**Bandomoji misija „Neįmanoma misija - Analyzeris 007“**

**Sistemos specifikacija**

**Sistemos architektūra:**

Sistema sudaroma iš šių komponentų:

Frontend – klientinės dalies informacijos atvaizdavimui ir naudotojo interakcijom, kuriama su Vue

Backend – duomenų, gautų iš duomenų gavybos komponento apdorojimui, talpinimui į duomenų bazę, bei šių duomenų perdavimui į frontend, kuriama su Django

Duomenų gavybos komponentas – komponentas skirtas išgauti duomenis iš misijai duoto puslapio ir juos perduoti backend apdorojimui.

Duomenų bazė –gautų duomenų saugojimas, naudojama Mongo dėl naudojimo paprastumo.

Sistemos architektūros diagrama:

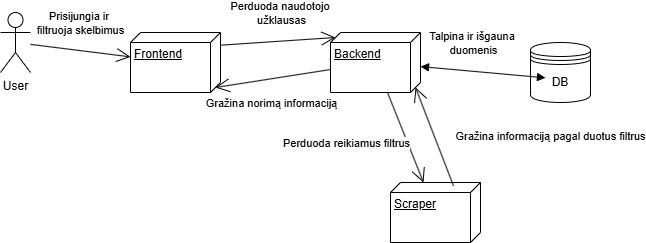


Figure Sistemos architektūros diagrama

**Duomenų bazės struktūra:**

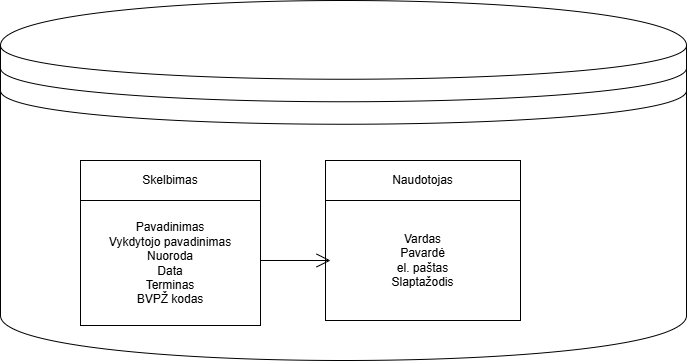


Figure Prototipinės duomenų bazės struktūra

**Vartotojai:**

Turėtų būti bent dvi naudotojo rolės:

**Admin rolė:**

Gali tvarkyti naudotojus. (bus ateityje)

Valdo duomenų gaviklio leidimus, prieigą, konfiguruoja sistemą. (bus ateityje)

**User rolė:**

Prisiregistravęs naudotojas gali prisijungti ir prisijungęs žiūrėti įmonių pirkimų informaciją.

Prisijungęs naudotojas gali filtruoti įmonių pirkimo informaciją pagal savo norimus filtrus.

Gali prisiregistruoti/prisijungti prie sistemos

Šiam prototipui nėra skirstymo rolėmis, bet kas gali daryti bet ką.

**Vartotojų istorijos:**

**Kaip naudotojas** norėčiau prisiregistruoti prie sistemos, kad galėčiau prisijungti prie sistemos.

**Kaip naudotojas** norėčiau prisijungti prie sistemos, kad galėčiau matyti norimą informaciją.

**Kaip naudotojas** norėčiau pamatyti sąrašą, su mane dominančia informacija

**Kaip naudotojas** norėčiau pamatyti atfiltruotą informaciją pagal mano pačio sukurtus filtrus.

**Kaip administratorius** norėčiau tvarkyti naudotojus.

**Testavimo planas**

**Testavimo tikslai:**

Duomenų gavimo komponentas testuojamas, kad įsitikintume, kad jis funkcionuoja korektiškai ir nuskaito ir išsaugo informaciją teisingai.

Backend testuojama, kad jo API funkcionuoja taip, kaip priklauso.

Backend testuojama, kad autentikacija veiktų kaip priklauso

DB testuojama tam, kad įsitikintume, kad pasirinkta DB palaiko duodamus duomenis

Frontend testuojama tam, kad įsitikinti, kad naudotojai gali juo naudotis.

Frontend testuojama tam, kad įsitikinti, jog duomenys atvaizduojami korektiškai

Bendrai sistema testuojama, kad užtikrinti jos greitaveiką padidėjusiam krūviui.

**Testavimo įrankiai:**

API endpoint testavimui naudojamas Postman

DB testavimui panaudoti galima SQLite Viewer internetinį įrankį („https://inloop.github.io/sqlite-viewer/“), taip pat pamatoma ar duomenų gavimo komponentas nuskaito informaciją korektiškai

Frontend testavimui galima naudoti Cypress

**Testuojami komponentai:**  
Duomenų gavybos komponentas:

Unit testai – Šiais testais įsitikinama tinkamas gavybos komponento funkcionalumas jo dalyse: HTML pagavimas, tuščių ir netinkamų laukų palaikymas.

Integration testai – įsitikinama gavybos komponento gebėjimas nuskaityti duomenis iš svetainės, bei korektiškas duomenų priskyrimas/išsaugojimas DB

Performance testai – įvertinamas gavybos komponento skaitymo greitis.

**Backend:**

Unit testai – testuojamos API endpoint funkcijos, tikrinami edge case jeigu tarkim duodama netinkama informacija laukeliuose.

Integration testai – testuojami API endpointai su DGK ir tikrais duomenimis tam, kad būtų užtikrintas suderinamumas.

Security testai – užtikrinama, kad visi endpointai turėtų reikalingą autentikaciją

**Frontend:**

Unit testai – Ištestuoti atskirus komponentus izoliuotai

Integration testai – Ištestuoti duomenų gavimą iš backend API ir rodymą ekrane

Usability testavimas – Ištestuoti UI keliuose įrenginiuose (jei įmanoma)

**Galimi tetavimo scenarijai:**

Prisijungimas:

Pateikiami tinkami prisijungimo duomenys -> naudotojas sėkmingai prisijungia.

Pateikiami neteisingi prisijungimo duomenys -> naudotojas gauna pranešimą apie neteisingus duomenis.

Pateikiamas neegzistuojantis naudotojas -> naudotojas gauna pranešimą apie neteisingus duomenis.

Trūksta duomenų pateikiant -> naudotojas gauna pranešimą apie neteisingus duomenis.

Registracija:

Pateikiami visi teisingai uzpildyti laukai -> naudotojas gauna pranešimą apie sėkmingą registraciją

Pateikiama esančio naudotojo informacija -> naudotojui pranešama, kad naudotojas su tokiais pačiais duomenimis egzistuoja

Pateikiama tuščias laukas -> naudotojui pranešama, kad tuščias laukas negali būti tuščias.

Filtravimas:

Nuskaitymas nepateikiant filtro -> nuskaito informaciją

Nuskaitymas pagal filtrus -> nuskaito informaciją griežtai pagal filtrus.

Testavimas:

Kuriant backend dalį, buvo testuojami API routes su Postman programine įranga. Testavimas parodė, kad Django karkasas yra greitas, sąlyginai greitai sukuriamas ir suteikia programuotojui daug įrankių.

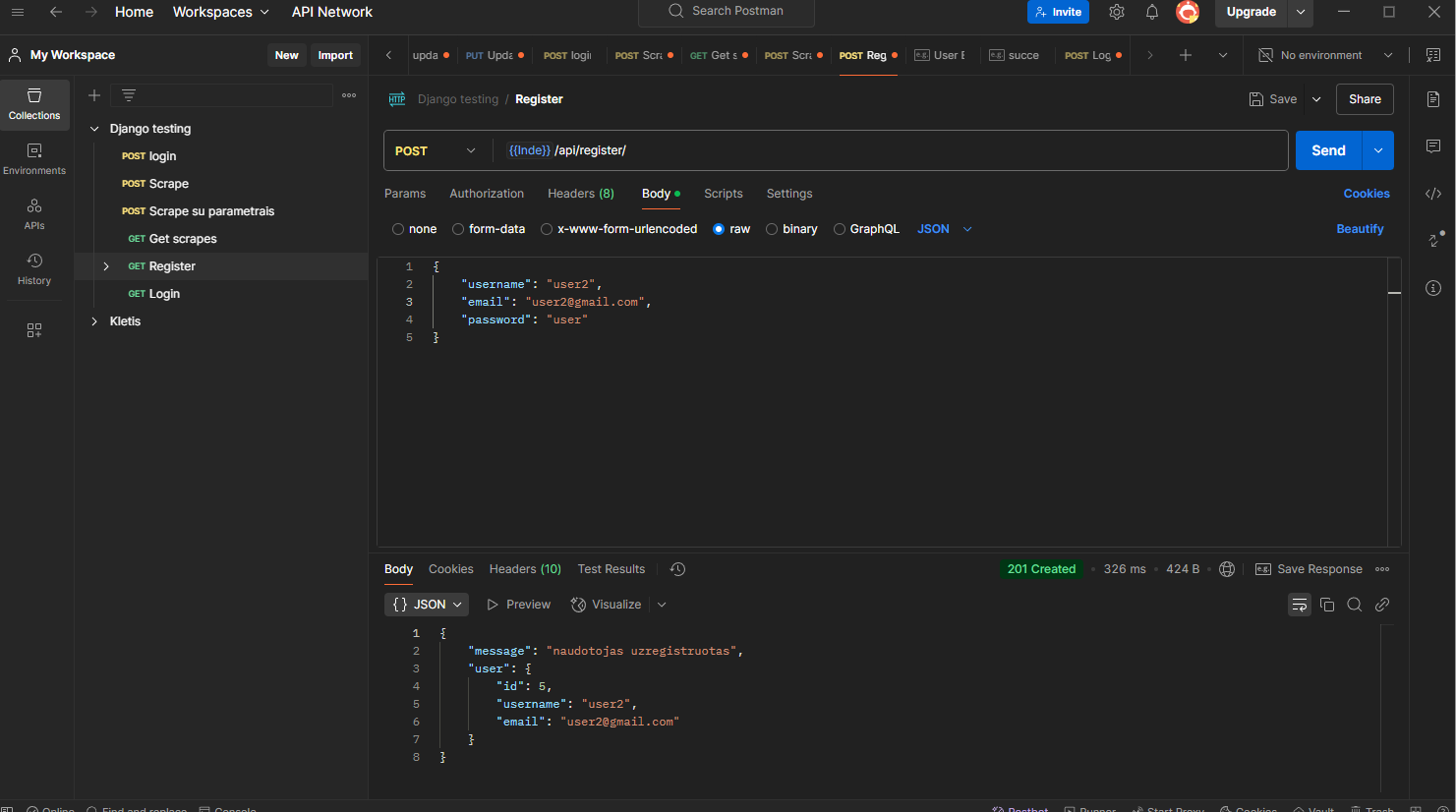


Figure Registracijos API endpoint testas

A screenshot of a computer

Description automatically generated

Figure Prisijungimo API endpoint testas

A screenshot of a computer

Description automatically generated

Figure Duomenų nuskaitymo be filtro API endpoint testas

A screenshot of a computer

Description automatically generated

Figure Parametrizuoto duomenų nuskaitymo API endpoint testas

A screenshot of a computer program

Description automatically generated

Figure Nuskaitytų nuomenų atvaizdavimo API endpoint testas

Vaizdai iš programos:

A screenshot of a computer

Description automatically generated

Figure Prisijungimo ekranas

A screenshot of a computer

Description automatically generated

Figure Registracijos ekranas

A screen shot of a graph

Description automatically generated

Figure Informacijos ekranas