PRIJAVA PROJEKTA - RSO SEMINAR 2025/2026

Osnovni podatki

Naslov projekta: Predelava aplikacije Kamen-Škarje-Papir v Cloud Native Mikro Storitve

Člani skupine: Bernard Kučina, Filip Merkan

Povezava do organizacije: https://github.com/RSO-skupina-03

Kratek opis projekta

Projekt predstavlja predelavo obstoječe monolitne aplikacije "Kamen-Škarje-Papir" ki je objavljena v GitHub organizaciji v sodobno cloud native arhitekturo z uporabo mikro storitev. Trenutna aplikacija, ki podpira dve različici igre (klasično KŠP in razširjeno KŠPOV z Ogenj/Voda), bo razdeljena na 4 neodvisnih mikro storitev, ki bodo omogočale boljšo skalabilnost, vzdrževanje in razširljivost. Rešitev bo rešila probleme monolitne arhitekture, kot so težko vzdrževanje, omejena skalabilnost in tesno povezanost komponent, ter zagotovila visoko dostopnost in odpornost na napake. Aplikacija bo podpirala večnajemniškost (multi-tenancy) z izolacijo podatkov in zahtevkov po `tenant_id`, usmerjanjem na podlagi poddomene ali HTTP glave ter uveljavljanjem varnostnih pravil na ravni posameznega najemnika.

Ogrodje in razvojno okolje

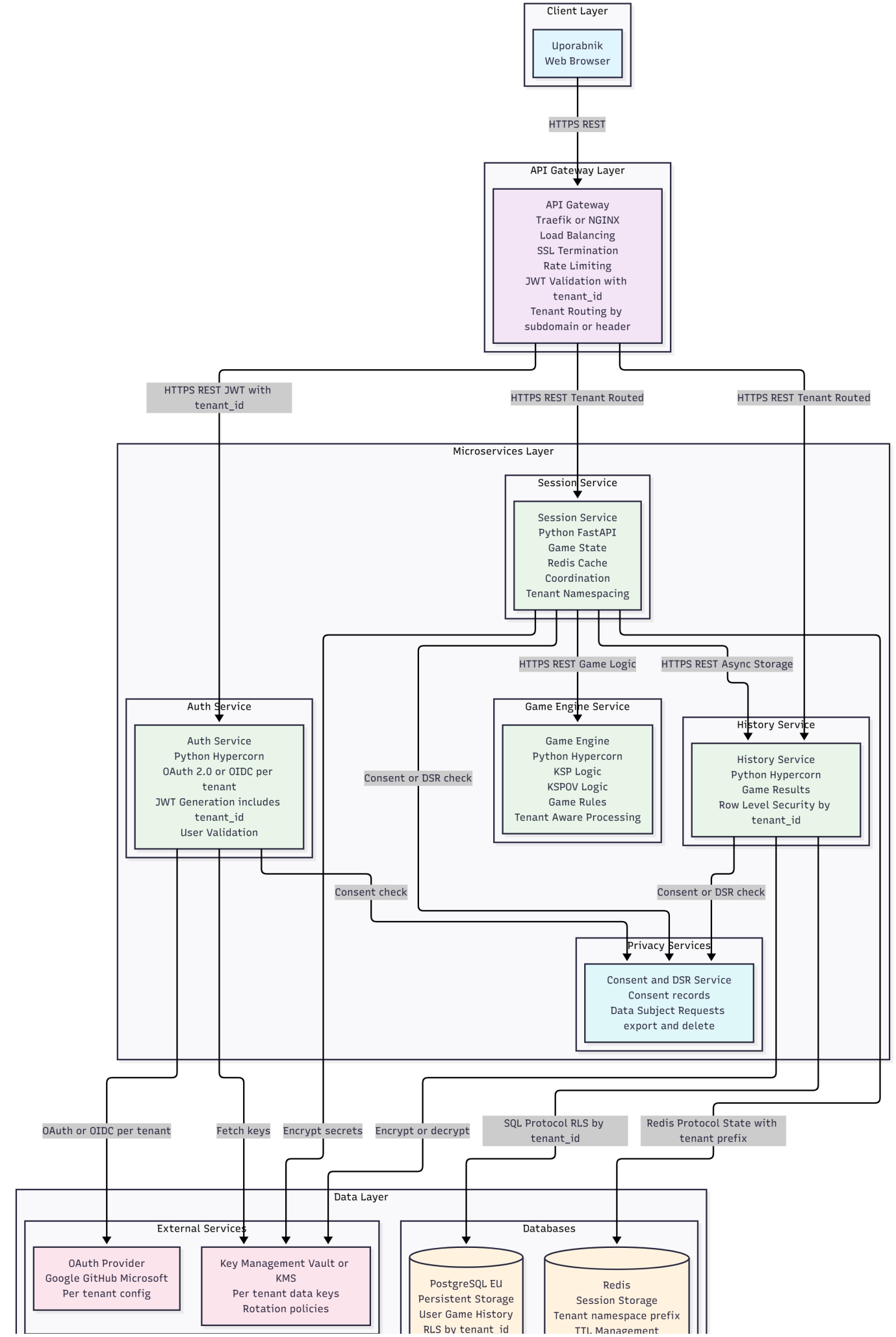
Tehnologije in ogrodja:

- **Backend:** Python 3.11+ (Hypercorn), Bottle (za migracijo)
- **Baze podatkov:** PostgreSQL (trajno shranjevanje), Redis (seje)
- **Containerizacija:** Docker, Docker Compose
- **Orkestracija:** Kubernetes
- **Komunikacija:** REST API
- **Avtentikacija:** JWT, OAuth 2.0 integracija
- **Monitoring:** Prometheus
- **CI/CD:** GitHub Actions
- **API Gateway:** Traefik/NGINX
- **Večnajemniškost:** JWT vsebuje `tenant_id`, usmerjanje zahtev po najemniku (poddomena ali glava), Redis imenski prostori per tenant, PostgreSQL RLS (Row-Level Security) po `tenant_id`

Razvojno okolje:

- **IDE:** Visual Studio Code
- **Verzije:** Git, GitHub
- **Testiranje:** Postman
- **Dokumentacija:** OpenAPI 3.0, Markdown

Shema arhitekture


Client Layer
Uporabnik
Web Browser
HTTPS REST
API Gateway Layer
API Gateway
Traefik or NGINX
Load Balancing
SSL Termination
Rate Limiting
JWT Validation with tenant_id
Tenant Routing by subdomain or header
HTTPS REST JWT with tenant_id
HTTPS REST Tenant Routed
HTTPS REST Tenant Routed
Microservices Layer
Session Service
Session Service
Python FastAPI
Game State
Redis Cache
Coordination
Tenant Namespacing
HTTPS REST Game Logic
HTTPS REST Async Storage
Auth Service
Auth Service
Python Hypercorn
OAuth 2.0 or OIDC per tenant
JWT Generation includes tenant_id
User Validation
Game Engine Service
Game Engine
Python Hypercorn
KSP Logic
KSPOV Logic
Game Rules
Tenant Aware Processing
History Service
History Service
Python Hypercorn
Game Results
Row Level Security by tenant_id
Consent or DSR check
Consent check
Consent or DSR check
Privacy Services
Consent and DSR Service
Consent records
Data Subject Requests
export and delete
OAuth or OIDC per tenant
Fetch keys
Encrypt secrets
Encrypt or decrypt
SQL Protocol RLS by tenant_id
Redis Protocol State with tenant prefix
Data Layer
External Services
OAuth Provider
Google GitHub Microsoft
Per tenant config
Key Management Vault or KMS
Per tenant data keys
Rotation policies
Databases
PostgreSQL EU
Persistent Storage
User Game History
RLS by tenant_id
Redis
Session Storage
Tenant namespace prefix
TTL Management

Komunikacijski protokoli:

- **API Gateway ↔ Storitve:** HTTP/HTTPS (REST)
- **Storitve ↔ Storitve:** HTTP (notranja komunikacija)
- **Session ↔ Game Engine:** HTTP (synchronous)
- **Session ↔ History:** HTTP (asynchronous)
- **Auth ↔ OAuth Provider:** OAuth 2.0 protokol
- **Session ↔ Redis:** Redis protocol
- **History ↔ PostgreSQL:** SQL protocol
- **JWT nosi `tenant_id`, Gateway izvaja tenant routing (poddomena ali glava), v PostgreSQL je uveljavljena RLS po `tenant_id`**

Seznam funkcionalnosti mikro storitev

1. AUTH SERVICE

Funkcionalnosti:

- Avtentikacija uporabnikov preko OAuth 2.0 (Google, GitHub, Microsoft)
- Generiranje in validacija JWT tokenov
- Upravljanje uporabniških skupin (subscribers/non-subscribers)
- Per-tenant konfiguracija OAuth ponudnikov
- JWT vključuje `tenant_id` in pravice v okviru najemnika
- Middleware za avtorizacijo
- OAuth callback handling

API Endpoints:

- `GET /auth/login` - OAuth redirect URL
- `GET /auth/callback` - OAuth callback handler
- `POST /auth/validate` - Validacija JWT
- `GET /auth/user-info` - Informacije o uporabniku
- `POST /auth/refresh` - Refresh JWT token

2. GAME ENGINE SERVICE

Funkcionalnosti:

- Implementacija igralne logike za KSP (3 možnosti)
- Implementacija igralne logike za KSPOV (5 možnosti)
- Deterministično računanje zmagovalca
- Validacija igralnih potez

API Endpoints:

- `POST /game/ksp/play` - Igra KSP poteze

- `POST /game/kspov/play` - Igra KSPOV poteze

3. SESSION SERVICE

Funkcionalnosti:

- Upravljanje aktivnih iger (Redis)
- Kreiranje novih iger
- Posodabljanje stanja iger
- Koordinacija med Game Engine in History
- Izolacija sejnika po najemniku (Redis imenski prostor z `tenant_id`)

API Endpoints:

- `POST /sessions/ksp` - Nova KSP igra
- `POST /sessions/kspov` - Nova KSPOV igra
- `POST /sessions/{id}/move` - Poteza v igri
- `GET /sessions/{id}` - Stanje igre

4. HISTORY SERVICE

Funkcionalnosti:

- Shranjevanje zaključenih iger (PostgreSQL)
- Pregled zgodovine iger
- Statistike uporabnikov
- Brisanje zgodovine
- Uveljavljanje RLS po `tenant_id` (dostop le do podatkov lastnega najemnika)

API Endpoints:

- `GET /history/ksp` - Zgodovina KSP iger
- `GET /history/kspov` - Zgodovina KSPOV iger
- `DELETE /history/ksp` - Brisanje KSP zgodovine
- `DELETE /history/kspov` - Brisanje KSPOV zgodovine
- `POST /history/ingest` - Shranjevanje rezultata

5. API GATEWAY / WEB UI

Funkcionalnosti:

- Usmerjanje zahtev na mikro storitve
- Statični frontend (HTML/CSS/JS)
- JWT middleware
- Rate limiting
- Load balancing
- Tenant-aware usmerjanje (poddomena npr. `tenant-a.example.com` ali glava `X-Tenant-ID`)

API Endpoints:

- Vsi obstoječi UI endpointi (`/`, `/ksp/`, `/kspov/`, itd.)

Primeri uporabe

Osnovni primeri uporabe:

1. **Prijava uporabnika**
    - Uporabnik dostopa do aplikacije prek poddomene svojega najemnika (npr. `acme.example.com`) ali odjemalec pošlje glavo `X-Tenant-ID`.
    - Uporabnik klikne "Prijavi se z Google/GitHub"
    - Sistem preusmeri na OAuth provider
    - OAuth provider vrne authorization code
    - Sistem izda JWT token, ki vključuje `tenant_id` in uporabniška pravila v okviru najemnika
2. **Nova igra KSP**
    - Uporabnik izbere "Nova igra KSP"
    - Session Service ustvari novo igro v imenskem prostoru najemnika
    - Sistem prikaže igralno površino
3. **Nova igra KSPOV**
    - Uporabnik izbere "Nova igra KSPOV"
    - Session Service ustvari novo igro z 5 možnostmi v imenskem prostoru najemnika
    - Sistem prikaže razširjeno igralno površino
4. **Igra poteze KSP**
    - Uporabnik izbere orožje (Kamen/Škarje/Papir)
    - Game Engine izračuna rezultat
    - Session Service posodobi stanje igre v okviru izbranega `tenant_id`
5. **Igra poteze KSPOV**
    - Uporabnik izbere orožje (Kamen/Škarje/Papir/Ogenj/Voda)
    - Game Engine izračuna rezultat z razširjenimi pravili
    - Session Service posodobi stanje igre v okviru izbranega `tenant_id`
6. **Pregled zgodovine KSP**
    - Uporabnik zahteva zgodovino KSP iger
    - History Service vrne shranjene rezultate filtrirane po `tenant_id`
    - Sistem prikaže statistike
7. **Pregled zgodovine KSPOV**
    - Uporabnik zahteva zgodovino KSPOV iger
    - History Service vrne shranjene rezultate filtrirane po `tenant_id`
    - Sistem prikaže statistike
8. **Brisanje zgodovine**

- Uporabnik zahteva brisanje zgodovine
- History Service počisti podatke iz PostgreSQL z uveljavljeno RLS po `tenant_id`
- Sistem potrdi uspešno brisanje

9. **Odjava uporabnika**

- Uporabnik se odjavi iz sistema
- Auth Service invalidira JWT token
- Sistem preusmeri na OAuth provider za odjavo
- Sistem preusmeri na prijavo

Multi-tenant primer uporabe - Prijava in igranje v okviru najemnika

Udeleženci: Uporabnik, API Gateway, Auth Service, Session Service, Game Engine, History Service

Tok:

1. Uporabnik odpre `acme.example.com` (ali pošlje `X-Tenant-ID: acme`).
2. API Gateway zabeleži `tenant_id` in doda ga v posredovane zahteve.
3. Uporabnik se prijavi prek OAuth; Auth Service izda JWT, ki vsebuje `tenant_id`.
4. Uporabnik ustvari novo KSPOV igro; Session Service ustvari sejo v Redis pod imenom prostora `tenant:acme:*`.
5. Uporabnik odigra poteze; Session Service kliče Game Engine, posodablja stanje v okviru `tenant_id`.
6. Po zaključku igre Session Service pošlje rezultat v History Service; v PostgreSQL je uveljavljena RLS po `tenant_id`.
7. Uporabnik pregleda zgodovino; vrnejo se samo rezultati z `tenant_id = acme`.

Kompleksnejši primer uporabe - Zaključek igre s shranjevanjem:

Udeleženci: Uporabnik, Session Service, Game Engine, History Service, Auth Service

Tok:

1. Uporabnik igra zadnjo potezo v KSPOV igri (15. poteza)
2. Session Service pošlje potezo v Game Engine
3. Game Engine vrne rezultat z oznako `isFinished: true`
4. Session Service shrani končno stanje v Redis
5. Session Service asinhrono pošlje rezultat v History Service
6. History Service shrani rezultat v PostgreSQL
7. Session Service počisti igro iz Redis
8. Uporabnik vidi končni rezultat in statistike

Komunikacije:

- Session → Game Engine (HTTP): `POST /game/kspov/play`
- Session → History (HTTP): `POST /history/ingest`
- UI → Session (HTTP): `POST /sessions/{id}/move`