

```
STATUS
ACCOUNT

class Banner
  attr_accessible :horiz, :link, :visible, :image, :position
  has_attached_file :image, styles: { vert: '220'

  before_create :assign_position

  def assign_position
    max = Banner.maximum(:position)
    self.position = max ? max + 1 : 0
  end
end
```

FULL STACK DEVELOPMENT
DOMAIN DRIVE DESIGN

AULA 08

SUMÁRIO

O QUE VEM POR AÍ?	3
HANDS ON	4
SAIBA MAIS	5
O QUE VOCÊ VIU NESTA AULA?	11
REFERÊNCIAS	12

EXEMPLO

O QUE VEM POR AÍ?

Bem-vindos e bem-vindas à mais uma etapa de aprendizado! Nesta aula vamos mergulhar no refinamento técnico no desenvolvimento de software, uma fase crucial que muitas vezes passa despercebida. Exploraremos como transformar requisitos de negócio em especificações técnicas detalhadas. Essa etapa é fundamental para garantir que as soluções propostas sejam viáveis e eficazes, minimizando os riscos técnicos e alinhando expectativas. Prepare-se para entender a importância desse processo e como ele pode impactar positivamente no sucesso dos seus projetos de software.

HANDS ON

Nesta seção prática, vamos acompanhar o processo de refinamento técnico de um Sistema de CRM Integrado. Discutiremos e analisaremos cada etapa da jornada do usuário no sistema de CRM, que vimos na aula anterior: desde a recepção de uma solicitação de orçamento por e-mail até a geração automatizada de propostas. Este é um exemplo real que ilustra a importância de entender e mapear cada detalhe do processo, garantindo a construção de uma solução técnica eficiente e adequada às necessidades do negócio. Vem com a gente!

SAIBA MAIS

Aqui, vamos aprofundar nosso entendimento sobre o refinamento técnico, uma etapa essencial no desenvolvimento de software. Discutiremos os desafios comuns enfrentados nos times de desenvolvimento quando essa etapa é negligenciada, como as considerações não mapeadas do desenvolvimento, o surgimento de bugs e de débitos técnicos. Exploraremos também as técnicas e estratégias utilizadas por especialistas para garantir um refinamento técnico eficaz, incluindo a análise da jornada da pessoa usuária, a definição de requisitos funcionais e não funcionais, e a arquitetura da solução.

Refinamento da jornada do usuário

Essa etapa tem como objetivo embasar a referência técnica que está participando desse refinamento. Na aula, fizemos o refinamento da Jornada do Usuário, anotando as dúvidas e considerações técnicas mapeadas.

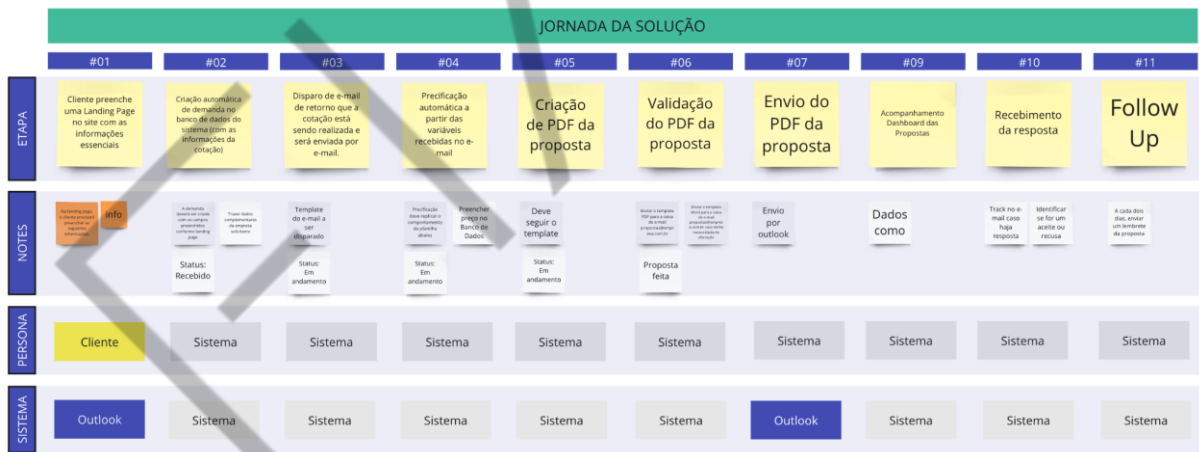


Figura 1 - A jornada da solução
Fonte: elaborado pelo autor (2024)

Nessa etapa, queremos mapear os requisitos técnicos necessários para que a solução resolva nosso problema e contemple todas as etapas da nossa jornada. É também nessa etapa que a liderança técnica ou a referência técnica da equipe tem a base para começar a pensar nas possibilidades técnicas para cumprir o objetivo do sistema.

Spikes e POCs

Os Spikes e POCs são métodos que usamos para validar soluções técnicas antes do desenvolvimento propriamente dito. Essas duas formas de validar as soluções são essenciais para mitigar os riscos de tecnologia da solução e garantir que nossa estratégia de desenvolvimento funciona para atingir o objetivo do desenvolvimento.

Indo para as definições, os Spikes são itens de desenvolvimento focados em explorar novas soluções e adquirir conhecimento. São feitos durante a sprint e comumente envolvem fluxos mais complexos, com validações mais complexas a serem feitas e por isso necessitam entrar na sprint.

As POCs (Proof Of Concept), ou Provas de Conceito, são um exercício de validação de uma ideia antes de levá-la para “produção”. Aqui em tecnologia, esse conceito se aplica à validação de soluções técnicas para ver se elas atendem nosso objetivo.

Com esses métodos, nós conseguimos sentir maior segurança ou não para resolver o problema, e nós temos mais clareza da dificuldade do desenvolvimento, nos permitindo dar estimativas mais próximas da realidade. Na aula, fazemos uma POC da solução do sistema CRM.

Desenho da Arquitetura

O desenho da Arquitetura de um software é um passo essencial para sabermos como as várias entidades e informações vão se comunicar. Para o nosso sistema de CRM, a arquitetura da solução poderia ficar como:

Front-end: interface da pessoa usuária desenvolvida com frameworks modernos como React ou Vue.js, proporcionando uma experiência de usuário fluida e responsiva. Esta camada se comunica com o back-end através de APIs RESTful ou GraphQL para buscar e enviar dados.

Back-end: construído com linguagens robustas, como Node.js, Python ou .NET Core, o back-end é responsável pela lógica de negócios, processamento de dados e armazenamento. Ele interage não só com o front-end, mas também com o banco de dados e outros serviços, como autenticação e autorização.

Banco de Dados: utilizamos sistemas de gerenciamento de banco de dados relacional (RDBMS) como PostgreSQL ou MySQL para armazenar informações do cliente, transações e interações. Para dados não estruturados ou de alta performance, pode-se optar por soluções NoSQL, como MongoDB.

Serviços de Autenticação: integração com serviços de autenticação segura, como Auth0 ou OAuth2, para garantir que os dados dos clientes sejam acessados apenas por usuários autorizados.

API Gateway: um ponto de entrada unificado para todas as requisições ao sistema, facilitando a gestão de APIs, a autenticação, a autorização e o monitoramento do tráfego. Isso ajuda a proteger e otimizar a comunicação entre os clientes e nossos serviços.

Serviços de Cloud e Microserviços: para garantir escalabilidade e flexibilidade, optamos por uma arquitetura baseada em microserviços, hospedados em plataformas de cloud como AWS, Google Cloud ou Azure. Isso permite que cada parte do sistema seja escalada independentemente, conforme necessário.

Sistema de Filas e Mensageria: para o processamento assíncrono de tarefas pesadas e integração entre diferentes partes do sistema, utilizamos sistemas de filas como RabbitMQ ou Kafka. Isso garante que as operações de back-end não bloqueiem a interação do usuário no front-end.

Ferramentas de Monitoramento e Log: implementação de soluções como ELK Stack (Elasticsearch, Logstash, Kibana) ou Prometheus e Grafana para monitorar a saúde da aplicação, desempenho e para diagnosticar problemas de forma proativa.

Segurança: além da autenticação e autorização, são implementadas medidas de segurança em todas as camadas, incluindo criptografia de dados em repouso e em trânsito, firewalls, e práticas de desenvolvimento seguro para proteger contra vulnerabilidades comuns.

Preenchimento do Requisito Técnico da Solução

Com as Provas de Conceito (POCs) realizadas e a Arquitetura da solução já desenhada, o preenchimento do requisito técnico da solução é realizado com base

nas considerações feitas durante o processo de refinamento. Neste contexto, o preenchimento deve abordar os seguintes tópicos:

Descrição Detalhada da Solução: apresentar uma visão completa da solução proposta, incluindo detalhes técnicos, componentes da arquitetura, e como cada parte contribui para atender aos requisitos do projeto.

Tecnologias e Ferramentas Utilizadas: especificar as tecnologias, linguagens de programação, frameworks, e ferramentas que serão utilizadas na implementação da solução, justificando a escolha de cada uma delas em relação aos requisitos e desafios técnicos identificados.

Integrações e Dependências: identificar quaisquer sistemas externos ou internos com os quais a solução precisará integrar-se, incluindo APIs, serviços de terceiros, bases de dados, etc., e descrever como essas integrações serão gerenciadas.

Estratégias de Implementação e Desenvolvimento: definir a metodologia de desenvolvimento, processos, e práticas que serão seguidas pela equipe técnica, incluindo abordagens ágeis, integração contínua/desdobramento contínuo (CI/CD), revisão de código, entre outros.

Segurança e Conformidade: descrever as medidas de segurança que serão implementadas para proteger a solução e os dados manipulados, bem como como a solução atende a regulamentos específicos ou padrões de conformidade relevantes.

Escalabilidade e Manutenibilidade: apresentar como a solução foi projetada para suportar crescimento e mudanças ao longo do tempo, incluindo estratégias para escalabilidade, balanceamento de carga, e manutenção.

Testes: detalhar o plano de testes para a solução, incluindo tipos de testes (unitários, integração, desempenho, segurança, etc.), ferramentas a serem usadas, e como o processo de testes será integrado ao ciclo de vida de desenvolvimento.

Documentação: especificar o tipo de documentação que será fornecida, abrangendo documentação técnica, manuais de usuário, e diretrizes de operação e manutenção.

Plano de Implantação: descrever o processo de implantação da solução, incluindo ambientes de desenvolvimento, teste, e produção, além de estratégias para rollout e rollback.

CrITÉrios de Aceitação e Validação: definir os critérios específicos que a solução deve atender para ser considerada completa e os métodos que serão utilizados para validar a solução contra esses critérios.

Estimativas

As estimativas de esforço no desenvolvimento de software é um assunto que gera muita polêmica e, em de forma geral, difícil de fazer. Muitas vezes, não levamos em consideração pontos importantes que aparecem como imprevistos durante o desenvolvimento. Outras vezes, simplesmente não conseguimos saber o quão difícil é fazer um código.

Nesse contexto, apresentamos duas maneiras de estimar. Uma delas, é o Planning Poker usando a sequência de Fibonacci. Damos notas de 1, 2, 3, 5, 8... considerando o tamanho da task e a complexidade técnica dela. Esse método é importante para provocar a discussão dos impactos do desenvolvimento na squad, mas dificilmente, sem uma base histórica, vamos conseguir ser assertivos nas estimativas.

A outra maneira é o método de Monte Carlo, que pode ser complementar às pontuações do Planning Poker. Ela traz os percentuais da probabilidade de um time conseguir entregar uma quantidade de pontos no período delimitado. Funciona assim:

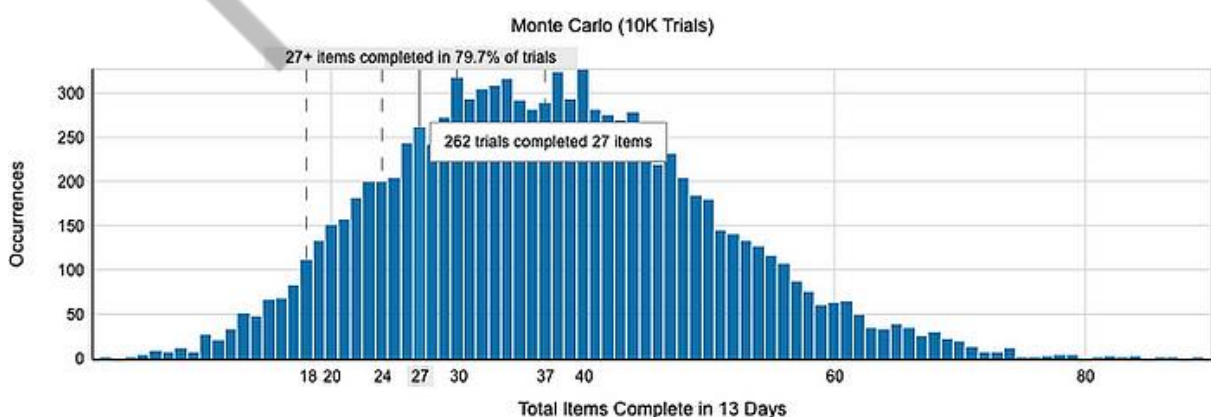


Figura 2 - Using Monte Carlo Forecasts in your Scrum Events

Fonte: Benjamin Huser-Berta - Medium (2023)

O gráfico da figura 2, conhecido como Monte Carlo na Agilidade, traz a quantidade de itens que um time consegue entregar com um percentual de assertividade. Por exemplo, no exemplo acima, retirado do artigo Using Monte Carlo Forecasts in your Scrum Events, do autor Benjamin Huser-Berta, o time analisado entrega 27 itens ou mais em 79,7% das sprints, o que nos diz que existe uma boa chance de entregarmos essa quantidade.



O QUE VOCÊ VIU NESTA AULA?

Nesta aula, exploramos a essência do refinamento técnico no desenvolvimento de software. Vimos como transformar requisitos de negócio em soluções técnicas viáveis, detalhando cada etapa do processo. Refletimos sobre as consequências de pular esta etapa crucial e aprendemos como ela é integrada no fluxo geral de desenvolvimento. Esta compreensão profunda do refinamento técnico é vital para o sucesso de qualquer projeto de software, garantindo soluções mais precisas, eficientes e alinhadas às necessidades do usuário final.

REFERÊNCIAS

AGARWAL, M. **Grooming in Agile Scrum**. 2023. Disponível em: <<https://www.techbeamers.com/agile-scrum-grooming/#:~:text=During%20grooming%2C%20Agile%20teams%20break,in%20next%203%2D4%20sprints.>>>. Acesso em: 07 fev. 2024.

HUSER-BERTA, B. **Using Monte Carlo Forecasts in your Scrum Events**. 2023. Disponível em: <<https://medium.com/serious-scrum/using-monte-carlo-forecasts-in-your-scrum-events-45ac3d37c2fd>>. Acesso em: 07 fev. 2024.

MCQUATER, R. **A Simplified Checklist for Technical Backlog Refinement**. 2021. Disponível em: <<https://spin.atomicobject.com/2021/01/04/technical-backlog-refinement/>>. Acesso em: 07 fev. 2024.

PALAVRAS-CHAVE

Palavras chave: Refinamento Técnico, Levantamento de Requisitos, Planning Poker.

EMENDAS

POS TECH