

Konfigūratoriaus API

Aprašymas

Konceptualios architektūros elementų atitikmuo

Swagger dokumentacija

Implementacija

Vidinės bibliotekos

NestJS moduliai

Komunikacija su mikroservisais

RabbitMQ

Autentifikacija

OIDC / VIISP

Environment variables

Operacijų žurnalas (Logging)

OpenTelemetry

Request Context

Docker konteineris

Bazinė informacija

Health check

Aprašymas

REST API e-paslaugų konfigūravimui ir administravimui. API veikia kaip **gateway** - koordinuoja veiksmus tarp daugelio mikroservisų per RabbitMQ.



Konceptualios architektūros elementų atitikmuo

- Paslaugų konstruktoriaus API
- Paslaugų administratoriaus API
- Veikia kaip gateway perduodant loginį įvykdymą į mikroservisų sluoksnį šiem API:
 - Naudotojo paskyros API
 - Naudotojo užsakymų API

- Naudotojo užsakymų rezultatų API
- Paslaugų teikimo API
- Paslaugų administratoriaus API
- Paslaugų užsakymų eilės API
- Institucijų paskyrų API

Swagger dokumentacija

Pilna API endpoint'ų dokumentacija prieinama per Swagger UI:

URL: <http://configurator-api.demo.spp.codigi.lt>

Swagger URL: <http://configurator-api.demo.spp.codigi.lt/api>

OpenAPI JSON: <http://configurator-api.demo.spp.codigi.lt/api-json>

Implementacija

Visas kodas talpinamas `/apps/configurator-api` aplanke.

Įgyvendinta naudojantis **NestJS** framework (Node.js).

Vidinės bibliotekos

Biblioteka	Paskirtis
<code>@spp/types</code>	Bendri TypeScript tipai ir DTO
<code>@spp/ms-clients</code>	RabbitMQ klientų moduliai komunikacijai su MS
<code>@spp/tsai-utils</code>	TypeScript AI utilities
<code>@spp/utils</code>	Bendri utilities

NestJS moduliai

Modulis	Aprašymas
<code>EServicesModule</code>	E-paslaugų ir versijų valdymas
<code>EServiceWorkflowsModule</code>	E-paslaugų workflow operacijos

EServiceRequestsModule	E-paslaugų užsakymų valdymas
EServiceRequestsProfilesModule	Užsakymų profilių valdymas
AddOnConfigurationsModule	Įskiepių konfigūracijų valdymas
PublishConfigurationsModule	Publikavimo konfigūracijų valdymas
StandardizedListsModule	Standartizuotų sąrašų valdymas
InstitutionsModule	Institucijų valdymas
InstitutionEmployeesModule	Institucijų darbuotojų valdymas
InstitutionsEServicesModule	Institucijų e-paslaugų priskyrimas
SppAdministratorsModule	SPP administratorių valdymas
LLModule	AI/LLM funkcionalumas
NotificationsModule	Pranešimų siuntimas
OIDCModule	OpenID Connect autentifikacija
AuthModule	Autentifikacijos guards ir VIISP integracija
HealthModule	Health check endpoints
AddOnMockModule	Mock įskiepių testavimui
StandardizedListsMockModule	Mock standartizuotų sąrašų testavimui
GraviteeMockModule	Gravitee API gateway mock

Komunikacija su mikroservisais

RabbitMQ

API komunikuoja su mikroservisais per **RabbitMQ** naudojant `@spp/ms-clients` biblioteką.

Queue naming pattern: `{ENV_PREF}_{queue_name}`

Kur `ENV_PREF` yra aplinkos prefiksas (pvz., `loc`, `dev`, `sta`, `prod`).

MS Client modulis	RabbitMQ Queue	Mikroservisas
<code>EServicesClientModule</code>	<code>{ENV_PREF}_e_services_queue</code>	<code>e-services</code>
<code>EServiceWorkflowsClientModule</code>	<code>{ENV_PREF}_e_service_workflows_queue</code>	<code>e-service-workflows</code>
<code>AddOnConfigurationsClientModule</code>	<code>{ENV_PREF}_add_on_configurations_queue</code>	<code>add-on-configurations</code>
<code>EServiceRequestsClientModule</code>	<code>{ENV_PREF}_e_service_requests_queue</code>	<code>e-service-requests</code>
<code>EServiceRequestsProfilesClientModule</code>	<code>{ENV_PREF}_e_service_request_profiles_queue</code>	<code>e-service-requests-profiles</code>
<code>EServiceRequestsPublishClientModule</code>	<code>{ENV_PREF}_e_service_requests_publish_queue</code>	<code>e-service-requests-publish</code>
<code>StandardizedListsClientModule</code>	<code>{ENV_PREF}_standardized-lists</code>	<code>standardized-lists</code>
<code>InstitutionsClientModule</code>	<code>{ENV_PREF}_institutions_queue</code>	<code>institutions</code>
<code>NotificationsClientModule</code>	<code>{ENV_PREF}_notifications_queue</code>	<code>notifications</code>
<code>SppAdministratorsClientModule</code>	<code>{ENV_PREF}_spp_settings_queue</code>	<code>spp-settings</code>

LLMClientModule	{ENV_PREF}_llm_queue	llm
-----------------	----------------------	-----

Autentifikacija

OIDC / VIISP

API implementuoja **OpenID Connect (OIDC)** protokolą su **VIISP** (Valstybinė informacinių išteklių sąveikumo platforma) autentifikacija.

OIDC endpoints:

- GET /oidc/.well-known/openid-configuration - OIDC konfigūracija
- GET /oidc/jwks - JSON Web Key Set
- GET /oidc/auth - Authorization endpoint
- POST /oidc/token - Token endpoint
- GET /oidc/userinfo - User info endpoint
- POST /oidc/viisp-data - VIISP callback

Naudojami guards:

Guard	Paskirtis
AuthGuard	JWT Bearer token validacija, naudotojo duomenų išgavimas
SppAdminGuard	SPP administratoriaus teisių tikrinimas

Token formatas: JWT Bearer token su claims: given_name , family_name , preferred_username

Environment variables

Kintamasis	Aprašymas	Pavyzdys
ENV_PREF	Aplinkos prefiksas	local , dev , sta , prod
NODE_ENV	Node.js aplinka	development , production

PORT	API portas	3000
RMQ_CONNECTION_STRING	RabbitMQ prisijungimo eilutė	amqp://user:password@spp-mq-rabbitmq:5672
REDIS_HOST	Redis serverio adresas	spp-cache-redis
REDIS_PORT	Redis portas	6379
DB_HOST	PostgreSQL serverio adresas	spp-db-postgres
DB_PORT	PostgreSQL portas	5432
DB_USERNAME	DB vartotojo vardas	root
DB_PASSWORD	DB slaptažodis	password
DB_DB	Duomenų bazės pavadinimas	spp
VIISP_SERVICE_URL	VIISP autentifikacijos servisas	https://test.epaslaugos.lt/services/services/auth
VIISP_WSDL_URL	VIISP WSDL	https://test.epaslaugos.lt/.../auth?wsdl
VIISP_PID	VIISP Provider ID	VSID000000000113
VIISP_CONFIGURATOR_API_POSTBACK_URL	VIISP callback URL	http://localhost:3000/oidc/viisp-data
VIISP_PORTAL_URL	VIISP portalo URL	https://test.epaslaugos.lt/portal/.../v2
OIDC_ISSUER_URL	OIDC issuer URL	http://configurator-api:3000
OIDC_PRIVATE_KEY	OIDC JWT private key (RS256)	(base64 encoded PEM)

<code>OIDC_PUBLIC_KEY</code>	OIDC JWT public key	(base64 encoded PEM)
<code>OIDC_KEY_ID</code>	OIDC Key ID	(UUID)
<code>GRAYLOG_HOST</code>	Graylog serverio adresas	<code>graylog</code>
<code>GRAYLOG_PORT</code>	Graylog portas	<code>12201</code>
<code>GRAYLOG_PROTOCOL</code>	Graylog protokolas	<code>udp</code>
<code>OTEL_EXPORTER_OTLP_ENDPOINT</code>	OpenTelemetry collector endpoint	<code>otel-collector:4317</code>
<code>OTEL_EXPORTER_OTLP_PROTOCOL</code>	OpenTelemetry protokolas	<code>grpc</code>
<code>OTEL_SERVICE_NAME</code>	Serviso pavadinimas telemetrijai	<code>configurator-api</code>
<code>OTEL_SERVICE_VERSION</code>	Serviso versija telemetrijai	<code>1.0.0</code>
<code>OTEL_TRACES_EXPORTER</code>	Traces eksporteris	<code>otlp</code>
<code>OTEL_LOGS_EXPORTER</code>	Logs eksporteris	<code>otlp</code>

Operacijų žurnalas (Logging)

OpenTelemetry

Naudojamas **OpenTelemetry** centralizuotam logų ir traces rinkimui.

Implementacija:

- `HttpRequestLoggingInterceptor` - HTTP užklausų/atsakymų logavimas
- Logs siunčiami į OpenTelemetry Collector per OTLP HTTP protokolą (`/v1/logs`)
- Collector persiunčia logs į **Graylog**

Loguojami duomenys:

- HTTP method, URL, status code
- Request/response duration (ms)

- Request ID (per `nestjs-cls`)
- Error details (message, name, stack)
- User agent, client IP

Request Context

`nestjs-cls` naudojamas request context tracking:

- Automatinis `X-Request-Id` propagavimas
- User context (`userPersonalCode`, `userFirstName`, `userLastName`, `userIsSuperAdmin`)

Docker konteineris

Bazinė informacija

Parametras	Reikšmė
Darbo direktorija	<code>/workspace/apps/configurator-api</code>
Portas	<code>3000</code>
Paleidimo komanda	<code>pnpm run start:dev</code>

Health check

```
1 | curl -fsSL http://localhost:3000/healthz
```