deti universidade de aveiro
departamento de electrónica, telecomunicações e informática

Projeto TQS - Apresentação Final

GAMERENT
PLAY • DISCOVER • UNLOCK FUN

Duarte Branco 119253
Filipe Viseu 119192
Samuel Vinhas 119405
André Ferreira 119061

Professores:
Ilídio Oliveira
Sérgio Freire

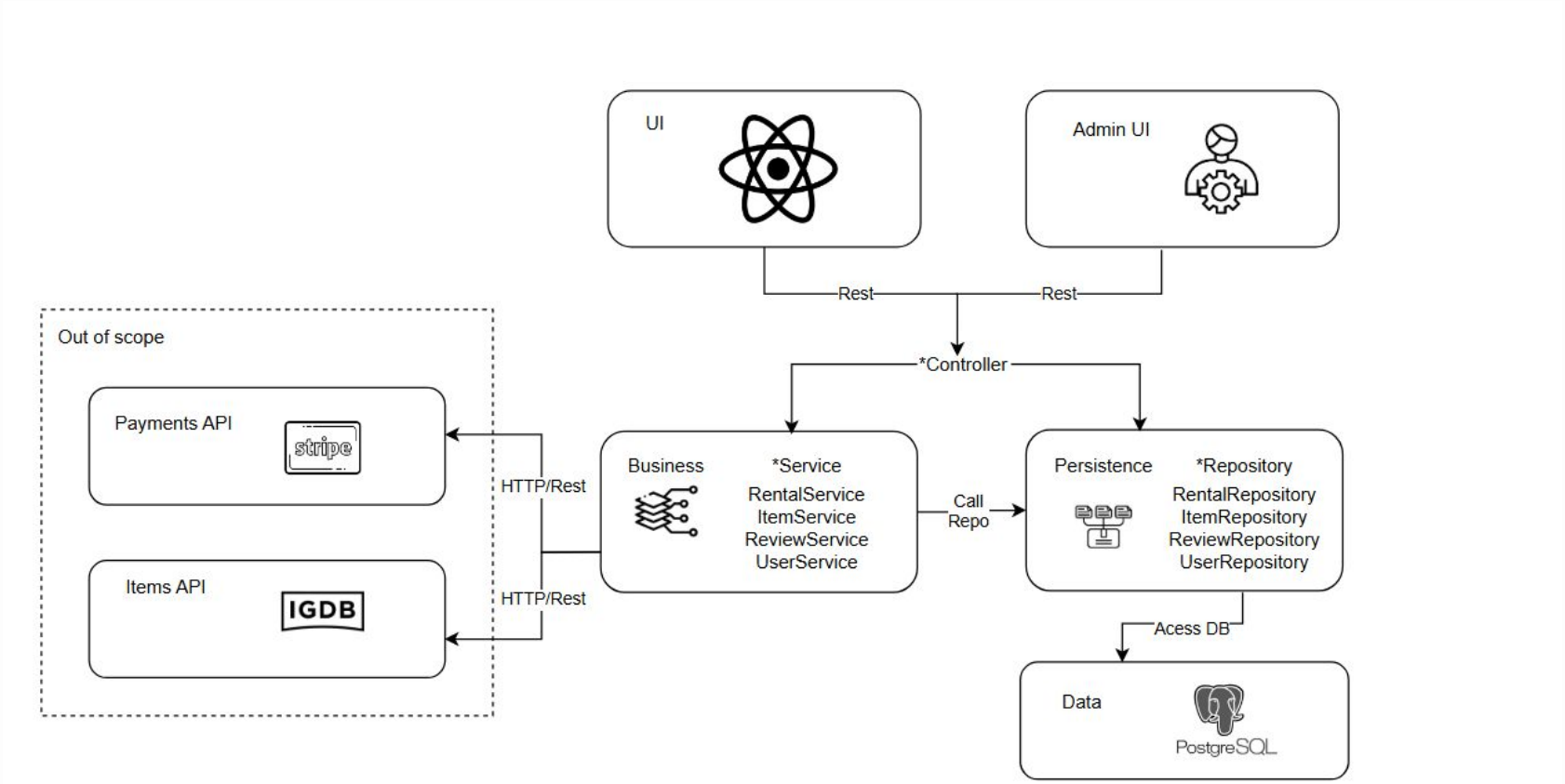
Contexto

GAMERENT
PLAY • DISCOVER • UNLOCK FUN

Objetivo:

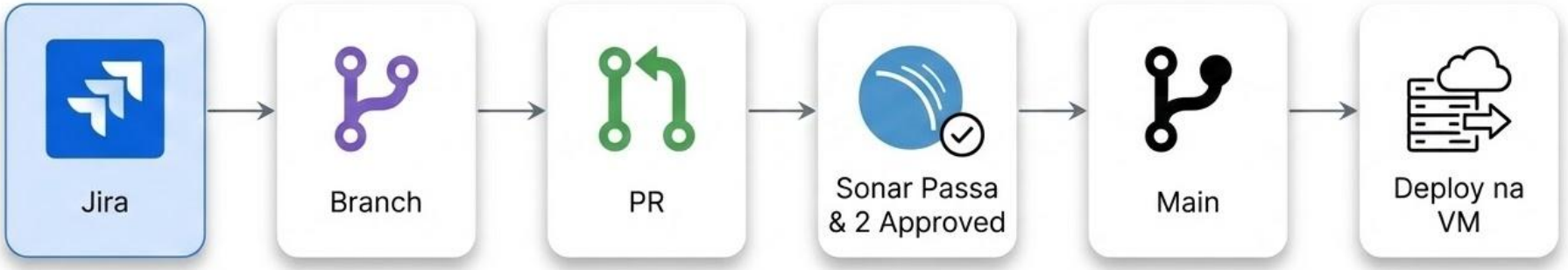
Desenvolver uma plataforma que permita aos utilizadores listar consolas, jogos e acessórios assim como alugar de outros utilizadores por períodos específicos a preços acessíveis

Arquitetura



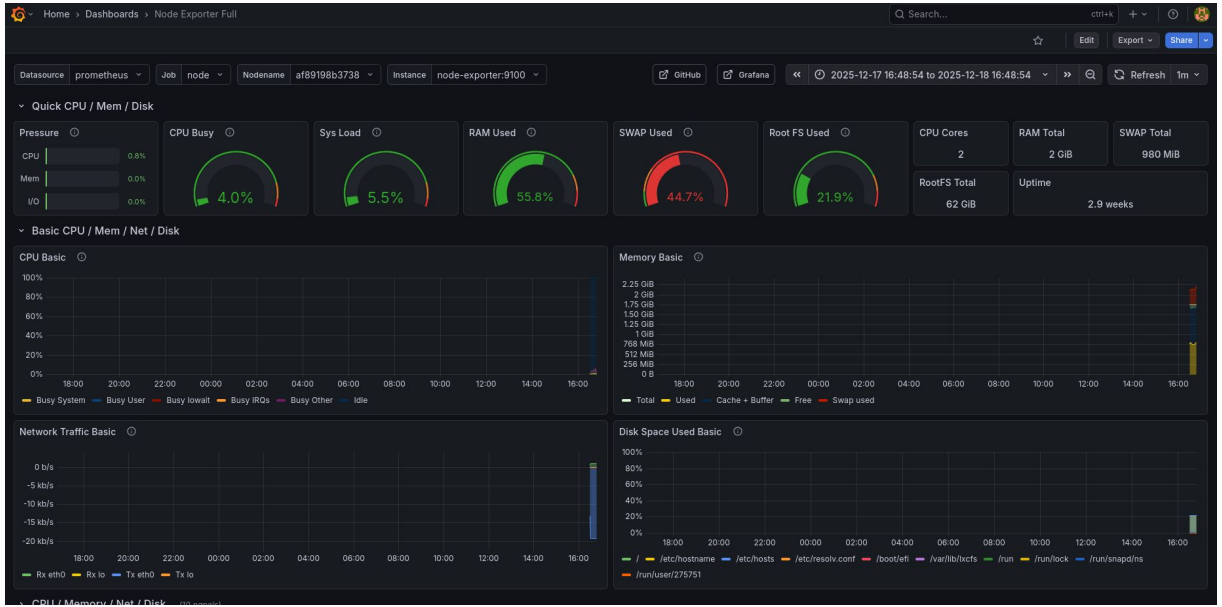
Demonstração

Workflow



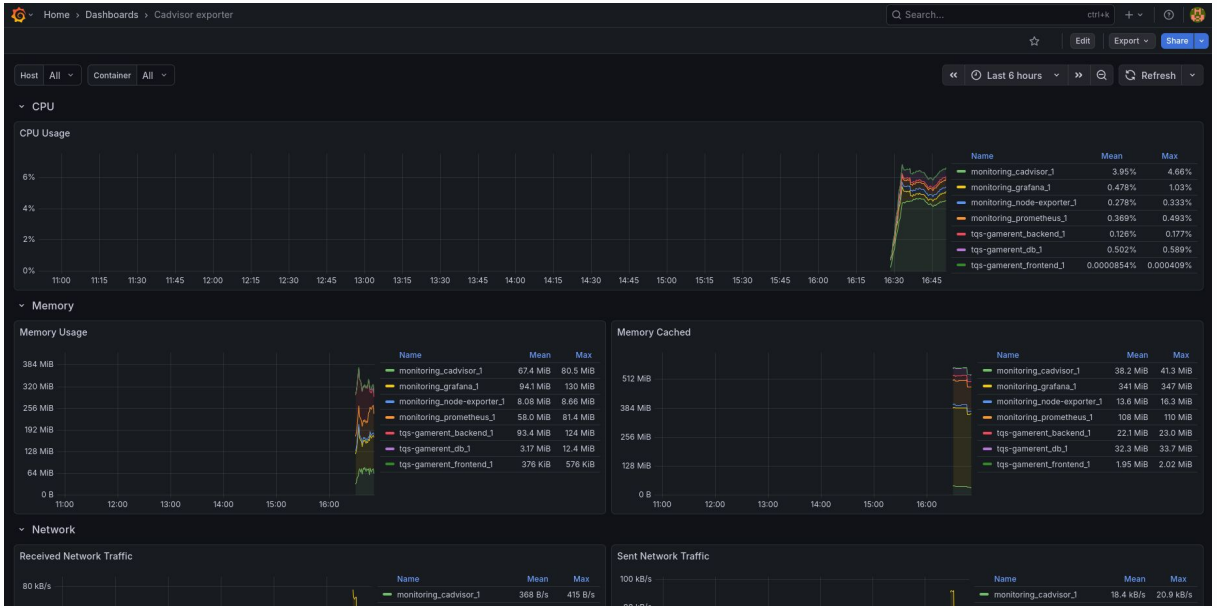
System Observability (Grafana)

Node Exporter Full



System Observability (Grafana)

Cadvisor exporter



Testes - Service

```java
@Test
void createBooking_ShouldCreateWhenNoConflict() {
    when(itemRepository.findById(1L)).thenReturn(Optional.of(item));
    when(bookingRepository.findByItemIdAndStatus(1L, BookingStatus.APPROVED))
        .thenReturn(List.of());
    when(bookingRepository.save(any(BookingRequest.class)))
        .thenAnswer(invocation -> invocation.getArgument(0));

    LocalDate start = LocalDate.now().plusDays(1);
    LocalDate end = LocalDate.now().plusDays(5);

    BookingRequest result = bookingService.createBooking(1L, 2L, start, end);


    assertNotNull(result);
    assertEquals(BookingStatus.PENDING, result.getStatus());
    long days = java.time.temporal.ChronoUnit.DAYS.between(
        LocalDate.of(2025,12,1), LocalDate.of(2025,12,5)) + 1;
    assertEquals(15.0 * (double) days, result.getTotalPrice());
    verify(bookingRepository, times(1)).save(any(BookingRequest.class));
}
```

Testes - Playwright

```java
@Test
void testUserCanBookItem() {
    String baseUrl = "http://localhost:3000";
    // Login
    page.navigate(baseUrl + "/auth");
    page.fill("input[type='email']", "booking-owner@test.com");
    page.fill("input[type='password']", "password123");
    page.click("button[type='submit']");
    page.waitForTimeout(2000);

    // Select item
    page.navigate(baseUrl + "/");
    page.waitForSelector(".item-card", new Page.WaitForSelectorOptions().setTimeout(5000));
    page.click(".item-card");
    page.waitForSelector("input[type='date']", new Page.WaitForSelectorOptions().setTimeout(5000));

    // Use locator.first() and locator.last() to fill each date input correctly
    java.time.LocalDate startDate = java.time.LocalDate.now().plusDays(100);
    java.time.LocalDate endDate = startDate.plusDays(5);

    Locator dateInputs = page.locator("input[type='date']");
    dateInputs.first().fill(startDate.toString());
    dateInputs.last().fill(endDate.toString());
    page.waitForTimeout(500);

    // Verify the button is no longer disabled
    Locator bookButton = page.locator("button:has-text('Request Booking')");
    assertFalse(bookButton.isDisabled(), "Request Booking button should be enabled when valid dates are selected");
}
```

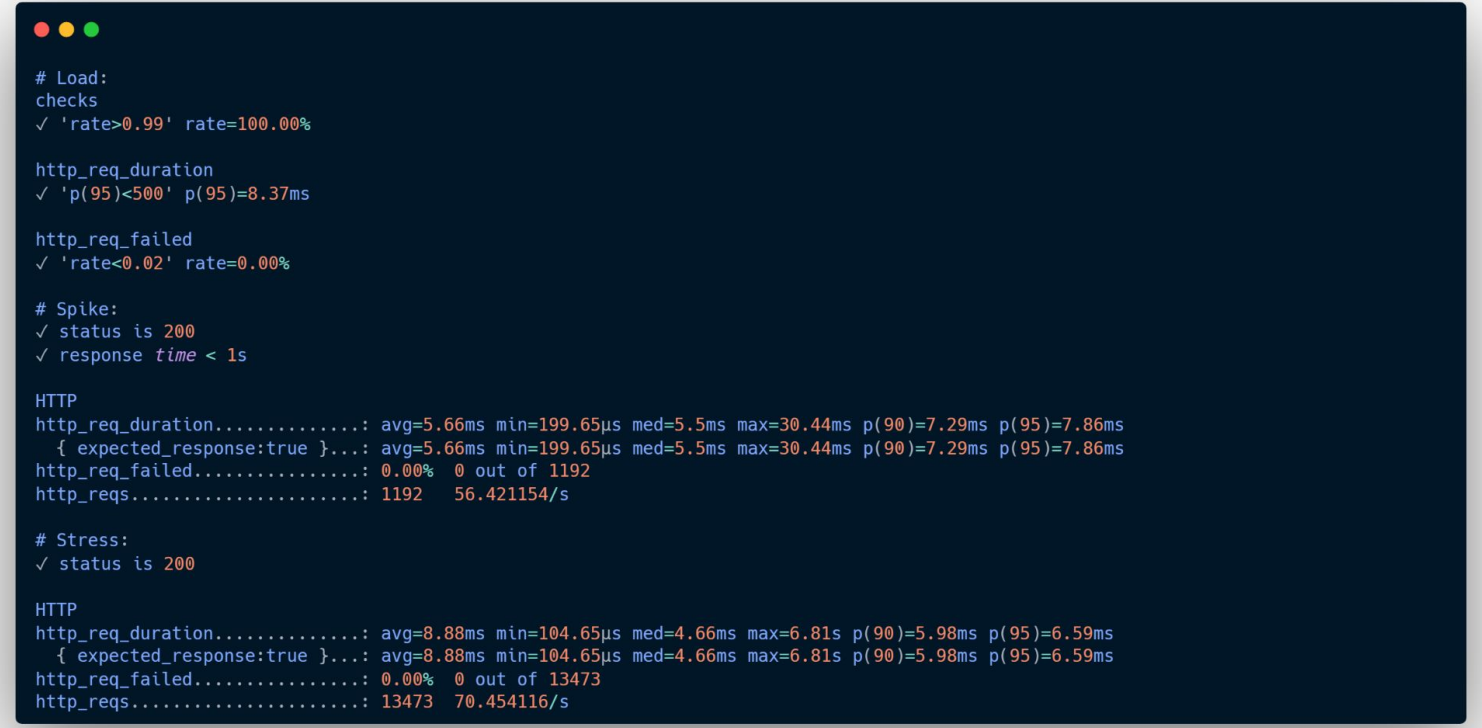
Testes - Cucumber

```
Feature: Book Item
  As a renter
  I want to book an item
  So that I can use it

  Scenario: Renter books an item
    Given a user "renter@example.com" exists
    And an item "Switch" exists owned by "owner@example.com"
    When the user "renter@example.com" books "Switch" for tomorrow
    Then the booking status should be "PENDING"
```

Testes - Integration

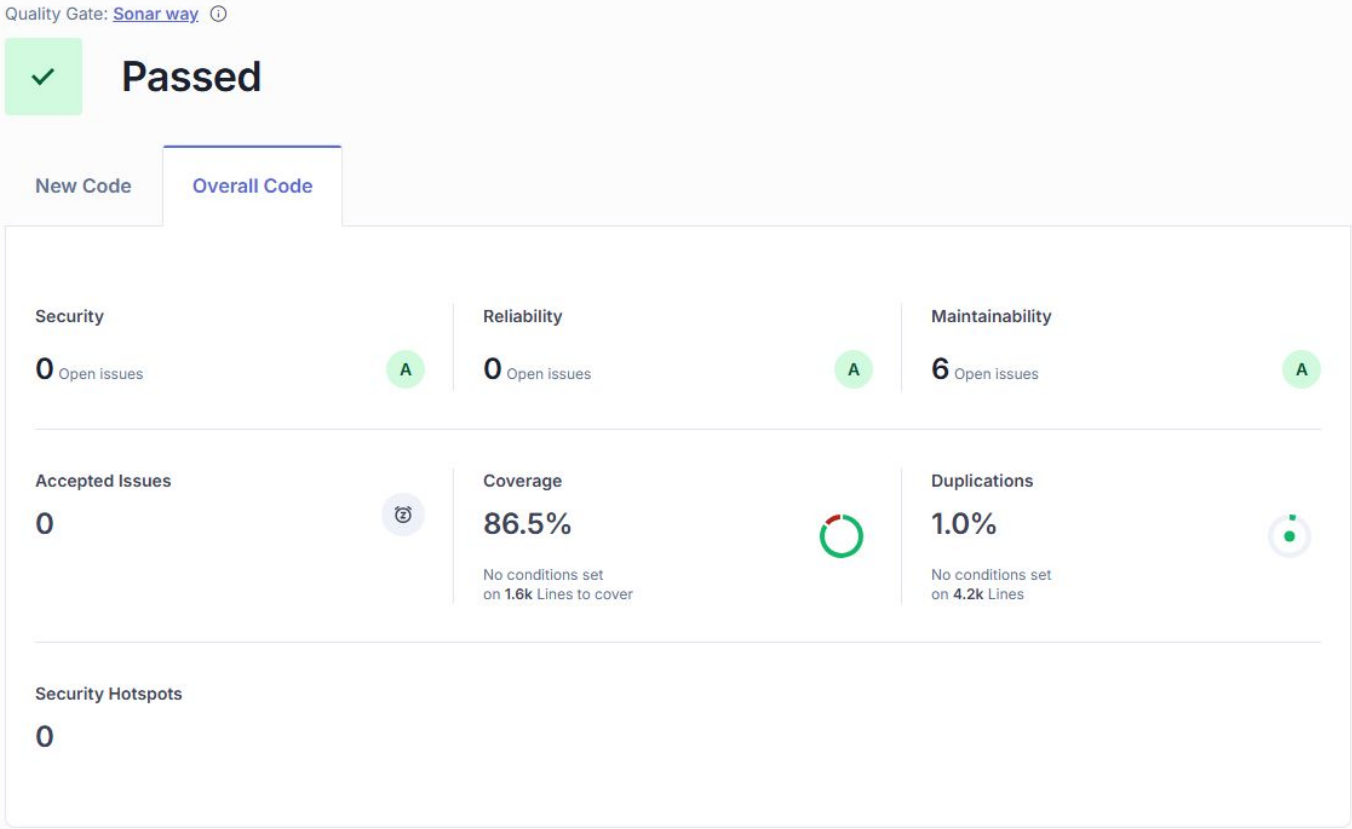
```java
@Test
@XrayTest(key = "TGR-27")
@Tag("integration")
void createBooking_ShouldReturn200AndCreateBooking() throws Exception {
    String json = """
        {
            "itemId": %d,
            "userId": %d,
            "startDate": "2035-12-01",
            "endDate": "2035-12-05"
        }
        """.formatted(item.getId(), renter.getId());

    mockMvc.perform(post("/api/bookings")
            .contentType(MediaType.APPLICATION_JSON)
            .content(json))
            .andExpect(status().isOk())
            .andExpect(jsonPath("$.id").isNumber())
            .andExpect(jsonPath("$.itemId").value(item.getId()))
            .andExpect(jsonPath("$.userId").value(renter.getId()))
            .andExpect(jsonPath("$.status").value("PENDING"));
}
```

Testes - K6

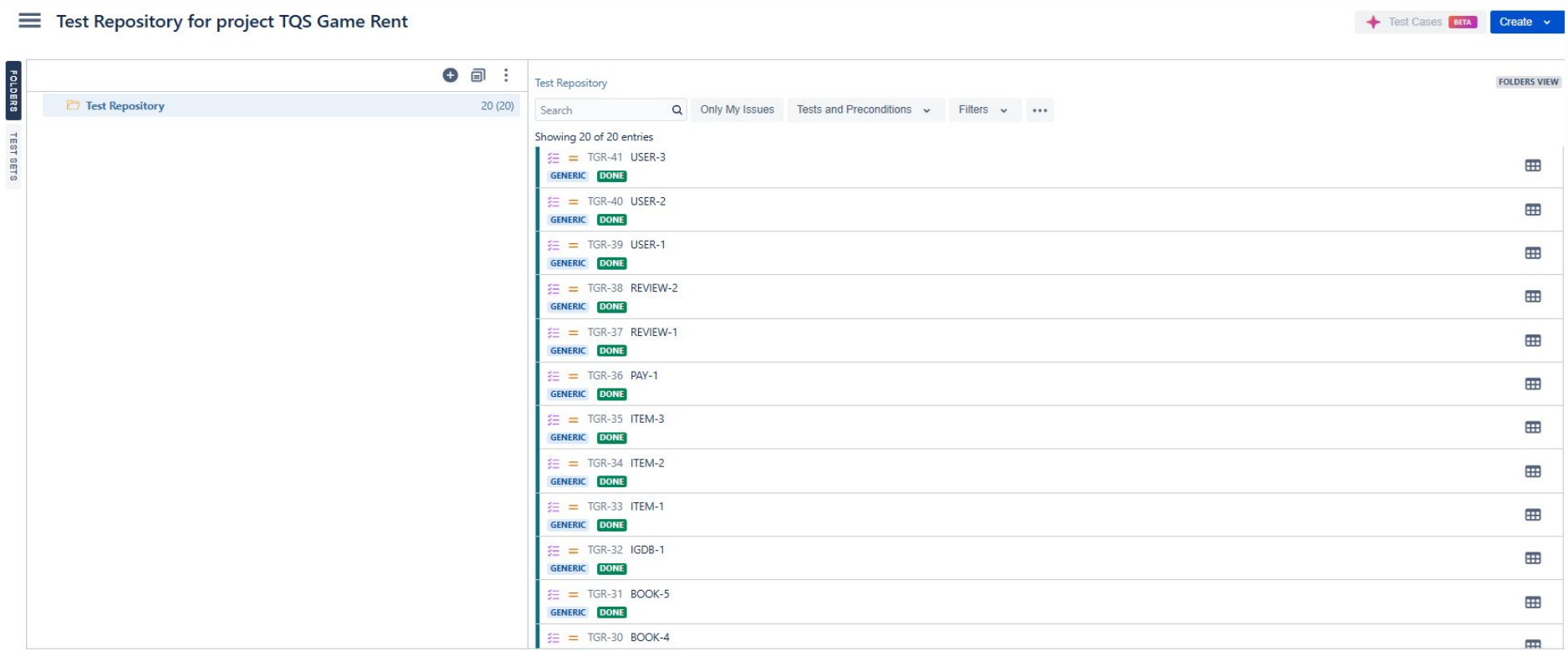


```
# Load:
checks
✓ 'rate>0.99' rate=100.00%

http_req_duration
✓ 'p(95)<500' p(95)=8.37ms

http_req_failed
✓ 'rate<0.02' rate=0.00%

# Spike:
✓ status is 200
✓ response time < 1s

HTTP
http_req_duration..............: avg=5.66ms min=199.65µs med=5.5ms max=30.44ms p(90)=7.29ms p(95)=7.86ms
  { expected_response:true }...: avg=5.66ms min=199.65µs med=5.5ms max=30.44ms p(90)=7.29ms p(95)=7.86ms
http_req_failed................: 0.00%  0 out of 1192
http_reqs......................: 1192   56.421154/s

# Stress:
✓ status is 200

HTTP
http_req_duration..............: avg=8.88ms min=104.65µs med=4.66ms max=6.81s p(90)=5.98ms p(95)=6.59ms
  { expected_response:true }...: avg=8.88ms min=104.65µs med=4.66ms max=6.81s p(90)=5.98ms p(95)=6.59ms
http_req_failed................: 0.00%  0 out of 13473
http_reqs......................: 13473  70.454116/s
```

Sonar


Quality Gate: Sonar way
Passed
New Code
Overall Code
Security
0 Open issues
A
Reliability
0 Open issues
A
Maintainability
6 Open issues
A
Accepted Issues
0
Coverage
86.5%
No conditions set on 1.6k Lines to cover
Duplications
1.0%
No conditions set on 4.2k Lines
Security Hotspots
0

Xray


Test Repository for project TQS Game Rent
Test Cases BETA
Create
FOLDERS
TEST SETS
Test Repository 20 (20)
Test Repository
FOLDERS VIEW
Search
Only My Issues
Tests and Preconditions
Filters
Showing 20 of 20 entries
TGR-41 USER-3 GENERIC DONE
TGR-40 USER-2 GENERIC DONE
TGR-39 USER-1 GENERIC DONE
TGR-38 REVIEW-2 GENERIC DONE
TGR-37 REVIEW-1 GENERIC DONE
TGR-36 PAY-1 GENERIC DONE
TGR-35 ITEM-3 GENERIC DONE
TGR-34 ITEM-2 GENERIC DONE
TGR-33 ITEM-1 GENERIC DONE
TGR-32 IGDB-1 GENERIC DONE
TGR-31 BOOK-5 GENERIC DONE
TGR-30 BOOK-4

Reflexões

Surpresas Positivas	Desafios
● Workflow & QA Gates	● Atrito Inicial
● Testes E2E	● Rigidez Sonarqube

Links

Deploy

GAMERENT
PLAY • DISCOVER • UNLOCK FUN

http://deti-tqs-13.ua.pt:3000

Jira

http://ua-tqs-gamerent.atlassian.net