



**Kauno technologijos universitetas**

Informatikos fakultetas

## **Projektų planavimo įrankis**

Baigiamasis bakalauro studijų projektas

---

**Nerijus Dulkė**

Projekto autorius

**Lekt. Dominykas Barisas**

Vadovas

---

**Kaunas, 2020**



**Kauno technologijos universitetas**

Informatikos fakultetas

## **Projektų planavimo įrankis**

Baigiamasis bakalauro studijų projektas

Programų sistemos (612I30002)

---

**Nerijus Dulkė**

Projekto autorius

**Lekt. Dominykas Barisas**

Vadovas

**Lekt. Rasa Mažutienė**

Recenzentė

---

**Kaunas, 2020**



**Kauno technologijos universitetas**

Informatikos fakultetas

Nerijus Dulkė

## **Projektų planavimo įrankis**

### **Akademinio sąžiningumo deklaracija**

Patvirtinu, kad mano, Nerijaus Dulkės, baigiamasis projektas tema „Projektų planavimo įrankis“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

---

(vardą ir pavardę įrašyti ranka)

---

(parašas)

Nerijus Dulkė. Projektų valdymo įrankis. Bakalauro studijų baigiamasis projektas vadovas lekt. Dominykas Barisas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informatikos mokslai, Programų sistemos.

Reikšminiai žodžiai: progresyvus saityno įrankis, *Service Worker*, projektų valdymas, mobilūs įrenginiai, veikia be interneto.

Kaunas, 2020. 51p.

### **Santrauka**

Darbe pristatomas saityno įrankis skirtas projektų valdymui ir bendradarbiavimui. Įvade apžvelgiamas darbo aktualumas, iškeliamas tikslas ir uždaviniai. Taip pat nagrinėjama projekto įgyvendinimo galimybės, konkurentai šiuo metu esantys rinkoje.

Projekto dalyje aprašomi sistemai keliami reikalavimai, techninė specifikacija, vartotojo sąsajos prototipai. Apžvelgiama ir projektavimo metodai ir eiga, bei sistemos projektas.

Vėlesniuose skyriuose aprašomi testavimo kriterijai, planas ir kaip jis buvo įgyvendinamas. Paminima kokio tipo automatiniai testai bus naudojami ir kaip jie testuoja sistemą.

Šiame darbe taip pat pateikiama dokumentacija su pavyzdžiais reguliariam vartotojui. Dokumentacijoje aprašoma ir kaip elgtis sistemos administratoriams, kaip pamatyti ir diagnozuoti klaidas.

Nerijus Dulkė. “Project Planning Tool”. Bachelor’s Final Degree Project / supervisor lect. Dominykas Barisas; Informatics Faculty, Kaunas University of Technology.

Study field and area (study field group): Computer Sciences, Software Systems.

Keywords: progressive web app, service worker, project management, mobile devices, offline compatible.

Kaunas, 2020. 51p.

### **Summary**

A tool for project management and collaboration will be presented in the thesis. The paper begins with an introduction explaining the relevance and main problems of the project. Project implementation feasibility and competitors in the rink are also reviewed in this topic.

The project section starts with the review of system requirements and technical specifics and then moves to reviewing user interface prototypes. Project section continues with the tools and techniques used for modelling and then finishes with a system’s project and analysis.

In the later sections project testing is introduced, including testing criteria, testing plan and how it will be used. Testing section also describes in detail about what type of automated tests where used and how are they testing this system.

Final section of the thesis is documentation. Workflows for a regular user are documented with examples. There is also a documentation section for system administrators to inspect and diagnose errors in the system.

## Turinys

<b>Lentelių sąrašas.....</b>	<b>7</b>
<b>Paveikslų sąrašas.....</b>	<b>8</b>
<b>Santrumpų ir terminų sąrašas.....</b>	<b>9</b>
<b>Įvadas .....</b>	<b>10</b>
<b>1. Analizė .....</b>	<b>11</b>
1.1. Techninis pasiūlymas .....	11
1.1.1. Sistemos apibrėžimas .....	11
1.1.2. Bendras veiklos tikslas .....	11
1.1.3. Sistemos pagrįstumas .....	11
1.1.4. Konkurencija rinkoje.....	11
1.1.5. Prototipai ir pagalbinė informacija.....	12
1.1.6. Ištekiai, reikalingi sistemai sukurti .....	12
1.2. Galimybių analizė .....	13
1.2.1. Techninės galimybės .....	13
1.2.2. Vartotojų pasiruošimo analizė .....	13
<b>2. Projektas.....</b>	<b>14</b>
2.1. Reikalavimų specifikacija.....	14
2.1.1. Komercinė specifikacija .....	14
2.1.2. Sistemos funkcijos.....	14
2.1.3. Vartotojo sąsajos specifikacija.....	17
2.1.4. Realizacijai keliami reikalavimai .....	20
2.1.5. Techninė specifikacija .....	21
2.2. Projektavimo metodai.....	22
2.2.1. Projektavimo valdymas ir eiga.....	22
2.2.2. Projektavimo technologija .....	22
2.2.3. Programavimo kalbos, derinimo, automatizavimo priemonės, operacinė sistemos .....	22
2.3. Sistemos projektas.....	23
2.3.1. Statinis sistemos vaizdas .....	23
2.3.2. Dinaminis sistemos vaizdas .....	25
<b>3. Testavimas .....</b>	<b>30</b>
3.1. Testavimo planas.....	30
3.2. Testavimo kriterijai .....	30
3.3. Komponentų testavimas .....	30
3.4. Integracinis testavimas .....	30
3.5. Vartotojo sąsajos testavimas .....	31
<b>4. Dokumentacija naudotojui .....</b>	<b>35</b>
4.1. Apibendrintas sistemos galimybių aprašymas.....	35
4.2. Vartotojo vadovas .....	35
4.3. Diegimo vadovas.....	45
4.4. Administravimo vadovas.....	47
<b>Rezultatai ir išvados.....</b>	<b>50</b>
<b>Literatūros sąrašas.....</b>	<b>51</b>

## **Lentelių sąrašas**

<b>lentelė 1</b> Konkurentų apžvalga.....	12
<b>lentelė 2</b> Vartotojo prisijungimo prie sistemos testavimas .....	32
<b>lentelė 3</b> Projekto laiko juosto vaizdavimo testavimas.....	33
<b>lentelė 4</b> Projekto kategorijos kūrimo testavimas .....	33
<b>lentelė 5</b> Projekto kategorijų grupės kūrimo testavimas.....	33
<b>lentelė 6</b> Projekto etapų kūrimas .....	34
<b>lentelė 7</b> Projekto užduočių kūrimas .....	34

## Paveikslų sąrašas

pav. 2.1 Sistemos panaudojimo atvejų UML diagrama .....	14
pav. 2.2 Autentifikacijos posistemės panaudojimo atvejų diagrama.....	15
pav. 2.3 Projektų posistemės panaudojimo atvejų diagrama.....	15
pav. 2.4 Administracijos posistemės panaudojimo atvejų diagrama .....	16
pav. 2.5 Prisijungimo langas .....	17
pav. 2.6 Sąrašo langas .....	18
pav. 2.7 Projekto laiko juostos langas.....	19
pav. 2.8 Projekto objektų pridėjimo iššokantis langas.....	20
pav. 2.9 Iteracinio programinės įrangos kūrimo modelio iliustracija .....	22
pav. 2.10 UML sistemos diegimo diagrama.....	23
pav. 2.11 Sistemos UML paketų diagrama .....	24
pav. 2.12 Duomenų bazės UML diagrama.....	25
pav. 2.13 Prisijungimo naudojant Google paskyrą sekų UML sekų diagrama .....	26
pav. 2.14 Pagrindinio lango užkrovimo UML sekų diagrama .....	27
pav. 2.15 Įrenginio registravimo momentiniams pranešimams UML sekų diagrama .....	28
pav. 2.16 Pranešimo gavimo UML sekų diagrama.....	29
pav. 3.1 API padengimas integraciniais testais .....	31
pav. 3.2 Vartotojo sąsajos automatiniai testai .....	32
pav. 4.1 Vartotojo prisijungimo langas .....	36
pav. 4.2 Pagrindinis sistemos langas, neturint jokių projektų .....	37
pav. 4.3 Naujo projekto sukūrimo iššokantis langas.....	38
pav. 4.4 Projektų sąrašas .....	39
pav. 4.5 Kategorijos kūrimo iššokantis langas .....	40
pav. 4.6 Užduoties kūrimo forma .....	41
pav. 4.7 Kategorijų grupės kūrimo forma .....	42
pav. 4.8 Projekto etapų pridėjimo forma.....	43
pav. 4.9 Užpildyto projekto laiko juostos pavyzdys .....	44
pav. 4.10 Projekto informacijos langas.....	45
pav. 4.11 Sistemos diegimo žingsniai naudojant <i>GitHub Actions</i> .....	46
pav. 4.12 <i>GitHub</i> konfigūracijos .....	47
pav. 4.13 Klaidos žurnalų langas .....	48
pav. 4.14 Išsamios klaidos detalės .....	49



### Santrumpų ir terminų sąrašas

- **UML** (angl. Unified Modeling Language) – labiausia paplitęs programinės įrangos specifikavimo standartas, skirtas projektuoti sistemas.
- **JWT** (angl. Json Web Token) – prieigos žetonų aprašymui ir naudojimui skirtas standartas, pagrįstas JSON kalbos sintakse.
- **API** (angl. Application Programming Interface) – tai programinė sąsaja, kurią suteikia sistema, kad programuotojai galėtų pasiekti sistemos funkcionalumą.
- **PWA** (angl. Progressive Web App) – progresyvi saityno aplikacija, palaikoma visų modernių interneto naršyklių ir suteikianti vartotojo patirtį panašią į tam įrenginiui pritaikytą programą (1).
- **W3C** (angl. World Wide Web Consortium) – tarptautinė organizacija, leidžianti programinės įrangos standartus saitynui.
- **SW** (angl. Service Worker) – scenarijus vykdomas fone. Šis scenarijus yra vykdomas naršyklės, tačiau atskirai nuo saityno puslapio, suteikiantis sistemai daug funkcionalumo, nereikalaujančio vartotojo įsikišimo.
- **ACU** (angl. Azure Compute Units) – specialūs sistemos galios skaičiavimo vienetai naudojami *Azure* platformoje.

## Ivadas

Saitynas yra labiausiai paplitusi platforma, leidžianti naudotis programine įranga visose operacinėse sistemose, kuriose veikia interneto naršyklė. Ši platforma nepriklauso jokiai vienai kompanijai ir joje galima patalpinti vieną programos versiją pasiekiamą beveik visiems įrenginiams. Deja dauguma dabartinių saityno įrankių susikoncentruoja tik į vieną platformą – tai yra kompiuterius. Tačiau su šiuolaikinėmis saityno technologijomis galima sukurti įrankius veikiančius kaip kiekvienam įrenginiui pritaikyta programinė įranga.

Darbo tikslas – sukurti projektų planavimo įrankį saitynui, pritaikyta naudoti visuose įrenginiuose ir išnaudojant šios platformos galimybes. Šiam tikslui įgyvendinti iškelti šie tikslai:

1. Išanalizuoti šiuo metu rinkoje esančius panašius įrankius;
2. Aprašyti reikalavimus išnaudojančius modernios saityno sistemos galimybes;
3. Sudaryti projektą, remiantis analizės rezultatais;
4. Realizuoti bei ištestuoti įrankį;
5. Paruošti įrankio dokumentaciją.

Darbo struktūra susideda iš šių dalių: analizė, projektas, testavimas, dokumentacija. Analizės skyriuje aprašomi sistemos reikalavimai, palyginami rinkoje esantys konkurentai ir patikrinamos įgyvendinimo galimybės. Projekto dalyje apžvelgiama techninė specifikacija, projektavimo metodai bei sistemos projektas. Testavimo skyrius susideda iš testavimo plano, kriterijų ir skirtingų testavimo būdų aprašymų. Paskutinis darbo skyrius yra dokumentacija, kurioje aprašyta kaip naudotis sukurtu įrankiu paprastiems vartotojams ir administratoriams.

Realizuoto projekto apimtis pamatuota kodo eilučių skaičiumi. Visą sistemos programinį kodą sudaro beveik 12 tūkst. eilučių. API dalį rašyta *TypeScript* kalba sudaro 5 tūkst. eilučių, saityno dalį naudojančią *Vue* biblioteką sudaro virš 6 tūkst. eilučių. Likusios eilutės sudaro konfigūracijų, stilių ir panašius failus.

## **1. Analizė**

Šiame skyriuje apibrėžiama sistemos idėja bei tikslas, sistema palyginama su kitomis panašiomis sistemomis rinkoje, bei aprašomos galimybės ir ištekliai.

### **1.1. Techninis pasiūlymas**

#### **1.1.1. Sistemos apibrėžimas**

Projektų planavimo įrankis – tai sistema, kuri leis patogiai vizualizuoti laiko juostoje ir valdyti vykdomus projektus, bei bendradarbiauti su kitais žmonėmis, naudojant bet koki įrenginį. Sistema skirta patogiai atvaizduoti, valdyti, bei paskirstyti komandos nariams projekto darbus, įvairius etapus bei terminus. Darbai gali būti patogiai suskirstyti į kelių lygių kategorijas ir atvaizduoti laiko juostoje. Pats įrankis yra progresyvioji saityno programa, tad gali būti pasiekama bet kur, bei išsaugota kaip programėlė ir gali būti naudojama net neturint interneto ryšio.

#### **1.1.2. Bendras veiklos tikslas**

Šios sistemos tikslas yra sukurti patogią ir intuityvią sistemą, kuria būtų galima naudotis skirtinguose įrenginiuose be papildomų pastangų, išnaudojant kuo daugiau progresyviosios saityno aplikacijos funkcijų.

Taip pat šiuo projektu siekiama uždirbti pelno, apribojant nemokamos versijos funkcionalumus, pvz.: nemokamoje versijoje galima susikurti ne daugiau 3 projektų.

#### **1.1.3. Sistemos pagrįstumas**

Egzistuoja daug vartotojų, kurie dirba prie kelių projektų vienu metu, tai dažnai priverčia žmones galvoti apie darbą, bei dirbti ne tik ofise, bet ir įvairiose vietose. Ne visada galima išsitraukti kompiuterį ir pradėti dirbti. Šis projektų planavimo įrankis gali veikti ne tik kompiuteriuose bet ir mobiliuosiuose įrenginiuose. Taip pat gali veikti ir be interneto, tad leidžia be jokio vargo išsitraukti telefoną pietaujant kavinėje ar skrendant lėktuve ir patikrinti projekto pabaigos terminus ir kitą informaciją.

#### **1.1.4. Konkurencija rinkoje**

Ieškant panašių įrankių rasta trys panašiausi variantai – „AHA!“, „Roadmunk“ ir „Toggl Plan“. Ši projektų planavimo sistema susitelkia į mobiliuosius įrenginius bei prieinamumą iš bet kur.

Pirmasis panašus įrankis yra „AHA!“. Šis įrankis turi daugumą tokių pačių funkcionalumų, bei mobilią aplikaciją. Tačiau „AHA!“ mobilioji aplikacija turi mažiau funkcionalumo nei WEB įrankis, tai taip pat atsispindi paskaičius vartotojų atsiliepimus apie aplikaciją – dauguma vartotojų yra nepatenkinti limituotu funkcionalumu (2). Šio įrankio kaina prasideda nuo 59\$ vienam vartotojui per mėnesį, tačiau turi specialius pasiūlymus startuoliams (3).

Antrasis panašus įrankis yra „Roadmunk“. Ši sistema pasižymi labai plačiu funkcionalumu, bei pritaikymu įvairiems atvejams. Taip pat turi atvirą API integracijoms, perkant *Professional* planą. Mobiliosios aplikacijos ši sistema neturi. Kaina prasideda nuo 19\$ per mėnesį, tačiau su stipriai ribotu funkcionalumu, pilną sistemos funkcionalumą galima gauti tik perkant *Professional* arba *Enterprise*

planus kurie prasideda nuo 99\$ vienam vartotojui per mėnesį, tai padaro šį įrankį brangiausią iš visų lyginamų (4).

Paskutinė lyginama sistema yra „*Toggl Plan*“. Ši sistema yra pakankamai nedidelė, tačiau turi pakankamai funkcionalumo. „*Toggl Plan*“ turi mobiliąją aplikaciją, tačiau tik Android sistemai. Šios aplikacijos nėra *Google Play* programėlių parduotuvėje, todėl sunku surasti atsiliepimų apie programėlę. Šis įrankis turi nemokamą planą su limituotu funkcionalumu, o pilną įrankį galima gauti už 9\$ vienam vartotojui per mėnesį (5).

**lentelė 1** Konkurentų apžvalga

Lyginimo kriterijai	<i>AHA!</i>	<i>Roadmunk</i>	<i>Toggl Plan</i>
Mobili aplikacija	+	-	Tik Android
Mobili aplikacija turi visą funkcionalumą	-	-	?
Kaina vienam vartotojui	Nuo 59\$/mėnesį	Nuo 19\$/mėnesį	9\$/mėnesį
Gaunama pakankamai funkcionalumo už mokamą planą	+	-	+
Turi nemokamą planą	Tik startuoliams	-	+

1 lentelėje palyginami visi panašūs įrankiai pagal tam tikrus kriterijus. Kriterijai susiję su mobiliaisiais įrenginiais bei kaina, nes ši projektų planavimo sistema fokusuojasi į mobilius įrenginius bei prieinamą kainą vartotojams, ne tik įmonėms.

### 1.1.5. Prototipai ir pagalbinių informacija

Vartotojo sąsajai sukurti nebuvo naudojami prototipai, tačiau buvo naudojamas *vue-material* paketas, suteikiantis „*Material Design*“ stiliaus komponentus (mygtukus, formas laukus ir pan.).

API taip pat kuriamas nenaudojant jokio prototipo.

### 1.1.6. Ištekliai, reikalingi sistemai sukurti

Sistema susideda iš 3 skirtingų posistemių:

- Autentifikacijos
- Projektų
- Administracijos

Didžiausią projekto dalys sudarys projektų posistemė, nes ten bus įgyvendintas pagrindinis funkcionalumas. Taip pat projektų posistemė turės sudėtingiausią iš visų posistemių vartotojo sąsają. Administracijos posistemės apimtis pati mažiausia ir neužtruks per daug laiko. Autentifikacija sudėtingiausias reikalavimas yra prisijungimas naudojant Google paskyrą, kas taip pat gali užtrukti daugiau laiko. Įvertinus visas šias posistemas sistemos kūrimas turėtų užtrukti apie 500 valandų. Atsižvelgiant į šį vertinimą ir laiko apribojimus, sistemai sukurti užtenka vieno žmogaus.

## **1.2. Galimybių analizė**

### **1.2.1. Techninės galimybės**

Sistemai sukurti naudojama populiarios technologijos – *Vue.js* ir *Node/Express.js*. Abi šios technologijos yra plačiai paplitusios ir turi didelę bendruomenę. Tai reiškia, kad technologijos yra subrendusios ir aktyviai vystomos, ir neturėtų sudaryti jokių kliūčių projekto eigoje. Taip pat, dėl šios bendruomenės yra įvairių paketų reikalingų sistemos integracijoms bei vystymui palengvinti.

Visą sistemą planuojama talpinti *Microsoft Azure* debesų platformoje, tad jokių papildomų infrastruktūrinių kliūčių nekils.

Atsižvelgiant į naudojamą technologiją ir pasirinktą talpinimo platformą, galima teigti, kad visos techninės galimybės yra išpildytos ir jokių kliūčių nesudarys.

### **1.2.2. Vartotojų pasiruošimo analizė**

Šio projekto pagrindinė auditorija yra projektų vadovai, tačiau tuo neapsiribojama ir naudoti gali bet kokie asmenys. Sistema turi intuityvią vartotojo sąsają, tad papildomo pasiruošimo, tam, kad naudotis sistema, nereikės.

## 2. Projektas

### 2.1. Reikalavimų specifikacija

#### 2.1.1. Komerčinė specifikacija

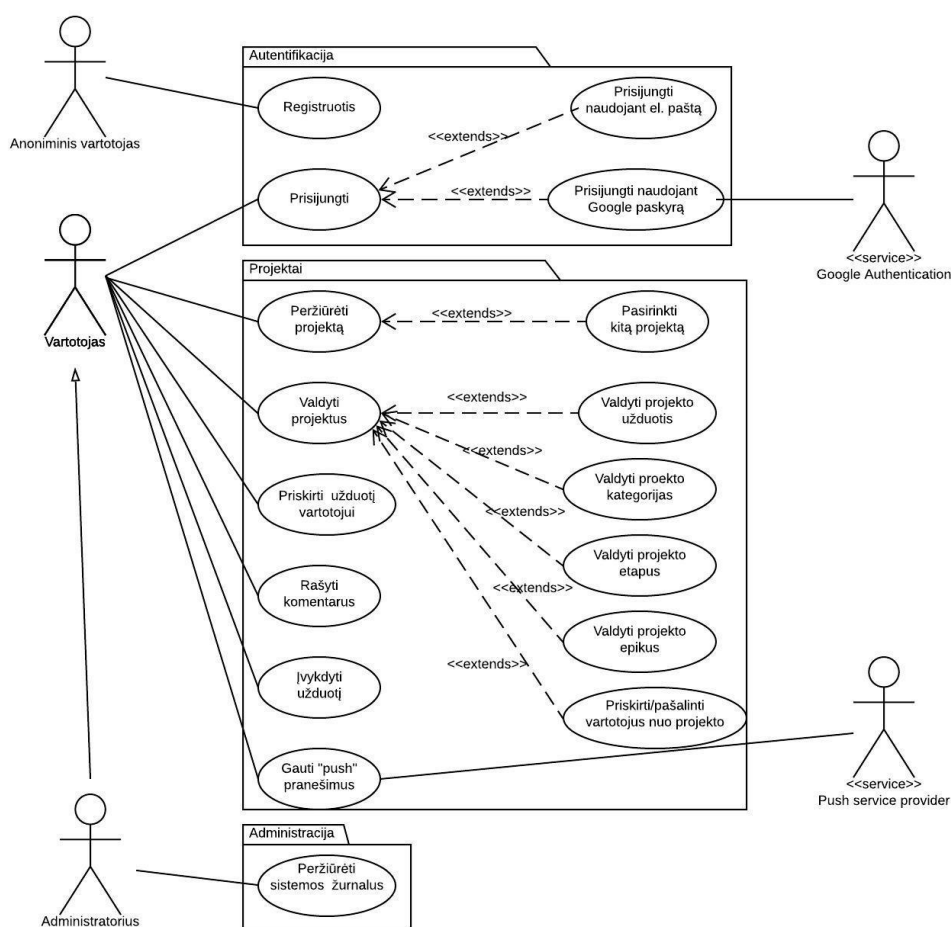
Tai yra mokslinis projektas, kuriamas bakalauro baigiamajam darbui. Projekto užsakovas – darbo vadovas lekt. Dominykas Barisas. Projektas skirtas naudoti, projektų vadovams, bei kitiems žmonėms norintiems geriau organizuoti asmeninius ar kitus projektus. Projekto biudžetas nėra apibrėžtas. Numatyta pabaigos data – 2020m. gegužės 18 diena. Iki šios datos projektas turi būti ištestuotas ir naudojamas.

#### 2.1.2. Sistemos funkcijos

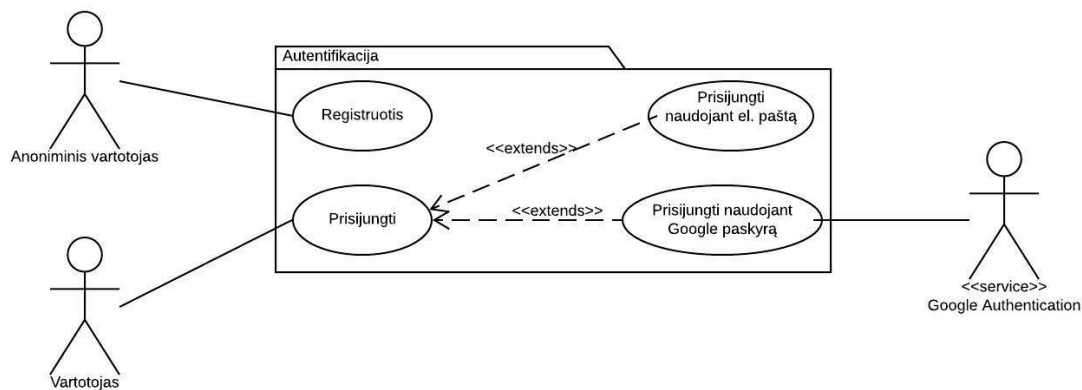
Sistemos funkcijos suskirstytos į tris posistemes:

1. Autentifikacijos
2. Projektų
3. Administracijos

Visos šios posistemės ir funkcijos pavaizduotos *UML* panaudojimo atvejų diagramomis 2.1 paveikslėlyje.

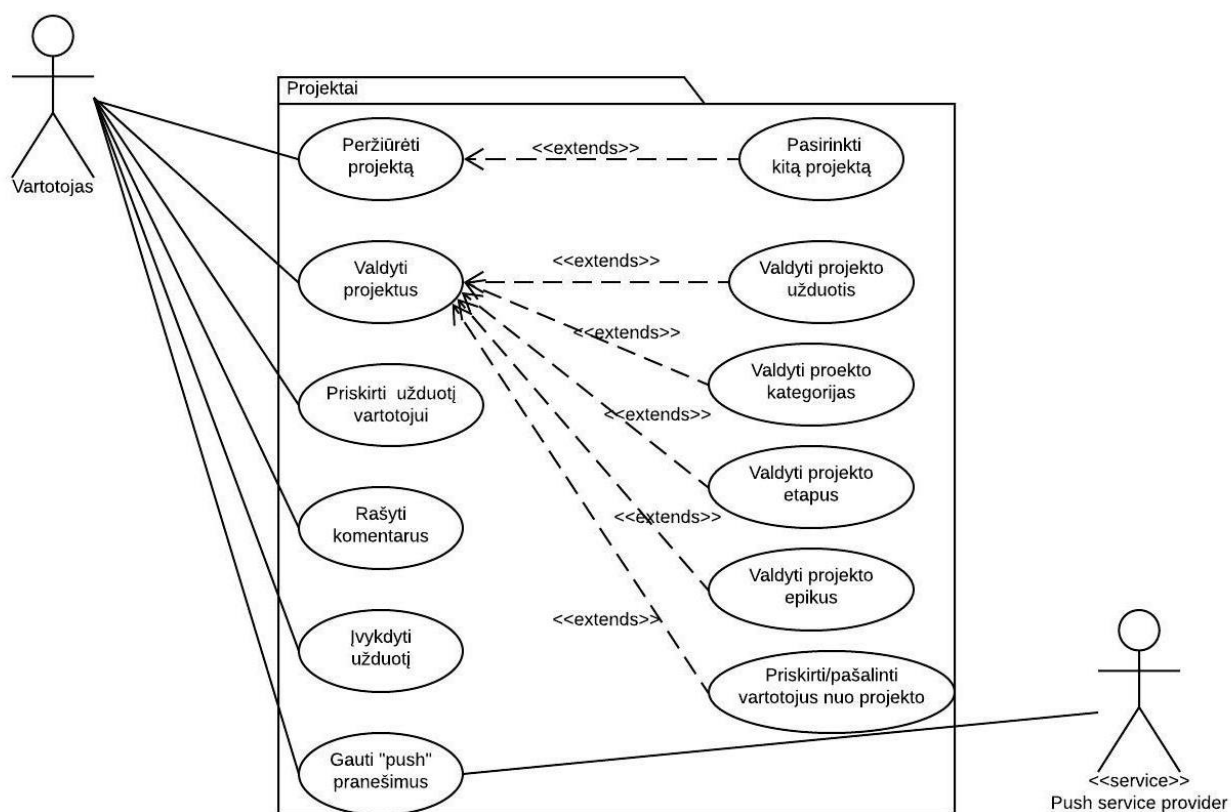


pav. 2.1 Sistemos panaudojimo atvejų UML diagrama



pav. 2.2 Autentifikacijos posistemės panaudojimo atvejų diagrama

Autentifikacijos posistemė apima prisijungimą ir registraciją į sistemą. Prisijungiant išduodamas JWT prieigos žetonas, vėliau naudojamas identifikuoti vartotoją darant įvairias užklausas. Į sistemą galima prisijungti naudojant el. paštą, naudotą registracijoje arba naudojant Google paskyrą. Naudojant Google paskyrą registracija nebūtina, tačiau vėliau galima nustatyti paskyros slaptažodį ir prisijungti prie sistemos naudojant el. paštą, taip pat lyg vartotojas būtų užregistruotas registracijos forma.



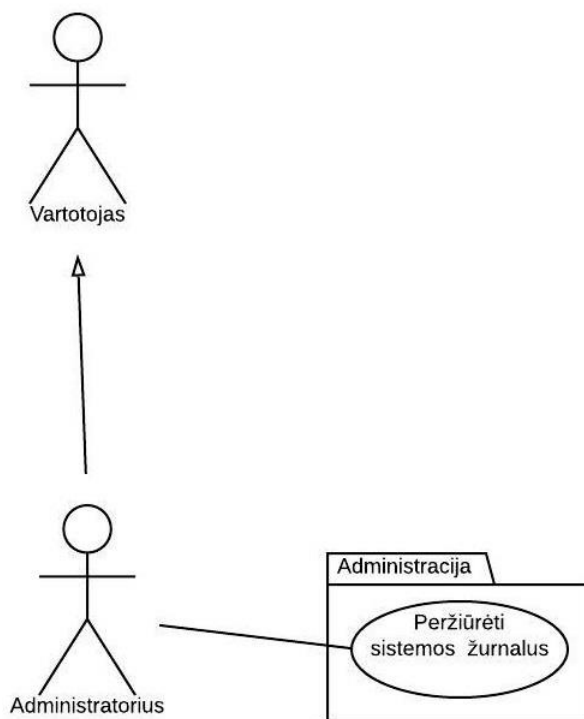
pav. 2.3 Projektų posistemės panaudojimo atvejų diagrama

Projektų posistemė apima visus projekto valdymo veiksmus. Prisijungęs vartotojas mato projekto laiko juostą ir joje pavaizduotas kategorijas, užduotis ir pan. Vartotojas gali turėti ar būti priskirtas keliems projektams ir gali pasirinkti kurio projekto laiko juostą nori matyti.

Į pačio projekto valdymą įeina jo sukūrimas ir meta-duomenų redagavimas, bei trynimasis, taip pat ir projekto objektų valdymas. Projektas susideda iš užduočių, kategorijų, etapų ir „epikų“. Užduotys turi savo nustatytą laiko terminą kada turėtų būti įvykdytos ir gali būti priskirtos vartotojui. Visos užduotys yra priskirtos tam tikrai kategorijai arba sub-kategorijai taip palaikant projektą organizuotą ir aiškų. Kategorijos taip pat gali būti priskirtos „epikams“ – tai yra kategorijų grupės padedančios projekte sukurti hierarchinę tvarką. Projektas gali turėti ir etapus – tam tikra data pažymėta laiko juostoje, pavyzdžiui, jei tai yra programinės įrangos kūrimo projektas, gali būti pažymėta data, kada sistema turi būti išleista ir prieinama vartotojams.

Visi šie projekto objektai gali būti sukuriama, redaguojami ir trunami vartotojo, jeigu tai yra jo projektas arba jam suteiktos redagavimo teisės. Jei vartotojas yra priskirtas prie šio projekto tik skaitymo režimu, jis gali tik peržiūrėti šio projekto laiko juostą, bei joje esančius objektus.

Vartotojai gali įsijungti pranešimus apie projekte vykstančius veiksmus, pavyzdžiui, įvykdytą šio vartotojo sukurtą užduotį. Šie pranešimai pristatomi net kai vartotojas šiuo metu nesinaudoja sistema, ir taip pat gali būti siunčiami į mobiliuosius įrenginius.



pav. 2.4 Administracijos posistemės panaudojimo atvejų diagrama

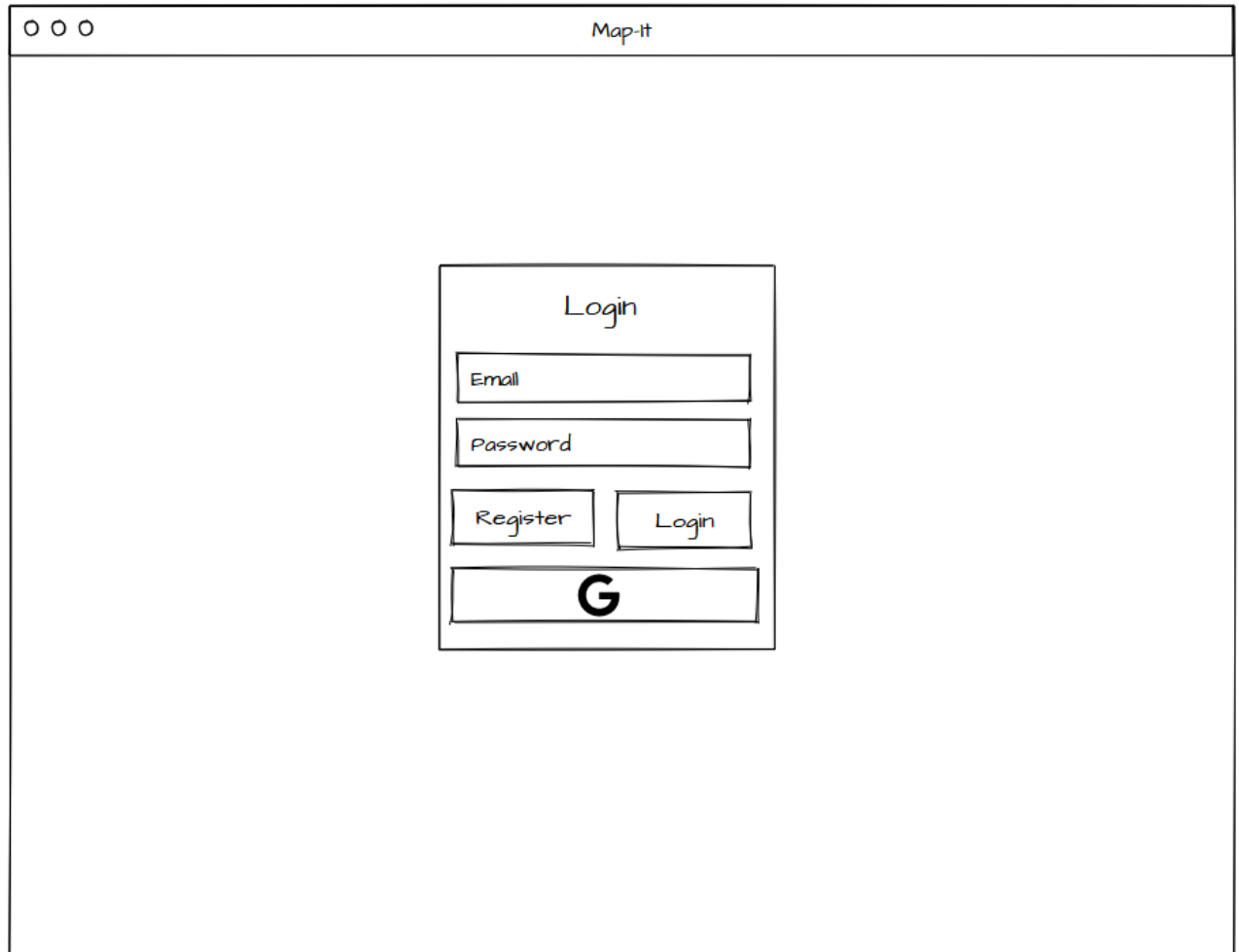
Administracijos posistemė neturi daug funkcionalumų. Ji skirta sistemos administratoriams surasti ir nustatyti sistemoje įvykusias klaidas. Tai padeda padaryti sistemos žurnalai (*angl. Logs*), juose talpinama visos sistemoje įvykusios klaidos.

Administratorius gali ne tik vykdyti administracines funkcijas, bet ir dirbti su projektais kaip paprastas vartotojas.



### 2.1.3. Vartotojo sąsajos specifikacija

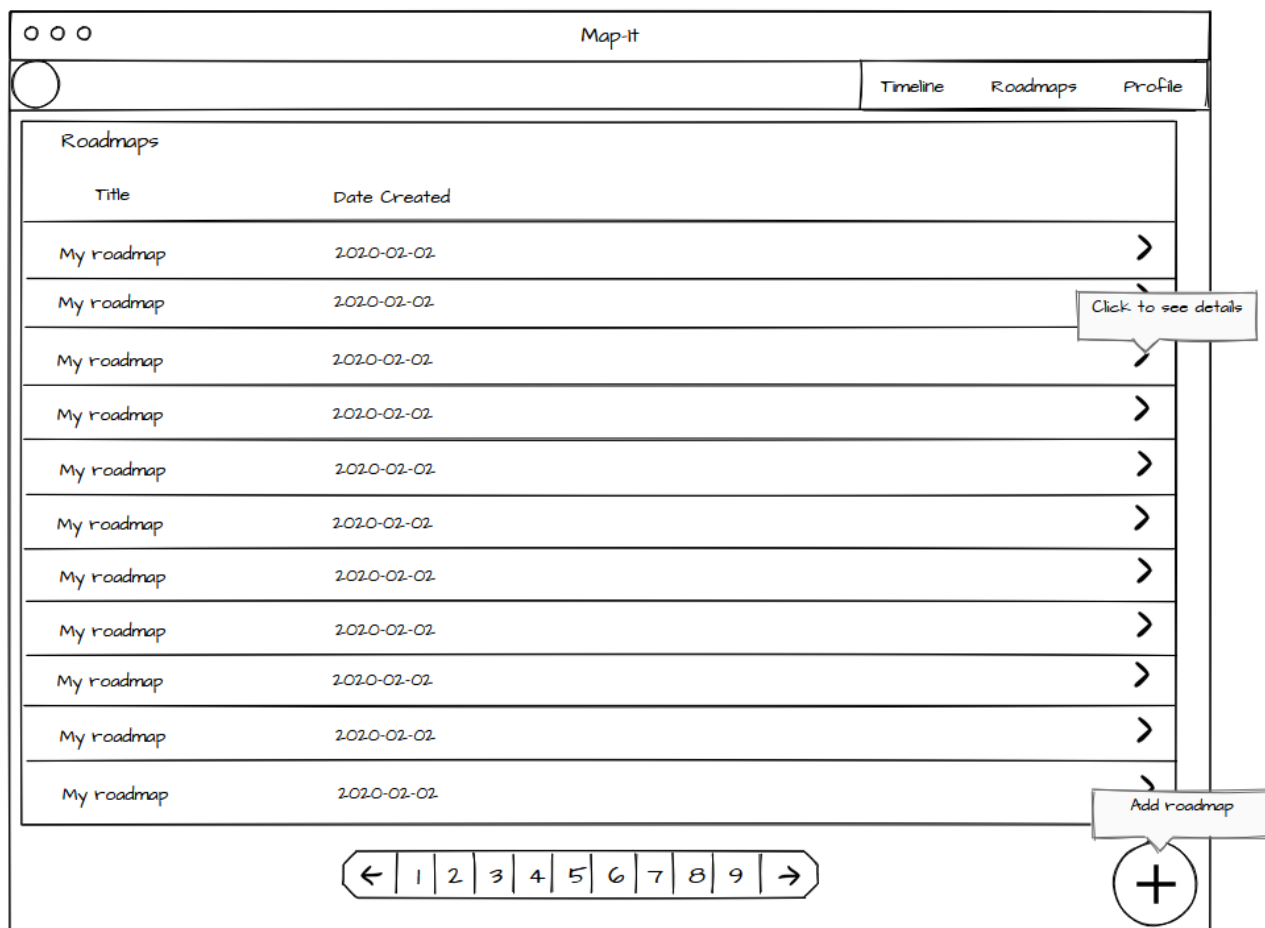
Vartotojo sąsajoms maketuoti buvo naudojamas *MockFlow* įrankis. Buvo sukurti eskizai tik unikaliems pagrindiniams langams, o kiti buvo daromi panašiu stiliumi be maketų. 2.5 – 2.8 paveikslėliuose pateikti pagrindinių langų maketai:



The image shows a wireframe of a login window. At the top, there is a title bar with three small circles on the left and the text 'Map-it' on the right. The main content area is a large rectangle. In the center of this area is a smaller rectangle representing the login form. Inside this form, the word 'Login' is centered at the top. Below it are two input fields: the first is labeled 'Email' and the second is labeled 'Password'. Below these fields are two buttons: 'Register' on the left and 'Login' on the right. At the bottom of the form is a wide button with a large, bold letter 'G' in the center.

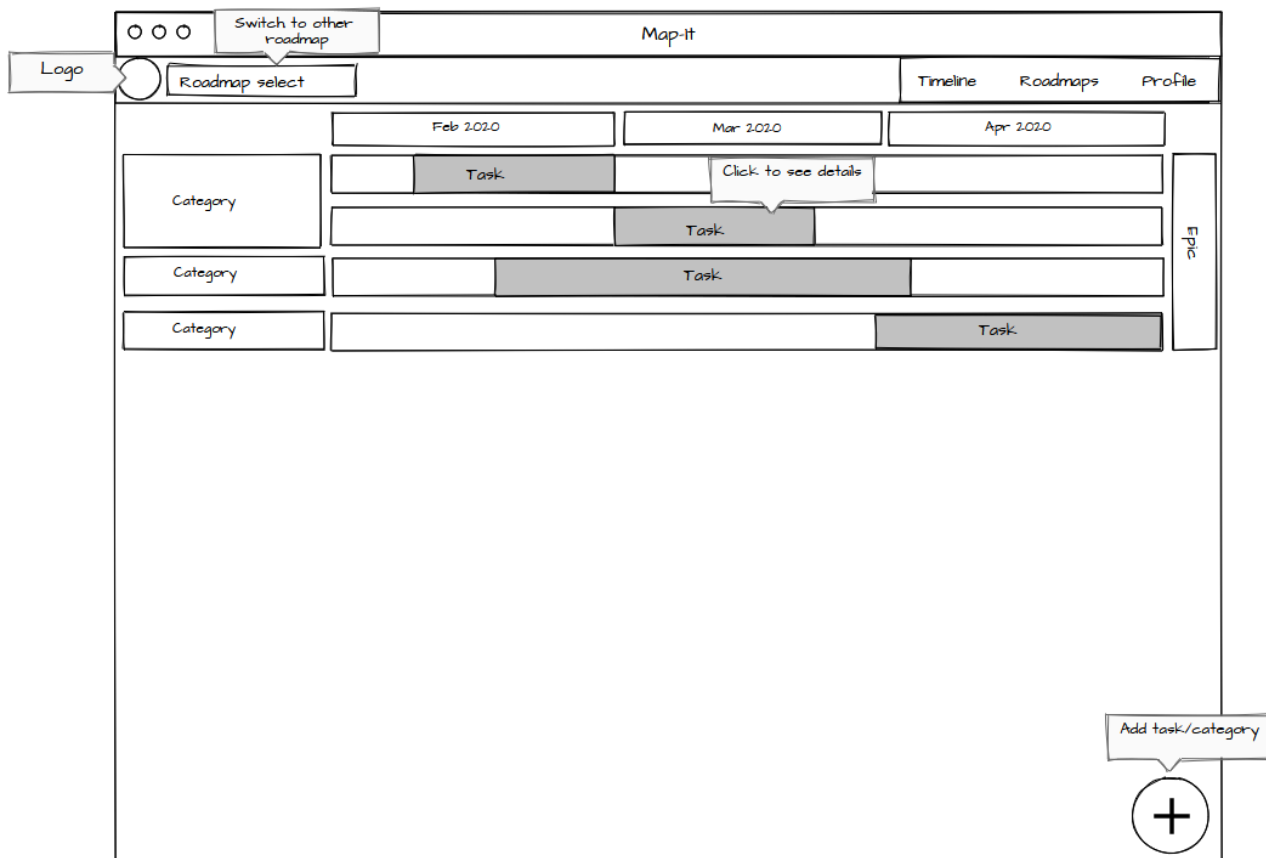
pav. 2.5 Prisijungimo langas

Prisijungimo lange yra nuorodos į registracijos langą ir prisijungimą naudojant Google paskyrą. Registracijos langas atrodo labai panašiai, bet turi daugiau formos laukų.



pav. 2.6 Sąrašo langas

Bendrinis sąrašo langas turi puslapių pasirinkimą, pridėjimo mygtuką, bei patį sąrašą objektų. Ant sąrašo eilutės galima paspausti ir pamatyti detalesnį objekto parašymą.



pav. 2.7 Projekto laiko juostos langas

Pagrindinis sistemos langas yra pavaizduotas 2.7 paveikslėlyje – projekto laiko juostos langas. Šiame lange matomi visi projekto objektai patogiai pavaizduoti laiko juostoje. Ant objektų galima paspausti, kad pamatyti jų meta duomenis ir kitas detales. Taip pat matomas projekto pasirinkimo laukas, kuriuo galima pakeisti projektą rodomą laiko juostoje. Lange yra pridėjimo mygtukas atidarantis iššokantį langą pavaizduotą 2.8 paveikslėlyje.

pav. 2.8 Projekto objektų pridėjimo iššokantis langas

Pridėjimo lange matomi skirtukai, ant kurių paspaudus pakeičiamas objekto tipas kurį bandoma pridėti. Panašus iššokantis langas naudojamas ir objektų redagavimui.

#### 2.1.4. Realizacijai keliami reikalavimai

Projekto realizacijai keliami šie reikalavimai:

- API metodų kodo padengimas testais turi būti bent 80%;
- Sistema turi būti konfigūruojama naudojant aplinkos kintamuosius - kode negali būti duomenų bazės prisijungimo duomenų ir panašių konfigūracijų;
- Sistema turi būti progresyvi saityno aplikacija (*PWA*) ir tam turi atitikti šiuos reikalavimus:
- Turi naudoti saugų (*HTTPS*) ryšio protokolą;
- Turi būti aprašytas W3C manifestas;
- Sistema turi veikti ir dingus interneto ryšiui;
- Vartotojo sąsaja turi teisingai veikti naudojant mobiliuosius įrenginius su mažesniais ekranais;
- Sistema turi taip pat veikti skirtingose naršyklėse;
- Kiekvienas sistemos langas turi turėti savo unikalų URL;

### 2.1.5. Techninė specifikacija

Kuriama sistema bus talpinama *Microsoft Azure Cloud* debesų platformoje. Sistemai pilnai veikti bus reikalingi šie resursai su nurodytais minimaliais parametrais:

- 1) *App Service* – virtualus serveris skirtas talpinti sistemos failus ir vykdyti kodą;
  - a) Testavimo aplinkai:
    - i) 100 *ACU*
    - ii) 1,75 GB atminties
  - b) Produkcini aplinkai:
    - i) 210 *ACU*
    - ii) 3,5 GB atminties
- 2) *Azure Database for PostgreSQL server* – duomenų bazės serveris skirtas laikyti sistemos duomenims;
  - a) Testavimo aplinkai:
    - i) 1 *vCore*
    - ii) 5 GB talpos
  - b) Produkcinei aplinkai:
    - i) 2 *vCore*
    - ii) 50 GB talpos
- 3) *Storage account* – failų saugykla skirta sistemos žurnalų (*angl. Logs*) ir duomenų bazės atsarginių kopijų saugojimui;
  - a) Testavimui ir produkcijai:
    - i) Tipas: *StorageV2 (general purpose v2)*
    - ii) Prieinamumo pakopa (*angl. Access tier*): *Standard/Cool*

## 2.2. Projektavimo metodai

### 2.2.1. Projektavimo valdymas ir eiga

Projektas buvo kuriamas naudojant iteracinį projektavimo modelį. Buvo atliktos iš viso septynios dviejų savaitių iteracijos. Į iteraciją įtraukiama funkcionalumų dizainas, programavimas, testavimas. 2.9 paveikslėlyje pavaizduota iteracinis modelis.



pav. 2.9 Iteracinio programinės įrangos kūrimo modelio iliustracija

Iteracijų tikslai išdėstyti tokiu eiliškumu:

1. Projekto infrastruktūra ir vartotojų prisijungimas
2. Projekto valdymas – užduočių, kategorijų kūrimas ir trynimasis
3. Laiko juostos atvaizdavimas
4. Administracinė dalis
5. Prisijungimas naudojant Google paskyrą
6. Etapai ir kategorijų grupės

### 2.2.2. Projektavimo technologija

Sistemos projektas kuriamas naudojant UML modeliavimo kalbą. UML diagramoms sukurti buvo naudojamas *LucidChart* ir *draw.io* projektavimo įrankiai.

### 2.2.3. Programavimo kalbos, derinimo, automatizavimo priemonės, operacinė sistema

Sistema naudoja *Node.js* aplinką kurią palaiko beveik visos modernios operacinės sistemos, tad buvo pasirinkta *Windows* operacinė sistema.

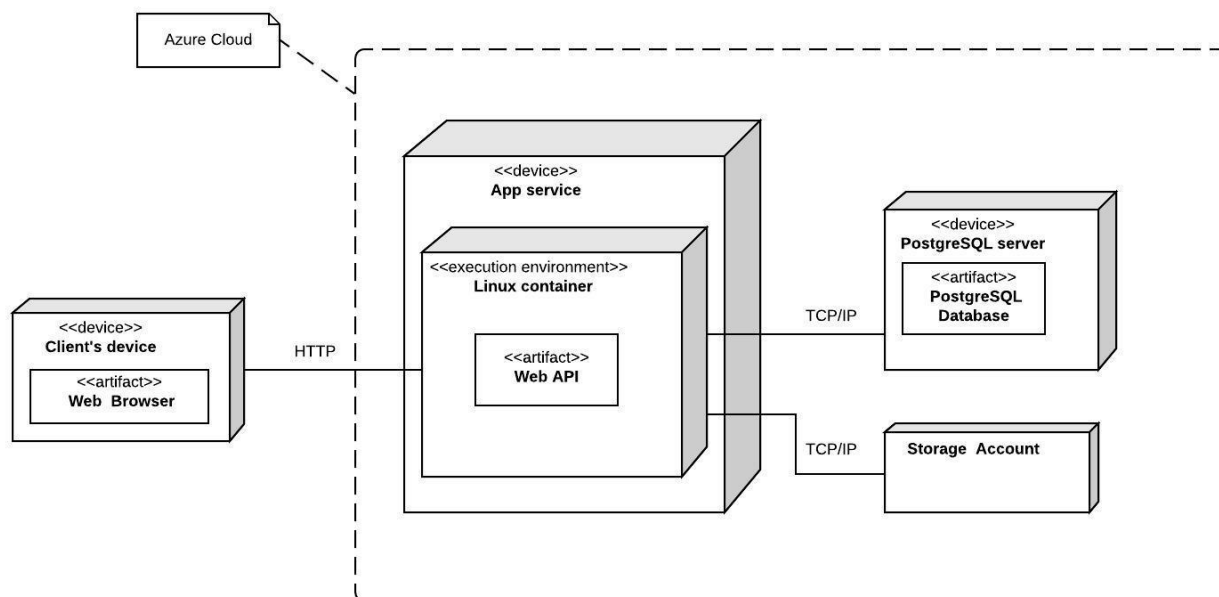
Sistemos kodo rašymui bei testavimui naudojama *Visual Studio Code* kodo redagavimo programa, su įvairiais papildiniais padedančiais derinimui, bei kodo kėlimui į *GitHub* kodo talpyklą. Lokali duomenų bazė (*PostgreSQL*) programavimui ir testavimui buvo sukurta *Docker* konteineryje. Duomenų bazės valdymui naudotas *PgAdmin* įrankis. Sistemos automatiniam testavimui naudojamas *Cypress* įrankis.

Sistema diegiama į testavimo aplinką *Azure Cloud* platformoje. Į šią aplinką ji diegiama naudojant *GitHub Actions* automatizavimo scenarijus. *GitHub Actions* taip pat buvo naudojami ir vykdyti automatinius testus prieš diegimą į testavimo aplinką.

## 2.3. Sistemos projektas

### 2.3.1. Statinis sistemos vaizdas

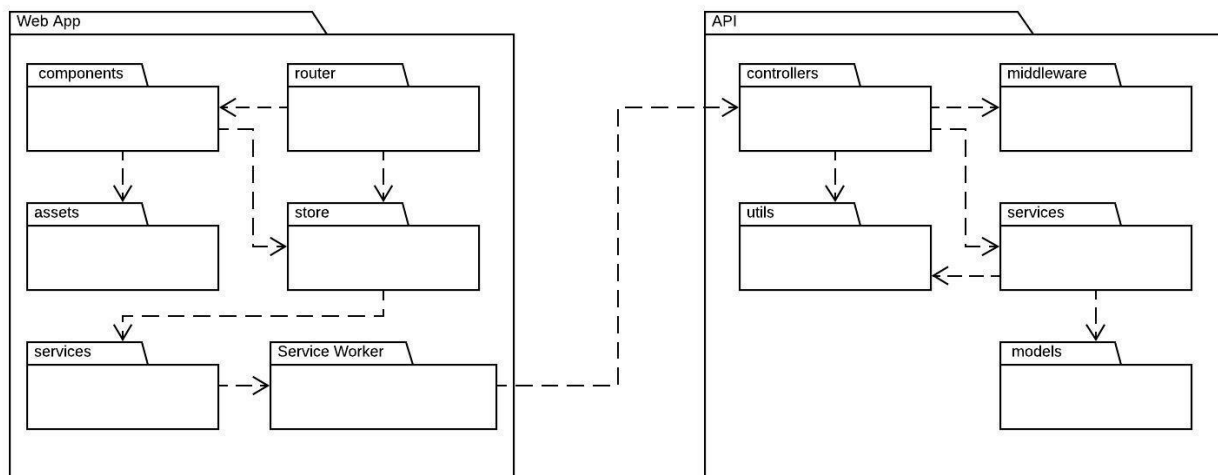
2.10 paveikslėlyje pavaizduota UML sistemos diegimo diagrama *Azure Cloud* platformoje - stambiausias sistemos išdėstymas.



pav. 2.10 UML sistemos diegimo diagrama

Diagrama yra pakankamai paprasta, nes visi sistemos komponentai diejami debesų platformoje, neapsisunkinant serverių diegimu ir priežiūra. Pagrindinis API ir vartotojo sąsaja yra *Node* aplikacija talpinama *App Service* sistemoje ir vykdoma *Docker* konteineryje su *Linux* operacinės sistemos pagrindu. API turi ryšį su duomenų baze ir failų talpykla.

Sistema turi du pagrindinius paketus – *Web* dalį ir *API* dalį. Smulkesnis sistemos vaizdas yra paketų diagramoje, matomoje 2.11 paveikslėlyje.



pav. 2.11 Sistemos UML paketų diagrama

Web dalis yra *Vue.js* aplikacija – vartotojo sąsaja. Ši aplikacija suskirstyta į šiuos paketus:

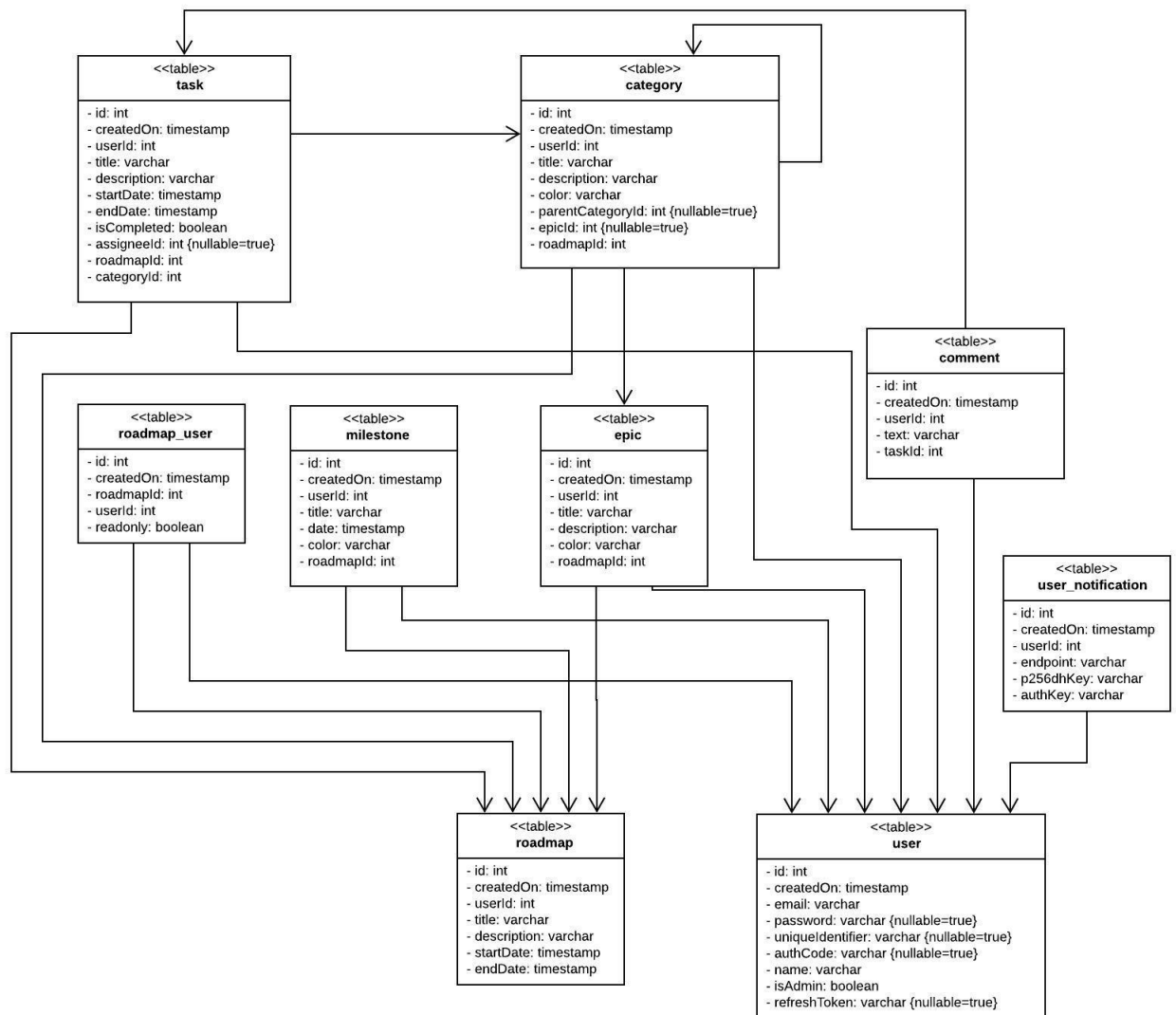
- *components* – *Vue* komponentai, naudojami atvaizduoti puslapius ir reaguoti į vartotojo veiksmus. Šis paketas ima ikonas, paveikslėlius ir panašius resursus iš *assets* paketo ir užkrauna duomenis kuriuos reikia atvaizduoti iš *store* paketo;
- *router* – virtualus maršrutizatorius pagal URL parenkantis koki komponentą rodyti;
- *assets* – paveikslėliai, ikonas, globalūs stiliai;
- *store* – *Vuex* duomenų saugykla, naudoja įvairius pagalbinius servisus, iš *services* paketo, duomenims iš API užkrauti;
- *services* – pagalbinės klasės ir metodai duomenims iš API gauti;
- *Service Worker* – servisas perimantis aplikacijos užklausas, saugoja gautus atsakus ir imituoja API kai dingsta interneto ryšys, taip palaikant aplikacijos veikimą ir be interneto.

API dalis yra *Node/Express.js* serveris tvarkantis projektų duomenis. Ši aplikacija susideda iš šių paketų:

- *controllers* – valdikliai parenkantys kokias funkcijas vykdyti pagal gautą užklausą;
- *middleware* – pagalbinės funkcijos vykdomos su kiekviena užklausa, pavyzdžiui, klaidų sugavimas ir įrašymas į žurnalus;
- *utils* – pagalbinės funkcijos naudojamos visoje aplikacijoje
- *services* – pagrindinės klasės su verslo logika. Daro veiksmus su duomenų baze naudojant klases iš *models* paketo;
- *models* – objektai naudojami duomenims iš duomenų bazės pasiekti. Beveik identiška struktūra kaip ir duomenų bazės modelio.

API naudoja *TypeORM* paketą kuris apjungia *models* pakete esančias klases su atitinkamomis lentelėmis duomenų bazėje. Šis paketas suteikia galimybę dirbti su duomenų bazėje esančiais duomenimis nerašant *SQL* užklausų kiekvieną kartą norint kažką pakeisti ar gauti (6). Pilnas duomenų bazės modelis pateiktas 2.12 paveikslėlyje.



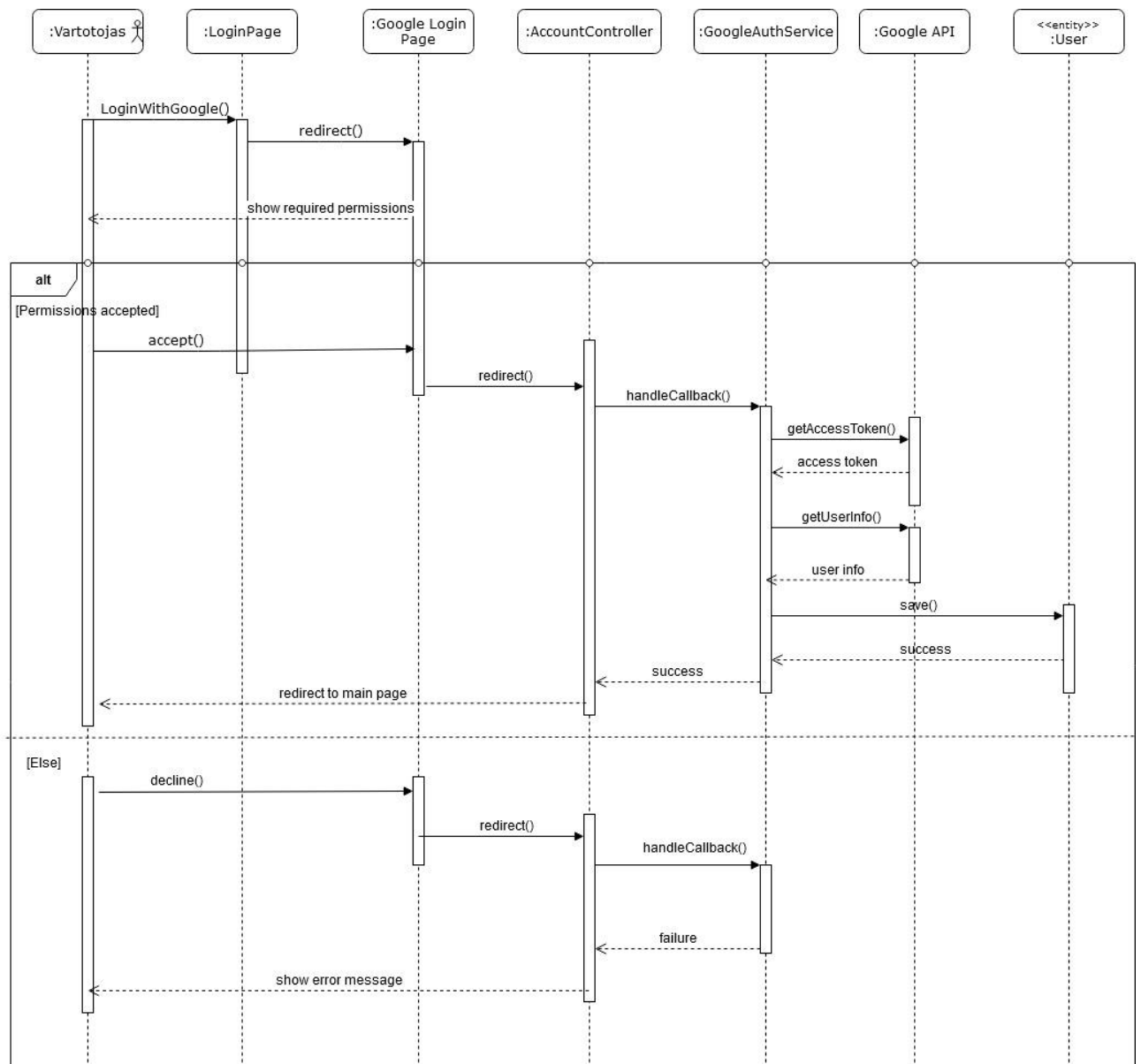


pav. 2.12 Duomenų bazės UML diagrama

Iš diagramos matyti, kad visi objektai susiję su projektu (*roadmap*) turi ryšį su vartotojo lentele (*user*), tai padeda atfiltruoti visus objektus priklausančiam prisijungusiam vartotojui. Projektas turi tiesioginį ryšį su vartotoju, kai šis vartotojas sukūrė šį projektą, taip pat prie projekto galima priskirti daugiau vartotojų naudojant *roadmap\_user* tarpinę lentelę. Kategorija (*category*) turi ryšį į save, taip leidžiant sukurti tėvines kategorijas ir sudaryti hierarchiją. Visi kiti ryšiai ir laukai atvaizduoti diagramoje – 2.12 paveikslėlyje.

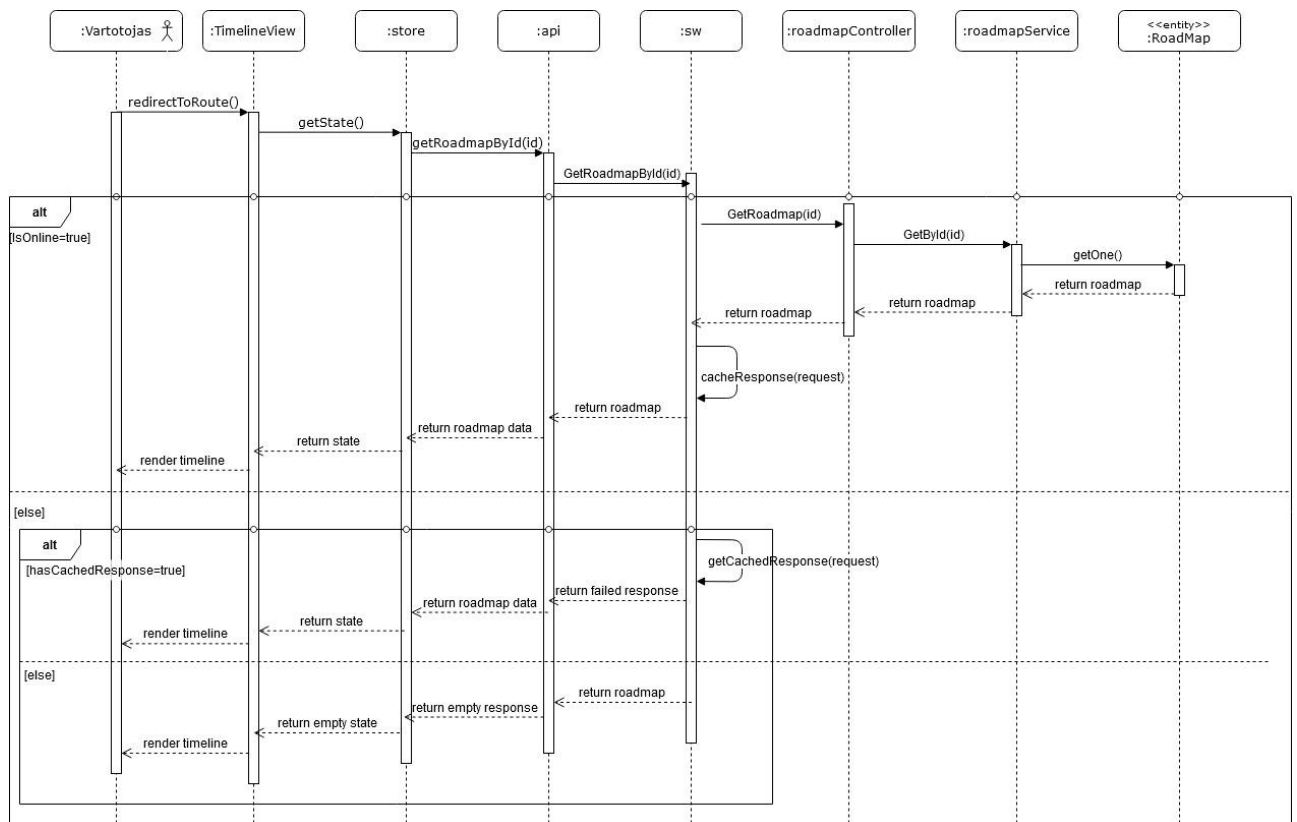
### 2.3.2. Dinaminis sistemos vaizdas

Dinaminis sistemos vaizdas atvaizduotas sekų diagramomis, 2.13 – 2.16 paveikslėliuose pavaizduotos svarbiausių sistemos dalių sekų diagramos. Šios diagramos pavaizduoja sąveika tarp sistemos klasių bei išorinių sąsajų, tokių kaip Google API ir pan.



pav. 2.13 Prisijungimo naudojant Google paskyrą sekų UML sekų diagrama

Viena iš svarbiausių sistemos dalių yra prisijungimas naudojant Google paskyrą. Šis procesas prasideda vartotojui paspaudus Google ikoną prisijungimo ekrane. Paspaudus šį mygtuką vartotojas nukreipiamas į Google prisijungimo langą, kur jis turi pasirinkti Google paskyrą, bei sutikti su prašomais leidimais. Po to vartotojas nukreipiamas į sistemos API su informacija iš Google, šiuos duomenis priima valdiklis ir perduoda *googleAuthService* klasei. Jeigu vartotojas nesutiko su prašomais leidimais jam grąžinama klaidos žinutė. Jeigu vartotojas sėkmingai suteikė prieigą, sistemos API kreipiasi į Google API, dėl prieigos žetono ir vartotojo informacijos. Pagal gautą informaciją identifikuojamas vartotojas, į Web dalį grąžinamas sugeneruotas prieigos žetonas ir vartotojas nukreipiamas į pagrindinį įrankio langą. Jeigu vartotojas prieš tai nebuvo užsiregistravęs, jam sukuriamą paskyrą su informacija gauta iš Google API ir vartotojas prijungiamas.



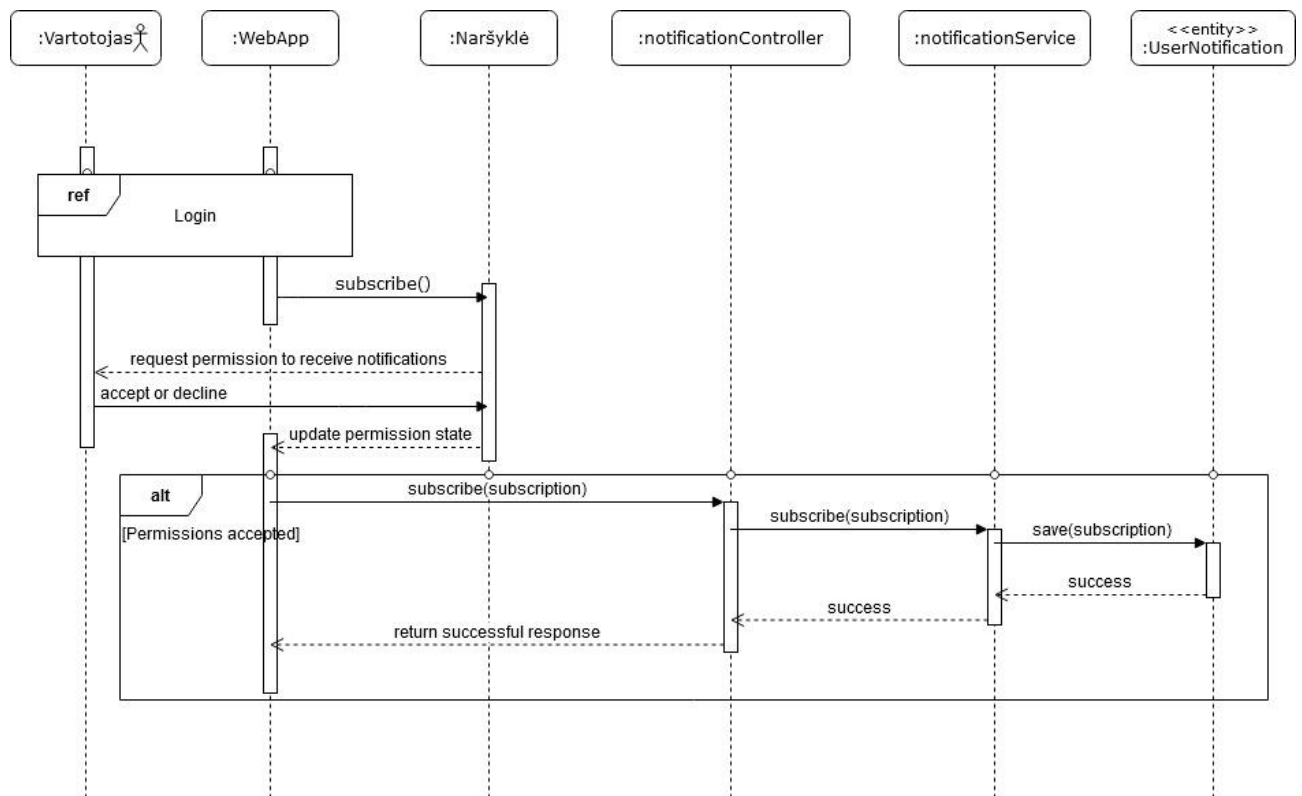
pav. 2.14 Pagrindinio lango užkrovimo UML sekų diagrama

Vartotojui prisijungus, jis nukreipiamas į pagrindinį puslapį, kur rodoma pasirinkto projekto laiko juosta. Tam, kad atvaizduoti laiko juostą, *TimelineView* komponentas bando gauti aplikacijos būseną iš būsenos valdymo komponento (*store*). Jeigu puslapis užkraunamas pirmą kartą, būseną yra tuščia ir valdymo komponentas kviečia pagalbinį servisą *api* gauti projekto informaciją iš sistemos API. Visas užklausas perima SW komponentas. SW daro tokią pačią užklausą ir patikrina ar ji įvykdyta sėkmingai, jeigu taip, grąžinamas gautas atsakas atgal į *api* servisą. Jeigu užklausos įvykdyti nepavyko, ieškoma tokios pačios užklausos atsako išsaugoto atmintyje. Radus atsaką jis grąžinamas atgal į *api* servisą, taip imituodamas sėkmingą API užklausą. Gali atsirasti tokia situacija, kad atmintyje nėra išsaugoto atsako į šią užklausą, tuo atveju, grąžinamas tuščias atsakas ir vartotojui parodoma žinutė, kad nepavyko gauti projekto duomenų. Gavus atsaką iš *api* serviso, būsenos valdiklis išsaugo gautą informaciją apie projektą ir perduoda būseną komponentui.

Panašiu principu veikia visos užklausos į sistemos API. Kai yra interneto ryšys, atsakas išsaugomas atmintyje, jeigu šiai užklausiai atsakas jau yra, jis atnaujinamas. Kai interneto ryšio nėra, bandoma surasti atsaką atmintyje ir grąžinti aplikacijai, kaip sėkmingai įvykdytą užklausą.

SW taip pat atmintyje išsaugo ir statinius saityno failus – HTML, JavaScript bei CSS failus. Tai suteikia galimybę bet kada išjungti ir įjungti aplikaciją be interneto ryšio.

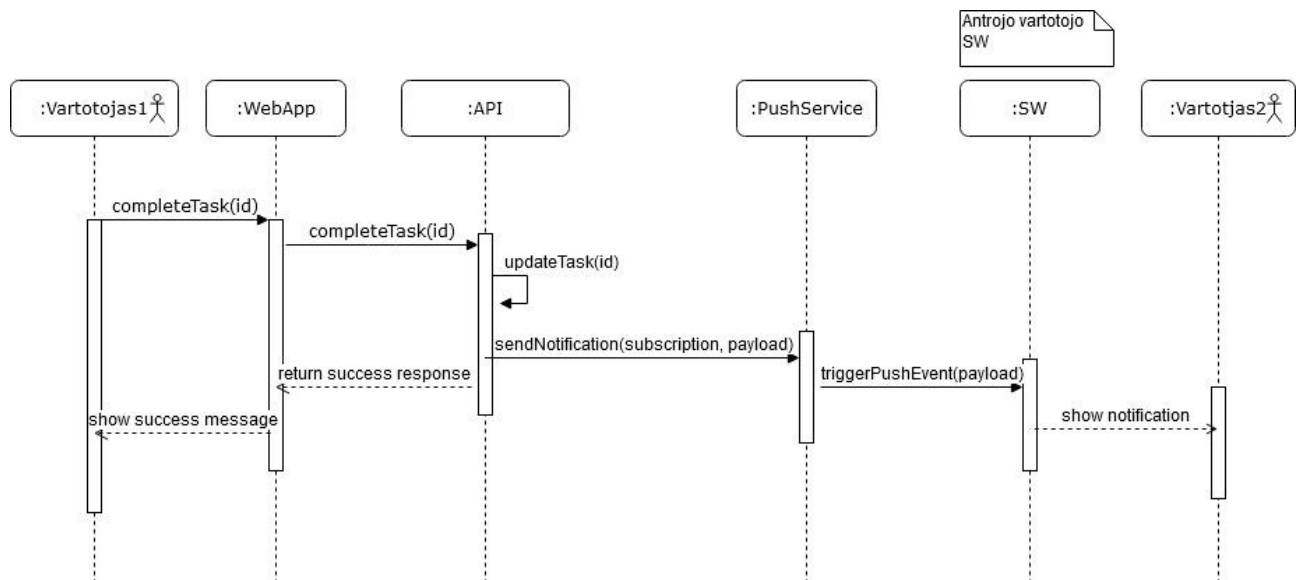
SW suteikia galimybę vartotojams gauti momentinius pranešimus, net ir nebūnant įjungus įrankio. Pranešimai gali būti pristatomi ir į mobiliuosius įrenginius, ir į stacionarius kompiuterius. Kad vartotojas gautų pranešimus į norimą įrenginį, iš pradžių reikia jį užregistruoti. Registracijos sekų diagrama pavaizduota 2.15 paveikslėlyje.



pav. 2.15 Įrenginio registravimo momentiniams pranešimams UML sekų diagrama

Įrenginio registracija vyksta vartotojui prisijungus arba vartotojui išijungus įrankį jau esant prisijungus. WEB aplikacija bando gauti registraciją iš naršyklės. Jeigu vartotojas dar nėra sutikęs gauti pranešimus, jam bus parodomas pranešimas, prašantis sutikimo gauti pranešimus. Vartotojui sutikus arba nesutikus, naršyklė atnaujiną leidimų būseną. Jeigu vartotojas nesutiko gauti pranešimų, seka baigiasi ir tolesni veiksmai nėra vykdomi. Tačiau jeigu vartotojas nori gauti pranešimus, aplikacija siunčia iš naršyklės gautą registracijos objektą į API. API susieja gautą registraciją su prisijungusiu vartotoju ir išsaugo duomenų bazėje tolesniam naudojimui.

Pranešimai vartotojui siunčiami kai kitas vartotojas įvykdo šio vartotojo sukurtą užduotį, bei kai vartotojas priskiriamas projektui. 2.16 paveikslėlyje pavaizduota pranešimo gavimo po užduoties atlikimo sekų diagrama.



pav. 2.16 Pranešimo gavimo UML sekų diagrama

Šioje diagramoje dalyvauja du vartotojai – pirmasis ir antrasis. Antrasis vartotojas yra užduoties autorius. Pirmajam vartotojui įvykdžius užduotį, siunčiama užklausa į API. Duomenų bazėje surandama ir atnaujinama užduoties informacijos. Pagal užduoties autorių surandama to vartotojo registracija išsaugota anksčiau. Jeigu registracija neegzistuoja, vartotojas nenori gauti pranešimų ir jis nebus siunčiamas. API naudojant *webpush* paketą, išsiunčia pranešimą pagal rastą registraciją. Pranešimų servisas (*Push Service*) nustatomas iš registracijoje esančių duomenų, dažniausiai priklauso nuo naudojamos naršyklės – „*Google Chrome*“ naudoja „*FCM*“, „*Safari*“ naudoja „*APNs*“ ir pan (7). Pranešimų servisas pagal gautą pranešimo ir registracijos informaciją, sužadina antrojo vartotojo SW ir šis parodo gautą pranešimą vartotojui. Tokiu principu veikia ir kitokių pranešimų siuntimas. Tai leidžia ateityje nesunkiai pridėti daugiau įvairių pranešimų.

### 3. Testavimas

Šiame skyriuje pateikiamas testavimo planas, testavimo kriterijai, bei skirtingų testavimo būdų veikimas ir aprašymai.

#### 3.1. Testavimo planas

Sistemos testavimas atliekamas šiais būdais:

1. Statinė kodo analizė;
2. Automatinis integracinis ir vienetų testavimas;
3. Automatinis pilnos sistemos testavimas;
4. Rankinis testavimas.

Statinė kodo analizė buvo atliekama iš kart rašant programinį kodą, naudojant *TSLint* ir *ESLint* įrankius. Šie įrankiai patogiai integruojasi į *Visual Studio Code* kodo redaktorių ir iš kart rašant pažymi klaidas, tokias kaip nenaudojami kintamieji, funkcijos ir panašios klaidos, įskaitant ir kodo stiliaus palaikymą.

Automatiniai testai vykdomi naudojant *Github Actions*, kiekvieną kartą prieš suliejant pastovaus vystymo kodo šaką su pagrindine šaka, atsiradus neveikiančių testų suliejimas atšaukiamas.

Rankinis testavimas vyksta po kiekvienos iteracijos patikrinant naujai sudiegtas funkcijas testavimo aplinkoje.

#### 3.2. Testavimo kriterijai

Sistema testuojama taikant šiuos kriterijus:

- Statinė kodo analizė yra sėkminga ir nerodo jokių klaidų (įskaitant ir kodo stiliaus nenuoseklumus);
- Visi automatiniai testai yra sėkmingi;
- API padengimas automatiniais testais siekia bent 80%;
- Pilnos sistemos automatiniais testais padengti bent pagrindiniai įrankio scenarijai;
- Naudojant API negalima gauti objektų kurių neturi matyti vartotojas;
- Sistema turi veikti taip, kaip apibrėžta specifikacijoje;
- Neturi būti neapdorotų sistemos klaidų, t. y. API negali grąžinti atsako su kodu 500.








#### 3.3. Komponentų testavimas

Komponentų testais padengta, palyginus, maža sistemos dalis, didžiausia dalis padegta integraciniais testais aprašytais 3.4 skyriuje. Šiais testais buvo padengti API funkcionalumai naudojantys išorinius servisus, pavyzdžiui, žurnalų rašymo servisas. Taip pat šiais testais padengta dalis WEB aplikacijos funkcionalumų, pavyzdžiui, laiko projekto laiko juostos formatavimo pagalbinės funkcijos.

API testuoti naudojamos *supertest* ir *mocha* bibliotekos, o WEB aplikacijai testuoti naudojama *jest* biblioteka.

#### 3.4. Integracinis testavimas

Integraciniai testai padengia didžiąją dalį API ir siekia net 92% - žr. 3.1 paveikslėlį.

File		Statements	Branches	Functions	Lines
src		92.11%	35/38	89.29%	25/28
src/controllers		89.82%	150/167	50%	2/4
src/middleware		66.67%	6/9	50%	1/2
src/models		99.51%	202/203	62.82%	49/78
src/services		93.09%	350/376	84.73%	111/131
src/services/util		79.66%	47/59	72.22%	13/18
src/utils		87.88%	58/66	76%	19/25

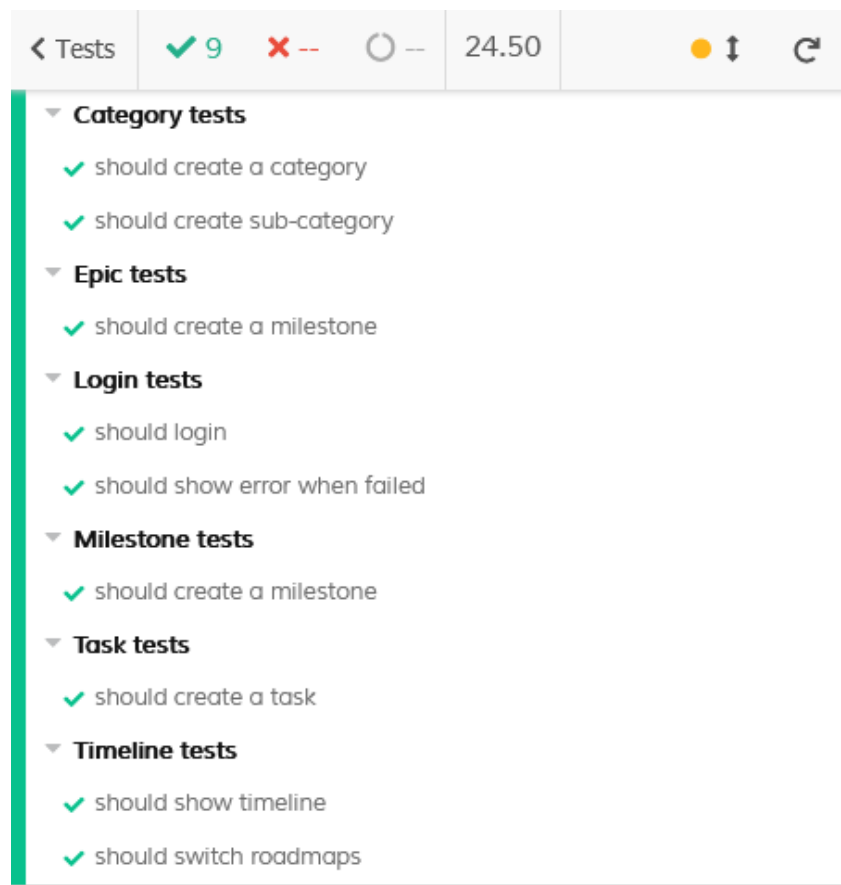
pav. 3.1 API padengimas integraciniais testais

Šie testai vykdomi naudojant *supertest* biblioteką, kuri leidžia sukurti netikrą serverį ir daryti užklausas į jį. Integraciniams testams taip pat naudojama testavimui skirta duomenų bazė kurioje kuriami netikri duomenys. Kiekvienas testas susikuria norimus duomenis duomenų bazėje, siunčia užklausą netikram serveriui ir testuoja ar gavo norimą atsaką. Tokio tipo automatiniai testai padengia visus API sluoksnius, nuo valdiklio iki duomenų bazės, todėl buvo pasirinkta būtent tokiais testais padengti didžiąją dalį sistemos.

### 3.5. Vartotojo sąsajos testavimas

Vartotojo sąsaja testuojama dviem būdais – rankiniu ir automatinio. Automatiniais testais padengta tik pagrindiniai scenarijai (žr. 3.2 paveikslėlį), todėl pro juos gali praslysti klaidos ir reikalingas rankinis testavimas. Automatiniais testais padengti šių pagrindinių funkcijų scenarijai:

- Kategorijų kūrimas
- Kategorijų grupių kūrimas
- Vartotojo prisijungimas prie sistemos
- Etapų kūrimas
- Užduočių kūrimas
- Laiko juostos atvaizdavimas
- Projektų keitimas



pav. 3.2 Vartotojo sąsajos automatiniai testai

Automatiniai testai rašomi ir vykdomi naudojant *cypress* karkasą. Šis karkasas leidžia nesudėtingai rašyti norimus vykdyti scenarijus *JavaScript* kalba. Automatiniai vartotojo sąsajos testai yra taip pat programuotojo atsakomybė ir turi būti aprašyti karu pagrindiniai scenarijai kartu su funkcionalumu. Šie testai sukuria testavimui paruoštus duomenis naudojant užklausą į API. Sukūrus duomenis, testai atidaro saityno programą ir vykdo aprašytus scenarijus. Testai parašyti *cypress* karkasu turi galimybę veikti ir be vartotojo įsikišimo, todėl buvo galima juos paleisti ir vykdyti automatiškai, naudojant *Github Actions*. Dėl šios integracijos automatiniai testai nesunkiai pastebi vieno ar kito funkcionalumo regresiją aktyvaus vystymo metu.

Rankinis vartotojo sąsajos testavimas vyksta po kiekvienos iteracijos. Tikrinami visi naujai pridėti funkcionalumai per tą iteraciją. Testavimas vykdomas naudojant įrankį iš vartotojo bei administratoriaus perspektyvos. Testuotojas naudoja įrankio naujomis funkcijomis, išbando kraštutinius scenarijus, bei klaidingų duomenų apdorojimą.

Žemiau lentelėse patiekti pagrindiniai testavimo scenarijai. Lentelėse aprašyta veiksmų seka, reikiami duomenys ir norimas rezultatas. Šiems testams turi būti naudojama aplinka su paruoštais testavimo duomenimis. Šiuos duomenis galima paruošti padarant „*GET /test-data/seed*“ užklausą ne produkcinėje aplinkoje.

**lentelė 2** Vartotojo prisijungimo prie sistemos testavimas

Scenarijus	Vartotojo prisijungimas
Veiksmų seka	Atidaroma aplikacija



	Suvedamas el. paštas ir slaptažodis Spaudžiamas mygtukas „Login“
<b>Duomenys</b>	„Email“: e2e-test@email.com „Password“: E2EsecurePassword
<b>Norimas rezultatas</b>	Vartotojas sėkmingai prisijungia prie sistemos ir yra nukreipiamas į projekto laiko juostos puslapį.

**lentelė 3** Projekto laiko juostos vaizdavimo testavimas

<b>Scenarijus</b>	Projekto rodymas
<b>Veiksmų seka</b>	Prisijungti su e2e-test@email.com vartotoju
<b>Duomenys</b>	-
<b>Norimas rezultatas</b>	Rodomas „E2E Roadmap1“ projektas. Projektas turi šiuos objektus: „E2E Task1“ užduotį; „E2E Task2“ įvykdytą užduotį; „E2E Category1“ kategoriją; „E2E Sub-Category1“ sub-kategoriją; „E2E Epic1“ kategorijų grupę; „E2E Milestone1“ etapą.

**lentelė 4** Projekto kategorijos kūrimo testavimas

<b>Scenarijus</b>	Kategorijos kūrimas
<b>Veiksmų seka</b>	Prisijungti su e2e-test@email.com vartotoju Paspausti objektų kūrimo mygtuką Pasirinkti „Category“ skirtuką Užpildyti formą nurodytais duomenimis Spausti mygtuką „Save“
<b>Duomenys</b>	„Title“: test category „Description“: this is a test description
<b>Norimas rezultatas</b>	Pasirinktame projekte sukuriamą naują kategoriją su nurodytais duomenimis

**lentelė 5** Projekto kategorijų grupės kūrimo testavimas

<b>Scenarijus</b>	Kategorijų grupės kūrimas
<b>Veiksmų seka</b>	Prisijungti su e2e-test@email.com vartotoju Paspausti objektų kūrimo mygtuką Pasirinkti „Epic“ skirtuką Užpildyti formą nurodytais duomenimis Spausti mygtuką „Save“
<b>Duomenys</b>	„Title“: test epic „Description“: this is a test description „Categories“: pasirenkama pirma kategorija iš pateikto sąrašo

<b>Norimas rezultatas</b>	Pasirinktame projekte sukurama nauja kategorijų grupė su nurodytais duomenimis ir apimanti tik pasirinktas kategorijas.
---------------------------	---

**lentelė 6** Projekto etapų kūrimas

Scenarijus	Etapų kūrimas
<b>Veiksmų seka</b>	Prisijungti su e2e-test@email.com vartotoju Paspusti objektų kūrimo mygtuką Pasirinkti „Milestone“ skirtuką Užpildyti formą nurodytais duomenimis Spausti mygtuką „Save“
<b>Duomenys</b>	„Title“: <i>test milestone</i> „Date“: pasirenkama paskutinė šio mėnesio diena
<b>Norimas rezultatas</b>	Sukuriamas etapas, pažymėtas paskutinę šio mėnesio dieną.

**lentelė 7** Projekto užduočių kūrimas

Scenarijus	Užduočių kūrimas
<b>Veiksmų seka</b>	Prisijungti su e2e-test@email.com vartotoju Paspusti objektų kūrimo mygtuką Pasirinkti „Task“ skirtuką Užpildyti formą nurodytais duomenimis Spausti mygtuką „Save“
<b>Duomenys</b>	„Title“: <i>test task</i> „Description“: <i>this is a test description</i> „Category“: pasirenkama pirma kategorija iš sąrašo „Start date“: nekeičiama, paliekama šiandienos data „End date“: pasirenkama paskutinė mėnesio diena
<b>Norimas rezultatas</b>	Sukuriama užduotis su visais nurodytais duomenimis. Laiko juostoje atvaizduota, nuo šiandienos datos iki einamo mėnesio galo.

Atlikus nurodytus testus bei panaudojus automatinius testus, nebuvo rasta jokių klaidų, visi scenarijai grąžino norimus rezultatus.

## **4. Dokumentacija naudotojui**

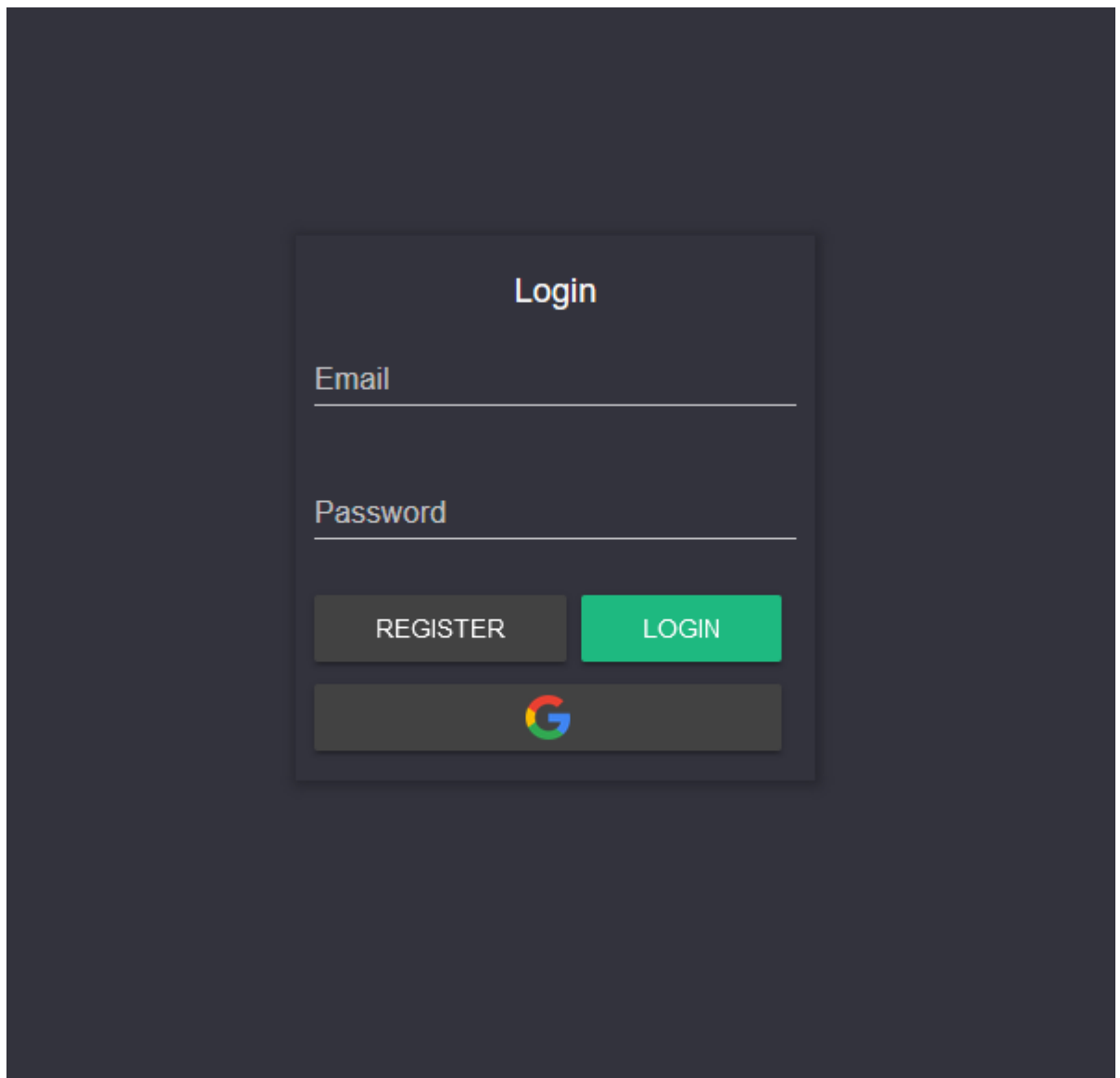
### **4.1. Apibendrintas sistemos galimybių aprašymas**

Realizuota sistema yra progresyvi saityno programa, leidžianti vartotojui valdyti savo projektus bei bendradarbiauti su kitais žmonėmis, pasiskirstyti darbus. Ši saityno programa yra palaikoma ne tik naršyklėse, bet ir daugumoje mobiliųjų ir stacionarių įrenginių, kaip atskira programa. Šis sistemos požymis leidžia naudotis įrankiu be interneto ryšio, bei gauti momentinius pranešimus tiesiai į savo įrenginį, net jo nenaudojant.

### **4.2. Vartotojo vadovas**

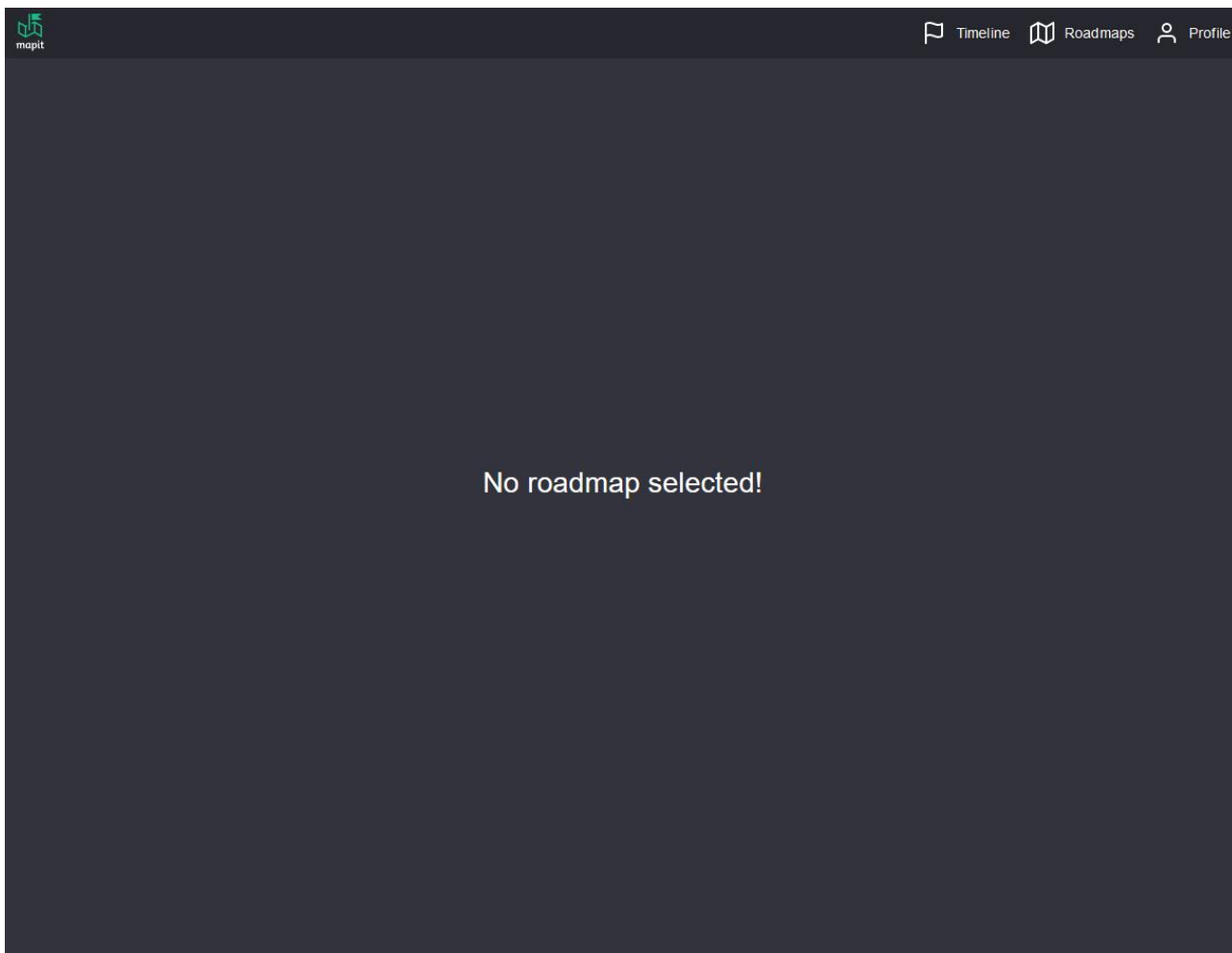
Žemiau pateikiamas pilnas scenarijus kaip pradėti naudotis įrankiu neturint jokios patirties. Scenarijus apima vartotojo prisijungimą, projektų kūrimą, projekto objektų (kategorijų, užduočių) kūrimą bei redagavimą. Taip pat ši dokumentacija parodys kaip prie to pačio projekto dirbti keliems žmonėms.

Tik atsidarius pagrindinį sistemos puslapį, rodomas vartotojo prisijungimo langas (4.1 paveikslėlis). Šiame lange reikia suvesti vartotojo prisijungimo duomenis arba spausti mygtuką „*Register*“ jeigu vartotojas dar nėra registruotas sistemoje. Taip pat galima prisijungti naudojant Google paskyrą, paspaudus ant mygtuko su Google ikona, šiuo atveju registruotis nereikia, net jei ir vartotojas naudoja sistemą pirmą kartą, paskyra bus sukurta naudojant duomenis gautus iš Google paskyros.



pav. 4.1 Vartotojo prisijungimo langas

Prisijungus naujam vartotojui, jis neturi jokių jam priskirtų, bei nuosavų projektų (4.2 paveikslėlis). Sukurti naują projektą galima paspaudus ant „Roadmaps“ navigacijos mygtuko ir tada spustelėjus žalią pliuso ikoną. Atidaromas iššokantis langas pavaizduotas 4.3 paveikslėlyje.

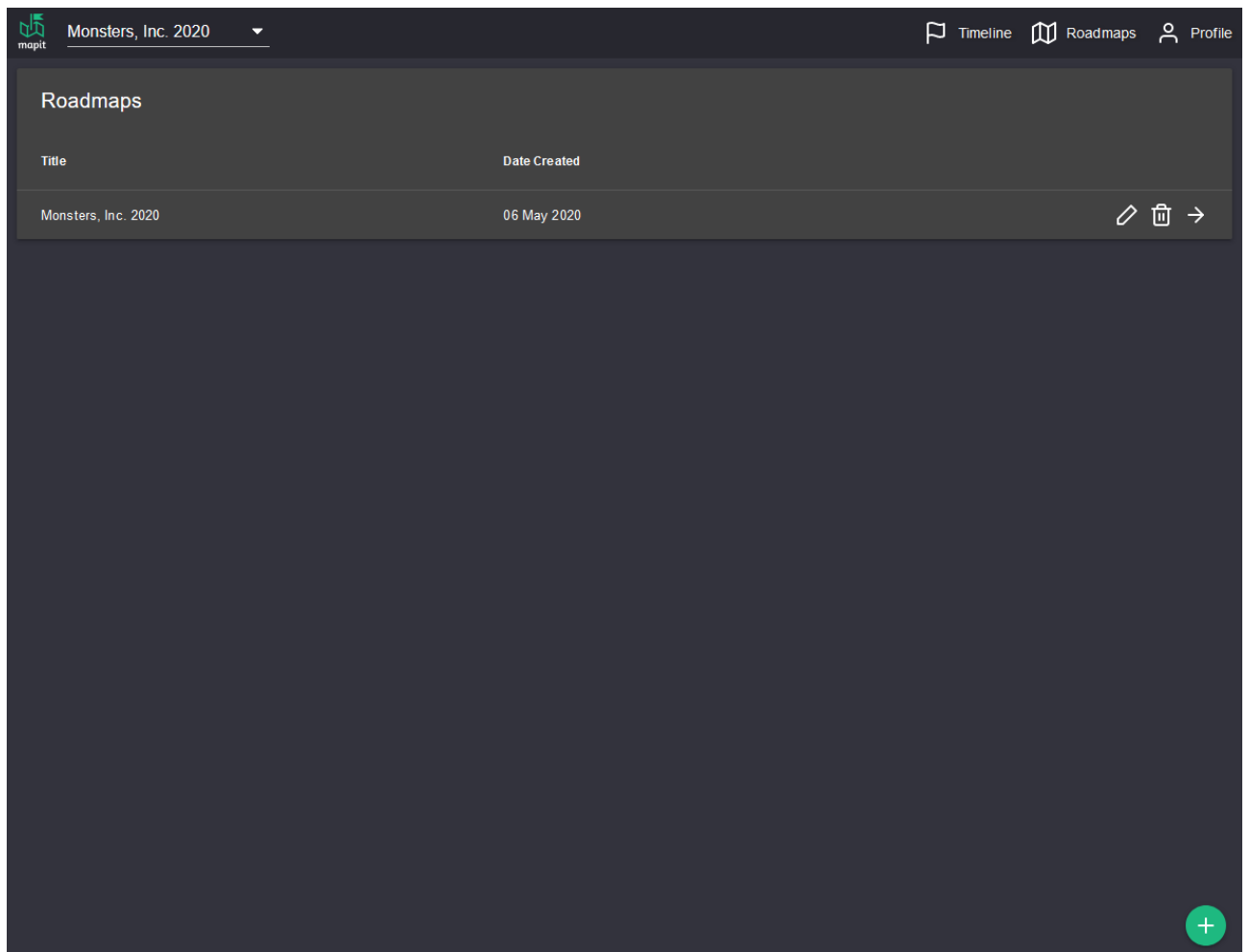


pav. 4.2 Pagrindinis sistemos langas, neturint jokių projektų

The image shows a dark-themed user interface for creating a new roadmap. At the top is a green header bar with the text "Create new roadmap". Below this is a form with several fields: a "Title" field with a single-line text input; a "Description" field with a larger multi-line text input; a "Start date" field containing a calendar icon, the date "2020-05-06", and a close button (X); and an "End date" field containing a calendar icon, the date "2020-05-06", and a close button (X). At the bottom right of the form are two buttons: a red "CANCEL" button and a green "SAVE" button.

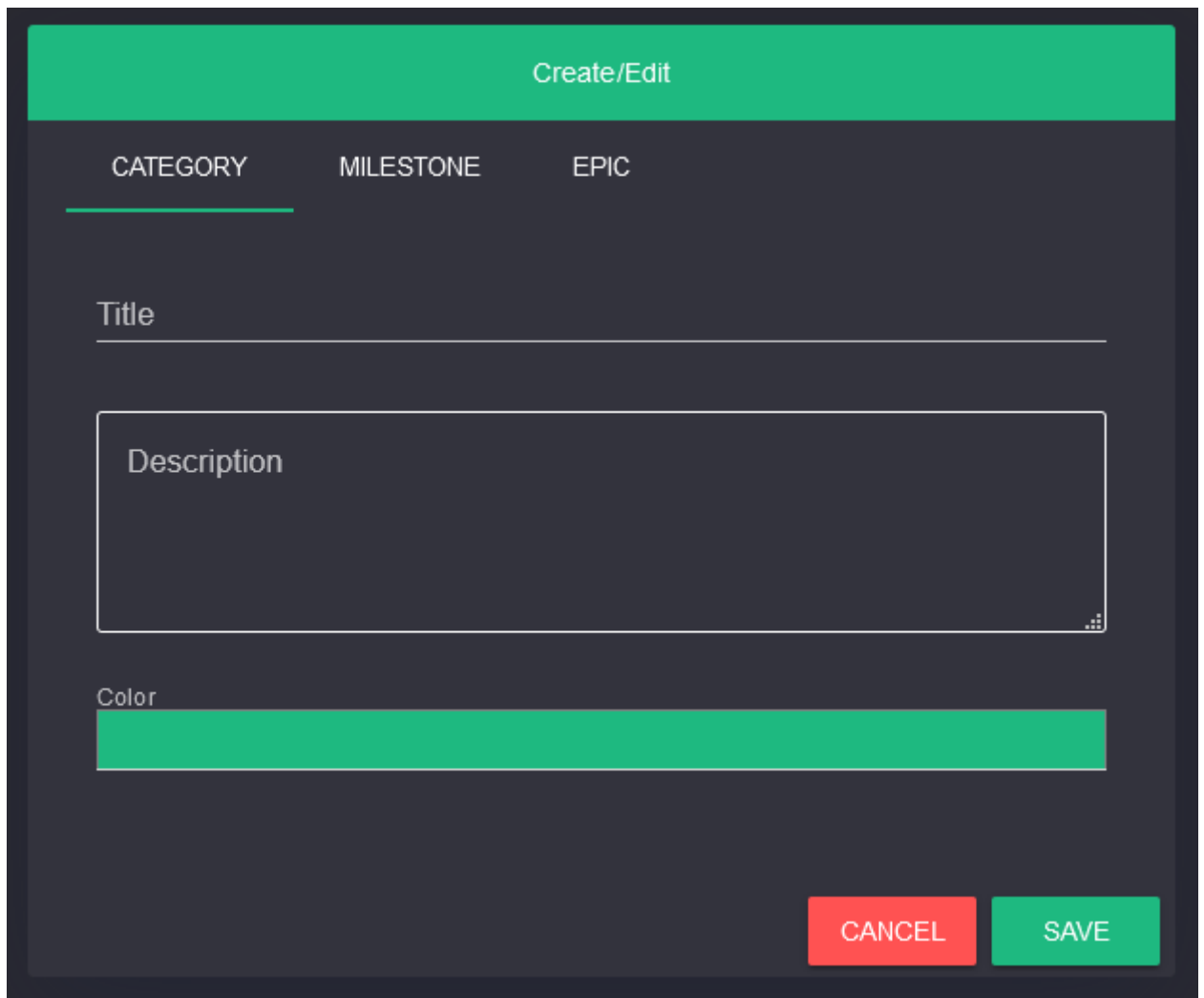
pav. 4.3 Naujo projekto sukūrimo iššokantis langas

Projekto kūrimo lange reikia įvesti pavadinimą, trumpą aprašymą, bei projekto pradžios ir pabaigos datas. Sukūrus naują projektą, jis atsiras projektų sąraše, jį atidaryti galima spustelėjus rodyklės ikoną esančią projekto eilutėje (4.4 paveikslėlis).



pav. 4.4 Projektų sąrašas

Naujame projekte dar nėra jokių objektų kuriuos galima atvaizduoti, juos galima sukurti spustelėjus žalią pliuso ikoną ekrano kampe. Iš pradžių atidaroma kategorijos kūrimo forma iššokančiame lange (4.5 paveikslėlis).



Create/Edit

CATEGORY MILESTONE EPIC

Title

Description

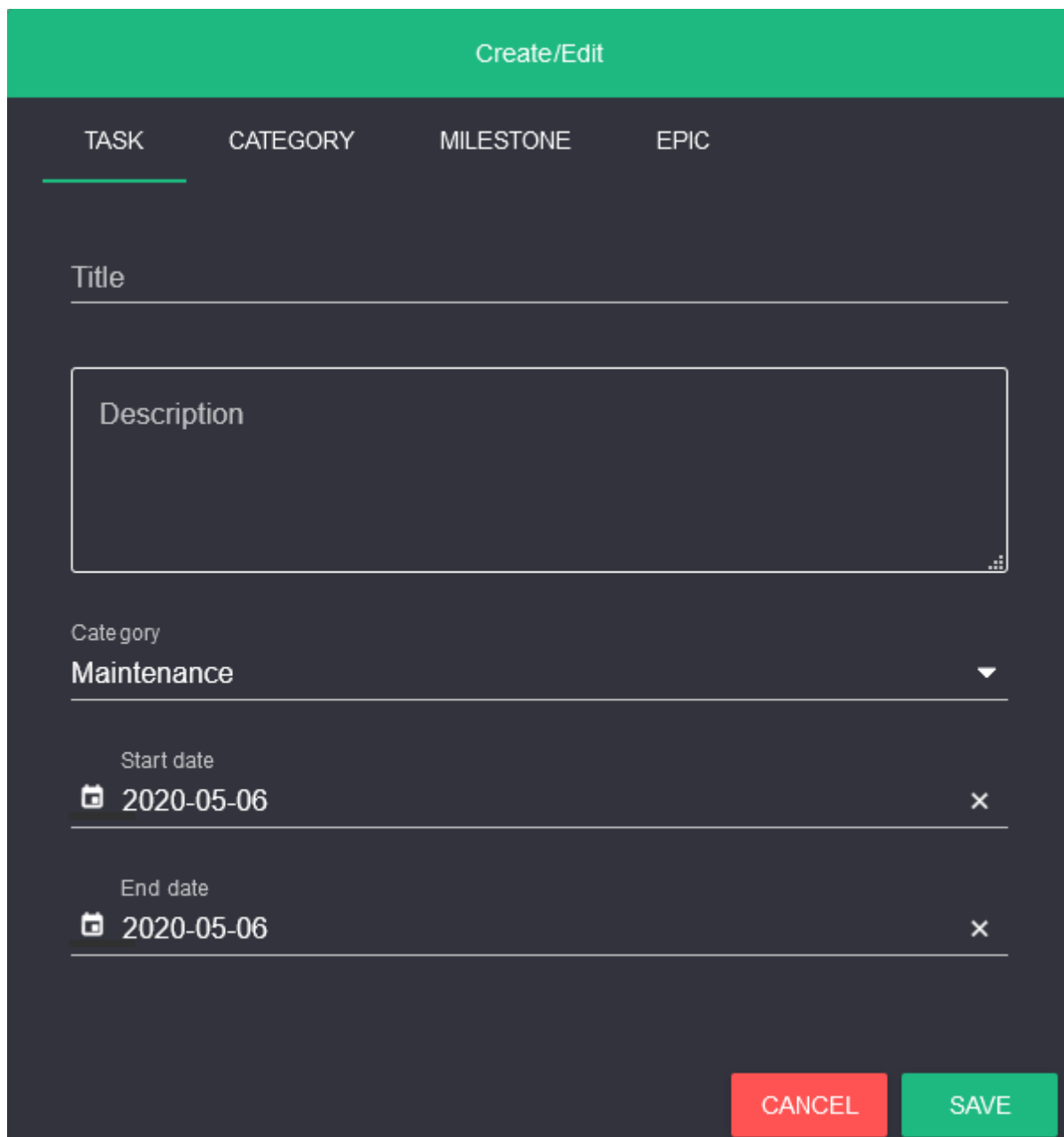
Color

CANCEL SAVE

pav. 4.5 Kategorijos kūrimo iššokantis langas

Norint sukurti kategoriją, reikia nurodyti jos pavadinimą, trumpą aprašymą, bei spalvą kuria norima atvaizduoti kategoriją ir darbus priskirtus šiai kategorijai. Sukūrus kategoriją galima vėl spustelėti pliuso ikoną ir kurti užduotis tai kategorijai. Užduoties kūrimo forma pavaizduota 4.6 paveikslėlyje.





Create/Edit

TASK CATEGORY MILESTONE EPIC

Title

Description

Category  
Maintenance

Start date  
2020-05-06

End date  
2020-05-06

CANCEL SAVE

pav. 4.6 Užduoties kūrimo forma

Norint sukurti užduotį reikia nurodyti jos pavadinimą, aprašymą, pasirinkti vieną iš sukurtų kategorijų, bei nurodyti užduoties pradžios ir pabaigos datą.

Pridėjus pakankamai kategorijų ir užduočių, projektas gali atrodyti gan didelis, todėl galima pridėti kategorijų grupes ir taip sudaryti hierarchiją projekte. Kategorijų grupes galima pridėti paspaudus pliuso ikoną ir pasirinkus „Epic“ skirtuką. Kategorijų grupių kūrimo forma pavaizduota 4.7 paveikslėlyje.

Create/Edit

TASK CATEGORY MILESTONE EPIC

Title

Description

Color

☐ Maintenance

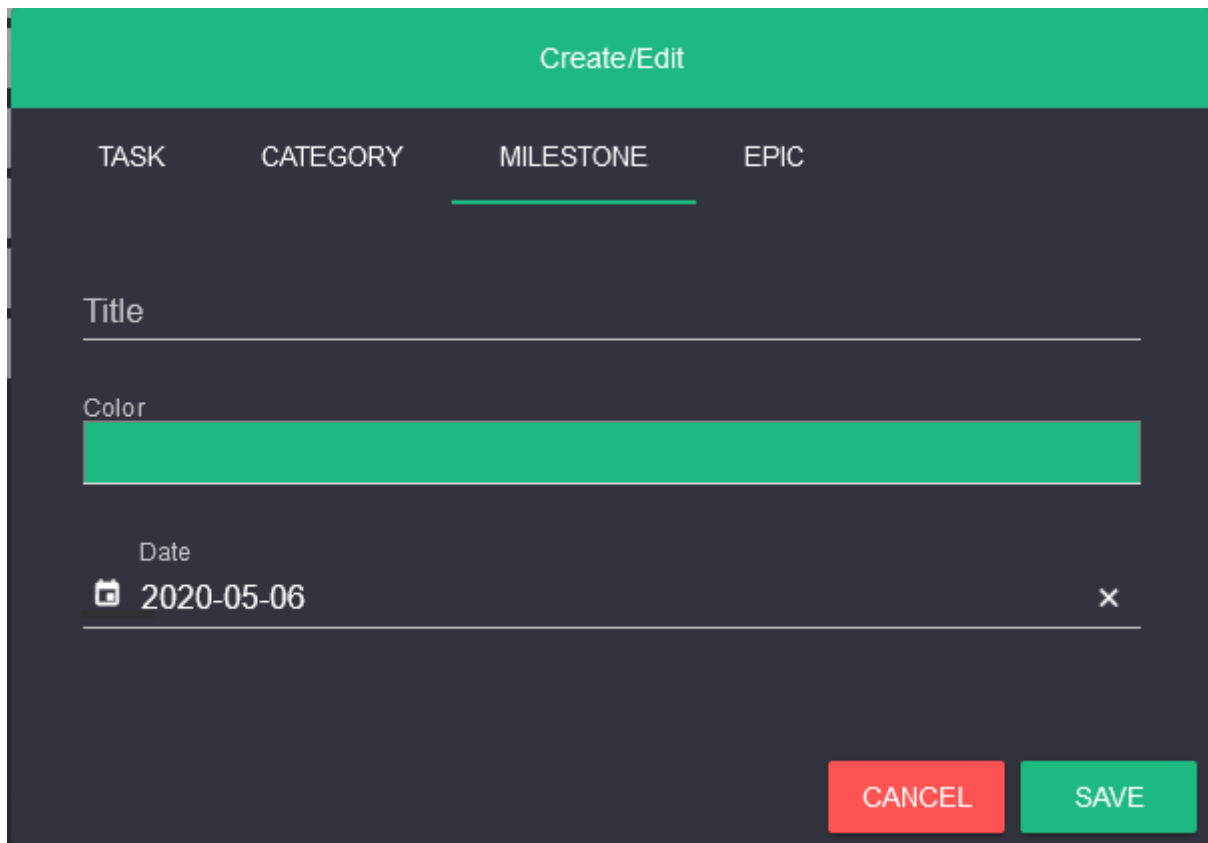
☐ R&D

☐ Vacations

CANCEL SAVE

pav. 4.7 Kategorijų grupės kūrimo forma

Kategorijų grupės kūrimo formoje reikia nurodyti grupės pavadinimą, aprašymą, pasirinkti spalvą, bei kategorijas, kurios bus įtrauktos į šią grupę.



Create/Edit

TASK CATEGORY MILESTONE EPIC

Title

Color

Date

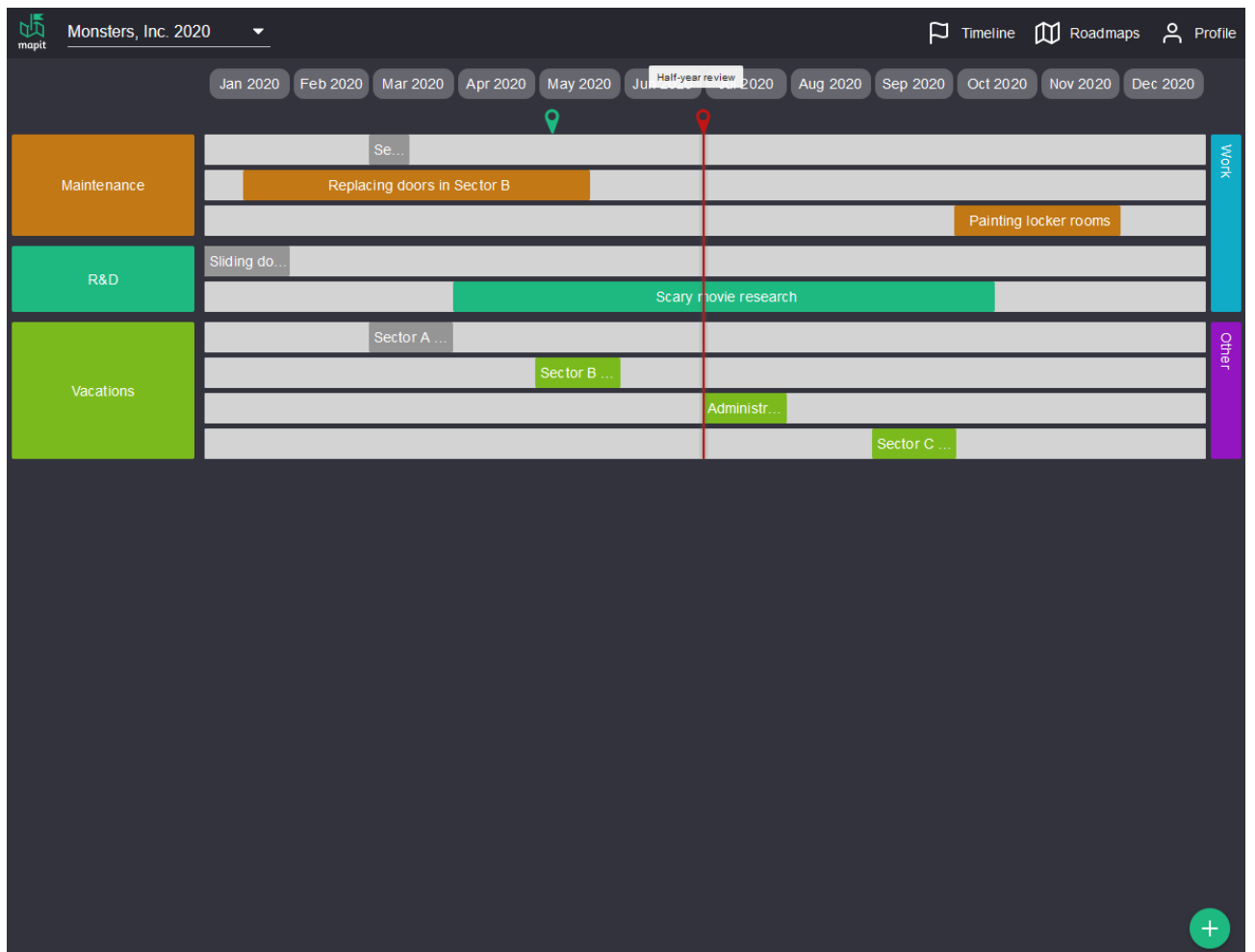
2020-05-06

CANCEL SAVE

pav. 4.8 Projekto etapų pridėjimo forma

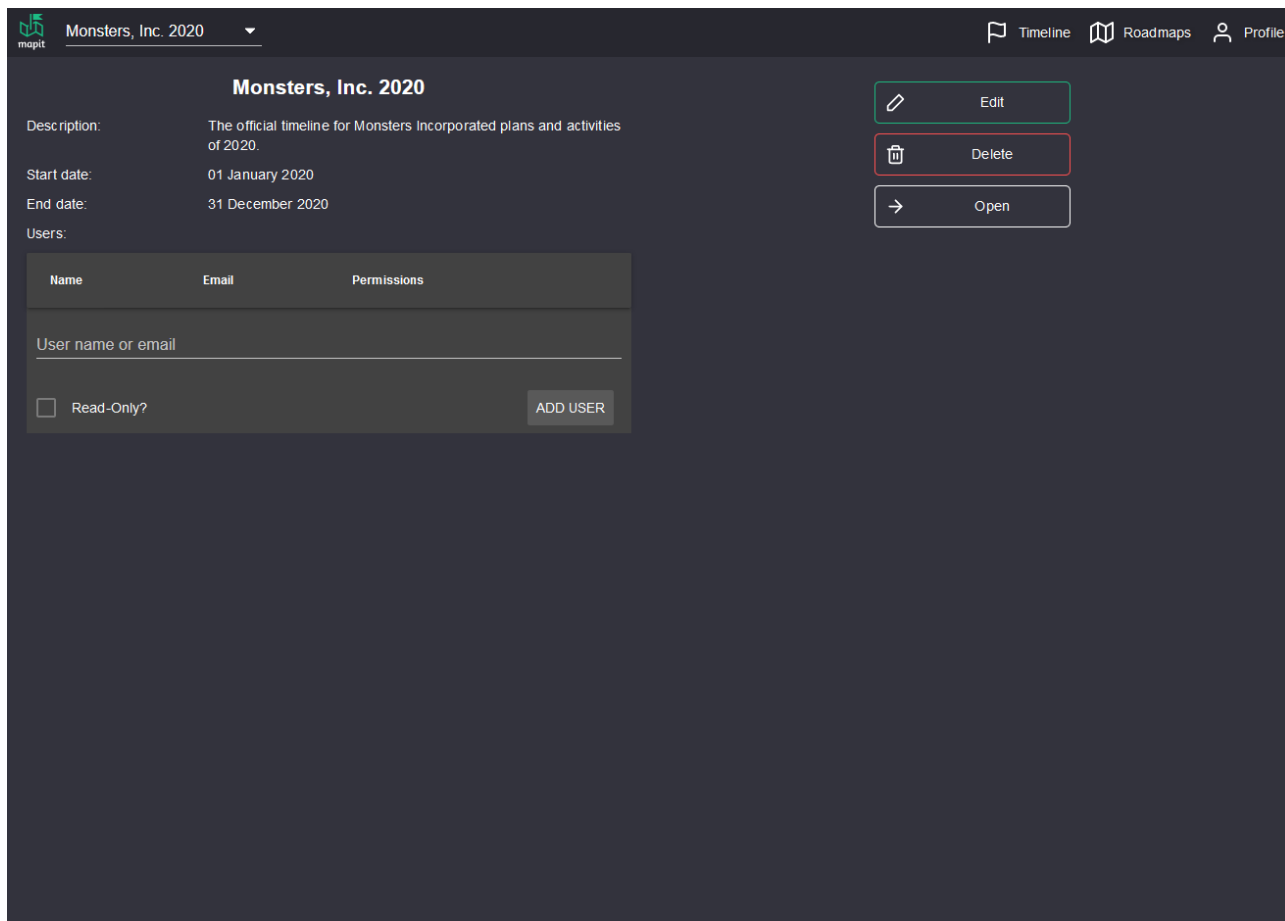
Į projektą taip pat galima pridėti etapus – žymėjimus laiko juostoje, tam tikrą datą, apie norimus įvykius. Iš 4.8 paveikslėlio galima matyti, kad etapo sukūrimui reikia nurodyti, pavadinimą, žymėjimo spalvą, bei datą. Kiekviename projekte yra pažymėta šiandienos data žaliu žymekliu.

Sukūrus visus norimus projekto objektus, viskas patogiai atvaizduojama laiko juostoje – pavyzdys 4.9 paveikslėlis. Paspaudus bet kokį objektą iššoka langas su jo detalėmis ir galimybe redaguoti.



pav. 4.9 Užpildyto projekto laiko juostos pavyzdys

Paruošus projektą galima jį priskirti kitiems prie jo dirbantiems žmonėms. Tai galima padaryti atidarius projektų sąrašą ir spustelėjus ant norimo projekto pavadinimo. Tada atidaromas projekto informacijos langas (4.10 paveikslėlis).

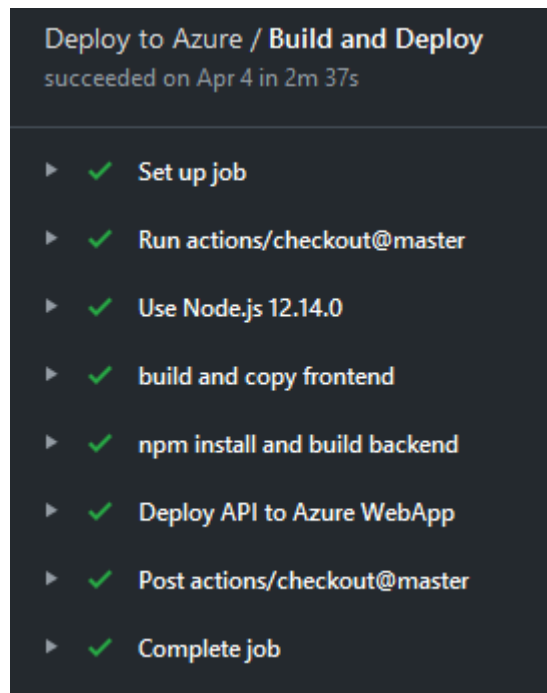


pav. 4.10 Projekto informacijos langas

Šiame lange rodomas vartotojų priskirtų šiam projektui sąrašas. Ši sąrašą galima papildyti suvedus kito vartotojo vardą arba el. pašta, ir paspaudus „Add User“ mygtuką. Taip pat prieš tai galima užžymėti lauką „Read-Only“, tam kad naujai pridėtas vartotojas galėtų matyti projektą tik skaitymo režimu. Tik skaitymo režimas leidžia vartotojui matyti projekto laiko juostą, įvykdyti jam priskirtas užduotis, bei rašyti komentarus, tačiau neleidžia kurti naujų objektų ir redaguoti esamų.

#### 4.3. Diegimo vadovas

Sistemos diegimas yra pilnai automatizuotas naudojant *GitHub Actions*. Programiniu kodu aprašyti diegimo scenarijai yra sužadinami vykstant nustatytiems veiksams *Git* versijavimo sistemoje. Sistemos diegimas į testavimo aplinką yra pradedamas kiekvieną kartą kai atsiranda naujų pakeitimų „master“ šakoje. Sistemos diegimas į produkcinę aplinką startuojamas rankiniu būdu. Diegimo žingsniai pavaizduoti 4.11 paveikslėlyje.













pav. 4.11 Sistemos diegimo žingsniai naudojant *GitHub Actions*

Sistemos diegimas susideda iš 5 pagrindinių žingsnių:

1. Kodo paėmimas iš kodo saugyklos;
2. Tinkamos *Node* aplinkos versijos parinkimas;
3. WEB aplikacijos paruošimas;
4. API paruošimas;
5. Diegimas į *Azure* platformą.

Tai pat prieš ir po šių žingsnių, vykdomi keli papildomi žingsniai reikalingi diegimo serverio paruošimui.


Tam, kad sistema automatinis testavimas ir diegimas pilnai veiktų, reikia pridėti *GitHub* konfigūracijas pavaizduotas 4.12 paveikslėlyje. Šių konfigūracijų nebuvimas pilnai nesustabdys sistemos veikimo, tačiau dėl šios priežasties gali neveikti kai kurie testai.





 API_HOST
 AZURE_API_PUBLISH_PROFILE
 AZURE_DEV_PUBLISH_PROFILE
 AZURE_SAS_TOKEN
 CYPRESS_RECORD_KEY
 DB_HOST
 DB_PASSWORD
 DB_USER
 GOOGLE_CLIENT_ID
 VAPID_KEYS_PUBLIC

pav. 4.12 *GitHub* konfigūracijos

#### 4.4. Administravimo vadovas

Sistemos klaidų žurnalai prieinami administratoriui pačiame įrankyje arba *Azure* failų saugykloje. Žurnalus įrankyje galima pamatyti paspaudus „*Logs*“ navigacijos mygtuką esantį viršuje. Žurnalų langas pavaizduotas 4.13 paveikslėlyje.


Monsters, Inc. 2020

 Timeline
  Logs
  Roadmaps
  Profile

## Logs

Page: 1 of 1

☐ Only errors

Log ID	Message	Level	Timestamp
Pa_Mq5hl3	Ayyy Imao	error	2019-12-25 18:57:38
lwQy4X2qW		error	2020-01-05 13:16:11
Rj-QYizUj	Failed handling google callback. Failed fetching access token. Code: 4/wwEA9m_G18Q9l0Bmp8bk-EenlrWx_eeqTuc8N8gxWzhwqHZAPOGoVxDek41h6Z7sc0PDLkUBYWlxbhEiaeJw_Y; User Id: 1; Error: redirect_uri_mismatch	warn	2020-02-24 21:13:55
RMD22-6E8	Failed handling google callback. Failed fetching access token. Code: 4/wwFuyq2g_O-prfKhjHDBqTzFJ0OFGZhHg3hIRKP9FAG17inzE6PCqrRCuOGwdjmnngRUaaUrrDYjzAmQDKVSW0mA; User Id: 1; Error: redirect_uri_mismatch	warn	2020-02-24 21:15:25
-tEziiPHT	Failed handling google callback. Failed fetching access token. Code: 4/wwFwuJQ505Tg9A-p4va9CYZ-PU8d2VDYPjaq31TjsaWitLw2nwMxZW70keVjqMlprfRtk3ZCuZTs8WLW-sTc; User Id: 1; Error: redirect_uri_mismatch	warn	2020-02-24 21:17:38
U1QeLiQlg	Cannot read property 'createQueryBuilder' of undefined	error	2020-02-24 21:18:40
fkP3wxDvO	Failed handling google callback. Failed fetching access token. Code: 4/wwH0FAwsiQPylcBsAi8ZjYt94TTJdeO0Qp2gVicF9cthW25Mfsew4SfA8Xi8ptTbTTDQXfajLKr_HrZOtFE0; User Id: 1; Error: redirect_uri_mismatch	warn	2020-02-24 21:18:56
0g4KAy06L	Cannot read property 'createQueryBuilder' of undefined	error	2020-02-24 21:37:42
kkZ-Up4IU	Cannot read property 'manager' of undefined	error	2020-03-22 17:51:57
a5sCEwOjbJ	Cannot read property 'manager' of undefined	error	2020-03-22 17:51:57
862u3l3uB	Cannot read property 'manager' of undefined	error	2020-04-02 13:50:16
26LorsNCUb	Cannot read property 'manager' of undefined	error	2020-04-02 13:50:16
DXbt0DWvb		error	2020-05-06 13:42:03

CLEAR LOGS

pav. 4.13 Klaidos žurnalų langas

Šiuos įrašus galima filtruoti, taip, kad būtų rodomos tik klaidos, pažymėjus „*Only errors*“ lauką. Taip pat galima išvalyti klaidas paspaudus „*Clear Logs*“ mygtuką. Paspaudus ant įrašo, atidaromas iššokantis langas su klaidos detalėmis, pavaizduotas 4.14 paveikslėlyje.



Preview Log Entry

Log ID:U1QeLiQIg

Level:error

Timestamp:2020-02-24 21:18:40

Message:Cannot read property 'createQueryBuilder' of undefined

Stack:TypeError: Cannot read property 'createQueryBuilder' of undefined  
at RoadmapService.<anonymous> (/home/site/wwwroot/build/services/roadmapService.js:27:66)  
at Generator.next (<anonymous>)  
at /home/site/wwwroot/build/services/roadmapService.js:8:71  
at new Promise (<anonymous>)  
at \_\_awaiter (/home/site/wwwroot/build/services/roadmapService.js:4:12)  
at RoadmapService.getAll (/home/site/wwwroot/build

Raw log:

```
{  "requestData": {    "url": "/api/roadmaps",    "params": {},    "query": {},    "body": {}  },  "level": "error",  "message": "Cannot read property 'createQueryBuilder' of undefined",  "log_id": "U1QeLiQIg"}
```

CLOSE

pav. 4.14 Išsamios klaidos detalės

Klaidos detalėse matoma daug naudingos informacijos, padėsiančios diagnozuoti ir išspręsti klaidą. Tarp šios informacijos matoma ir kokia API užklausa buvo daroma įvykus šiai klaidai.

## Rezultatai ir išvados

Atlikus darbą prieita prie šių išvadų:

1. Atlikus konkurentų analizę paaiškėjo, kad yra daug sistemų turinčių panašias galimybes. Tačiau labai mažai tokių įrankių koncentruojasi į mobiliuosius įrenginius, kai kurie išvis nededa pastangų į šią sritį. Taip pat dauguma įrankių turi didžiulę kainą, tam kad vartotojas galėtų naudoti visą sistemos funkcionalumą. Dėl šių priežasčių nuspręsta kurti šį projektų valdymo įrankį.
2. Buvo pasirinkta *Node.js* technologija naudota sistemos kūrimui, dėl jos paplitimo ir bendruomenės. Plati ekosistema suteikia daug įvairių paketų ir bibliotekų pritaikomų įvairioms problemoms spręsti, bei integracijoms.
3. Darbo metu buvo naudojamas testavimu pagrįstas programavimas. Tai užtikrino 92% kodo padengimą testais, sistemos patikimumą, bei sumažino regresijos klaidų skaičių. Pastebėta, kad didelė dalis klaidų buvo surasta automatiškai leidžiamų testų *GitHub* platformoje.
4. Dabartinė sistemos versija turi tam tikrų apribojimų veikiant be interneto ryšio. *Service Worker* komponentas stipriai padėjo išspręsti daugumą susijusių problemų, tačiau dar nesukurtas funkcionalumas, leidžiantis vartotojams daryti atnaujinimus be interneto ryšio.
5. Pastebėta, kad naudojant progresyviąsias saityno technologijas galima pasiekti beveik tiek pat kiek turint atskirą mobilią aplikaciją, tačiau sutaupomi kaštai, nes reikia vystyti tik vieną sistemą.
6. Dėl riboto laiko sistema nebuvo pilnai įgyvendinta. Ateityje planuojama papildyti sistemą – suteikiant pilną funkcionalumą (įskaitant atnaujinimus) naudojantis be interneto ryšio, patobulinti vartotojo sąsają patogesniu valdymu.

## Literatūros sąrašas

1. Richard, Sam ir LePage, Pete. What are Progressive Web Apps? *Web.Dev.* [Tinkle] 2020 m. Vasario 24 d. [Cituota: 2020 m. Gegužės 7 d.] <https://web.dev/what-are-pwas/>.
2. AHA! - App on Google Play. *Google Play Store.* [Tinkle] [Cituota: 2020 m. Gegužės 7 d.] <https://play.google.com/store/apps/details?id=io.aha.mobile>.
3. Aha! Pricing. *Aha!* [Tinkle] [Cituota: 2020 m. Gegužės 7 d.] <https://www.aha.io/pricing>.
4. Roadmunk plans and pricing. *Roadmunk.* [Tinkle] [Cituota: 2020 m. Gegužės 7 d.] <https://roadmunk.com/pricing>.
5. Toggl Plan. *Toggl Plan.* [Tinkle] [Cituota: 2020 m. Gegužės 7 d.] <https://toggl.com/plan>.
6. TypeORM Documentation. *TypeORM.* [Tinkle] [Cituota: 2020 m. Gegužės 7 d.] <https://typeorm.io/>.
7. Sending VAPID identified WebPush Notifications via Mozilla's Push Service. *Mozilla Services.* [Tinkle] 2016 m. Rugpjūčio 23 d. [Cituota: 2020 m. Gegužės 7 d.] <https://blog.mozilla.org/services/2016/08/23/sending-vapid-identified-webpush-notifications-via-mozillas-push-service/>.
8. Progressive web apps (PWAs). *MDN.* [Tinkle] [Cituota: 2020 m. Gegužės 7 d.] [https://developer.mozilla.org/en-US/docs/Web/Progressive\\_web\\_apps](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps).
9. Service Worker API. *MDN.* [Tinkle] [Cituota: 2020 m. Gegužės 7 d.] [https://developer.mozilla.org/en-US/docs/Web/API/Service\\_Worker\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API).