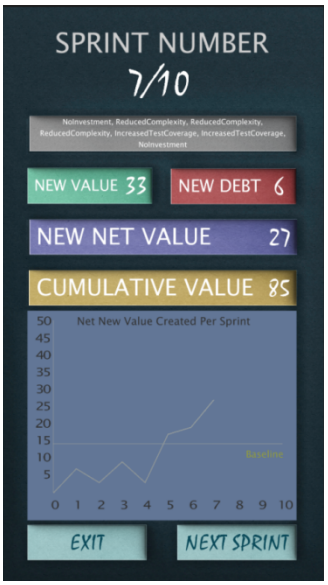


Technical Debt

Contexto

O objetivo desta atividade de aprendizagem é apresentar o conceito de *Technical debt no* contexto do desenvolvimento de software, fazendo com que experienciem o seu impacto e para que tentem evitá-lo.

Objetivos de Aprendizagem e Resultado

Depois de jogar neste cenário, os alunos serão capazes de:

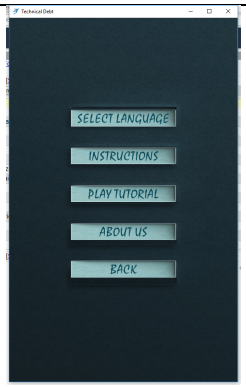
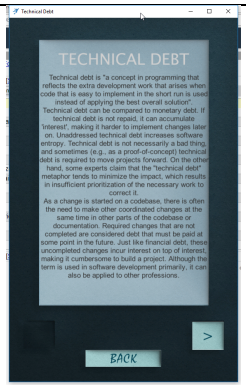
- Compreender o conceito de "technical debt".
- Aplicar os melhores procedimentos e técnicas para equilibrar a *technical debt* num projeto de desenvolvimento de software

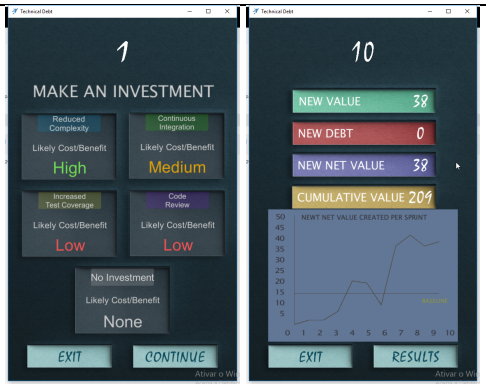
Como utilizar o LEAP

Neste jogo, o jogador assume o papel de gestor de uma equipa de desenvolvimento de software, a trabalhar num projeto que durará 10 *sprints*. O seu trabalho é criar o maior valor de software até o final do projeto. Para isso, o jogador tem de equilibrar o processo de desenvolvimento de software que gera novo valor (NV), mas aumenta a *technical debt* (TD) e a redução real da TD por meio de várias medidas de investimento disponíveis.

Durante cada turno, a equipa de desenvolvimento de software tem uma capacidade finita de criar valor de software e lidar com *technical debt*. No início do jogo, o jogador tem um certo número de pontos disponíveis para criar NV e um certo número de pontos para TD.

Em cada turno (representando um sprint), o jogador lança um dado para criar valor e conta o total de todos os pontos obtidos. Depois, o jogador lança o dado da *technical debt* e totaliza esse número. O novo valor líquido (NNV) criado em cada turno é o total do NV menos o total do DT.

O ecrã inicial permite iniciar um novo jogo ou ver as opções de configuração.	
O menu de opções permite configurar a linguagem, ver algumas instruções e jogar um tutorial. A primeira vez que se joga, o jogador deve verificar as instruções e o tutorial. Mostre aos alunos como mudar o idioma do jogo.	 
Jogar o tutorial é importante porque permite ver qual o efeito da aplicação de um investimento. **Complexidade reduzida** Efeito: remova 7 pontos em média da TD e adicione-os ao NV. Custo: 7 pontos em média de NV para TF por 3 sprints. **Integração continua** Efeito: remova 3,5 pontos em média da TD e adicione-os ao NV. Custo: 10,5 pontos em média de NV para TD para 2 sprints. **Maior cobertura de teste** Efeito: remova 3 pontos da TD a cada sprint. Custo: 3,5 pontos em média de NV para TD por 3 sprints. **Revisão de código**	

Efeito: Baixa o DT por um valor aleatório. Custo: 3,5 pontos em média de NV para TD para 2 sprints. **Nenhum investimento** Efeito: nenhum Custo: nenhum	
Comece o jogo e explique as cinco opções aos alunos, apenas para recapitular os conceitos aprendidos no tutorial. Depois disso, deixe os jogadores escolherem as opções e seguirem o ciclo de 10 sprints.	
No final, verifique os resultados de todos os alunos. Peça aos alunos com a pontuação mais alta que expliquem o seu método aos restantes.	

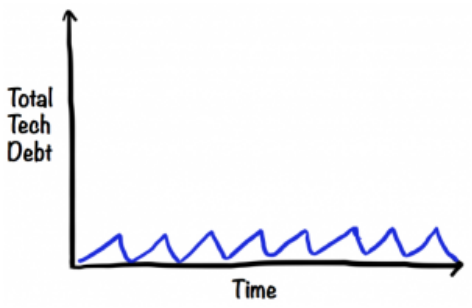
Colaboração da turma

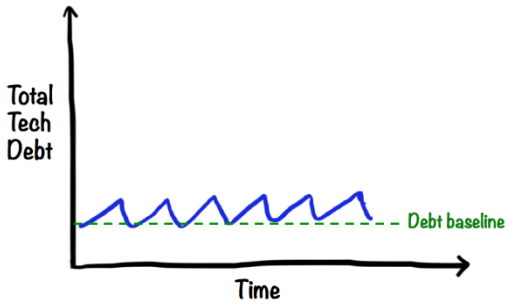
Perguntas para orientar a discussão na sala de aula:

- Faz sentido não desenvolver a melhor solução imediatamente?
- A necessidade de recalcular implica que o código existente é mau e ineficiente?
- O conceito de “technical debt” faz sentido em outras tecnologias de desenvolvimento de software, como nos modelos em cascata?
- Qual seria a curva ideal para a dívida técnica?

Avaliação

Peça aos alunos que joguem e avaliem a curva da dívida técnica que alcançaram, comparando com a curva “ideal” (imagem da esquerda) ou “real ideal” (imagem da direita), propostas por Henrik Kniberg. Faça os alunos explicarem a relação da curva que alcançaram com essas duas.





Materiais auxilares

- Techopedia, Technical debt: https://www.techopedia.com/definition/27913/technical-debt
- Wikipedia, Technical debt: https://en.wikipedia.org/wiki/Technical_debt
- Ward Cunningham, Debt Metaphor: https://www.youtube.com/watch?v=pqeJFYwnkjE
- Steve McConnell, 10x Software Development Best Practices: Technical Debt: http://www.construx.com/10x_Software_Development/Technical_Debt/
- Henrik Kniberg, Good and Bad Technical Debt: http://blog.crisp.se/2013/10/11/henrikkniberg/good-and-bad-technical-debt