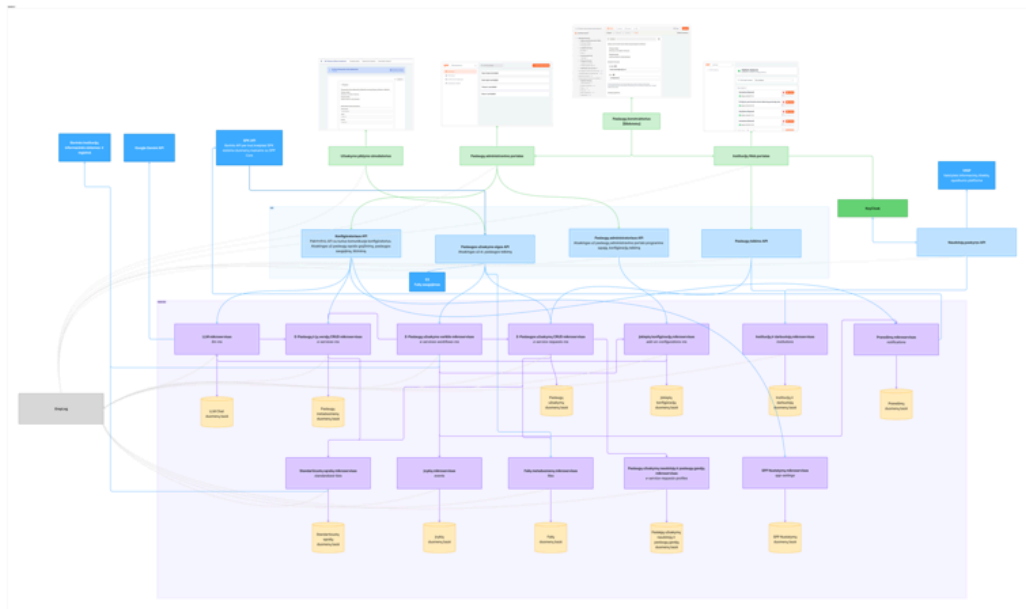


Sistemos apžvalga

Sistemos struktūra ir technologiniai aspektai

Sistema realizuojama naudojant N-Tier sluoksнинę architektūrą, aiškiai atskiriant technologinius sluoksnius – naudotojo sąsają, API užklausų apdorojimą, verslo logiką ir duomenų saugojimą.



Sistemos technologinis skaidymas:

- Web aplikacijos (naudotojo sąsajos)
- API
- Mikroservisai
- Duomenų bazės
- Loginimas (klaidų ir operacijų žurnalas)

Konceptualios architektūros modulių atvaizdavimas į realizacijos komponentus:

- **Paslaugų teikimo rezultatų duomenų bazė** realizuojama per Įvykių mikroservisą (events) ir jo dedikuotą duomenų bazę, kurioje registruojami įvykiai, susiję su e-paslaugomis, jų užsakymais ir paslaugų teikimo rezultatais.
- **Paslaugų teikimo rezultatų variklis** realizuojamas per E-paslaugų užsakymo variklio mikroservisą (e-services-workflows-ms), atsakingą už užsakymų eigos valdymą ir rezultatų registravimą.

- Paslaugos užsakymo pildymo naudotojo sąsajos simulatorius realizuojamas kaip Web aplikacija „Užsakymo pildymo simulatorius“.
- **Institucijos informacinės sistemos**, atsakingos už el. paslaugos teikimą, integracinės sąsajos simulatorius buvo pristatytas I inkremente, tačiau po suderinimo su VSSA jo funkcionalumas perimamas viešai prieinamų paslaugų ir šis atskiras komponentas III inkremente laikomas out-of-scope.

Sistema yra būsenos nesauganti (**stateless**), neprisirišusi prie konkretaus serverio ar mikroserviso. Tai leidžia konteinerius diegti keliuose serveriuose, o bet kuriai naudotojo užklausiai apdoroti pakanka užklaustos konteksto, kuriame yra visa reikalinga informacija. Visi sistemos komponentai konteinerizuojami naudojant **Docker**.

Naudojamos atvirojo kodo technologijos:

- **Docker** konteinerizacijai
- **PostgreSQL** duomenų bazėms
- **Node.js, NestJS, TypeScript** API ir mikroservisams
- **React ir TypeScript** naudotojo sąsajų kūrimui
- **RabbitMQ** asinchroniniam API ir mikroservisų komunikavimui
- **KeyCloak** naudotojų prisijungimų prie vidinių web aplikacijų valdymas
- **Redis** duomenų cache sluoksniui
- **AWS S3** dokumentų failų saugojimui
- **OpenTelemetry** operacijų žurnalui visuose API (Veikia su GrayLog)
- **Sentry** klaidų sekimui

Komponentų aprašymas

Web aplikacijos (naudotojo sąsajos)

- ~~Paslaugų konstruktorius~~ - perkeliamas į biblioteką ir pernaudojamas paslaugų administravimo ir institucijų web portale.
- Užsakymo pildymo simulatorius – atitinka konceptualų Paslaugos užsakymo pildymo naudotojo sąsajos simulatorių ir leidžia imituoti paslaugos užsakymo pildymo procesą.
- Paslaugų administravimo portalas
- Institucijų web portalas

Web aplikacijos komunikuoja su API naudodamos **REST API** protokolą.

API komponentai

- **Konfigūatoriaus API**

- **Paslaugos užsakymo eigos API**
- **Paslaugų administratoriaus API** (Implementuojamas su **Konfigūradoriaus API** kadangi yra gateway principo)
- **Paslaugų teikimo API** (Implementuojamas su **Konfigūradoriaus API** kadangi yra gateway principo)

API atsakingi už naudotojo užklausų apdorojimą, autentifikaciją, parametrų patikrinimą, užklausos kūno validavimą bei atitinkamo mikroserviso iškvietimą.

Mikroservisai

Mikroservisai atlieka pagrindines verslo logikos operacijas (CRUD, užsakymų valdymas). Jie struktūrizuojami pagal verslo logikos sritis, vengiant cikliškų priklausomybių.

- [**E-paslaugų ir versijų CRUD mikroservisas \(e-services-ms\)**](#) – tvarko e-paslaugų metaduomenis, naudoja dedikuotą duomenų bazę.
- [**E-paslaugų užsakymo variklio mikroservisas \(e-services-workflows-ms\)**](#) – valdo užsakymų eigą, duomenų validaciją, išskviečia įskiepių ir publikavimo mikroservisus ir realizuoja konceptualaus Paslaugų teikimo rezultatų variklio funkcijas..
- [**E-paslaugos užsakymų CRUD mikroservisas \(e-service-requests-ms\)**](#) – valdo e-paslaugų užsakymus, turi atskirą duomenų bazę.
- [**Įskiepių konfigūracijų mikroservisas \(add-on-configurations-ms\)**](#) – įskiepių konfigūracijų valdymas, naudoja atskirą duomenų bazę.
- [**E-paslaugos publikavimo konfigūracijos CRUD mikroservisas \(publish-configuration-ms\)**](#) – valdo publikavimo konfigūracijas.
- [**E-paslaugos publikavimo mikroservisas \(e-service-requests-publish-ms\)**](#) – atlieka publikavimo veiksmus, seka publikacijas, tvarko nesėkmingų publikacijų kartojimą.
- [**LLM mikroservisas \(llm-ms\)**](#) – atlieka veiksmus naudojant dirbtinio intelekto debesijos paslaugas, turi dedikuotą LLM Chat duomenų bazę. Detaliau aprašyta skyriuje „Integracija su debesijos DI paslaugomis“.
- [**Standartizuotų sąrašų mikroservisas \(standardized-lists\)**](#) - valdo standartizuotus sąrašus, naudoja dedikuotą duomenų bazę.
- [**Įvykių mikroservisas \(events\)**](#) - valdo įvykius, naudoja dedikuotą duomenų bazę ir realizuoja konceptualią Paslaugų teikimo rezultatų duomenų bazę, kaupdama įvykius, susijusius su e-paslaugomis, jų užsakymais ir paslaugų teikimo rezultatais.
- [**Failų metaduomenų mikroservisas \(files\)**](#) - valdo failų metaduomenis, naudoja dedikuotą duomenų bazę.

- [Paslaugų užsakymų naudotojų ir paslaugų gavėjų mikroservisas \(e-service-requests-profiles\)](#) - valdo paslaugų užsakymų naudotojus ir paslaugų gavėjus, naudoja dedikuotą duomenų bazę.
- [Institucijų ir darbuotojų mikroservisas \(institutions\)](#) - valdo institucijas ir institucijų darbuotojus, naudoja dedikuotą duomenų bazę.
- [Pranešimų mikroservisas \(notifications\)](#) - valdo pranešimų siuntimą vartotojams.
- [SPP Nustatymų mikroservisas \(spp-settings\)](#) - Globalių nustatymų saugojimui. Pvz.: SPP administratorių duomenys.

Duomenų bazės

- LLM Chat duomenų bazė
- Paslaugų metaduomenų duomenų bazė
- Paslaugų užsakymų duomenų bazė
- Įskiepių konfigūracijų duomenų bazė
- Publikavimo konfigūracijų duomenų bazė
- Standartizuotų sąrašų duomenų bazė
- Įvykių duomenų bazė – realizuoja ir konceptualią Paslaugų teikimo rezultatų duomenų bazės funkciją, kaupdama visus su paslaugų teikimu susijusius ir kitus sistemos įvykius
- Failų duomenų bazė
- Paslaugų užsakymų naudotojų ir paslaugų gavėjų duomenų bazė
- Institucijų ir darbuotojų duomenų bazė
- Pranešimų duomenų bazė
- SPP Nustatymų duomenų bazė

Integracija su debesijos DI paslaugomis

SPP sistema integruojasi su Google Gemini DI paslauga per dedikuotą LLM mikroservisą (llm-ms). Mikroservisas naudoja dviejų modelių architektūrą: Gemini Flash (konteksto planavimui ir lengvoms užklausoms) ir Gemini Pro (pagrindiniam atsakymų generavimui).

DI funkcionalumas

AI Chat – interaktyvus DI asistentas, padedantis konstruoti ir konfigūruoti e-paslaugas. Naudoja konteksto planavimą ir įrankių (function calling) mechanizmą tiesioginiams e-paslaugos modelio pakeitimams.

Komunikacija su Gemini vyksta per HTTPS, naudojant oficialų SDK (`@google/generative-ai`). API raktas saugomas aplinkos kintamajame (`GEMINI_API_KEY`).

Integracijos su išorinėmis institucijų IS ir registrais

Integracijos su išoriniais registrais ir institucijų informacinėmis sistemomis realizuojamos per **įskiepių (add-on) mechanizmą**. Įskiepiei leidžia e-paslaugos darbo eigos (workflow) žingsniuose iškviesti išorinę logiką – tiek registrų duomenų užklausas, tiek institucinių IS operacijas.

Įskiepių veikimo principas

Įskiepių konfigūracija apibrėžia: išorinio endpoint'o URL, HTTP metodą, įvesties/išvesties parametrų atvaizdavimą į formos laukus, transformacijos skriptus (pre/post) ir kvietimo sąlygas. E-paslaugų užsakymo variklio mikroservisas (e-services-workflows-ms) vykdo įskiepius pagal žingsnio konfigūraciją – automatiškai paleidžia HTTP užklausas arba JavaScript skriptus ir gautus rezultatus priskiria atgal į formos laukus.

Loginimas (audito ir klaidų žurnalas)

- [Graylog](#) naudojamas klaidų ir operacijų žurnalui

Integracijos su SPK

SPP sistema teikia atskirą API komponentą – **Paslaugos užsakymo eigos API** (e-service-workflows-api) – skirtą integracijai su SPK.

SPK integracijos API

Sritis	API endpoint'ai	Aprašymas
Užsakymų kūrimas	POST /v1.0/e-services/:eServiceId/e-service-requests	SPK inicijuoja e-paslaugos užsakymą, perduodama naudotojo ir paslaugos gavėjo duomenis
	POST /v1.0/e-services/:eServiceId/versions/:versionId/e-service-requests	Užsakymo kūrimas nurodant konkrečią versiją
Laukų reikšmių saugojimas	PUT /v1.0/e-service-requests/:id/field-values	SPK perduoda naudotojo suvestus formos laukų duomenis

Žingsnių pateikimas	POST /v1.0/e-service-requests/:id/steps/:stepId	SPK pateikia užpildyto žingsnio formos duomenis, sistema validuoja ir pereina į kitą žingsnį
Užsakymų sąrašas ir būsenos	GET /v1.0/e-service-requests	SPK gauna užsakymų sąrašą su būsenomis, filtravimo galimybėmis
Užsakymo detalės	GET /v1.0/e-service-requests/:id	SPK gauna konkretaus užsakymo būseną, žingsnių eigą, laukų reikšmes
E-paslaugų sąrašas	GET /v1.0/e-services , GET /v1.0/institutions/:id/e-services	SPK gauna publikuotų e-paslaugų sąrašą pagal instituciją
Failų įkėlimas	Per failų valdymo endpoint'us	SPK įkelia užsakymo priedų failus
Pranešimai	Per pranešimų endpoint'us	Pranešimai apie užsakymų būsenos pasikeitimus
Standartizuoti sąrašai	GET /v1.0/standardized-lists	SPK gauna standartizuotų sąrašų duomenis formos laukams užpildyti

API aprašai ir naudojamos duomenų struktūros

Žemiau pateikiamos nuorodos, kuriomis galima pasiekti SPP ir SPK naudojamų API aprašus bei OpenAPI dokumentaciją; detalūs paslaugų modeliai ir duomenų struktūros (įskaitant EServiceVersionDto kaip e-paslaugos DSL modelį) aprašyti šiose OpenAPI specifikacijose.

SPP

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


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

OpenAPI JSON (Failas): [📄](#) configurator-openapi IV.json

SPK

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

OpenAPI JSON (Web): <http://workflows-api.demo.spp.codigi.lt/api-json>

OpenAPI JSON (Failas):  workflows-openapi IV.json.json

Diegimo aplinka ir programinės įrangos pasiskirstymas

Šiuo metu SPP sprendimas veikia tiekėjo demonstracinėje aplinkoje, sudiegtoje AWS infrastruktūroje. Visi sistemos komponentai konteinerizuojami naudojant Docker ir yra būsenos nesaugantys (stateless), todėl gali būti horizontaliai skalėjami per kelis serverius.

IV inkremento metu planuojama SPP sprendimą sudiegti Užsakovo aplinkose; tikslus programinės įrangos pasiskirstymas per serverius ir tinklus bus detalizuotas, atsižvelgiant į Užsakovo infrastruktūros ypatumus.

Nepriklausomai nuo diegimo aplinkos, technologinis sprendimo pagrindas nesikeičia ir naudoja šioje dokumentacijoje išvardintas atvirojo kodo technologijas (Docker, PostgreSQL, Node.js/NestJS, React, RabbitMQ, Keycloak ir kt.).