

LAB Quarkus

Desenvolvendo um Sistema para Eleição
usando Quarkus Framework

Thiago Poiani



[linkedin.com/in/thpoiani](https://www.linkedin.com/in/thpoiani)

Vídeo 01 - Orientações Gerais

Visão Geral e Expectativas da Aula

Plano de Aula

Planejamento do Escopo

Definição de Escopo, Proposta Arquitetural, Proposta Técnica

Plano de Aula

github.com/thpoiani/lab-quarkus

 Watch ▼

LEITURA RECOMENDADA

INTERAÇÃO

Planejamento do Escopo

Definição do Escopo

Candidatos são listados, cadastrados e editados

Todos os candidatos registrados participam de uma eleição, quando for iniciada

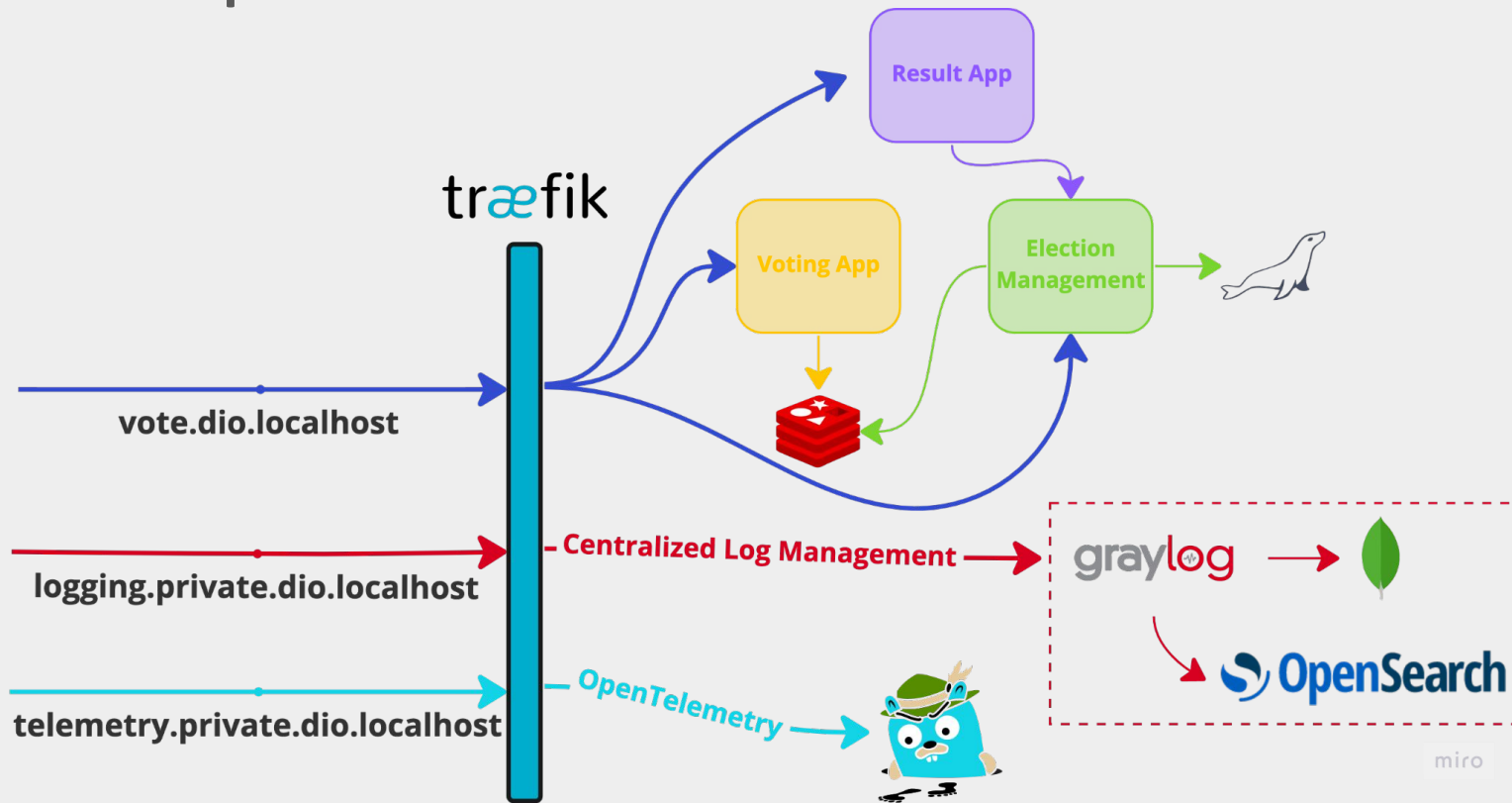
Candidatos recebem votos de eleitores

Resultado disponível em *tempo real*

Requisitos não funcionais: confiabilidade, disponibilidade, interoperabilidade, escalabilidade

Planejamento do Escopo

Proposta Arquitetural



Planejamento do Escopo

Proposta Técnica - Containerização

Docker

Docker é uma plataforma de **virtualização de contêineres** que permite que os desenvolvedores empacotem e distribuam aplicativos e serviços como contêineres, fornecendo um ambiente isolado e padronizado para que esses aplicativos possam ser executados em qualquer ambiente de hospedagem.

Ao contrário da virtualização tradicional, onde uma máquina virtual completa é criada para executar um aplicativo, o Docker virtualiza o sistema operacional em nível de processo, permitindo que vários contêineres compartilhem o mesmo kernel do sistema operacional do host.

Cada contêiner do Docker inclui todos os componentes necessários para executar um aplicativo, incluindo código, bibliotecas, dependências e configurações. Isso torna o processo de implantação e execução de aplicativos muito mais fácil e eficiente, pois os desenvolvedores podem criar e testar contêineres localmente em suas máquinas de desenvolvimento e, em seguida, implantá-los em qualquer ambiente de hospedagem que execute o Docker.

O Docker é amplamente utilizado por desenvolvedores e operadores de infraestrutura para construir, empacotar e implantar aplicativos em escala. Ele permite que as equipes de desenvolvimento e operações colaborem em um ambiente padronizado, eliminando muitos dos desafios comuns de implantação e gerenciamento de aplicativos em escala.

Planejamento do Escopo

Proposta Técnica - Containerização

Docker Compose

Docker Compose é uma ferramenta que permite **definir e executar aplicativos multi-contêiner em um único arquivo de configuração**. Ele é usado para orquestrar a implantação de vários contêineres Docker como parte de um único aplicativo, definindo as dependências e configurações entre eles.

Com o Docker Compose, os desenvolvedores podem criar, implantar e gerenciar aplicativos compostos por vários contêineres de forma fácil e eficiente. Isso é especialmente útil para aplicativos complexos que exigem a execução de vários serviços, como bancos de dados, servidores web, servidores de aplicativos e muito mais.

O Docker Compose permite que os desenvolvedores definam a configuração de cada contêiner em um arquivo YAML, incluindo as imagens do contêiner, as variáveis de ambiente, as portas expostas, os volumes e as dependências entre os contêineres. Em seguida, ele pode ser usado para construir, iniciar e parar todos os contêineres necessários para executar o aplicativo de forma fácil e automática.

O Docker Compose é amplamente utilizado por desenvolvedores e equipes de operações de infraestrutura que trabalham com aplicativos compostos por vários serviços e que desejam simplificar o processo de desenvolvimento, teste e implantação. Ele também é amplamente suportado por plataformas de nuvem, como o Amazon Web Services (AWS), o Microsoft Azure e o Google Cloud Platform (GCP).

Planejamento do Escopo

Proposta Técnica - Tráfego

Traefik

Traefik é um software de **roteamento de tráfego** e **balanceamento de carga** projetado para ser usado em **ambientes de contêineres**, como o Docker. Ele atua como um **proxy reverso** que pode rotear solicitações HTTP e TCP para diferentes serviços em seu cluster de contêineres, garantindo que o tráfego chegue ao serviço correto com base em regras de roteamento configuráveis.

Traefik é projetado para ser fácil de configurar e pode detectar automaticamente novos serviços conforme eles são adicionados ou removidos do seu ambiente de contêineres. Ele também suporta integrações com vários provedores de orquestração de contêineres, como o **Kubernetes**, o Docker Swarm e o Mesos.

Entre seus recursos, o Traefik suporta balanceamento de carga, SSL/TLS com renovação automática, autenticação e autorização, gerenciamento de tráfego em várias regiões geográficas e muito mais. Graças a esses recursos, o Traefik é uma escolha popular para empresas que executam aplicativos em ambientes de contêineres e precisam garantir que o tráfego seja distribuído de forma eficiente e segura.

Planejamento do Escopo

Proposta Técnica - Armazenamento

MariaDB

MariaDB é um **sistema de gerenciamento de banco de dados relacional** (RDBMS) que foi criado a partir do código-fonte do MySQL, após sua aquisição pela Oracle Corporation em 2009. O nome "MariaDB" vem do nome da filha mais nova do fundador do MySQL, Michael Widenius.

O MariaDB é compatível com a API do MySQL e oferece recursos adicionais, como suporte para vários motores de armazenamento, incluindo o XtraDB, que é uma versão aprimorada do InnoDB, e o Aria, que é um motor de armazenamento transacional de alto desempenho.

O MariaDB é distribuído como software livre e de código aberto, sob os termos da licença GPL. Ele é amplamente utilizado por empresas e organizações em todo o mundo, incluindo grandes empresas como Google, Wikipedia, Red Hat e IBM.

Entre seus recursos, o MariaDB oferece suporte para transações ACID, replicação de banco de dados em tempo real, particionamento de tabela, clustering de banco de dados e muitos outros recursos que tornam o gerenciamento de bancos de dados em escala muito mais fácil e eficiente. Além disso, o MariaDB é altamente escalável e pode ser facilmente implantado em ambientes de nuvem, contêineres e ambientes de alta disponibilidade.

Planejamento do Escopo

Proposta Técnica - Armazenamento

Redis

Redis é um **sistema de armazenamento de chave-valor em memória** de código aberto que é amplamente usado como **banco de dados, cache e broker de mensagens**. Ele é projetado para ser extremamente rápido e escalável, oferecendo tempos de resposta de leitura/gravação em memória muito baixos.

Redis é um banco de dados NoSQL que armazena dados na memória, permitindo que ele seja muito rápido e eficiente para operações de leitura e gravação. Além disso, o Redis possui recursos avançados de armazenamento em cache, como a expiração de chaves, o armazenamento de dados em disco e o suporte para operações de lista, conjunto, hash e ordenação.

O Redis é amplamente utilizado em aplicações que exigem alta velocidade de processamento de dados, como mídias sociais, jogos, análise de dados, e-commerce e muito mais. Além disso, ele também é usado como cache em aplicativos da web para reduzir a carga em bancos de dados e melhorar o desempenho de consultas.

O Redis é distribuído como software livre e de código aberto e oferece suporte a vários sistemas operacionais, incluindo Linux, macOS e Windows. Ele pode ser facilmente integrado com várias linguagens de programação, como Python, Java, Ruby e muitas outras.

Planejamento do Escopo

Proposta Técnica - Observabilidade Gerenciamento Centralizado de Logs

Graylog

Graylog é uma **plataforma de gerenciamento de logs** e **análise de dados em tempo real** de código aberto que ajuda as empresas a coletar, armazenar, pesquisar e analisar dados de log de várias fontes em um único local. Ele oferece uma interface de usuário baseada na web que permite visualizar e analisar dados de log em tempo real, o que ajuda a solucionar problemas e identificar tendências rapidamente.

O Graylog é altamente escalável e pode ser facilmente integrado com várias fontes de dados, incluindo syslog, **GELF**, JSON, CEF e muitos outros. Ele também suporta a análise de dados não estruturados, como logs de aplicativos, eventos de rede e mensagens de IoT. O Graylog permite que os usuários definam alertas para eventos importantes, criem painéis personalizados e visualizem dados em gráficos e tabelas.

Além disso, o Graylog oferece recursos avançados, como indexação de dados em tempo real, busca rápida, clusterização e suporte a plugins personalizados. Ele pode ser implantado em nuvens públicas, privadas ou híbridas, bem como em contêineres e ambientes de alta disponibilidade.

O Graylog é amplamente utilizado por empresas em todo o mundo, incluindo grandes organizações como IBM, SAP e SoundCloud. Ele é distribuído como software livre e de código aberto, sob os termos da licença GNU General Public License v3.

Planejamento do Escopo

Proposta Técnica - Observabilidade Gerenciamento Centralizado de Logs

OpenSearch

OpenSearch é uma **plataforma de busca e análise de dados em tempo real** de código aberto, baseada no projeto **Elasticsearch**, que permite aos usuários pesquisar e analisar grandes quantidades de dados estruturados e não estruturados. É uma alternativa totalmente compatível e gratuita do Elasticsearch, criada por um grupo de empresas liderado pela AWS.

OpenSearch oferece recursos avançados de busca e análise de dados em tempo real, como pesquisa de texto completo, pesquisa geoespacial, análise de dados em tempo real, análise de séries temporais e pesquisa de agregação. Ele é projetado para lidar com grandes volumes de dados e escalabilidade horizontal, permitindo que os usuários adicionem e dimensionem nodes conforme necessário.

O OpenSearch é compatível com vários clientes e bibliotecas, incluindo o Kibana, Logstash e Beats. Ele pode ser usado para uma ampla gama de casos de uso, como pesquisa e análise de registros de aplicativos, análise de segurança, análise de rede e muito mais.

O OpenSearch é distribuído como software livre e de código aberto sob os termos da licença Apache 2.0. Ele pode ser executado em várias plataformas, incluindo Linux, macOS e Windows, além de ser suportado por várias ferramentas de gerenciamento de contêineres, como o Docker e Kubernetes.

Planejamento do Escopo

Proposta Técnica - Observabilidade Gerenciamento Centralizado de Logs

MongoDB

MongoDB é um **banco de dados NoSQL** (não relacional) de código aberto, **orientado a documentos**, projetado para lidar com grandes volumes de dados, fornecendo alta escalabilidade e desempenho. Ele é uma alternativa aos bancos de dados relacionais tradicionais, como MySQL, PostgreSQL e Oracle.

No MongoDB, os dados são armazenados em documentos no formato JSON (JavaScript Object Notation) e, ao contrário dos bancos de dados relacionais, o MongoDB não requer um esquema definido previamente para armazenar dados. Isso permite uma grande flexibilidade para modelar dados complexos, pois cada documento pode ter sua própria estrutura e campos variáveis.

O MongoDB suporta várias operações de consulta e análise de dados, incluindo consultas ad-hoc, agregação de dados, indexação e pesquisa geoespacial. Ele também oferece recursos avançados, como replicação automática, sharding e balanceamento de carga para escalabilidade horizontal, e suporta várias linguagens de programação, incluindo Python, Java, Ruby e muitas outras.

O MongoDB é amplamente utilizado em aplicativos da web, aplicativos móveis, jogos, mídias sociais e muitas outras áreas em que é necessária a capacidade de processar grandes volumes de dados de forma rápida e escalável. Ele é distribuído como software livre e de código aberto sob os termos da licença Server Side Public License (SSPL).

Planejamento do Escopo

Proposta Técnica - Observabilidade Rastreamento

Jaeger Tracing

Jaeger Tracing é uma **plataforma de rastreamento** de distribuição de código aberto que ajuda a monitorar e depurar aplicativos distribuídos e microserviços. Ele fornece visibilidade sobre como as solicitações são processadas em diferentes serviços e em diferentes camadas de um aplicativo, permitindo que os desenvolvedores identifiquem gargalos de desempenho e otimizem a carga de trabalho do aplicativo.

O Jaeger foi desenvolvido originalmente pelo Uber e agora é um projeto de código aberto da Cloud Native Computing Foundation (CNCF), o que significa que é amplamente adotado pela comunidade de desenvolvedores de software.

O Jaeger suporta vários protocolos, incluindo o OpenTracing, que é um padrão de rastreamento distribuído, e pode ser integrado com várias bibliotecas de código aberto e linguagens de programação. Ele oferece recursos como a geração de rastreamentos de solicitações, a coleta de métricas de desempenho e a visualização de dados em um painel interativo baseado na web.

Ao fornecer informações detalhadas sobre o fluxo de solicitações em um aplicativo distribuído, o Jaeger permite que os desenvolvedores identifiquem rapidamente problemas de desempenho e depurem problemas em um ambiente de produção. Isso ajuda a melhorar a experiência do usuário e reduzir o tempo de inatividade do aplicativo.

Planejamento do Escopo

Proposta Técnica - Linguagem de Desenvolvimento

Java

Java é uma **linguagem de programação** de alto nível, orientada a objetos e multiplataforma, criada pela Sun Microsystems (adquirida posteriormente pela Oracle Corporation) nos anos 90. Ela foi projetada para ser simples, portátil e segura, permitindo que os desenvolvedores escrevam código uma vez e o executem em diferentes plataformas, como Windows, Linux e macOS.

Uma das principais características do Java é sua portabilidade. Os programas Java são compilados em um formato intermediário chamado bytecode, que pode ser executado em qualquer máquina virtual Java (JVM) compatível. Isso significa que os aplicativos Java podem ser executados em diferentes sistemas operacionais sem a necessidade de recompilar o código-fonte.

Além disso, o Java é conhecido por ser uma linguagem robusta e segura. Ele inclui recursos avançados, como coleta de lixo, verificação de tipos e gerenciamento automático de memória, que ajudam a evitar problemas comuns de programação, como vazamentos de memória e erros de ponteiro.

O Java é amplamente utilizado para desenvolvimento de aplicativos empresariais, jogos, aplicativos móveis, sistemas embarcados e muito mais. Ele suporta uma ampla gama de bibliotecas e frameworks que tornam o desenvolvimento de aplicativos mais fácil e eficiente.

O Java continua sendo uma das linguagens de programação mais populares e amplamente utilizadas no mundo, com uma grande comunidade de desenvolvedores e uma ampla gama de ferramentas e recursos disponíveis para os desenvolvedores.

Planejamento do Escopo

Proposta Técnica - Linguagem de Desenvolvimento

JDK

Java Development Kit é um conjunto de **ferramentas de desenvolvimento** de software usadas para criar aplicativos Java. Ele contém um conjunto de bibliotecas, compiladores, depuradores e outras ferramentas para desenvolver, testar e depurar programas em Java. O JDK inclui o JRE (Java Runtime Environment), que é usado para executar aplicativos Java em um ambiente de produção. O JDK é necessário para desenvolver aplicativos Java e é distribuído gratuitamente pela Oracle, a empresa que atualmente mantém e desenvolve a linguagem de programação Java.

<https://whichjdk.com/>



<https://sdkman.io/>



Planejamento do Escopo

Proposta Técnica - Framework

Quarkus

Quarkus é um **framework** de código aberto para **desenvolvimento de aplicativos Java** criado com foco em eficiência de recursos e tempos de inicialização ultrarrápidos. Ele foi projetado para ajudar os desenvolvedores a criar aplicativos Java de alta performance e escaláveis, especialmente para a nuvem.

O Quarkus utiliza tecnologias modernas, como a plataforma **GraalVM** e o Kubernetes, para criar aplicativos Java altamente otimizados que consomem menos recursos e têm tempos de inicialização muito rápidos. Ele é construído em cima do Java Standard Edition (SE) e suporta vários frameworks, como o Eclipse MicroProfile e o Spring.

Com o Quarkus, os desenvolvedores podem criar aplicativos Java com recursos avançados, como suporte a serviços web RESTful, reatividade, segurança, monitoramento de métricas e muito mais. Ele oferece um conjunto completo de ferramentas para desenvolvimento, teste e implantação de aplicativos, incluindo suporte para ambientes de contêiner e nuvem.

O Quarkus é uma solução ideal para desenvolvedores que buscam construir aplicativos Java modernos, rápidos e escaláveis para ambientes de nuvem. Ele é mantido pela Red Hat e está disponível como software livre e de código aberto sob a licença Apache 2.0.

Planejamento do Escopo

Proposta Técnica - Framework

Quarkus - Java Native Image

Java Native Image é uma ferramenta do Quarkus que permite a **criação de executáveis nativos a partir de aplicativos Java**. Esses executáveis são criados a partir do bytecode Java, mas são compilados para código nativo da plataforma específica, permitindo que o aplicativo seja executado diretamente na máquina host sem a necessidade de uma máquina virtual Java (JVM).

A principal vantagem do Java Native Image é a **redução significativa do tempo de inicialização do aplicativo** e o **menor consumo de memória** em comparação com a execução do aplicativo em uma JVM. Isso é particularmente útil em ambientes de contêiner, onde os recursos são limitados e a escalabilidade é essencial.

Além disso, a compilação do código Java em código nativo pode melhorar a segurança do aplicativo, uma vez que reduz o tamanho da superfície de ataque e a exposição a vulnerabilidades de segurança conhecidas da JVM.

O Java Native Image é suportado pelo Quarkus e é construído sobre a plataforma **GraalVM**, que é uma máquina virtual Java de alta performance que inclui um compilador de código nativo. O Quarkus fornece uma integração simplificada com o Java Native Image, permitindo que os desenvolvedores criem facilmente executáveis nativos a partir de seus aplicativos Java.

Planejamento do Escopo

Proposta Técnica - Testabilidade

JUnit

JUnit é uma biblioteca de **teste de unidade** para a linguagem de programação Java. Ele fornece um framework simples para escrever e executar testes de unidade automatizados em aplicativos Java. O JUnit permite que os desenvolvedores definam as classes de teste, cada uma contendo um conjunto de métodos de teste que verificam a funcionalidade de uma determinada parte do código.

Os testes JUnit são escritos como métodos Java que chamam o código a ser testado e verificam se o resultado está correto. Os testes são organizados em classes de teste, que contêm vários métodos de teste. O JUnit fornece uma série de ferramentas de teste, como anotações, assertions, runners e test suites, que ajudam os desenvolvedores a criar e executar testes de unidade automatizados de forma eficiente e confiável.

A principal vantagem do JUnit é a sua simplicidade e facilidade de uso. Ele é amplamente utilizado em projetos Java para garantir a qualidade do código e evitar regressões. Além disso, o JUnit pode ser integrado com outras ferramentas de teste, como o **Mockito** (para simular objetos) e o **Testcontainers** (para testes de integração), para criar um conjunto completo de testes automatizados para um aplicativo Java.

Planejamento do Escopo

Proposta Técnica - Testabilidade

Testcontainers

Testcontainers é uma biblioteca de código aberto para **testes de integração que fornece contêineres Docker pré-configurados e escaláveis para testes automatizados**. Ele permite que os desenvolvedores escrevam testes de integração em seus aplicativos que dependem de recursos externos, como bancos de dados, serviços web, filas de mensagens e muito mais, sem a necessidade de configurar e gerenciar esses recursos manualmente.

Com o Testcontainers, os desenvolvedores podem escrever testes mais confiáveis e consistentes, sem se preocupar com o ambiente de teste e a infraestrutura subjacente. Ele oferece uma ampla gama de imagens pré-configuradas do Docker para serviços populares, como MySQL, PostgreSQL, MongoDB, Elasticsearch, Redis, Kafka e muito mais. Além disso, ele permite que os desenvolvedores criem suas próprias imagens personalizadas do Docker para testar seus aplicativos em ambientes específicos.

O Testcontainers é fácil de usar e pode ser integrado a diferentes frameworks de teste, como JUnit, TestNG e Spock. Ele fornece uma API Java simples e intuitiva para criar, gerenciar e destruir contêineres Docker durante os testes de integração. Além disso, ele pode ser executado localmente ou em um ambiente de integração contínua (CI), como o Jenkins, Travis CI ou GitLab CI.

Planejamento do Escopo

Proposta Técnica - Mapeamento Objeto-Relacional

Hibernate

Hibernate é um framework de **mapeamento objeto-relacional** para Java, que facilita o acesso e o gerenciamento de bancos de dados relacionais em aplicativos Java. Ele é uma implementação da especificação Java Persistence API (JPA), que define uma interface padrão para gerenciar objetos em um ambiente de banco de dados relacional.

O Hibernate permite que os desenvolvedores usem a linguagem Java para interagir com o banco de dados, em vez de ter que escrever SQL diretamente. Ele mapeia as classes Java para as tabelas do banco de dados, e vice-versa, permitindo que as operações de banco de dados sejam realizadas usando objetos Java.

Além disso, o Hibernate oferece recursos adicionais, como o gerenciamento automático de transações, cache de objetos, otimização de consultas e suporte a vários bancos de dados. Isso torna o desenvolvimento de aplicativos Java que acessam bancos de dados muito mais fácil e produtivo.

Planejamento do Escopo

Proposta Técnica - Migração

Flyway

Flyway é uma ferramenta de **migração de bancos de dados** que ajuda a gerenciar as mudanças no esquema de um banco de dados de forma programática. Ele é projetado para ser executado automaticamente e em sincronia com o processo de construção de um aplicativo, permitindo que as alterações no banco de dados sejam versionadas e distribuídas com o código-fonte.

O Flyway suporta vários bancos de dados, incluindo PostgreSQL, MySQL, Oracle, SQL Server e muitos outros. Ele funciona adicionando scripts SQL que contêm as alterações no esquema do banco de dados a uma pasta específica, que é então executada automaticamente na ordem especificada pela versão.

O Flyway também inclui recursos para rastrear e gerenciar versões e estados de migração, bem como suporte para rollback de migrações e verificação de integridade do banco de dados. Essas funcionalidades tornam o processo de gerenciamento de migração de banco de dados mais fácil e automatizado, o que é especialmente útil em projetos grandes e complexos.

github.com/thpoiani/lab-quarkus/discussions/1

Vídeo 01 - Orientações Gerais #1

 Pinned thpoiani started this conversation in General



thpoiani

34 minutes ago

Maintainer

edited



Original comment in Portuguese - [Translate to English](#)

Olá e seja bem-vindo! É ótimo ter você aqui depois de assistir ao **Vídeo 01 - Orientações Gerais da Aula - Desenvolvendo um Sistema para Eleição Usando Quarkus Framework** distribuído pela [DIO](#).
Espero que você tenha gostado e que tenha aprendido algo novo. Este fórum é um espaço para discussões e compartilhamento de conhecimentos, ideias e experiências entre desenvolvedores como nós.
Sinta-se à vontade para participar das discussões e fazer perguntas. Juntos, podemos criar uma comunidade de aprendizado e colaboração.



1 comment

Oldest

Newest

Top



thpoiani

7 minutes ago

Maintainer

Author

edited



Original comment in Portuguese - [Translate to English](#)

O que você acha de comentar um pouco sobre o que achou da primeira aula? Aproveite e deixe seu LinkedIn aqui.
Considere me adicionar: <https://www.linkedin.com/in/thpoiani/>



0 replies

Vídeo 02 - Configuração

Preparação do ambiente

SDKMAN!: Java 17 + GraalVM + Native Image

`quarkus create app`

Docker Compose: contêineres

Build e Versionamento

Deployment

SDKMAN!

Java 17

```
sdk list java
```

```
sdk install java 17.0.6-tem
```

```
sdk use java 17.0.6-tem
```

```
› java --version
```

```
openjdk 17.0.6 2023-01-17
```

```
OpenJDK Runtime Environment Temurin-17.0.6+10 (build 17.0.6+10)
```

```
OpenJDK 64-Bit Server VM Temurin-17.0.6+10 (build 17.0.6+10, mixed mode, sharing)
```

```
› env | grep JAVA_HOME
```

```
JAVA_HOME=/Users/thpoiani/.sdkman/candidates/java/17.0.6-tem
```

SDKMAN!

GraalVM 22

```
sdk list java
```

```
sdk install java 22.3.r17-grl
```

```
sdk use java 22.3.r17-grl
```

```
› java --version
```

```
openjdk 17.0.5 2022-10-18
```

```
OpenJDK Runtime Environment GraalVM CE 22.3.0 (build 17.0.5+8-jvmci-22.3-b08)
```

```
OpenJDK 64-Bit Server VM GraalVM CE 22.3.0 (build 17.0.5+8-jvmci-22.3-b08, mixed mode, sharing)
```

```
› env | grep JAVA_HOME
```

```
JAVA_HOME=/Users/thpoiani/.sdkman/candidates/java/22.3.r17-grl
```

```
› env | grep GRAALVM_HOME
```

```
GRAALVM_HOME=/Users/thpoiani/.sdkman/candidates/java/22.3.r17-grl
```

SDKMAN!

Native Image

```
sdk use java 22.3.r17-grl  
gu install native-image
```

SDKMAN!

Quarkus

```
sdk list quarkus
```

```
sdk install quarkus 2.16.4.Final
```

```
› which quarkus
```

```
/Users/thpoiani/.sdkman/candidates/quarkus/current/bin/quarkus
```

```
› quarkus --version
```

```
2.16.4.Final
```

Docker Compose

<https://github.com/thpoiani/lab-quarkus/blob/main/docker-compose.yml>

<https://github.com/thpoiani/lab-quarkus/blob/main/common.yml>

```
docker compose up -d reverse-proxy
```

```
docker compose up -d jaeger
```

```
docker compose up -d mongodb opensearch
```

```
docker compose up -d graylog
```

```
curl -H "Content-Type: application/json" \  
      -H "Authorization: Basic YWRtaW46YWRtaW4=" \  
      -H "X-Requested-By: curl" \  
      -X POST -v -d '{"title":"udp
```

```
input", "configuration": {"recv_buffer_size": 262144, "bind_address": "0.0.0.0", "port": 12201, "de  
compress_size_limit": 8388608}, "type": "org.graylog2.inputs.gelf.udp.GELFUDPInput", "global": t  
rue}' http://logging.private.dio.localhost/api/system/inputs
```

```
docker compose up -d caching database
```

Quarkus

Setup

```
quarkus create app me.dio:election-management \  
  --extension='resteasy-reactive,logging-gelf,opentelemetry,smallrye-context-propagation,smallrye-health' \  
  --no-code
```

application.properties

```
quarkus.application.name=election-management  
quarkus.shutdown.timeout=5S
```

```
%prod.quarkus.log.handler.gelf.enabled=true  
%prod.quarkus.log.handler.gelf.additional-field."app".value=${quarkus.application.name}  
%prod.quarkus.log.handler.gelf.include-full-mdc=true  
%prod.quarkus.log.console.format=%d{HH:mm:ss} %-5p traceId=%X{traceId}, parentId=%X{parentId},  
spanId=%X{spanId}, sampled=%X{sampled} [%c{2.}] (%t) %s%n
```

```
%prod.quarkus.opentelemetry.enabled=true  
%dev.quarkus.opentelemetry.enabled=false
```

Quarkus

JVM Build

```
./mvnw package
```

```
docker build -f src/main/docker/Dockerfile.jvm -t dio/election-management .
```

cicd-build.sh

Quarkus

Native Image Build

```
./mvnw package -Pnative
```

```
docker build -f src/main/docker/Dockerfile.native -t dio/election-management .
```

Quarkus

Progressive Deployment

Blue Green Deployment

cicd-blue-green-deployment.sh

Blue-green deployment é uma **técnica de implantação de software** que envolve a criação de dois ambientes de produção idênticos, um ambiente "azul" (blue) e um ambiente "verde" (green). Enquanto uma versão do aplicativo é executada no ambiente azul, a nova versão é implantada e testada no ambiente verde. Quando a nova versão é considerada estável, o tráfego do usuário é direcionado do ambiente azul para o ambiente verde.

Dessa forma, o ambiente azul permanece disponível e em execução para que possa ser facilmente restaurado em caso de problemas. Esse método é usado para reduzir o tempo de inatividade e minimizar os riscos de falhas durante a implantação de software, permitindo que a nova versão seja implantada e testada antes que o tráfego do usuário seja redirecionado para ela.

github.com/thpoiani/lab-quarkus/discussions/2

Vídeo 02 - Configuração #2

thpoiani started this conversation in General



thpoiani 24 minutes ago

Maintainer

...

Original comment in Portuguese - [Translate to English](#)

Olá novamente! Nesta aula trabalhamos com contêineres e preparamos o ambiente para deployment. Desenvolvemos um Pipeline de *Continuous Integration and Continuous Deployment* rústico, mas que será útil para os conceitos da Aula.

Agora tenho uma pergunta para você: Quais são possíveis melhorias para esse processo de CI/CD?

Pense fora da caixa e sugira qualquer coisa.

► Exemplos (abra depois de comentar para prevenir viés):

🌟 Aproveitando o assunto, sugiro fortemente que acesse o documento de [Deployment Pipeline Reference Architecture](#) da Amazon



1



1 comment

Oldest

Newest

Top



thpoiani 1 minute ago

Maintainer

Author

...

Original comment in Portuguese - [Translate to English](#)

Você conhece [maven-release-plugin](#)? Esse plugin pode simplificar o processo de build e release para projetos maven.



1



0 replies

github.com/thpoiani/lab-quarkus/discussions/3

Vídeo 02 - Configuração - Pergunta adicional #3

Unanswered thpoiani asked this question in Q&A



thpoiani 8 minutes ago

Maintainer

...

Original comment in Portuguese - [Translate to English](#)

Seguindo o padrão do projeto, quais alterações são necessárias em `cicd-build.sh` e `docker-compose.yml` para que seja executado builds para JVM ou Native Image?



Vídeo 03 - Configuração

Aplicações e Discussão Arquitetural

Padrões de Arquitetura de Software

IntelliJ

quarkus dev

Quarkus Dev Services

Padrões de Arquitetura de Software

Arquitetura Hexagonal, Arquitetura Limpa, Arquitetura Cebola

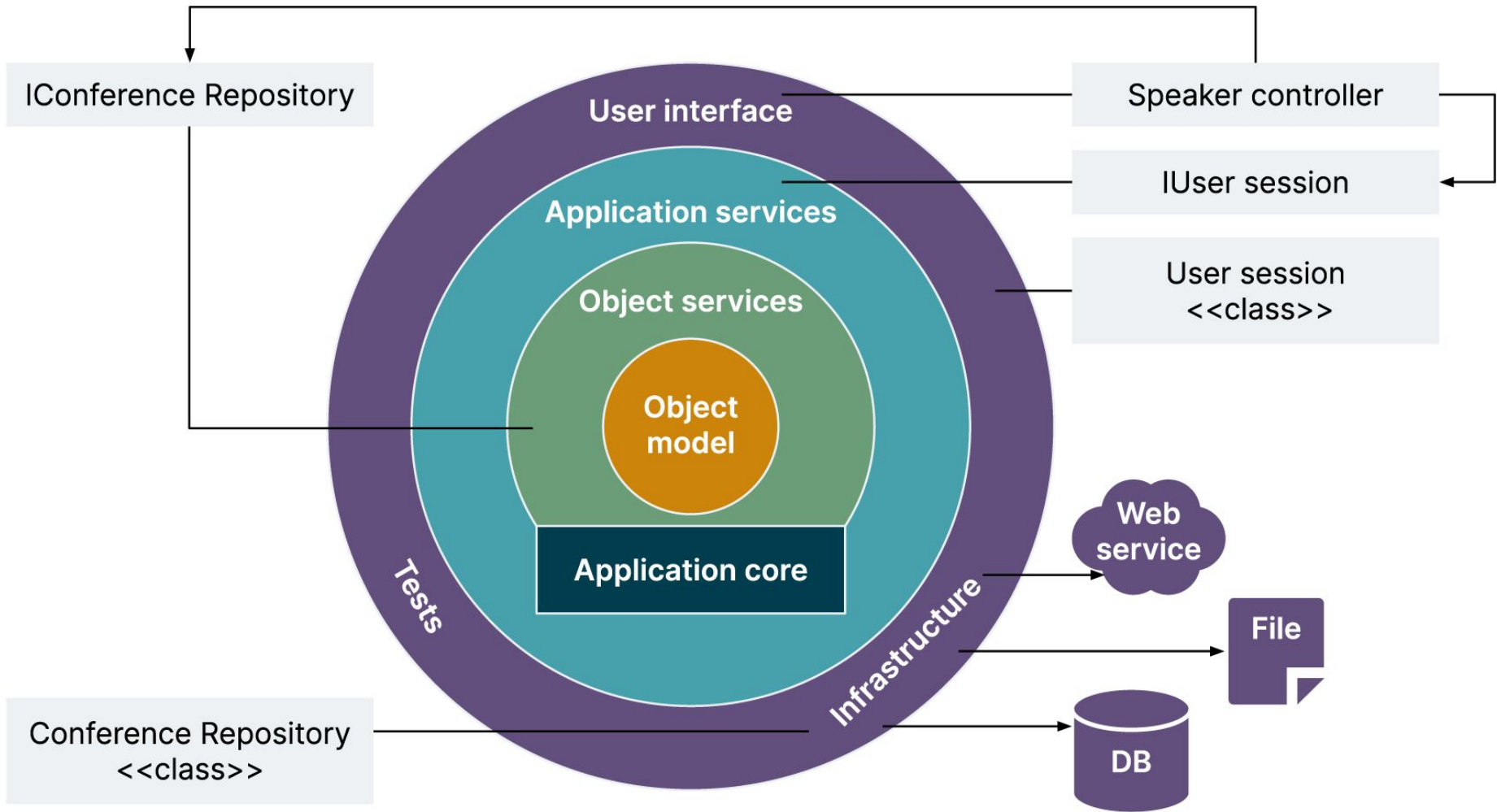
A **arquitetura hexagonal**, a **arquitetura limpa** e a **arquitetura cebola** são todos modelos de arquitetura de software que buscam separar as responsabilidades e promover a modularidade e a manutenibilidade do código.

A **arquitetura hexagonal** é um modelo que coloca o núcleo da aplicação no centro, cercado por portas (interfaces) que fornecem entradas e saídas para a aplicação, e adaptadores que conectam essas portas à infraestrutura externa, como bancos de dados e APIs de terceiros. Essa arquitetura promove a separação de preocupações e permite que a lógica de negócios seja testada independentemente da infraestrutura externa.

A **arquitetura limpa** é um modelo que coloca a lógica de negócios no centro, cercada por camadas que fornecem abstrações para a infraestrutura externa. A arquitetura limpa tem como objetivo isolar a lógica de negócios da infraestrutura externa e promover a testabilidade, manutenibilidade e escalabilidade do código.

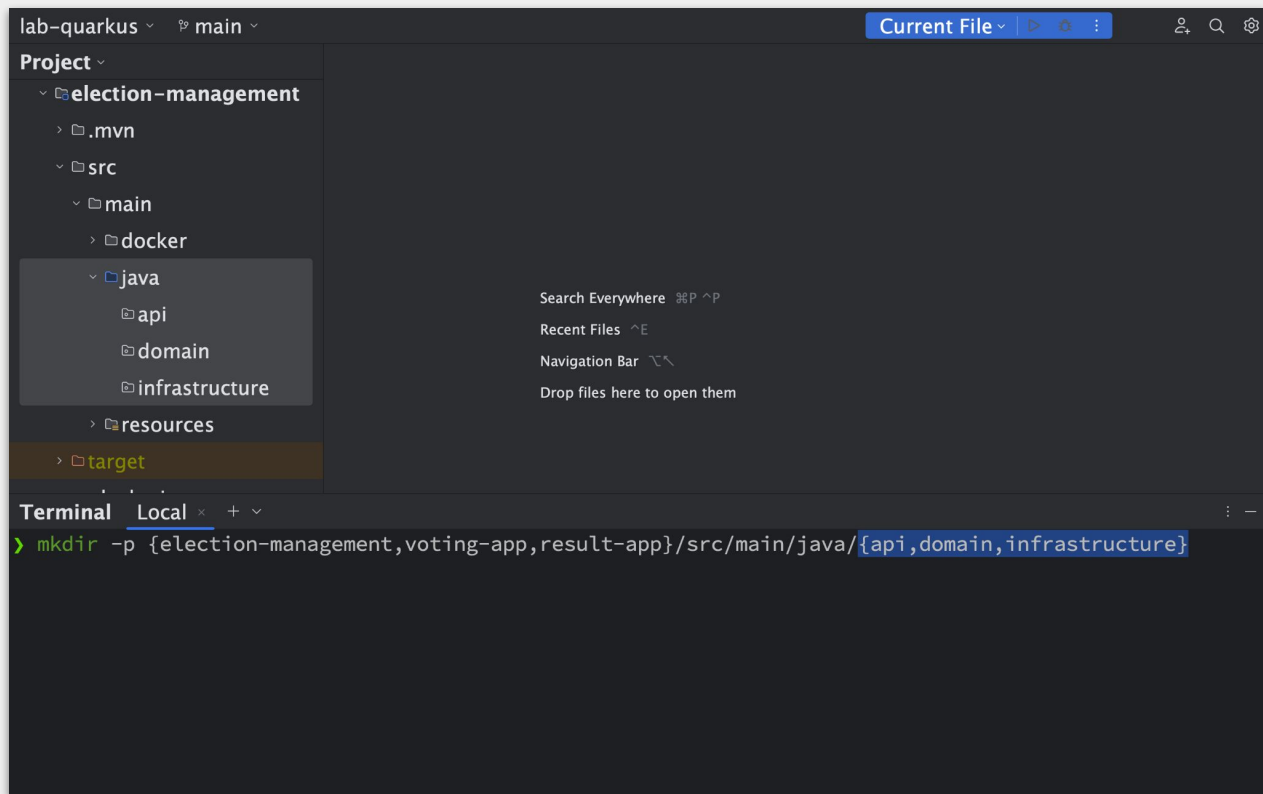
A **arquitetura cebola** é um modelo que coloca a lógica de negócios no centro, cercada por camadas que fornecem abstrações para as camadas externas. A arquitetura cebola tem como objetivo garantir que as camadas mais internas sejam independentes das camadas externas, permitindo que a lógica de negócios seja testada de forma isolada. A arquitetura cebola é semelhante à arquitetura limpa, mas é mais focada na independência das camadas internas.

<https://www.thoughtworks.com/insights/blog/architecture/demystify-software-architecture-patterns>



Padrões de Arquitetura de Software

Estrutura do Projeto



IntelliJ quarkus dev

The screenshot displays the IntelliJ IDEA IDE interface. On the left, the Project Explorer shows a project named 'lab-quarkus' with a directory structure including 'idea', 'election-management', 'mvn', 'src', 'main', 'docker', 'java', 'api', 'domain', 'infrastructure', 'resources', 'target', and '.dockerignore'. The main editor window shows the 'Current File' with the following configuration:

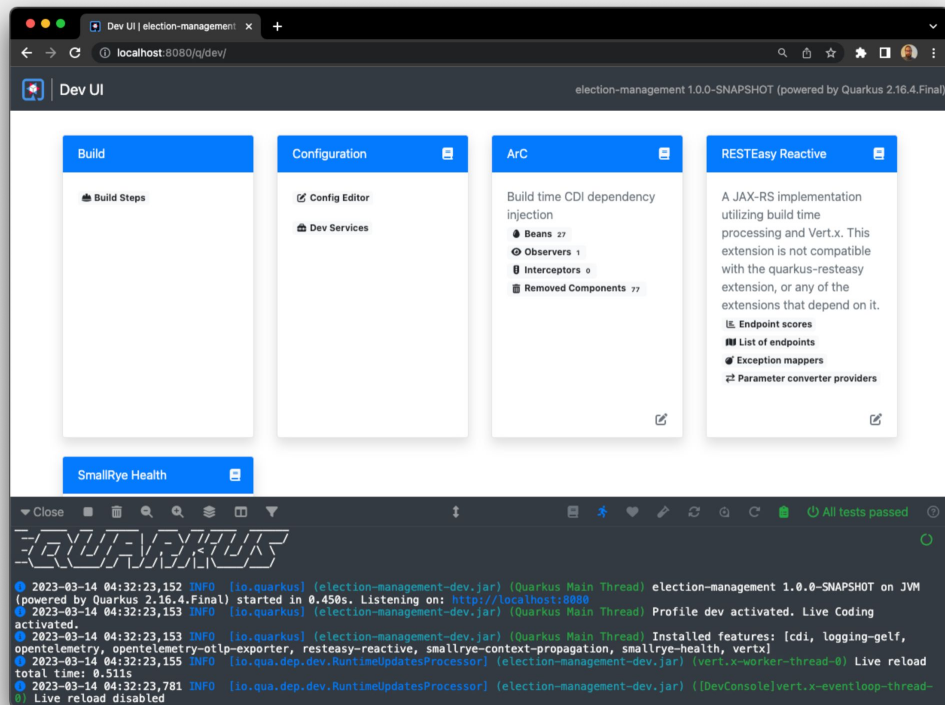
```
1 quarkus.application.name=election-management
2
3 # LOGGING
4 %prod.quarkus.log.handler.gelf.enabled=true
5 %prod.quarkus.log.handler.gelf.include-full-mdc=true
6 %prod.quarkus.log.console.format=%d{HH:mm:ss} %-5p traceId=%X{traceId}, parentId=%X{parentId}, spanId=%X{s
7
8 # OPENTELEMETRY
9 %prod.quarkus.opentelemetry.enabled=true
10 %dev.quarkus.opentelemetry.enabled=false
```

At the bottom, the Terminal window shows the following logs:

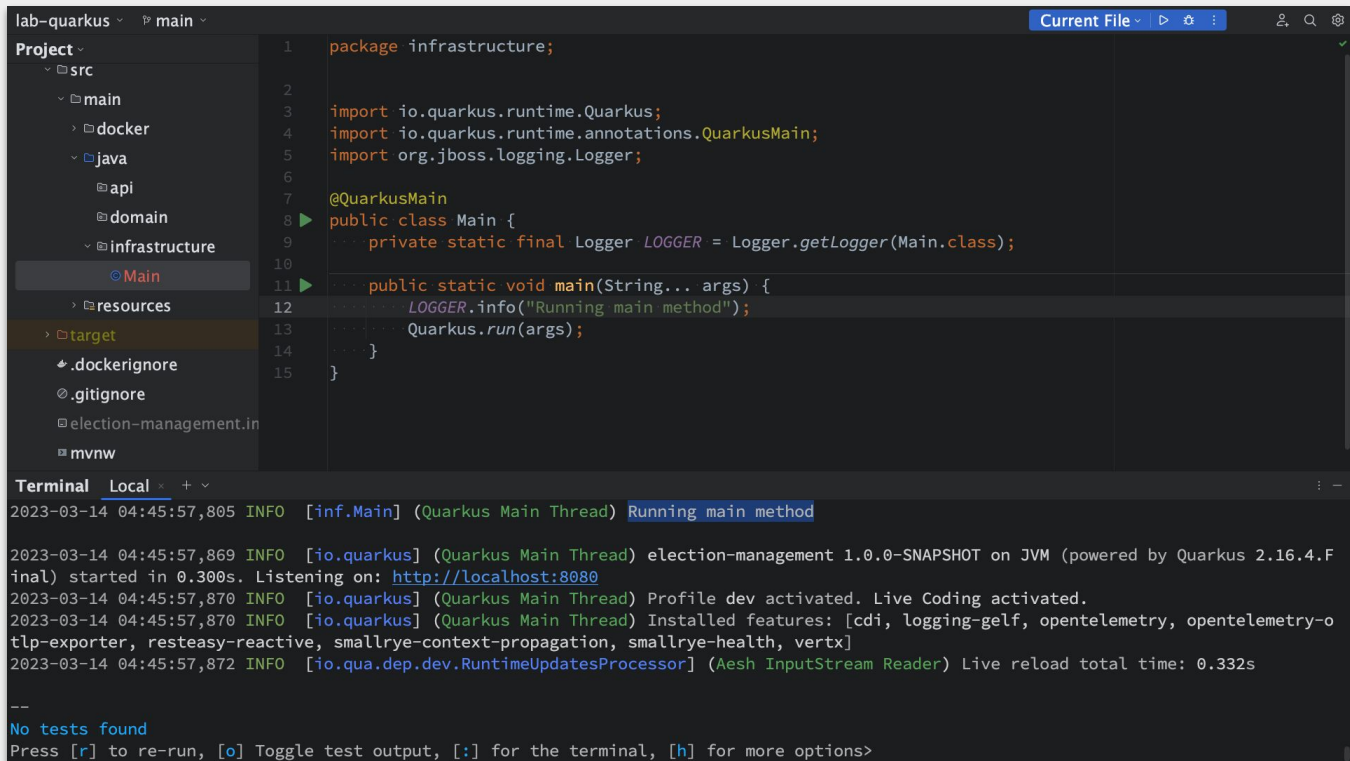
```
2023-03-14 04:21:37,015 INFO [io.quarkus] (Quarkus Main Thread) election-management 1.0.0-SNAPSHOT on JVM (powered by Quarkus 2.16.4.F
inal) started in 0.310s. Listening on: http://localhost:8080
2023-03-14 04:21:37,016 INFO [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
2023-03-14 04:21:37,017 INFO [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, logging-gelf, opentelemetry, opentelemetry-o
tlp-exporter, resteasy-reactive, smallrye-context-propagation, smallrye-health, vertx]
2023-03-14 04:21:37,018 INFO [io.qua.dep.dev.RuntimeUpdatesProcessor] (Aesh InputStream Reader) Live reload total time: 0.348s
2023-03-14 04:24:53,257 INFO [io.qua.dep.dev.RuntimeUpdatesProcessor] (Aesh InputStream Reader) Restarting quarkus due to changes in M
ain.class.
2023-03-14 04:24:53,264 INFO [io.quarkus] (Quarkus Main Thread) election-management stopped in 0.006s
```

Quarkus Dev Services

<http://localhost:8080/q/dev/>



Quarkus Dev Services LifeCycle



The screenshot shows an IDE with a project named 'lab-quarkus'. The left sidebar displays the project structure, including 'src/main/java/infrastructure/Main'. The main editor shows the source code of the 'Main' class, which is a Quarkus main class. The bottom terminal window displays the output of the application, showing the Quarkus Dev Services lifecycle logs.

```
lab-quarkus ▾ main ▾
Project ▾
  ▾ src
    ▾ main
      ▾ docker
      ▾ java
        ▾ api
        ▾ domain
        ▾ infrastructure
          ● Main
      ▾ resources
      ▾ target
      .dockerignore
      .gitignore
      election-management.in
      mvnw
  
```

```
1 package infrastructure;
2
3 import io.quarkus.runtime.Quarkus;
4 import io.quarkus.runtime.annotations.QuarkusMain;
5 import org.jboss.logging.Logger;
6
7 @QuarkusMain
8 public class Main {
9     private static final Logger LOGGER = Logger.getLogger(Main.class);
10
11     public static void main(String... args) {
12         LOGGER.info("Running main method");
13         Quarkus.run(args);
14     }
15 }
```

```
Terminal Local x + ▾
2023-03-14 04:45:57,805 INFO [inf.Main] (Quarkus Main Thread) Running main method

2023-03-14 04:45:57,869 INFO [io.quarkus] (Quarkus Main Thread) election-management 1.0.0-SNAPSHOT on JVM (powered by Quarkus 2.16.4.Final) started in 0.300s. Listening on: http://localhost:8080

2023-03-14 04:45:57,870 INFO [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.

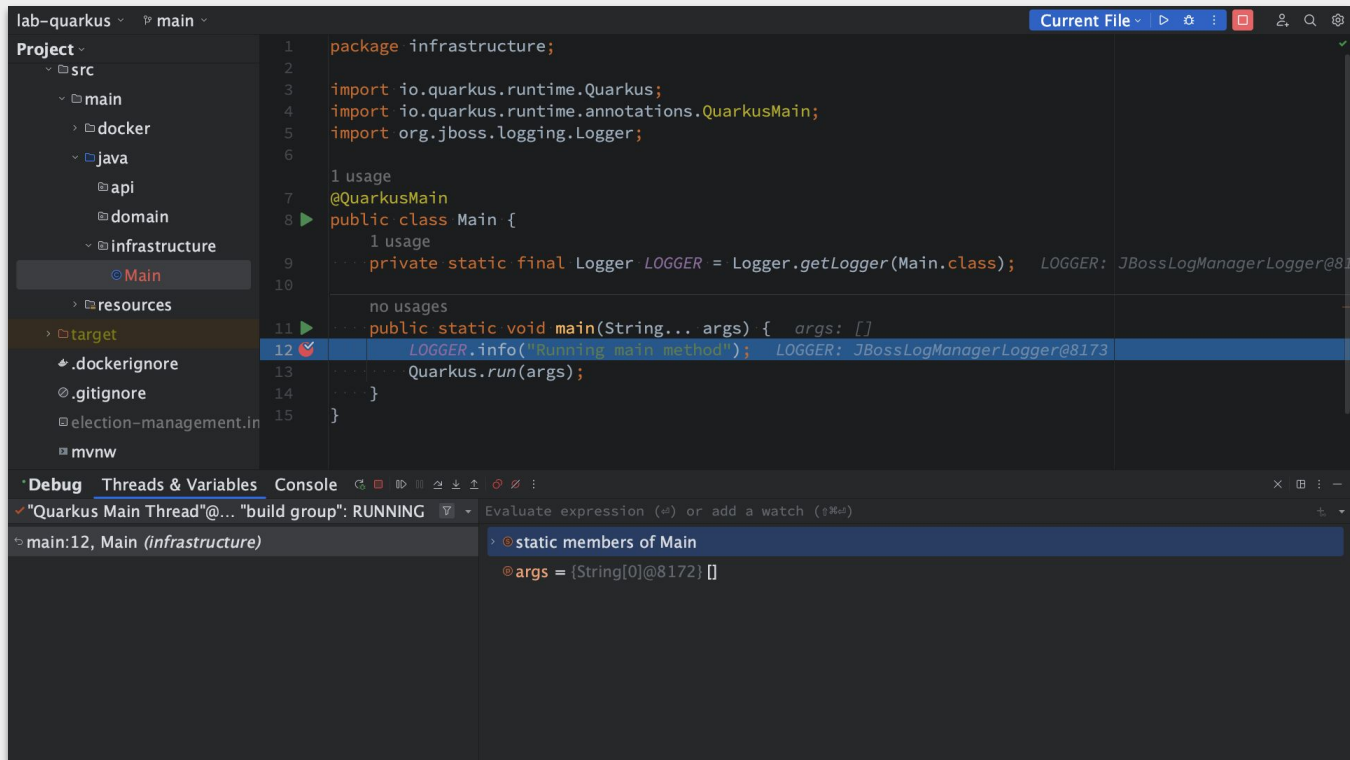
2023-03-14 04:45:57,870 INFO [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, logging-gelf, opentelemetry, opentelemetry-tlp-exporter, resteasy-reactive, smallrye-context-propagation, smallrye-health, vertx]

2023-03-14 04:45:57,872 INFO [io.qua.dep.dev.RuntimeUpdatesProcessor] (Aesh InputStream Reader) Live reload total time: 0.332s

--
No tests found
Press [r] to re-run, [o] Toggle test output, [:] for the terminal, [h] for more options>
```

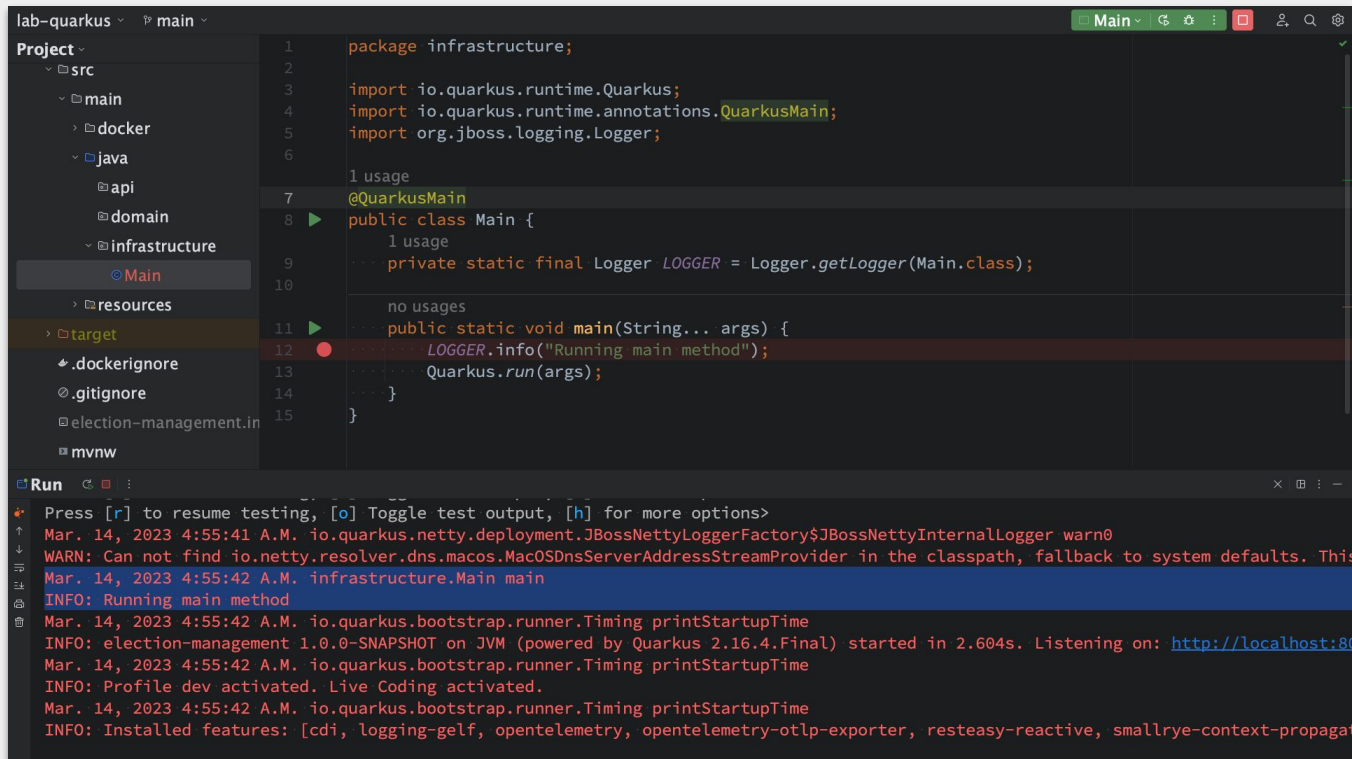
Quarkus Dev Services

Debug - Run > Attach to Process



Quarkus Dev Services

Run Main.main()



The screenshot shows an IDE with a project named 'lab-quarkus' and a file named 'Main.java' in the 'src/main/java/infrastructure' directory. The code defines a 'Main' class with a 'main' method that logs 'Running main method' and runs Quarkus. The 'Run' button is highlighted, and the output console shows the execution logs.

```
1 package infrastructure;
2
3 import io.quarkus.runtime.Quarkus;
4 import io.quarkus.runtime.annotations.QuarkusMain;
5 import org.jboss.logging.Logger;
6
7 @QuarkusMain
8 public class Main {
9     private static final Logger LOGGER = Logger.getLogger(Main.class);
10
11     public static void main(String... args) {
12         LOGGER.info("Running main method");
13         Quarkus.run(args);
14     }
15 }
```

Run

Press [r] to resume testing, [o] Toggle test output, [h] for more options>

Mar. 14, 2023 4:55:41 A.M. io.quarkus.netty.deployment.JBossNettyLoggerFactory\$JBossNettyInternalLogger warn0
WARN: Can not find io.netty.resolver.dns.macos.MacOSDnsServerAddressStreamProvider in the classpath, fallback to system defaults. This

Mar. 14, 2023 4:55:42 A.M. infrastructure.Main main
INFO: Running main method

Mar. 14, 2023 4:55:42 A.M. io.quarkus.bootstrap.runner.Timing printStartupTime
INFO: election-management 1.0.0-SNAPSHOT on JVM (powered by Quarkus 2.16.4.Final) started in 2.604s. Listening on: <http://localhost:8080>

Mar. 14, 2023 4:55:42 A.M. io.quarkus.bootstrap.runner.Timing printStartupTime
INFO: Profile dev activated. Live Coding activated.

Mar. 14, 2023 4:55:42 A.M. io.quarkus.bootstrap.runner.Timing printStartupTime
INFO: Installed features: [cdi, logging-gelf, opentelemetry, opentelemetry-otlp-exporter, resteasy-reactive, smallrye-context-propagat

Referências

<https://quarkus.io/guides/maven-tooling#dev-mode>

<https://quarkus.io/guides/dev-services>

<https://quarkus.io/guides/lifecycle#the-main-method>

<https://quarkus.io/guides/config#configuring-quarkus>

github.com/thpoiani/lab-quarkus/discussions/4

Vídeo 03 - Configuração #4

thpoiani started this conversation in General



thpoiani now Maintainer

...

Obrigado por acompanhar mais um vídeo!

Dessa vez comentamos sobre padrões de arquitetura e iniciamos a estrutura de um projeto com Quarkus.

Posso citar algumas vantagens ao utilizar um padrões de arquitetura: modularidade, testabilidade, redução de acoplamento, ...

Você havia trabalhado com algum desses padrões ou com algum outro estilo (*Model-View-Controller*, *Model-View-Presenter*, *Model-View-ViewModel*, *Event Driven Architecture*, ...)?

Estou curioso... me conte sobre a sua primeira impressão com Quarkus durante esse vídeo 😊



Vídeo 04 - Desenvolvimento

Gerenciamento de Candidatos - Serviço

Domain Model

Java Record

Testing

Test Driven Development

Mocking

Service Layer

Dependency Injection

Repository

Query Object

Builder Pattern

Referências

Domain Model

<https://martinfowler.com/eaCatalog/domainModel.html>

<https://docs.oracle.com/en/java/javase/17/language/records.html>

Testing

<https://quarkus.io/guides/getting-started-testing>

<https://quarkus.io/guides/continuous-testing>

<https://martinfowler.com/bliki/TestDrivenDevelopment.html>

<https://www.thoughtworks.com/insights/blog/test-driven-development-best-thing-has-happened-software-design>

<https://www.thoughtworks.com/insights/topic/testing>

Service Layer

<https://martinfowler.com/eaCatalog/serviceLayer.html>

<https://www.martinfowler.com/articles/injection.html>

<https://quarkus.io/guides/cdi-reference>

Repository

<https://martinfowler.com/eaCatalog/repository.html>

<https://martinfowler.com/eaCatalog/queryObject.html>

<https://martinfowler.com/dslCatalog/constructionBuilder.html>

github.com/thpoiani/lab-quarkus/discussions/5

Vídeo 04 - Desenvolvimento #5

thpoiani started this conversation in General



thpoiani 1 hour ago

Maintainer

edited ...

Original comment in Portuguese - [Translate to English](#)

Além de desenvolver, eu também gosto muito de me envolver com Arquitetura e Engenharia de Software 😊
E finalmente colocamos a mão na massa!! Muitos conceitos foram apresentados nesse vídeo: [Domain Model](#), [Java Records](#), [TDD](#), [Service Layer](#), [Dependency Injection](#), [Repository](#), [Query Object](#) e [Builder Pattern](#).

Montamos um serviço simples que atende nossas necessidades iniciais e segue um fundamento essencial: **baixo acoplamento e alta coesão**. E tudo isso foi desenvolvimento com **continuous testing**.

Compartilhe comigo:

- quais desses conceitos são novidades para você?
- você tem o costume de desenvolver testes unitário automatizados?
- quais são as vantagens de tem um processo com teste contínuo?

↑ 1 😊

Vídeo 05 - Desenvolvimento

Gerenciamento de Candidatos - Repositório

Migration

Flyway

Testcontainers

Data Mapper

Hibernate ORM

Referências

Migration

<https://martinfowler.com/articles/evodb.html>

<https://quarkus.io/guides/flyway>

<https://quarkus.io/guides/databases-dev-services>

Data Mapper

<https://martinfowler.com/eaCatalog/dataMapper.html>

<https://quarkus.io/guides/hibernate-orm>

<https://quarkus.io/guides/datasource>

github.com/thpoiani/lab-quarkus/discussions/6

Vídeo 05 - Desenvolvimento #6

thpoiani started this conversation in General



thpoiani now

Maintainer

...

Esse vídeo foi um pouco complicado para mim 😊

Meu planejamento era usar [Hibernate Reactive with Panache](#) para aproveitar a linguagem reativa do Quarkus, contudo ainda existem alguns problemas de compatibilidade para usar Flyway com conexão reativa. Existe um *workaround* para essa issue [quarkusio/quarkus#10716](#), mas isso trouxe outros problemas com o Testcontainers. Decidi voltar ao [Hibernate ORM](#) padrão.

Quem nunca sofreu achando que seria uma simples implementação, mas no final levou horas para resolver um problema? Compartilhe uma história parecida 😊



1



github.com/thpoiani/lab-quarkus/discussions/7

Vídeo 05 - Pergunta adicional #7

Unanswered thpoiani asked this question in Q&A



thpoiani now

Maintainer

...

Original comment in Portuguese - [Translate to English](#)

Com a configuração atual, `secrets`, como usuário e senha do banco de dados, são passados diretamente para o `docker-compose.yml`.

Gostaria de receber sugestões de outras abordagens para aumentar a segurança desse sistema.

► Minha sugestão - abra depois de responder



Vídeo 06 - Desenvolvimento

Gerenciamento de Candidatos - **API**

JSON Rest Services

Data Transfer Object

Integration Test

Referências

API

<https://martinfowler.com/eaCatalog/remoteFacade.html>

<https://quarkus.io/guides/rest-json>

<https://quarkus.io/guides/resteasy-reactive>

<https://quarkus.io/guides/openapi-swaggerui>

Data Transfer Object

<https://martinfowler.com/eaCatalog/dataTransferObject.html>

Testing

<https://quarkus.io/guides/getting-started-testing#restassured>

<https://quarkus.io/guides/getting-started-testing#quarkus-integration-test>

<https://martinfowler.com/bliki/IntegrationTest.html>

github.com/thpoiani/lab-quarkus/discussions/8

Vídeo 06 - Desenvolvimento #8

thpoiani started this conversation in General



thpoiani now Maintainer

...

Original comment in Portuguese - [Translate to English](#)

Agora temos um serviço exposto e conseguimos começar a consultar a API.
Além disso fizemos alguns testes de integração, verificamos tracing e logging.

Você havia trabalhado anteriormente com testes de integração? É uma boa oportunidade para validação de contratos e verificações end-to-end.

OpenTelemetry nos dá uma visão interessante do comportamento da aplicação. **Observabilidade** é um tópico sempre em alta. Contudo não estamos adicionado um componente para Métricas. Algum sugestão de ferramenta?



Vídeo 07 - Desenvolvimento

Gerenciamento de Eleição - Evento

Gerenciamento de Eleições

Redis

Event Driven

Redis **Pub**/Sub

Referências

Redis

<https://quarkus.io/guides/redis>

<https://quarkus.io/guides/redis-reference>

<https://redis.io/commands/zadd>

Event Driven

<https://martinfowler.com/articles/201701-event-driven.html>

<https://redis.io/docs/manual/pubsub>

github.com/thpoiani/lab-quarkus/discussions/9

Vídeo 07 - Desenvolvimento #9

thpoiani started this conversation in General



thpoiani 8 minutes ago

Maintainer

edited ...

Original comment in Portuguese - [Translate to English](#)

Olá! Nesse vídeo montamos um sistema simples de mensageria com Redis PUBSUB.

Quais outros serviços de mensageria você conhece? RabbitMQ, SNS/SQS, Kafka, ... ?

E quais são os principais motivos de usar tal serviço em comparação com REST? *(hehe essa não é a pergunta ideal, que tal: sincronicidade vs assincronicidade?)*

Tenho uma referência muito interessante sobre isso e recomendo fortemente que assista esse vídeo:

<https://developertoarchitect.com/lessons/lesson137.html> ★



1



github.com/thpoiani/lab-quarkus/discussions/10

Vídeo 07 - Pergunta adicional #10

Unanswered thpoiani asked this question in Q&A



thpoiani now

Maintainer

edited ...

Original comment in Portuguese - [Translate to English](#)

Na nova migration adicionada nesse vídeo, não usei chaves estrangeiras.

Claro que tudo depende muito da natureza dos dados envolvidos, mas você consegue pensar em alguns cenários onde seguir **normalização de bancos de dados**, que realmente é um conceito importante, muito fortemente pode ser uma desvantagem?



1



Vídeo 08 - Desenvolvimento

Aplicação para Votação

Lifecycle onStartUp

Redis

Redis Pub/Sub

Memoization/Caching

Reactive

Mutiny

Referências

Lifecycle

<https://quarkus.io/guides/lifecycle>

Redis

<https://redis.io/commands/zrange>

<https://redis.io/commands/zincrby>

Memoization/Caching

<https://quarkus.io/guides/cache>

Reactive

<https://quarkus.io/guides/getting-started-reactive>

<https://quarkus.io/guides/quarkus-reactive-architecture>

<https://quarkus.io/guides/mutiny-primer>

<https://smallrye.io/smallrye-mutiny>

github.com/thpoiani/lab-quarkus/discussions/11

Vídeo 08 - Desenvolvimento #11

thpoiani started this conversation in General



thpoiani now

Maintainer

...

Nesse vídeo falamos rapidamente sobre Programação Reativa.
A arquitetura do Quarkus é fortemente inspirada nesse paradigma.

Você conhece outras bibliotecas para programação reativa? [ReactiveX \(RxJava\)](#), [Project Reactor](#), [Vert.X](#), [Mutiny](#), ...

★ Fique atento ao **Project Loom**: <https://blogs.oracle.com/javamagazine/post/java-loom-virtual-threads-platform-threads>

Project Loom é uma iniciativa para melhorar o desempenho e a escalabilidade do processamento concorrente em Java. O objetivo principal do projeto é tornar mais fácil e eficiente escrever código concorrente, especialmente para aplicações com muitas threads, implementando uma nova forma de lidar com as threads chamada de "Virtual Threads".

Possivelmente o Project Loom estará disponível na próxima **versão LTS de Java (21)**, com planejamento de release para **Setembro de 2023**.

● Fique atento, pois essa feature tem potencial para matar programação reativa!

Sugiro assistir essa entrevista com [@briangoetz](#), um dos arquitetos da linguagem Java que está liderando evoluções na linguagem:
<https://www.youtube.com/watch?v=9sl7gK94gLo&t=1156>



1



Vídeo 09 - Desenvolvimento

Gerenciamento de Eleição - Sincronismo

Scheduler

Referências

Scheduler

<https://quarkus.io/guides/scheduler>

LEITURA RECOMENDADA

github.com/thpoiani/lab-quarkus/discussions/12

Vídeo 09 - Desenvolvimento #12

thpoiani started this conversation in General



thpoiani now Maintainer

...

Original comment in Portuguese - [Translate to English](#)

Estamos usando um Scheduler/cron para uma tarefa periódica de sincronismo entre o Redis e MariaDB.

Quais outras soluções você consegue pensar para substituir esse cron?



1



Vídeo 10 - Desenvolvimento

Resultados

Qualifier

Rest Client

Server Sent Events

Referências

Qualifiers

<https://quarkus.io/guides/cdi#you-talked-about-some-qualifiers>

Rest Client

<https://quarkus.io/guides/rest-client>

<https://quarkus.io/guides/rest-client-reactive>

Server Sent Events

https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events

github.com/thpoiani/lab-quarkus/discussions/13

Vídeo 10 - Desenvolvimento #13

thpoiani started this conversation in General



thpoiani now Maintainer

...

Esse é o último vídeo de desenvolvimento.

Dessa vez deixamos um endpoint exposto para exibir os resultados para a abordagem de **Server Sent Events**.

Como você desenvolveria um client (frontend) para consumir esse endpoint e exibir o resultado da eleição?



1



Vídeo 11 - Demonstração

Encerramento e Proposta de Desafios

Demonstração

Load Testing

Client React

Desafios

- **(Developer)** Adicione métricas nas aplicações usando **Grafana** como dashboard
 - <https://grafana.com> <https://quarkus.io/guides/micrometer>
- **(Developer)** Finalize a **implementação do gerenciamento de candidatos** no election-management
 - Remoção, Listagem com Paginação
- **(Developer)** Substitua o gerenciamento de candidatos por **REST Data with Panache**
 - <https://quarkus.io/guides/rest-data-panache>
- **(Developer)** Implemente alterações no código para tornar as aplicações mais reactivas/assíncrona
 - <https://quarkus.io/guides/resteasy-reactive#asyncreactive-support>
- **(DevOps)** Provisione uma **réplica de leitura do banco de dados**, habilitando múltiplas conexões nos repositories
 - <https://mariadb.com/kb/en/setting-up-replication> <https://quarkus.io/guides/hibernate-orm#multiple-persistence-units>
- **(DevOps)** Planeje a migração desse sistema para um ambiente **Kubernetes**
- **(DevSecOps)** Planeje um processo de **Modelagem de Ameaças** considerando técnicas de detecção e prevenção de DDoS
 - https://owasp.org/www-community/Threat_Modeling_Process
- **(Architect)** Documente ADRs para: 1- **evitar múltiplos votos de uma mesma origem**, 2: **detecção de fraude incluindo auditoria**
 - <https://docs.aws.amazon.com/prescriptive-guidance/latest/architectural-decision-records/appendix.html>
- **(Data Scientist)** Modele um workflow para **Análise Preditiva** do resultado da eleição
- **(Product Owner)** Crie um roadmap com novas funcionalidades

Muito obrigado por ter
acompanhado o Lab Quarkus!



[linkedin.com/in/thpoiani](https://www.linkedin.com/in/thpoiani)