**Exercício – Engenharia de Software**

**Prof: Alex**

**24.10.2024**

**Gabriel Boos Duarte e Gabriel Cestari Sanchez**

**Como juntar regra de negócio, infra e software em uma mesma arquitetura?**

Existem alguns passos que podem ser usados para ajudar a juntar regra de negócio, infraestutura e software, podem ser usado alguns passos que podem ajudar:

**Modelagem e Documentação**:

* **Archimate**: Para modelagem de arquitetura empresarial, ajudando a visualizar a interconexão entre regras de negócio e componentes técnicos.
* **Ardoq**: Para modelagem visual e colaboração em arquiteturas dinâmicas.

**Estrutura de software:**

1. **Modelo de Domínio**: Use DDD para criar um modelo de domínio que reflita as regras de negócio e para facilitar a integração entre diferentes partes do sistema e manutenção.
2. **Arquitetura Hexagonal**: Garanta que a lógica de negócio esteja isolada das interações externas, usando adaptadores para integrar diferentes fontes de dados e interfaces.

**Archimate**

**O que é**: Archimate é uma linguagem de modelagem gráfica projetada para descrever a arquitetura empresarial, oferecendo uma notação padronizada para representar componentes de negócios, aplicações e infraestrutura de TI.

Exemplo:Diagrama

Descrição gerada automaticamente

**Como funciona**:

1. **Representação Visual**: Utiliza uma série de símbolos e diagramas para modelar diferentes camadas da arquitetura, como:
   * **Camada de Negócio**: Representa processos, serviços e funções de negócio.
   * **Camada de Aplicação**: Mostra como os aplicativos apoiam os processos de negócio.
   * **Camada de Tecnologia**: Refere-se à infraestrutura, como servidores e redes.
2. **Conexões e Relações**: Os elementos podem ser conectados por relações que ilustram como as partes da arquitetura interagem, como um processo de negócio que depende de um serviço de aplicação conectado a um componente de infraestrutura.
3. **Facilita a Comunicação**: A notação padrão ajuda a comunicar a arquitetura de forma clara e concisa para diferentes stakeholders, promovendo compreensão mútua sobre como as regras de negócio se integram com a infraestrutura e o software.
4. **Análise de Impacto**: Permite visualizar o impacto de mudanças nas regras de negócio sobre a infraestrutura e vice-versa, auxiliando na tomada de decisões.

**Ardoq**

**O que é**: Ardoq é uma ferramenta de modelagem e documentação que possibilita criar visualizações dinâmicas de arquiteturas empresariais e sistemas complexos, com foco na colaboração e flexibilidade.

**Como funciona**:

1. **Modelagem Interativa**: Permite a criação de modelos de arquitetura de maneira colaborativa, facilitando a construção de diagramas que representam a relação entre regras de negócio, software e infraestrutura.
2. **Visualizações Dinâmicas**: Oferece a criação de visualizações personalizadas que refletem diferentes aspectos da arquitetura, úteis para explorar a desagregação das regras de negócio em componentes de software e suas conexões à infraestrutura.
3. **Documentação em Tempo Real**: Possibilita que informações sejam documentadas e atualizadas em tempo real, garantindo que todos os stakeholders tenham acesso às informações mais recentes sobre a arquitetura.
4. **Integração de Dados**: Permite a integração com outras fontes de dados, ajudando a alinhar as informações da arquitetura com processos de negócio existentes e outras ferramentas utilizadas.

**Exemplo**:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Juntando as Ferramentas**

Para integrar regras de negócio, infraestrutura e software utilizando Archimate e Ardoq, considere as seguintes etapas:

1. **Definição do Modelo de Domínio**: Use DDD junto com Archimate para criar um modelo que represente as regras de negócio, mapeando as interações entre os domínios.
2. **Criação de Diagramas**: Utilize Archimate para construir diagramas que conectem a camada de negócios à camada de aplicativos e tecnologia, identificando as dependências dos processos de negócio.
3. **Documentação Colaborativa**: Implemente Ardoq para documentar esses modelos de forma colaborativa, permitindo atualizações e visualizações em tempo real pelas equipes de negócios e TI.
4. **Análise de Impacto e Ajustes**: Use as capacidades analíticas de Archimate e a flexibilidade de Ardoq para avaliar e ajustar a arquitetura conforme necessário em resposta a mudanças nas regras de negócio.
5. **Feedback Contínuo**: Estabeleça um ciclo de feedback onde as equipes de negócios possam revisar e sugerir alterações nas representações da arquitetura, assegurando que reflitam com precisão as necessidades da organização.

Ao combinar Archimate e Ardoq, você consegue criar uma representação integrada e dinâmica da arquitetura empresarial, facilitando a colaboração e a compreensão entre todos os envolvidos.

**DDD**

DDD (Domain-Driven Design) é uma abordagem de desenvolvimento de software que foca na modelagem e no design de sistemas complexos, centrando-se na lógica de negócio e nas regras do domínio. O conceito principal é que o software deve refletir de maneira precisa o entendimento do domínio em que opera.

1. **Alinhamento com o Negócio**: DDD promove colaboração entre desenvolvedores e especialistas do domínio, garantindo que o software reflita as reais necessidades do negócio.
2. **Gerenciamento de Complexidade:** Cria um modelo de domínio claro e bem estruturado, facilitando a compreensão e a manutenção do sistema de acordo com mudanças de negócio, software ou infraestutura.
3. **Flexibilidade e Evolução**: Facilita a adaptação a mudanças nos requisitos de negócio, permitindo que a arquitetura evolua sem grandes reescritas.

**Arquitetura hexagonal**

A Arquitetura Hexagonal, também conhecida como Ports and Adapters, é um padrão de design de software que visa desacoplar a lógica de negócio de suas interações externas, como interfaces de usuário, bancos de dados e serviços de terceiros. O conceito foi introduzido por Alistair Cockburn.

1. **Desacoplamento**: A lógica de negócio é isolada das interações externas, permitindo que alterações em uma parte do sistema não afetem outras partes.
2. **Flexibilidade**: Facilita a troca de tecnologias (por exemplo, mudar o banco de dados ou a interface de usuário) sem impactar a lógica de negócio, pois as dependências são gerenciadas através de adaptadores.
3. Modularidade: A estrutura modular permite testar a lógica de **negócio** de forma isolada, usando mocks ou stubs para simular as interações externas.