Testing the PetSitters platform

www.pets-care-u2srs.ondigitalocean.app

Team: Brigade 40

Final Academic Project on
Software Testing

AIT TRAINING CENTER

+49 30 5201 4182

Introduction:

Team:
Vladimir Romaikin
Denys Parkhomenko
Dana Streltsova
Kseniia Ivanova

The PetSitters platform is designed to search and book pet care services. Users can register, create a profile, find verified performers, leave reviews and manage their orders in their personal account. The site includes a rating system and a user-friendly interface for interaction between pet owners and caregivers.

The cycle of our project

Technology stack used

Test planning and management
- Jira – for sprint planning and task management
- TestLink – for maintaining test cases and managing test documentation

Test automation
- Java – the main programming language for automated tests
- Selenium WebDriver – for testing the UI of a web application
- JUnit – a testing framework for writing and running tests
- Gradle – for managing dependencies and building a project
- IntelliJ IDEA – a development environment for writing code

CI/CD and reporting
- Jenkins – for running automated tests in CI/CD
- Allure – for generating test completion reports

API testing
- Postman – for writing and running API tests
- Newman – for automating API tests and CI/CD integration

Load testing
- Artillery – for API stress testing and load checking

Logging and debugging
- Logback – для логирования событий во время выполнения тестов

Team: Brigade 40

Infographics

www.pets-care-u2srs.ondigitalocean.app

200 test cases

64 negative

136 positive

31 Tests are automated

37 Bag Report

functional tests **40**

boundary values and validation **20**

5 compatibility testing

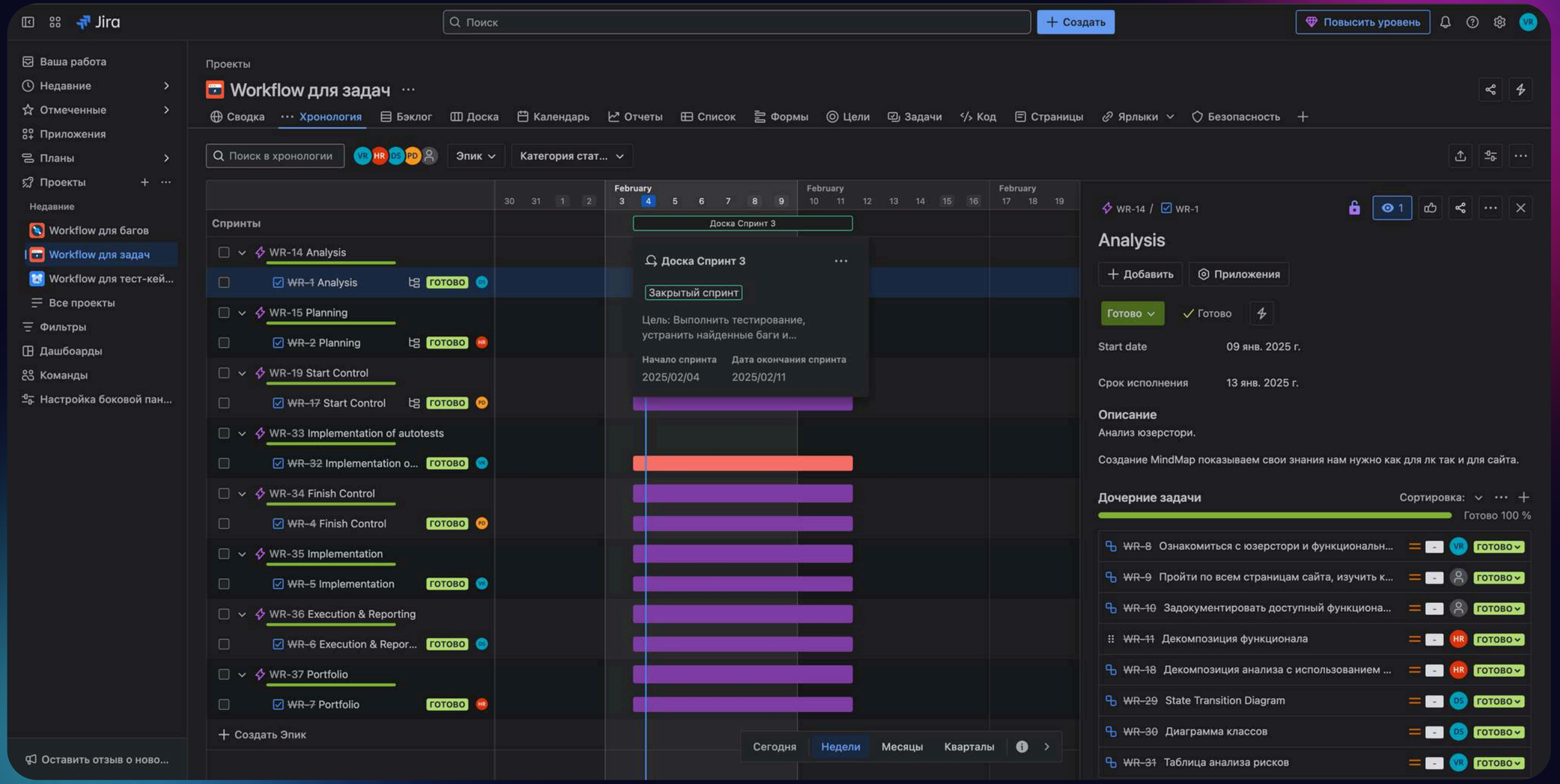
dependency testing: **5**

security testing **15**

UI/UX testing **15**

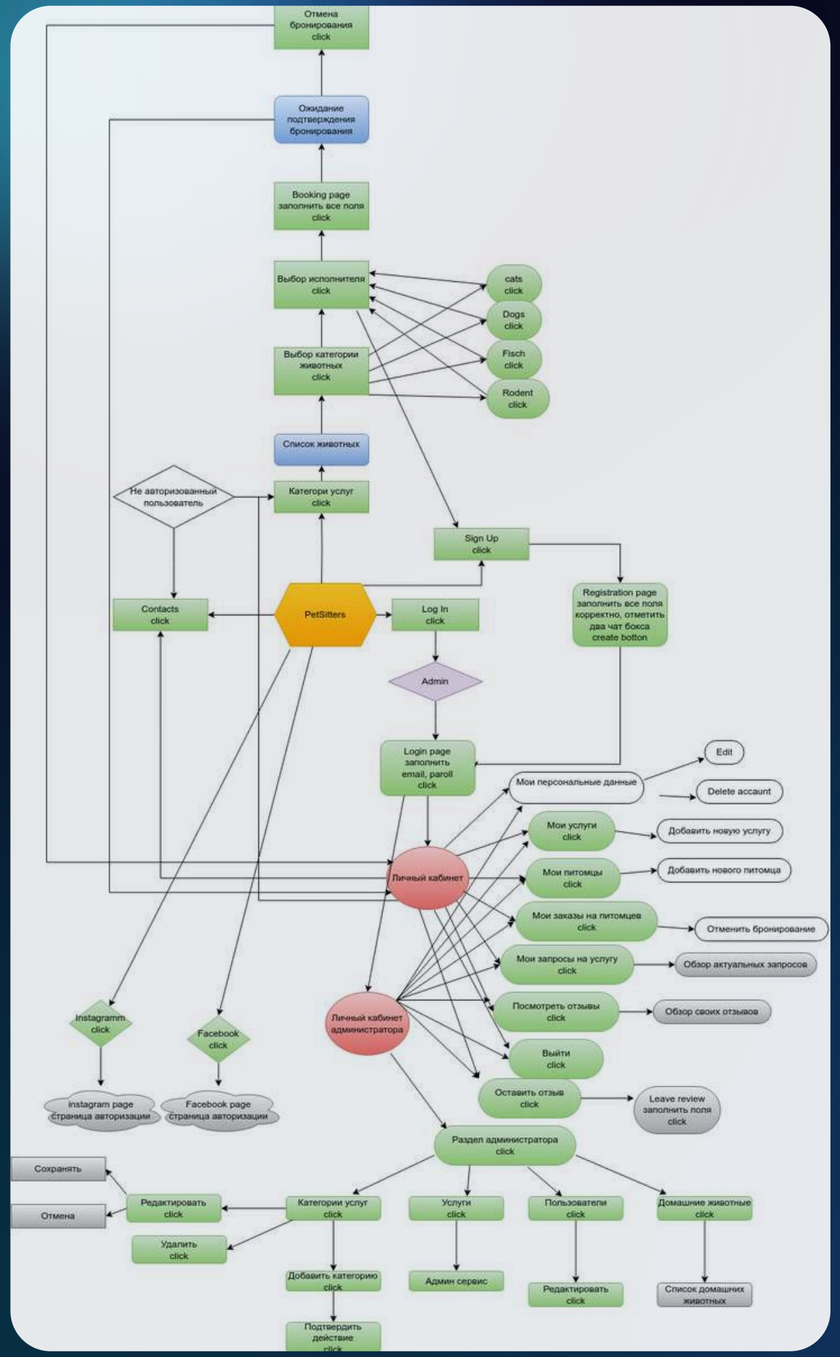
100 API and Load Tests

Jira



First Sprint - Analysis and Planning

Process Modeling of the PetSitters Platform



Purpose of the analysis:

- Defining key user roles and their interactions with the system.
- Determining the platform's functionality.
- Determining key usage scenarios.
- Forming a basis for testing.

Main user roles:

- Unauthorized user:
- Can view the list of services and contacts.
- Cannot book services or manage the profile.
- Authorized user:
- Gets access to the personal account.
- Can manage the profile, bookings, pets and leave reviews.
- Administrator:
- Manages users, service categories and pets.
- Has extended access to the platform's functionality.

Main user roles:

- Registration and authorization: Login to the system with division of roles.
- Booking services: Selecting a performer, waiting for confirmation.
- Profile management: Changing personal data, deleting an account.
- Reviews and ratings: Ability to leave and view reviews.
- Administrative functions: Adding, editing and deleting services.

Conclusions::

- Process modeling helped to identify key testing scenarios.
- The main functional modules of the platform were identified.
- Based on the diagram, a test plan was formed that covers all possible user scenarios.

Download

The purpose of a class diagram is:

- Defining key platform entities and their attributes.
- Defining relationships between system objects.
- Forming the basis for designing the database and application logic.

Conclusions:

- The class diagram defines the key entities of the system and their relationships.
- The entire logic of the platform is reflected, including booking, pet management, services and reviews.
- The main relationships between objects are worked out (one to many, one to one).
- Based on the class diagram, you can design the database structure and API system.

This analysis helped the team understand which objects needed detailed testing and how they interacted in the system.

AIT Hi-tech team

CLASS DIAGRAM	
EXECUTOR	DANA STRELTSOVA
COMPLETION DATE	13 / 01 / 25
VERSION	1.0
WEBSITE	PETS-CARE-U2SRS
PAGES	1 / 1

RELATIONSHIPS BETWEEN CLASSES

1. USER ↔ PETS:
ONE USER CAN OWN MULTIPLE PETS.
RELATIONSHIP: "1 OWNS *".
2. USER ↔ BOOKING:
ONE USER WITH THE ROLE "PET SITTER" CAN HANDLE MULTIPLE BOOKINGS.
RELATIONSHIP: "1 MANAGES *".
3. PETS ↔ BOOKING:
ONE PET CAN BE ASSOCIATED WITH MULTIPLE BOOKINGS.
RELATIONSHIP: "1 IS BOOKED IN *".
4. PETS ↔ CATEGORY:
EACH PET BELONGS TO ONE CATEGORY (E.G. CATS, DOGS).
RELATIONSHIP: "1 BELONGS TO 1".
5. REVIEW ↔ USER:
ONE USER CAN LEAVE MULTIPLE REVIEWS.
RELATIONSHIP: "1 LEAVES *".
6. BOOKING ↔ SERVICES:
ONE BOOKING IS ASSOCIATED WITH ONE SERVICE.
RELATIONSHIP: "1 INCLUDES 1".

"PETS CARE" IS A WEB APPLICATION FOR MANAGING PET CARE. THE SYSTEM ALLOWS USERS (PET OWNERS) TO BOOK SERVICES FOR THEIR PETS, LEAVE REVIEWS OF THE SERVICES PROVIDED, AND MANAGE INFORMATION ABOUT THEIR PETS. ADMINISTRATORS CAN MANAGE PET CATEGORIES, SERVICES, AND USERS. KEY FEATURES INCLUDE BOOKING, MANAGING PET DATA AND REVIEWS, AND NOTIFICATION SUPPORT.

Class diagram

Download

Purpose of the analysis:

- Defining key characteristics of the system taking into account the FURPS model.
- Separation of functional and non-functional requirements.
- Identifying criteria that affect the quality, convenience, performance and security of the platform.

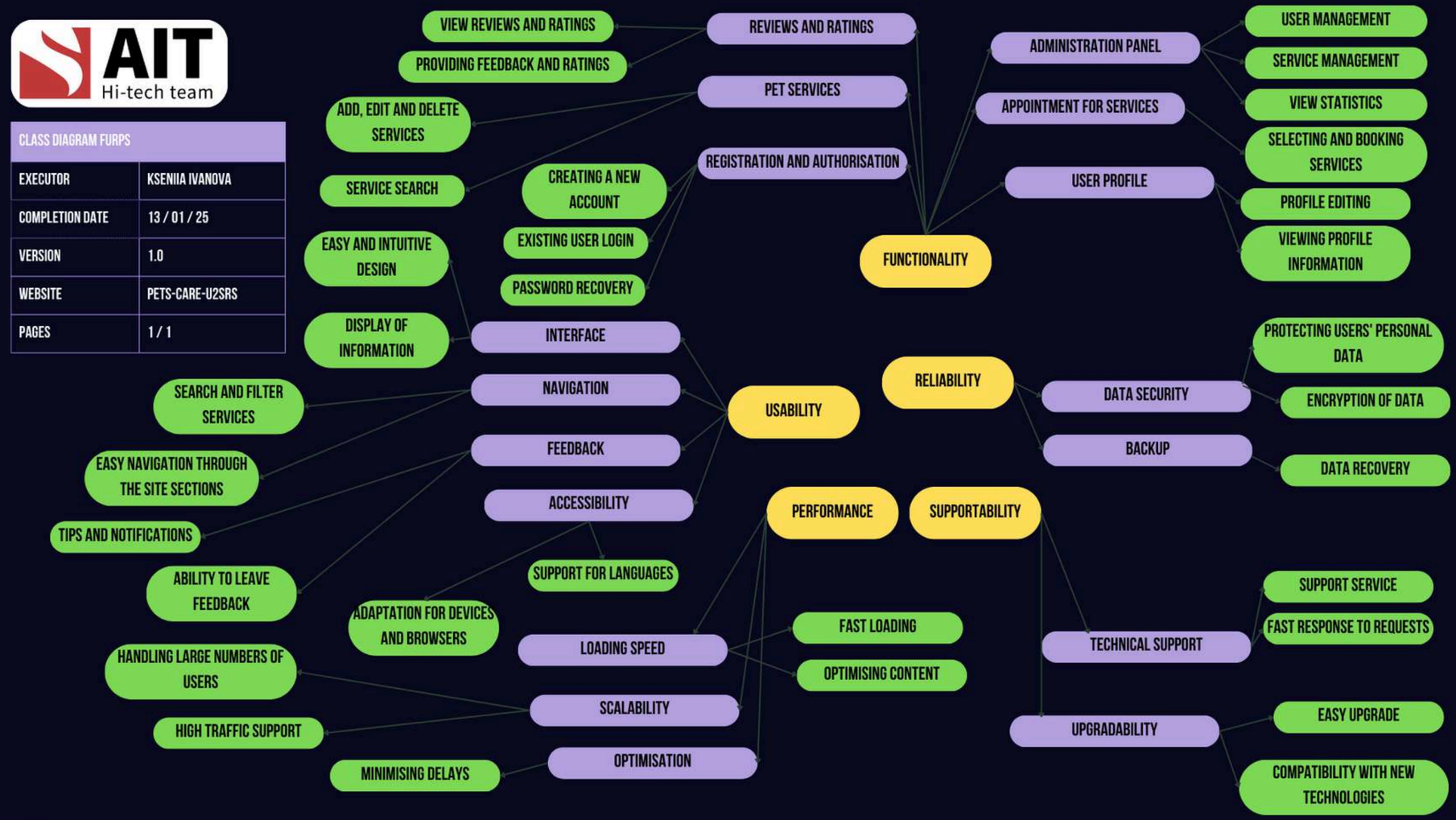
Conclusions:

- FURPS decomposition helped to identify key characteristics of the system.
- Critical aspects were identified: performance, usability, security.
- Directions for testing, improvement and further development were identified.
- This analysis was used to create a test plan and develop a testing strategy.

🔹 F – Functionality: basic system capabilities (authorization, booking, service management).
🔹 U – Usability: intuitive interface, navigation, adaptation to devices.
🔹 R – Reliability: data protection, backup, failure resistance.
🔹 P – Performance: speed, scalability, load optimization.
🔹 S – Supportability: updateability, compatibility, technical support.

AIT Hi-tech team

CLASS DIAGRAM FURPS	
EXECUTOR	KSENIIA IVANOVA
COMPLETION DATE	13 / 01 / 25
VERSION	1.0
WEBSITE	PETS-CARE-U2SRS
PAGES	1 / 1



Decomposition Analysis Using FURPS

Download

Download

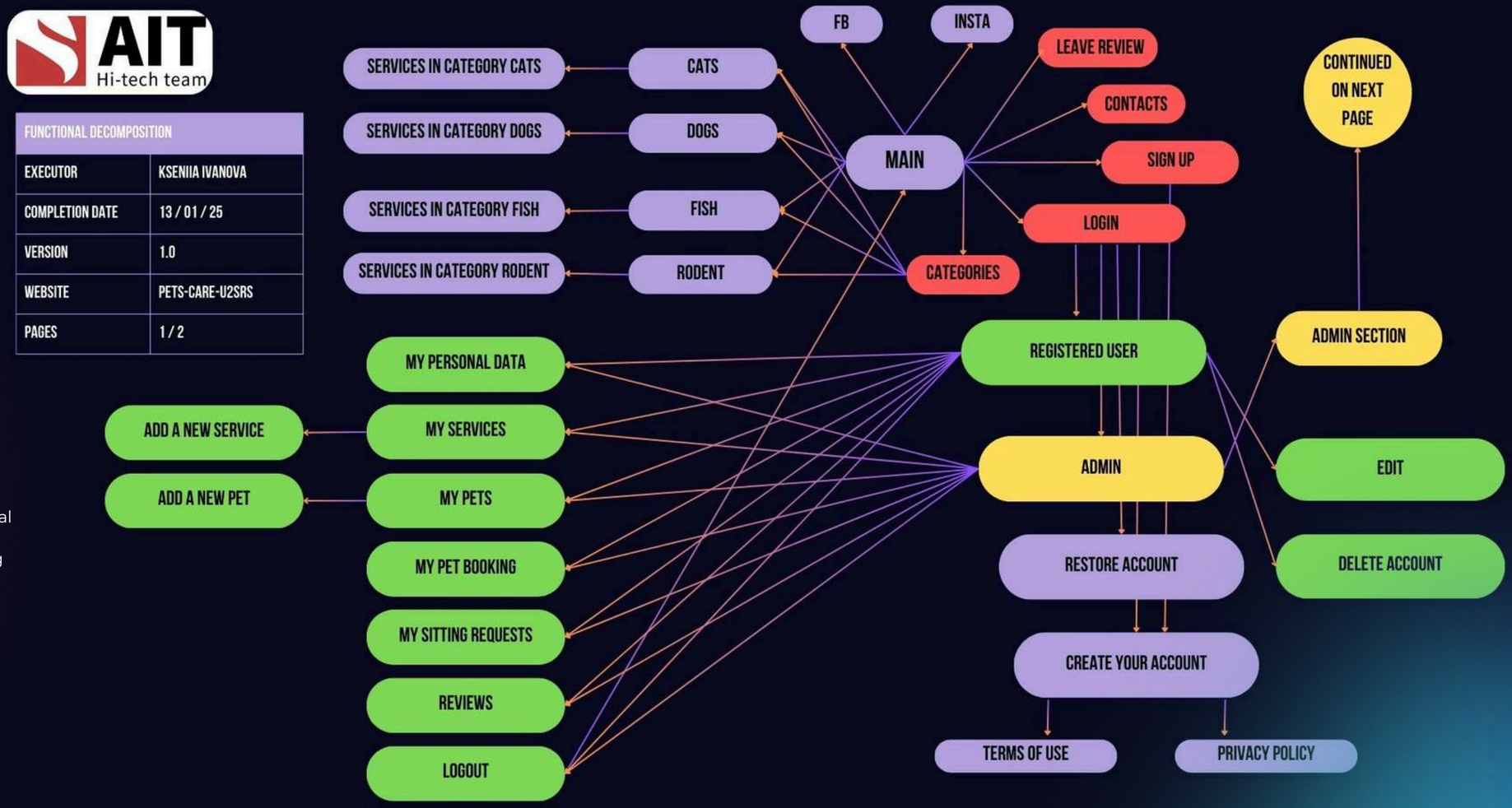
FUNCTIONAL DECOMPOSITION	
EXECUTOR	KSENIIA IVANOVA
COMPLETION DATE	13 / 01 / 25
VERSION	1.0
WEBSITE	PETS-CARE-U2SRS
PAGES	1 / 2

Purpose of the analysis:

- Dividing the system into main functional modules.
- Determining the relationships between components.
- Formation of a testing structure based on key functional areas.

Conclusions:

- Decomposition allowed us to identify the key functional blocks of the system.
- Critical areas requiring special attention during testing were identified.
- A test plan structure was created that covered all aspects of the platform operation.
- This analysis was used to build a testing strategy and verify the correctness of the platform operation.

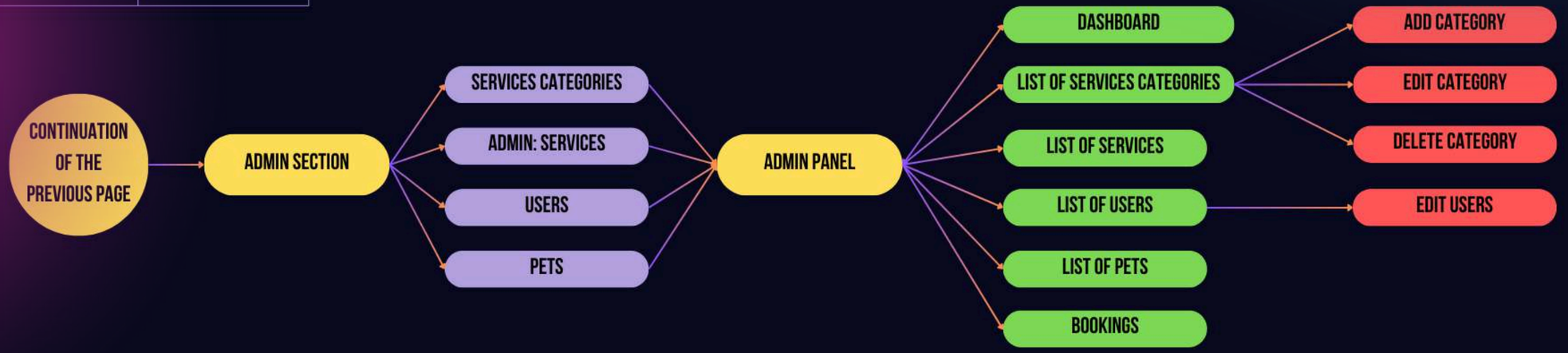


Decomposition of the functionality of the PetSitters platform

Continuation

AIT Hi-tech team

FUNCTIONAL DECOMPOSITION (ADMIN PANEL)	
EXECUTOR	KSENIIA IVANOVA
COMPLETION DATE	13 / 01 / 25
VERSION	1.0
WEBSITE	PETS-CARE-U2SRS
PAGES	2 / 2



Decomposition of the functionality of the PetSitters platform

Risk Analysis of the PetSitters Project

Purpose of the analysis:

- Identifying potential risks during platform testing.
- Assessing the likelihood and impact of each risk.
- Determining risk mitigation measures.

Conclusions:

- Risk analysis has identified key threats in the testing process.
- Measures have been developed to reduce the likelihood of problems occurring.
- This analysis is used to control quality and improve testing efficiency.

ID риска	Название риска	Описание	Вероятность	Влияние	Приоритет	Меры минимизации	Ответственный
R001	Недопонимание требований	Некоторые аспекты функциональности сайта могут быть неясными из юзерстори или спецификации.	Средняя	Среднее	Средний	Повторное изучение юзерстори, консультация с преподавателями, уточняющие вопросы.	Участники проекта
R002	Ограниченное время	Сжатые сроки спринтов могут привести к неполной отработке навыков тестирования.	Высокая	Среднее	Высокий	Приоритизация заданий, фокус на критическом функционале.	Лидер команды
R003	Ошибки в тест-кейсах	Тест-кейсы могут быть некорректно сформулированы или не покрывать весь функционал.	Средняя	Высокое	Высокий	Проведение ревью тест-кейсов внутри команды, использование шаблонов и чек-листов.	Участники проекта
R004	Ограниченный доступ к среде	Возможны сбои или трудности с доступом к сайту для тестирования.	Низкая	Низкое	Низкий	Убедиться в доступности всего функционала сайта заранее, использовать альтернативные среды при необходимости.	Участники проекта
R005	Нехватка опыта команды	Неопытность в использовании инструментов тестирования может замедлить процесс.	Средняя	Среднее	Средний	Практическое изучение инструментов (Postman, Selenium), привлечение преподавателей.	Участники проекта
R006	Сложности с интеграцией инструментов	Проблемы при настройке инструментов автоматизации, таких как Selenium или TestLink.	Средняя	Высокое	Высокий	Заранее подготовить инструкции по настройке, привлечение специалистов для помощи.	Участники проекта
R007	Неполное покрытие функционала	Не весь функционал сайта может быть охвачен тестами из-за ограниченного времени или пропусков.	Высокая	Высокое	Критический	Составление чек-листов для быстрого покрытия основных функций, проведение smoke-тестов.	QA Lead
R008	Пропуск багов	Ошибки в тестировании могут привести к пропущенным багам.	Средняя	Высокое	Высокий	Подготовка четкой структуры тест-кейсов, привлечение команды для совместного выполнения тестов.	QA команда

Risk Analysis of the PetSitters Project

Continuation

ID риска	Название риска	Описание	Вероятность	Влияние	Приоритет	Меры минимизации	Ответственный
R009	Недостаток обратной связи	Преподаватели могут не успеть вовремя предоставить обратную связь по тестированию.	Низкая	Среднее	Средний	Регулярное обращение за консультацией, запросы по конкретным вопросам.	Участники проекта
R010	Проблемы с пониманием API	Недостаток знаний в области тестирования API может привести к пропуску ошибок.	Средняя	Высокое	Высокий	Проведение учебных сессий по Postman и RestAssured, использование документации API.	Участники проекта
R011	Неполная документация проекта	В проекте могут отсутствовать описания всех возможностей сайта или структуры данных.	Высокая	Высокое	Критический	Использование имеющихся данных для восстановления структуры, запросы уточнений у преподавателей.	QA команда
R012	Проблемы с разными браузерами	Сайт может некорректно отображаться в некоторых браузерах.	Средняя	Среднее	Средний	Проверка на основных браузерах (Chrome, Firefox, Edge).	QA команда
R013	Ошибки в отчетности	Некорректная подготовка баг-репортов может привести к недостаточной детализации проблем.	Средняя	Среднее	Средний	Проведение ревью баг-репортов, использование шаблонов отчетов.	QA Lead

Download

Test plan Pet Service

Using Page Object Model (POM):

Link to Test plan

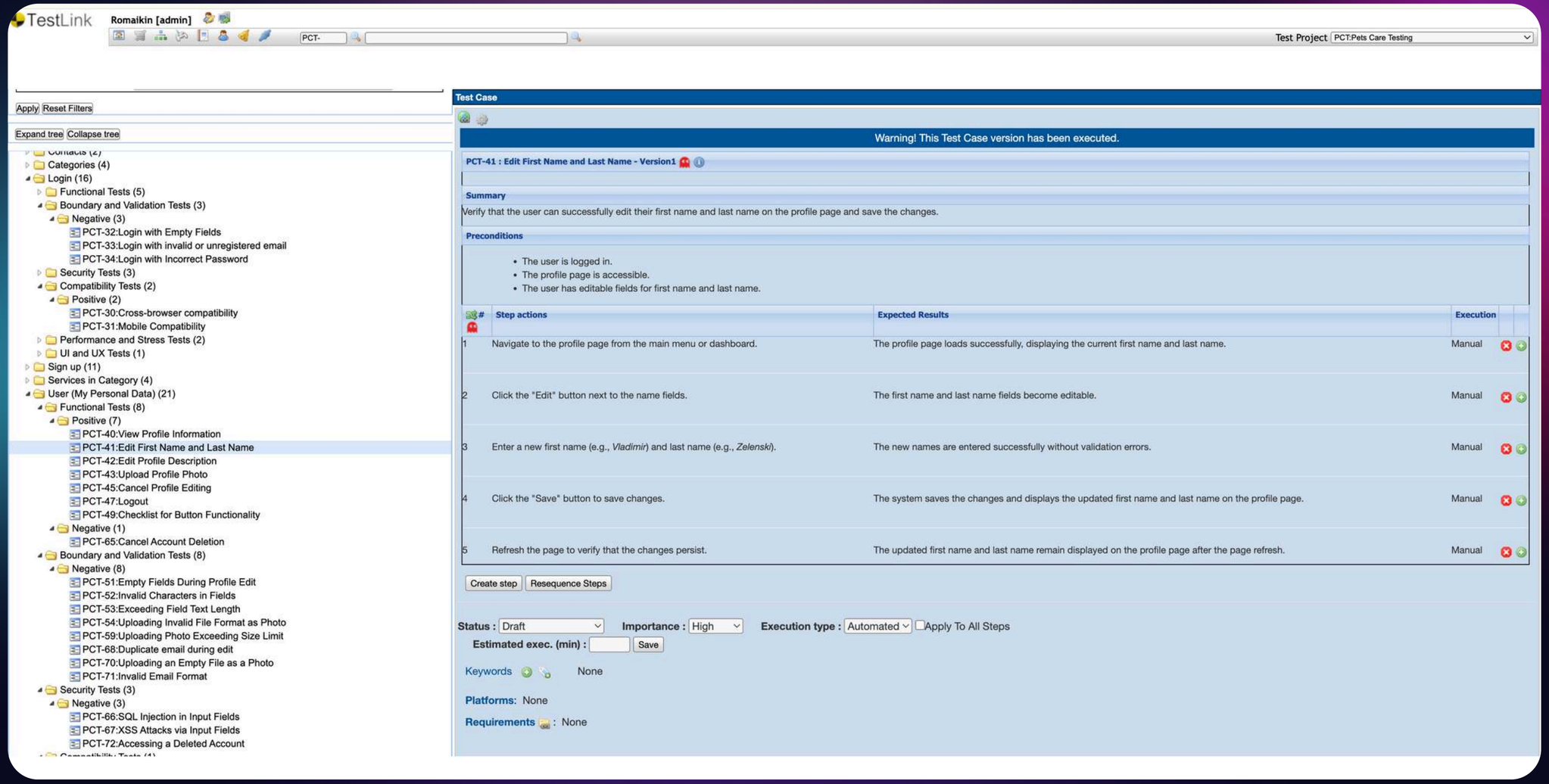
Main characteristics of the test plan
Version: 2.0
Creation date: 13.01.2025
Team: Brigade 40
Plan level: Master – covers functional, integration, system and acceptance testing.

Testing objectives:
Ensuring quality and compliance of the system
Checking functionality, security, performance and compatibility
Determining the testing strategy and criteria for success

Test coverage:
Functional tests: checking registration, booking, service management.
Boundary and validation tests: system operation with erroneous inputs.
UI/UX testing: user-friendliness of interaction.
Security testing: protection against SQL injections, XSS attacks.
Load testing: operation under high loads.
API testing: correct operation of API endpoints.

Resolution:
PetSitters test plan covers functional, UI, API and load testing using Page Object Model (POM).
Methods: automated and manual testing, security and performance testing.
Tools: Selenium, Postman, Artillery, Jenkins, TestLink.
Success criteria: at least 90% successful tests, no critical defects, stable operation of key functions.
Goal: ensure stability, security and compliance before release.

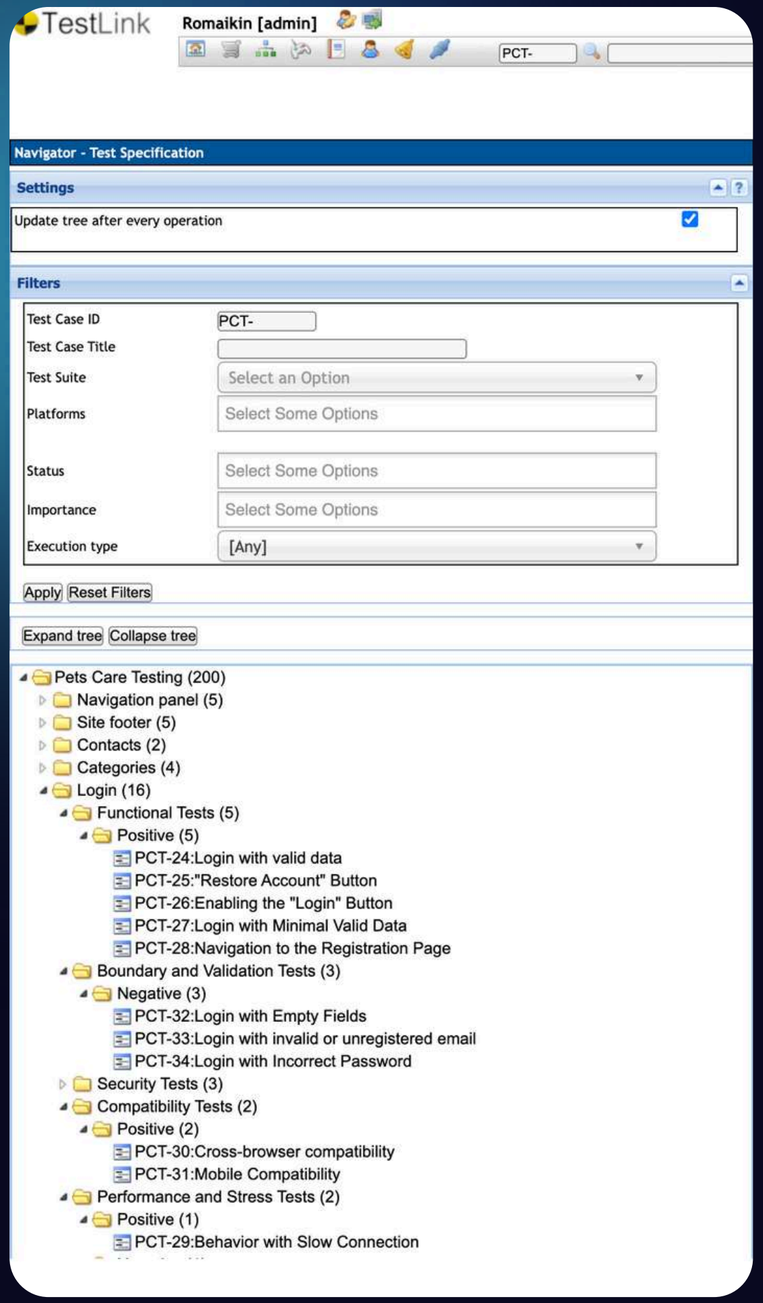
TestLink



All 200 test cases are designed in TestLink

TestLink Analytics

Distribution by platforms / structure and test results



Test Project : **Pets Care Testing**
Test Plan : **Test Plan - Pets Care Testing**

Important Notice

Platforms has been defined for this test plan.
Use of platforms has impact on metrics, because
a test case that must be executed in N platforms is considered as N test cases on metrics.
Example: if user U1 has been assigned execution of Test Case TC1
on platform X and Y, then user U1 has TWO test cases to execute NOT ONE

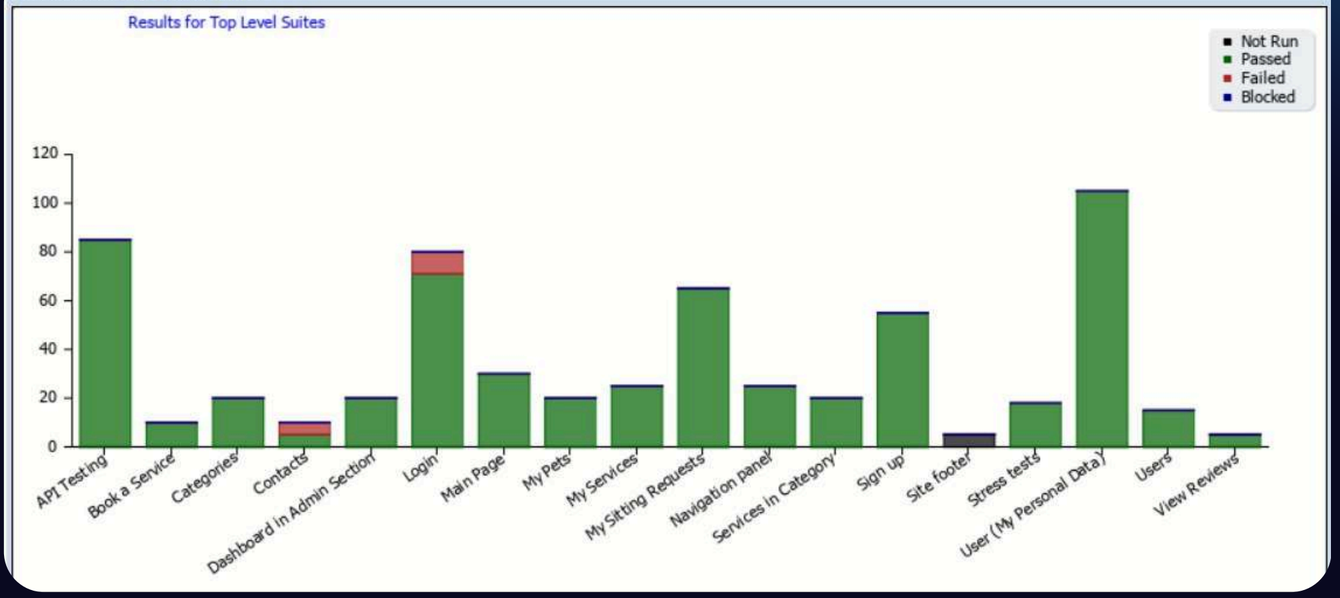
Results by Platform

Platform	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Artillery	6	0	0.0	6	100.0	0	0.0	0	0.0	100.0
Mac Chrome	101	0	0.0	98	97.0	3	3.0	0	0.0	100.0
Mac Safari	101	0	0.0	99	98.0	2	2.0	0	0.0	100.0
Newman	6	0	0.0	6	100.0	0	0.0	0	0.0	100.0
Postman	96	5	5.2	91	94.8	0	0.0	0	0.0	94.8
Windows 10 Chrome	101	0	0.0	98	97.0	3	3.0	0	0.0	100.0
Windows 10 Edge	101	0	0.0	98	97.0	3	3.0	0	0.0	100.0
Windows 10 Firefox	101	0	0.0	98	97.0	3	3.0	0	0.0	100.0

This report shows the result for each platform linked to this test plan.

Overall Build Status

Results for Top Level Suites



TestLink

General information about testing
Testing Project: Pets Care Testing
Test Plan: Pets Care Testing
Total Test Cases: 200
Executed Tests: 100% (all run)
Average Test Success Rate: 97-100% depending on platform
Main Testing Platforms:
Artillery
Mac (Chrome, Safari)
Windows 10 (Chrome, Edge, Firefox)
Newman, Postman

General information about testing
Results by platform

- Artillery: 6 tests, 100% successful
- Mac Chrome: 101 tests, 97% successful, 3% unsuccessful
- Mac Safari: 101 tests, 98% successful, 2% unsuccessful
- Windows 10 Chrome, Edge, Firefox: 101 tests, 97% successful, 3% unsuccessful
- Postman: 96 tests, 94.8% successful

Conclusion:

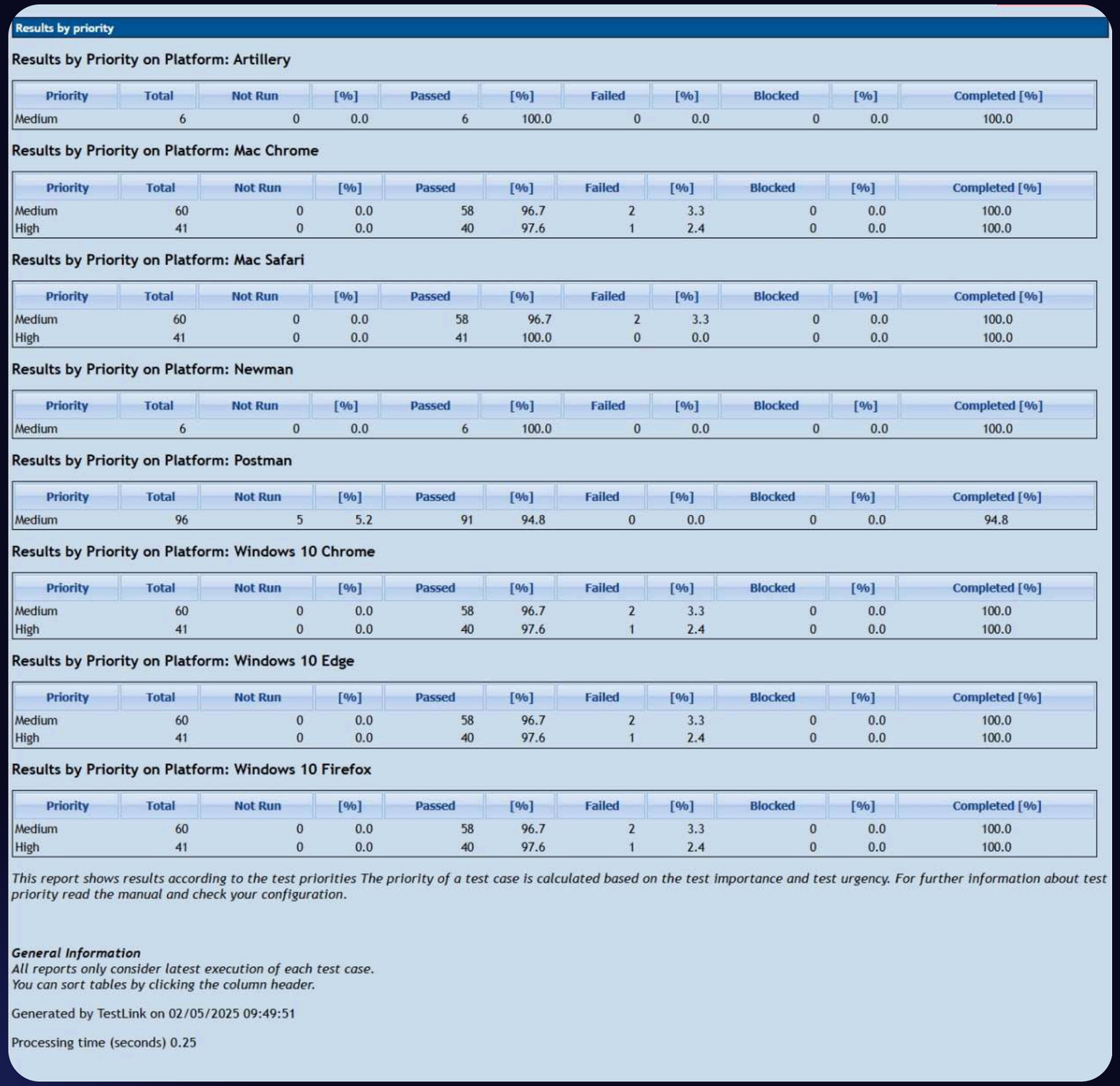
- On all platforms, tests were run without blocking.
- A small number of failed tests (2-5%) may indicate platform-dependent errors or environment instability.
- The highest success rate was on Mac Safari (98%), and the lowest on Postman and newman (94.8%).

Results by test categories

- The largest number of tests were carried out for:
  - API Testing (85 tests)
  - Login (16 tests)
  - User Data (21 tests)
  - Sign Up (11 tests)
- Tests with the highest number of failed checks:
  - Login (several validation errors)
  - Postman tests (lowest success rate 94.8%)

Conclusion:

- API testing was carried out on a large scale (85 cases), which ensured a high level of coverage.
- Login validation requires additional verification, as there are unsuccessful tests.
- Registration (Sign Up) and work with user data (User Data) were successful.



Results by priority

Results by Priority on Platform: Artillery

Priority	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Medium	6	0	0.0	6	100.0	0	0.0	0	0.0	100.0

Results by Priority on Platform: Mac Chrome

Priority	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Medium	60	0	0.0	58	96.7	2	3.3	0	0.0	100.0
High	41	0	0.0	40	97.6	1	2.4	0	0.0	100.0

Results by Priority on Platform: Mac Safari

Priority	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Medium	60	0	0.0	58	96.7	2	3.3	0	0.0	100.0
High	41	0	0.0	41	100.0	0	0.0	0	0.0	100.0

Results by Priority on Platform: Newman

Priority	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Medium	6	0	0.0	6	100.0	0	0.0	0	0.0	100.0

Results by Priority on Platform: Postman

Priority	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Medium	96	5	5.2	91	94.8	0	0.0	0	0.0	94.8

Results by Priority on Platform: Windows 10 Chrome

Priority	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Medium	60	0	0.0	58	96.7	2	3.3	0	0.0	100.0
High	41	0	0.0	40	97.6	1	2.4	0	0.0	100.0

Results by Priority on Platform: Windows 10 Edge

Priority	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Medium	60	0	0.0	58	96.7	2	3.3	0	0.0	100.0
High	41	0	0.0	40	97.6	1	2.4	0	0.0	100.0

Results by Priority on Platform: Windows 10 Firefox

Priority	Total	Not Run	[%]	Passed	[%]	Failed	[%]	Blocked	[%]	Completed [%]
Medium	60	0	0.0	58	96.7	2	3.3	0	0.0	100.0
High	41	0	0.0	40	97.6	1	2.4	0	0.0	100.0

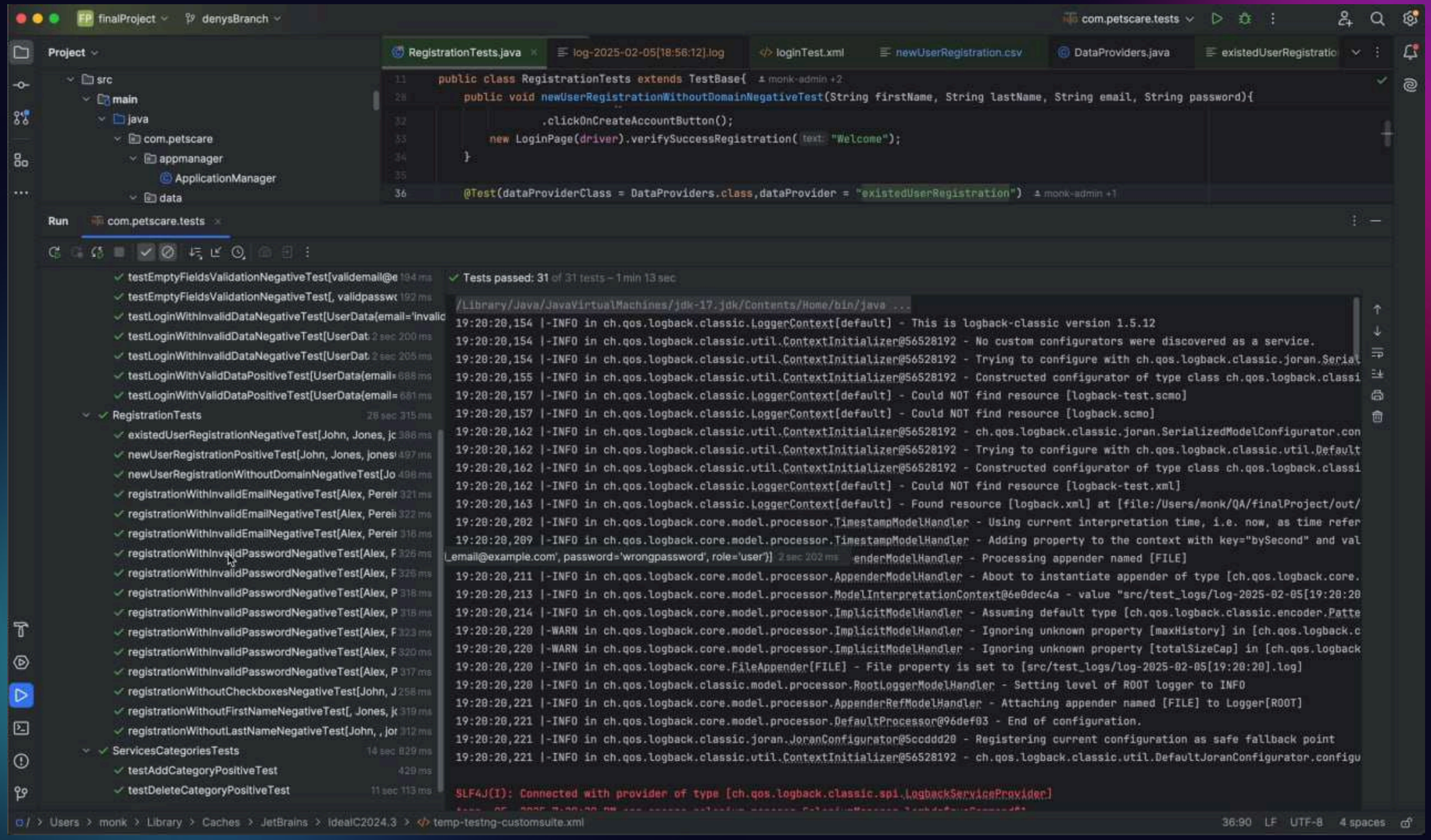
This report shows results according to the test priorities The priority of a test case is calculated based on the test importance and test urgency. For further information about test priority read the manual and check your configuration.

General Information
All reports only consider latest execution of each test case.
You can sort tables by clicking the column header.

Generated by TestLink on 02/05/2025 09:49:51

Processing time (seconds) 0.25

IntelliJ IDEA (Click for video)

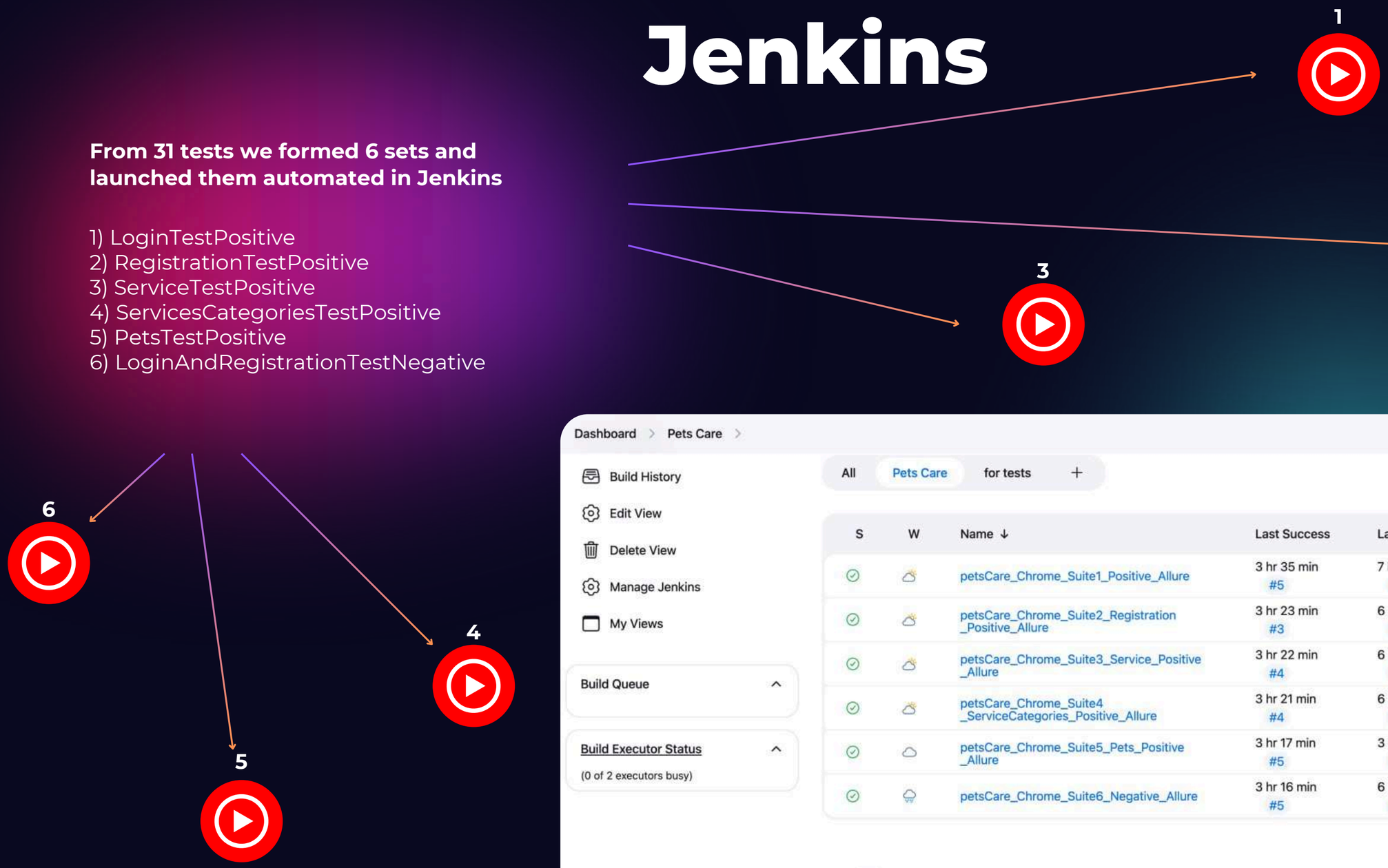


We have automated 31 tests

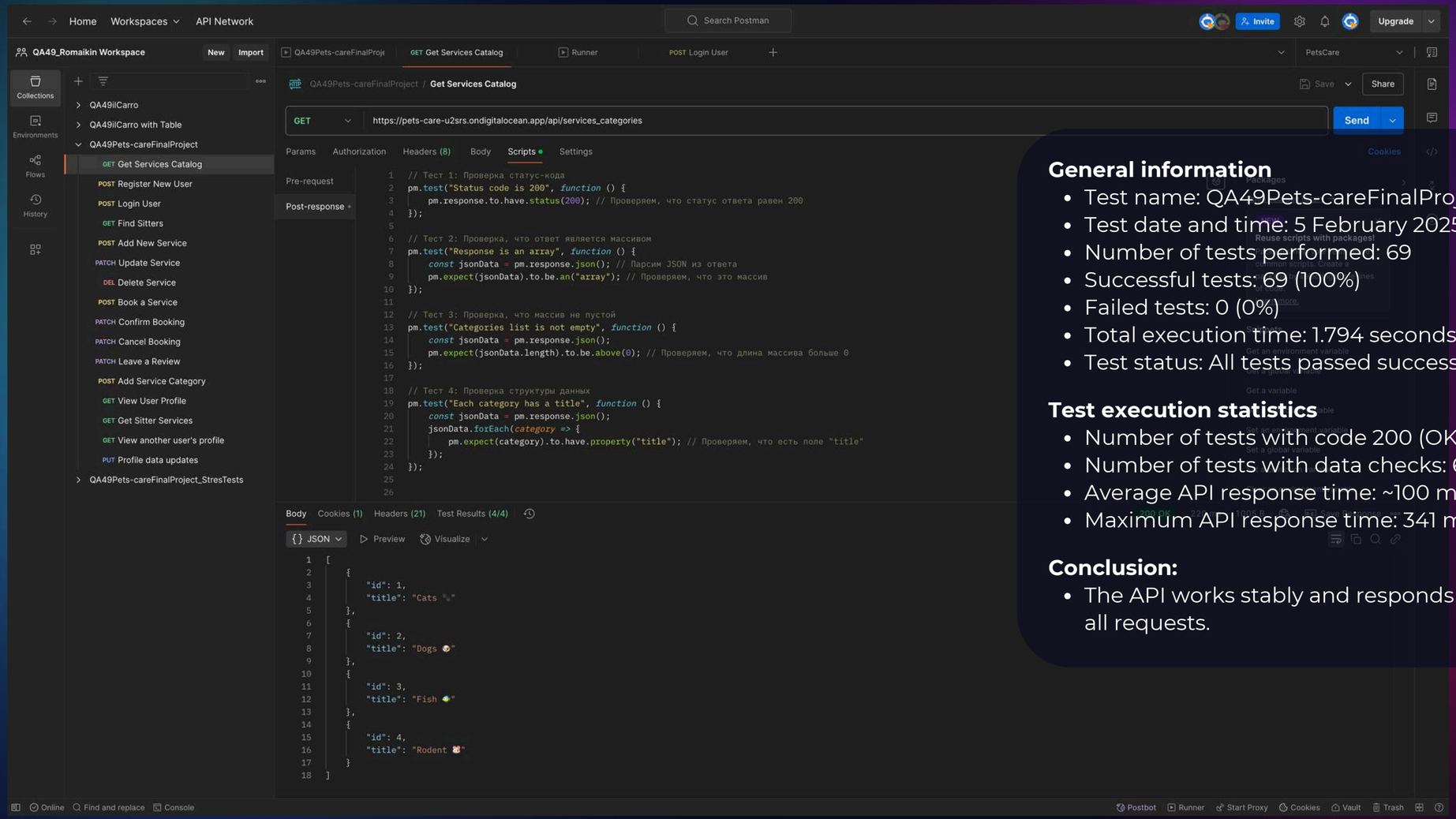
Jenkins

From 31 tests we formed 6 sets and launched them automated in Jenkins

1) LoginTestPositive
2) RegistrationTestPositive
3) ServiceTestPositive
4) ServicesCategoriesTestPositive
5) PetsTestPositive
6) LoginAndRegistrationTestNegative



Postman



General information

- Test name: QA49Pets-careFinalProject
- Test date and time: 5 February 2025, 08:01 UTC
- Number of tests performed: 69
- Successful tests: 69 (100%)
- Failed tests: 0 (0%)
- Total execution time: 1.794 seconds
- Test status: All tests passed successfully

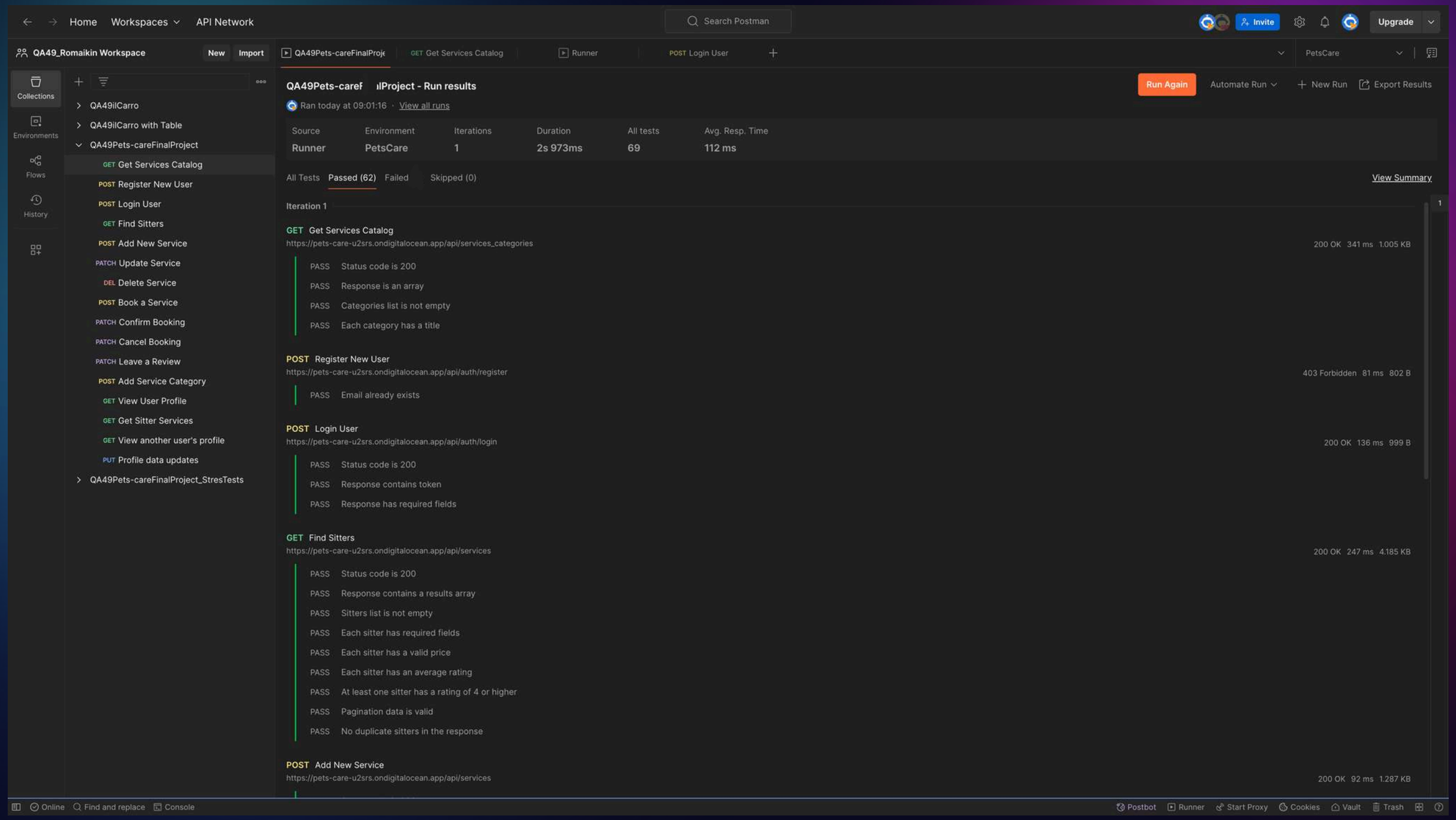
Test execution statistics

- Number of tests with code 200 (OK): 16
- Number of tests with data checks: 69
- Average API response time: ~100 ms
- Maximum API response time: 341 ms

Conclusion:

- The API works stably and responds correctly to all requests.

Postman


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
QA49_Romaikin Workspace
New
Import
QA49Pets-careFinalProje
GET Get Services Catalog
Runner
POST Login User
PetsCare
Collections
Environments
Flows
History
QA49IlCarro
QA49IlCarro with Table
QA49Pets-careFinalProject
GET Get Services Catalog
POST Register New User
POST Login User
GET Find Sitters
POST Add New Service
PATCH Update Service
DEL Delete Service
POST Book a Service
PATCH Confirm Booking
PATCH Cancel Booking
PATCH Leave a Review
POST Add Service Category
GET View User Profile
GET Get Sitter Services
GET View another user's profile
PUT Profile data updates
QA49Pets-careFinalProject_StresTests
QA49Pets-careF IlProject - Run results
Run Again
Automate Run
New Run
Export Results
Ran today at 09:01:16 · View all runs
Source Runner
Environment PetsCare
Iterations 1
Duration 2s 973ms
All tests 69
Avg. Resp. Time 112 ms
All Tests
Passed (62)
Failed
Skipped (0)
View Summary
Iteration 1
GET Get Services Catalog
https://pets-care-u2srs.ondigitalocean.app/api/services_categories
200 OK 341 ms 1.005 KB
PASS Status code is 200
PASS Response is an array
PASS Categories list is not empty
PASS Each category has a title
POST Register New User
https://pets-care-u2srs.ondigitalocean.app/api/auth/register
403 Forbidden 81 ms 802 B
PASS Email already exists
POST Login User
https://pets-care-u2srs.ondigitalocean.app/api/auth/login
200 OK 136 ms 999 B
PASS Status code is 200
PASS Response contains token
PASS Response has required fields
GET Find Sitters
https://pets-care-u2srs.ondigitalocean.app/api/services
200 OK 247 ms 4.185 KB
PASS Status code is 200
PASS Response contains a results array
PASS Sitters list is not empty
PASS Each sitter has required fields
PASS Each sitter has a valid price
PASS Each sitter has an average rating
PASS At least one sitter has a rating of 4 or higher
PASS Pagination data is valid
PASS No duplicate sitters in the response
POST Add New Service
https://pets-care-u2srs.ondigitalocean.app/api/services
200 OK 92 ms 1.287 KB
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash

Load and stress testing

Load tests performed in newman

```
StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 37ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 95/100

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 38ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 96/100

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 37ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 97/100

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 35ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 98/100

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 38ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 99/100

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 35ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 100/100

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 34ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title
```

	executed	failed
iterations	100	0
requests	100	0
test-scripts	200	0
prerequest-scripts	100	0
assertions	400	0
total run duration: 5.6s		
total data received: 18kB (approx)		
average response time: 38ms [min: 33ms, max: 186ms, s.d.: 15ms]		

```
vladimirromajkin@iMac-Vladimir Test %
```

100 consecutive requests

```
StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 36ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 995/1000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 36ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 996/1000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 34ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 997/1000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 37ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 998/1000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 36ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 999/1000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 40ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 1000/1000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 39ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title
```

	executed	failed
iterations	1000	0
requests	1000	0
test-scripts	2000	0
prerequest-scripts	1000	0
assertions	4000	0
total run duration: 55.6s		
total data received: 180kB (approx)		
average response time: 39ms [min: 32ms, max: 176ms, s.d.: 9ms]		

```
vladimirromajkin@iMac-Vladimir Test %
```

1000 consecutive requests

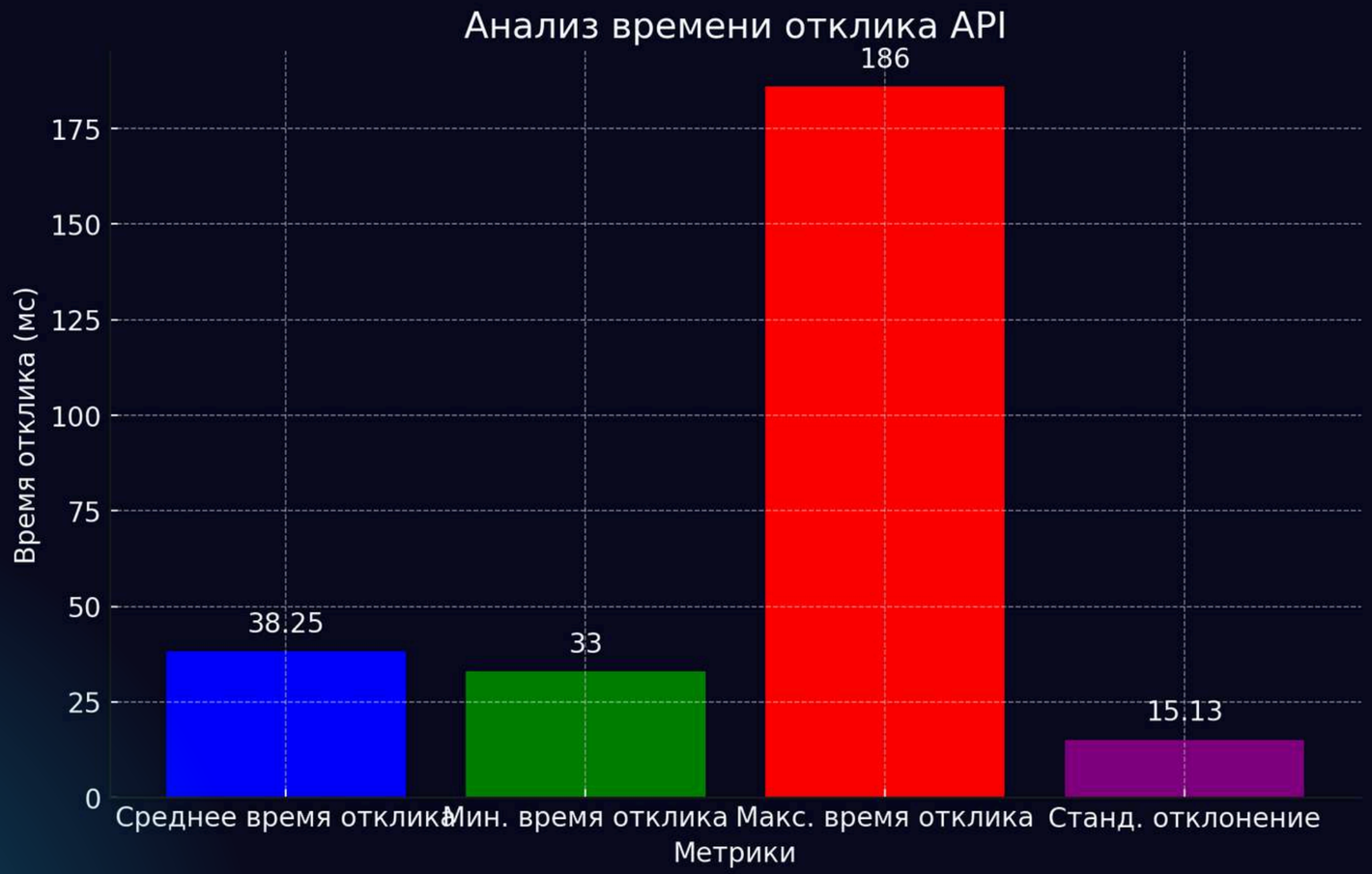
```
StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 38ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 4995/5000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 35ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 4996/5000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 41ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 4997/5000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 36ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 4998/5000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 34ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 4999/5000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 41ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title

Iteration 5000/5000

→ StresTest
GET https://pets-care-u2srs.ondigitalocean.app/api/services_categories [200 OK, 1.1kB, 38ms]
✓  Status code is 200
✓  Response is an array
✓  Categories list is not empty
✓  Each category has a title
```

	executed	failed
iterations	5000	0
requests	5000	0
test-scripts	10000	0
prerequest-scripts	5000	0
assertions	20000	0
total run duration: 4m 33.7s		
total data received: 900kB (approx)		
average response time: 38ms [min: 33ms, max: 192ms, s.d.: 9ms]		

```
vladimirromajkin@iMac-Vladimir Test %
```

5000 consecutive requests

100 Sequential Requests Test Report



General information

- Test name: StresTest
- Test goal: Checking the performance of API GET /api/services_categories under load
- Number of iterations: 100 requests
- Test status: All tests passed successfully

API Performance Metrics

- Average response time: 38.25 ms (fast response time, indicates good optimization)
- Minimum response time: 33 ms (best recorded result)
- Maximum response time: 186 ms (one-time spike, possible delays under load)
- Standard deviation of response time: 15.13 ms (small variability, acceptable stability)

Conclusions

- The API consistently processes a load of 100 consecutive requests without errors (stability confirmed).
- Response time remains within acceptable limits (the service works quickly).
- Response structure meets requirements, data is correct (the service returns correct data).

1000 Sequential Requests Test Report

General information

- Test name: StresTest
- Test goal: Checking the performance of API GET /api/services_categories under load
- Number of iterations: 1000 requests
- Test status: All tests passed successfully

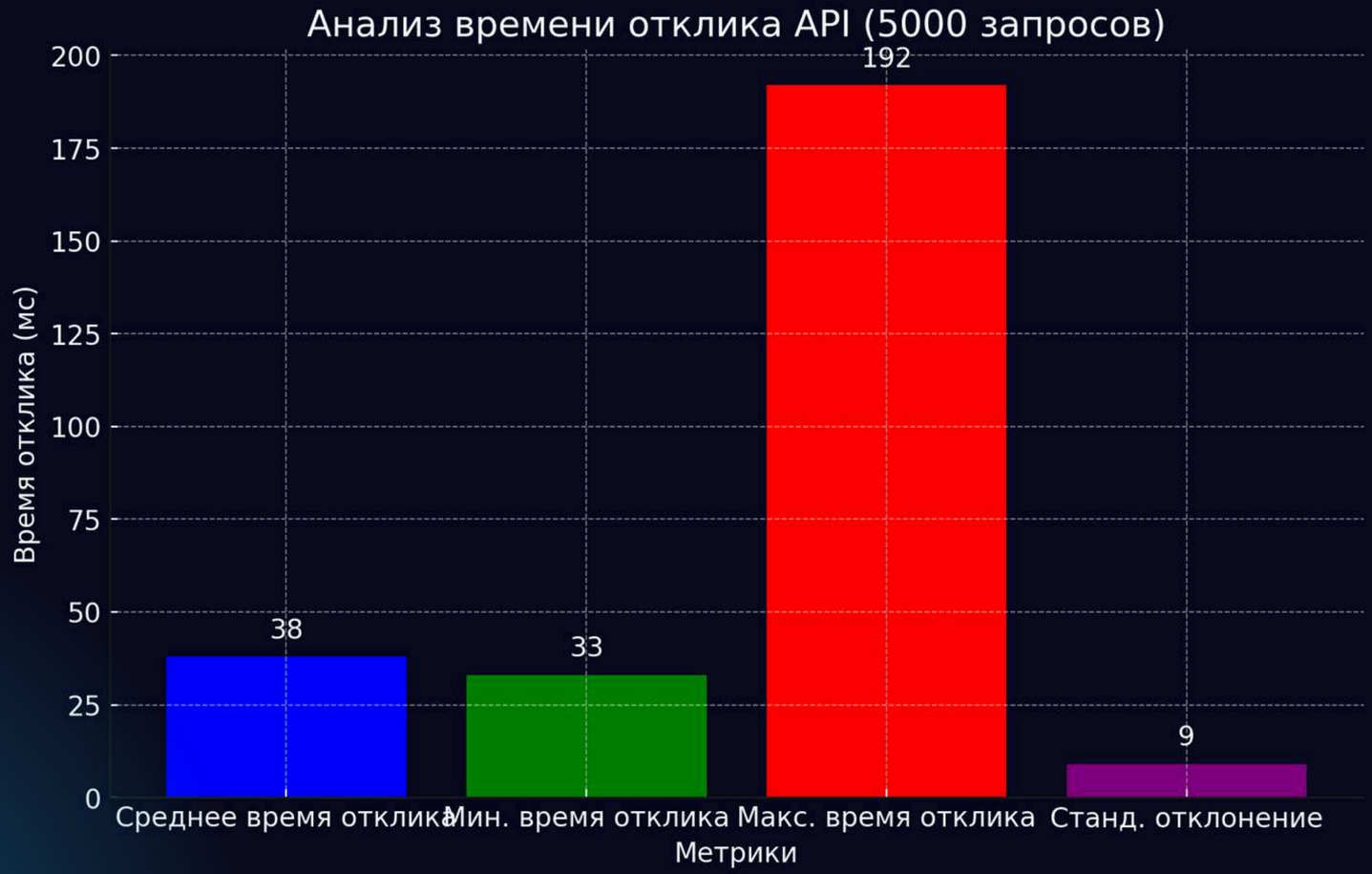
API Performance Metrics

- Average response time: 39.25 ms (within the norm for web services)
- Minimum response time: 32 ms (the fastest request processing)
- Maximum response time: 176 ms (a one-time spike, possibly due to external factors)
- Standard deviation of response time: 9.28 ms (an indicator of the stability of response time)

Conclusions

- The API correctly withstands 1000 consecutive requests (confirmed high resilience).
- The response time is stable and remains within acceptable values (no significant delays).
- The response structure meets the requirements, the data is correct (the API works as expected).

5000 Sequential Requests Test Report



General information

- Test name: StresTest
- Test goal: Checking the performance of API GET /api/services_categories under load
- Number of iterations: 5000 requests
- Test status: All tests passed successfully

API Performance Metrics

- Average response time: 38 ms (very fast request processing time)
- Minimum response time: 33 ms (best result)
- Maximum response time: 192 ms (one-off spike, possibly caused by server load)
- Standard deviation of response time: 9 ms (low variability, indicating good stability)

Conclusions

- The API can handle 5,000 consecutive requests without failure (high stability).
- The response time is stable and remains within acceptable values (optimal speed).
- The response structure meets the requirements, the data is correct (the service processes requests without errors)

API Load Testing Summary Report

General information

- Test name: StresTest
- Test goal: Test the stability and performance of the API GET /api/services_categories under different load levels.
- Number of iterations: 100, 1000 and 5000 requests (three load levels).
- Test status: All tests passed successfully (the API processed all requests without errors).

Conclusion:

- The average response time remains stable as the load increases (around 38-39 ms).
- The minimum response time remains almost unchanged (~32-33 ms).
- The maximum response time has jumps, especially at 5000 requests (192 ms), which may indicate server overload.
- The standard deviation decreases as the load increases, indicating a more stable response in long-term testing.

Comparative statistics of test execution

Метрика	100 запросов	1000 запросов	5000 запросов
Всего запросов	100	1000	5000
Успешные запросы	100 (100%)	1000 (100%)	5000 (100%)
Проваленные запросы	0 (0%)	0 (0%)	0 (0%)
Всего тестов	400	4000	20000
Проваленные тесты	0 (0%)	0 (0%)	0 (0%)

Comparative performance metrics

Метрика	100 запросов	1000 запросов	5000 запросов
Среднее время отклика (мс)	38.25	39.25	38.00
Минимальное время (мс)	33	32	33
Максимальное время (мс)	186	176	192
Стандартное отклонение (мс)	15.13	9.28	9.00

API Load Testing Summary Report

Here is a summary graph of the API response time analysis under different load levels (100, 1000, and 5000 requests). The lines show:

- Blue – average response time (almost constant).
- Red – maximum response time (increases with increasing load).
- Purple – standard deviation (decreases with increasing load, indicating stability).

Conclusions from the graph

The API shows stable average response time (~38 ms) as the number of requests increases.

Maximum latencies sometimes increase (up to 192 ms), which requires analysis - perhaps the server is experiencing peak loads.

The standard deviation decreases, which indicates better predictability of response times under high loads.

Load and stress testing

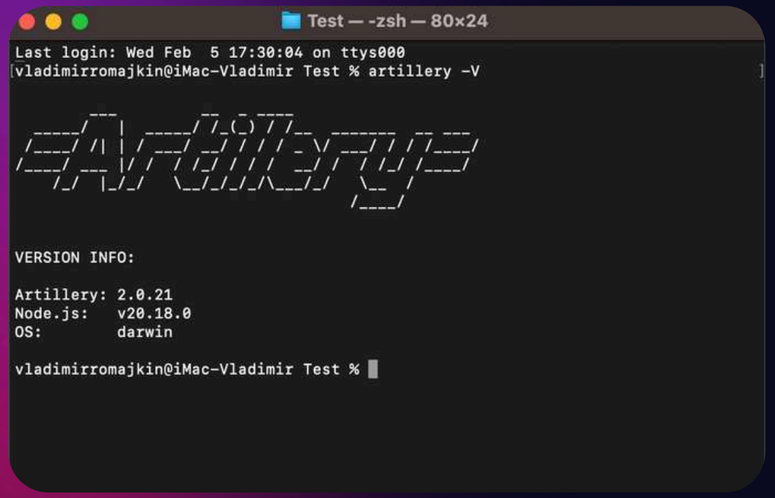
Load tests conducted in Artillery

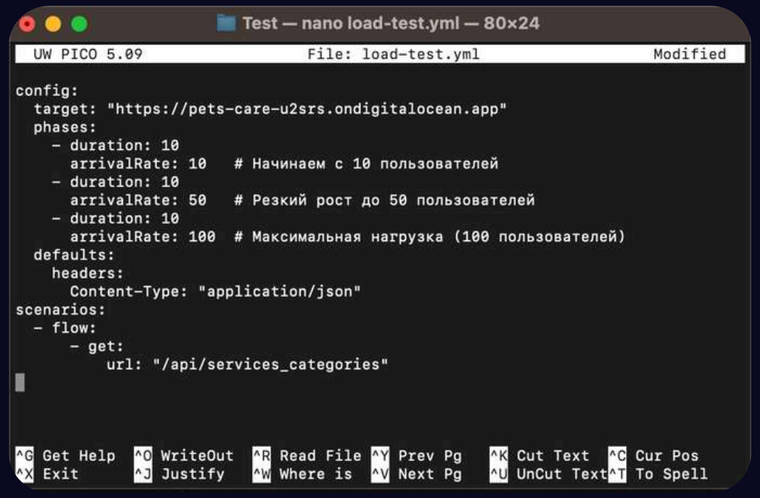
Purpose of testing

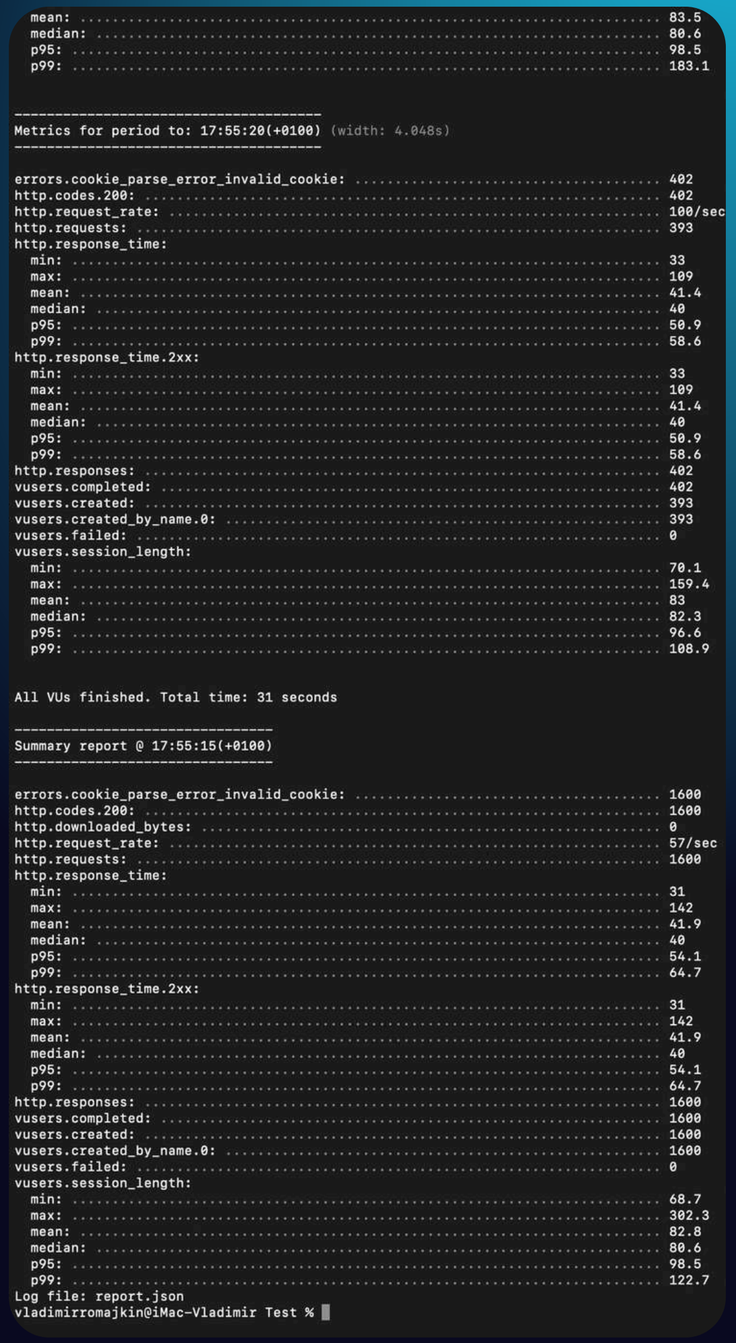
Testing API performance under high load with parallel requests using Artillery.
Testing allows you to determine:

- How does the API handle a large number of concurrent users?
- What response time metrics does the system demonstrate?
- Are there timeouts and errors when the number of parallel requests increases?
- How stable are the API responses under load?

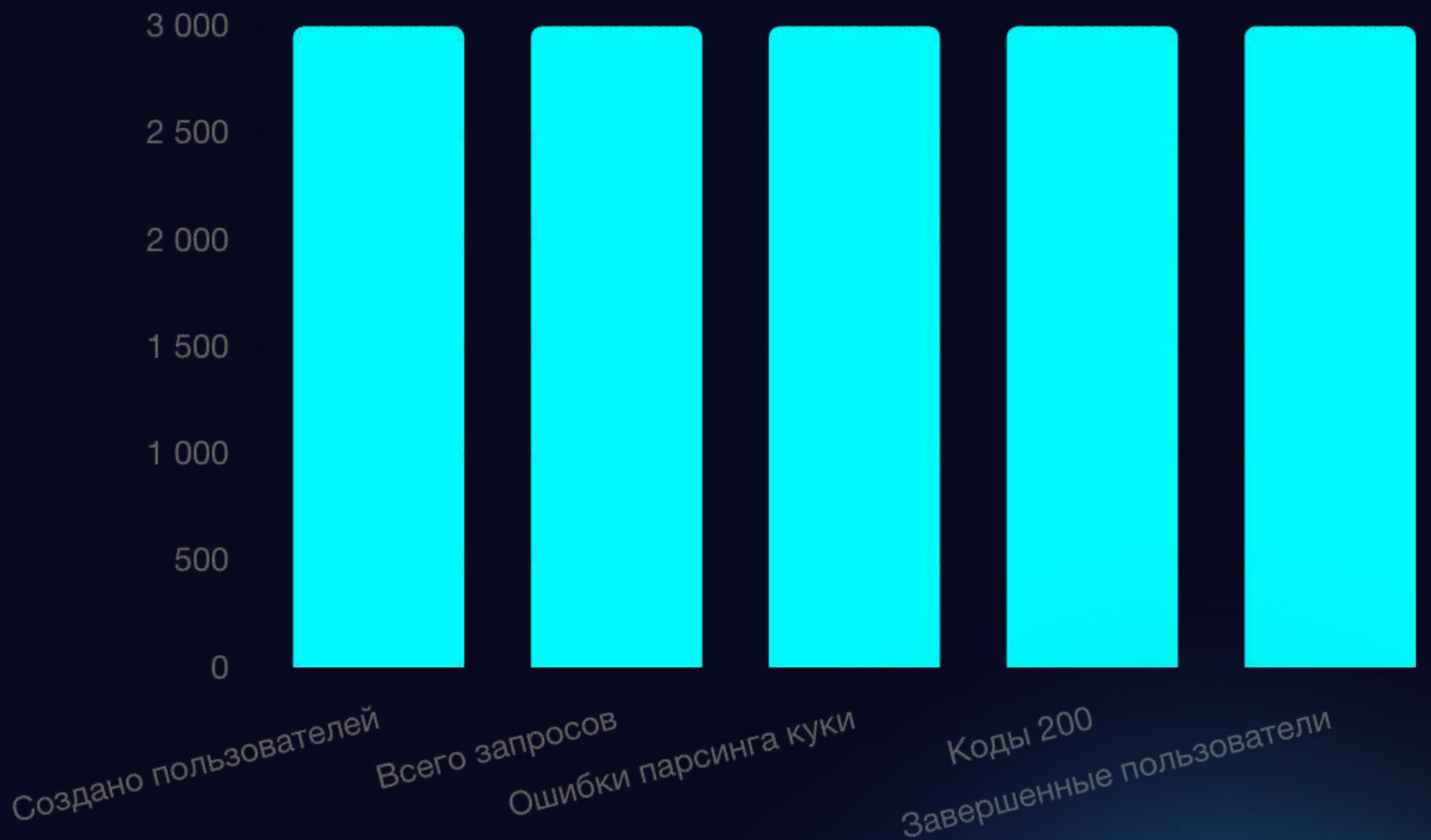
Modern web systems must be able to handle a large number of concurrent users and provide fast responses.
To achieve this, it is important to perform load testing with tools such as Artillery, which allow you to simulate real-world load scenarios.







Artillery Load Test (50 rps)



General information

- Testing goal: Testing the stability and performance of the API /api/services_categories at 50 parallel requests per second.
- Number of iterations: 3000 requests
- Testing status: All tests passed successfully (code 200 for 100% of requests)

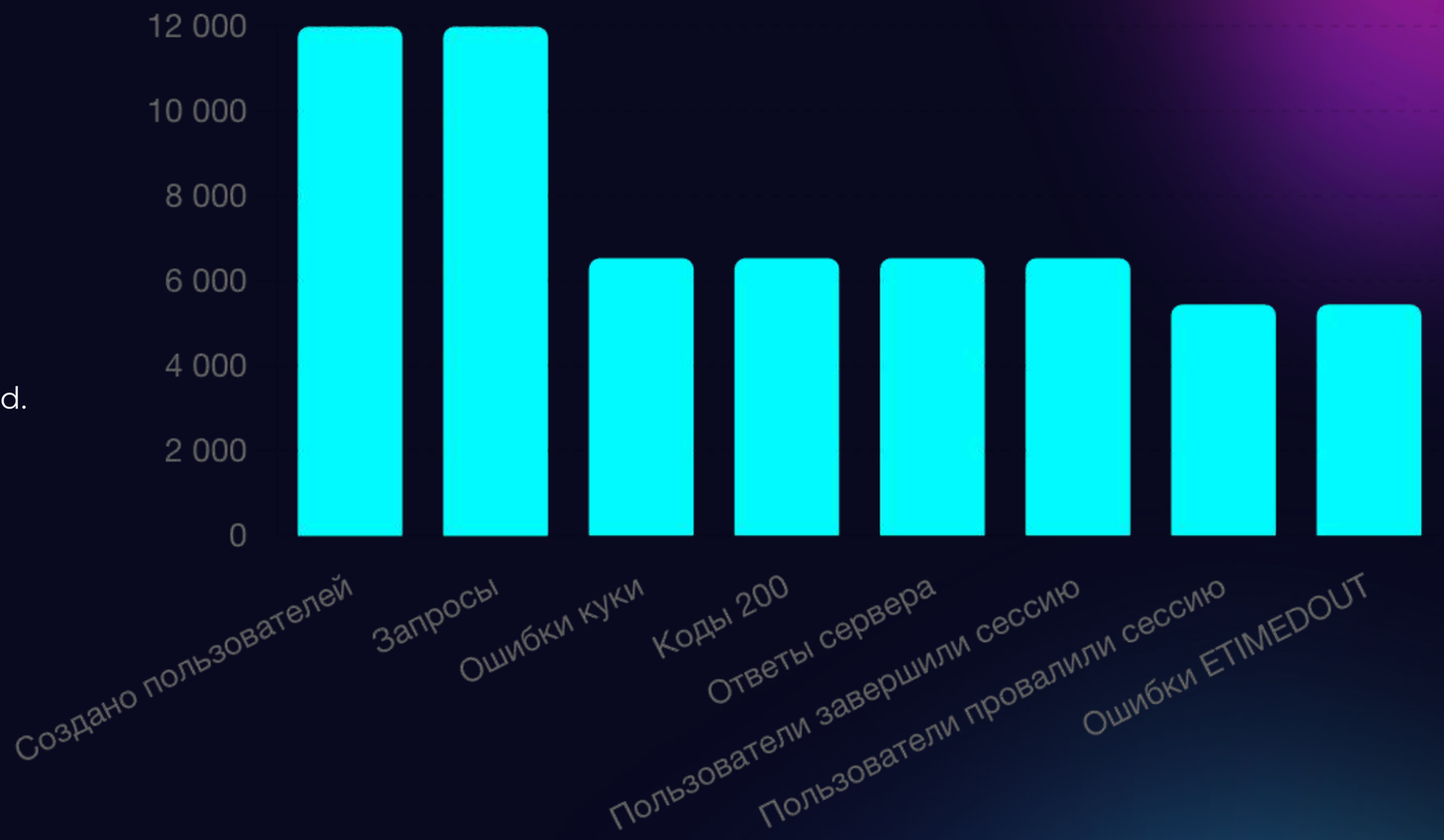
API Performance Metrics

- Average response time: 40.6 ms
- Minimum response time: 3 ms
- Maximum response time: 185 ms
- Standard deviation of response time: 9.5 ms
- Median response time (p50): 39.3 ms
- 90% of requests (p90) complete in less than 47 ms
- 99% of requests (p99) complete in 71.5 ms
- Rare spikes up to 172.5 ms

Conclusions

- The API can handle a load of 50 rps stably, but needs to fix some cookie errors.
- The API does not return useful data, which makes it inoperative.
- There are response time jumps up to 185 ms, which requires analysis.

Artillery Load Test (100 rps)



General information

- Test objective: Evaluate the stability and performance of the /api/services_categories API at 100 parallel requests per second.
- Number of iterations: 12,000 requests
- Test status: Partially successful - high percentage of timeouts

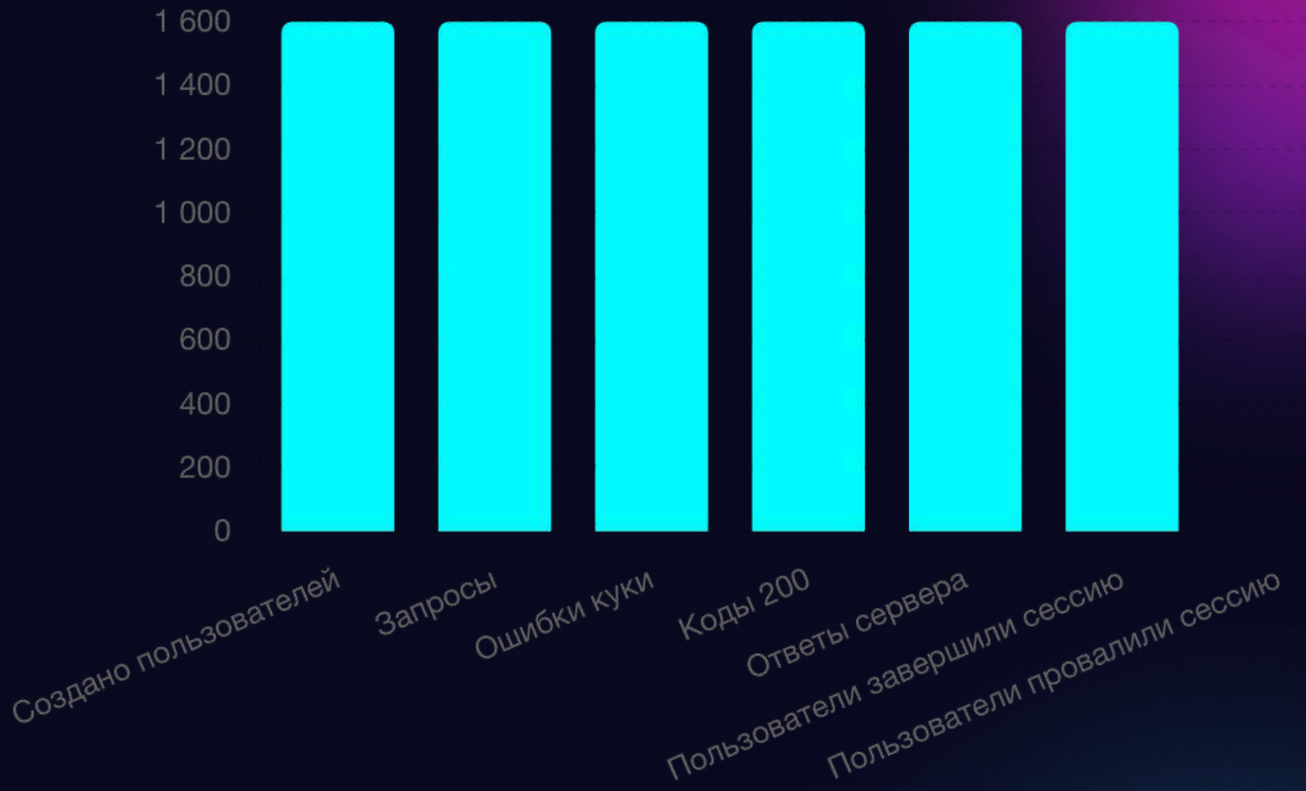
API Performance Metrics

- Average response time: 56.9 ms
- Minimum response time: 3 ms
- Maximum response time: 839 ms
- Standard deviation of response time: 108.9 ms
- Median response time (p50): 40 ms
- 90% of requests (p90) complete in less than 108.9 ms
- 95% of requests (p95) complete in 144 ms
- 99% of requests (p99) complete in 242.3 ms
- Rare spikes up to 658.6 ms

Conclusions

- The API can handle a load of 100 rps stably, but requires optimization.
- 45.4% of requests end in timeouts, which is unacceptable for a highly loaded API.
- The API does not return useful data, which makes it inoperative.
- There are response time jumps up to 839 ms, which requires analysis.

Artillery Burst Load Test



General information

- Testing goal: Checking the /api/services_categories API operation under burst load (burst testing).
- Number of iterations: 1,600 requests (simulating a sudden increase in load).
- Testing status: All requests were processed successfully (code 200 for 100% of requests).

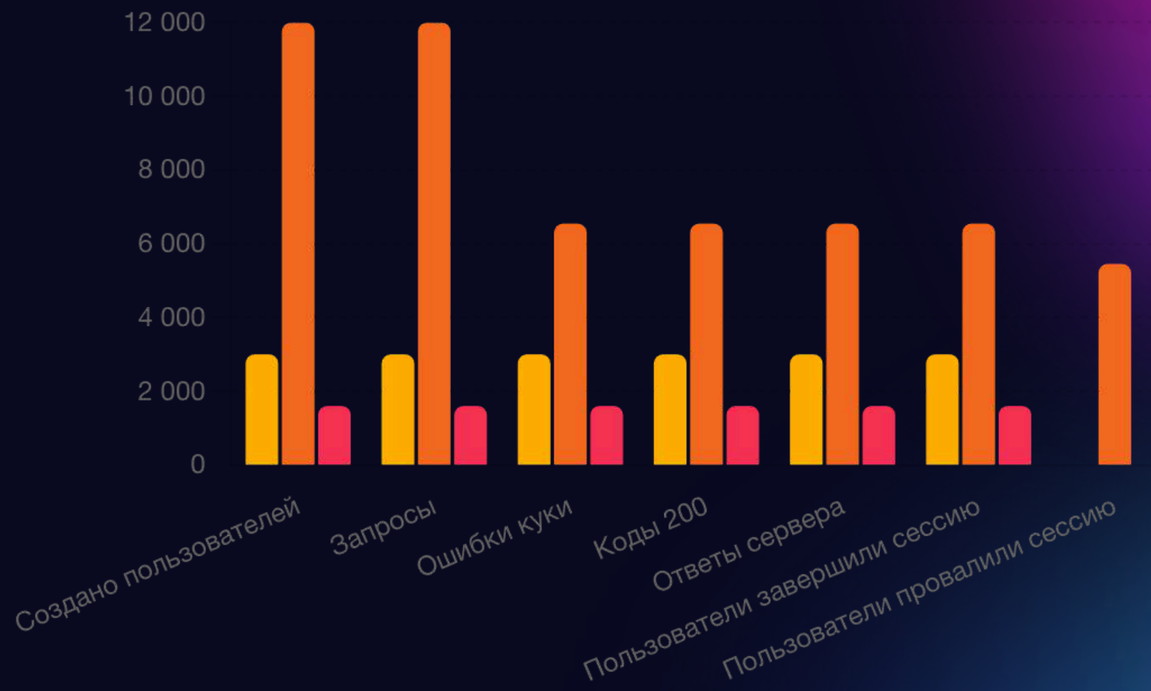
API Performance Metrics

- Average response time: 41.9 ms (optimal).
- Minimum response time: 31 ms (best result).
- Maximum response time: 142 ms (one-time spike, but within normal limits).
- Standard deviation of response time: 9.8 ms (stability indicator).
- Median response time (p50): 40 ms (half of requests are processed faster).
- 90% of requests (p90) are completed in less than 48.9 ms (confident operation under load).
- 95% of requests (p95) are completed in 54.1 ms (almost all requests are completed quickly).
- 99% of requests (p99) are completed in 64.7 ms (some delays, but no critical spikes).
- Rare spikes up to 133 ms (one-time delays that do not affect overall stability).

Conclusions

- The API consistently withstands burst load (burst test), without failures or timeouts.
- Cookie parsing errors on 100% of requests, requires fixing (recurring issue that requires attention).
- API does not always return useful data (0 bytes), server logic needs to be checked.
- Minimal response time spikes, but maximum delays are worth analyzing.

API Load Testing Summary Report (Artillery)



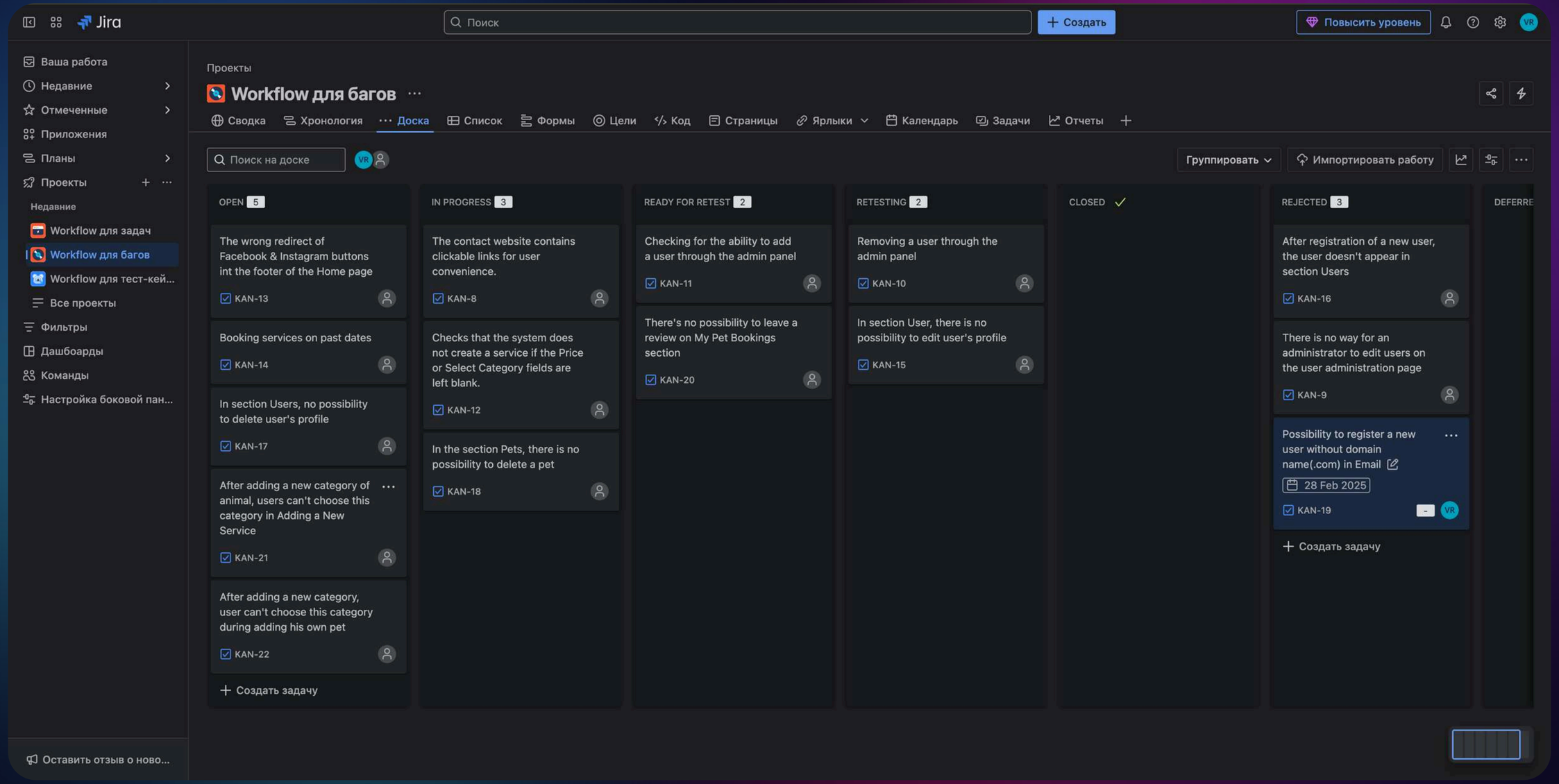
Conclusion:

- 50 rps and Burst tests showed 100% successful requests, but the API does not return data.
- 100 rps caused 45.4% of timeouts, indicating server overload.
- All tests had cookie parsing errors, which requires fixing.
- Average response time increases with increasing load (40.6 → 56.9 ms).
- Maximum response time at 100 rps reached 839 ms, indicating latency.
- Standard deviation is higher in the 100 rps test (108.9 ms), indicating instability.
- Burst test shows good results, but the API still does not return data.
- The API handles 50 rps and Burst load stably, but loses requests at 100 rps (45.4% of timeouts).
- Cookie parsing errors are present in all tests, which requires fixing.
- Maximum latencies at 100 rps are too high (839 ms), which requires optimization.
- The API does not return useful data (0 bytes in all tests), which makes it unusable.
- The API reliably handles 50 rps and Burst load, but loses requests at 100 rps (45.4% of timeouts).
- Cookie parsing errors are present in all tests, this needs to be fixed.
- Maximum latencies at 100 rps are too high (839 ms), which requires optimization.
- The API does not return useful data (0 bytes in all tests), which makes it unusable.

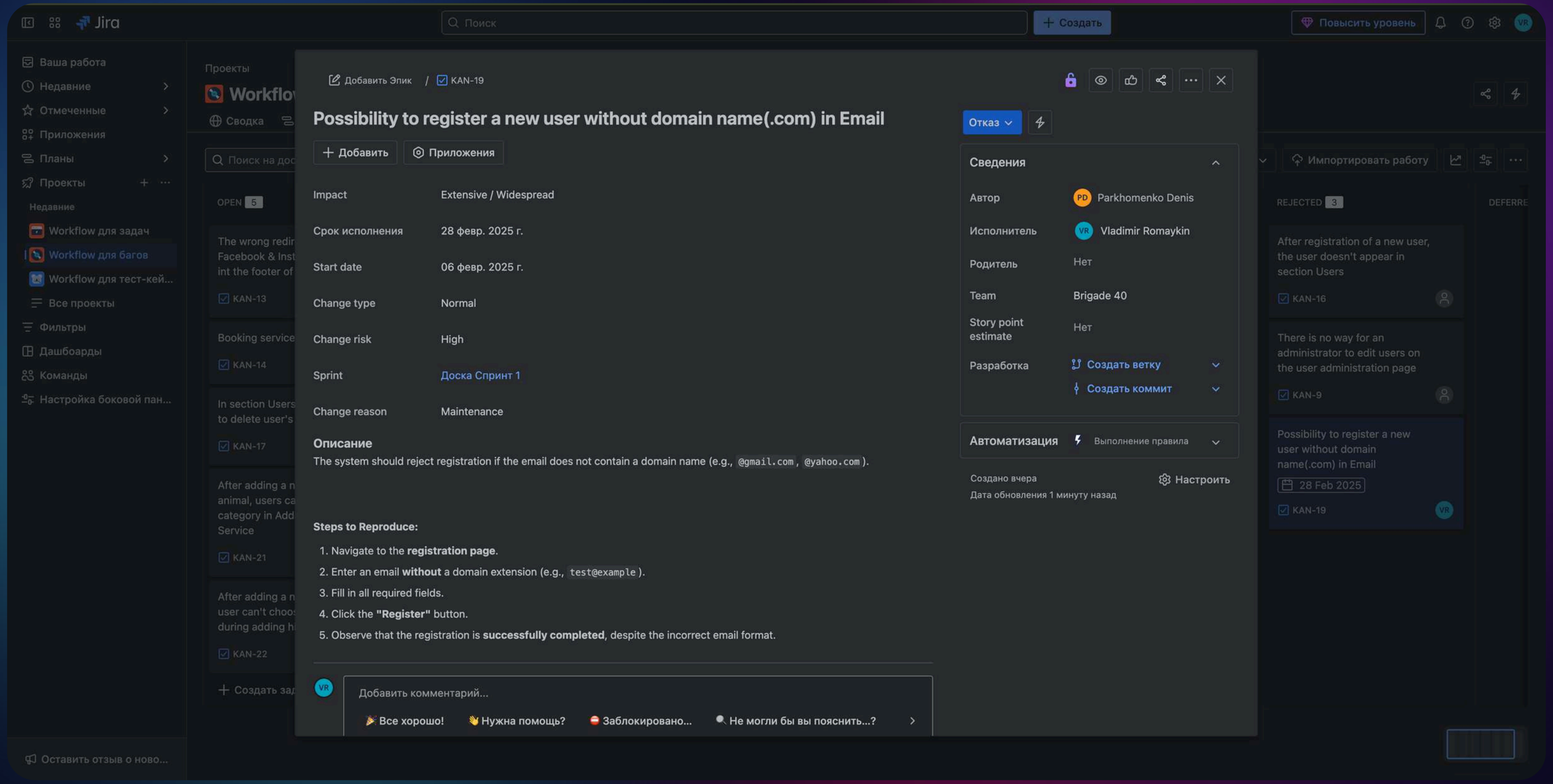
Recommendations

- Fix cookie parsing errors – check cookie handling on the server.
- Optimize API performance under high load (100 rps) – consider load balancing, increase timeouts, analyze bottlenecks.
- Fix data return issue – check why API is not sending useful information to the client.
- Retest after fixes – to verify improvements.

Jira


Jira
Поиск
Создать
Повысить уровень
Ваша работа
Недавние
Отмеченные
Приложения
Планы
Проекты
Недавние
Workflow для задач
Workflow для багов
Workflow для тест-кей...
Все проекты
Фильтры
Дашбоарды
Команды
Настройка боковой пан...
Оставить отзыв о ново...
Проекты
Workflow для багов
Сводка
Хронология
Доска
Список
Формы
Цели
Код
Страницы
Ярлыки
Календарь
Задачи
Отчеты
Поиск на доске
Группировать
Импортировать работу
OPEN 5
The wrong redirect of Facebook & Instagram buttons int the footer of the Home page
KAN-13
Booking services on past dates
KAN-14
In section Users, no possibility to delete user's profile
KAN-17
After adding a new category of animal, users can't choose this category in Adding a New Service
KAN-21
After adding a new category, user can't choose this category during adding his own pet
KAN-22
Создать задачу
IN PROGRESS 3
The contact website contains clickable links for user convenience.
KAN-8
Checks that the system does not create a service if the Price or Select Category fields are left blank.
KAN-12
In the section Pets, there is no possibility to delete a pet
KAN-18
READY FOR RETEST 2
Checking for the ability to add a user through the admin panel
KAN-11
There's no possibility to leave a review on My Pet Bookings section
KAN-20
RETESTING 2
Removing a user through the admin panel
KAN-10
In section User, there is no possibility to edit user's profile
KAN-15
CLOSED
REJECTED 3
After registration of a new user, the user doesn't appear in section Users
KAN-16
There is no way for an administrator to edit users on the user administration page
KAN-9
Possibility to register a new user without domain name(.com) in Email
28 Feb 2025
KAN-19
Создать задачу
DEFERRE

Jira


Jira
Поиск
Создать
Повысить уровень
Ваша работа
Недавние
Отмеченные
Приложения
Планы
Проекты
Недавние
Workflow для задач
Workflow для багов
Workflow для тест-кей...
Все проекты
Фильтры
Дашбоарды
Команды
Настройка боковой пан...
Оставить отзыв о ново...
Проекты
Сводка
Импортировать работу
OPEN 5
REJECTED 3
DEFERRE
KAN-13
KAN-14
KAN-17
KAN-21
KAN-22
KAN-16
KAN-9
KAN-19
28 Feb 2025
Добавить Эпик / KAN-19
Possibility to register a new user without domain name(.com) in Email
Добавить
Приложения
Impact Extensive / Widespread
Срок исполнения 28 февр. 2025 г.
Start date 06 февр. 2025 г.
Change type Normal
Change risk High
Sprint Доска Спринт 1
Change reason Maintenance
Описание
The system should reject registration if the email does not contain a domain name (e.g., @gmail.com, @yahoo.com).
Steps to Reproduce:
1. Navigate to the registration page.
2. Enter an email without a domain extension (e.g., test@example).
3. Fill in all required fields.
4. Click the "Register" button.
5. Observe that the registration is successfully completed, despite the incorrect email format.
Добавить комментарий...
Все хорошо!
Нужна помощь?
Заблокировано...
Не могли бы вы пояснить...?
Отказ
Сведения
Автор Parkhomenko Denis
Исполнитель Vladimir Romaykin
Родитель Нет
Team Brigade 40
Story point estimate Нет
Разработка
Создать ветку
Создать коммит
Автоматизация
Выполнение правила
Создано вчера
Дата обновления 1 минуту назад
Настроить

Thank you for your attention!