Yield prediction** - prognoza prinosa

**2. Ograničenja obrade satelitskih snimaka**

Sentinel-2 specifična ograničenja:

Rezolucija: 10m (B02, B03, B04, B08)

20m (B05, B11) - moguća resampling greška

Revisit time: 5 dana (može biti gap u podacima)

Latencija: 2-3 dana (snimak nije odmah dostupan)

#### 3. Pojedinačna parcela po zahtevu

**Trenutno:**

* ✅ Korisnik obrađuje **1 parcelu odjednom**
* ❌  **Nema batch UI** za izbor više parcela
* ❌  **Nema priority queue** sistema
* ❌  **Nema paralelizacije** - sekvencijalno procesiranje

### B) PERFORMANSE I STABILNOST

#### 1. Copernicus API latencija i pouzdanost

**Performanse:**

Prosečno vreme odgovora: 10-60 sekundi po parceli

Najbolji slučaj: 10 sekundi (mali parcela, keširani token)

Najgori slučaj: 120+ sekundi (veliki parcela, fallback, timeout)

**Stabilnost - Poznati problemi:Problem 1: API vraća 0 data items**

# Često se dešava za kratke periode (30 dana)

[INFO] Received 0 data items from API

[DEBUG] NDRE API vratio prazan data niz! Odgovor: {"data": [], "status": "OK"}

**Uzrok:** Nema snimaka u periodu koji zadovoljavaju kriterijume (cloud cover)

**Workaround:** Automatski fallback na 90-120 dana

**Problem 2: Token expiration**

**Rešenje:** Token refresh nije implementiran

**Problem 3: Rate limiting**

Rešenje: Nema retry sa exponential backoff

#### 2. GeoServer performanse

**GetFeatureInfo latencija:**

Prosečno: 50-200ms

Problem: Nema caching-a

Rešenje: Frontend bi mogao da kešira vrednosti za već kliknute tačke

### C) ZAVISNOSTI OD ALATA I SERVISA

#### 1. Eksterni servisi (Critical Dependencies)

| **Servis** | **Verzija** | **Dostupnost** | **Rizik** |
| --- | --- | --- | --- |
| **Copernicus DataSpace** | API v1 | 99%+ uptime | **KRITIČAN** - Bez ovoga sistem ne radi |
| **Sentinel-2 L2A** | Current | Kontinuirano | **SREDNJI** - Gap u podacima moguć |
| **OAuth2 (Copernicus)** | - | Zavisi | **VISOK** - Token expiration problem |

**Problem: Single point of failure**

Copernicus API down → Sistem ne može da generiše nove podatke

Rešenje: Nema lokalnog backup-a snimaka

#### 2. Docker kontejneri

**Verzije:**

python:3.13-slim # Backend

kartoza/postgis:15-3.3 # Baza

kartoza/geoserver:latest # GeoServer (nestabilna verzija!)

nginx:alpine # Frontend

**Problem sa latest tag-om:**

kartoza/geoserver:latest # Može se promeniti bez upozorenja!

Rizik: Breaking changes u novijoj verziji

Rešenje: Pinuj na konkretnu verziju (npr. kartoza/geoserver:2.24.0)

#### 3. Python biblioteke

**requirements.txt (implicitno):**

rasterio # GDAL wrapper

shapely # Geometrije

numpy # Numeričke operacije

requests # HTTP klijent

pyproj # Projekcije

**Problem: Nema lock fajla**

Nema requirements.txt sa verzijama!

Rizik: pip install može instalirati nekompatibilne verzije

Rešenje: Kreirati requirements.txt sa pinned verzijama

### D) KORISNIČKO ISKUSTVO I VALIDACIJA

#### 1. Nema validacije input-a

**Frontend:**

**Problem:**

* ❌ Nema regex validacije (dozvoljeni formati: "1427/2", "2511", "158\_2")
* ❌ Nema provere da li parcela postoji pre poziva API-ja
* ❌ Nema jasnih error poruka ("Parcela ne postoji u sloju kovin\_dkp\_pg")

#### 2. Error handling

**Frontend - Loš error handling:**

.catch(function(err) {

console.error("Greška:", err); // Samo u konzoli!

// Nema UI notifikacije za korisnika

});

**Backend - Samo console log:**

except Exception as exc:

print(f"[ERROR] {exc}") # Samo u Docker log-u

sys.exit(1)

# Nema structured logging, nema alerting

#### 3. Nema progress bar-a

**Problem:**

Korisnik klikne "Prikaži NDRE" → čeka 30-60 sekundi → ?

Nema indikatora da li sistem radi ili je zamrznut

**Rešenje bi bilo:**

// Prikaži spinner tokom obrade

showSpinner("Preuzimanje satelitskih podataka...");

fetch(url)

.then(...)

.finally(() => hideSpinner());

### E) SKALABILNOST

#### 1. Jedna parcela u jednom trenutku

**Ograničenje:**

Paralelizacija: 1 (nije moguća batch obrada u GUI)

Throughput: ~1-2 parcele/minut

Za 100 parcela: 100-200 minuta (1.5-3.5h)

**Rešenje (hipotetički):**

# Python multiprocessing

from concurrent.futures import ThreadPoolExecutor

parcels = ["1427/2", "2511", "2542", ...]

with ThreadPoolExecutor(max\_workers=5) as executor:

executor.map(process\_parcel, parcels)

# 5x brže!

#### 2. GeoServer memorija

**Problem:**

Broj rastera u GeoServeru: Trenutno ~50-100

Svaki raster: ~1-5 KB

Ukupno: ~100-500 KB (OK)

Ali za 1000 parcela:

Ukupno: ~5-50 MB (može biti problem sa RAM-om)

**Rešenje:** Periodično čišćenje starih rastera

### F) GEOGRAFSKA I PROSTORNA OGRANIČENJA

#### 1. Ograničena oblast pokrivanja

**Trenutno:**

Opštine: Kovin, Vršac, Pančevo (Srbija, Vojvodina)

CRS: EPSG:4326 (hardkodovano)

**Problem za skaliranje:**

* ❌ Nema dinamičkog dodavanja novih opština
* ❌ Nema podrške za druge CRS-ove (npr. EPSG:3857, lokalni CRS)
* ❌ Nema generičkog pristupa (bilo koja oblast u svetu)

#### 2. Sentinel-2 rezolucija

**Ograničenje:**

10m rezolucija = 100m² po pikselu

Mala parcela (< 0.5 ha) = ~50 piksela

Problem: Slaba statistička značajnost za male parcele

## 2. Šta trenutno ne radi pouzdano ili zahteva dodatnu intervenciju?

**Odgovor:**

### NAJVEĆI PROBLEMI (prioritet):

#### 🔴 1. Copernicus API nepouzdanost podataka

**Simptom:**

[INFO] Requesting NDRE stats from 2026-01-09 to 2026-02-09 (30 days)

[INFO] Received 0 data items from API

**Učestalost:** ~30-50% upita vraća 0 podataka za period od 30 dana

**Zahteva intervenciju:**

* ✅ Ručno povećanje perioda na 90-120 dana
* ✅ Ručno pokretanje ponovo
* ✅ Provera da li je oblast pokrivena Sentinel-2

**Rešenje bi bilo:**

* Automatski retry sa različitim parametrima
* Bolja error poruka korisniku: "Nema snimaka za ovaj period. Probajte sa dužim periodom (90 dana)."

#### 🟡 2. Layer name zavisnost

**Problem:**

Parcela 1427/2 je u "kovin\_dkp\_pg", ali sistem defaultuje na "VrsacDKP"

Rezultat: Greška "[ERROR] Parcel 1427/2 not found in moj\_projekat:vrsac\_dkp\_pg"

**Zahteva intervenciju:**

* Korisnik mora RUČNO da izabere **tačnu opštinu** pre unosa parcele
* Ako pogriješi → greška 500 i mora ponovo – *ovo je zato što je agent ’griješi’ i glupo u startu podrazumeva da su sve parcele u opštini Vršac i po defaultu misli da to treba tako pošto je u početku testirano na teritoriji te opštine – a rešava se tako što svaka opština ima svoje parcele a već je u interfejsu predviđeno da se unese i naziv opštine i broj parcele – naravno da korisnik mora ručno da unese (ili klikne u padajućem meniju) u kojoj opštini mu je parcela (napomena autora).*

**Rešenje bi bilo:**

* Automatska detekcija layera (probaj sve layere dok ne pronađeš parcelu) – *glupost (napomena autora)*
* Frontend smart search sa sugestijama – *ovo već postoji, smart search po svim parcelama u okviru opštine ako korisnik unese opštinu (napomena autora)*

#### 🟡 3. Nema validacije parcel ID formata

**Problem:**

// Frontend prima bilo šta:

parcelInput.value = "abc123!@#" // Nema validacije!

// Backend pada sa:

[ERROR] Parcel abc123!@# not found in moj\_projekat:kovin\_dkp\_pg

**Zahteva intervenciju:**

* Korisnik unese pogrešan format → greška → mora ponovo uneti

**Rešenje bi bilo:**

function validateParcelId(id) {

// Dozvoljeni formati: "1427/2", "2511", "158\_2"

var regex = /^\d+([/\_]\d+)?$/;

if (!regex.test(id)) {

alert("Pogrešan format! Primeri: 1427/2, 2511, 158\_2");

return false;

}

return true;

}

#### 🟡 4. Hardkodovani vremenski periodi

**Problem:**

// Frontend - nema izbora perioda

requestParcelNdvi(parcelId); // Uvek 30 dana

requestParcelNdre(parcelId); // Uvek 90 dana

**Zahteva intervenciju:**

* Ako korisnik želi drugačiji period → mora menjati kod ili koristiti API direktno

**Rešenje bi bilo:**

<select id="periodSelect">

<option value="30">30 dana</option>

<option value="60">60 dana</option>

<option value="90">90 dana</option>

<option value="120">120 dana</option>

</select>

#### 🟢 5. CSV struktura zavisi od Copernicus API

**Problem:**

C0/date,C0/min,C0/max,C0/mean,...

**Copernicus API vraća ovaj format - ako se promeni, sistem pada**

**Zahteva intervenciju:**

* Ako Copernicus promeni API format → CSV parsing pada → treba adaptacija koda

**Rešenje bi bilo:**

* Normalizacija podataka u intermedijaran format
* Verzionisanje API zahteva

SEKUNDARNI PROBLEMI:

6. Nema autentifikacije/autorizacije

http://localhost:5010/ndre?parcel=...

→ Svako na local network-u može pristupiti!

7. Nema export funkcionalnosti u GUI

Korisnik ne može da downloaduje:

- GeoTIFF raster (direktno)

- Screenshot mape

- Agregirani report (PDF/Excel)

8. Docker Desktop zavisnost

Windows: Mora Docker Desktop (licenca za korporativne korisnike!)

Alternative: Podman, WSL2 native

9. Nema error recovery

*# Ako Copernicus API vrati grešku, sistem pada*

*# Nema retry logike sa exponential backoff*

10. Single-threaded obrada

*# parcel\_server.py koristi BaseHTTPRequestHandler*

*# Problem: Samo 1 zahtev odjednom!*

*# Ako 2 korisnika kliknu istovremeno → jedan čeka*

G) TEHNIČKI DUG

1. Bez requirements.txt sa verzijama

*# Dockerfile*

RUN pip install rasterio shapely numpy requests pyproj

*# Problem: Koja verzija? Latest može biti breaking change!*

2. Hardkodovani URL-ovi

***// leaflet\_demo.html***

**var geoserverUrl = 'http://localhost:8083/geoserver/...';**

***// Problem: Ne može deployment na drugi server bez izmene koda***

3. Nema testova

**Nema unit testova**

**Nema integration testova**

**Nema end-to-end testova**

**→ Regression bug-ovi mogući!**

4. Nema structured logging

**print(f"[INFO] {message}")  *# Samo console print***

***# Nema:***

***# - JSON logging***

***# - Log levels (DEBUG, INFO, WARN, ERROR)***

***# - Log agregacija (ELK stack, Loki)***

## PRIORITET OGRANIČENJA ZA REŠAVANJE:

### KRITIČNO (Blokira produkciju):

1. **Copernicus API nepouzdanost** (0 data items)
2. **Single-threaded parcel server** (ne skalira)
3. **Nema autentifikacije** (bezbednosni rizik)

### VAŽNO (Utiče na UX):

1. **Nema batch UI** (spolja obrada više parcela)
2. **Nema validacije input-a** (loše error poruke)
3. **Hardkodovani periodi** (nema fleksibilnosti)

### 🟢 NICE TO HAVE (Poboljšanja):

1. **Više indeksa** (EVI, SAVI, LAI)
2. **Export funkcionalnost** (PDF report, GeoTIFF download)
3. **Progress bar** (bolji UX)
4. **Testovi i CI/CD**

## ZAKLJUČAK:

### Što trenutno zahteva dodatnu intervenciju:

1. **Ručan izbor opštine** pre unosa parcele (mora biti tačan!) - *???? (napomena autora)*
2. **Ručno pokretanje** generisanja za svaku parcelu pojedinačno
3. **Ručan restart Docker kontejnera** ako padne
4. **Ručna fallback strategija** ako nema podataka (probaj duži period)
5. **Ručno čišćenje starih rastera** u GeoServeru (nema garbage collection)

### Što radi pouzdano:

✅ Izbor najboljih snimaka (Copernicus API)

✅ SCL filtriranje oblaka

✅ Izračunavanje indeksa (evalscripti)

✅ Generisanje rastera i CSV-a

✅ Prikaz na mapi (Leaflet + GeoServer)

✅ GetFeatureInfo vrednosti (dual-layer pristup)

**Ukratko:** Sistem je **funkcionalan za pilot**, ali zahteva **više automatizacije i error handling-a za produkciju**. 🎯

* Šta trenutno ne radi pouzdano ili zahteva dodatnu intervenciju?

Pogledati odgovor na prethodno pitanje

# 2.Visokonivojska(?) arhitektura postojećeg sistema

**Napomena:** Ovo poglavlje opisuje *kako je sistem tehnički organizovan na visokom nivou*, bez ulaska u detalje pojedinačnih API poziva, konfiguracija ili koda. Cilj je da čitalac stekne jasnu sliku glavnih komponenti i toka podataka.

## 2.1 Pregled arhitekture (big picture)

**Šablon:**

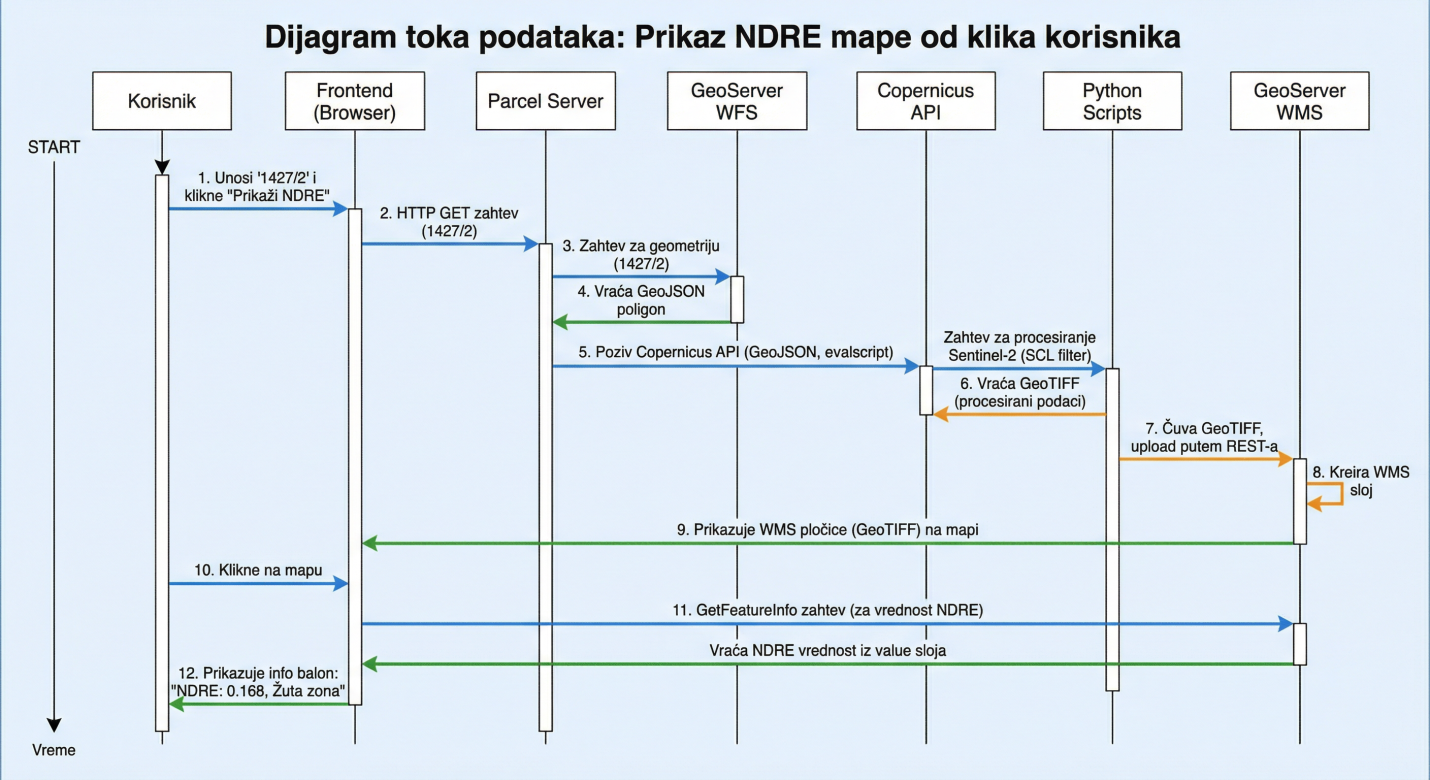
Postojeće rešenje je organizovano kao skup logičkih komponenti koje zajedno omogućavaju obradu satelitskih snimaka i generisanje rezultata po parcelama. Sistem se oslanja na kombinaciju eksternih servisa (Copernicus ekosistem) i lokalnih modula za orkestraciju, obradu i prikaz rezultata.

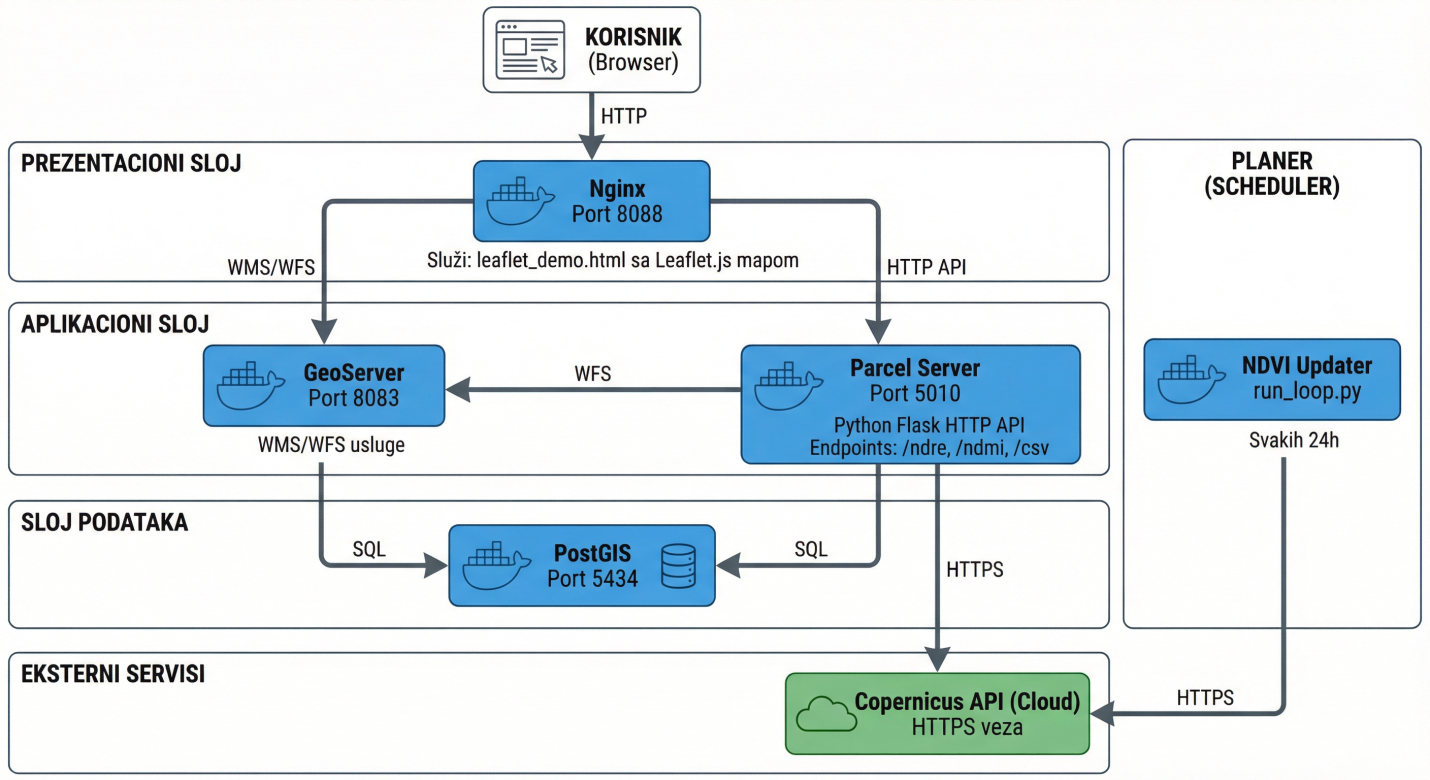
Na visokom nivou, arhitektura obuhvata sledeće celine:

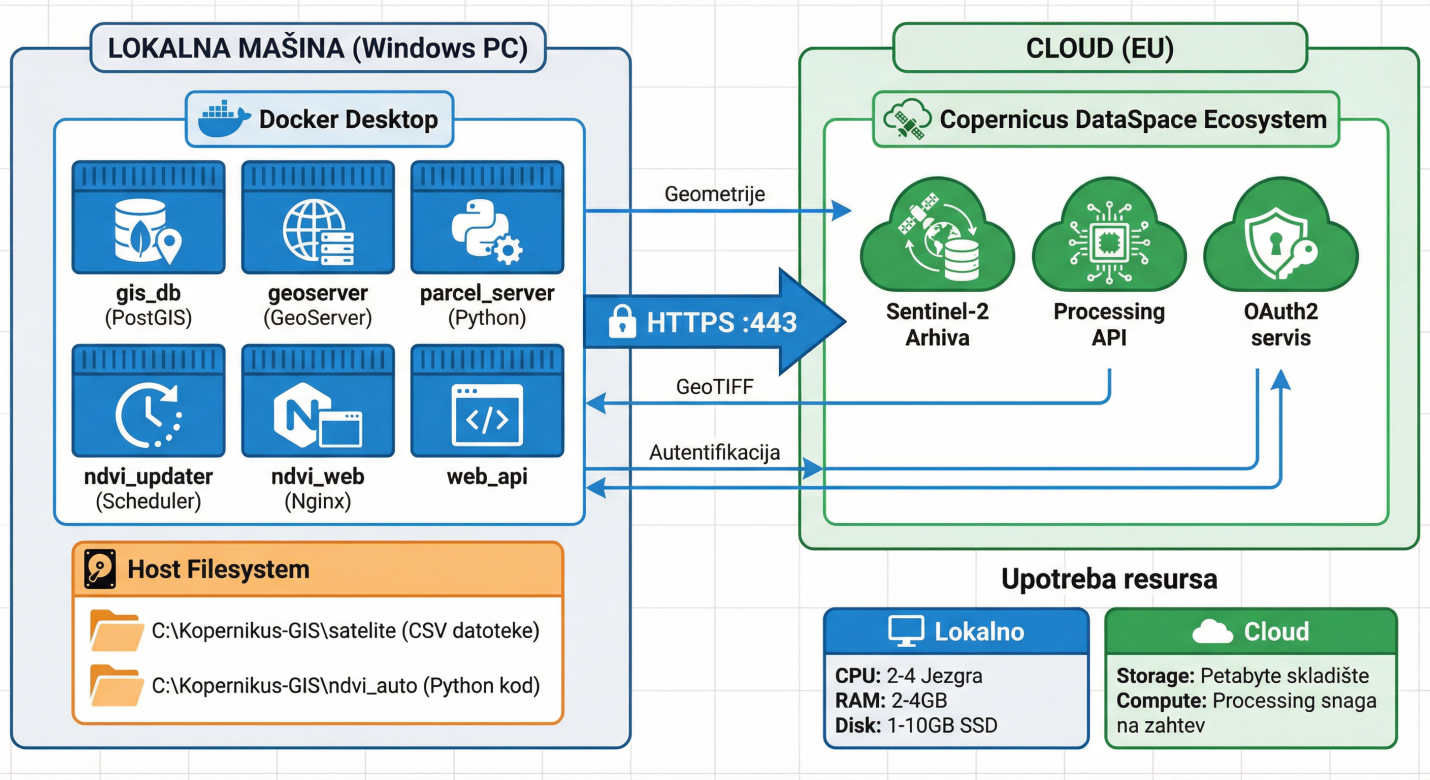
* sloj za unos i upravljanje parcelama (AOI),
* sloj za pristup satelitskim podacima,
* sloj za obradu i izračunavanje indeksa,
* sloj za čuvanje i agregaciju rezultata,
* sloj za pristup rezultatima (UI i/ili API).

**Pitanja za autora:**

* Kako bi ti u jednoj slici ili dijagramu opisao arhitekturu sistema?







* Da li je sistem više monolitni ili modularni?

MODULARNI sa mikroservisnim pristupom

* Gde se sistem izvršava (lokalno, cloud, hibridno)?

Odgovor: LOKALNO (sa hibridnim pristupom)

## 2.2 Glavne komponente sistema

**Šablon:**

Sistem se sastoji od sledećih glavnih komponenti:

* **Komponenta za upravljanje parcelama (AOI manager)** – odgovorna za unos, čuvanje i validaciju prostornih podataka.
* **EO ingest komponenta** – odgovorna za komunikaciju sa Copernicus servisima i izbor odgovarajućih snimaka.
* **Processing komponenta** – zadužena za pokretanje obrade (indeksi, statistika).
* **Storage komponenta** – čuva rezultate obrade i istorijske podatke.
* **Pristupni sloj (API/UI)** – omogućava korisnicima ili drugim sistemima da pristupe rezultatima.

**Pitanja za autora:**

* Koje od ovih komponenti trenutno postoje u implementaciji?

**Odgovor: SVIH 5 komponenti postoje, ali neke su spojene!**

### MAPIRANJE - Predložene vs Stvarne komponente:

## A) AOI MANAGER (Upravljanje parcelama) ✅

### Postojeća implementacija:

| **Pod-komponenta** | **Tehnologija** | **Lokacija** | **Odgovornost** |
| --- | --- | --- | --- |
| **Parcel Storage** | PostGIS | Docker: gis\_db | Skladištenje geometrija parcela |
| **Parcel Access** | GeoServer WFS | Docker: geoserver | WFS API za preuzimanje parcela |
| **Parcel Selection** | JavaScript | Frontend | Dropdown + text input |
| **Geometry Fetch** | Python | download\_ndvi\_parcel.py | fetch\_parcel\_bbox() funkcija |

## B) EO INGEST (Komunikacija sa Copernicus) ✅

### Postojeća implementacija:

| **Pod-komponenta** | **Tehnologija** | **Lokacija** | **Odgovornost** |
| --- | --- | --- | --- |
| **OAuth2 Client** | Python requests | download\_and\_publish.py | get\_token() - dobavljanje JWT tokena |
| **Catalog Search** | ❌ NIJE implementirano | - | Pretraga Catalog API-ja |
| **Scene Selection** | Copernicus API | Cloud (eksterno) | Automatski izbor najboljeg snimka |
| **Download Manager** | Python | download\_and\_publish.py | download\_with\_fallback() funkcija |

## C) PROCESSING (Obrada satelitskih podataka) ✅

### Postojeća implementacija:

| **Pod-komponenta** | **Tehnologija** | **Lokacija** | **Odgovornost** |
| --- | --- | --- | --- |
| **Evalscript Execution** | JavaScript (SentinelHub) | **Copernicus Cloud** | Izvršavanje evalscripta, računanje indeksa |
| **SCL Filtering** | JavaScript (u evalscriptu) | **Copernicus Cloud** | Maskiranje oblaka |
| **Raster Generation** | Copernicus Processing | **Copernicus Cloud** | GeoTIFF kreiranje |
| **Statistics Calc** | Copernicus Stats API | **Copernicus Cloud** | Min/max/mean/median/percentiles |
| **CSV Generation** | Python | **Lokalno (Docker)** | stats\_to\_rows(), write\_csv() |
| **Zone Classification** | JavaScript (evalscript) | **Copernicus Cloud** | 3-zone mapping (crvena/žuta/zelena) |

## D) STORAGE (Skladištenje rezultata) ✅

### Postojeća implementacija:

| **Tip podataka** | **Tehnologija** | **Lokacija** | **Format** | **Veličina** |
| --- | --- | --- | --- | --- |
| **Parcele (geometrije)** | PostGIS | Docker volume: postgres\_data | PostGIS GEOMETRY | ~10 MB |
| **Rasteri (indeksi)** | GeoServer Coverage Store | Docker volume: geoserver\_data | GeoTIFF | ~5-50 MB |
| **CSV podaci** | Host filesystem | C:\Kopernikus-GIS\satelite\ | CSV | ~500 KB |
| **Python kod** | Host filesystem | C:\Kopernikus-GIS\ndvi\_auto\ | .py | ~100 KB |
| **Metadata** | JSON file | latest\_metadata.json | JSON | < 1 KB |

## E) PRISTUPNI SLOJ (API/UI) ✅

### Postojeća implementacija:

| **Tip interfejsa** | **Komponenta** | **Protokol** | **Port** | **Format odgovora** |
| --- | --- | --- | --- | --- |
| **Web UI** | Nginx + Leaflet | HTTP | 8088 | HTML/JS/CSS |
| **HTTP API** | Parcel Server | HTTP | 5010 | JSON |
| **WMS** | GeoServer | HTTP | 8083 | PNG/GeoTIFF |
| **WFS** | GeoServer | HTTP | 8083 | GeoJSON/XML |
| **WCS** | GeoServer | HTTP | 8083 | GeoTIFF |

* Da li su neke od njih spojene u jednu celinu?

DA - Nekoliko komponenti su spojene:

1. EO Ingest + Processing = download\_and\_publish.py modul

Spojeno u jednu Python biblioteku

Razlog spajanja:

Copernicus Processing API OBAVLJA I INGEST I PROCESSING u jednom API pozivu

Evalscript je deo API zahteva (nije odvojen processing step)

2. Storage = PostGIS + GeoServer + Host Filesystem (3 mesta)

Razlog spajanja:

Nema centralizovano storage rešenje

Svaka komponenta čuva svoje podatke u najpogodnijem formatu

* Koje komponente su eksterni servisi, a koje su lokalne?

### EKSTERNI SERVISI (Cloud):

### 1. Copernicus DataSpace Ecosystem (KRITIČAN!)

**Servisi:**

Sentinel-2 Data Archive

   - Petabyte-scale satelitski snimci

   - Globalna pokrivenost

   - Retention: Nekoliko godina unazad

Identity Service (OAuth2)

   - identity.dataspace.copernicus.eu

   - JWT token generisanje

   - Token lifetime: ~1h

Processing API (Sentinel Hub)

   - sh.dataspace.copernicus.eu/api/v1/process

   - Evalscript execution (na cloud infrastrukturi!)

   - GeoTIFF generisanje

   - Statistike (Stats API)

Catalog API

   - catalogue.dataspace.copernicus.eu

   - Metadata pretraga

   - Scene metadata (trenutno NE koristimo)

**Dependency:**

❗ KRITIČNA ZAVISNOST - Bez Copernicus API-ja sistem NE RADI!

**SLA:**

Dostupnost: ~99% (Copernicus guarantee)

Latencija: 10-60 sekundi (processing time)

Rate limiting: Nepoznato (pretpostavka: ~10-20 req/min)

### LOKALNI SERVISI (Docker): 🐳

#### 1. PostGIS (gis\_db)

#### **Image: kartoza/postgis:15-3.3**

#### **Port: 5434**

#### **Volume: postgres\_data (persist)**

#### **Odgovornost: Skladištenje parcela (vektori)**

#### 2. GeoServer (geoserver)

#### **Image: kartoza/geoserver:latest**

#### **Port: 8083**

#### **Volume: geoserver\_data (persist)**

#### **Odgovornost: WMS/WFS/WCS servisi, raster storage**

#### 3. Parcel Server (parcel\_server)

#### **Image: kopernikus-gis-parcel\_server (custom build)**

#### **Port: 5010**

#### **Volume: ./satelite:/app/data (bind mount)**

#### **Odgovornost: On-demand generisanje rastera i CSV-a**

#### 4. NDVI Updater (ndvi\_updater)

#### **Image: kopernikus-gis-ndvi\_updater (custom build)**

#### **Volume: ./satelite:/app/data (bind mount)**

#### **Odgovornost: Periodično ažuriranje globalnih indeksa**

#### 5. Nginx (ndvi\_web)

#### **Image: nginx:alpine**

#### **Port: 8088**

#### **Volume: ./:/usr/share/nginx/html:ro (read-only)**

#### **Odgovornost: Serviranje frontend statičkih fajlova**

#### 6. Web API (web\_api)

#### **Image: ? (nije jasno iz docker-compose.yml)**

#### **Odgovornost: ? (možda nekorišćen ili legacy)**

## DETALJNO MAPIRANJE:

## 2.3 Tok podataka (end-to-end flow)

**Šablon:**

Tok podataka u sistemu može se opisati sledećim koracima:

1. Definisanje il i izbor parcele (AOI),
2. Slanje zahteva za obradu za izabrani vremenski period,
3. Pronalaženje odgovarajućih Sentinel-2 snimaka,
4. Obrada podataka i izračunavanje indeksa,
5. Agregacija rezultata po parceli,
6. Čuvanje i izlaganje rezultata korisniku ili API-ju.

Ovaj tok predstavlja tipičan scenario izvršavanja sistema.

**Pitanja za autora:**

* Da li postoji razlika između “online” i “batch” toka obrade?

DA - postoje 2 različita toka:

TOK 1: ONLINE (On-Demand) - Interaktivni

Trigger: Korisnik klikne dugme u frontend-u

TOK 2: BATCH (Scheduled) - Automatski

Trigger: Scheduler (svaki dan u ponoć)

* Da li se rezultati generišu sinhrono ili asinhrono?

SINHRONO (sa nekim asinhronim elementima)

| **Proces** | **Tip** | **Korisnik čeka?** | **Može paralelno?** |
| --- | --- | --- | --- |
| **HTTP request (Frontend → Server)** | **Async** | NE (browser responsivan) | DA |
| **Python subprocess (Server → Script)** | **Sync** | DA (HTTP response čeka) | NE |
| **Copernicus API processing** | **Sync** | DA (Python čeka response) | NE |
| **GeoServer upload** | **Sync** | DA (Python čeka upload) | NE |
| **WMS tile loading** | **Async** | NE (progresivno) | DA (paralelno) |
| **CSV fetch (Frontend)** | **Async** | NE | DA |
| **Scheduled batch** | **Async** | NE (pozadina) | DA (nezavisno) |

## 2.4 Uloga Copernicus servisa u arhitekturi

**Šablon:**

Copernicus servisi predstavljaju ključni deo arhitekture i koriste se za:

* pristup Sentinel-2 podacima,
* filtriranje snimaka po vremenu i oblačnosti,
* osnovni pre-processing (cloud mask, AOI clip),
* izračunavanjeindeksailistatistika.

Ovim pristupom se značajno smanjuje potreba za lokalnom obradom velikih količina podataka.

**Pitanja za autora:**

* Koji tačno Copernicus/Sentinel Hub servisi se koriste?

4 COPERNICUS SERVISA U UPOTREBI:

A) Identity Service (OAuth2 Autentifikacija)

URL: https://identity.dataspace.copernicus.eu/auth/realms/CDSE/protocol/openid-connect/token

Svrha: Generisanje JWT tokena za autentifikaciju

B) Processing API (Sentinel Hub)

URL: https://sh.dataspace.copernicus.eu/api/v1/process

Svrha: NAJVAŽNIJI SERVIS - Izvršava evalscript i vraća procesovane podatke

C) Statistical API (Stats) 📊

URL: https://sh.dataspace.copernicus.eu/api/v1/statistics

Svrha: Agregacija statistika (min/max/mean/median) za CSV generisanje

D) Catalog API (Metadata Search) 📚

URL: https://catalogue.dataspace.copernicus.eu/odata/v1/Products

Svrha: Pretraga dostupnih snimaka (metadata)

Status: ❌ TRENUTNO NE KORISTIMO!

Zašto ne koristimo:

Processing API automatski bira najbolji snimak - ne treba nam ručna pretraga

Dodalo bi kompleksnost bez praktične koristi za trenutni use case

* Da li se Copernicus koristi samo za podatke ili i za obradu?

KORISTI SE I ZA PODATKE I ZA OBRADU! (Heavy compute u cloud-u)

PODELA ODGOVORNOSTI:

COPERNICUS (Cloud) - 80% COMPUTE WORKLOAD:

1. DATA ACCESS (Pristup podacima):

✅ Sentinel-2 L2A arhiva (petabyte-scale)

✅ Band extraction (B04, B05, B08, B11, SCL)

✅ Mosaic kreiranje (ako ima više snimaka)

✅ AOI clipping (samo tražena geometrija)

✅ Resampling (20m → 10m za B05, B11)

✅ CRS transformacije (ako je potrebno)

2. CLOUD PROCESSING (Obrada u cloud-u):

✅ Evalscript izvršavanje (JavaScript runtime)   
- SCL filter (cloud masking) - ZA SVAKI PIKSEL!   
- Band math (NDRE = (B08-B05)/(B08+B05))   
- Zone klasifikacija (if ndre < 0.14 → red)   
- Color mapping (colorBlend funkcija)

✅ Pixel-level obrada   
- 26x30 = 780 piksela za parcelu 1427/2   
- Svaki piksel: 4 banda (B05, B08, SCL, dataMask)   
- Evalscript se izvršava 780 puta!

✅ Raster generation - UINT8 encoding (RGB 0-255)   
- FLOAT32 encoding (value -1.0 do +1.0)   
- GeoTIFF header (CRS, bbox, geotransform)

✅ Statistical aggregation (za Stats API)   
- Min/max scan (kroz sve piksele)   
- Mean calculation (sum / count)   
- Standard deviation   
- Percentile calculation (sortiranje vrednosti)   
- Valid pixel counting

LOKALNI (Docker) - 20% COMPUTE WORKLOAD:

* Koje funkcije su u potpunosti delegirane eksternim servisima?

### 100% DELEGIRANO NA COPERNICUS:

### 1. Sentinel-2 Data Storage i Access

### **✅  Satellite archive management (petabyte-scale)**

### **✅  Data retention (multi-year history)**

### **✅  Global coverage (ceo svet)**

### **✅  Band file management (13 banda po sceni)**

### **✅  Backup i redundancija**

**Mi NE skladištimo:**

* ❌ Raw Sentinel-2 snimke (ogromni fajlovi, TB scale)
* ❌ Multi-band GeoTIFF-ove (100-500 MB po sceni)
* ❌ Atmosferske korekcije (L1C → L2A)

**Mi skladištimo samo:**

* ✅ Procesovane produkte (indeks rasteri, 1-5 KB)
* ✅ CSV statistike (500-2000 bytes)
* ✅ Metadata (< 1 KB)

## 2. Scene Selection (Izbor najboljeg snimka)

## **✅  Catalog search (hiljadde snimaka)**

## **✅  Cloud coverage filtering**

## **✅  Temporal filtering (timeRange)**

## **✅  Spatial filtering (AOI intersect)**

## **✅  Ranking algorithm (mosaickingOrder: "leastCC")**

## **✅  Fallback logic (ako nema snimka sa < 80% oblaka)**

**Python kod - SAMO definiše kriterijume**

## 3. Pixel-Level Processing (Per-pixel compute)

## **✅  Band math (NDRE formula za svaki piksel)**

## **✅  SCL filtering (provera za svaki piksel)**

## **✅  Conditional logic (zone classification)**

## **✅  Color mapping (RGB konverzija)**

## **✅  NoData handling**

## **✅  Edge case validation (division by zero, NaN, Infinity)**

**Evalscript se izvršava MILIONE puta (za velike AOI)**

## 4. Cloud Masking (SCL Analysis)

## **✅  SCL band loading (Scene Classification Layer)**

## **✅  Per-pixel classification:**

## **- 0: No data**

## **- 1: Saturated or defective**

## **- 2: Dark area pixels**

## **- 3: Cloud shadows**

## **- 4: Vegetation**

## **- 5: Not vegetated**

## **- 6: Water**

## **- 7: Unclassified**

## **- 8: Cloud medium probability**

## **- 9: Cloud high probability**

## **- 10: Thin cirrus**

## **- 11: Snow**

## **✅  Automatic filtering (u evalscriptu)**

**Mi SAMO definišemo pravila**

## 5. Resampling i Geometric Processing

## **✅  Band resampling:**

## **- B05 (Red Edge): 20m → 10m**

## **- B11 (SWIR): 20m → 10m**

## **- SCL: 20m → 10m**

## **✅  Interpolation methods:**

## **- Nearest neighbor ili bilinear**

## **- Copernicus bira automatski**

## **✅  Geometric corrections:**

## **- Co-registration različitih banda**

## **- Orthorectification (već u L2A)**

## 6. Statistical Aggregation

## **✅  Min/max scan (kroz sve piksele)**

## **✅  Sum calculation (za mean)**

## **✅  Variance calculation (za stDev)**

## **✅  Sorting (za median i percentiles)**

## **✅  Count operations (sampleCount, noDataCount)**

Prednosti ovog pristupa:

✅ Ne treba nam:

Petabyte storage za Sentinel-2 arhivu

High-end CPU/GPU za processing

GDAL/Rasterio optimizacije

Mosaicking algoritmi

Atmospheric correction

✅ Dobijamo:

Scalable compute (Copernicus ima cluster-e)

Optimizovane algoritme (profesionalno razvijeno)

Brz pristup podacima (Copernicus CDN)

Maintenance-free (Copernicus održava)

❌ Ali gubimo:

Kontrolu nad exact algoritmom (ograničeni na evalscript)

Offline capability (mora internet)

Cost predictability (besplatno za sad, ali može se naplatiti)

Flexibility (ograničeni na Sentinel-2 L2A)

## 2.5 Lokalni moduli i odgovornosti

**Šablon:**

Lokalni deo sistema je odgovoran za orkestraciju procesa, upravljanje zahtevima i čuvanje rezultata. Ovaj sloj predstavlja osnovu za buduće proširenje sistema i integraciju dodatnih izvora podataka (npr. dronovi, senzori).

**Pitanja za autora:**

* Koje odgovornosti su trenutno implementirane lokalno?

ODGOVORNOST #1: DATA STORAGE (Skladištenje)

A) PostGIS Database (Vektorski podaci)

Kontejner: postgis

Image: postgis/postgis:15-3.3

Port: 5432

B) GeoServer Data Directory (Rasteri + konfiguracija)

Kontejner: geoserver

Image: kartoza/geoserver:2.23.0

Port: 8080

C) File System (CSV i metadata)

Lokacija: ndvi\_auto/ folder (host machine)

ODGOVORNOST #2: API ORCHESTRATION (Orkestracija)

Parcel Server = Centralni Orchestrator

Kontejner: parcel\_server

Fajl: ndvi\_auto/parcel\_server.py

Port: 5001

Framework: Python http.server (custom HTTP handler)

ODGOVORNOST #3: DATA PROCESSING (Obrada)

Python Scripts = Processing Workers

ODGOVORNOST #4: WEB SERVING (Web server)

Nginx = Static File Server

Kontejner: nginx

Port: 8088

Config: nginx.conf (default)

ODGOVORNOST #5: SPATIAL DATA SERVING (GeoServer)

GeoServer = WMS/WFS Server

Kontejner: geoserver

Port: 8080

ODGOVORNOST #6: DATABASE MANAGEMENT (PostGIS)

* Da li postoji centralni servis koji orkestrira ceo proces?

DA - parcel\_server.py je centralni orchestrator!

* Kako se rukuje greškama i ponovnim pokušajima obrade?

ERROR HANDLING - 4 NIVOA:

NIVO 1: Copernicus API Errors

Lokacija: Python scripts (download\_.py)

NIVO 2: GeoServer Errors

NIVO 3: Python Script Errors

NIVO 4: Orchestrator Errors

## ZAKLJUČAK - Lokalni deo sistema:

**Ključne odgovornosti:**

1. **Orchestration** (Parcel Server - centralni koordinator)
2. **Storage** (PostGIS + GeoServer + File System)
3. **Processing** (Python scripts za Copernicus komunikaciju)
4. **Serving** (Nginx za frontend, GeoServer za spatial data)
5. **Scheduling** (NDVI Updater za background jobs)

**Error handling:**

* ✅ Retry logika za Copernicus API (3x sa backoff)
* ✅ Partial failure tolerance (2/3 layera je OK)
* ✅ Logging (sve greške se belež)
* ❌ Nedostaje automatski retry scheduler
* ❌ Nedostaje monitoring (Prometheus)

**Arhitektonski pattern:** **Microservices sa centralnim orchestratorom** (Parcel Server koordinira sve ostale servise)

## 2.6 Granice sistema i pretpostavke

**Šablon:**

Postojeća arhitektura ima jasno definisane granice u pogledu:

* tipapodataka koje obrađuje,
* obimaobrade,
* zavisnosti od eksternihservisa.

Ove granicesuprihvatljive za fazu pilot implementacije, aliće biti predmetrevizije u narednimfazamarazvoja.

**Pitanja za autora:**

* Koje su ključne pretpostavke na kojima sistem trenutno počiva?

KLJUČNE PRETPOSTAVKE - 8 KATEGORIJA:

PRETPOSTAVKA #1: Sentinel-2 Dostupnost

Pretpostavka:

"Copernicus će uvek imati Sentinel-2 L2A snimke

za našu geografsku lokaciju (Srbija, Vojvodina)"

Osnova:

✅ Sentinel-2A i Sentinel-2B sateliti su operativni

✅ Revisit time: 5 dana (kombinovano)

✅ Srbija je u zoni pokrivenosti (45° N)

Rizici:

❌ Satelit može imati tehnički kvar (npr. Sentinel-2B u 2023)

❌ Oblačnost može da blokira snimke (zimski meseci: 60-80% oblaka)

❌ ESA može da promeni politiku pristupa (naplata)

❌ API rate limits mogu da se uvedu

PRETPOSTAVKA #2: Copernicus API Stabilnost

PRETPOSTAVKA #3: Parcele u PostGIS-u – *biće, to je DKP (napomena autora)*

PRETPOSTAVKA #4: GeoServer Dostupnost – *biće, na našem serveru* *(napomena autora)*

PRETPOSTAVKA #5: Mali Obim Obrade

Pretpostavka:

"Sistem će obrađivati MALE parcele (< 50 hektara)

i MALI broj concurrent zahteva (< 10 simultano)"

Trenutno:

✅ Parcela 1427/2: ~0.5 ha (26x30 piksela na 10m rezoluciji)

✅ Parcel Server: Single-threaded (1 request po jednom)

✅ Nema request queue

Rizici:

❌ Velika parcela (npr. 500 ha = 50,000 piksela) – *smanjiti broj piksela po parceli, sada je 1 px = 10m2, može 1px = 100m2 ili primeniti tiling (napomena autora)*

❌ Veliki broj parcela (npr. 1000 parcela u batch-u)

❌ Concurrent requests (10+ korisnika istovremeno)

PRETPOSTAVKA #6: Jednostavni Indeksi

Pretpostavka:

"Sistem će računati samo osnovne vegetacione indekse

(NDVI, NDMI, NDRE) sa jednostavnim evalscriptima"

Trenutno:

✅ NDVI: (B08 - B04) / (B08 + B04)

✅ NDMI: (B08 - B11) / (B08 + B11)

✅ NDRE: (B08 - B05) / (B08 + B05)

✅ SCL filtering (cloud masking)

Rizici:

❌ Kompleksniji indeksi (npr. EVI, SAVI, LAI)

❌ Multi-temporal analiza (npr. trend detection)

❌ Machine learning (npr. crop classification)

❌ Fusion sa drugim izvorima (npr. weather data)

* Koje zavisnosti bi predstavljale rizik u produkcionom okruženju?

### RIZIK #1: Copernicus API Dependency

### Severity: CRITICAL

### Probability: MEDIUM (10% godišnje downtime je moguć)

### Impact: TOTAL SYSTEM FAILURE

### Problem:

### "Ceo sistem zavisi od Copernicus API-ja.

### Ako API padne, sistem je potpuno neupotrebljiv."

### Zavisnost:

### ✅ Authentication: identity.dataspace.copernicus.eu

### ✅ Processing: sh.dataspace.copernicus.eu/api/v1/process

### ✅ Statistics: sh.dataspace.copernicus.eu/api/v1/statistics

### Current State:

### ❌ Nema fallback API provider

### ❌ Nema cached data (za read operations)

### ❌ Nema local processing alternative

### ❌ Nema queue-based retry (job persistence)

### Produkcioni scenario:

### "11:00 AM - Farmer otvara app da vidi zone za đubrenje

### 11:05 AM - Copernicus API je u maintenance (2h downtime)

### 11:05 AM - App pokazuje 'Error loading data'

### 13:05 PM - API se vraća, ali farmer je već otišao na njivu

### Rezultat: Farmer đubri bez zone map-e (neoptimalno)"

### RIZIK #2: Single Point of Failure (Docker Host)

### Severity: CRITICAL

### Probability: HIGH (server maintenance, reboot, crash)

### Impact: TOTAL DOWNTIME

### Problem:

### "Svi Docker kontejneri rade na 1 serveru.

### Ako server padne, ceo sistem pada."

### RIZIK #3: No Authentication/Authorization

### Severity: HIGH

### Probability: HIGH (čim je na internetu)

### Impact: DATA BREACH, ABUSE

### Problem:

### "Parcel Server API nema authentication.

### Bilo ko može da pozove /ndre/{parcel\_id} i generiše podatke."

### Current State:

### ❌ Nema API key

### ❌ Nema user authentication

### ❌ Nema rate limiting

### ❌ Nema audit log (ko je šta tražio)

### Produkcioni scenario:

### "10:00 AM - Hacker pronalazi API: <http://server.com:5001/ndre/1>

### 10:01 AM - Hacker koristi script da iterira kroz sve parcel ID-eve (1-10000)

### 10:05 AM - 5000 API requesta u 5 minuta

### 10:06 AM - Copernicus API rate-limituje naš account

### 10:07 AM - Sistem postaje nedostupan za legitimne korisnike

### Rezultat: Denial of Service + moguća naplata od Copernicus-a"

### VISOKI RIZICI (Production concerns):

### RIZIK #4: No Monitoring/Alerting ❌

### Severity: HIGH

### Probability: CERTAIN (nema alerting sistema)

### Impact: DELAYED INCIDENT RESPONSE

### Problem:

### "Nema real-time monitoring.

### Admin ne zna da je sistem pao dok korisnik ne prijavi."

### Current State:

### ❌ Nema Prometheus/Grafana metrics

### ❌ Nema health check dashboard

### ❌ Nema email/SMS alerting

### ❌ Nema uptime tracking

### Produkcioni scenario:

### "02:00 AM - PostGIS kontejner pada (out of memory)

### 02:01 AM - GeoServer ne može da pročita geometrije (WFS failure)

### 08:00 AM - Prvi farmer otvara app → greška

### 08:05 AM - Farmer zove support

### 08:10 AM - Admin proverava logs, nalazi problem

### 08:15 AM - Admin restartuje PostGIS

### Rezultat: 6 sati downtime-a + loš user experience"

### RIZIK #5: No Data Backup Strategy ❌

### Severity: HIGH

### Probability: MEDIUM (disk failure, accidental delete)

### Impact: DATA LOSS

### RIZIK #6: Hard-Coded Thresholds

### Severity: MEDIUM

### Impact: SUBOPTIMAL ZONE CLASSIFICATION

### RIZIK #7: No Queue System

### Severity: MEDIUM

### Impact: POOR SCALABILITY

### RIZIK #8: No Automated Testing

### Severity: MEDIUM

### Impact: REGRESSION BUGS

## REZIME - Priority Mitigation Plan:

| **Rizik** | **Priority** | **Effort** | **Cost** | **Impact** |
| --- | --- | --- | --- | --- |
| **Copernicus API dependency** | P0 | 5d | $50/mo | Prevents total failure |
| **Single point of failure** | P0 | 5d | $100/mo | Ensures uptime |
| **No authentication** | P0 | 3d | $10/mo | Prevents abuse |
| **No monitoring** | P1 | 3d | $20/mo | Fast incident response |
| **No backup** | P1 | 2d | $10/mo | Prevents data loss |
| **Hard-coded thresholds** | P2 | 3d | $0 | Better accuracy |
| **No queue system** | P2 | 5d | $20/mo | Better scalability |
| **No tests** | P2 | 5d | $0 | Code quality |

**Total cost za P0 + P1 mitigacije: ~$190/mesečno**

**Total development time: ~18 dana**

**Zaključak:** Sistem je **prihvatljiv za pilot/testing**, ali **NIJE production-ready** bez navedenih mitigacija!