THIAGO ADRIANO

PÓS TECH

SOFTWARE ARCHITECTURE

ARQUITETURA DE SOFTWARE

AULA 02

SUMÁRIO

O QUE VEM POR AÍ?

Nesta aula você vai aprender os primeiros passos na elaboração de uma boa arquitetura de software. Vamos avançar no processo de levantamento de requisitos até a modularização de um sistema.

HANDS ON

No nosso primeiro vídeo, nós aprendemos sobre a importância dos processos e da modelagem tática no levantamento de requisitos de um sistema, e como a linguagem ubíqua nos auxilia em todas as etapas de desenvolvimento.

Para reforçar o seu conhecimento sobre esta etapa de levantamento de requisitos, nós simulamos um fluxo onde um docente, o professor Kaku, foi o analista, e o professor Thiago foi o Stakeholder:

Analista:

- Quais as principais funcionalidades do sistema.
- Depois dessa etapa, qual seria a próxima?
- Quais bugs o sistema traz que mais travam vocês?

Stakeholder:

- A principal funcionalidade do sistema é cortar um vídeo em tempo real. Ex.: aconteceu algo agora, precisamos ter o vídeo no sistema para que possamos editá-lo e publicá-lo nas redes sociais dos nossos veículos.
- Escolher as redes sociais das editorias que tenham vínculo com o vídeo. Ex.: um acidente: deveríamos publicar o vídeo no Brasil Urgente. Se for algo de entretenimento: poderíamos subir no canal x.…
- Em alguns momentos o sistema cai e perdemos tudo que estávamos fazendo, direcionando para a tela de login.

Na próxima videoaula, o professor apresenta um print do REC na sua primeira versão, e depois apresenta algumas formas de dividir o código em módulos utilizando o C#.

No último vídeo desta aula, demonstramos como foi a quebra de módulos da nova versão do REC, conforme a figura 1 – “Módulos do sistema”:

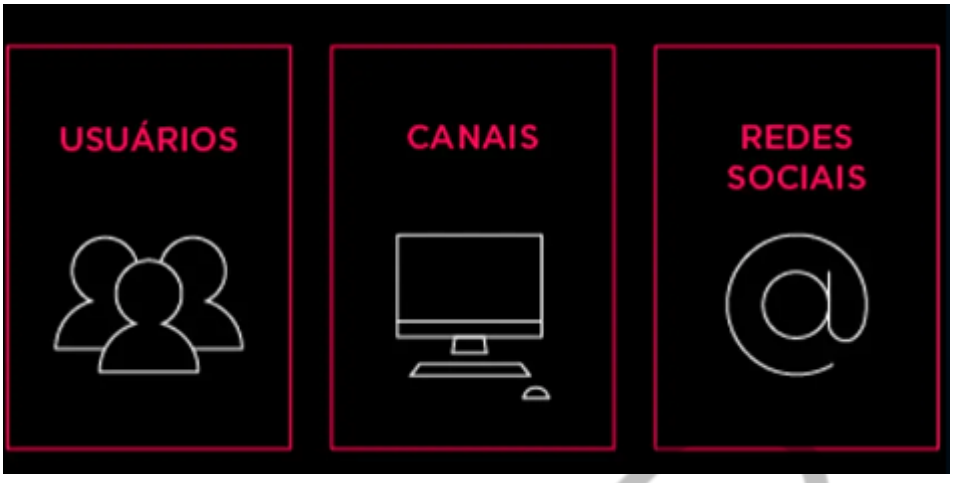

Figura 1 – Módulos do sistema
Fonte: Elaborado pelo autor (2023)

Cada um dos blocos da imagem acima é um módulo do sistema, como:

- Usuários (com gerenciamento de perfil).
- Canais (com gerenciamento).
- Redes sociais (para compartilhamento de conteúdo).

SAIBA MAIS

PROCESSO E MODULARIZAÇÃO

Processo de desenvolvimento de software é uma abordagem sistemática e disciplinada para o desenvolvimento de software, que visa garantir a qualidade, eficiência e eficácia do software produzido. Este processo é composto por várias etapas, como análise de requisitos, design, implementação, testes e manutenção. Cada uma dessas etapas é importante para garantir que o software produzido atenda aos requisitos do usuário e funcione corretamente.

Para quem já trabalha com DDD (Domain Driven Design), esta etapa seria a parte da modelagem tática, pois precisamos entrevistar os nossos clientes/ stakeholders (pessoas que conhecem o negócio), para que possamos fazer o levantamento de todos os requisitos.

Nesse momento é muito importante que o analista tenha em mente as regras de negócio, com foco em tudo que o sistema precisa ter para existir, pois esses detalhes serão cruciais para a arquitetura do sistema. Um ponto importante que precisa ser lembrado neste momento e que muitos analistas pecam por não registrar, seria a linguagem ubíqua, pois ela vai nos ajudar no decorrer do desenvolvimento do sistema, quando precisarmos voltar para conversar com os stakeholders.

Mas caso este seja o seu primeiro contato com DDD, nós temos uma disciplina focada neste assunto, onde você vai aprender tudo sobre esta abordagem, combinado?!

Mas para que você possa ter um melhor entendimento sobre alguns pontos cruciais que utilizaremos no curso, segue um resumo de alguns tópicos que utilizaremos sobre DDD.

Modelagem tática é um processo de análise e planejamento que se concentra na definição de ações específicas para alcançar objetivos operacionais e estratégicos de uma organização. Na modelagem tática, as equipes e líderes de uma empresa criam planos detalhados para atingir metas de curto e médio prazo, identificando os recursos e as atividades necessárias para executá-las.

Esse processo envolve a análise dos recursos da empresa, como pessoal, finanças, tecnologia e infraestrutura, bem como a avaliação do ambiente externo, como a concorrência e as tendências do mercado. Com base nessas informações, a equipe de liderança desenvolve planos táticos para melhorar a eficiência da empresa e aumentar sua capacidade de competir no mercado.

A modelagem tática é uma parte fundamental do planejamento estratégico geral de uma organização, pois ajuda a garantir que as metas e objetivos sejam alcançados em um nível mais operacional e concreto.

Linguagem ubíqua (também conhecida como linguagem onipresente) é uma técnica de modelagem de software que envolve a adoção de uma linguagem comum, clara e consistente que pode ser entendida por todos os envolvidos no desenvolvimento do software, incluindo os desenvolvedores, usuários, clientes e outras partes interessadas. Essa técnica busca criar uma comunicação mais efetiva entre os diferentes envolvidos no desenvolvimento do software, para que todos possam ter uma compreensão compartilhada dos requisitos, objetivos e funcionalidades do software em questão.

A linguagem ubíqua é usada para definir os conceitos e termos que são importantes para o negócio e para o software, e ajuda a garantir que todos os envolvidos no projeto estejam falando a mesma língua. Essa abordagem pode ser particularmente útil em projetos de software que envolvem muitas partes interessadas e que têm uma complexidade significativa.

Ela é frequentemente utilizada em conjunto com outras técnicas de modelagem, como a modelagem de domínio, para ajudar a criar uma representação mais clara e precisa do software e do negócio em questão. A adoção de uma linguagem ubíqua pode contribuir para uma melhor comunicação entre os envolvidos no projeto de software, ajudando a reduzir mal-entendidos e a garantir que o software produzido atenda aos requisitos e expectativas do usuário.

Como leitura complementar sobre este assunto, é recomendável ler o livro: “Implementando Domain-Driven Design”, do autor Vaughn Vernon.

Este é um livro que explora em profundidade a aplicação do Domain-Driven Design (DDD) na prática. O seu foco é demonstrar como a abordagem do DDD pode

ajudar a modelar o domínio do problema em questão e usar essa modelagem para guiar o design do sistema.

MODULARIZAÇÃO

Modularização de sistemas é uma técnica de design e organização de sistemas em que o sistema é dividido em módulos independentes que desempenham funções específicas. Cada módulo é um componente separado do sistema que pode ser desenvolvido, testado e mantido de forma independente dos outros módulos.

A modularização de sistemas tem várias vantagens. Em primeiro lugar, ela torna o sistema mais fácil de entender, pois cada módulo é responsável por uma função específica. Isso também torna o sistema mais fácil de modificar e atualizar, pois as mudanças podem ser feitas em um módulo sem afetar o resto do sistema. Além disso, a modularização pode tornar o sistema mais flexível, pois novos módulos podem ser adicionados facilmente sem afetar o restante do sistema.

A modularização de sistemas também pode ajudar a melhorar a qualidade do software. Como cada módulo é desenvolvido, testado e mantido separadamente, é mais fácil detectar e corrigir erros em cada módulo individualmente. Além disso, a modularização pode tornar o processo de desenvolvimento mais eficiente, pois diferentes equipes podem trabalhar em módulos diferentes simultaneamente.

No entanto, a modularização de sistemas requer uma abordagem cuidadosa e planejada. Os módulos devem ser cuidadosamente projetados para garantir que as interfaces entre eles sejam claras e bem definidas. Além disso, a modularização requer um planejamento cuidadoso para garantir que os módulos sejam organizados de uma maneira lógica e coerente.

Para ficar mais evidente, vejamos um exemplo de modularização de sistemas.

Imagine o seguinte cenário: você está trabalhando em um sistema de comércio eletrônico que consiste em vários módulos.

Um dos módulos pode ser responsável por gerenciar os pedidos, outro módulo pode ser responsável pelo processamento de pagamentos, outro pode lidar com o gerenciamento de estoque e assim por diante. Cada um desses módulos é

responsável por uma função específica e é desenvolvido, testado e mantido independentemente dos outros módulos.

Os módulos também podem se comunicar entre si por meio de interfaces claras e bem definidas. Por exemplo, o módulo de processamento de pagamentos pode se comunicar com o módulo de gerenciamento de pedidos para verificar se um pedido específico foi pago antes de marcá-lo como "processado".

A modularização desse sistema de comércio eletrônico traria várias vantagens. Por exemplo, seria mais fácil modificar e atualizar o sistema, já que as mudanças poderiam ser feitas em um módulo sem afetar o restante do sistema.

Também seria mais fácil detectar e corrigir erros, pois cada módulo é desenvolvido, testado e mantido separadamente. Além disso, diferentes equipes poderiam trabalhar em módulos diferentes simultaneamente, tornando o processo de desenvolvimento mais eficiente.

Em resumo, a modularização de sistemas pode ser aplicada a vários tipos de sistemas, incluindo sistemas de comércio eletrônico, sistemas de gerenciamento de estoque, sistemas bancários e muitos outros. A modularização ajuda a tornar o software mais fácil de entender, modificar e manter, o que pode resultar em um software de alta qualidade e eficiente.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula nós aprendemos sobre a importância dos processos de desenvolvimento de software e sobre a importância de modularizar os seus sistemas.

Não se esqueça de que também estamos disponíveis no Discord! Estamos com você em cada etapa do processo. Participe para poder conversar com os(as) colegas, experts e Community Manager da nossa equipe. Aproveite!

REFERÊNCIAS

EVANS, E. **Domain-Driven Design: Atacando as complexidades no coração do software**. Rio de Janeiro: Alta Books, 2016.

VERNON, V. **Implementando Domain-Driven Design**. Rio de Janeiro: Alta Books, 2013.

PALAVRAS-CHAVE

Arquitetura de software. Modularização. Processo de Desenvolvimento de Software.