PHIHUB - MEDICAL MANAGEMENT APP

TQS GROUP PROJECT

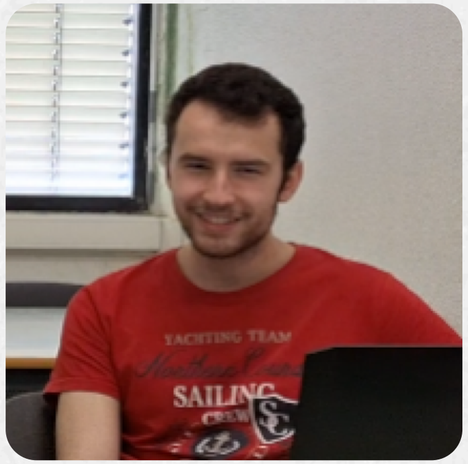
PHIHUB

Daniel Madureira [107603]
João Luís [107403]
Pedro Ramos [107348]
Rodrigo Aguiar [108969]

OUR TEAM

Daniel Madureira

QA Engineer



Rodrigo Aguiar

Team Manager

Nuno Luís

Product Owner

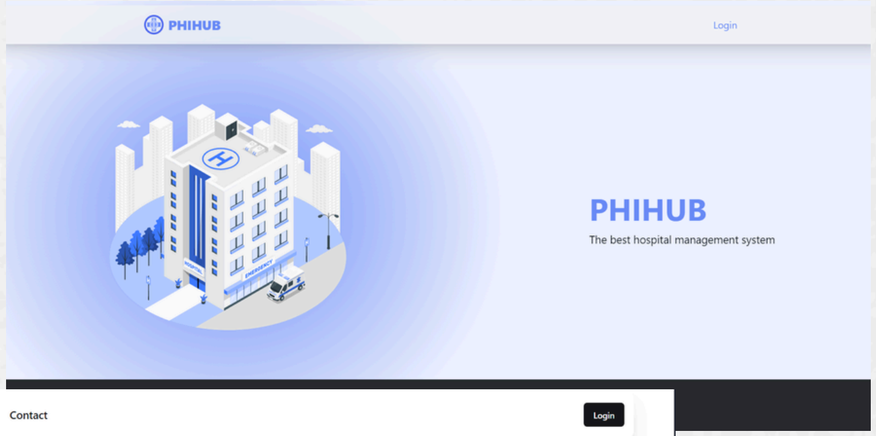
Pedro Ramos

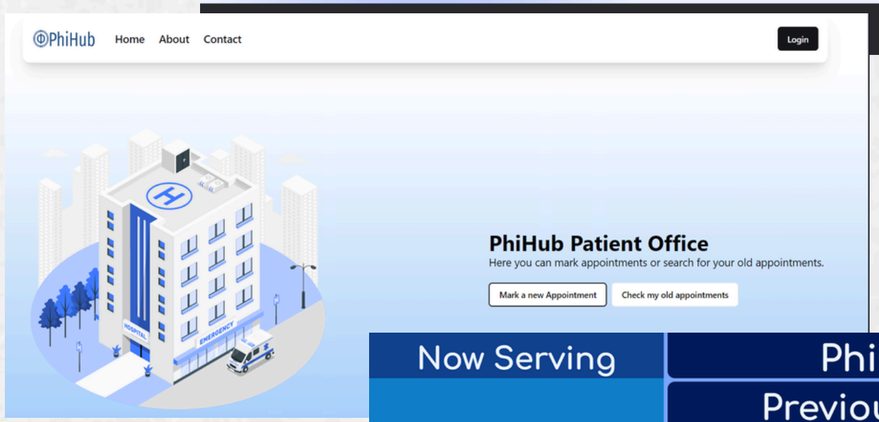
DevOps Master

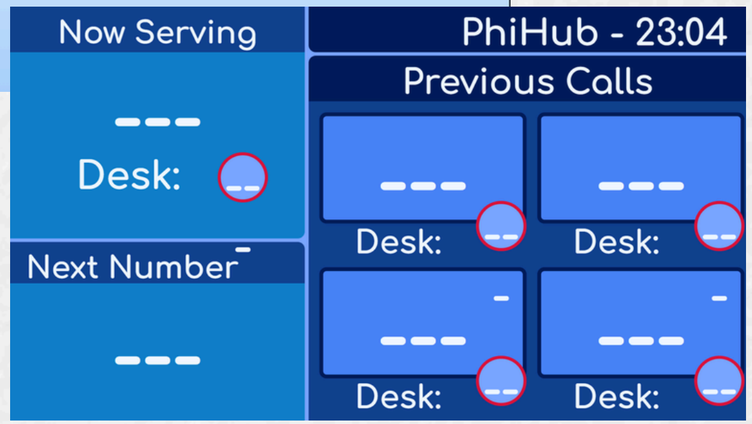
PRODUCT CONCEPT

An integrated ICT solution to handle:

- Patient’s appointments;
- Appointment’s management;
- Staff management;
- Medic’s creation;
- Queue lines and priority lines.



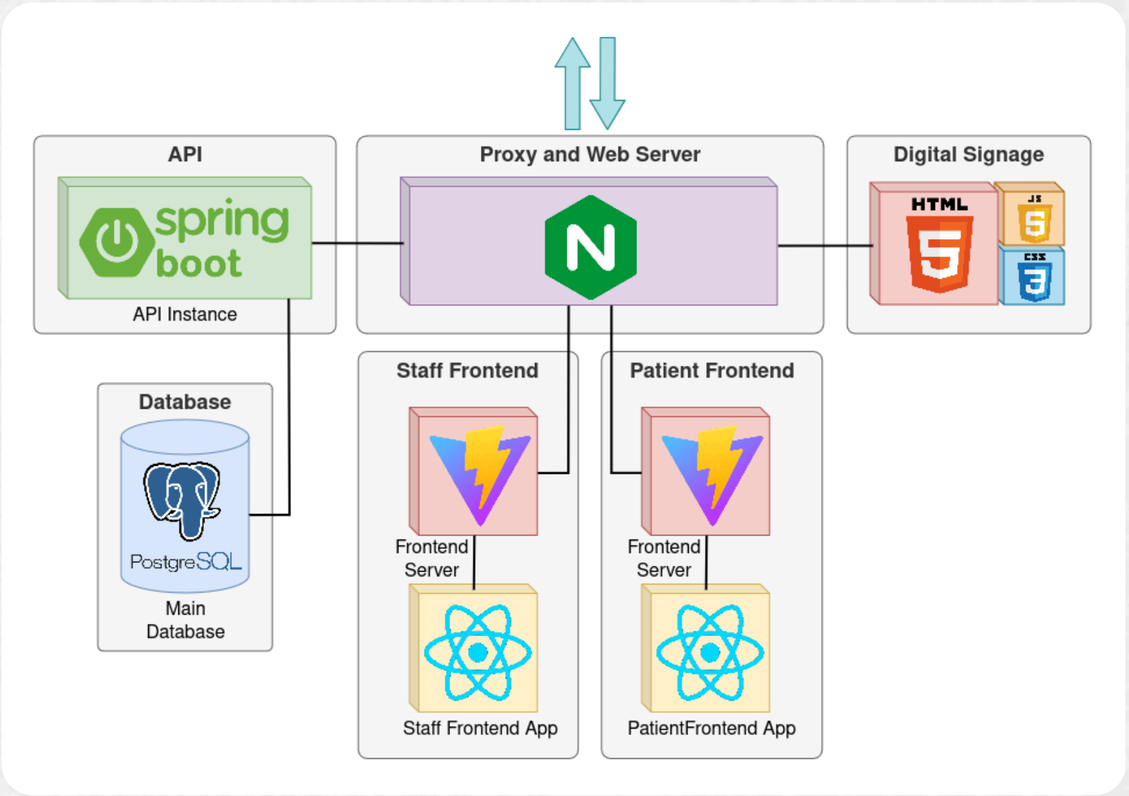




ARCHITECTURE

The final system consists of:

- A NGINX Proxy and Web Server;
- A general Backend;
- A Patient oriented Frontend for clients;
- A Staff oriented Frontend for medics and staff personnel;
- A Digital Signage for waiting room screens.



TOOLS

JIRA

Issue Tracker

Integrated with Xray to import automatic test results to JIRA

GITHUB

Remote Code Repository

Various components organized into submodules with a main repository, which includes all submodules.

Project Management

Agile Development with Weekly Sprints

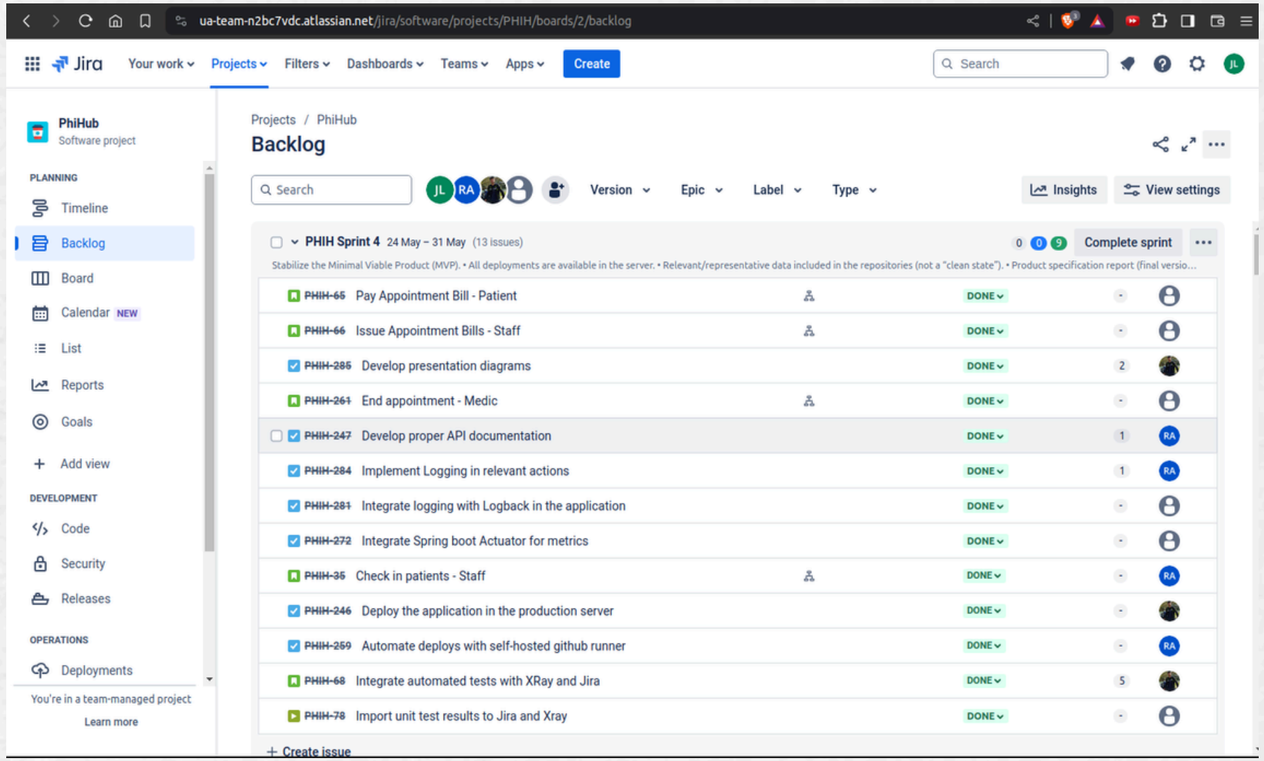
Define what user stories we will work on

Create subtasks for each user story in JIRA

For each subtask create a Feature Branch

Develop the wanted code and tests

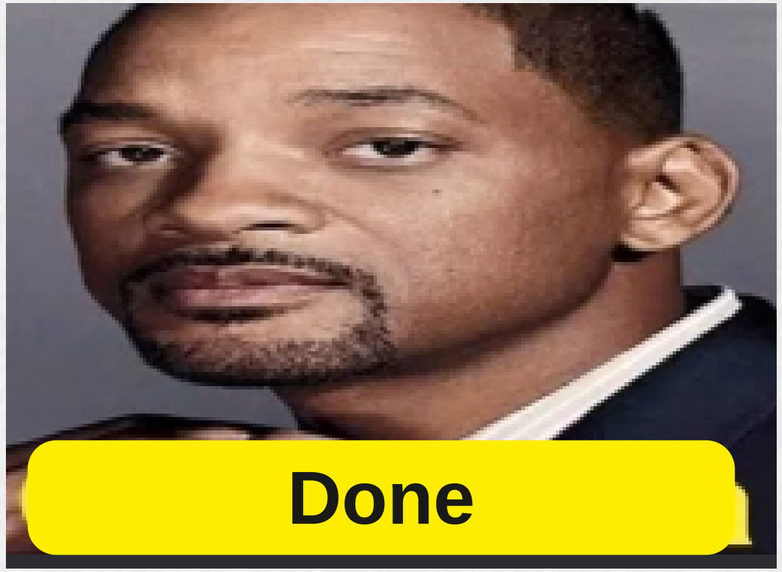
Create a pull request to the main branch of the submodule



Definition of Done

To define if a user story is done we agree to the following rules:

- Developed solution meets all user story requirements
- Supplied with automatic unit, integration and functional tests
- Test coverage of at least 80%
- Code reviewed by at least 1 other team member
- No bugs or security issues



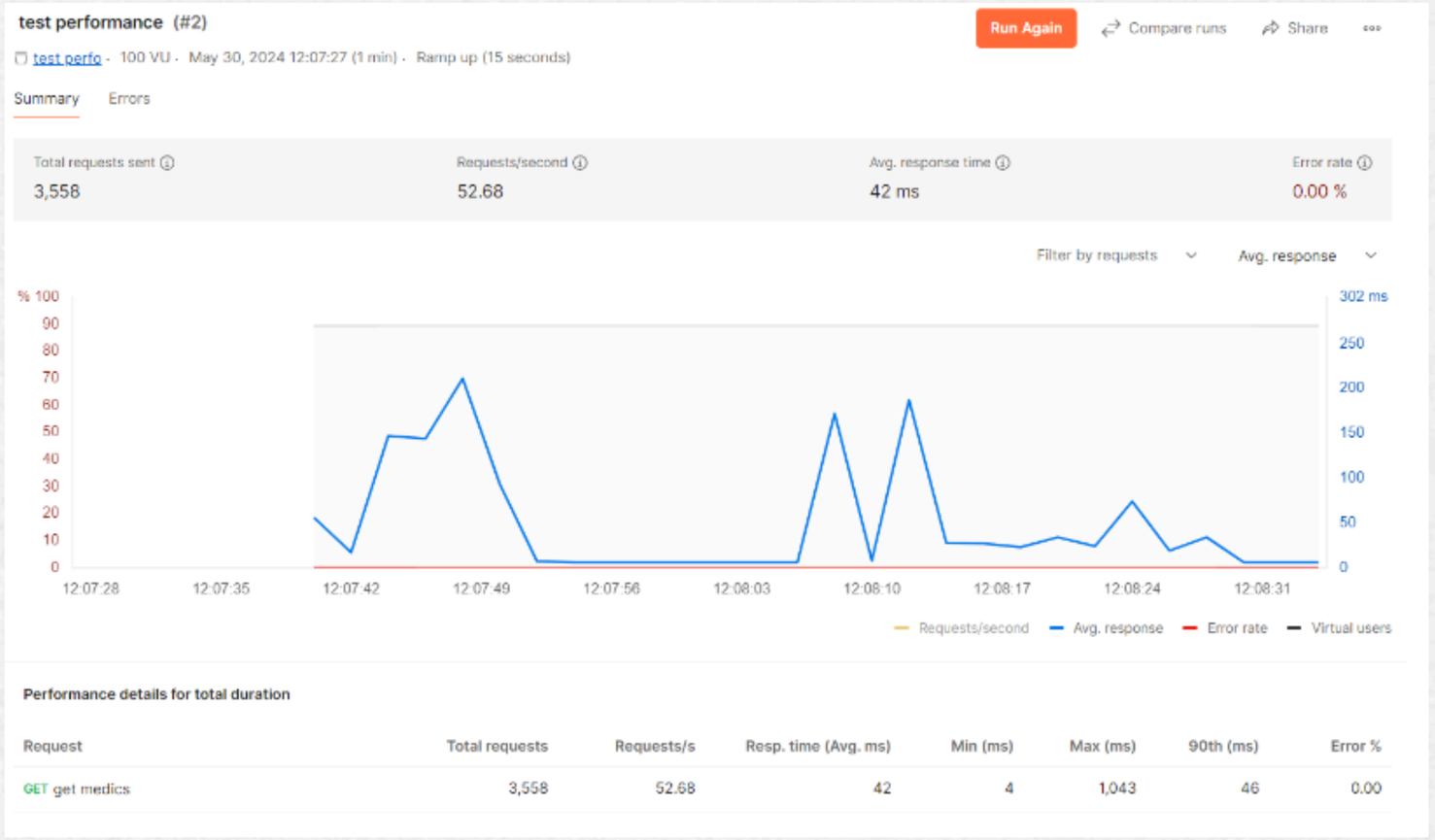
Testing Plan

For general testing, unit tests were developped to cover:

- The repositories;
- The services;
- The controllers;
- The full integrated backend;

Aditionally, frontend and full application tests were created to allow testing the most common user workflows, as to ensure the integration of the systems and the frontend application work properly.

Performance Testing



Simulated load with 100 virtual users

Total of 3558 requests in 1 minute

Average of 52 requests per second

Average response time: 42 ms

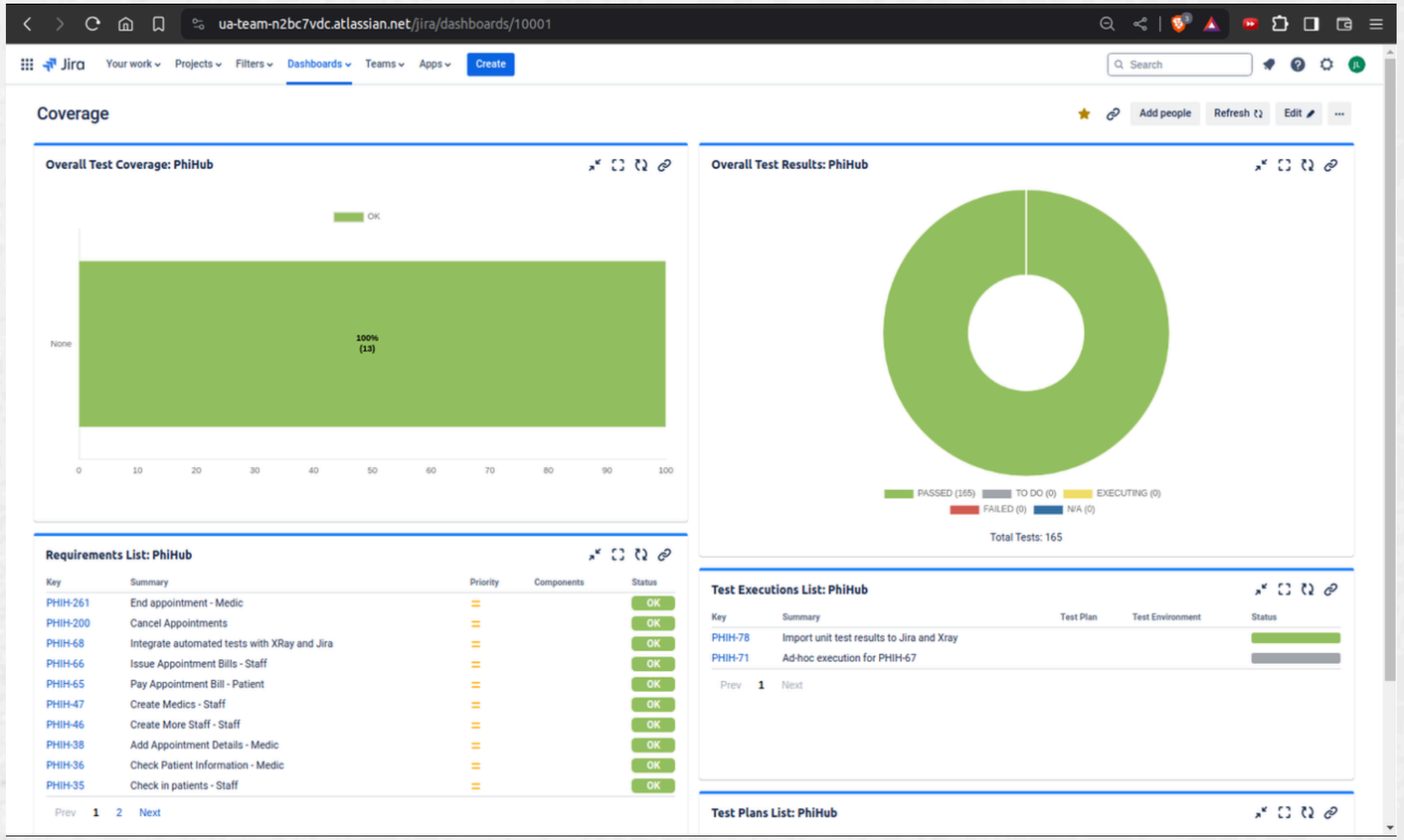
Quality Assurance

X-ray - Native Test Management for Jira;

Grouped test in test sets;

Linked various test sets to Issues;

All issues covered and all tests passed;



CI / CD

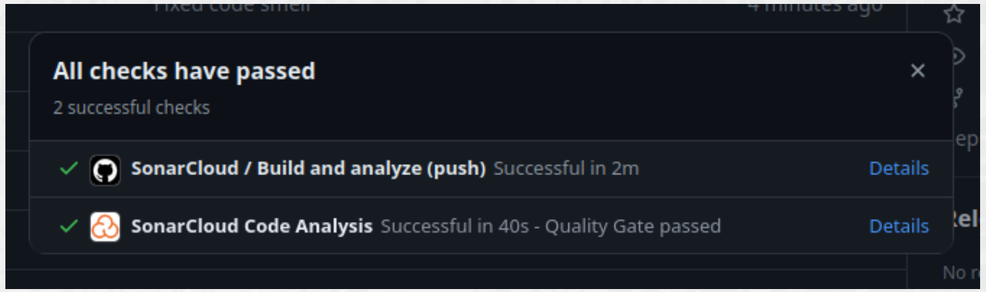
AUTOMATIC TESTING

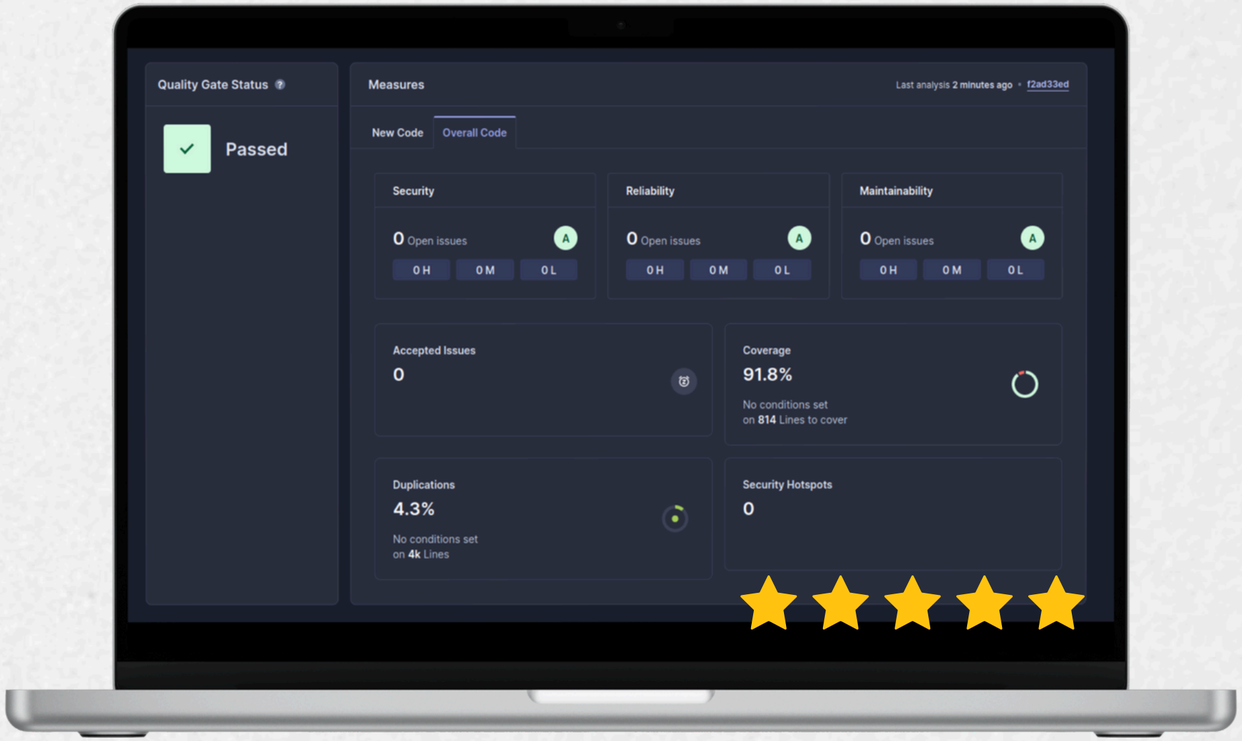
SONARCLOUD ANALYSIS

On every push or merge, the implemented tests are executed automatically to ensure that no broken or bad changes go to production.

The results are then published to SonarCloud, a static code and test result analyzer that makes it easy to check for problematic code changes.

Sonarcloud code analysis is done in both frontend and backend components.

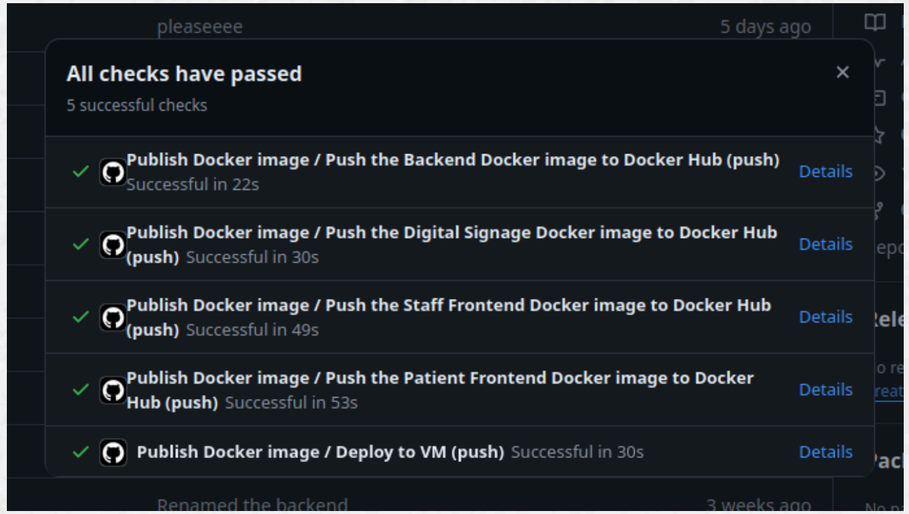




CI / CD

AUTOMATIC DEPLOYMENT

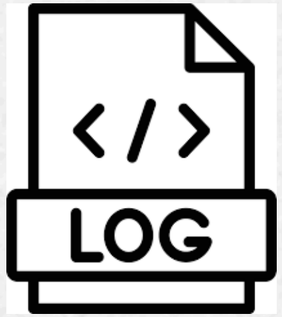
- Build and push Docker Images to DockerHub
- Self Hosted Github Runner running on the Virtual Machine
- Pull the latest images from DockerHub and build them




```yaml
deploy:
  runs-on: self-hosted
  name: "Deploy to VM"
  needs: [build-backend-image, build-signage-image, build-staff-image, build-patient-image]

  steps:
  - name: "Checkout"
    uses: actions/checkout@v4
  - name: "Deploy"
    run: |
        echo "CREATE DATABASE phihub;" > init.sql
        docker compose -f docker-compose.prod.yml pull
        docker compose -f docker-compose.prod.yml up -d --remove-orphans
        docker container prune -f
        docker image prune -af
        docker builder prune -af
```

System Observability



Logging

Relevant action logging with Logback

Logging to both console and file

Healthcheck

API Healthcheck with Spring Actuator

/actuator/health endpoint

Log analysis

Various attempts were made at integrating Spring with an ELK Stack

Unfortunately, they were unsuccessful =/

DEMO

THANKS!

Do you have any questions?

deti-tqs-18.ua.pt