Arquitetura Hexagonal

Módulo 3 - Implementação Java

Módulos

1. ~~Teoria, conceitos, valores e metáforas.~~
2. ~~Projeto e Design Java.~~
3. Implementação com Java.

Abertura

Nesse módulo 3 faremos a implementação um estudo de caso **completo e 100% funcional**, usando JDK, IntelliJ, TDD, JUnit, Java 12, Java Module, Spring Framework, JavaFX e banco de dados relacional HSQDB.

Arquitetura Hexagonal

Projeto Java: Transferência Bancária

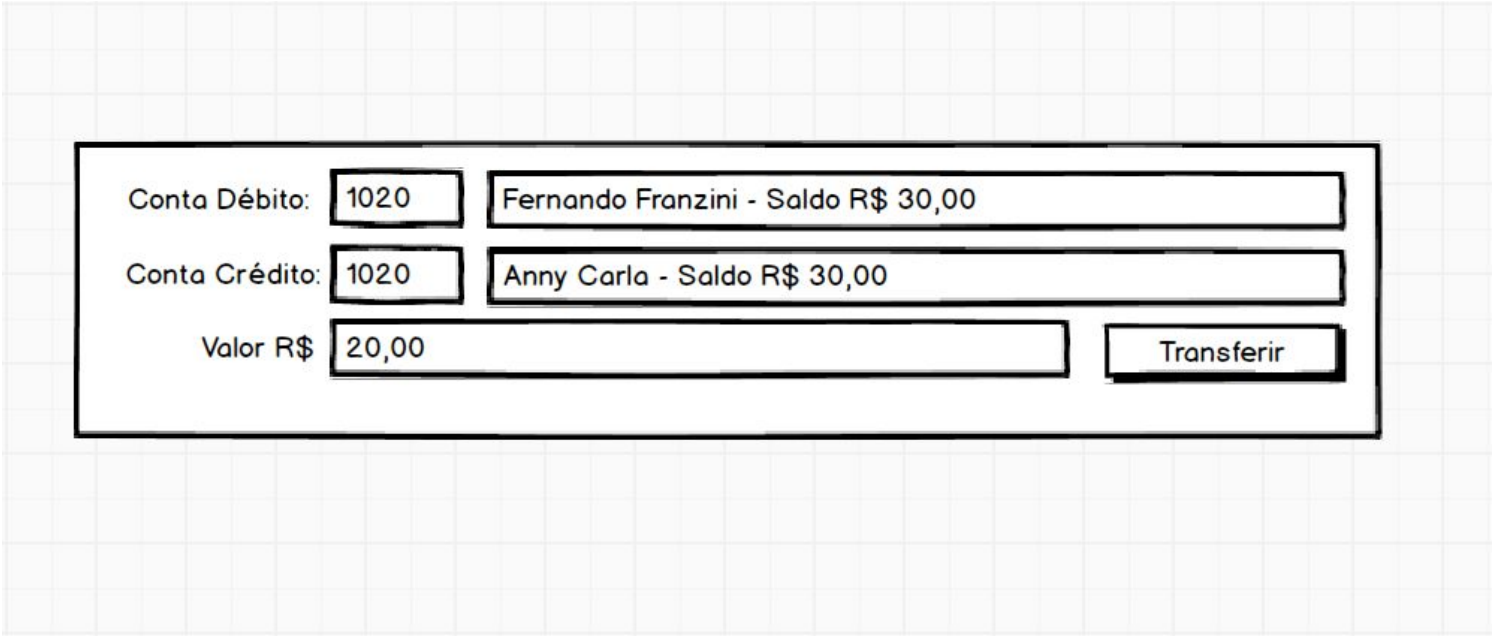
Projeto Java: Transferência Bancária

Vamos desenvolver, testar e executar um projeto hexagonal completo com Java.

Ele será pequeno, mas suficiente para implementar todo o ciclo e as decisões mais fundamentais sobre esse padrão arquitetural.

Projeto Java: Transferência Bancária

Protótipo:



Projeto Java: Transferência Bancária

Base de Dados Relacional:

```
create table conta (
  numero integer primary key,
  saldo decimal (10,2),
  correntista varchar(200)
);
```

Projeto Java: Transferência Bancária

Eu não vou ensinar e nem comentar tópicos:

- Intellij, Tdd, JUnit, Sintaxe Java, OOP, Polimorfismo, DbC, Spring e JavaFX.

Vamos copiar o projeto, pontuando as metáforas e estruturas da arquitetura hexagonal. Se porventura eu usar algo que você não conhece, anote o tópico e continue no projeto. Posteriormente, fora do curso, você corre atrás para aprender.

Projeto Java: Transferência Bancária

TDD:

O projeto foi desenvolvido usando TDD, com vários ciclos de refatoração. Mas, no exercício vamos apenas copiar o fonte desenvolvido.

Projeto Java: Transferência Bancária

Dúvidas e comentários?

Arquitetura Hexagonal

Decisões de Design: Transferência Bancária

Decisões de Design: Transferência Bancária

Como apresentado no tópico "Opções de Design", arquitetura hexagonal vem incompleta para você completar com suas decisões de design.

Seguem as do projeto de transferência entre contas:

Decisões de Design: Transferência Bancária

Projeto:

- 3 projetos separados: sistema.jar (hexagonal), servicos.jar (lado direito) e desktop (lado esquerdo).
- Dependências frameworks via maven.
- Dependências hexagonal via maven system local.
- Organização de pacotes misto de DDD e metáforas hexagonal.
- Organização de negócio via EAA: Modelo de Domínio e Serviço de Domínio.

Decisões de Design: Transferência Bancária

Projeto:

- Organização das Portas Primárias: várias interfaces, cada uma agrupando operações relacionadas. Mas no projeto teremos apenas 1 grupo.
- Transferência de dados: variáveis simples.
- Consumo de objetos de backservices somente nos objetos de portas, deixando a regra de negócio puro POJO sem dependência com o mundo externo.

Decisões de Design: Transferência Bancária

Tecnologias:

- Dependências frameworks java via maven.
- Banco de dados relacional usaremos HSQDB. Para desenvolvimento e homologação vamos usar em modo "embedded database". Para produção vamos usar em modo "local jvm".
- Front-End usaremos desktop via JavaFX.

Decisões de Design: Transferência Bancária

Tecnologias:

- Provedor de persistência Spring JDBC.
- Provedor de transação Spring Transaction.
- Provedor de IoC Spring IoC.
- Validação de negócio manual, nenhum framework.
- Builds IoC via Java Config manual, nenhum framework.

Decisões de Design: Transferência Bancária

Dúvidas e comentários?

Arquitetura Hexagonal

Ambiente de Desenvolvimento: Transferência Bancária

Ambiente Java: Transferência Bancária

JDK:

Vc fica livre para usar qualquer jdk de sua preferência.
Eu vou usar OpenJDK, é grátis para estudo, demonstração e produção.

Ambiente Java: Transferência Bancária

IDE IntelliJ

Vamos desenvolver todo o projeto usando intellij.

Segue curso de instalação e tunning:

- https://www.udemy.com/intellij-ide-para-desenvolvedores-java/

Ambiente: Transferência Bancária

Dúvidas e comentários?

Arquitetura Hexagonal

Implementação: Transferência Bancária

Implementação Java: Transferência Bancária

Parte1: Projeto Sistema (Hexágono)

Vamos implementar as regras de negócio, processos de negócio, portas dirigidas, seus mocks, portas condutoras e testar tudo com TDD.

Build 1: Adaptador Testes -> Sistema <- Adaptadores Mocks

1. Criar o projeto "sistema".
2. Configurar o POM.
3. Iniciar java module.
4. Copiar os fontes na sequência do manual.
5. Finalizar java module.
6. Gerar o jar.

Implementação Java: Transferência Bancária

Parte 2: Projeto Front-End (Lado Esquerdo)

Vamos implementar o adaptador real usando JavaFX.

Build 2: Adaptador Interface Gráfica -> Sistema <- Adaptadores Mocks

1. Criar o projeto "desktop".
2. Configurar o POM.
3. Java Module.
4. Criar a estrutura dos pacotes.
5. Copiar os fontes na sequência do manual.

Implementação Java: Transferência Bancária

Parte 3: Projeto Serviços (Lado Direito)

Vamos implementar o adaptador real de banco de dados usando Spring JDBC e HSQLDB.

1. Criar o projeto "serviços".
2. Configurar o POM.
3. Java module.
4. Copiar os fontes na sequência do manual.
5. Gerar o jar.

Implementação Java: Transferência Bancária

Parte 3: Build Homologação

Build 3: Adaptador Interface Gráfica -> Sistema <- Adaptadores Serviços Homologação

1. Configurar o POM.
2. Configurar java module.
3. Config IoC.
4. Executar em homologação.

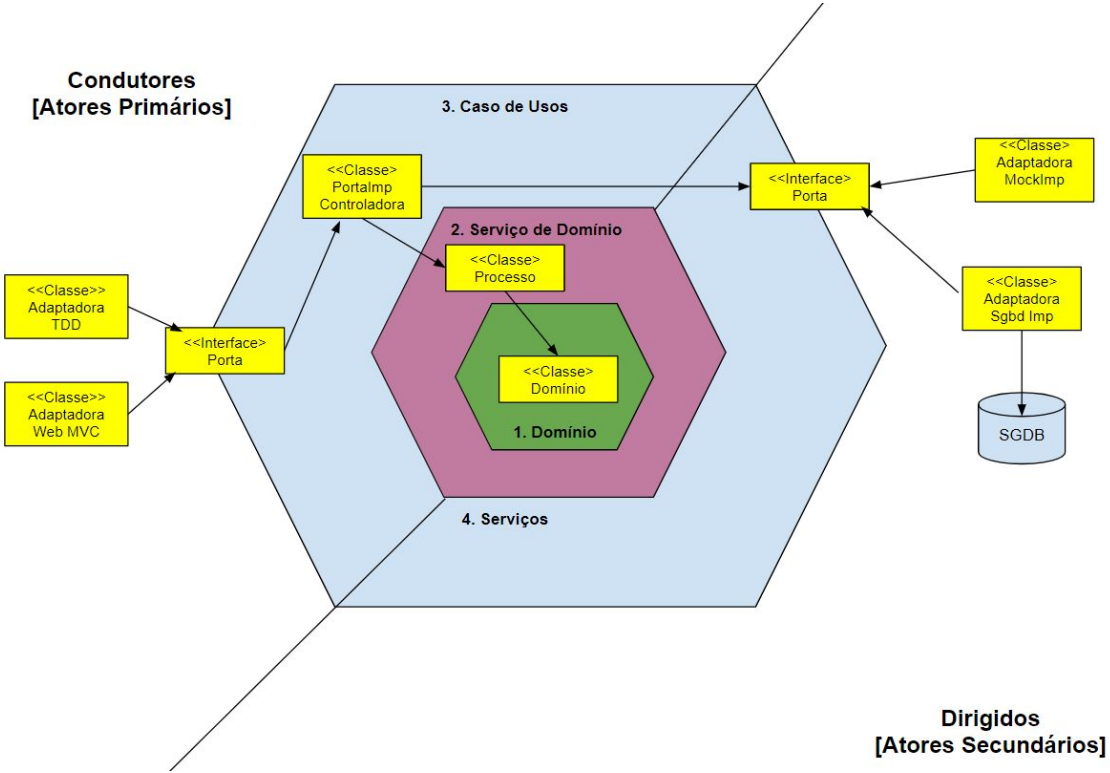
Implementação Java: Transferência Bancária

Parte 4: Build Produção

Build 4: Adaptador Interface Gráfica -> Sistema <- Adaptadores Serviços Produção

1. Criar banco de dados de produção.
2. Config IoC.
3. Executar produção.
4. Intercalar nos vários builds.

Implementação Java: Transferência Bancária



Implementação Java: Transferência Bancária

Dúvidas e comentários?

Arquitetura Hexagonal

Implementação Java: Fechamento

Fechamento

1. Teoria, conceitos, valores e metáforas.
2. Projeto e Design Java.
3. Implementação com Java.
4. Build 1 - Adaptador Tdd -> Core <- Mocks.
5. Build 2 - Adaptador Real -> Core <- Mocks.
6. Build 3 - Adaptador Real -> Core <- Adaptador Real em Homologação.
7. Build 4 - Adaptador Real -> Core <- Adaptador Real em Produção.
8. Todos os conceitos foram implementados.

Fechamento