

ENGENHARIA DE SOFTWARE

DAMAS

TAMG

Universidade Federal Fluminense

Relatório 1: Projeto Jogo de Damas

Equipe:

Alan Gomes (Inteligência Artificial)

Gabriel Ferreira (Implementação e testes)

Matheus Pimentel (Implementação e testes)

Thiago Freitas (Gerencia, Design, implementação UI)

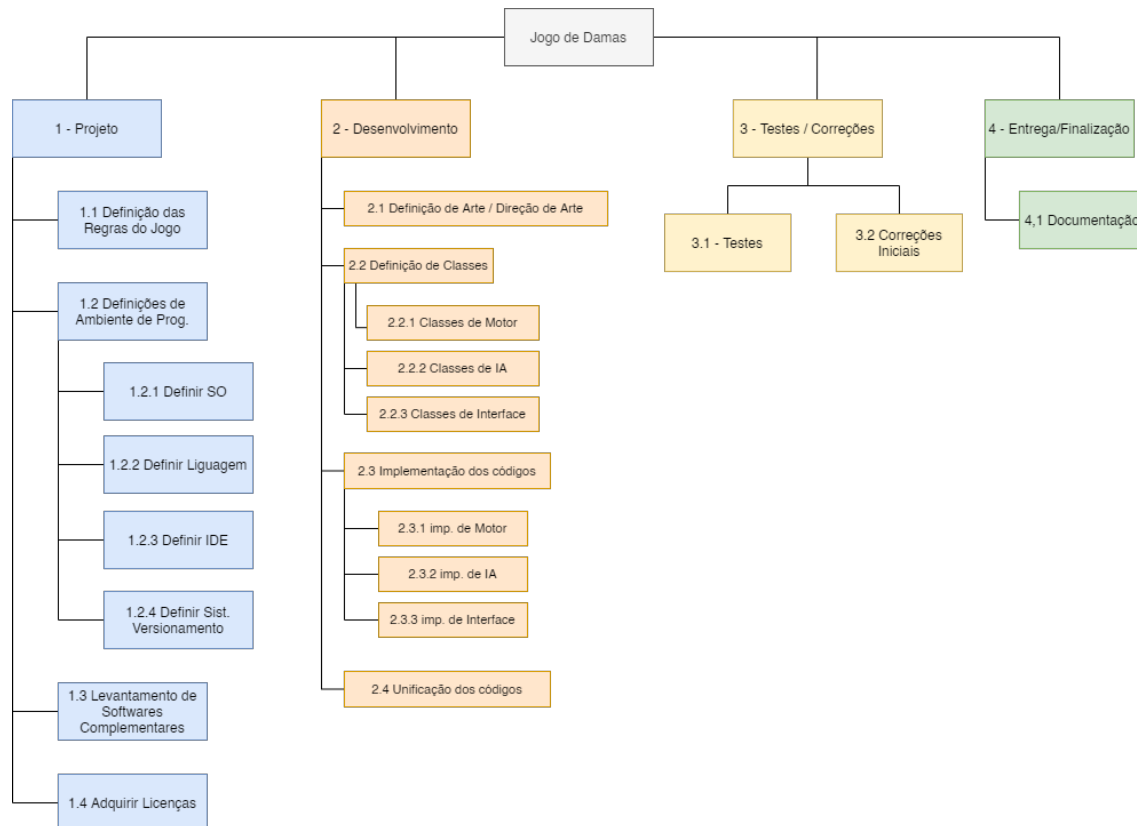
Ano: 2020

Data do Doc.: 14/10/2020

Repositório do Projeto: <https://github.com/thiagofsf/tamg-damas>

EAP

O EAP foi desenvolvido considerando pacotes de tarefas mais generalizados, nas iterações (ou sprints) esses pacotes foram (e serão) desmembrados em tarefas menores.



DESCRIÇÃO DAS ATIVIDADES

1. Projeto

1.1. Definição das Regras do Jogo

Atividade: Escrever um documento com as regras a serem implementadas no jogo.

Recursos: 1 membro (Alan);

Tempo: 3 dias;

Custo: R\$0,00;

Dependência:

Nenhuma Dependência;

1.2. Definições de Ambiente de Programação

1.2.1. Definir Sistema Operacional

Atividade: Definir o sistema operacional para desenvolvimento e para compatibilidade do produto final.

Recursos: Todos os membros, conexão e equipamento para videoconferência;

Tempo: 10 min;

Custo: R\$0,00;

Dependência:

Nenhuma Dependência;

1.2.2. Definir Linguagem

Atividade: Definir a linguagem para desenvolvimento.

Recursos: todos os membros, conexão e equipamento para videoconferência;

Tempo: 10 min;

Custo: R\$0,00;

Dependências:

Nenhuma Dependência;

1.2.3. Definir IDE

Atividade: Definir se haverá padronização de ambiente de desenvolvimento e qual.

Recursos: todos os membros, conexão e equipamento para videoconferência;

Tempo: 10 min;

Custo: R\$0,00;

Dependências:

- Sistema Operacional Definido;

- Linguagem Definida;

1.2.4. Definir Sistema de Versionamento

Atividade: Definir sistema de versionamento a ser usado.

Recursos: todos os membros;

Tempo: 10 min;

Custo: R\$0,00;

Dependências:

Nenhuma Dependência;

Repositório Escolhido: GitHub;

Atividade: Criar repositório, adicionar membros.

Recursos: Um membro (Thiago), computador conectado à internet;

Tempo: 30min;

Custo: R\$0,00;

Dependências:

- Sistema de Versionamento escolhido;

1.3. Levantamento de Softwares Complementares

Atividade: Definir outros softwares que possam ser necessários durante as fases do projeto, incluindo edição de textos e criação de artes.

Recursos: Todos os membros;

Tempo: 10 min;

Custo: R\$0,00;

Dependências:

- Ambiente de desenvolvimento definido (item 1.2);

1.4. Adquirir Licenças

Atividade: Adquirir as licenças de softwares;

Recursos: Todos os membros;

Tempo: 10 min;

Softwares levantados + custo:

PyCharm: US\$20/mês/usuário -> US\$80/mês;

MS Office: R\$30/mês (inclui 6 pessoas);

Adobe Creative Cloud: R\$224,00/mês (só 1 licença será usada);

Dependências:

- Levantamento de Softwares Complementares Concluído;

2. Desenvolvimento

2.1. Definição de Arte / Direção de Arte

Atividade: Elaboração da aparência do jogo, sprites, visual, telas etc.

Recursos: 1 membro (Thiago);

Tempo: 3 dias;

Custo: R\$0,00;

Dependências:

Sem dependências;

Atividade: Exportação dos sprites.

Recursos: 1 membro (Thiago);

Tempo: 1 dia;

Custo: R\$0,00;

Dependências:

- Elaboração gráfica concluída;

2.2. Definição de Classes

2.2.1. Classes de Motor

Atividade: Definir em um documento as classes que serão usadas para execução do jogo em si. As classes de motor são as que guardam o estado do jogo e das peças de cada jogador. No documento definir a entrada e saída de cada classe (documentação).

Recursos: 2 membros;

Tempo: 5 dias;

Custo: R\$0,00;

Dependências:

- Etapas de Projeto Concluídas;

2.2.2. Classes de IA

Atividade: Definir em um documento as classes que serão usadas para execução do jogador máquina. As classes definidas analisam o estado do “motor” e opera uma jogada, fornecendo como saída parâmetros de entrada para a classe motor (ou jogo).

Recursos: 2 membros;

Tempo: 5 dias;

Custo: R\$0,00;

Dependências:

- Etapas de Projeto Concluídas;

2.2.3. Classes de Interface

Atividade: Definir em um documento as classes que serão usadas para a interface, essas classes analisam o motor (jogo) e exibe na tela o estado atual do jogo.

Recursos: 2 membros;

Tempo: 5 dias;

Custo: R\$0,00;

Dependências:

- Etapas de Projeto Concluídas;

2.3. Implementação

2.3.1. Implementação de Motor

Atividade: Implementação das Classes criadas na etapa 2.2.1, a implementação será subdividida de acordo com a quantidade de classes criadas.

Recursos: 2 membros;

Tempo: 15 dias;

Custo: R\$0,00;

Dependências:

- Definição de classes (etapa 2.2.1);

2.3.2. Implementação de IA

Atividade: Implementação das classes criadas na etapa 2.2.2, a implementação será dividida de acordo com a quantidade de classes criadas.

Recursos: 2 membros;

Tempo: 15 dias;

Custo: R\$0,00;

Dependências:

- Definição de classes (etapa 2.2.2);

2.3.3. Implementação da Interface

Atividade: Implementação das classes criadas na etapa 2.2.3, a implementação será dividida de acordo com a quantidade de classes criadas.

Recursos: 2 membros;

Tempo: 15 dias;

Custo: R\$0,00;

Dependências:

- Definição de classes (etapa 2.2.3);

2.4. Testes de Unificação dos Códigos

Atividade: juntar códigos das 3 áreas desenvolvidas (Motor, IA e Interface) e verificar o funcionamento.

Recursos: todos os membros;

Tempo: 2 dias;

Custo: R\$0,00;

Dependências:

- Implementação de classes (passo 2.3);

3. Testes / Correções

3.1. Testes

Atividade: Elaborar e implementar os testes para verificar se todas as partes da aplicação estão funcionando corretamente.

Recursos: 2 pessoas;

Tempo: 8 dias;

Custo: R\$0,00

Dependências:

-Etapas de Desenvolvimento Concluídas;

3.2. Correções Iniciais

Atividade: A partir dos resultados dos testes, elaborar a correção das falhas observadas inicialmente antes da entrega.

Recursos: Todos os membros;

Tempo: 5 dias;

Custo: R\$0,00;

Dependências:

- Testes (etapa 3.1);

4. Entrega / Finalização

Atividade: Entrega da aplicação / disponibilização.

Recursos: 1 membro;

Tempo: 1 dia;

Custo: R\$0,00;

Dependências:

- Etapas de Desenvolvimento Concluídas (etapa 2);

4.1. Documentação

Atividade: Desenvolver a documentação da aplicação.

Recursos: Todos os membros;

Tempo: 30 dias;

Custo: R\$0,00;

Dependências:

- Etapas de Desenvolvimento Concluídas (etapa 2);

- Etapas de Testes e Correções Concluídas (etapa 3);

CUSTOS

- Custos de Recursos Humanos:

- Hora de Trabalho: R\$50,00/h;
- Estimativa: 20h/sem -> 80h/mês -> R\$4.000,00/mês;
- 4 membros: R\$16.000,00/mês;
- Duração estimada do projeto: 2 meses -> R\$32.000,00;

- Custos de Softwares:

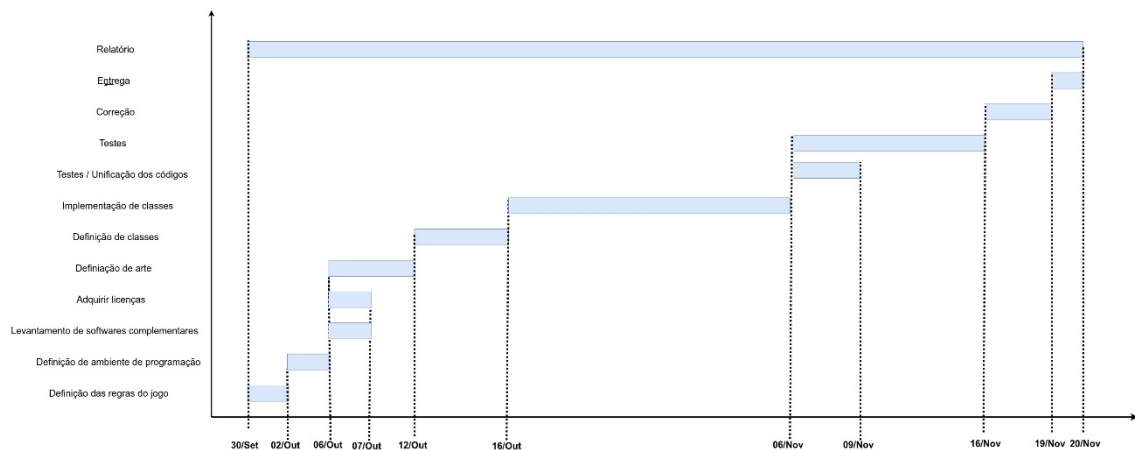
- PyCharm:
 - Custo: US\$20/mês -> aprox. R\$111,00/mês;
 - Tempo de uso: 2 meses -> aprox. R\$222,00;
- Office:
 - Custo: R\$30,00/mês;
 - Tempo de uso: 2 meses -> R\$60,00;
- Adobe Creative Cloud:
 - Custo: R\$224,00/mês;
 - Tempo de uso: 2 meses -> R\$448,00;

- Custos de Estrutura:

- Energia Elétrica:
 - Custo (a cada 1h): Aprox.: R\$10,00/membro (via simulador Enel);
 - Total de horas estimado: 160h -> R\$1.600,00;
 - Total: 4 membros x 160h -> R\$3.400,00;
- Internet:
 - Custo: R\$100,00/mês -> 4 membros: R\$400,00;
 - Total estimado: 2 meses -> R\$800,00;

- Custo total Estimado: R\$36.930,00.

DIAGRAMA DE GANTT



PLANO DE RISCOS

Risco 1:

- Cenário 1: **Custos do ambiente de Programação**
- Risco 1: Falta de orçamento para implementação.
- Probabilidade: 20% (aumento dos requisitos exigidos)
- Impacto: 0,6 (maior custo de recursos humanos, softwares e estrutura)
- Exposição: 0,12
- Prioridade: alta

		impacto					
probabilidade		0	0,2	0,4	0,6	0,8	1
	100%	0	0,2	0,4	0,6	0,8	1
	80%	0	0,16	0,32	0,48	0,64	0,8
	60%	0	0,12	0,24	0,36	0,48	0,6
	40%	0	0,8	0,16	0,24	0,32	0,4
	20%	0	0,4	0,8	0,12	0,16	0,2
	0%	0	0	0	0	0	0

- Mitigação:
 - Plano de contenção: exigir do fornecedor um orçamento detalhado.
 - Plano de contingência: obter garantia desses gastos, usando softwares gratuitos, diminuição da qualidade do projeto.

- Monitoramento:
 - A cada semana
 - calcular os custos gerados na semana
 - verificar se está dentro dos orçamentos
 - atualizar a avaliação de probabilidade e impacto do risco

Risco 2:

- Cenário 2: **Problemas de conectividades**
- Risco 3: Com o verão, o clima mais propenso a tempestades causando quedas de internet e energia elétrica.
 - Probabilidade: 5% (quedas de energia e internet)
 - Impacto: 0,8 (atraso do projeto)
 - Exposição: 0,04
 - Prioridade: baixa

	impacto						
		0	0,2	0,4	0,6	0,8	1
probabilidade	100%	0	0,2	0,4	0,6	0,8	1
	80%	0	0,16	0,32	0,48	0,64	0,8
	60%	0	0,12	0,24	0,36	0,48	0,6
	40%	0	0,8	0,16	0,24	0,32	0,4
	20%	0	0,4	0,8	0,12	0,16	0,2
	5%	0	0,01	0,02	0,03	0,04	0,05
	0%	0	0	0	0	0	0

- Mitigação:
 - Plano de contenção: proteger os eletrônicos contra quedas e oscilações de energia com estabilizadores, filtros de linha ou nobreaks.
 - Plano de contingência: contratação de um plano de 4g caso o problema seja queda de internet. Para queda de energia elétrica uma opção seria providenciar um nobreak.
- Monitoramento:
 - A cada semana
 - revisar se houve quedas de energia ou internet
 - reajustar o planejamento do tempo perdido

Risco 3:

- Cenário 3: **Dificuldade de gerenciamento de tempo**
- Risco 3: dificuldade de gerenciamento do tempo em função de atividades paralelas (outras disciplinas)
 - Probabilidade: 40% (provas e trabalhos na mesma semana)
 - Impacto: 0,8 (atraso do projeto)
 - Exposição: 0,32
 - Prioridade: alta

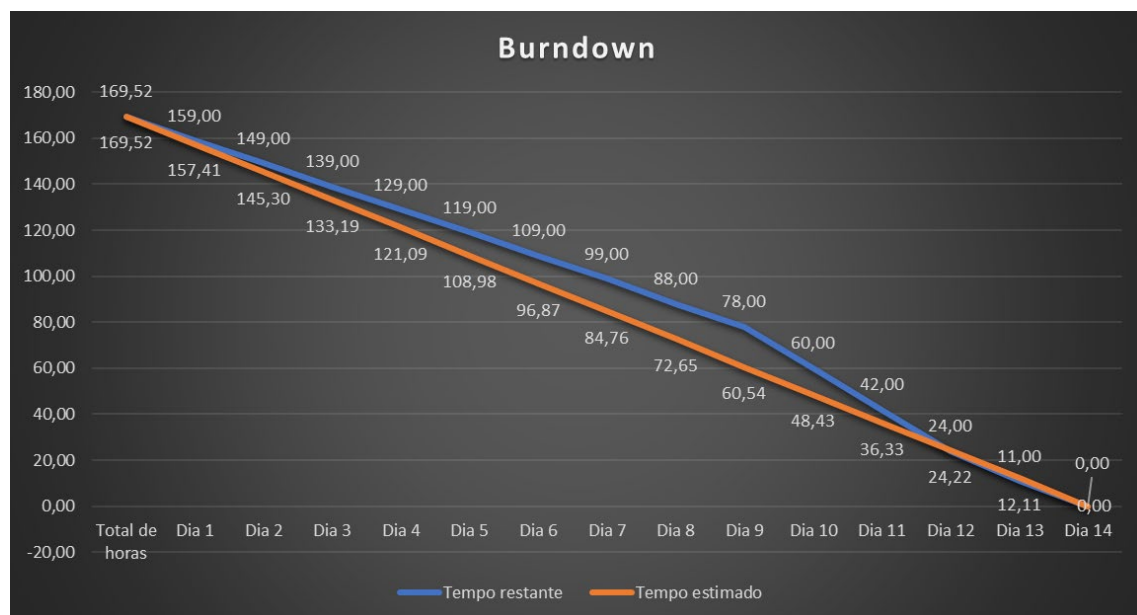
	impacto						
		0	0,2	0,4	0,6	0,8	1
probabilidade	100%	0	0,2	0,4	0,6	0,8	1
	80%	0	0,16	0,32	0,48	0,64	0,8
	60%	0	0,12	0,24	0,36	0,48	0,6
	40%	0	0,8	0,16	0,24	0,32	0,4
	20%	0	0,4	0,8	0,12	0,16	0,2
	5%	0	0,01	0,02	0,03	0,04	0,05
	0%	0	0	0	0	0	0

- Mitigação:
 - Plano de contenção: planejamento do tempo em função das atividades paralelas ao projeto.
 - Plano de contingência: organização de atividades em sistemas de Kanban(trello)
- Monitoramento:
 - A cada semana
 - planejar as atividades no trello
 - reajustar o tempo de acordo com as atividades.

MONITORAMENTO E CONTROLE

Sprint Backlog - Lista das tarefas do Sprint 001 (Semanas 30/09/2020 a 14/10/2020)

- Definir sistema operacional;
- Definir linguagem;
- Definir IDE;
- Definir sistema de versionamento;
- Definir outros *softwares* complementares;
- Adquirir as licenças dos *softwares*;
- Criar repositório no sistema de versionamento escolhido;
- Incluir membros no sistema de versionamento escolhido;
- Definir regras do jogo;
- Criar logomarca;
- Elaborar aparência geral do jogo;
- Elaborar tela de início do jogo;
- Elaborar tela de informações do jogo
- Elaboração parcial da tela do jogo
- Elaborar *sprites*;
- Exportar *sprites*;



Análise do valor agregado:

- O projeto tem custo de R\$36.930,00
- A duração planejada é de 8 semanas
- Estamos na 3ª semana
- Cálculo dos custos até o momento:

Mão de obra: 169,52 (horas totais de todos os membros) * R\$50 (mão de obra) = R\$8476,00

Softwares: R\$111 (PyCharm) + R\$30 (Office) + R\$224 (Adobe) + R\$400 (Internet para os 4 membros) = R\$765,00

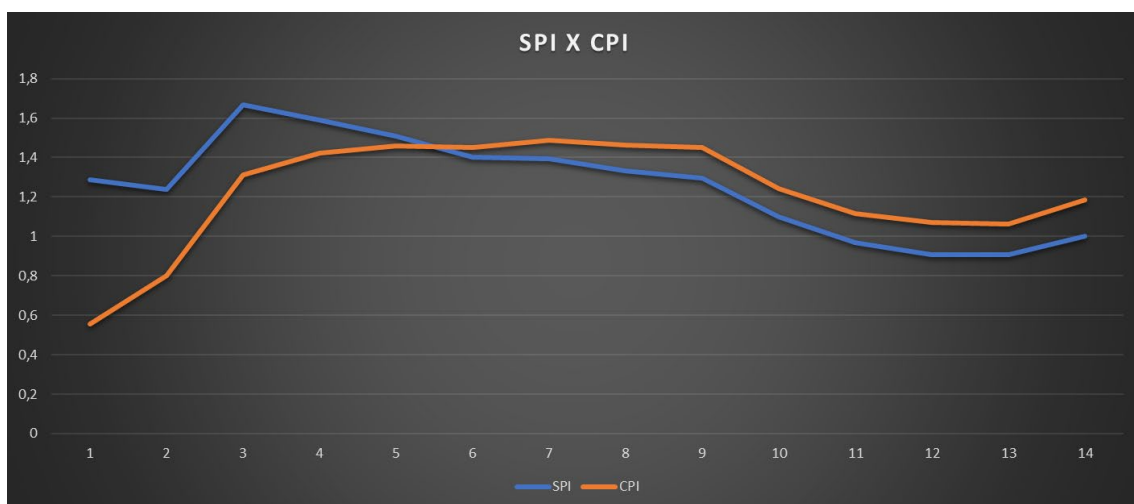
Energia Elétrica: 169,52 * R\$10 = R\$1695,20

Custo total: R\$10.936,20

Atividades	Total de horas	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6	Dia 7	Dia 8	Dia 9	Dia 10	Dia 11	Dia 12	Dia 13	Dia 14
Definir sistema operacional	0,17	0,17													
Definir linguagem	0,17	0,17													
Definir IDE	0,17	0,17													
Definir sistema de versionamento	0,17	0,17													
Definir outros softwares complementares	0,17	0,17													
Adquirir as licenças dos softwares	0,17	0,17													
Criar repositório no sistema de versionamento escolhido	0,25	0,25													
Incluir membros no sistema de versionamento escolhido	0,25	0,25													
Definir regras do jogo	72,00	9,00	10,00	10,00	10,00	10,00	10,00	10,00	3,00	8,00	4,00				
Criar logomarca	12,00														
Elaborar aparência geral do jogo	12,00									6,00	5,00				
Elaborar tela de início do jogo	12,00									10,00	2,00				
Elaborar tela de informações do jogo	12,00									2,00	10,00				
Elaboração parcial da tela do jogo	12,00										6,00	6,00			
Elaborar sprites	12,00											10,00	2,00		
Exportar sprites	24,00											2,00	11,00		11,00
Tempo restante	169,52	159,00	149,00	139,00	129,00	119,00	109,00	99,00	88,00	78,00	60,00	42,00	24,00	11,00	0,00
Tempo estimado	169,52	157,41	145,30	133,19	121,09	108,98	96,87	84,76	72,65	60,54	48,43	36,33	24,22	12,11	0,00
Soma de horas		10,52	10,00	10,00	10,00	10,00	10,00	10,00	11,00	10,00	18,00	18,00	18,00	13,00	11,00
Custo total	R\$ 10.936,20														
Valor planejado		R\$ 678,67	R\$ 1.323,80	R\$ 1.968,93	R\$ 2.614,06	R\$ 3.259,18	R\$ 3.904,31	R\$ 4.549,44	R\$ 5.259,08	R\$ 5.904,21	R\$ 7.065,44	R\$ 8.226,66	R\$ 9.387,89	R\$ 10.226,56	R\$ 10.936,20
Percentual planejado de conclusão		6%	12%	18%	24%	30%	36%	42%	48%	54%	65%	75%	86%	94%	100%
Valor agregado		R\$ 874,90	R\$ 1.640,43	R\$ 2.280,86	R\$ 2.921,29	R\$ 3.561,72	R\$ 4.202,15	R\$ 4.842,58	R\$ 5.483,01	R\$ 6.123,44	R\$ 7.284,67	R\$ 8.445,90	R\$ 9.607,13	R\$ 10.768,36	R\$ 11.929,59
Percentual real de conclusão		8%	15%	30%	38%	45%	50%	58%	64%	70%	71%	73%	78%	85%	100%
Custo real		R\$ 1.570,00	R\$ 2.048,00	R\$ 2.498,00	R\$ 2.918,00	R\$ 3.368,00	R\$ 3.768,00	R\$ 4.268,00	R\$ 4.788,00	R\$ 5.278,00	R\$ 6.258,00	R\$ 7.158,00	R\$ 7.956,00	R\$ 8.736,00	R\$ 9.236,00
SPI		1,2891255	1,2391813	1,6663172	1,589773	1,5099762	1,4005288	1,3942371	1,3308734	1,2965909	1,0989701	0,9704329	0,9086421	0,908983094	1
CPI		0,5572566	0,8009912	1,3133947	1,4241796	1,4611906	1,4511843	1,4861753	1,4618145	1,4504244	1,2407641	1,1153152	1,0721765	1,064076236	1,184084019

* A planilha está disponível no arquivo excel anexado ao projeto;

Gráfico SPI X CPI



Sistema de Versionamento

Para este projeto fez-se uso do sistema de versionamento Git, O Git é um sistema de controle distribuído. Dentre as motivações pode-se citar:

- A abordagem peer to peer, diferentemente de outros sistemas de versionamento como o Subversion, que segue um modelo baseado em cliente-servidor;
- A possibilidade de ter infinidade de ramos de código independentes;
- A simplicidade para criação, exclusão e fusão dos ramos (não leva tanto tempo);
- O fato de o Git trabalhar com operações atômicas, ou seja, uma ação pode ter sucesso ou falhar (neste caso sem fazer alterações). Isso evita que uma operação com falha deixe o repositório instável.
- O fato de tudo ser armazenado dentro da pasta .git, diferentemente de outros sistemas que armazenam em pastas ocultas;
- O modelo de dados que ajuda a garantir integridade criptográfica de qualquer coisa no repositório usado pelo Git.
- O Git apresenta uma área de teste ou índice, permitindo aos desenvolvedores a formatação de commits e obtenção de feedbacks antes da aplicação dele.

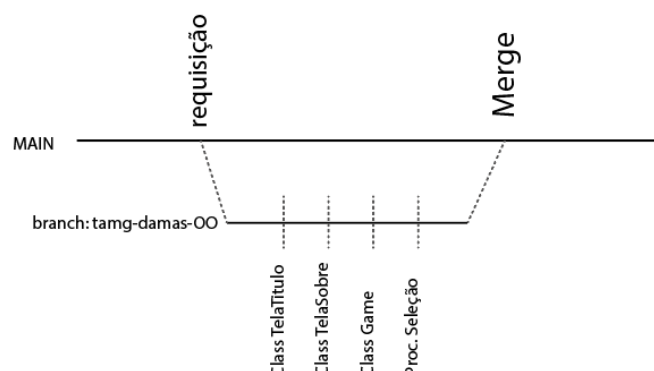
Para usar o Git, a ferramenta GitHub foi escolhida, há um repositório no GitLab testado também mas a dificuldade em setar algumas configurações (por parte da equipe) tornaram o uso desinteressante.

O repositório está disponível no link: <https://github.com/thiagofsf/tamg-damas>

Estratégia de Ramificação

A estratégia de ramificação utilizada é a **Organização por Requisições**, neste caso, o ramo principal (main) está sendo usado para integração das modificações enquanto ramos auxiliares podem ser usados para implementar funções, classes ou incorporar algo ao projeto.

A estratégia já foi usada durante o desenvolvimento, uma vez que a equipe escolheu mudar o formato do código, passando a usar orientação a objetos para encapsular os procedimentos e permitir que um arquivo seja desenvolvido sem que outro seja alterado. Dessa forma, um novo branch foi criado para fazer essa modificação, ao concluir a modificação e ver que o funcionamento da aplicação estava em ordem, esse conteúdo do branch foi mesclado ao main (modificações integradas).

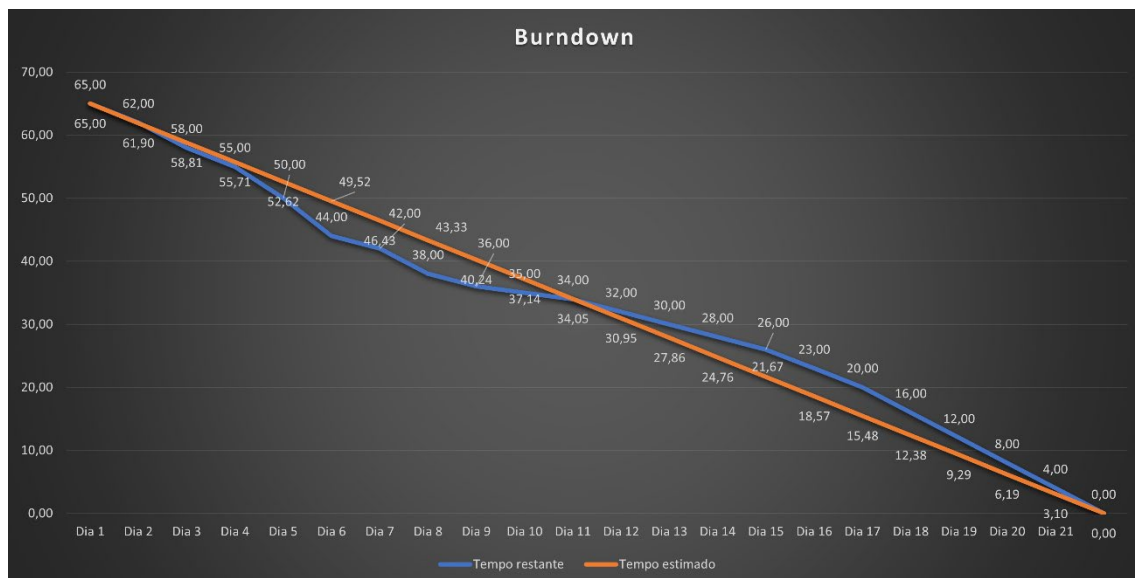


MONITORAMENTO E CONTROLE

Sprint Backlog - Lista das tarefas dos Sprints 002-004 (Semanas 14/10/2020 a 04/11/2020)

- Revisar Artes;
- Exportar novos *Sprites*;
- Procedimento Jogo;
- Procedimento Jogo - Desenhar tabuleiro;
- Procedimento Jogo - Desenhar Peças;
- Relatórios e Gráficos;
- Transformar Procedimentos em Classes;
- Criar Classe Da Tela Inicial;
- Criar Classe Da Tela Sobre;
- Transformar Procedimento Jogo em Classe Jogo;
- Upar alterações no *Branch tamg-damas-oo*;
- Carregar tabuleiro e peças na posição inicial de jogo;
- Definir método para detectar jogadas possíveis;
- Testar código;
- Preparar *Slides*;
- Definir método para seleção de células (Parcial);

Gráfico de Burndown



Análise do valor agregado:

- O projeto tem custo de R\$36.930,00
- A duração planejada é de 8 semanas
- Estamos na 6ª semana
- Cálculo dos custos até o momento:

Mão de obra: 65 (horas totais de todos os membros) * R\$50 (mão de obra) = R\$3.250,00

Softwares: R\$111 (PyCharm) + R\$30 (Office) + R\$224 (Adobe) + R\$400 (Internet para os 4 membros) = R\$765,00

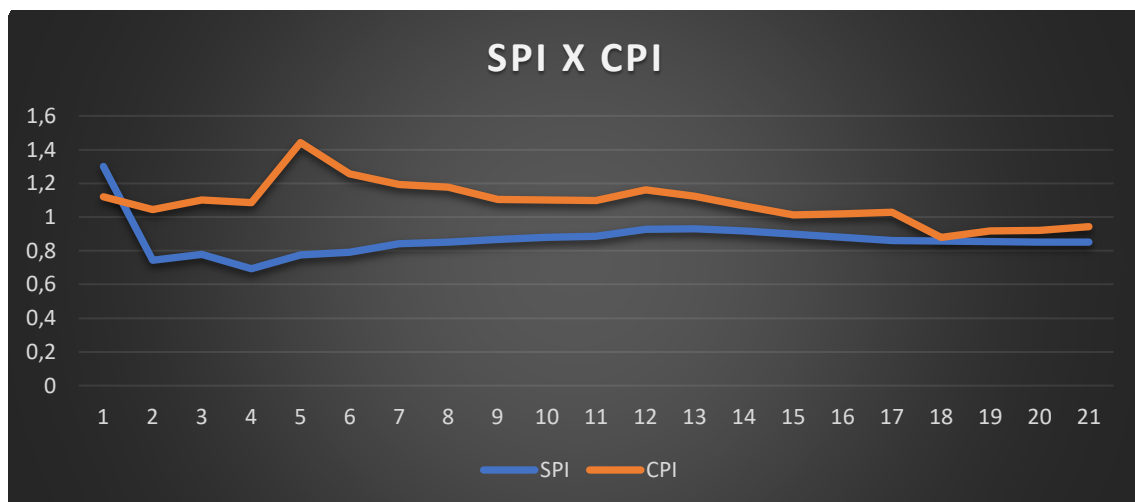
Energia Elétrica: 65 * R\$10 = R\$650,00

Custo total: R\$4.665,00

Atividades	Total de horas	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6	Dia 7	Dia 8	Dia 9	Dia 10	Dia 11	Dia 12	Dia 13	Dia 14	Dia 15	Dia 16	Dia 17	Dia 18	Dia 19	Dia 20	Dia 21
Atividade 1	1,00	1,00																				
Atividade 2	1,00		1,00																			
Atividade 3	1,00			1,00																		
Atividade 4	1,00				1,00																	
Atividade 5	1,00					1,00																
Atividade 6	1,00						1,00															
Atividade 7	1,00							1,00														
Atividade 8	1,00								1,00													
Atividade 9	1,00									1,00												
Atividade 10	1,00										1,00											
Atividade 11	1,00											1,00										
Atividade 12	1,00												1,00									
Atividade 13	1,00													1,00								
Atividade 14	1,00														1,00							
Atividade 15	1,00															1,00						
Atividade 16	1,00																1,00					
Atividade 17	1,00																	1,00				
Atividade 18	1,00																		1,00			
Atividade 19	1,00																			1,00		
Atividade 20	1,00																				1,00	
Atividade 21	1,00																					1,00
Soma de horas		1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
Valor planejado	R\$ 4.665,00																					
Valor planejado de conclusão		R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00	R\$ 250,00
Valor planejado de conclusão		50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%
Valor planejado de conclusão		50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%	50%
Custo real		1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
SPI		1,00	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943	0,74207943
CPI		1,00	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891	1,04517891

* A planilha está disponível no arquivo Excel anexado ao projeto;

Gráfico SPI X CPI



Mudança na Estratégia de Ramificação

Para correção da estratégia de ramificação dada a certa simplicidade do projeto e prover uma maior facilidade de integração da equipe, além de agilizar a atualização do repositório dada a quantidade de tarefas concorrentes, passa-se a realizar todas as alterações diretamente no ramo principal, esta estratégia (chamada de caótica) vai prover uma certa agilidade e funcionou melhor para a equipe, resolvendo de forma mais rápida os problemas e provendo atualizações mais rápidas também.

MONITORAMENTO E CONTROLE

Sprint Backlog - Lista das tarefas dos Sprints 005-007 (Semanas 04/11/2020 a 25/11/2020)

- Correção de *bug* ao selecionar células na extremidade (erro ao calcular possibilidades);
- Procedimento de movimento do jogador;
- Método de Análise de Jogadas Obrigatórias;
- Método de pulo (comer peça);
- Método de verificação de vitória/perda;
- Método de Pulo (pulo para trás);
- Método para verificar quando a peça vira dama;
- Definir jogadas obrigatórias (para damas);
- Jogadas possíveis para damas quando há peças para comer;
- Jogadas possíveis da dama (obrigatórias ou não);
- Pulo de damas;
- Encadeamento;
- Função que testa se existem jogadas possíveis (se não o jogador perde);
- Troca de turno;
- Método de jogada da IA;
- Testes finais e correção de *bugs*;
- Testes com usuários reais;
- Finalização de relatório e slides;
- Gravação e *commit* final;

Gráfico de Burndown



Testes

Considerando os atrasos sofridos durante o desenvolvimento do projeto, a equipe se viu com muito pouco tempo para a implementação de testes automatizados a tempo de garantir a funcionalidade dos componentes desenvolvidos. Para contornar essa dificuldade os testes foram realizados de forma manual. Seguem os testes realizados e como foram implementados:

Testes Visuais:

Primeiramente testamos visualmente, rodando mesmo a aplicação e visualizando o comportamento, ao detectar um erro (aplicação fecha devido a erro, comportamento inadequado mediante a uma ação), prosseguimos para análise das unidades.

Testes de unidade e integração:

Para testar o correto funcionamento de cada unidade (vamos ver Procedimentos como unidades nesse contexto) criou-se uma variável de teste da matriz de jogo. Esta matriz foi utilizada para realização dos testes, **forçando as situações** e garantindo que os procedimentos estavam retornando corretamente dadas as situações criadas. Nesse momento, essa matriz atua como uma espécie de “*stub*”, fornecendo dados de entrada para que as unidades fossem testadas.

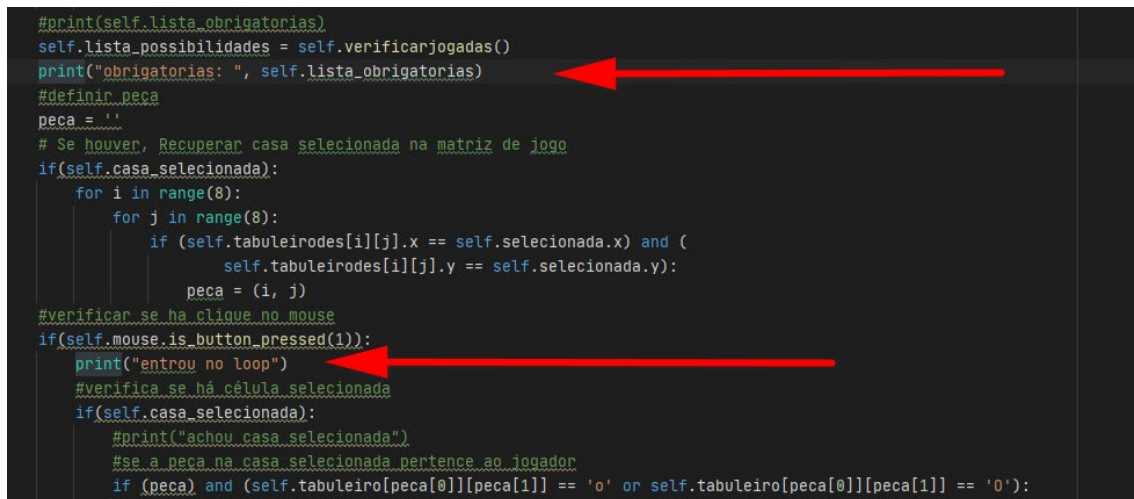
```
#tabuleiro de jogo
self.tabuleiro = [['x', '-', 'x', '-', 'x', '-', 'x', '-'],
                  ['- ', 'x', '-', 'x', '-', 'x', '-', 'x'],
                  ['x', '-', 'x', '-', 'x', '-', 'x', '-'],
                  ['- ', '-', '-', '-', '-', '-', '-', '-'],
                  ['- ', '-', '-', '-', '-', '-', '-', '-'],
                  ['- ', 'o', '-', 'o', '-', 'o', '-', 'o'],
                  ['o', '-', 'o', '-', 'o', '-', 'o', '-', '-'],
                  ['- ', 'o', '-', 'o', '-', 'o', '-', 'o']]

#uso em testes
self.tabuleirot = [['x', '-', 'x', '-', 'o', '-', 'x', '-'],
                   ['- ', 'x', '-', 'x', '-', '-', '-', 'x'],
                   ['x', '-', 'x', '-', '-', '-', '-', 'x'],
                   ['- ', '-', '-', '-', '-', '-', '-', '-'],
                   ['- ', '-', '-', '-', '-', '-', 'x', '-'],
                   ['- ', 'o', '-', '-', '-', '-', '-', 'o'],
                   ['o', '-', 'o', '-', 'x', '-', 'o', '-'],
                   ['- ', 'o', '-', 'o', '-', 'o', '-', 'o']]
```

Algumas situações que foram “forçadas” para testes:

- Peça na extremidade (testar se detectava e transformava em dama)
- Peças na posição de “comer” outra peça (verificar se as jogadas obrigatórias eram detectadas, nesses casos o jogador ou a IA apenas poderia selecionar dentro de uma lista de jogadas obrigatórias)
- Dama na posição de “comer” (verificar se as possibilidades de jogo estavam sendo carregadas corretamente)
- Encadeamento (criar situação em que várias peças deveriam ser “comidas” e verificar se os procedimentos detectavam e atuavam corretamente, impedindo por exemplo a virada do turno).
- Encadeamento com Damas
- Comer a última peça e verificar se o jogo acabava
- Deixar um dos jogadores sem jogadas possíveis e verificar se o jogo acabava

Para acompanhar as respostas e valores assumidos por variáveis durante o procedimento, além de indicar que caminhos o procedimento percorreu, foram usados prints no console.



```
#print(self.lista_obrigatorias)
self.lista_possibilidades = self.verificarjogadas()
print("obrigatorias: ", self.lista_obrigatorias)
#definir_pecas
peca = ''
# Se houver, Recuperar casa selecionada na matriz de jogo
if(self.casa_selecionada):
    for i in range(8):
        for j in range(8):
            if (self.tabuleirodes[i][j].x == self.selecionada.x) and (
                self.tabuleirodes[i][j].y == self.selecionada.y):
                peca = (i, j)
#verificar se ha clique no mouse
if(self.mouse.is_button_pressed(1)):
    print("entrou no loop")
    #verifica se há célula selecionada
    if(self.casa_selecionada):
        #print("achou casa selecionada")
        #se a peca na casa selecionada pertence ao jogador
        if (peca) and (self.tabuleiro[peca[0]][peca[1]] == 'o' or self.tabuleiro[peca[0]][peca[1]] == 'O'):
```

Usando esses simples recursos de teste, vários bugs e erros puderam ser identificados e corrigidos, garantindo o funcionamento desses procedimentos.

A maior parte dos testes foram executadas nos procedimentos de turnos (turno do jogador e turno de IA) e além de testarem o funcionamento individual dos procedimentos (**teste de unidade**), também testou o funcionamento em conjunto (**teste de integração**).

O que testamos?

1. Verificamos a lista retornada pelo procedimento que detectava jogadas obrigatórias, garantindo que jogadas onde peças poderiam ser “comidas” fossem listadas corretamente. Por meio deste teste, bugs nos loops do procedimento que verifica a jogada foram detectados e corrigidos.
2. Testamos a lista de jogadas possíveis, verificando se as jogadas possíveis estavam corretamente carregadas (a lista de possibilidades detecta para quais casas uma peça selecionada pode ser movida, acendendo-as no tabuleiro). Este teste detectou erro em um dos loops, onde uma das possibilidades não era carregada, esse erro foi corrigido.
3. Testamos o caminho da jogada no turno do jogador (verificando se entrou na condicional, se detectou uma célula selecionada, se esta era válida, se o clique estava dentro de uma célula possível para qual se poderia movimentar, verificando se haveria peça a ser “comida” e enquanto isso mostrando valores que foram considerados relevantes para a correção de possíveis erros, como: as coordenadas da peça selecionadas, as coordenadas para onde o jogador pretende movimenta-la, a lista que foi usada para verificar se depois dessa jogada haveria um encadeamento, ou seja, se essa peça poderia “comer” outra peça) e por meio deste testes erros nesses procedimentos foram corrigidos, apesar de só ter ocorrido quanto ao encadeamento que não havia sido detectado.
4. Testamos o caminho executado na jogada da IA, a partir do qual se pôde corrigir um erro na qual a IA jogava com a peça do jogador, isso acontecia aparentemente devido a algum tipo de

bug no framework onde o estado do mouse estava sendo guardado e recuperado no loop, por meio deste teste, implementamos uma verificação que ao notar uma célula do jogador selecionada no início do turno da IA desfazia esta seleção, removendo a seleção, permitindo então que o procedimento de IA fizesse a sua própria seleção.

Testes de aceitação:

Expomos possíveis usuários ao aplicativo, permitindo que jogassem para verificar se algum erro ou bug seria identificado na execução. Nenhum erro foi observado, os jogadores conseguiram navegar e usar a aplicação sem problemas.

*A maior parte dos prints usados nos testes foram mantidos no código comentados a fim de propósitos de consulta.

DIFICULDADES ENFRENTADAS

Em primeiro lugar o tempo, se tornou bastante complicado fazer o gerenciamento do tempo uma vez que todos os membros do grupo além de outras disciplinas, atualmente estão trabalhando. Com a pandemia e a mudança para o sistema home office, as demandas de trabalho aumentaram como um todo, seja no acadêmico quanto no profissional.

Em alguns momentos, instabilidade de rede e problemas de conexão atrapalharam, uma vez que essa era a principal ferramenta para comunicação da equipe.

Separar um tempo para se encontrar e ver o andamento do projeto todo o dia, já que praticamente todo dia algum detalhe do projeto era desenvolvido de alguma forma.

O uso do repositório, se tornou mais fácil compartilhar o código no drive, uma vez que uma atualização era automaticamente propagada a todos e a maior parte do tempo a equipe trabalhou em conjunto, conversando em vídeo chamada.

Todas essas dificuldades culminaram no atraso do projeto (aparente) que foi contornado conforme o possível para garantir a entrega de um produto funcional na data prevista.

PARTICIPAÇÃO

Todos os membros do grupo participaram de forma igual no desenvolvimento.