# Roteiro de exercícios sobre Gerência de projetos, disciplina Engenharia de Software e Sistemas

## Paulo Borba

## Centro de Informática

## Universidade Federal de Pernambuco

Adicione as respostas imediatamente após as perguntas, usando uma outra cor. Leia com calma as instruções, e reflita com cautela sobre as consequências de cada uma das atividades a seguir.

## Aula 1

1. Assumindo que você já criou um *fork* do projeto exemplo no roteiro de gerência de configuração, acesse o seu *fork* do projeto exemplo (não o [original](https://github.com/pauloborba/teachingassistant)). Para autorizar a criação de issues no seu *fork*, selecione a opção *settings* do seu *fork*, procure por *Issues* e clique no *checkbox* correspondente. Selecione a aba *Projects*, logo abaixo do nome do seu *fork* no GitHub, leia as instruções em *Learn More*, crie um projeto com o template *Automated kanban with reviews*, e cole a seguir uma imagem com o que é mostrado pelo seu *browser*.
2. Na aba *Issues*, você pode cadastrar novas tarefas a serem realizadas, e depois visualizá-las na aba *Projects* clicando em *Add cards* (no canto superior direito da aba *Projects*). O mesmo pode ser feito direto na aba *Projects*, primeiro adicionando uma nota (clicando em **+**) e depois convertendo-a para uma *issue*. Crie uma tarefa (*user-story*) "Desenvolver cenários de cadastro de auto-avaliação" e a coloque no *Backlog* (paínel *To do*). Observe os detalhes da tarefa, como tipo (*labels*), descrição, *milestone*, etc. Em ferramentas mais avançadas de gerenciamento, haveria informação também sobre a pontuação (*estimate*) da tarefa, a *epic* associada, e ajuda no cálculo dos limites de pontos a serem alocados em uma iteração. Qual a diferença entre o criador da tarefa e os responsáveis (*assignees*) pela mesma? Qual a utilidade do *Milestone*? Por que você acha que o desenvolvedor preferiu não criar uma tarefa em separado para cada possível cenário associado ao cadastro de auto-avaliação?
3. Observe os paíneis *To do* e *In progress*? Qual a diferença entre eles? Para que a ferramenta dá suporte aos dois tipos de paíneis? Em outras ferramentas, tem-se um outro paínel, tipicamente chamado de *icebox*, que é onde inicialmente colocamos tarefas desejáveis mas que ainda não foram priorizadas. Assim, as não priorizadas ficam no *icebox*, e as priorizadas ficam no *To do*, enquanto não são alocadas para execução em uma dada iteração. Quais as vantagens e desvantagens desse terceiro paínel?
4. Crie no *To do* as tarefas "Desenvolver cenários de carregamento de notas a partir de planilhas Excel" e "Desenvolver cenários de carregamento de alunos matriculados a partir de planilhas Excel". Na disciplina, focaremos em tarefas com essa granularidade. Basta preencher os campos principais, como tipo e descrição. Se eventualmente essas tarefas forem priorizadas para entrar no *In progress*, aí você poderá investir mais tempo nelas e preencher as demais informações. Cole a seguir uma imagem com o que é mostrado pelo seu *browser* após a criação das tarefas.
5. Uma *epic* pode ser usada para representar uma *feature*, e assim agrupar e visualizar o progresso das tarefas associadas a ela. Em ferramentas que permitem criar *epics*, qual seria a vantagem de associar as duas tarefas que você criou na questão anterior à *epic* "Importação de Excel"? No GitHub, pode-se simular grosseiramente o conceito de epic através dos *Labels* de uma tarefa.
6. Preencha as demais informações, como *assignee*, de uma das tarefas que você criou, e a mova para o paínel *In progress*. Cole a seguir uma imagem com o que é mostrado pelo seu *browser* após essa ação.
7. Finalize e conclua a tarefa que foi movida para o paínel *In progress*. Note que, em uma situação real, a conclusão da tarefa não será definida por você, desenvolvedor ou gerente (assumindo que ele não tem o papel de *product owner*). Por quem você acha que será?

## Aula 2

1. Considere que a tarefa "Desenvolver funcionalidade de exportação de relatórios de turmas" envolve a escrita do relatório em um determinado arquivo, e a tarefa "Desenvolver funcionalidade de análises de relatórios de turmas" envolve a leitura desse arquivo. Considerando que essas tarefas vão ser desenvolvidas em paralelo por dois desenvolvedores diferentes, que interface você definiria entre essas tarefas?
2. Considerando que a funcionalidade associada à tarefa "Desenvolver funcionalidade de análises de relatórios de turmas" vai ser disponibilizada como um serviço a ser utilizado por outros sistemas que estão sendo desenvolvidos em paralelo ao assistente de ensino, que interface você definiria entre os sistemas?
3. Assumindo que uma equipe de desenvolvimento de software realizou 30, 34, 36 e 35 pontos em quatro iterações consecutivas, indique qual a *velocity* da equipe, assumindo que a equipe contou com mais um desenvolvedor a partir da segunda iteração.
4. Assumindo que a velocidade atual da sua equipe de desenvolvimento é de 30 pontos, qual passará a ser a velocidade da mesma com a chegada de um novo desenvolvedor?
5. Considere que as seguintes tarefas estão no *backlog* do seu projeto: **T1** desenvolver cenários de cadastro de alunos (1 ponto, baixa prioridade), **T2** desenvolver cenários de cadastro de notas de alunos (1 ponto, baixa prioridade), **T3** desenvolver cenários de importação de arquivo Excel com notas de alunos (3 pontos, alta prioridade), **T4** desenvolver cenários de importação de arquivo Excel com dados dos alunos (3 pontos, baixa prioridade), **T5** desenvolver cenários de cálculo de média de alunos (2 pontos, alta prioridade), **T6** desenvolver cenários de notificação de notas e média de alunos (3 pontos, média prioridade), **T7** desenvolver cenários de auto-avaliação (3 pontos, média prioridade), **T8** refatorar a *feature* de controle de acesso (1 ponto, baixa prioridade).
   1. Supondo que a velocidade atual da equipe é de 7 pontos, que tarefas você escolheria para realizar na próxima iteração (*n*)? Por que?
   2. Considerando a mesma suposição, que tarefas você escolheria para realizar na iteração seguinte (*n+1*)? Por que?
   3. Supondo que o especialista na API de leitura de arquivos Excel não tivesse disponibilidade para trabalhar na iteração *n*, mas apenas na *n+1*, que tarefas você escolheria para realizar na iteração *n*? Por que? E que tarefas você escolheria para realizar na iteração *n+1*? Por que?
   4. Supondo que as tarefas **T1** e **T2** têm que ser realizadas ao mesmo tempo por dois desenvolvedores, qual a interface que o arquiteto deveria definir para que a implementação das funcionalidades pudesse ser feita de forma independente, sem excesso de comunicação entre os desenvolvedores?
   5. Supondo que você acabou de realizar a tarefa **T1**, que a tarefa **T2** está sendo desenvolvida por um outro desenvolvedor, e que as tarefas **T3** e **T7** estão programadas para a iteração atual, qual das duas você escolheria como a sua próxima tarefa a ser realizada? Por que?
   6. Supondo que não houvesse na equipe de desenvolvimento especialistas na API de notificação a ser utilizada pela tarefa **T6**, e que o uso da mesma pudesse afetar significativamente a definição da arquitetura do sistema, você deveria antecipar ou postergar a realização dessa tarefa?
   7. Que pontuação e prioridade você daria para uma tarefa de desenvolvimento da *feature* de cadastro de professores? Por que? E uma tarefa de desenvolvimento da *feature* de cálculo da média de alunos reprovados em um conjunto de turmas? Por que?

## Aula 3

1. Cadastrar as atividades do projeto como *issues* do GitHub, e gerenciar o projeto
   1. No caso de projeto que não foi criado pelo professor
      1. criar, no GitHub, para a equipe, o repositório do projeto (inicialmente só com um arquivo README) e enviar o *link* para o canal do projeto no Slack
      2. cada membro da equipe cria o seu *fork*, caso a equipe tenha decido por essa organização
      3. verificar se o professor deu *fork* no seu projeto
      4. nas entregas futuras do projeto da disciplina, enviar PR para o *fork* que será criado pelo professor, não para o seu repositório
   2. No caso de projeto que foi criado pelo professor
      1. criar, no GitHub, para a equipe, um *fork* do repositório do projeto
      2. nas entregas futuras do projeto da disciplina, enviar PR para o repositório original do professor
   3. Cada aluno cadastra as suas tarefas **no repositório GitHub da equipe** (todos têm que ter permissão de escrita)
      1. uma para cada cenário, para juntamente detalhar e especificar o cenário, implementar os testes associados, e implementar a funcionalidade associada na aplicação
      2. pelo menos uma tarefa de refatoração
      3. coloque a seguir os *links* para as tarefas criadas
   4. A equipe, em conjunto, prioriza as tarefas e define duas iterações curtas (as tarefas da primeira iteração ficam em *In Progress*, as da segunda ficam em *To do*)
      1. cada iteração com pelo menos duas tarefas por aluno
      2. o objetivo de cada iteração é gerar uma versão funcional do produto, que possa prontamente ser usada pelo cliente, mesmo que não tenha tantas funcionalidades; o importante é que as funcionalidades presentes estejam funcionando bem
      3. coloque a seguir uma imagem dos paíneis GitHub do seu projeto, ressaltando as suas tarefas
   5. A equipe, em conjunto, realiza as reuniões de acompanhamento (*daily scrum meeting*)
      1. no *slack*, ou
      2. presencialmente (e cada um envia o seu resumo para o *slack*)
      3. pelo menos nos dias de aula de acompanhamento, mas idealmente todo dia em que trabalhar no projeto