

Informatikos fakultetas

**T120B162 Programų sistemų testavimas**

**4 laboratorinis darbas**

|  |  |  |
| --- | --- | --- |
|  | Studentas: | Mangirdas Kazlauskas, IFF-4/1 |
| Dėstytojas: | doc. Šarūnas Packevičius |

Kaunas 2017

Turinys

[1. Įvadas 3](#_Toc500188030)

[2. Projektų valdymo įrankio pasirinkimas 3](#_Toc500188031)

[3. Projektų valdymo įrankio aplinkos paruošimas 3](#_Toc500188032)

[4. Testavimo užduočių paruošimas ir užregistravimas 7](#_Toc500188033)

[5. Testavimo užduočių atlikimas 12](#_Toc500188034)

[6. Klaidų ir jų ištaisymo registravimas 14](#_Toc500188035)

[7. Išvados 16](#_Toc500188036)

# Įvadas

**4 laboratorinio darbo tikslas –** panaudoti projektų valdymo įrankį savo programinės įrangos kūrimo bei testavimo procese.

**Darbo uždaviniai:**

1. Susirasti bei išmokti naudotis projektų valdymo įrankiu;
2. Tikslingai panaudoti įrankį programinės įrangos kūrimo bei testavimo procesuose;
3. Pateikti įrankio naudojimosi rezultatus bei įžvalgas laboratorinio darbo ataskaitoje.

# Projektų valdymo įrankio pasirinkimas

Rinkoje yra siūlomas ne vienas projektų valdymui skirtas įrankis:

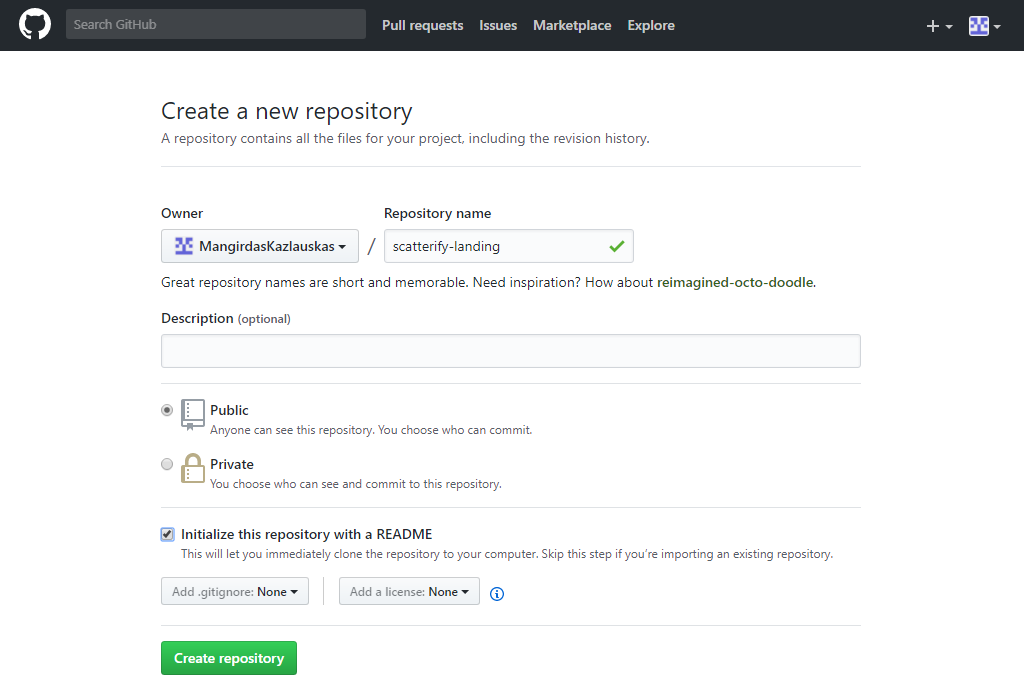
1. Bitbucket
2. GitHub
3. Jira
4. Redmine

Iš šių rastų įrankių, buvo pasirinktas GitHub. Šiam sprendimui įtaką darė keli veiksniai:

1. Įrankis yra nemokamas. Tiesa, nemokamas jis yra tada, kai talpinami projekto failai (repozitorija) yra prieinami viešai (public), kitu atveju, t.y., jei norima, kad failai būtų privatūs, įrankis yra mokamas. Tačiau studentams suteikia „Student Developer pack|, kuris leidžia sukurti privačias repozitorijas (neribotą kiekį);
2. Bendruomenė, kuri naudojasi šiuo įrankiu, yra didelė, todėl iškilus bet kokiems klausimams, susijusiems su GitHub, labai nesunku surasti atsakymą ar iškilusios problemos sprendimą;
3. Įrankis yra minimalistinis, todėl jį perprasti galima daug greičiau nei kitus, daugiau papildomų funkcijų turinčius įrankius (pvz., Jira);
4. GitHub turi daugybę integracijų su kitais įrankiais ar programavimo aplinkomis (pvz., Visual Studio, Atom.io), kas pagreitina programinės įrangos kūrimo procesą, padaro jį patogesniu.

# Projektų valdymo įrankio aplinkos paruošimas

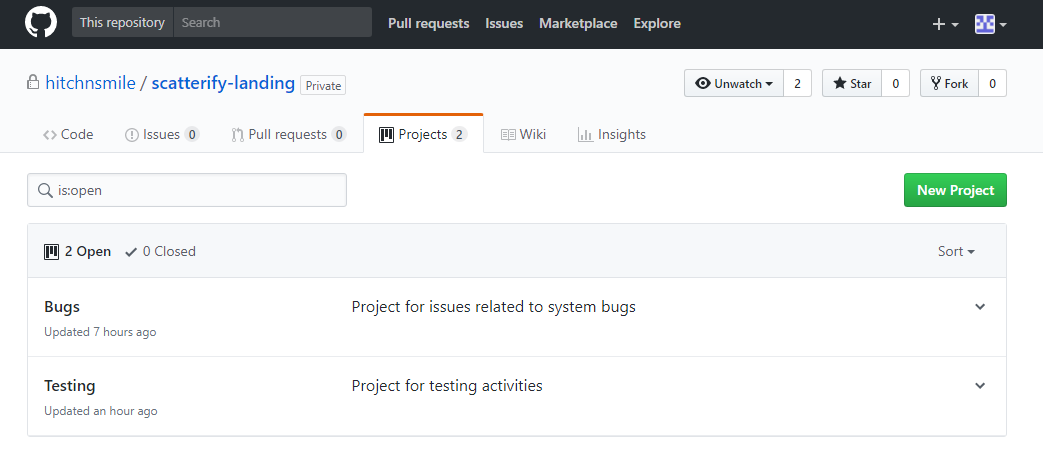
Pirmiausia GitHub yra sukuriamos repositorijos, kuriose bus talpinami kuriamos sistemos failai (1 pav.). GitHub įrankį naudosiu kuriamos sistemos kliento pusės programiniam kodui valdyti bei saugoti, todėl yra sukuriamos dvi repozitorijos: scatterify-landing (prezentaciniam puslapiui talpinti bei valdyti) ir scatterify-dashboard (internetinei aplikacijai talpinti bei valdyti).



1 pav. Repozitorijos sukūrimo langas

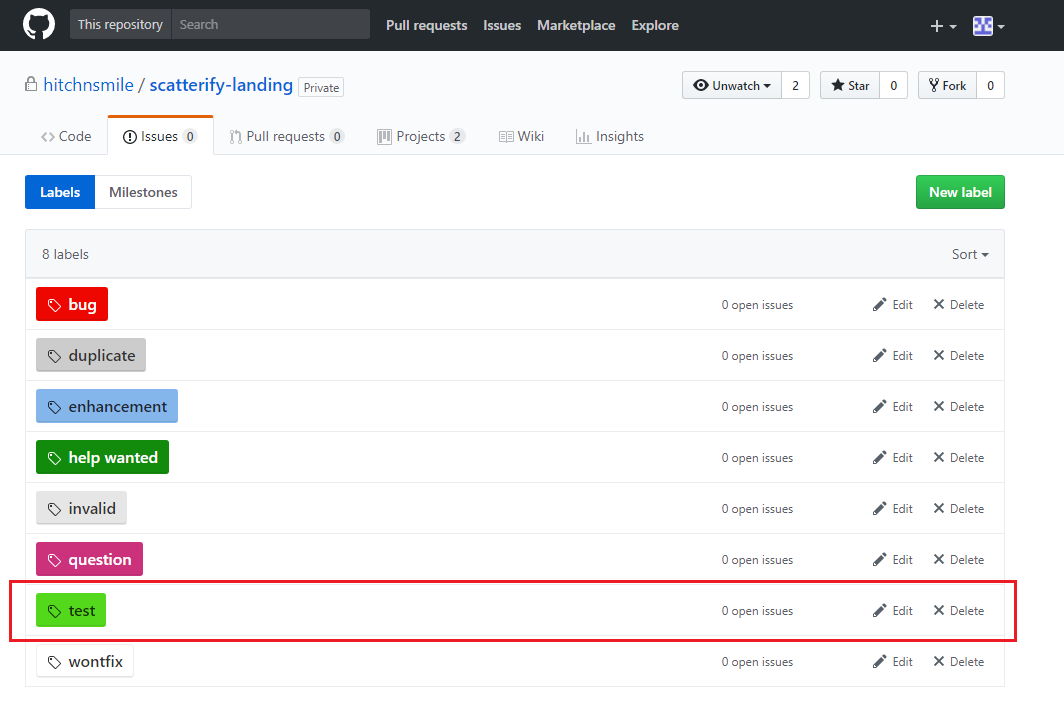
Užregistravus repozicotijas GitHub svetainėje, kiekvienos repozitorijos „Projects“ skiltyje yra užregistruojami projektai, kurių paskirtis – tam tikrų procesų valdymas programinės įrangos kūrimo, testavimo eigoje. Pvz., kiekvienoje repozitorijoje yra sukuriama po 2 projektus (2 pav.):

1. Bugs – projektas, skirtas registruoti, valdyti klaidas, atsiradusias naudojantis kuriama sistema arba atsiradusias sistemos kūrimo metu (pvz., pats programuotojas užregistruoja atsiradusią klaidą kode, kurią vėliau ištaisys);
2. Testing – projektas, skirtas testavimo procesui valdyti, t.y., šiam projektui priskiriami testai, kurie turi būti atlikti testuotojų.



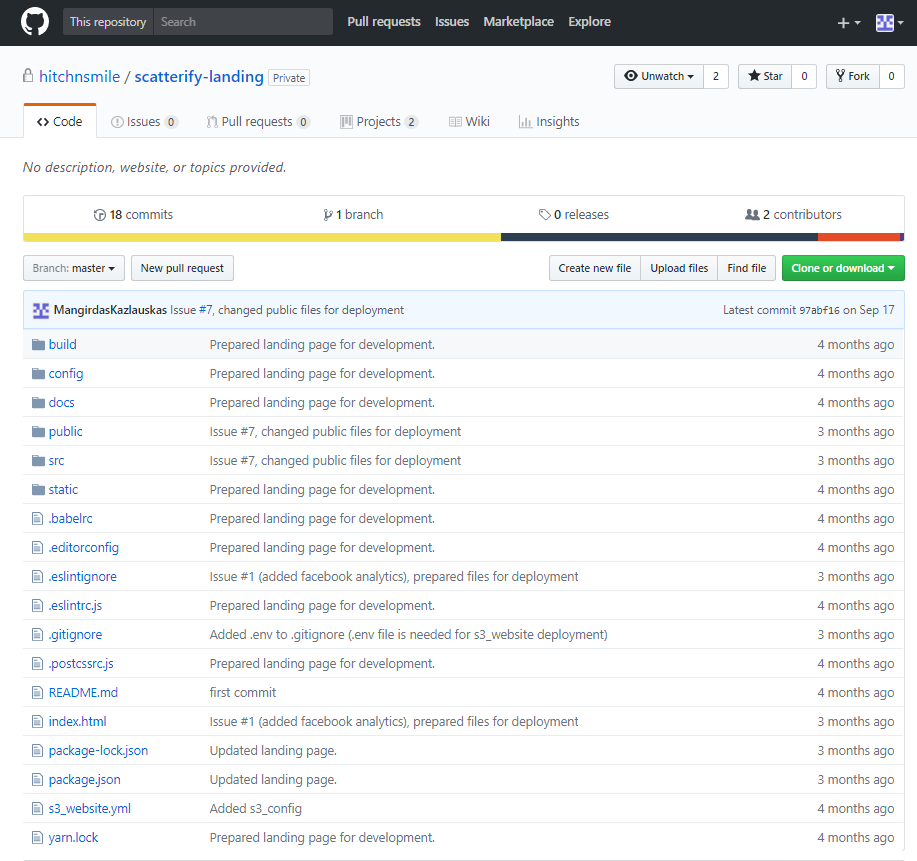
2 pav. Svetainės reprezentacinio puslapio “Projects” skiltis GitHub

Užregistruotos klaidos, testai ar kitos užduotys gali būti pažymėtos etiketėmis, kurios padeda geriau atskirti skirtingas užregistruotas užduotis, pagal etiketes galima atsirinkti konkrečias užduotis paieškoje. Pagal nutylėjimą, etiketė „bug“ jau yra sukurta. Tačiau tarp etikečių nėra jokios, kuri tiktų testavimo užduotims žymėti. Tam tikslui sukuriama nauja – „test“ etiketė (3 pav.).

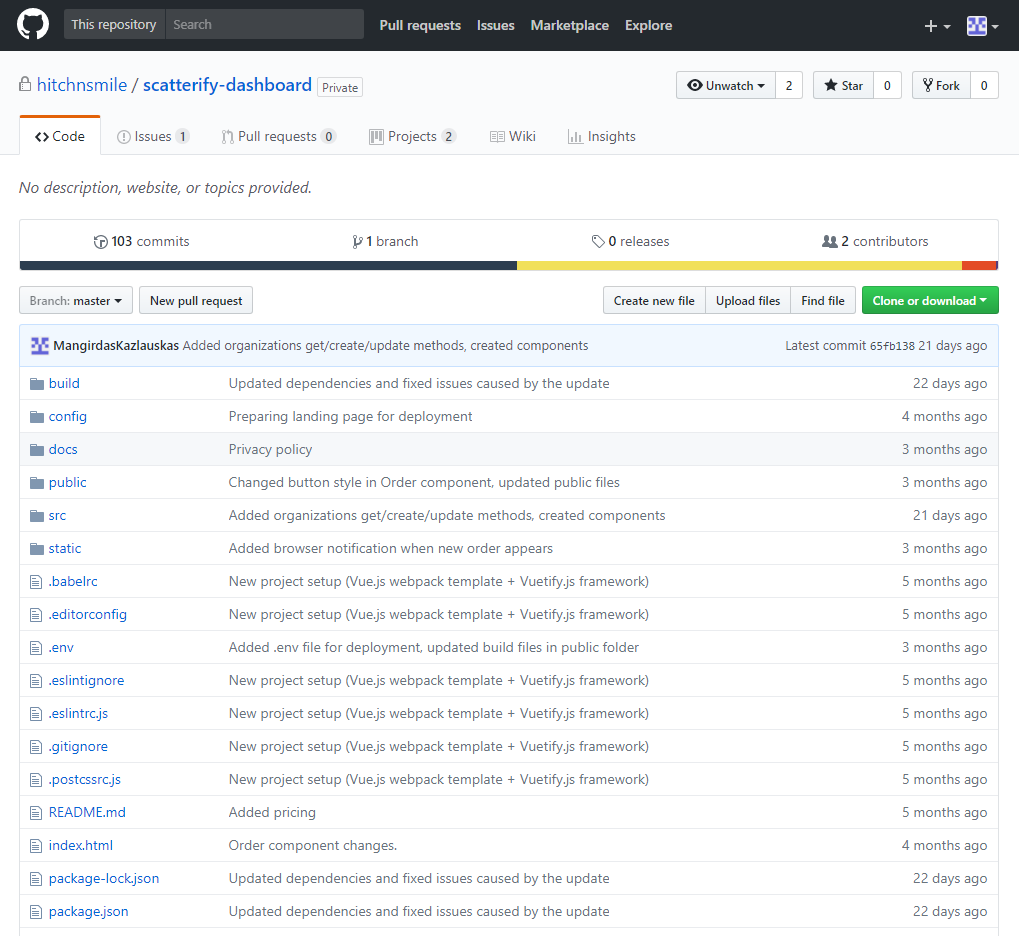


3 pav. Galimos pasirinkti etiketės registruojant užduotį GitHub

Kad būtų ką testuoti ir testuotojai galėtų paleisti sistemą lokalioje aplinkoje, repozitorijose yra patalpinami projektų failai: prezentacinio puslapio projekto failai (4 pav.) ir internetinės aplikacijos failai (5 pav.). Jie į repozitorijas yra patalpinami naudojantis Git versijų kontrolės įrankio pagalba.

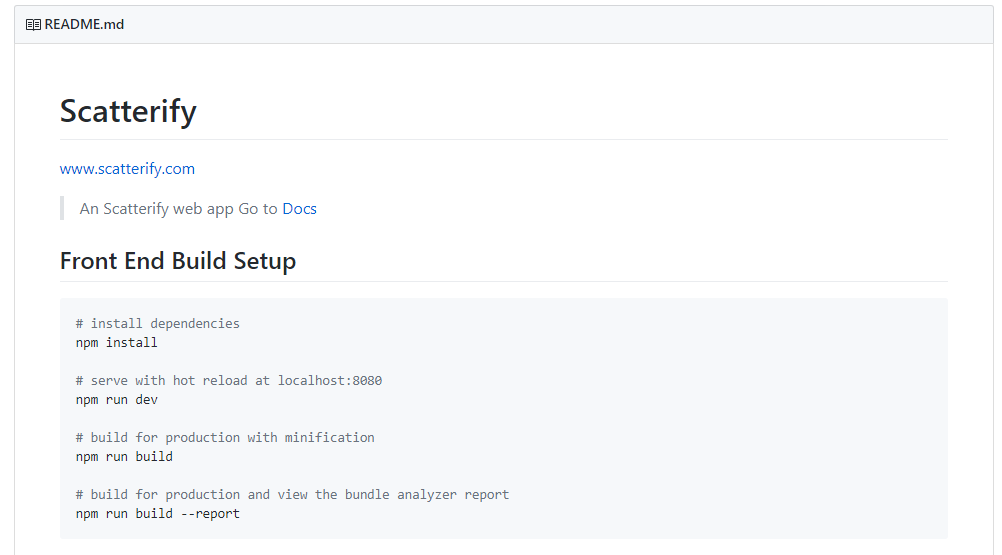


4 pav. Reprezentacinio svetainės puslapio failai, patalpinti repozitorijoje



5 pav. Internetinės aplikacijos failai, patalpinti repozitorijoje

Internetinę aplikaciją paleisti lokalioje aplinkoje yra gana sudėtinga, todėl repozitorijoje kartu pateikiamos instrukcijos, nurodančios, ką reikia daryti, norint sistemą paleisti lokaliai kompiuteryje (6 pav.).

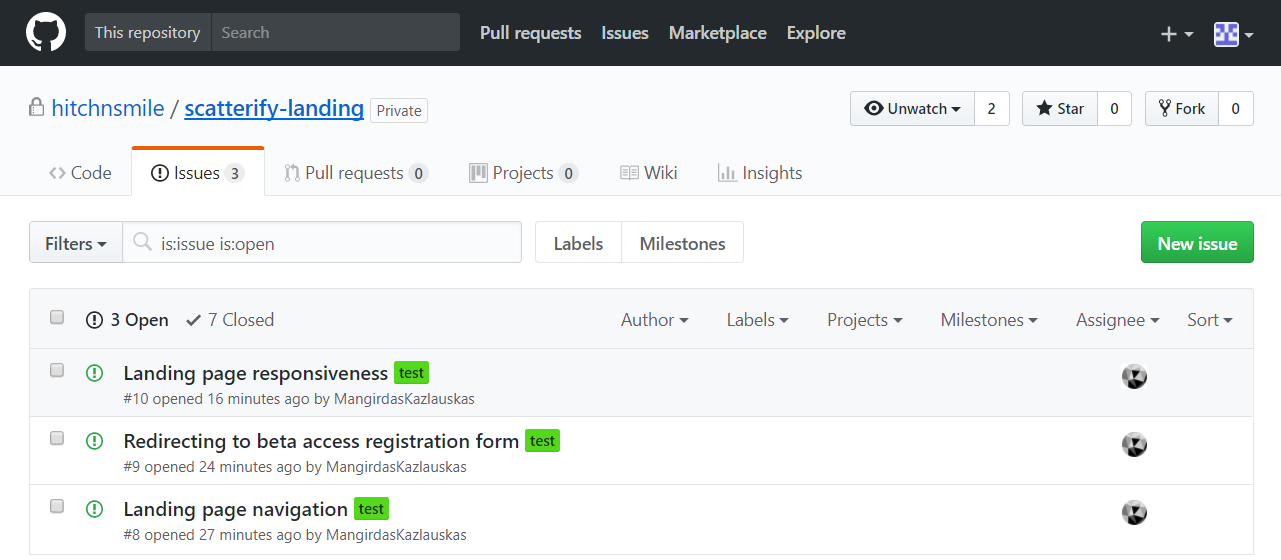


6 pav. Instrukcijos, nurodančios internetinės aplikacijos projekto diegimo lokalioje aplinkoje žingsnius

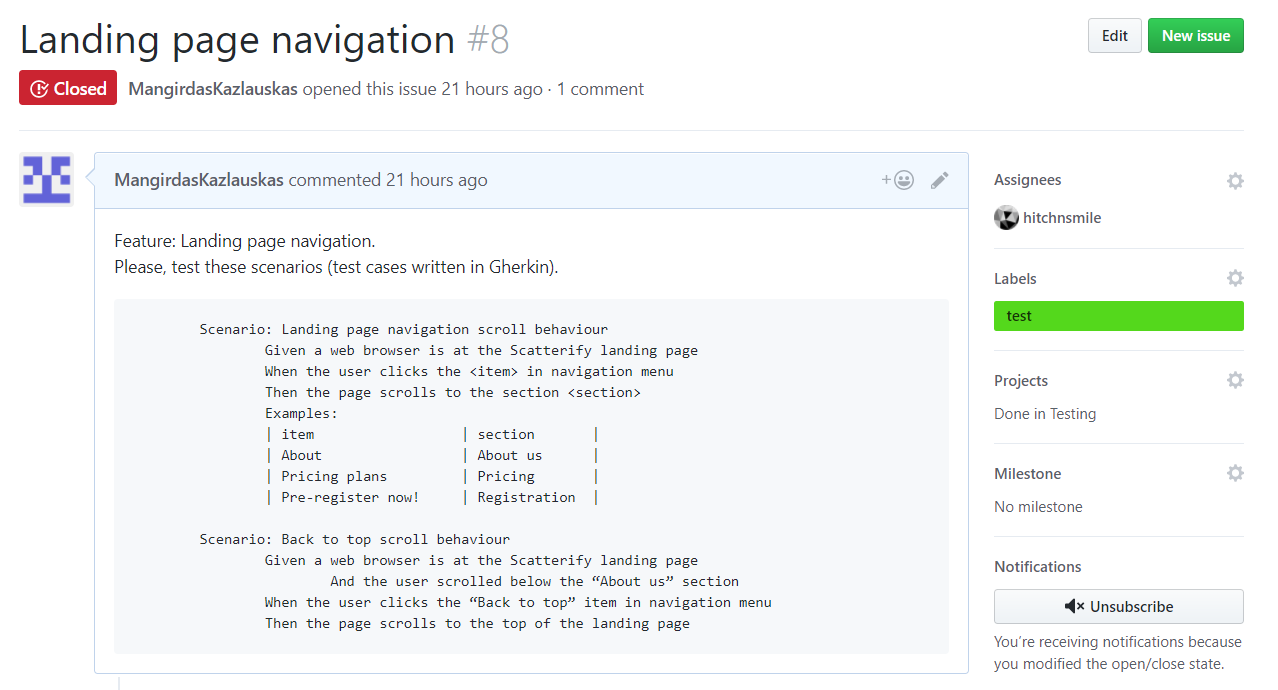
# Testavimo užduočių paruošimas ir užregistravimas

Remiantis 1 laboratorinio darbo metu paruoštu testavimo planu (testavimo scenarijais), GitHub aplinkoje buvo užregistruotos tokios testavimo užduotys:

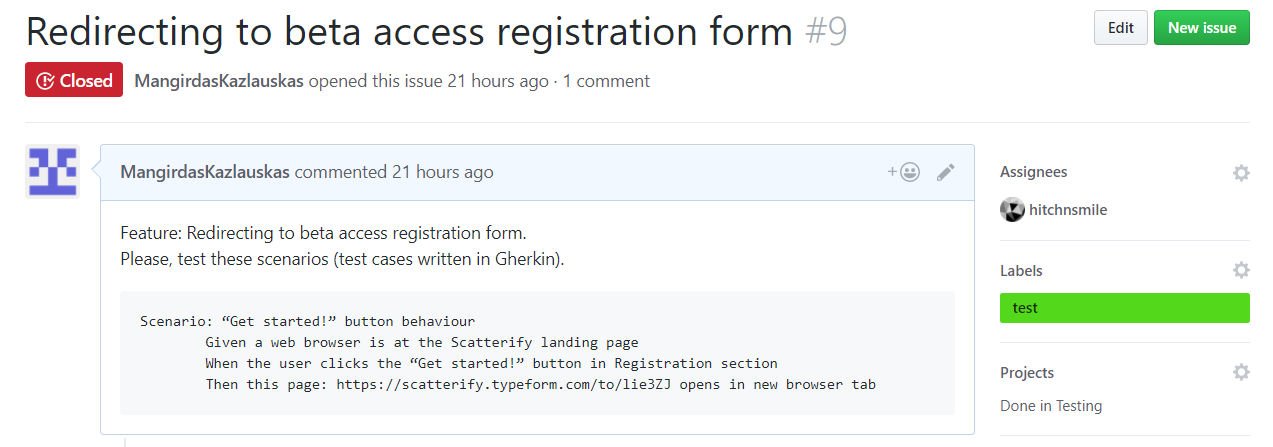
1. Prezentacijos puslapio testavimo užduotys (7-10 pav.):



7 pav. Užregistruotos testavimo užduotys prezentacijos puslapiui testuoti



8 pav. Puslapio navigacijos testavimo užduotis

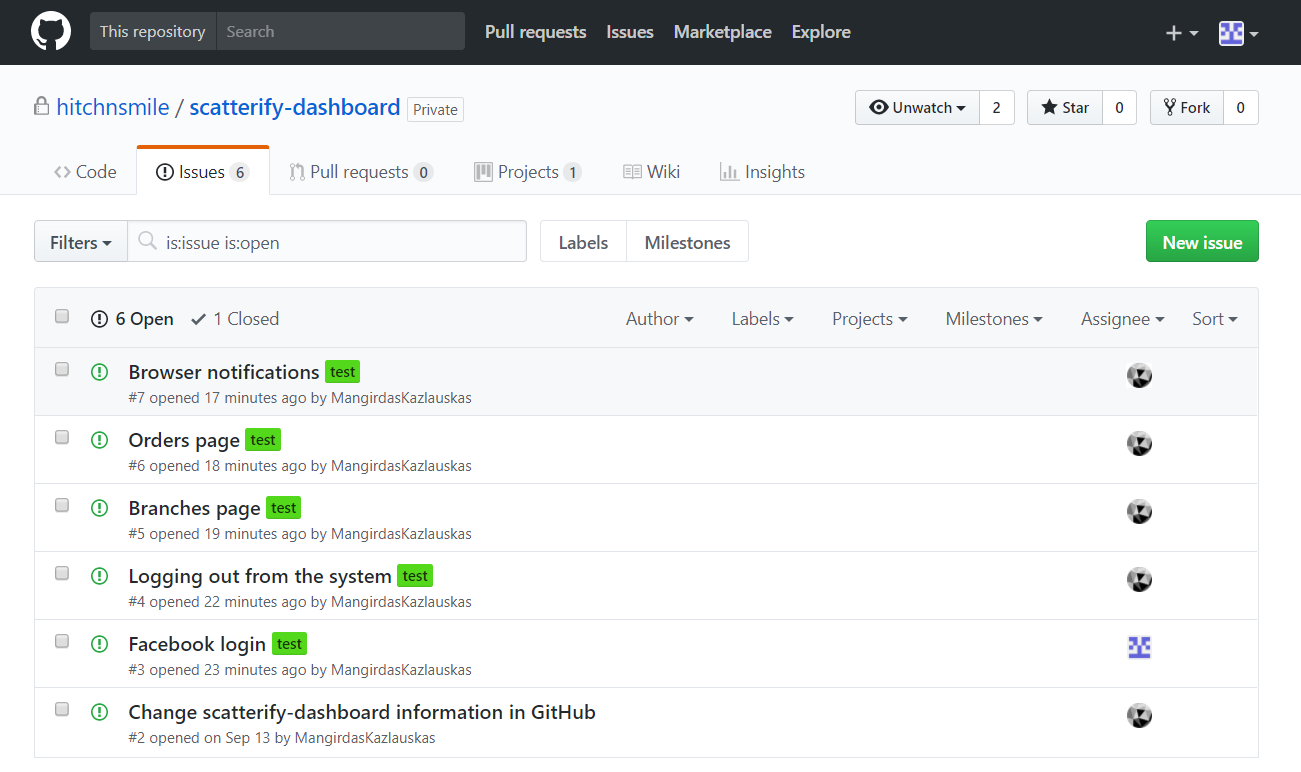


9 pav. Naudotojo nukreipimo į išorinę svetainę testavimo užduotis

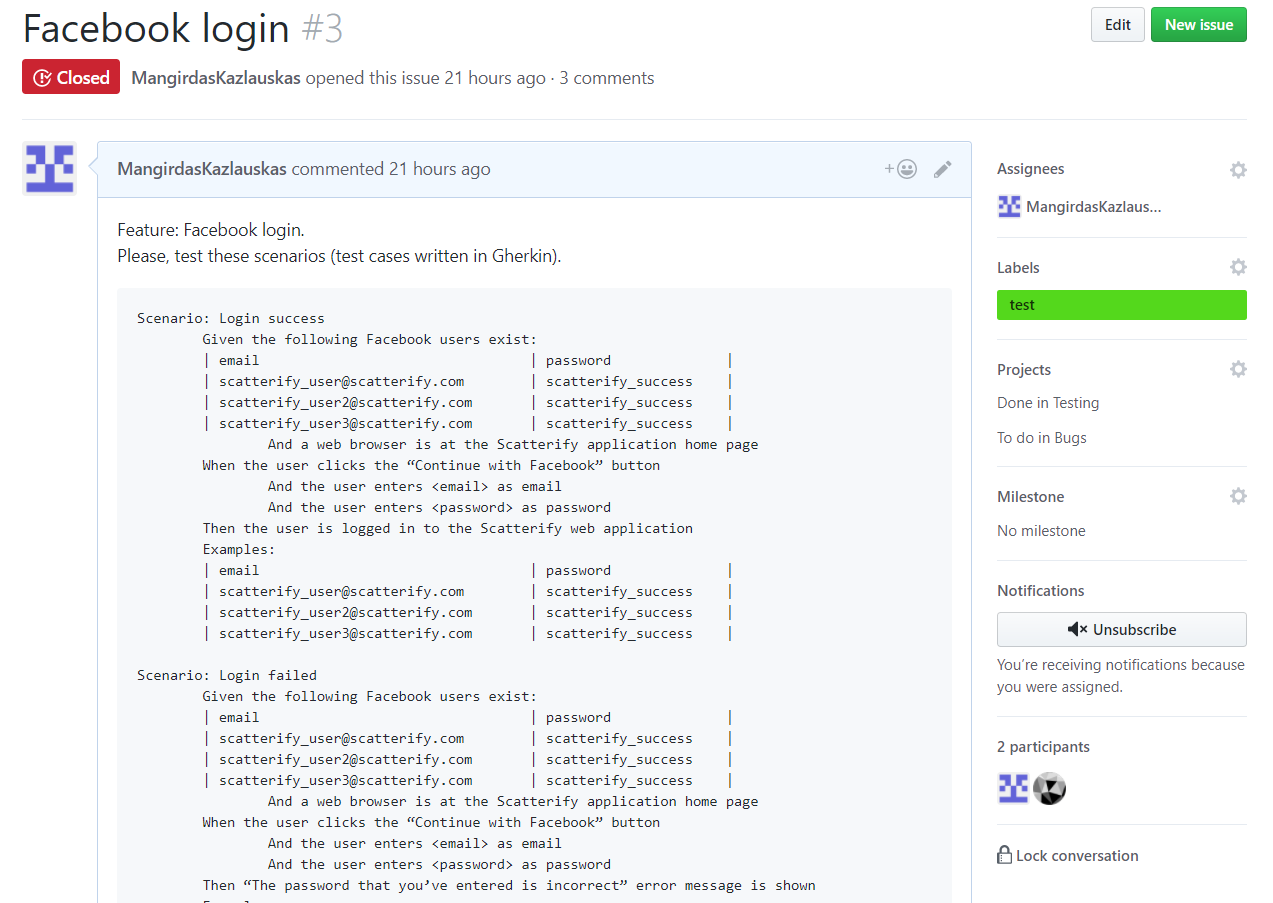


10 pav. Prezentacijos puslapio prisitaikymo prie mobiliųjų įrenginių ekranų testavimo užduotis

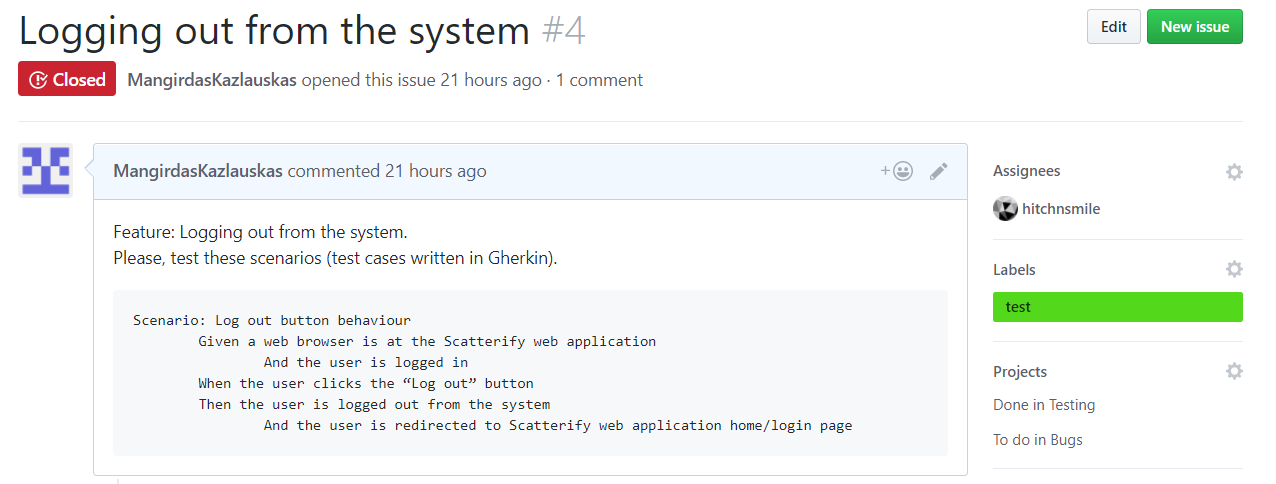
1. Internetinės aplikacijos testavimo užduotys (11-16 pav.):



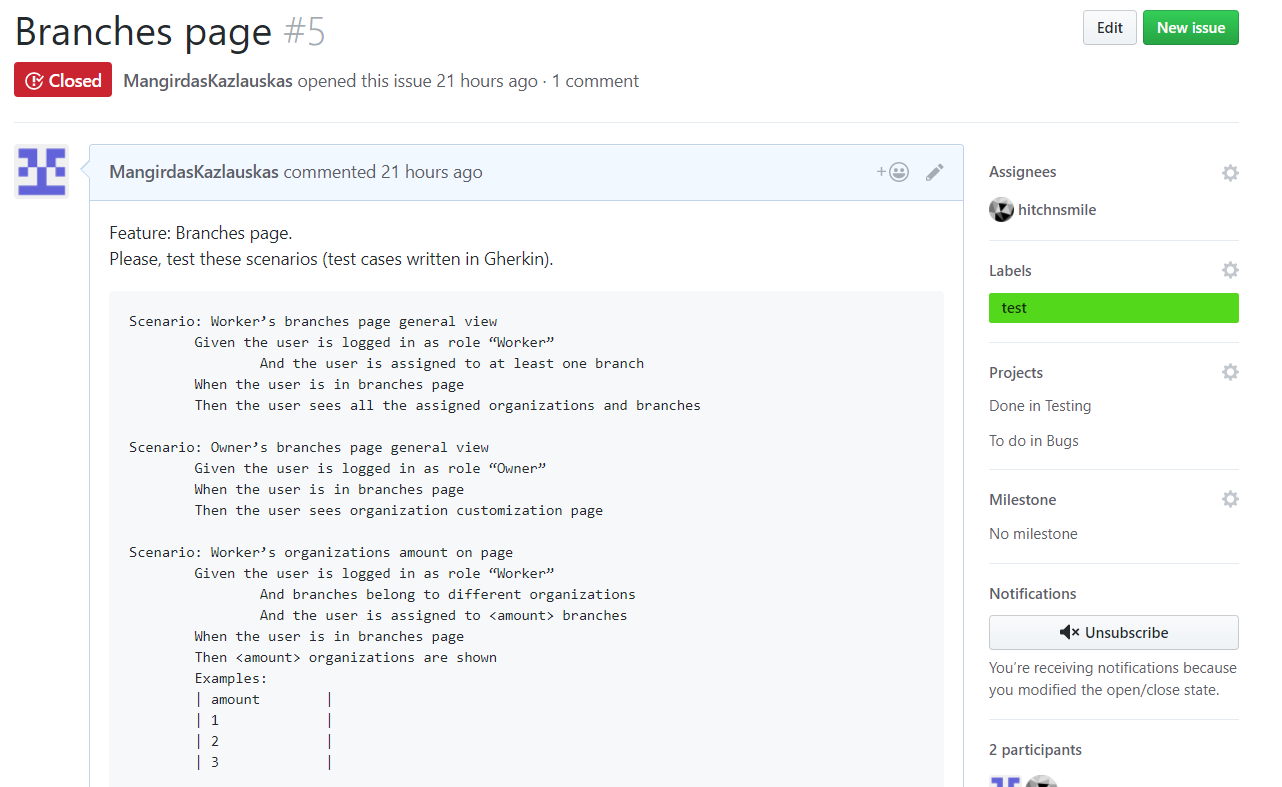
11 pav. Užregistruotos testavimo užduotys internetinei aplikacijai testuoti



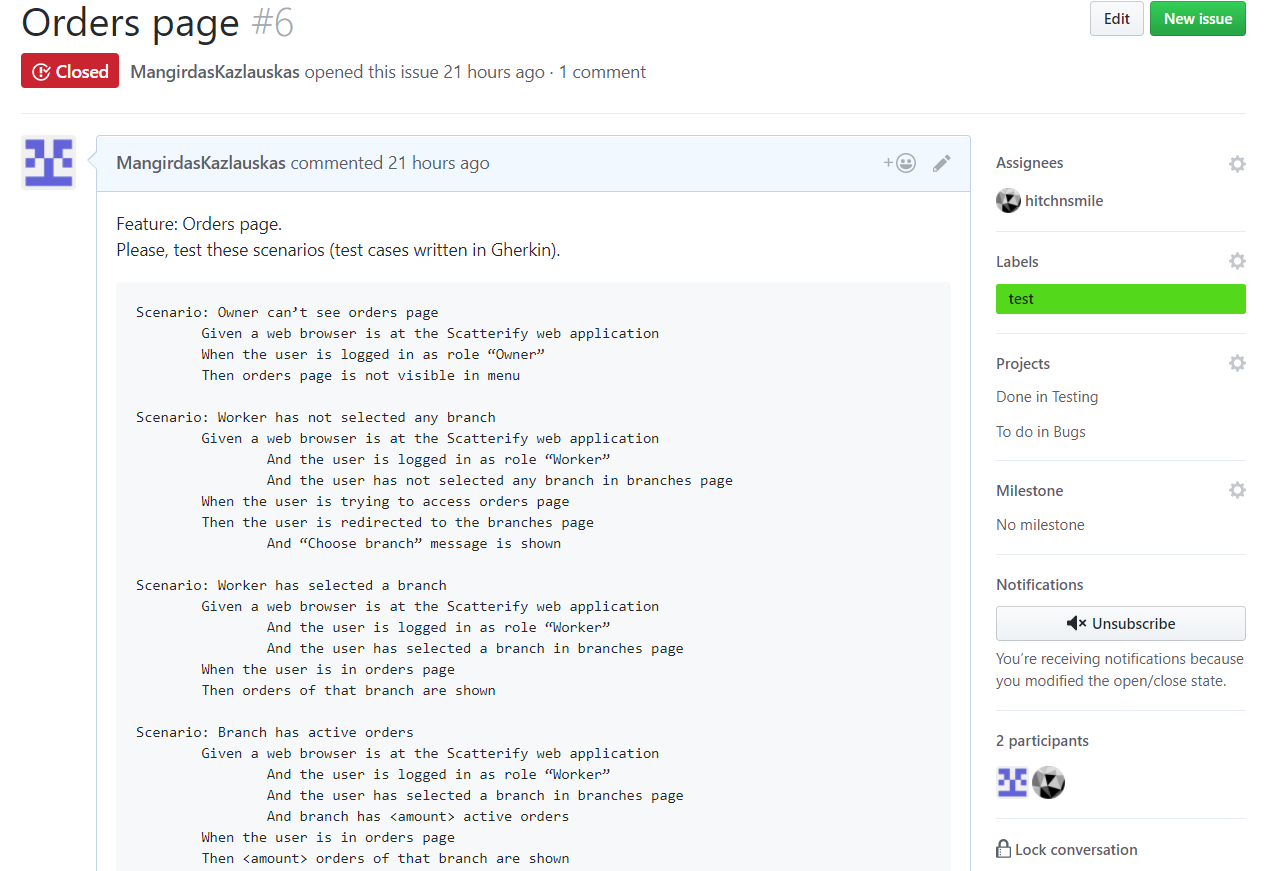
12 pav. Prisijungimo prie internetinės aplikacijos testavimo užduotis



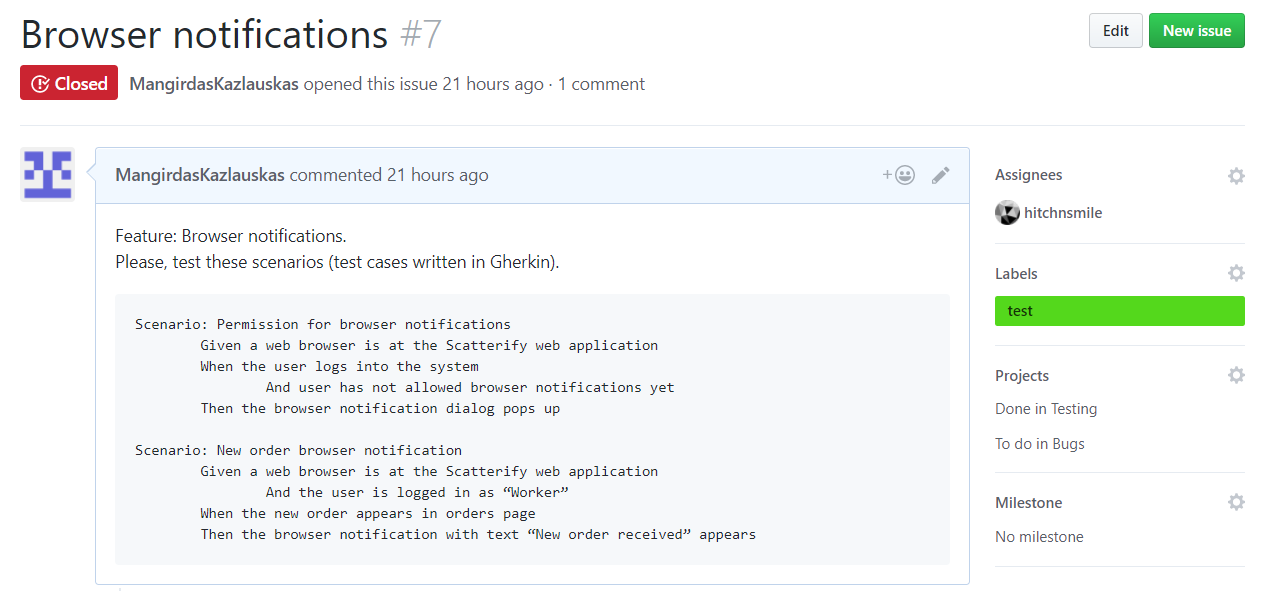
13 pav. Atsijungimo nuo internetinės aplikacijos testavimo užduotis



14 pav. Prekybos vietų testavimo užduotis

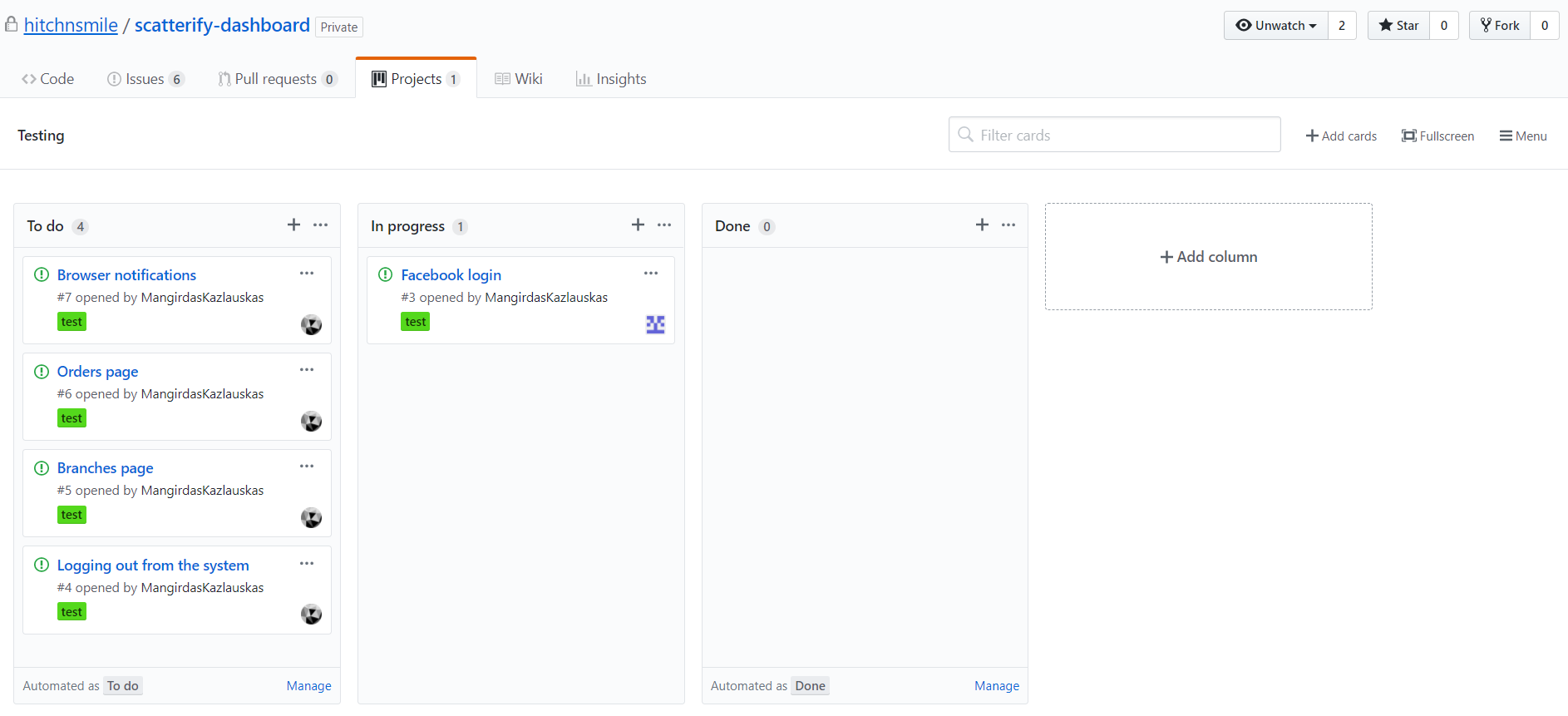


15 pav. Užsakymų testavimo užduotis



16 pav. Internetinės naršyklės pranešimų testavimo užduotis

Taip pat sukurtos testavimo užduotys buvo priskirtos “Testing” projektui atitinkamose repozitorijose bei testavimo eigos kortelėse buvo priskirtos žingsniui “To do” (17 pav.).

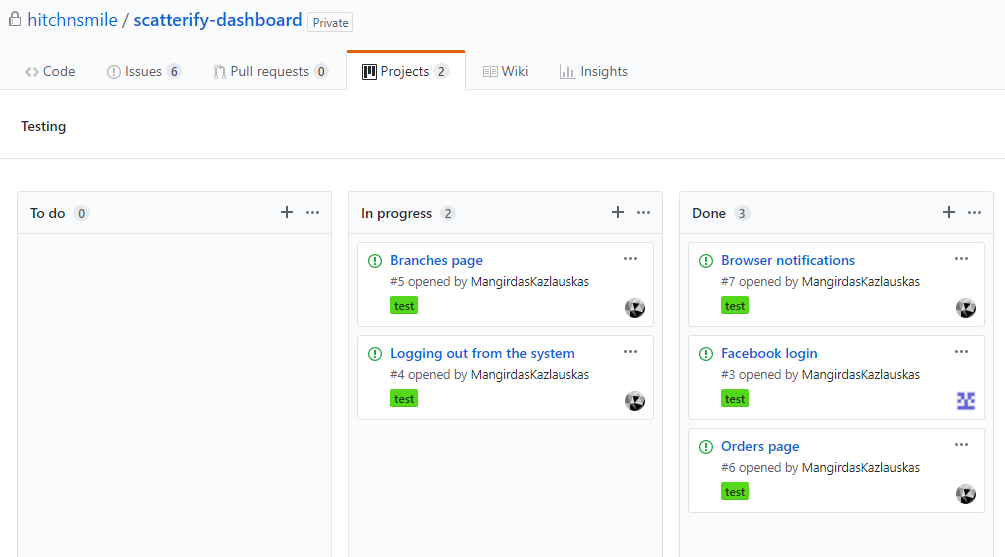


17 pav. Internetinės aplikacijos projekto „Testing“ užduočių eigos langas

# Testavimo užduočių atlikimas

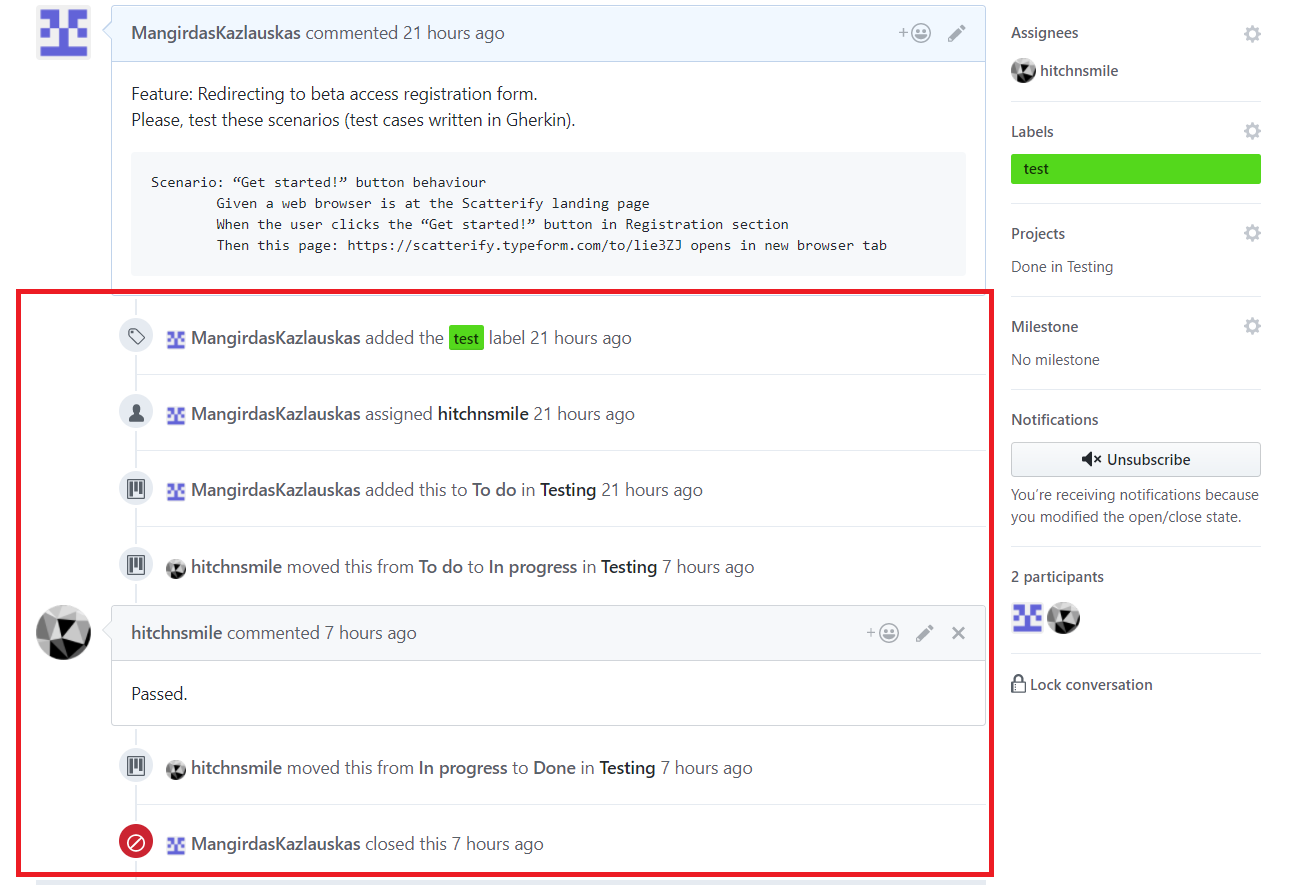
Aprašytos testavimo užduotys buvo atliekamos tokia tvarka:

1. Testuotojas, kuriam priskirta užduotis, atidaro Issues langą bei išsifiltruoja sau priskirtas užduotis;
2. Testuotojas projekto „Testing“ lange norimą atlikti užduotį perkelia į „In progress“ stulpelį;
3. Remdamasis priskirta užduotimi, testuotojas atlieka joje aprašytus testavimo žingsnius;
4. Jei testavimo užduoties atlikimo metu testuotojui iškyla koks klausimas, užduoties komentarų skiltyje jis parašo savo komentarą, kuriame paklausa reikiamo klausimo. Jei viskas gerai – testuotojas komentare parašo, kad testavimo užduotis atlikta sėkmingai (18 pav.);
5. Kai testavimo užduotis baigta, testuotojas ją perkelia į „Done“ stulpelį;
6. Priklausomai nuo situacijos, pats testuotojas arba užduoties kūrėjas pažymi testavimo užduotį kaip „Closed“ – užbaigta ir uždaryta.



18 pav. Testavimo eiga projekto “Testing” lange

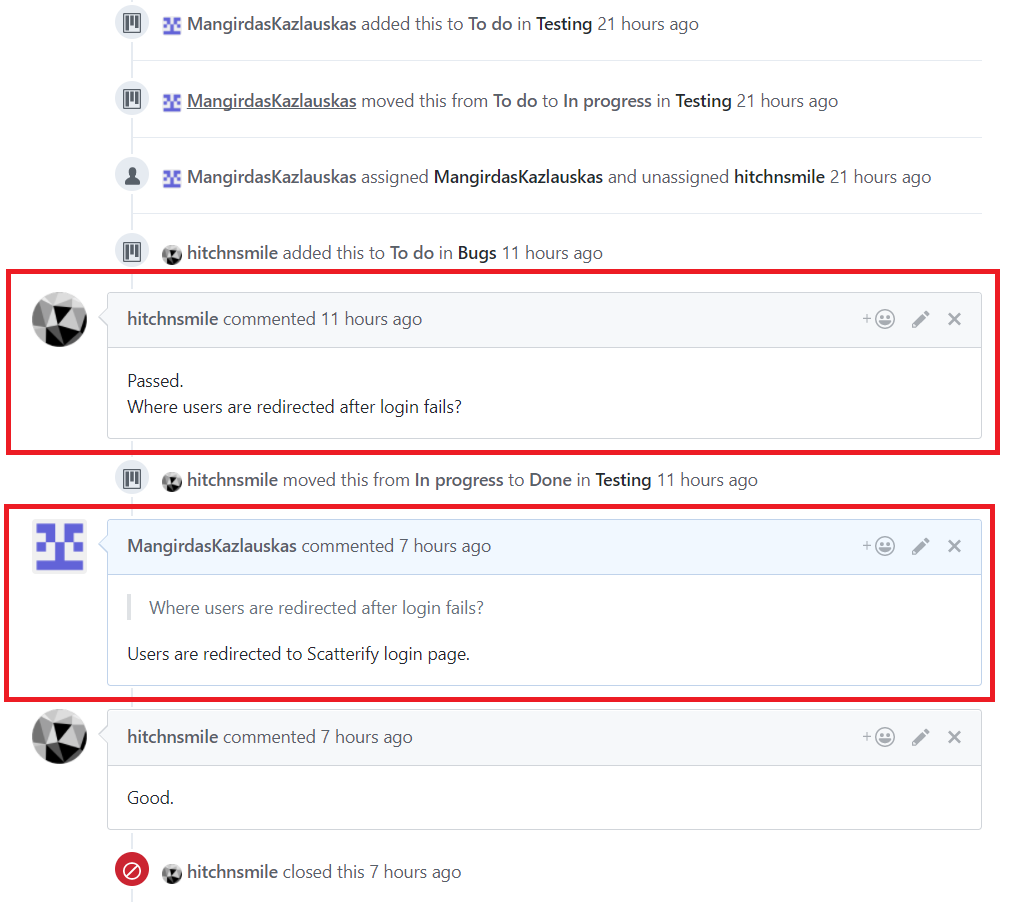
Verta paminėti, kad GitHub aplinkoje kiekvienas atliekamos užduoties žingsnis yra saugomas, todėl galima nesunkiai matyti visą konkrečios užduoties eigą (kokiuose etapuose buvo užduotis, kokie naudotojai atliko konkretų veiksmą ir pan.). Tai yra matoma 19 paveikslėlyje:



19 pav. Naudotojo nukreipimo į išorinę svetainę testavimo užduoties eiga

Kaip matome 19 pav., užduoties eiga atitinka visus anksčiau paminėtus testavimo užduoties atlikimo žingsnius. Tai padeda sekti atliekamos užduoties eigą, taip pat galima kiekvieną užduotį užsiprenumeruoti, taip gaunant pranešimus į el. paštą apie pasikeitusį užduoties statusą – tai padeda sekti ypač svarbių užduočių eigą.

20 paveikslėlyje pateikiamas pavyzdys, kaip testuotojas gali komunikuoti su užduoties kūrėju dėl iškilusių klausimų:

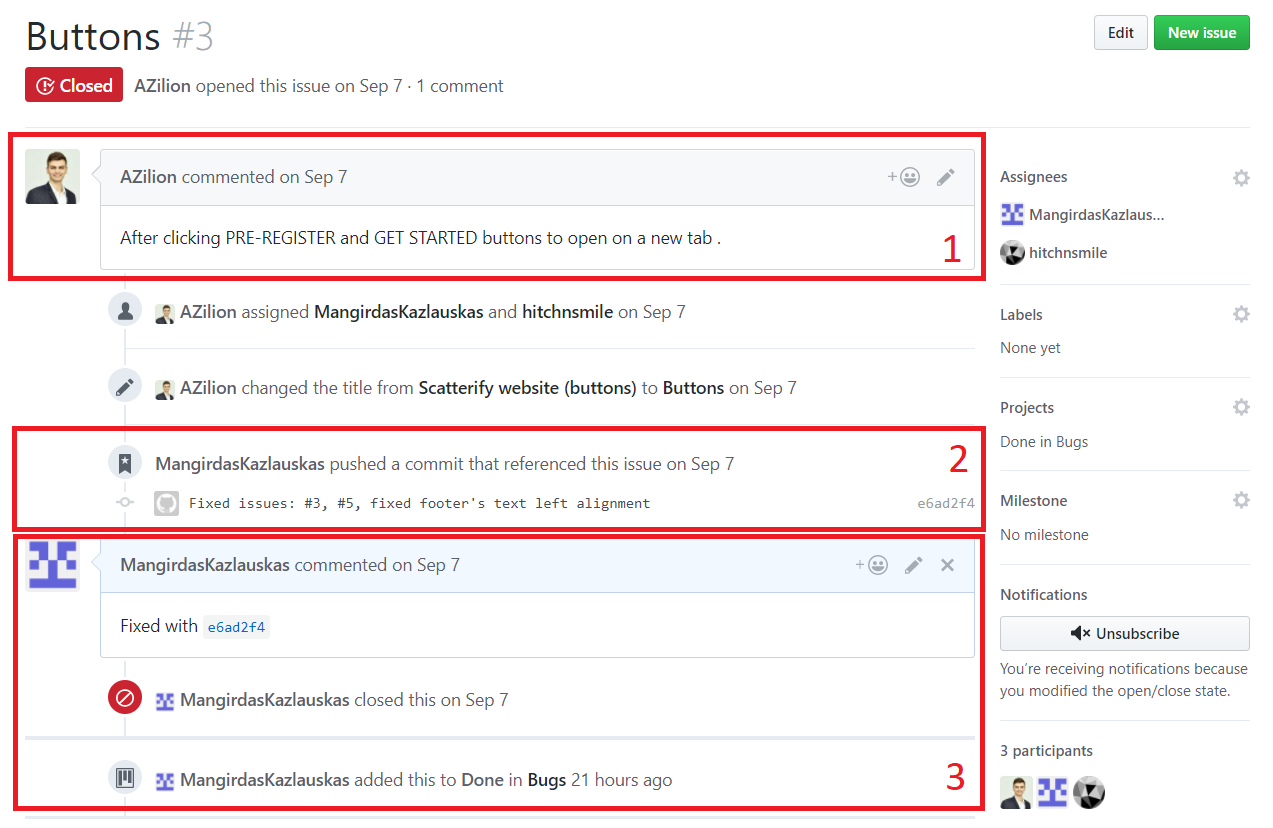


20 pav. Testuotojo ir užduoties kūrėjo komunikacijos pavyzdys

Iš pavyzdžio matome, kad aprašyta testavimo užduotis yra atlikta (pagal aprašymą), tačiau testuotojui iškilo klausimas apie užduotyje nenumatytą variantą – kas turi nutikti, jei prisijungimas prie sistemos bus nesėkmingas. Užduoties kūrėjas atsako į užduotą klausimą ir, šiuo atveju, testuotojas, sulaukęs atsakymo, pats pakeičia testavimo užduoties statusą į užbaigtą ir uždarytą.

# Klaidų ir jų ištaisymo registravimas

Taip pat GitHub sistemoje yra registruojamos sistemos klaidos, kurios buvo pastebėtos naudojantis sistema. Taip pat klaidos gali būti ir „patobulinimo“ pobūdžio, kai iš esmės priskirta užduotis nėra klaida, tačiau būtų geresnis kitoks dizaino ar sistemos logikos sprendimas, kuris pagerintų sistemos kokybę. Būtent tokios užduoties pavyzdys pateikiamas 21 paveikslėlyje:



21 pav. Priskirtos užduoties atlikimo pavyzdys

Kaip matome 21 paveikslėlyje, pirmiausia yra sukuriama užduotis bei paaiškinama, ką reikia padaryti – taip pat kaip ir testavimo užduotyje. Vėliau užduotis yra atliekama ir pataisytas programos kodas yra įkeliamas į repozitoriją – įkeliant failus, komentare yra parašomas užduoties kodas (šiuo atveju #3), pagal kurį GitHub atskiria, jog tam tikras kodo pataisymas yra susijęs su užregistruota užduotimi, todėl užduoties atlikimo eigoje atsiranda laukelis, kad į repozitoriją yra įkeltas galimas klaidos pataisymas ar su užduoties atlikimu susijęs kodas. Vėliau programuotojas parašo, kad užduotis yra atlikta, nurodydamas konkretaus kodo versijos numerį. Paspaudus ant šio numerio, galima matyti, kokie failai ar kokios konkrečios kodo vietos buvo pakoreguotos, sprendžiant vieną ar kitą užduotį (22 pav.).



22 pav. Pakeistos kodo vietos

Kaip matome iš 22 pav., su šia užduotimi susijęs kodas yra 100 eilutėje, kurioje parodoma, kas buvo pridėta *target=“\_blank“,* kas buvo reikalinga sukurtai užduočiai išpildyti.

Verta paminėti, kad GitHub sistemoje yra saugomos visos kodo versijos, todėl bet kuriuo momentu galima atsekti, koks kodas buvo tam tikru metu ar kas buvo pakeista po vienos ar kitos užduoties atlikimo – tai yra reikalinga, kai norima grąžinti kodo pakeitimus į pirminę būseną, ar norint analizuoti, kokios kodo vietos buvo keičiamos daugiausiai.

# Išvados

1. Programinės įrangos kūrimo proceso valdymui buvo surastas šiai užduočiai atlikti tinkamas įrankis GitHub;
2. GitHub buvo naudojamas programinės įrangos kūrimo bei testavimo procesuose, taip procesus padarant lengviau valdomais, detalesniais bei aiškesniais;
3. Ataskaitoje pateiktos įžvalgos, suformuluotos naudojantis projektų valdymo įrankiu GitHub, naudojimosi rezultatai pateikti laboratorinio darbo ataskaitoje.