Relatório do primeiro seminário de ES2

Alunos:
Daniel Marcondes Bougleux Sodré
José Victor de Paiva e Silva
André Fernandes Gonçalves
Juan Müller Pereira Evangelista
Lucas Tavares Sousa

OBS: Alguns dos documentos gerados não cabem neste relatório, portanto, qualquer referência a eles será acompanhada de um link direto para o acesso. Além disso, os próprios documentos também estarão na entrega.

1 - Escopo do projeto:

- Justificativa do Projeto:
    - Uma Empresa de Jogos de Tabuleiro, sentindo as baixas nas vendas durante a Pandemia de Covid-19, decidiu contratar uma equipe para desenvolver versões digitais de jogos de tabuleiros.

- Finalidade do Projeto:
    - Criar uma versão do jogo War, originalmente em tabuleiro, para *desktop* e fiel às regras do jogo original, para que pessoas possam jogar umas com as outras sem a necessidade de se juntarem em um mesmo lugar.

- Objetivo(s) do Projeto:
    - Entregar o projeto completo segundo o planejamento montado, cumprindo o prazo estipulado e dentro do orçamento previsto.

- Descrição do Produto: via seções de Requisitos e Casos de Uso.

- Stakeholders do Projeto:
    - Clientes da Empresa de Jogos de Tabuleiro:
        - Interessado em ter uma versão digital do jogo de tabuleiro War.
        - O jogo digital deve ser fiel ao original de tabuleiro.
        - O jogo deve permitir interação entre os participantes durante a partida, como um chat de texto.

    - Dono da Empresa de Jogos de Tabuleiro:
        - Interessado no desenvolvimento da versão digital do jogo War para manter os clientes durante a época de pandemia.
        - Ter um sistema onde o cliente cadastrado na empresa possa acessar o jogo de casa.

- Entregas do Projeto:
    - Entrega 1: Toda a parte de planejamento do projeto completa.
    - Entrega 2: Menu inicial, criação da partida, e funções básicas para o game loop: *Setup* da partida; recebimento e movimentação de tropas; funções de ataque e troca de cartas.
    - Entrega 3: Jogo completo. Incluindo a implementação da IA e *multiplayer online*.

- Estimativas de Tempo e Custo: Estima-se terminar o projeto até 17/09/2021, com um custo total de R$4.217,51.

- Exclusões do Projeto: A princípio, o projeto engloba todas as funcionalidades para um jogo War *online*.

- Critérios de Aceitação:
    - Entrega 1: Uma descrição, documentação e arquitetura de jogo que condiga com o esperado dos stakeholders e que tenha potência de cumprir sua finalidade ao final do desenvolvimento.
    - Entrega 2: Uma versão simples com o mapa e algumas funções implementadas que respeitem as regras do jogo.
    - Entrega 3: Uma versão completa onde o jogador pode, ao menos, jogar localmente contra a IA ou outros jogadores.

- Premissas:
    - Todos os integrantes do grupo irão participar ativamente do desenvolvimento do jogo até o prazo estimado.
    - Será usado no projeto uma linguagem de programação em que os programadores já estão familiarizados.

- Restrições:
    - O jogo deve ser entregue até 17/09.
    - O jogo deve se basear em alguma versão de War.
    - Deverá haver implementação de uma IA que poderá jogar contra os jogadores.
    - O projeto não pode ultrapassar o orçamento de R$4.639,26.

2 - Gestão de Equipe:

Plano de reuniões:

- Duas reuniões semanais (sábados e domingos).
    - Sábado: Revisão do que foi feito na semana anterior.
    - Domingo: Planejamento do que será feito na próxima semana.

Foi criado um servidor no Discord para organizar a comunicação e informações pertinentes ao projeto, assim como um quadro no Trello para atribuir tarefas e marcar prazos de entrega.

O membro José Victor ficou responsável como gerente, organizador do servidor no Discord e no Trello, com o auxílio de Daniel como sub-gerente. A equipe como um todo é igualmente responsável pelas tarefas de planejamento e desenvolvimento do software.

3 - Arquitetura:

Linguagem: Python.

- Biblioteca(s): Pygame, PPlay.
- Backend As A Service: Firebase.

O jogo será executado no computador do usuário, com muitas funções relacionadas aos gráficos e interação do usuário com o jogo. As informações pertinentes às ações dos jogadores serão enviadas ao *Firebase.* O programa na máquina do jogador ficará escutando modificações(jogadas) em um banco de dados relacionado a partida. Outras informações guardadas no banco de dados, como perfil do jogador, também poderão ser buscadas no *Firebase*.

4 - Escopo do Produto:

4.1 - Requisitos funcionais:

4.1.1 - Requisitos de UX e UI:

A) Dentro da Partida:

- **RF00** O Sistema deve ser capaz de apresentar o mapa mundi com distinção entre cada território e continentes.
- **RF01** O sistema deve realizar um feedback visual:
    - RF01.1 Após cada batalha(ataques em territórios).
    - RF01.2 Ao distribuir e embaralhar cartas.
    - RF01.3 Ao distribuir fichas de exército.
    - RF01.4 Ao deslocar exércitos.
    - RF01.5 Ao começar e terminar turnos.
    - RF01.6 Ao começar e terminar a partida.
    - RF01.7 No final da partida, deve haver uma tela de comemoração se o jogador venceu, ou de derrota, caso tenha perdido.
- **RF02** O sistema deve ser capaz de sortear uma cor para cada participante do jogo.
- **RF03** O sistema deve ter um chat de texto durante o jogo que permita aos participantes se comunicarem durante a partida.

B) Fora da Partida:

- **RF04** O sistema deve permitir que o jogador mude as configurações de apresentação do jogo: volume do som, tela cheia ou modo janela.
- **RF05** O sistema deve ser capaz de apresentar uma interface distinta para o usuário que não tem permissão para jogar.
- **RF06** O sistema deve permitir que um jogador veja dados estatísticos como número de vitórias, derrotas e total de partidas jogadas para cada jogador que está na partida e na sala de Multiplayer.

4.1.2 - Requisitos Sobre as Regras do Jogo:

A) Itens que compõem o Jogo

- **RF07** O sistema deve possuir uma lista de 42 cartas que representam os territórios, cada uma com respectivos símbolos: círculo, triângulo e quadrado.
- **RF08** O sistema deve possuir 2 cartas coringas.
- **RF09** O sistema deve ter disponível 6 dados(3 vermelhos e 3 amarelos) para cada jogada de ataque e defesa.
- **RF10** O sistema deve possuir uma lista de 14 cartas que representam os objetivos para ganhar o jogo.
- **RF11** O sistema deve ser capaz de retirar do jogo os objetivos impossíveis de se concluir.
- **RF12** O sistema deve ser capaz de distribuir os exércitos iniciais.

B) Durante a Iniciação do Jogo

- **RF13** O sistema deve ser capaz de sortear quais territórios cada jogador irá receber.
- **RF14** O sistema deve ser capaz de distribuir uma carta de objetivo para cada jogador.

C) A cada rodada

- **RF15** O sistema deve dividir a ordem de jogadas em turnos, onde cada jogador pode jogar apenas no seu devido turno.
- **RF16** O sistema deve ser capaz de reconhecer quando um jogador atingiu seu objetivo.
    - **RF16.1** O sistema deve ser capaz de identificar o número de territórios conquistados por um jogador, e quais territórios são esses, bem como em quais continentes ficam.
    - **RF16.2** O sistema deve ter dados sobre qual jogador(ou Bot) eliminou qual outro jogador(ou Bot).
- **RF17** O sistema deve ser capaz de dar novas fichas de exército para os jogadores a cada início de rodada de acordo com sua situação corrente.
    - **RF17.1** A partir de territórios possuídos, divide a quantidade de territórios por 2 e a parte inteira é a quantidade de exércitos que o jogador irá receber.
    - **RF17.2** Se o jogador tiver algum continente inteiro conquistado, deverá receber o bônus de exército respectivo ao continente.
    - **RF17.3** O mínimo de exércitos a dar para um jogador é 3.
- **RF18** O sistema deve ser capaz de reconhecer que ataques podem ser feitos.
    - **RF18.1** Para atacar, o território atacante deve fazer fronteira direta com o território atacado.
    - **RF18.2** Para atacar, deve haver ao menos 2 exércitos no território atacante.
- **RF19** O jogador deve ser capaz de escolher se vai atacar, quantas vezes irá atacar e quem vai atacar.
- **RF20** O jogador deve ser capaz de escolher onde colocar suas fichas recebidas a cada começo de rodada.
    - **RF20.1** Os exércitos provenientes de bônus de continente conquistado devem ser colocados no continente em questão.

- **RF21** O sistema deve dar uma carta de território no final da rodada para o jogador que conquistou algum território.
- **RF22** O sistema deve reconhecer quando o monte de cartas acaba, e deve recolher e embaralhar as cartas à parte do jogo para formar um novo monte.

D) Troca de cartas

- **RF23** O jogador pode trocar três cartas com figuras iguais ou com figuras diferentes, durante sua jogada, por exércitos.
    a) **RF23.1** O valor da troca começa com 4 e vai aumentando 2 a 2. Esse incremento é por trocas na partida, e não por trocas por jogador.
    b) **RF23.2** O jogador não é obrigado a trocar cartas até ter 5 cartas.
    c) **RF23.3** Para cada carta trocada de um território que o jogador domine, ele recebe +2 exércitos extras que devem ser colocados obrigatoriamente no território da carta trocada.
    d) RF23.4 O sistema deve permitir ao jogador que possui uma carta coringa, escolher qual forma ele deseja para completar o requisito da troca de cartas.

E) A cada ataque

- **RF24** O sistema deve seguir todas as regras de ataque durante as jogadas de ataque.
    - **RF24.1** O sistema deve permitir que no máximo 3 exércitos do território atacante sejam utilizados na batalha (Um para cada dado de ataque).
    - **RF24.2** O jogador tem que poder escolher quais exércitos usar durante o ataque, bem como de qual território vai partir o ataque.
    - **RF24.3** O jogador não pode atacar com todo o exército que tem no território atacante, deve ficar ao menos um exército ocupante de fora do ataque.
    - **RF24.4** O sistema deve permitir que seja deslocado para o território conquistado apenas a quantidade de exércitos sobreviventes ao ataque.
    - **RF24.5** Os dados vermelhos são jogados pelo jogador atacante e o jogador defensor joga os dados amarelos.
    - **RF24.6** O atacante usa quantos dados ele tiver de exércitos atacantes, e o defensor usa quantos dados ele tiver de exércitos defensores.
    - **RF24.7** O maior dado do atacante é comparado com o maior dado de defesa, o menor dado do atacante é comparado com o menor dado do defensor. O dado do meio do atacante é comparado com o dado do meio do defensor.
    - **RF24.8** Se o atacante ganhar a comparação de dados, um exército do defensor sai do tabuleiro, se não, um exército atacante sai do tabuleiro.
- **RF25** Se um jogador não houver mais exércitos no tabuleiro, ele é eliminado do jogo.
- **RF26** O jogador que efetuar o ataque de eliminação em outro jogador, recebe todas as cartas do jogador eliminado.
    - **RF26.1** Se o jogador que recebeu cartas de um eliminado ficar com mais de 5 cartas, o sistema deverá embaralhar e virar as cartas para que o jogador sorteie quantas cartas forem necessárias para ficar com apenas 5.

F) Para deslocar exércitos

- **RF27** Depois que o jogador terminar seus ataques, ele poderá deslocar suas tropas.

    - **RF27.1** O sistema deve permitir que um mesmo exército se mova no máximo uma vez por turno.
- **RF28** O jogador deve ser capaz de escolher se vai deslocar seus exércitos, e de onde para onde irá deslocar seus exércitos.

4.1.3 - Requisitos de Single Player:

- **RF29** Jogar no modo offline, contra Bots apenas.
- **RF30** jogadores no modo convidado tem acesso apenas a este modo de jogo.

4.1.4 - Requisitos do Multiplayer:

- **RF31** Caso o número de jogadores seja menor do que a quantidade mínima requerida (4 jogadores), o sistema deve utilizar de bots para suprir a quantidade necessária de jogadores.
    - **RF31.1** Os bots devem funcionar utilizando de inteligência artificial, sem necessária ação de terceiros
    - **RF31.2** Os bots só podem realizar ações válidas (as mesma ações que outros jogadores poderia realizar)
    - **RF31.3** A inteligência artificial não pode considerar qualquer informação sigilosa de outro jogador (Ex: objetivo) para realizar sua ação
    - **RF31.4** A inteligência artificial precisa reconhecer seu objetivo, distribuição dos exércitos aliados e inimigos, cores dos demais jogadores para decidir sua próxima ação
- **RF32** O modo multiplayer deve ser online, com os jogadores se conectando uns aos outros através da internet, em uma sala.
    - **RF32.1** A sala de multiplayer tem um host, que pode convidar e expulsar jogadores da sala.
    - **RF32.2** O Host pode abrir ou trancar a sala para terceiros.
    - **RF32.3** A sala deve ter um chat por texto.
- **RF33** Deve haver um saguão geral, onde todas as salas abertas ficam expostas.
- **RF34** O jogador deve ser capaz de abandonar a partida na qual está participando.
- **RF35** O sistema deve ser capaz de substituir o jogador que abandonou a partida por uma IA.

4.1.5 - Requisitos de Autenticação:

- **RF36** O sistema deve permitir que o jogador seja autenticado com email e senha.
- **RF37** Usuários que não compraram o jogo podem apenas jogar modo Single Player.

4.1.6 - Requisitos de Histórico de Partidas:

- **RF38** O sistema deve possuir um histórico de partidas de cada jogador com pelo menos as partidas jogadas por ele durante os últimos 30 dias.
- **RF39** O histórico de partida deve conter quem venceu o jogo, se o jogador foi vencedor ou perdedor, qual a duração da partida e qual foi o objetivo do vencedor.

- **RF40** O sistema deve ser capaz de armazenar dados estatísticos sobre os jogadores como número de vitórias, derrotas e partidas jogadas.

4.2 - Requisitos não-funcionais

4.2.1. Disponibilidade:

- **DS01** O sistema online deve estar disponível em 98% do tempo nos dias úteis, das 18h às 23h, e 99% nos dias não úteis das 12h às 23h.

4.2.2. Portabilidade:

- **PR01** O jogo deve rodar no Windows.

4.2.3. Eficiência:

- **EF01** O jogo não deve ocupar mais de 1GB de RAM.

4.2.4. Segurança:

- **SE01** Alterações no banco de dados só devem ocorrer quando o usuário estiver logado.

4.2.5. Usabilidade:

- **US01** Um jogador que não conhece o jogo deve ser capaz de criar ou entrar em uma partida em até 5 minutos.

5 - Estimativas de esforço e custo:

5.1 - EAP

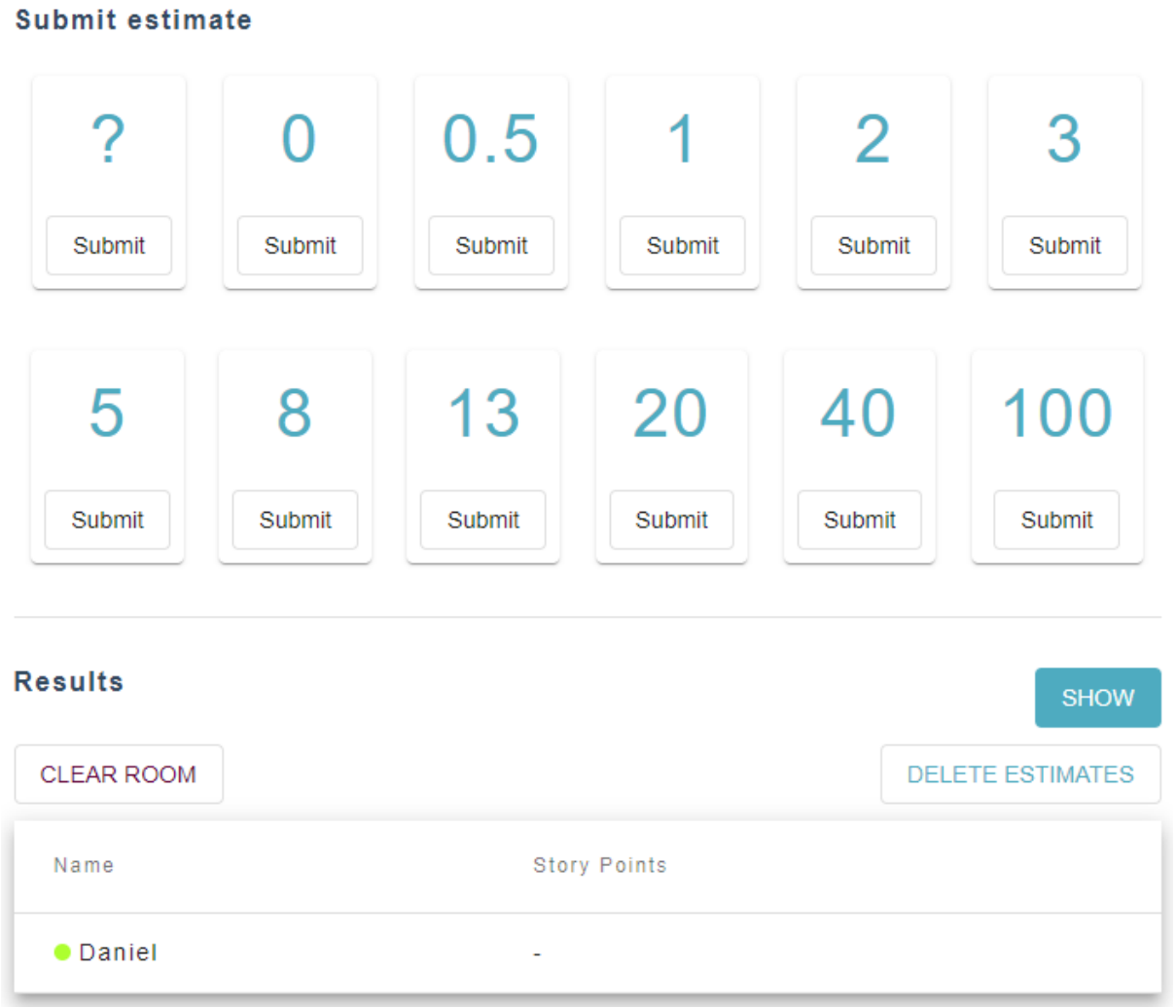
Para caracterizar o EAP foram gerados dois documentos: Um diagrama do EAP, e um dicionário do EAP.
O diagrama do EAP foi construído em cima de 3 fases do projeto: Planejamento, implementação e finalização e testes. Cada uma dessas fases é composta de subprodutos e pacotes de trabalho. Como não é possível colocar o diagrama neste relatório, segue o link do Diagrama do EAP.

Já o dicionário do EAP contém informações mais detalhadas relacionadas a cada pacote de trabalho(Descrição, Dependências, Estimativa de Esforço e Custo Estimado). Pelo mesmo motivo do diagrama, segue o link do Dicionário do EAP.

5.2 - Planning Poker

O planning poker foi feito para estimar a duração de cada atividade do Dicionário da EAP em homem-hora. Todos os membros do grupo participaram do planning poker. Para aplicar a técnica foi utilizado o site Scrum Poker-Online.

O planning poker foi feito da seguinte forma:



Escolhido um determinado pacote de trabalho, cada participante seleciona um número de homem-hora que acha que o pacote de trabalho irá requerer e clica em submit. Uma vez que cada participante tivesse submetido seu valor, o dono da sala mostrava os valores na tela. Feito isso, discutimos os valores discrepantes e chegamos a um acordo geral na estimativa. Esse processo foi repetido para cada pacote de trabalho e os resultados foram armazenados no documento do Dicionário do EAP.

5.3 - Orçamento

Baseado no salário médio de estagiários, decidimos que o custo/hora de cada desenvolver será R$16,67. Segundo as estimativas no Planning Poker, o total de esforço homem/hora no projeto será 253. Resultando assim em um custo total de R$4.217,51 (O custo de cada pacote de trabalho está no Dicionário do EAP). Para atingir uma margem de lucro de 10%, o orçamento fica em R$4.639,26.

Total de Esforço em Atividades:	253
Custo por Devs/Hora	R$ 16,67

Custo total das Atividades:	R$ 4.217,51
Orçamento:	R$ 4.639,26

6 - Cronograma:

O cronograma foi montado em uma planilha, para organizar os prazos de início e fim(mínimos e máximos) de cada atividade prevista na EAP, com margem para revisão. Foi montado também o Gráfico de Gantt, a ordem e dependência das atividades, o caminho crítico e as folgas das atividades fora do caminho crítico. A planilha de cronograma ficou muito grande, para ter uma visualização mais clara, segue o link do Cronograma.
Além disso, foi feito um diagrama do caminho crítico. segue o link do diagrama.
No cronograma, há também um caminho chamado “pseudo-crítico”, pois há uma grande carga de pacotes de atividades que poderia ser feita em paralelo, porém a equipe de desenvolvedores é pequena e logo sobraram atividades que só poderão ser realizadas após o caminho crítico, por conta disso, foi decidido entre a equipe que as folgas delas serão zero.

7 - Análise de Riscos:

7.1 Riscos:

- de Processo:
  - Cronograma incorreto
  - Ignorância sobre *multiplayer online*.
  - Conflito entre *API 's*.
  - Quantidade insuficiente de desenvolvedores.
- de Produto:
  - Manipulação da partida.
  - Problemas de desempenho.

7. 2 Riscos ordenados por exposição:

7.2.1 Risco: Manipulação da partida.

- Descrição: Possibilidade de manipulação da partida por parte do *host*.
- Probabilidade: Média (0,5).
- Impacto: Alto (0,9).
- Exposição: Alta (0,45).
- Plano(s) de contenção:
  - Tomar cuidados no desenvolvimento do código, para evitar que usuários mal intencionados possam corrompê-lo.
- Plano(s) de contingência:
  - Banir a conta do usuário identificado através de um botão de *report*.
- Monitoramento:
  - Duração: Durante todo o ciclo de vida do jogo, desde o início do projeto até o fim do suporte ao jogo.

        - Revisar possíveis falhas do código durante sua implementação.
        - Analisar jogadores reportados.

7.2.2 Risco: Cronograma incorreto.

- Descrição: Estipulação incorreta do tempo necessário para realização de pelo menos uma das iterações.
- Probabilidade: Média (0,5).
- Impacto: Alto (0,7). Possibilidade de não entregar todas as funcionalidades prometidas da iteração.
- Exposição: Alta (0,35).
- Plano(s) de contenção:
    - No caso de uma iteração levar menos que o esperado, adiantar o início da próxima iteração.
    - Começar cada iteração o mais cedo possível, para identificar possíveis problemas o quanto antes.
    - Planejamento bem detalhado de todos os requisitos de cada funcionalidade, assim como o passo a passo de sua execução.
- Plano(s) de contingência:
    - Utilizar de mais horário que o antecipado para concluir corretamente a iteração.
- Monitoramento:
    - Duração: Durante todo o planejamento e implementação do jogo.
    - Comunicação ativa entre os desenvolvedores para possíveis suspeitas de atraso.

7.2.3 Risco: Ignorância sobre *multiplayer online*.

- Descrição: Dificuldade na implementação do *multiplayer online* com Firebase.
- Probabilidade: Média (0,7).
- Impacto: Baixo (0,3). Entrega do projeto sem o *multiplayer online*, apenas o *multiplayer offline*.
- Exposição: Média (0,21).
- Plano(s) de contenção:
    - Aprender o mais cedo possível sobre implementação do *multiplayer online*, antes mesmo de sua implementação ser definida no cronograma.
- Plano(s) de contingência:
    - Não entregar o programa com *multiplayer online*, apenas o *multiplayer offline*.
- Monitoramento:
    - Duração: Durante a iteração reservada para implementação do *multiplayer online*.
    - Testes frequentes durante a respectiva iteração para avaliar o funcionamento correto do *multiplayer online*.

7.2.4 Risco: Quantidade insuficiente de desenvolvedores.

- Descrição: Por qualquer motivo, pelo menos um dos desenvolvedores não conseguiu produzir o que foi planejado.
- Probabilidade: Baixa (0,2).
- Impacto: Alto (0,8).
- Exposição: Média (0,16).
- Plano(s) de contenção:
  - Estabelecimento de uma boa rede de comunicação entre os desenvolvedores.
- Plano(s) de contingência:
  - Distribuição do trabalho não realizado entre os outros desenvolvedores.
- Monitoramento:
  - Duração: Durante toda a implementação do jogo.
  - Discussão ativa entre os desenvolvedores para possíveis problemas individuais.

7.2.5 Risco: Conflito entre *API 's*.

- Descrição: Mais especificamente, durante a implementação do chat, há a possibilidade de conflitos entre PPlay e tkinter.
- Probabilidade: Média (0,4).
- Impacto: Baixo (0,2).
- Exposição: Baixa (0,08).
- Plano(s) de contenção:
  - Estudar as limitações do PPlay.
- Plano(s) de contingência:
  - Não implementação do chat.
- Monitoramento:
  - Duração: Durante a iteração reservada para implementação do *multiplayer online*.
  - Testes frequentes durante a respectiva iteração para avaliar o funcionamento correto do chat.

7.2.6 Risco: Problemas de desempenho.

- Descrição: Possibilidade de baixo desempenho do produto final devido a implementação de algumas bibliotecas utilizadas.
- Probabilidade: Média (0,4).
- Impacto: Médio (0,4).
- Exposição: Baixa (0,08).
- Plano(s) de contenção:
  - Melhor compreender as limitações das bibliotecas utilizadas.
- Plano(s) de contingência:
  - Atualização com implementação de funções análogas, mas com foco no desempenho.
- Monitoramento:
  - Duração: Durante toda a implementação do jogo.
  - Realização de testes para averiguar o desempenho.

8 - Sprint

8.1 - Detalhamento de uma Sprint(primeira Sprint):

8.1.1- Lista de Atividades:

O principal objetivo dessa Sprint foi terminar toda a documentação/tarefas requeridas para a primeira entrega, com exceção das atividades referente a apresentação do trabalho, vídeo e o próprio relatório. Essas atividades são: Lista de Requisitos Funcionais e não funcionais, Casos de Uso, Análise de Risco, Descrição do Escopo do Projeto, Diagrama de Classes, EAP, Monitoramento Controle, Orçamento, Cronograma, Planning Poker.

8.1.2 - Atividades Concluídas:

Embora tenha sido necessário mais recursos para terminar essas atividades do que inicialmente planejado, como será mostrado no monitoramento e controle, conseguimos terminar todas as atividades, as quais foram planejadas para Sprint.

8.1.3 - Atividades Pendentes:

Não tivemos nenhuma atividade pendente, já que todas elas foram concluídas. O que demonstra que nossa Sprint alcançou os resultados esperados.

8.2 - Duração de cada interação

Nosso grupo decidiu que cada interação terá uma duração de duas semanas, ou seja, 14 dias.

8.3 - Entregável de cada interação

O que nosso grupo planejou para cada interação foi:
1ª interação: Lista de Requisitos funcionais e não funcionais, Casos de Uso, Análise de Riscos, EAP, Planning Poker, Orçamento, Diagrama de Classes, Cronograma, Monitoramento e Controle e Descrição do escopo do projeto.

2ª interação: Recebimento de Tropas, Login, Combate, Histórico de Partidas, Organização e funcionalidades do mapa, Componentes do Menu e Componentes do Mapa, Testes Unitários.

3ª interação: Estratégia de conquistar 24 territórios, Estratégia de conquistar Ásia e AMS||AF, Estratégia de conquistar AMN + AF||OC, Estratégia de conquistar os continentes EU + AMS||OC + 1 Qualquer, Estratégia de conquistar 18 territórios com no mínimo 2 tropas em cada ,Estratégia de Destruir o exército da cor X, Perfil do Jogador, Chat, Configurações, Criar Partida, Troca de Cartas, Movimentação de Tropas, Testes Unitários.

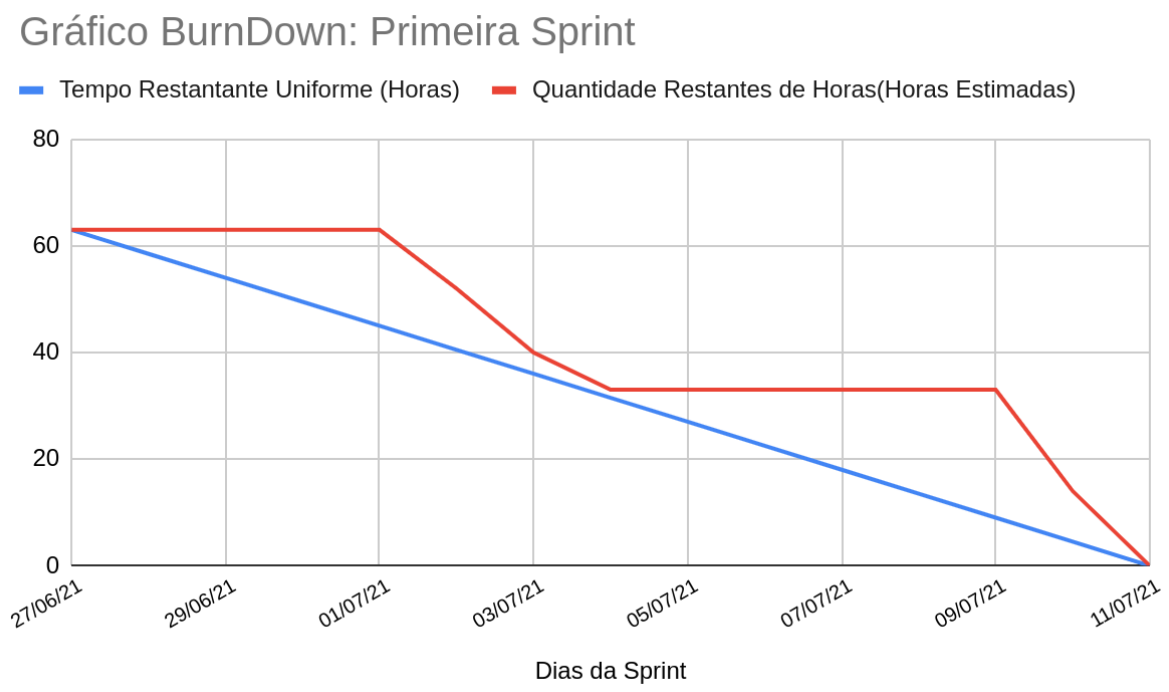
4ª interação: Revisão e Acabamentos, Testes de Aceitação, Testes de Sistema, Testes de integração, Testes Unitários.

9 - Monitoramento e Controle:

9.1 - Gráfico de BurnDown:

O gráfico de burndown, é um meio que temos como avaliar um andamento de um sprint. Para isso, o gráfico tem duas linhas, a primeira (no nosso caso em azul), mostra uma diminuição linear quantidade de tempo restante em horas planejadas para a sprint, a segunda linha demonstra a quantidade de horas restante reais, ou seja, a soma das horas planejadas das atividades não terminadas.

Para fazer esse gráfico foi preciso calcular, primeiramente, a quantidade total de horas e de dias da sprint para fazer a redução linear e, além disso, conforme terminavam as atividades colocamos essa horas em uma planilha para no término da sprint montarmos o gráfico da quantidade restante de horas reais



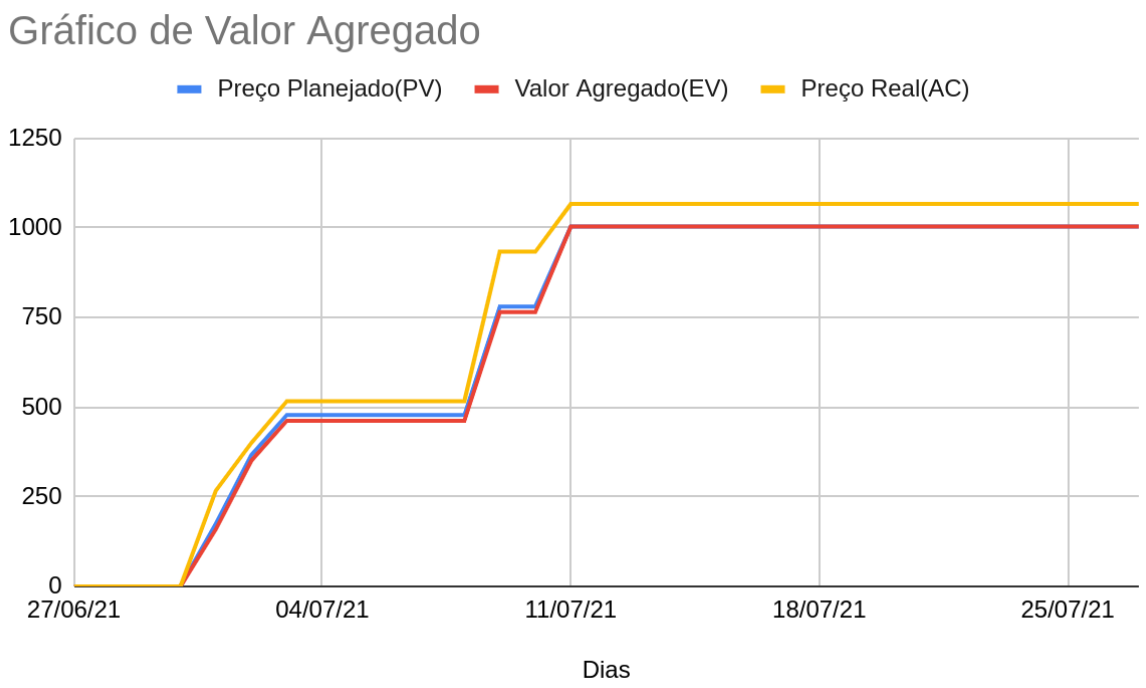
Segue o link da planilha: BurnDown

9.2 - Gráfico de Valor Agregado:

O gráfico de Valor Agregado, possui como principal objetivo avaliar o projeto em questão nos gastos do projeto. Em primeiro lugar calculamos a porcentagem planejada de conclusão do projeto com o passar do tempo e por sua vez a utilizamos para calcular os gastos planejados do projeto em função do tempo, esse dado é chamado de Preço Planejado(PV).

Quando o projeto começa a ser desenvolvido, ao terminar uma tarefa adicionamos a porcentagem de conclusão do projeto referente aquela tarefa em uma planilha com o dia do seu término, e da mesma forma calculamos o gasto dessas tarefas por dia, esse valor é chamado de Valor Agregado(EV). Por fim, repetimos o processo para calcular o Valor Agregado(EV), mas utilizando o tempo realmente gasto no projeto, dessa forma calculamos o Preço Real(AC)

Obs. Nesta entrega os valores de Preço Planejado e Valor Agregado, são basicamente os mesmo devido ao fato que o cronograma, sendo uma das últimas tarefas realizadas foi muito baseado nas tarefas já concluídas. Esse erro não deve persistir para as demais entregas



Valores Atuais de:
PV: R$ 1.003,60
EV: R$ 1.003,60
AC: R$ 1.066,88

Conclusões matemáticas:
SPI = $\frac{ev}{pv} = \frac{1003,6}{1003,6}$ = 1, já que valor é igual a 1, significa que a gente está no cronograma

CPI = $\frac{ev}{pv} = \frac{1003,6}{1066,88}$ = 0,94, já que esse valor é menor do que 1, isso quer dizer que estamos gastando mais do que foi planejado.

Podemos perceber que mesmo o projeto custando mais que o planejado, ou seja, estamos gastando mais recursos humanos para realizar as tarefas do que o tempo estipulado inicialmente, ainda estamos no prazo. Esse fato aconteceu porque a equipe como um todo se esforçou para recuperar o atraso que equívocos do planejamento poderiam proporcionar. Por outro lado, a diferença do preço real com o preço planejado não é muito gritante, o que demonstra que o planejamento está próximo da realidade. Por fim, em respeito ao valor de SPI, ser exatamente 1, isso é devido a gente ser capaz de concluir todas e apenas a tarefas que a gente estipulou serem essenciais para a primeira entrega do trabalho
Segue o link da planilha: Valor Agregado

10 - Versão Parcial do Produto:

Antes de iniciar a programação em si, a equipe decidiu detalhar melhor a arquitetura da implementação através da geração de casos de uso e um diagrama de classes. O diagrama de classes e casos de uso ficaram muito grandes, para uma melhor visualização do resultado, acesse os links abaixo:

- Casos de Uso
- Diagrama de Classes