Fatec São José dos Campos
10 anos 2006-2016
Prof. Jessen Vidal

Centro Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

Curso Superior de Banco de Dados

Aprendizagem por Projetos Integrados (API)

Prof. Emanuel Mineda Carneiro
emanuel.mineda@fatec.sp.gov.br
São José dos Campos - SP

Roteiro

- API
- Desenvolvimento Ágil
- Papéis
- Exemplo

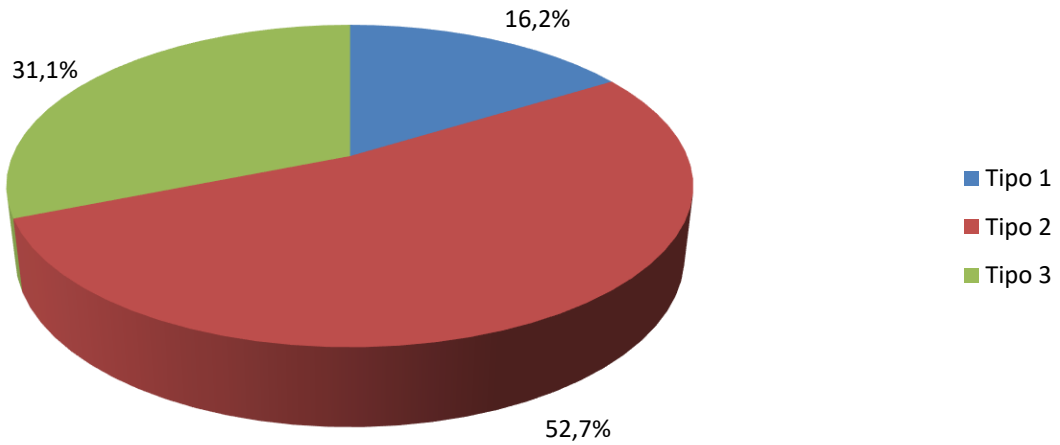
API

- **Objetivo**: Desenvolvimento de um aplicativo de Banco de Dados para uma empresa real
- **Disciplinas Participantes:** Arquitetura e Modelagem de Banco de Dados, Engenharia de Software, Laboratório em Desenvolvimento em Banco de Dados II, Linguagem de Programação I
- **P2:** Emanuel Mineda Carneiro
- **Tecnologias de uso obrigatório (requisitos não funcionais):**
  - **Linguagem de programação Java**
  - **Banco de Dados modelo relacional**
  - **Acesso ao banco de dados com JDBC**

Desenvolvimento Ágil

- **Motivação**: Fracasso da metodologia tradicional

The Standish Group – The Chaos Report
Projetos de Software - 1994



- **Tipo 1 –** Bem sucedido
- **Tipo 2 –** Desafiado (problema no prazo, custo ou qualidade)
- **Tipo 3 –** Fracassado

Desenvolvimento Ágil

- **Desenvolvimento tradicional**
  - Uma única entrega ao final do projeto

- **Desenvolvimento ágil**
  - Várias pequenas entregas

Desenvolvimento Ágil

- **Vantagens**
  - A cada entrega é possível receber *feedback* e corrigir possíveis problemas no processo de desenvolvimento
    - **Melhor qualidade – É possível identificar mais cedo problemas em requisitos**
    - **Menor retrabalho – A ocorrência de inconformidades aumenta proporcionalmente à quantidade de software desenvolvida**
  - Maior colaboração
    - **Multidisciplinaridade** – Há apenas uma única equipe, com todas as competências necessárias para desenvolver o produto
      - **Isso diminui a ociosidade e promove o comprometimento. Não existe o sentimento de "fiz a minha parte, agora é com a outra equipe"**
    - **Ambiente colaborativo** – Os membros da equipe trabalham em um ambiente sem divisórias, para facilitar a troca verbal de informações
    - **Comunicação** – Cada membro comunica seu progresso e suas dificuldades diariamente em uma cerimônia chamada *daily meeting*

Papéis

- **Scrum Master –** Facilitador (Remove obstáculos ao desenvolvimento)
  - Verifica se há deficiência de **conhecimento** ou **recursos**
  - Busca treinamentos e ajuda
  - Cuida para que a equipe não perca prazos importantes
  - Auxilia na resolução de conflitos internos
  - Garante que os processos sejam seguidos
  - **Ações esperadas:**
    - Garantir que cada tarefa possua data prevista de entrega
    - Garantir que cada tarefa seja trabalhar por 1 único aluno
    - Garantir que cada tarefa gere, no mínimo, 1 commit/push
    - Acompanhar por meio de daily meeting

Papéis

- **Product Owner (PO) –** Representante do Cliente
  - Detalha requisitos
  - Garante que as entregas agreguem valor ao cliente
  - Prioriza requisitos
  - **Ações esperadas:**
    - Identificar usuários da aplicação
    - Entender como funciona o processo a ser transformado em sistema
    - **Entender os problemas que motivaram o projeto**
    - Preparar User Stories antes do início dos Sprints
    - Conversar com a empresa parceira, no mínimo, 1 vez por semana

User Stories

- Uma *User Story* é uma funcionalidade que agrega valor ao negócio do cliente
  - **Não pode conter vocabulário técnico da equipe de desenvolvimento**
    - Exemplos:
      - **Correto**: Como consultor de vendas desejo poder consultar valor de produtos por meio de código de barras para agilizar o atendimento ao cliente
      - **Errado**: Desejo um **banco de dados** para armazenar informações de produtos
  - **Deve indicar claramente qual o valor agregado**
    - Exemplo: Como consultor de vendas desejo poder consultar valor de produtos por meio de código de barras **para agilizar o atendimento ao cliente**
  - **Deve apresentar quais usuários vão usufruir da funcionalidade**
    - Exemplo: Como **consultor de vendas** desejo poder consultar valor de produtos por meio de código de barras para agilizar o atendimento ao cliente
  - **Requisitos devem ter uma prioridade associada**
    - Quanto maior a prioridade, maior o valor agregado ao negócio

User Stories

- Uma *User Story* é uma funcionalidade que agrega valor ao negócio do cliente (cont.)
  - As expectativas do cliente devem ser documentadas. Isso diminui a chance de se entregar algo que o cliente não aprove
    - Qual a aparência desejada para as telas associadas ao requisito?
      - *Wireframes*

- Quais informações aparecerão? Onde? Em que momento?
- Como funciona a interação com cada item?
- Existem cálculos? Quais as fórmulas?
- Que validações devem ser executadas? Quais as mensagens de erro?

User Stories no API

- Antes de cada Sprint, o PO deve colher as *User Stories* com a empresa parceira e documentá-las no Git com o auxílio da equipe
- Cada *User Story* deve conter:
  - **Problema:** Deve ficar claro qual problema a *User Story* resolve
  - **Requisito:** Qual requisito do cliente está associado
  - **Prioridade**: Número inteiro que indica a ordem de desenvolvimento. Requisitos diferentes não podem ter a mesma prioridade! Exemplo: O requisito de prioridade 1 será o primeiro a ser desenvolvido no API
  - **Descrição** no formato: Como <tipo de usuário> desejo <funcionalidade> para <valor de negócio>
  - **Esboço de tela** (*wireframe*)
  - ***Definition of Done***: Lista de entregáveis acordada com a empresa parceira. Exemplo: código no github, guia de instalação, guia de usuário, etc

User Stories no API

- As *User Stories* a serem desenvolvidas na próxima Sprint devem ser informadas à empresa parceira durante a *Review*
  - *User Stories* devem ser previamente validadas com a empresa parceira
  - *User Stories* a serem desenvolvidas em uma Sprint devem se encontrar documentadas no Git até **o primeiro dia útil do Sprint**. **A não entrega do documento causará perda de pontos para toda a equipe**

User Stories no API

- Negociação de *User Stories*:
  - Alterações nas *User Stories* devem ser documentadas e enviadas por Slack à empresa parceira. A entrega realizada será avaliada tendo em vista as *User Stories* propostas
  - Os alunos devem propor *User Stories* à empresa parceira de acordo com o conhecimento e a capacidade de desenvolvimento da equipe
  - A aplicação entregue deve solucionar um problema. Aplicações com dados fixos e nenhuma funcionalidade não serão aceitas

Comunicação

- Ao se comunicar com pessoas envolvidas com o API usando Slack ou E-mail:
  - Comece com um cumprimento. Exemplo: “Bom dia”
  - Não use gírias
  - Use “por favor” ao solicitar algo
  - Faça perguntas objetivas, cuja resposta seja “Sim”, “Não”, uma única palavra ou uma lista de palavras
    - Exemplos de boas perguntas: “Bom dia. Para o requisito Gerenciamento de Jogos eu pensei numa tela de cadastro de jogos. Que informações vocês gostaria de armazenar para cada jogo?”; “Boa tarde. Para o campo de CPF da tela de clientes, será preciso validar se ele é válido?”
  - Os participantes trabalham e não tem disponibilidade todos os dias. Por conta disso, espere 2 dias até realizar um *follow-up*
    - Exemplo de *follow-up*: “Bom dia. Vocês teriam alguma posição com relação à minha última pergunta?”.
    - Caso não haja resposta ao *follow-up* em até 8 horas, procurem o P2

Dúvidas?