

MagicLook



Equipa:

Ivan Horoshko - 120603 (Team Coordinator & DevOps master)

Maria-Aleksandra Korjenevskaya - 118769 (Product owner)

Gonçalo Floro Garcia Ferreira - 120189 (QA Engineer)



Conceito do Produto

O produto é uma aplicação online que serve para alugar roupas de cerimónia, permitindo que utilizadores encontrem, reservem e devolvam roupas para eventos únicos, promovendo a reutilização e reduzindo o desperdício.

Personas



Camila (Staff)
Idade: 30 anos



Alice (Cliente)
Idade: 25 anos

Epics

Epic 1: “Reserva de Roupa”

Epic 2: “Gestão de Itens”

Epic 3: “Reporte de Danos”

Epic 4: “Perfis de Utilizadores”

Epic 5: “Pesquisa de Roupas”

Epic 6: “Entrada na Página Staff”

Epic 7: “Pagamento”

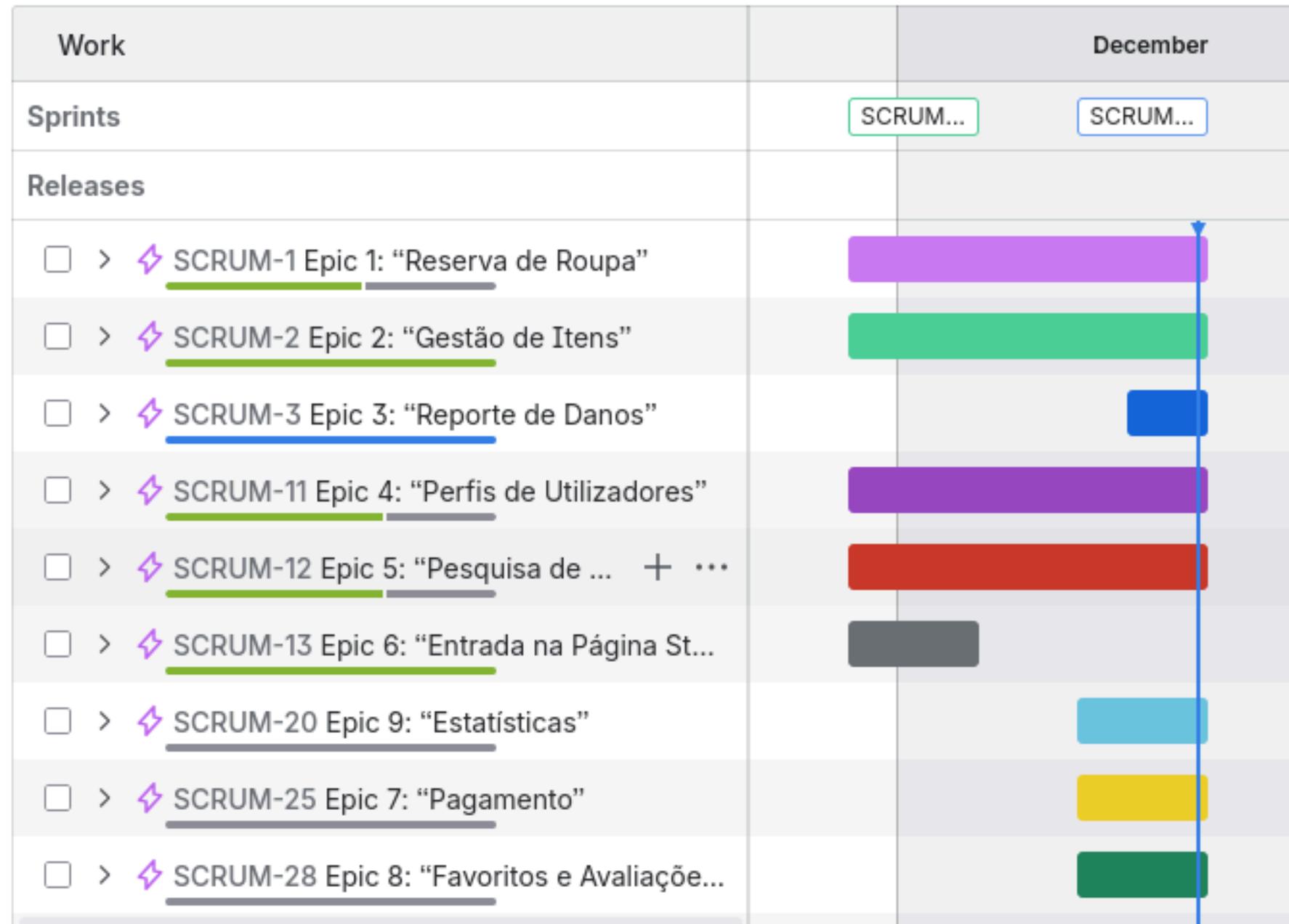
Epic 8: “Favoritos e Avaliações”

Epic 9: “Estatísticas”





Jira

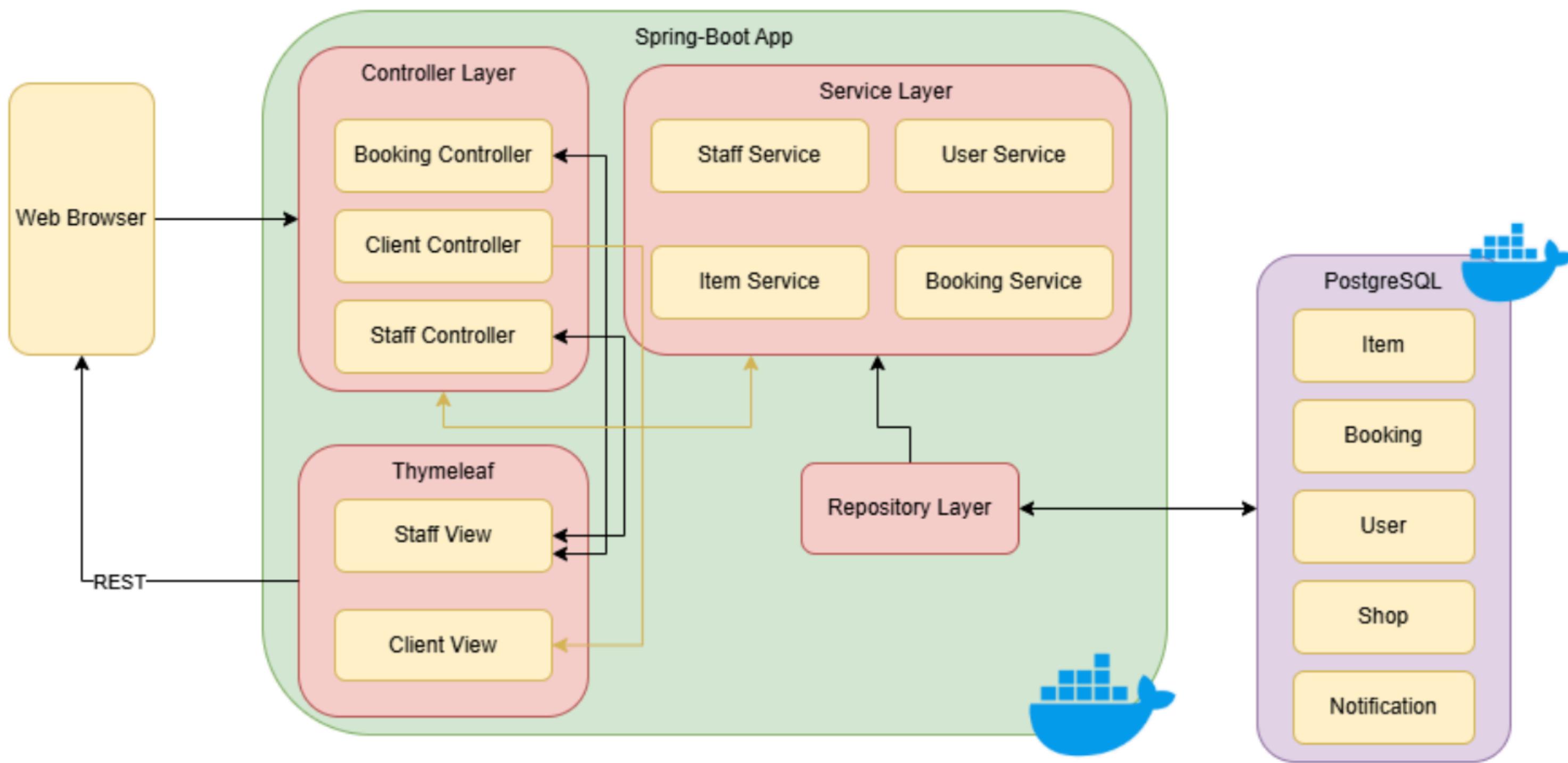


Resumo	Test Type	Estado	Status
GET /magiclook/staff/item → lists items for logged-in s...	Generic	A FAZER	PASSED
POST /magiclook/staff/itemsingle/update/{id} -> upda...	Generic	A FAZER	PASSED
DELETE /magiclook/staff/item/{id}/size/{size} -> delete...	Generic	A FAZER	PASSED
POST /magiclook/staff/item → with image saves to file...	Generic	A FAZER	PASSED
POST /magiclook/staff/item → success creates item an...	Generic	A FAZER	PASSED
em → not logged in redirects ...	Generic	A FAZER	PASSED
em/{id} -> update item	Generic	A FAZER	PASSED
em → invalid size returns error...	Generic	A FAZER	PASSED
n/{id} → shows item details fo...	Generic	A FAZER	PASSED
'item/{id}/size/{size}' -> delete...	Generic	A FAZER	PASSED

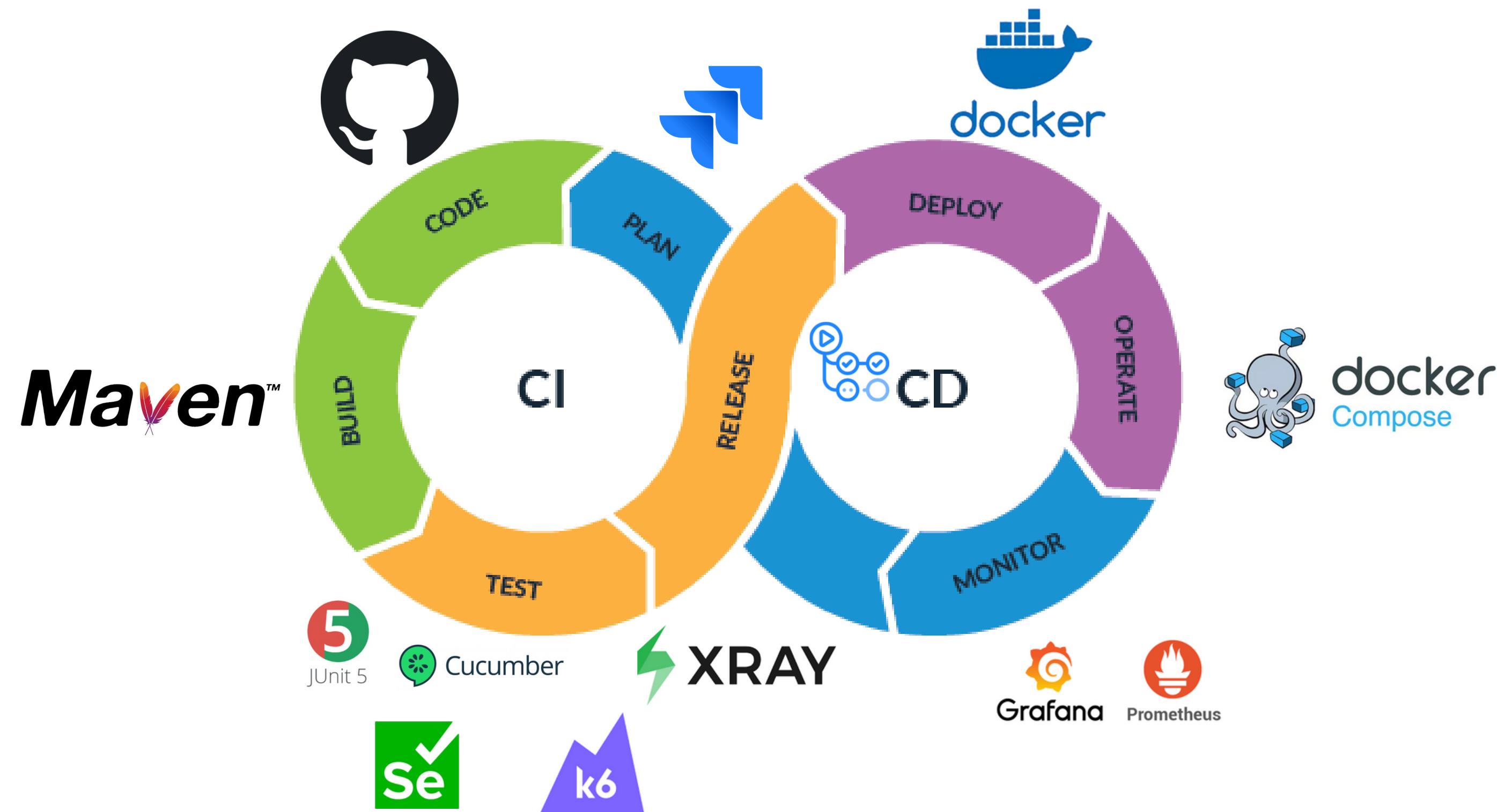
Xray



Arquitetura



CI/CD Pipeline

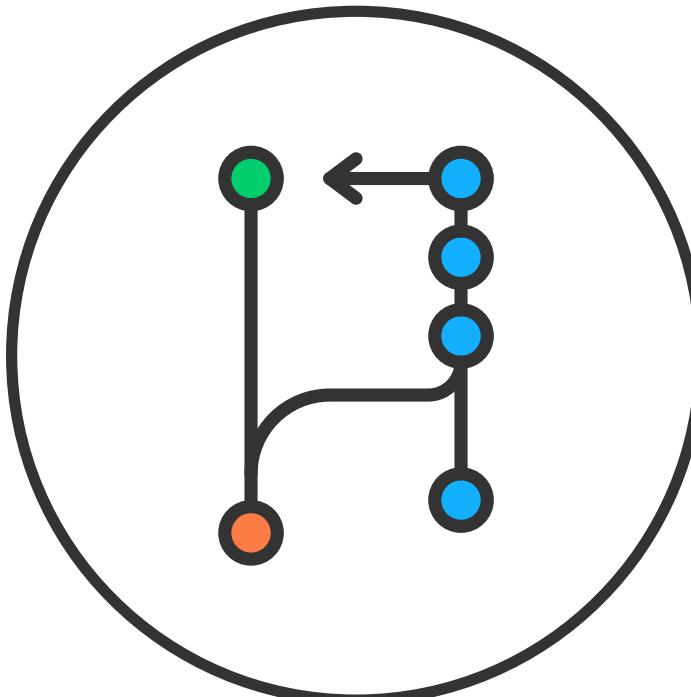




Regras de Github

Cada vez que se pretende adicionar uma nova funcionalidade, é criada uma nova branch com a seguinte convenção:

- SCRUM-[Nº do Jira]-[Nº histórica]-[Nome da história]



Template

: US[Nº da História]

[Nome da História]

[Descrição da História]

Critérios de Aceitação

- [] Critério 1
- [] Critério 2

Definição de Concluído

- [] Testes unitários passam
- [] Testes de integração passam
- [] Cobertura de código $\geq 80\%$
- [] Critérios Aceitação são cumpridos
- [] Passa o Quality Gate do SonarCloud
- [] Aprovado pelo menos por 1 membro da equipa

Os Pull requests têm de seguir um template e só podem ser aceites com a aprovação de outro membro da equipa, para além de só poderem ser aceites quando cumprirem todos os critérios de aceitação e todos os respetivos testes



Conjunto QA - backend

Testes unitários:

- Serviços e controladores
- JUnit5, Mockito

```
@Test
void testCreateBooking_ItemNotFound() {
    when(itemRepository.findById(bookingRequest.getItemId()))
        .thenReturn(Optional.empty());

    RuntimeException exception = assertThrows(RuntimeException.class, () -> {
        bookingService.createBooking(bookingRequest, testUser);
    });

    assertEquals("Item não encontrado", exception.getMessage());
    verify(itemRepository, times(1)).findById(bookingRequest.getItemId());
    verify(bookingRepository, never()).save(any(Booking.class));
}
```



Conjunto QA - backend

Testes de integração:

- H2 Database
- Spring Boot Test

```
@DisplayName("POST /magiclook/staff/item → success creates item and redirects")
void addItem_successCreatesItemAndRedirectsToDashboard() {
    String sessionCookie = loginAsStaff(seededUsername, seededPassword);

    // Check if exists
    Optional<Item> foundItem = itemRepository.findByAllCharacteristics("Vestido Azul", "Seda", "Azul",
        "Zara", "F", "Vestido", "Curto", seededShopId);

    // Get initial itemSingle instances for that item
    Integer initialCount = 0;
    List<ItemSingle> itemSingles = List.of();
    if (foundItem.isPresent()) {

        Item item = foundItem.get();

        itemSingles = itemSingleRepository.findByItem_ItemId(item.getItemId());
        initialCount = itemSingles.size();

    }

    // Make POST request
    ResponseEntity<String> response = restTemplate.exchange(
        url("/magiclook/staff/item"),
        HttpMethod.POST,
        multipartBody(false, null, sessionCookie),
        String.class);

    Assertions.assertThat(response.getStatusCode())
        .isEqualTo(HttpStatus.FOUND);
```



Conjunto QA - backend

Testes de Performance:

- K6

```
import { check, sleep } from 'k6';

const BASE_URL = 'http://localhost:8080/magiclook';

export const options = {
  stages: [
    { duration: '30s', target: 50 },
    { duration: '2m', target: 100 },
    { duration: '30s', target: 150 },
    { duration: '1m', target: 150 },
    { duration: '30s', target: 100 },
    { duration: '30s', target: 50 },
    { duration: '30s', target: 0 },
  ],
  thresholds: {
    http_req_duration: ['p(95)<1500'],
    http_req_failed: ['rate<0.03'],
  },
};

const USER_POOL = Array.from({ length: 1000 }, (_, i) => `testuser${i}`);
const ITEMS = [1, 2, 3, 4, 5];

export default function () {
  const userIndex = _VU * _ITER % USER_POOL.length;
  const username = USER_POOL[userIndex];

  // Login (assume users pre-created)
  const loginRes = http.post(` ${BASE_URL}/login`, {
    username: username,
    password: 'test123',
  }, {
    headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    redirects: 0,
  });
}
```



Conjunto QA - frontend

Testes de interface:
• Cucumber &
Selenium

The screenshot shows a user interface for managing items in a store. On the left, there's a dark sidebar with the 'MagicLook Staff' logo and several menu items: 'Admin' (selected), 'Porto', 'STAFF' (button), 'Dashboard', 'Itens da Loja' (selected, showing 9 items), 'Reservas' (showing 12 items), and 'Sair'. The main area is titled 'Itens da Loja' and has a sub-section 'Adicionar Novo Item'. The form fields are as follows:

- Género da Roupa:** Seleccione o género...
- Tamanho:** Seleccione o tamanho...
- Nome:** Ex: Vestido Rosa
- Marca:** Ex: Zara
- Material:** Seleccione o material...
- Cor:** Ex: Rosa
- Tipo de Roupa:** Seleccione primeiro o género...
- Subtipo:** Seleccione o subtipo...
- Preço Original (€):** 0.00
- Preço de Aluguel (€):** 0.00
- Imagen do Item:** (File input field)

A blue button at the top right says '+ Adicionar Item'. In the background, there's a blurred image of a woman wearing a patterned dress.



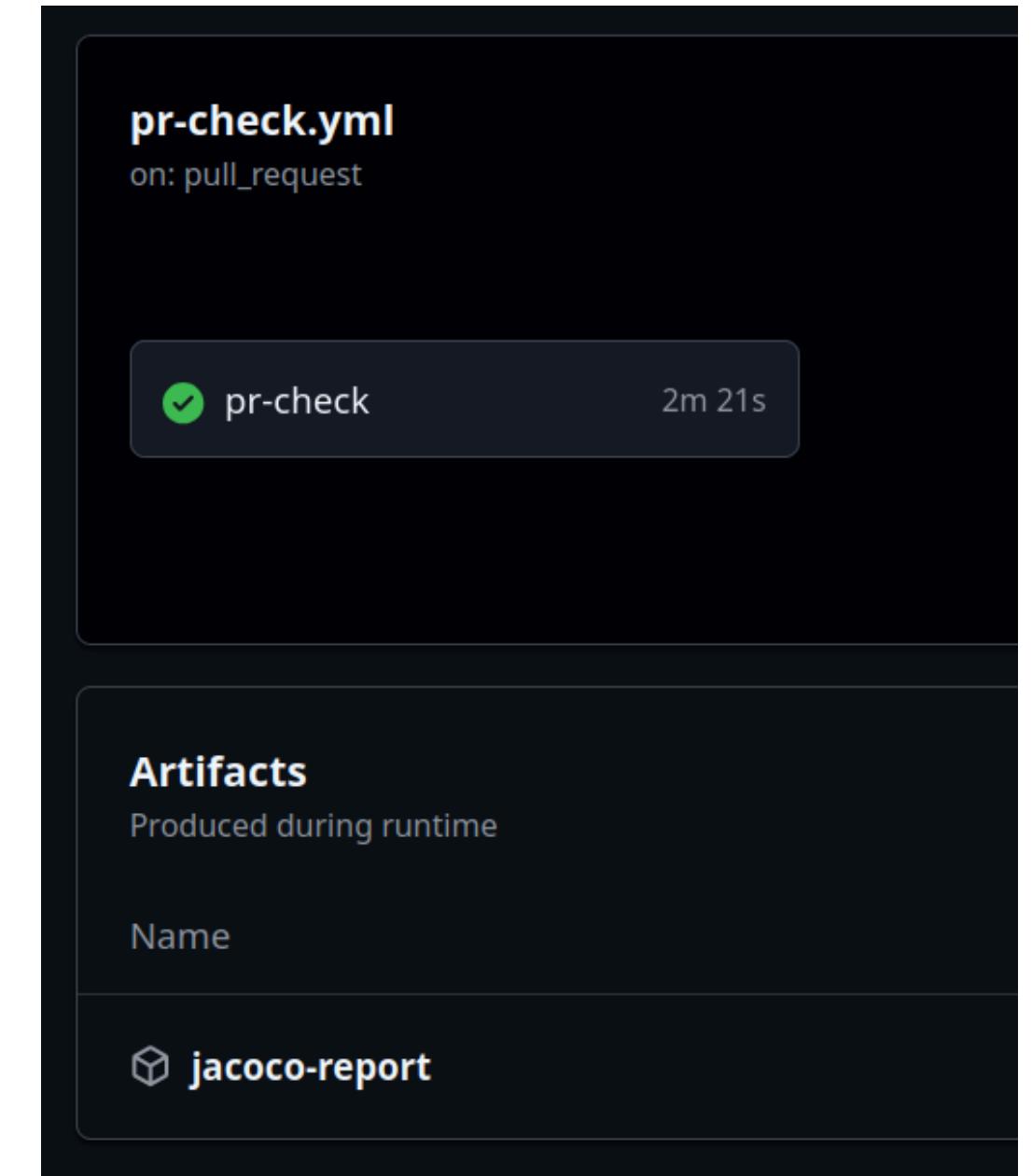
Implementação da QA

Workflow(CI/CD):

- Trigger: Em todos os Pull Request
- Backend: testes unitários e de integração
- Relate de todos os resultados no Jira(Xray)

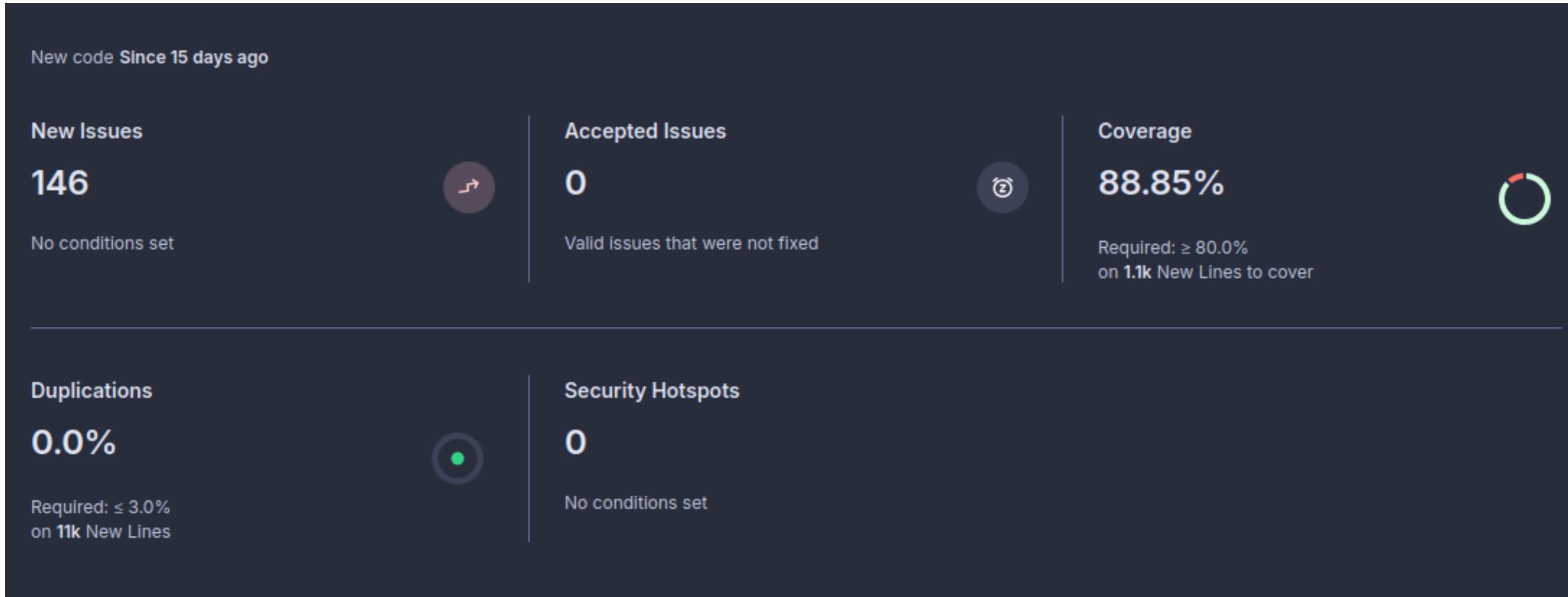
Quality gates & Acceptance criteria:

- Testes: todos passam(unitários e de integração)
- Cobertura de SonarQube: 80%





SonarQube



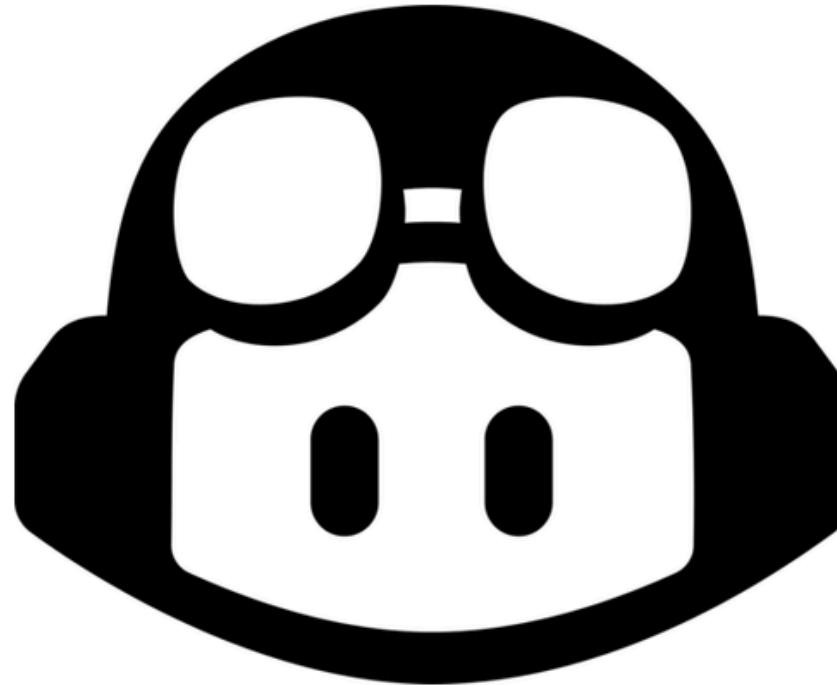


Grafana & Prometheus





Ferramentas de AI



- Resolução de problemas mais rápido: redução do tempo de debugging ao corrigir rapidamente problemas de segurança e qualidade
- Automatização de geração de testes: ajuda na integração de testes com coverage consistente



Demo

<http://deti-tqs-20.ua.pt>



Revisão do Projeto

Pontos fortes	Pontos fracos	Trabalho futuro
Boa definição do produto e boa organização do projeto	Não cumprimos todos os objetivos que tínhamos definido	Transferência de itens entre lojas
Boa definição de protocolo QA	Falta de testes frontend	Estatísticas, favoritos, marcação de experimentação de roupa
	Definimos muitas tarefas complexas	Aplicação de multas a utilizadores com entregas em atraso



Lições Aprendidas

Importância da automatização de trabalho

Importância de uma boa organização tanto de
equipa como do projeto