

Eksperimentinės plėtros veiklos ataskaita

Projekto pavadinimas: Nurodyti

Projekto numeris: J05-LVPA-K-01-XXXX

Projekto vykdytojas: *UAB „Omnisend“*

Projekto partneriai: UAB „Tokenmill“

Veiklos numeris: Nurodyti

Veiklos pavadinimas: *Nurodyti*

Ataskaitos data: 2018-12-17

Lapų skaičius: Nurodyti

**Turinys**

[1. Veiklos tikslas 3](#_Toc350342211)

[2. Veiklos užduotys 3](#_Toc350342212)

[3. Įvesties duomenys 3](#_Toc350342213)

[4. Panaudotos technologijos, įrankiai, metodai 3](#_Toc350342214)

[5. Veiklos vykdymo eiga 3](#_Toc350342215)

[6. Užduočių atlikimas 3](#_Toc350342216)

[7. Įvykdyti pirkimai 3](#_Toc350342217)

[8. Pasiekti rezultatai 4](#_Toc350342218)

[9. Nepasiekti rezultatai 4](#_Toc350342219)

[10. Veiklos pakeitimai 4](#_Toc350342220)

[11. Rezultatų naujumas ir nauda projektui 4](#_Toc350342221)

# Veiklos tikslas

Sukurti metodiką ir prototipą prekių elektroninės prekybos sistemose automatiniam aprašymų generavimui, panaudojant natūralios kalbos generavimo (NLG) ir dirbtinio intelekto (AI) metodus.

# Veiklos užduotys

Šiame etape buvo apibrėžtos tokios veiklos užduotys:

* 1. *Automatizuotas dokumento plano redaktorius* – vartotojo sąsaja skirta:
     1. dokumento planų redagavimui,
     2. generuojamo teksto variantų peržiūrai bei patvirtinimui.
  2. *Natūralios kalbos generavimo (NLG) generatoriaus administravimo įrankis* – priemonė skirta generavimo sistemos valdymui, duomenų peržiūrai ir tvarkymui.
  3. *Natūralios kalbos (NLP) modelių kūrimo ir valdymo priemonės* – kalbos generavime naudojamų kalbos modelių veikimo aplinka.
  4. *RAGS architektūros detalizavimas pagal šio projekto reikalavimus* – daugialypė architektūra skirta visam NLG procesui formalizuoti.

# Automatizuotas dokumento plano redaktorius

## Veiklos aprašas

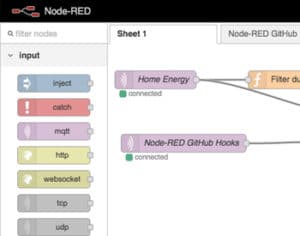
Daugialypio dokumento ruošimo (angl: *augmented writer*) vartotojo aplinka, leidžianti aprašyti komunikacijos tikslus ir bendrą dokumento planą, yra vienas iš esminių kuriamo produkto komponentų. Šis komponentas turi teikti galimybę kurti dokumentų prototipus, matyti generavimo rezultatus bei daryti reikiamas korekcijas generavimo parametruose.

Šalia teksto generavimo klausimų, šiame komponente didelę dalį užima optimalios vartotojo sąsajos kūrimas. Šio instrumento pagalba vartotojas iš esmės atlieka programavimo uždavinius, todėl sąsaja turi būti tokia, kad vartotojas, nebūtinai susipažinęs su pagrindiniais programavimo principais (tačiau galintis naudotis sudėtingesnėmis MS Excel funkcijomis) galėtų ja naudotis.

Paskiausiai, ši aplinka veikia kaip vartotojo duomenų rinkimo priemonė, tam, kad vėliau, naudojant DI (angl: *AI*) technologijas būtų galima dalinai automatizuoti optimalių planų parinkimą.

## Veiklos vykdymo eiga

Daugialypio dokumento kūrimo įgyvendinimas yra atliktas panaudojant vizualinio programavimo paradigma (angl: *visual programming*). Vizualinis programavimas yra naudojamas įvairiose srityse nuo vaikų programavimo mokymo, iki daiktų interneto sprendimų kūrimo. Nors vizualinis programavimas išlieka programavimu ir reikalauja bazinių programavimo principų suvokimo, šie principai gali būti greitai išmokti, matant programos išdėstymą atliktą, grafinių blokų ir jų sąsajų, pagalba.

  
Pav 1: Daiktų interneto vizualinio programavimo įrankio pavyzdys

Kalbos generavimo produkto kontekste, panašus įrankis turi leisti apibrėžti esminius dokumento plano komponentus:

1. Dokumento segmentus, jų komunikacijos tikslus
2. Pranešimus, jų tvarką (jei naudotojas pageidauja)
3. Naudotinas tikslias kalbines išraiškas
4. Sąlygas, nusakančias, kokie pranešimai prie kokių sąlygų turi būti realizuoti (pvz. pranešti aukštą kainą tik po prekės savybių aprašo).

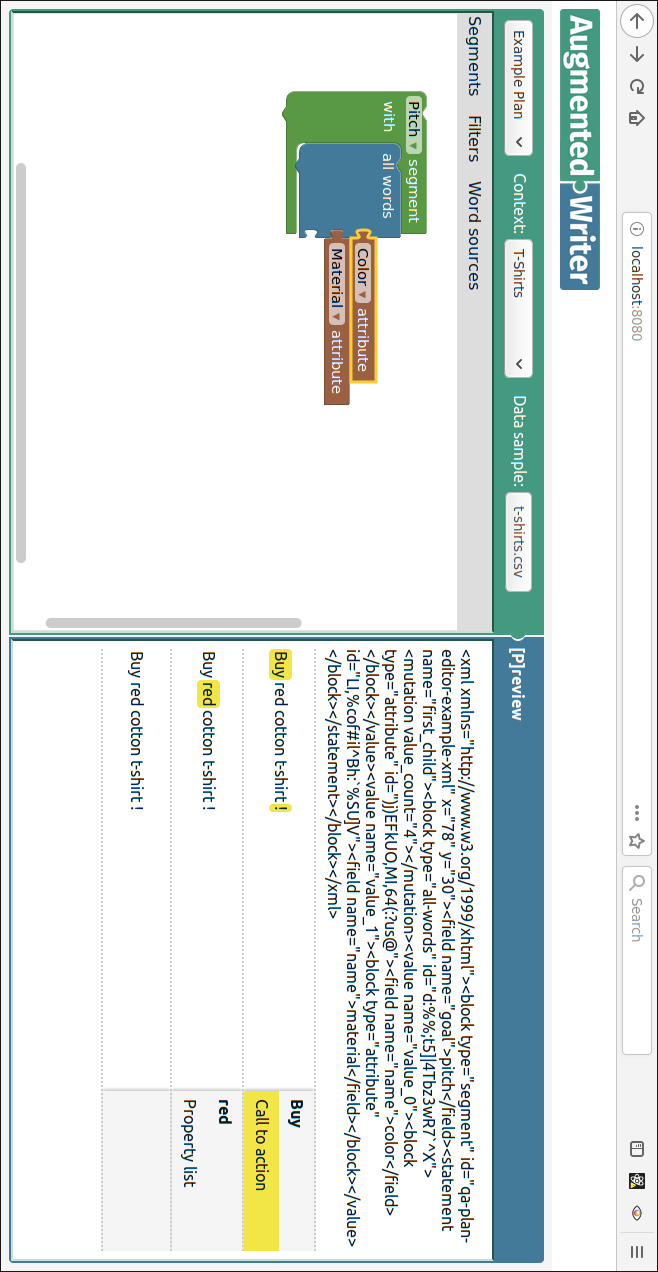
Renkantis galimą vizualinio programavimo sprendimą NLG sprendimui mes vadovavomės (ir toliau naudosim vystant mūsų įrankį), šia studija:

*Usability Analysis of Visual Programming Environments: a ‘cognitive dimensions’ framework (1996)*

## Rezultatai

Sukurtas dokumento plano redaktorius naudoja *Google Blockly* (<https://developers.google.com/blockly/>) atviro kodo vizualinės programavimo kalbos produktą, kaip bazinę biblioteką. Šios bibliotekos pagrindu buvo pradėtas kurti kalbos generavimui skirto programavimo įrankio kūrimas.

Naudojantis šiuo instrumentu vartotojas gali įkrauti pavyzdinius prekių duomenis, kurti dokumento planą ir matyti kokie tekstai, naudojant sukurtą planą, yra generuojami. Prototipo ekrano vaizdas pateikiamas žemiau.

  
Pav 2: Dokumento plano redaktoriaus prototipas

Dokumento Plano įrankis (kitas angliškas darbinis pavadinimas *Augmented Writer*) leidžia atlikti šias funkcijas:

1. Pasirinkti generavimo kontekstą: marškinėliai, batai, kelnės ir taip toliau
2. Įkelti prekių inventoriaus duomenų pavyzdį (jis bus naudojamas generuojant tekstus konkrečioms prekėms)
3. Ekrano kairėje leidžiama konstruoti dokumento planą. Ekrano vaizde nepateiktas kitas darbo proceso pavyzdys, kai pirminis planas yra automatiškai sukonstruojamas iš pavyzdinio teksto. T.y. vartotojui nereikia kurti programos nuo pradžių, ji automatiškai sugeneruojama pagal pavyzdį.
4. Ekrano dešinėje, naudotojas gali matyti, pavyzdinius sugeneruotus tekstus. Šioje dalyje taip pat planuojama vartotojui suteikti įrankius leidžiančius patvirtinti ar atmesti, siūlomo teksto variantus.

Tolimesniuose projekto etapuose šis įrankis bus vystomas suteikiant vartotojui daugiau išraiškos priemonių:

1. Teksto šakojimas pagal sąlygas
2. Dokumento formatavimo valdymas
3. Galimybė perpanaudoti planus
4. Ir kiti panašūs komponentai

# NLG duomenų struktūrų administravimo įrankis

## Veiklos aprašas

Pirmojo projekto etapo metu nebuvo iki galo nustatyta, kaip bus administruojami NLG sistemos duomenys. Šio etapo metu buvo sukurtas administravimo įrankio prototipas skirtas visų domenų, dalyvaujančių generavimo, procese valdymas.

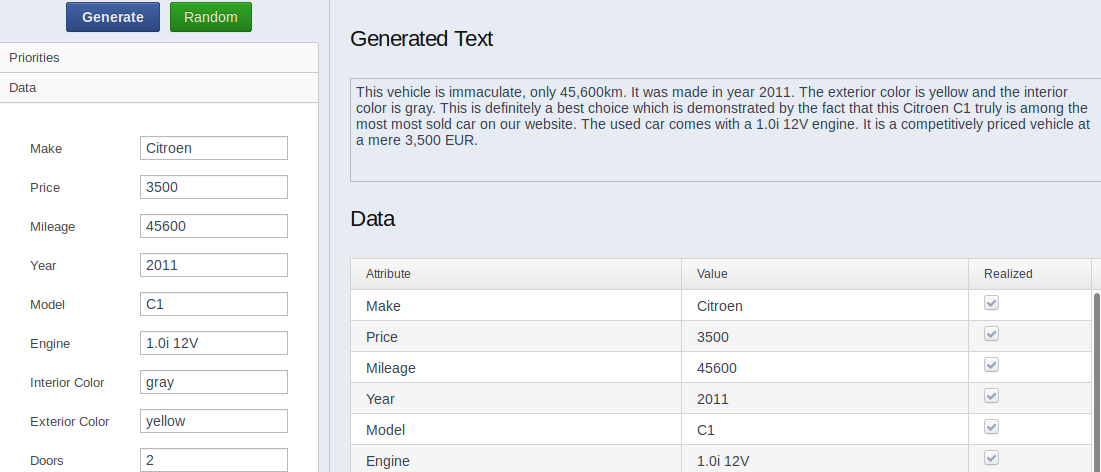
Ši priemonė turi atitikti duomenų ir procesų architektūrą aprašytą šio dokumento skyriuje – *5. RAGS Architektūra.*

## Veiklos vykdymo eiga

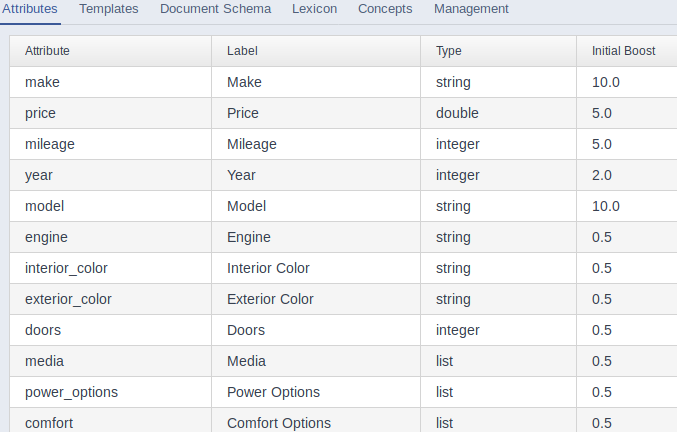
Remiantis aprašytomis NLG struktūromis ir duomenų administravimo poreikiais. Buvo aprašyti vartotojo sąsajos reikalavimai. Sekančiame skyriuje pateikiama sąsajos funkcionalumo santrauka.

## Rezultatai

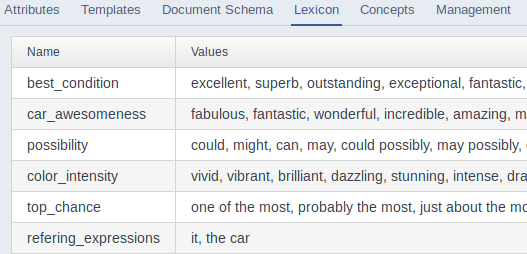
Generavimo sistemos ekrano vaizdas, rodantis, kaip valdomi realizuojami atributai bei teksto savybės (plano variacijų ribojimui). Ekrane galima matyti, kokie atributai yra naudojami ir koks tekstas yra sugeneruotas.

  
Pav 3: Teksto generavimo parametrų ir rezultatų peržiūra

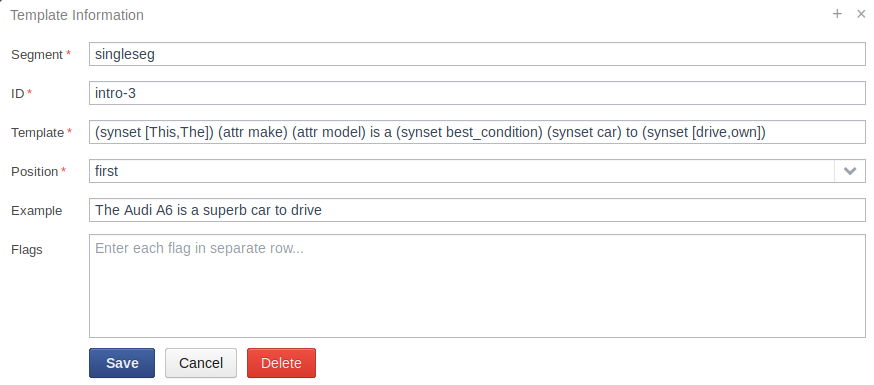
Generavimo sistemos ekrano vaizdas, rodantis, prekių atributų (žinių domenų bazės, naudojant RAGS terminologiją) valdymas

  
Pav 4: Prekių atributų valdymas

Sinonimų ir kitokių žodynų, naudojamų generuojant konkrečias išraiškas, valdymas.

  
Pav 5: Žodynų valdymo priemonė

Dalinių šablonų redagavimo langas. Pats šablonų formatas ir kūrimo principai buvo aprašyti pirmoje projekto ataskaitoje.

  
Pav 6: Dalinio šablono redagavimo langas

Tai yra pradinė administravimo įrankio versija ir ji nebus skirta naudoti galutiniam klientui. Ši sistema yra prieinama tik NLG paslaugą administruojantiems asmenims.

# RAGS architektūra

## Veiklos aprašas

NLG sistemos naudoja įvairius architektūras bei duomenų saugojimo ir reprezentavimo principus. Įgyvendindami prekių aprašymo NLG sistemą, mes siekiame naudoti universalesnę RAGS (angl: *Reference Architecture for Generation Systems*) architektūrą, kuri leidžia komponentų ir duomenų panaudojimą tarp skirtingų NLG sistemų. RAGS naudojimas leistų mums lengviau naudoti egzistuojančias NLG sistemas, ar, dalį mūsų sprendimo paskelbti atviru kodu, jei bus apibrėžtas tinkamas posistemis ir jo platesnis poreikis.

RAGS naudojimo tikslai, kurie turėtų palengvinti, pilną projekto įgyvendinimą:

1. Sukurti perpanaudojamus duomenų aprašus ir algoritmų kontraktus;
2. Aprašyti duomenų perdavimo ir procesų sinchronizavimo principus;
3. Įgalinti NLG modelių aprašymo formalizavimą, tam, kad jų veikimo principai ir ribojimai yra aiškiau apibrėžti

## Veiklos vykdymo eiga

RAGS architektūros apibrėžimo žingsniai:

* 1. Aukšto lygio lingvistinių duomenų tipų apibrėžimas. Duomenų tipų apibrėžimas neturi turėti jokių, nuo implementacijos priklausomų detalių.
  2. Žemesnio lygio duomenų tipų apibrėžimas. Šio apibrėžimo rėmuose, turi būti nustatyta kokie yra galimi konkretūs duomenų tipai ir galimos reikšmės. Kadangi NLG duomenys yra grafo pavidalo, atitinkamai žemesnio lygio duomenų specifikacija yra grafo formos.
  3. Tikslūs duomenų struktūrų aprašymai, taip, kaip jie bus grąžinami per NLG programines sąsajas (API).

## Rezultatai

Aprašant konkrečius duomenų tipus buvo remiamasi analize padaryta šiame darbe:

A Reference Architecture for Generation Systems, *Article in Natural Language Engineering · September 2004*

Įvardinti ir pradėtos įgyvendinti duomenų struktūros pagal šiuos aukšto lygio duomenų struktūras:

1. *Konceptualūs* *duomenys* - duomenų struktūros kurios yra už NLG sistemos ribų. Pirmoje šio projekto ataskaitoje (*5.1. Prekių aprašymo kalbinės struktūros analizė* skyrius) buvo aprašytos konkrečios, prekių aprašyme naudojamos duomenų struktūros. RAGS aukšto lygio koncepciniame kontekste atsiribojama nuo konkrečių savybių ir reikalaujama, kad koncepcinės duomenų struktūros turėtų tik šiuos laukus:
   1. **name** (atributo vardas): esybės vardas, kuris unikaliai nurodo esybės tipą (spalva, gamintojas, kaina);
   2. **gen-id** - NLG sistemos priskiriamas unikalus identifikavimo numeris;
   3. **kb-id** (angl: *knowlege base id*, liet: *žinių bazės id*) – atitinkamo duomens identifikatorius sistemoje, iš kurios yra skaitomi šie duomenys. E-parduotuvės atveju tai gali būti inventoriaus identifikacinis numeris.
   4. **type** (duomenų tipas) - neprivalomas laukas, kūrį NLG sistema turi galėti automatiškai nustatyti. Palaikomi tipai (atkreipti dėmesį į tai, kad šie duomenų tipai nėra žemesnio lygio programavimo duomenų tipai, t. y. s*kaičius* mūsų atveju gali būti bet kokio tipo skaitmuo: *integer, double, ir t.t.*:
      1. *data*
      2. *skaičius*
      3. *tekstas*
      4. *masyvas*
2. *Retoriniai duomenys*: Retoriniai duomenys aprašo sąsajas tarp teiginių dokumento viduje. Sąsajų tipai yra priklausomi nuo taikymo srities ir gali būti vienokie el-prekyboje ir visai kitokie medicinoje. Detalesnė šio komponento specifikacija bus atlikta vėlesniuose projekto etapuose. Šiuo metu apibrėžiamos šios, el-prekybos kontekste, veikiančios sąsajos:
   1. **description** (aprašymas) - atskiro prekės atributo aprašymas;
   2. **motivation** (motyvavimas) - priežasčių, kodėl reikia pirkti prekę, suteikimas;
   3. **compare** (palyginimas) - palyginimas su kitomis prekėmis, pvz *pigu* arba *brangu.*
3. *Dokumento duomenys* - dokumento struktūros aprašymas, kokie paragrafai ir kokia eile jie minimi tekste. Dokumento lygio duomenų struktūros yra labai artimos Retorinio lygio duomenims ir tolimesniuose projekto vystymo etapuose šie du duomenų tipai gali būti apjungti. Šiuo metu sistema yra įgyvendinta taip, kad retoriniai ir dokumento duomenys yra apjungti vienoje *Dokumento Plano* struktūroje.
4. *Semantinės duomenų struktūros* - konkrečių teiginių aprašymo duomenų struktūra; šios duomenų struktūros yra priklausomos nuo kalbos ir yra skirtingos lietuvių, anglų ir kitoms kalboms. Šios duomenų struktūros gali būti įgyvendintos dviem būdais:
   1. **Taisyklinis**. Šiuo metu turimas NLG sistemos prototipas veikia šiuo būdu, kur semantinė duomenų struktūra yra aprašyta per dalinį šabloną:

*/ Objektas: marškinėliai*

*Komunikacinis tikslas: aprašymas /*

*---*

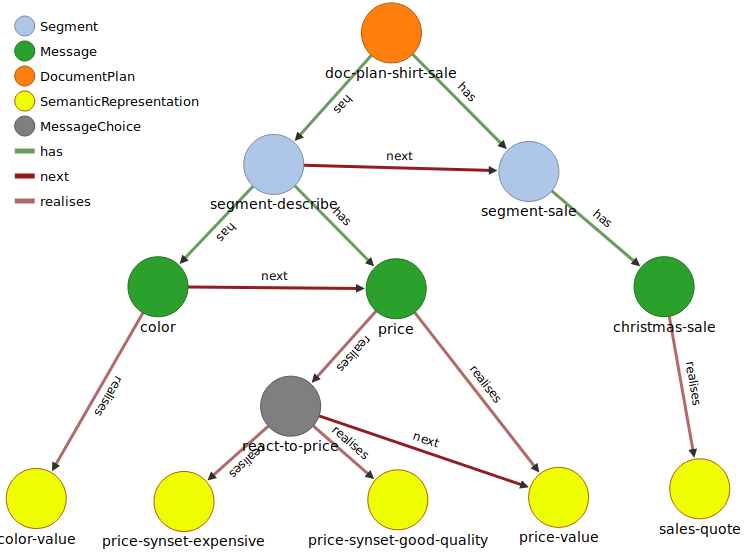
*(attr color) marškinėliai*

*gamintojas (synset quality)* *(attr designer)*

* 1. **Giliuoju mokymu paremtas modelis**.Alternatyvus būdas yra išmokti šias reprezentacijas iš esamų aprašymų pavyzdžių. Šiuo metu yra vykdomi eksperimentai tokio modelio paruošimui, tačiau giliojo mokymo technologija nėra gerai pritaikyta naujovių generavimui.

1. *Sintaksės duomenų struktūros* - duomenys aprašantys, kaip turi būti generuojamas sakinys. Šios duomenų struktūros buvo kuriamos pirmojo etapo metu ir yra aprašytos *5.3 Teksto realizavimas* skyriuje
2. *Citatos duomenų struktūra* - labai paprasta duomenų struktūra atspindinti tikslias frazes, kurios turi būti naudojamos sugeneruotame tekste be jokių pakeitimų.

RAGS architektūroje apibrėžiamos duomenų struktūros yra grafo tipo, todėl jų įgyvendinimui buvo pasirinkta grafo duomenų bazė Orient DB (<https://orientdb.org/>, tačiau prie būtent šios grafo duomenų bazės nėra prisirišama, naudojant universalią darbo su grafu kalbą - *Gremlin*). Sukurto duomenų modelio pavyzdys pateikiamas žemiau esančiame paveiksle.

  
Pav 7: NLG plano reprezentacija grafo duomenų bazėje

Grafo duomenų bazės struktūra bus tobulinama sekančio etapo metu, todėl šiuo metu detaliai nedokumentuojama. Pagrindiniai principai yra šie:

1. **Grafo elementai** (angl: *vertices*):
   1. *DocumentPlan* – pagrindinis planą reprezentuojantis elementas;
   2. *Segment* – dokumento segmentai, atspindintys pranešimo tikslus (aprašas, pardavimas)
   3. *Message* – pranešimai (faktai) apie kuriuos kalbama dokumente
   4. *SemanticRepresentation* – kalbinis fakto atvaizdavimas
2. **Grafo ryšiai** (angl: *edges*):
   1. *has* – ryšis nusakantis *visumos-dalies* santykį.
   2. *realizes* – realizavimo ryšis, rodantis, kad žemesnio lygio komponentas įgyvendina aukštesnįjį. Pvz Message:price yra realizuotas būdvardžiu iš sinonimų rinkinio ir kainos reikšme.

RAGS reprezentavimo grafo duomenų bazėje sistema bus toliau tobulinama tolimesniuose etapuose.

# NLP modelių valdymas

## Veiklos aprašas

Pirmoje projekto ataskaitoje 5*.5 Natūralios Kalbos Analizės Sistema* skyriuje buvo aptartos technologijos, kurios turės būti naudojamos prekių aprašymo kalbos analizei. Ataskaitoje aprašyti modeliai bus kuriami ir tobulinami viso projekto metu, tačiau jų valdymui ir programiniam naudojimui reikalingos sąsajos leidžiančios NLP modelių registravimą ir iškvietimą.

## Veiklos vykdymo eiga

Remiantis pirmosios ataskaitos anotavimo rezultatais, buvo įgyvendinti keli eksperimentiniai kalbos modeliai galintys tekste atpažinti šias esybes:

1. spalva
2. naudotos medžiagos
3. kaklo kirpimo tipas
4. *ateityje visi reikalingi drabužių srities kalbiniai modeliai*

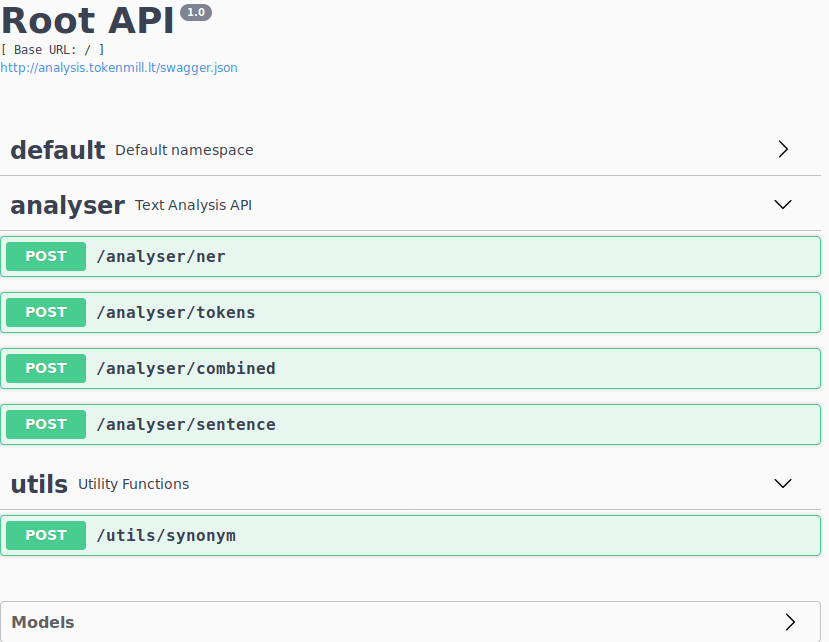
Šiems modeliams, ir turint mintyje neribotą modelių kiekį, buvo įgyvendinta jų administravimo ir naudojimo sąsaja.

## Rezultatai

NLP servisas šiuo metu yra pasiekiamas šiuo adresu:

http://analysis.tokenmill.lt/

Visos serviso funkcijos yra dokumentuotos naudojant Swagger dokumentavimo įrankį:

  
Pav 8: NLP serviso Swagger ekrano vaizdas

Šiame dokumente nepateikiamas viso funkcionalumo ir REST API aprašymas, nes jis yra prieinamas per Swagger specifikaciją. Žemiau, kaip pavyzdys, pateiktas tik vieno serviso detalus aprašymas.

Žemiau esančiame ekrano vaizde matyti:

1. REST operacijos aprašas
2. Operacijos įvesties duomenų tipų aprašymas
3. Rezultato duomenų aprašymas
4. Papildoma dokumentacija

  
Pav 9: NLP paslaugos aprašo pavyzdys

Apibendriant, NLP paslaugų paketas leidžia:

1. atlikti teksto analizę, gaunant *stand-off* (tai toks anotavimo būdas, kai anotacijų duomenų struktūros pateikiamos atskirai nuo anotuojamo teksto, daugiau čia https://wiki.tei-c.org/index.php/Stand-off\_markup) tipo lingvistines anotacijas.
2. gauti registruotų sinonimų žodynus
3. registruoti esybių anotavimo modelius, priregistruoti modeliai bus prienami per *stand-off* anotacijas.

# **Įvykdyti paslaugų pirkimai**

Amazon AWS serveriai ir paslaugos (dalis paslaugų šiuo metu nepereina nemokamo naudojimo limito):

1. *Amazon Lambda* – paslaugos ir NLG operacijos
2. *Amazon ES* – tekstynų saugojimas
3. *Amazon VCP* – saugumo infrastruktūra
4. *Amazon DinamoDB* – darbinių duomenų saugykla

# Užduočių atlikimas

Pateikti informaciją, kokios užduotys buvo atliktos projektą vykdančių asmenų, kas jas atliko, kiek laiko tam skyrė, naudoti lentelės formą.

| **Nr.** | **Užduotis** | **Darbuotojai/darbovietė** | **Laiko sąnaudos valandomis** |
| --- | --- | --- | --- |
| 1. | RAGS analizė ir įgyvendinimas | Žygimantas Medelis, UAB “TokenMill” | 300 |
| 2. | RAGS analizė ir įgyvendinimas | Dainius Jocas UAB “TokenMill” | 200 |
| 3. | NLP modelių įgyvendinimas | Šarūnas Navickas UAB “TokenMill” | 519 |
| 5. | NLG administrasvimo sąsaja | Dainius Jocas UAB “TokenMill” | 240 |
| 7. | Dokumento plano redaktorius | Tomas Čerkaukas UAB “TokenMill” | 399 |
| 8. | Dokumento plano redaktorius | Žygimantas Medelis, UAB “TokenMill” | 211 |

# Pasiekti rezultatai

Kaip ir pirmame etape pagrindinis technologinis neapibrėžtumas yra efektyvus teksto planų valdymas. Dokumento plano atitikimas norimiems komunikacijos tikslams gali būti sprendžiamas įvairiais būdais. Pirmojo etapo metu buvo tiriamas ir įgyvendinamas visiškai automatizuotas planavimo modulis. Šio projekto metu buvo realizuotas planavimo modulis leidžiantis vartotojui nustatyti planavimo kryptį.

## Grafo planavimas – hibridinis modelis

Šiuo metu vyraujantis giliojo mokymo (angl: *deep learning*) metodas sunkai tinka uždaviniams kurie nepatenka į objektų klasifikavimo kategoriją. Todėl mūsų taikomas metodas leis užtikrinti naujo turinio generavimą. Kombinuojant pilnai automatizuotą ir žmogaus kuruojamą planavimą mes galėsime pasiekti tikslesnį komunikacijos tikslo ir sugeneruoto teksto atitikimą. Kaip matyti iš RAGS architektūros realizavimo grafo duomenų struktūrose, planavimo uždaviniai yra konvertuojami į grafo analizės uždavinius. Kur teksto generavimas susiveda į optimalaus kelio grafe paiešką.

Patvirtinant pirmojo etapo išvadą, kad gilusis mokymas negali spręsti kryptingo (t. y. atitinkančio griežtus komunikacijos tikslus) teksto generavimo uždavinių, norime pabrėžti, kad giliojo mokymo technologijos nėra visiškai atsisakoma ir sekančiuose projekto etapuose ji bus naudojama lokaliems teksto išraiškų generavimo uždaviniams. T.y. ši technologija bus naudojama konkrečių frazių parinkimui.

## Vizualinis programavimas kalbinių pranešimų modeliavimui

Svarbus šio projekto etapo pasiekimas yra pradėta įgyvendinti galimybė taikyti vizualinio programavimo principus, leidžiančius vartotojui aprašyti teksto planus. Šis komponentas bus vystomas dar bent du projekto vystymo etapus, tikslas yra sukurti specialią lingvistinio vizualaus programavimo kalbą, prieinamą vartotojams neturintiems gilių programavimo įgūdžių. Tačiau jau dabar, naudojant mūsų įrankį, galima kurti dokumento planus realizuojančius, pasirinktus komunikavimo tikslus.

## RAGS – komponuojama kalbos generavimo architektūra

RAGS architektūros detalizavimas yra kitas svarbus šio etapo rezultatas. NLG sistemos susideda iš daugelio sudėtingų, skirtingais principais veikiančių komponentų. Teisingas duomenų ir procesų modelis leidžia užtikrinti efektyvią komunikaciją tarp skirtingų komponentų, taip pat įgalina mūsų kuriamos sistemos palyginimą su kitais, dažniausiai akademiniame kontekste sukurtais, NLG sprendimais. Tai savo ruožtu atveria galimybes tam tikruose taškuose įgyvendinti NLG komponentų perpanaudojimą tarp skirtingų projektų.

## Infrastruktūra darbui su kalbos analizės projektais

Tęsiant praeito etapo darbą natūralios kalbos analizės srityje, šiame etape buvo realizuotas modulis, leidžiantis valdyti neribotą NLP modulių kiekį. Kadangi generavimo modulis turi turėti prieigą prie visos eilės kalbinių modelių, sklandus šio komponento veikimas užtikrina tiek patį generavimo greitį tiek mažina pastangas reikalingas NLG modelių kūrimui.

Norint minimizuoti pastangas reikalingas serverių priežiūrai visa infrastruktūra ir sprendimo veikimas yra pradedamas įgyvendinti naudojant *serverless* architektūra. Tai mums leidžia koncentruotis į NLG, NLP ir Planavimo uždavinių sprendimą, pačios infrastruktūros valdymą perkeliant į atitinkamos paslaugos tiekėją (šiuo atveju *Amazon AWS*)

# Nepasiekti rezultatai

Visi numatyti rezultatai buvo pasiekti.

# **Veiklos vykdymo metu atliktų tyrimų eigos pakeitimai**

Veiklos pakeitimų nebuvo.

# **Rezultatų naujumas ir nauda projektui**

Pagrindiniai pasiektų rezultatų naujumo aspektai:

1. Gilesnė RAGS architektūros analizė ir jos pritaikymas projekto reikmėms yra pagrindinis šio etapo rezultatas. Nors pati RAGS architektūra buvo sukurta 2006 metais iki šiol nėra (bent publikuojamų viešai) komercinių ar atviro kodo sprendimų. Pagrinde yra tik akademiniai NLP projektai naudojantys RAGS. Mūsų atliktas šios architektūros susiejimas su grafo duomenų bazėmis įgalina lengvesnį praktinį RAGS taikymą.
2. Pirmosios ataskaitos rezultatuose buvo pasiekta ši išvada

*Labai svarbus šio etapo tyrimo rezultatas yra daugialypio rašymo (angl: augmented writing) aplinkos poreikio identifikavimas.*

Šiame etape mes įgyvendinome tokio redaktoriaus prototipą. Kuris visiškai naujame kontekste panaudoja vizualinio programavimo kalbos paradigmą (angl: *Visual Programming Language)*, pritaikant ją natūralios kalbos programavimui. Tolimesnis šio komponento vystymas bus pagrindinė senkančio projekto etapo vystymo kryptis.

1. Paskiausias, bet labai svarbus, techninis projekto rezultatas yra eilės programinių sąsajų sukūrimas, kuris, kartu su projekto partneriais, leidžia pradėti integruoti sprendimą į platesnius panaudojimo scenarijus.