

ISPGAYA

instituto superior politécnico

Licenciatura em Engenharia Informática
Projeto de Engenharia Informática em Contexto Empresarial.
3.º ano

Paulo Ricardo Figueiredo Pinto Monteiro Guimarães

GAME DEVELOPER
DESENVOLVIMENTO DO JOGO
STATE OF AFFAIRS

Relatório de estágio orientado pelo Professor Doutor Sérgio Francisco Sargo
Ferreira Lopes e apresentada à Escola Superior de Ciência e Tecnologia

Junho de 2025

Dedicatória

A toda a minha família dedico o presente trabalho, sem eles nada disto teria sido possível, apoiaram e acreditaram em mim, mesmo nos momentos mais delicados. O vosso amor e incentivo foram primordiais para alcançar esta conquista.

É justamente a possibilidade de realizar um sonho que torna a vida interessante.

(Coelho, 1990)

Agradecimentos

A realização do presente relatório de estágio define a conclusão daquele que foi um percurso muito importante na minha formação enquanto aluno de Engenharia Informática, e não teria sido possível sem a ajuda de algumas pessoas, a quem deixo um muito obrigado.

Quero expressar o meu profundo agradecimento ao Professor Doutor Sérgio Francisco Sargo Ferreira Lopes pela orientação e apoio prestado ao longo deste processo.

Agradecer também efusivamente ao Eng.º Ricardo Nunes, pela receção, orientação e pela oportunidade de integrar num projeto tão ambicioso e desafiante.

Um especial obrigado à *Top Sigma, Lda*, pela oportunidade de realizar este estágio num ambiente profissional, experiente no ramo, estimulante e acima de tudo, muito enriquecedor, que pude aplicar e desenvolver os meus conhecimentos em programação.

Quero também agradecer a todos os professores da Licenciatura em Engenharia Informática do ISPGAYA, que ao longo dos anos contribuíram para a minha formação.

Por fim, não posso deixar de agradecer à minha namorada, família e amigos pelo apoio, motivação e incentivo constantes, essenciais para a conclusão desta importante etapa da minha vida.

A todos, o meu muito obrigado.

Resumo

Este relatório descreve todo o progresso durante o estágio curricular realizado na empresa *Top Sigma, Lda*, para a unidade curricular Projeto de Engenharia Informática em Contexto Empresarial. O estágio teve como objetivo central oferecer uma vasta panóplia de experiências ao nível de desenvolvimento de jogos com ênfase na utilização do motor de desenvolvimento *Unity* e a linguagem de programação *C#*.

Todo o projeto consistiu essencialmente na elaboração de um jogo, em particular um *Vertical Slice*, de simulação política, ambientado no contexto da União Soviética pós-Stalin ainda em rescaldo da Segunda Guerra Mundial, entre 1953 e 1956. Esta *Vertical Slice* representa uma secção completa e jogável que demonstra o ciclo de decisão política, a gestão de sub-fações internas pertencentes ao partido Comunista e as dinâmicas estratégicas num regime centralizado durante os primeiros anos de uma nova era liderada por uma das 3 ideologias mais fortes do momento, com visões Reformistas, Militarizadas ou Equilibradas.

As atividades principais incluíram a implementação de sistemas de aprovação de políticas, lógica de comportamento de sub-fações, eventos de crise dinâmica e *feedback* em tempo real ao jogador, utilizando estruturas orientadas a objetos, manipulação de dados em *JSON (JavaScript Object Notation)* e integração de interfaces de utilizador (*UI/UX - User Interface / User Experience*). Para o controlo de versões foi utilizado *Git*.

No presente relatório encontram-se descritos o local de estágio, as tecnologias utilizadas, bem como as principais atividades e projetos desenvolvidos. No final, é apresentada uma reflexão crítica sobre as aprendizagens adquiridas e as dificuldades encontradas ao longo do processo de estágio.

Abstract

This report describes the entire progress of the curricular internship carried out at the company Top Sigma, Lda, as part of the course unit Software Engineering Project in a Business Context. The internship's main objective was to provide a broad range of practical experiences in game development, with a focus on using the Unity game engine and the C# programming language.

The project consisted primarily of the creation of a Vertical Slice for a political simulation game, set in the post-Stalin Soviet Union, still recovering from World War II, between 1953 and 1956. This Vertical Slice represents a complete and playable section of the game, demonstrating the political decision-making cycle, internal faction management, and strategic dynamics in a centralized regime during the early years of a new era led by one of three dominant ideologies: Reformist, Militarized, or Balanced.

The main activities included the implementation of policy approval systems, faction behavior logic, dynamic crisis events, and real-time player feedback - using object-oriented structures, JSON data manipulation, and integration of user interfaces (UI/UX). Version control throughout the development process was managed using Git.

This report presents the internship context, technologies used, and the main activities and systems developed. It concludes with a critical reflection on the skills acquired and challenges encountered during the internship experience.

Lista de abreviaturas e siglas

- CPU Central Processing Unit (Unidade Central de Processamento).
- DDR Double Data Rate (Taxa Dupla de Transferência)
- E2E End-to-End (Teste de Ponta a Ponta).
- EUA Estados Unidos da América.
- GB Gigabyte.
- GDP Gross Domestic Product (Produto Interno Bruto).
- Gol Group of Interest (Grupo de Interesse).
- GPU Graphics Processing Unit (Unidade de Processamento Gráfico).
- IDE Integrated Development Environment (Ambiente de Desenvolvimento Integrado).
- JSON JavaScript Object Notation.
- KGB Komitet Gosudarstvennoy Bezopasnosti (Comité de Segurança do Estado).
- MGB Ministerstvo Gosudarstvennoy Bezopasnosti (Ministério da Segurança do Estado).
- MVD Ministry of Internal Affairs (Ministério dos Assuntos Internos).
- NATO North Atlantic Treaty Organization (Organização do Tratado do Atlântico Norte).
- QA *Quality Assurance* (Garantia de Qualidade)
- RAM Random Access Memory (Memória de Acesso Aleatório).
- SSD Solid State Drive (Disco de Estado Sólido).
- TI Tecnologias de Informação.
- UI/UX User Interface/User Experience (Interface do Utilizador/Experiência do Utilizador).

Índice

Dedicatória	iii
Agradecimentos.....	iv
Resumo.....	v
Abstract.....	vi
Lista de abreviaturas e siglas	vii
Introdução.....	12
1 Caracterização da Entidade de Estágio	13
1.1 Descrição da Empresa	13
1.2 Departamento e Equipa onde decorreu o Estágio.....	13
2 Projeto Desenvolvido	14
2.1 Descrição Geral do Projeto	14
2.2 Contexto Narrativo	14
2.3 Projetos Similares e Soluções Existentes.....	15
3 Bases Teóricas	16
3.1 Fundamentos de Jogos de Simulação Política	16
3.2 <i>Unity</i> e Desenvolvimento de Jogos.....	17
4 Requisitos e Arquitetura do Sistema.....	18
4.1 Requisitos Funcionais.....	18
4.2 Requisitos Não Funcionais	19
4.3 Arquitetura do Sistema	21
4.3.1 <i>Diagrama de Casos de Utilização</i>	23
4.3.2 <i>Fluxogramas</i>	27
5 Atividades e Trabalho Realizado.....	29
5.1 Organização do Trabalho.....	29
5.2 Desenvolvimento de Funcionalidades.....	31
5.2.1 <i>Unity Essentials</i>	32
5.2.2 <i>Develop Policy Proposal System</i>	33
5.2.3 <i>Develop Political Consequences of Legislative Failure</i>	35
5.2.4 <i>Create Placeholder UI to Consult Historical Policy Data</i>	38
5.2.5 <i>Define Industry Sector Activities and Resource Flow</i>	40
5.2.6 <i>Develop Debt and Economic Crisis Mechanics</i>	41

5.2.7	<i>Implement Legislative Sessions per Turn</i>	43
5.2.8	<i>Implement EventLog Formatting System</i>	45
5.2.9	<i>Display personalized “Welcome to Office” onboarding note</i>	49
5.3	Documentação Técnica dos Sistemas Principais	50
5.3.1	<i>Estrutura Central de Países</i>	50
5.3.2	<i>Sistema de fações</i>	51
5.3.3	<i>Sistema de Políticas</i>	52
5.3.4	<i>Sistema de Situações</i>	54
5.3.5	<i>Sistema Modular de Efeitos</i>	55
5.3.6	<i>Sistema de Classes Sociais (Social Strata)</i>	59
5.4	Refatoração e Otimização de Código.....	60
5.4.1	<i>Refactor GoverningBody Structure</i>	61
5.4.2	<i>Refactor Effect Structure</i>	61
5.4.3	<i>Implement Faction System Architecture</i>	62
5.5	Testes e Validação.....	63
5.5.1	<i>Implementação de Testes End-to-End (E2E)</i>	63
5.5.2	<i>Teste E2E: Caminho Reformista</i>	63
5.5.3	<i>Teste E2E: Caminho Conservador</i>	65
5.5.4	<i>Teste E2E: Caminho Equilibrado</i>	68
6	Cronograma	71
6.1	Planeamento Inicial.....	71
6.2	Execução Real.....	73
6.3	Análise de Desvios e Justificações	75
7	Meios Utilizados.....	75
7.1	Meios Humanos (equipa de trabalho)	75
7.2	Meios Materiais (<i>hardware, software</i> , ferramentas)	76
8	Problemas e Decisões Tomadas.....	76
8.1	Problemas Encontrados	76
8.2	Decisões e Justificações	77
9	Análise de Resultados	77
9.1	Avaliação dos Objetivos Atingidos.....	77
9.2	Impacto Pessoal e Profissional	78
	Conclusões.....	79
	Referências bibliográficas	81

Glossário.....	84
----------------	----

Índice de Figuras

Figura 1. <i>Diagrama UML Casos de Utilização Iniciar jogo</i>	24
Figura 2. <i>Diagrama UML Casos de Uso Office Scene</i>	25
Figura 3. <i>Diagrama UML Casos de Uso Politburo Scene</i>	26
Figura 4. <i>Diagrama de Fluxo do menu inicial</i>	27
Figura 5. <i>Diagrama de Fluxo do OfficeScene</i>	28
Figura 6. <i>Diagrama de Fluxo do Politburo</i>	29
Figura 7. <i>Logo grupo Atlassian</i>	30
Figura 8. <i>Menu principais</i>	31
Figura 9. <i>Novo jogo URSS</i>	32
Figura 10. <i>Método Start e ForceUpdatePolicyDropdown</i>	33
Figura 11. <i>Método ShowPolicyDetails</i>	34
Figura 12. <i>Detalhes das políticas</i>	35
Figura 13. <i>Estabilidade política</i>	35
Figura 14. <i>Contexto da estabilidade</i>	36
Figura 15. <i>Ativação de efeitos</i>	36
Figura 16. <i>Exemplo de situação</i>	37
Figura 17. <i>Ativação da situação</i>	37
Figura 18. <i>Método Initialize</i>	38
Figura 19. <i>Método RefreshArchiveUI</i>	39
Figura 20. <i>Arquivo de políticas votadas</i>	39
Figura 21. <i>Setor Industrial</i>	41
Figura 22. <i>Cálculo da dívida nacional</i>	41
Figura 23. <i>Verificação do estado da dívida</i>	42
Figura 24. <i>Dossiê do office</i>	42
Figura 25. <i>Exemplo de Política</i>	43
Figura 26. <i>Método EvaluateSimulatedVoteOutcome</i>	44
Figura 27. <i>Método HandleSimulatedVoteOutcome</i>	45
Figura 28. <i>Enum dos tipos de EventLog</i>	46
Figura 29. <i>Templates de EventLog</i>	46
Figura 30. <i>Carregamento dos Logs</i>	47
Figura 31. <i>Filtros de Logs</i>	48
Figura 32. <i>Logs no Politburo</i>	48
Figura 33. <i>Exemplo de estrutura de dados</i>	49
Figura 34. <i>Escolha ideológica</i>	50
Figura 35. <i>Fluxo de fações</i>	52
Figura 36. <i>Fluxo de sessão legislativa</i>	53
Figura 37. <i>Fluxo de políticas</i>	53
Figura 38. <i>Fluxo de situações</i>	54
Figura 39. <i>Lógica de ativação de situações</i>	55
Figura 40. <i>Exemplo de efeitos</i>	56
Figura 41. <i>Aplicação de efeitos em políticas</i>	57

Figura 42. <i>Aplicação de efeitos em situações</i>	58
Figura 43. <i>Aplicação de efeitos graduais</i>	58
Figura 44. <i>Fluxo dos estados sociais</i>	60
Figura 45. <i>Aplicação de efeitos em estados sociais</i>	60
Figura 46. <i>Caminho Reformista</i>	64
Figura 47. <i>Caminho Conservador</i>	67
Figura 48. <i>Caminho Equilibrado</i>	70
Figura 49. <i>Gráfico de Gantt</i>	73
Figura 50. <i>Cronograma</i>	74

Índice de tabelas

Tabela 1. <i>Caso de teste Caminho Reformista</i>	65
Tabela 2. <i>Caso de teste Caminho Conservador</i>	67
Tabela 3. <i>Caso de teste Caminho Equilibrado</i>	70
Tabela 4. <i>Cronograma de estágio</i>	71

Introdução

Este relatório é elaborado como parte da unidade curricular Projeto de Engenharia Informática em Contexto Empresarial, da Licenciatura em Engenharia Informática. Este apresenta como foco central todas as tarefas realizadas no decorrer do estágio, ocorrido entre dia 11 de fevereiro a dia 20 de junho de 2025, coincidindo parcialmente com o início e final do semestre. O trabalho espelha-se na aplicação prática de todos os conhecimentos adquiridos no decorrer do meu percurso académico, focado essencialmente em linguagem C# para a programação e *Unity* como motor de desenvolvimento.

No decorrer do estágio foi proposta a elaboração de um *Vertical Slice* de um jogo de simulação política que consiste numa versão jogável, ainda em desenvolvimento, denominado de *State of Affairs*, situado na antiga União Soviética num período de tensão com a morte de *Josef Stalin* e ainda em recuperação do final da Segunda Guerra Mundial. O jogo teria de contar com ciclos completos escolhas políticas, a gestão de sub-fações internas e a reação das dinâmicas sociais e geopolíticas da época. Além disso, a aplicação de boas práticas de programação orientada a objetos, otimização de desempenho, integração de interfaces de utilizador (UI/UX) e produção de documentação técnica.

O trabalho foi inteiramente desenvolvido sob uma metodologia iterativa, com base em *sprints* quinzenais de modo que fossem abordadas diferentes áreas do jogo, assim como políticas, simulação económica, votação, etc. O início do estágio é marcado com um pequeno curso de *Unity*, de modo que uma base sólida fosse assegurada para o restante desenvolvimento do jogo.

O presente documento encontra-se organizado para acompanhar todo o percurso do estágio. Apresenta inicialmente o enquadramento do projeto e os respetivos objetivos, seguido de uma descrição da entidade acolhedora e das principais características do jogo *State of Affairs*. Seguem-se as tecnologias utilizadas, o desenvolvimento dos vários módulos e funcionalidades implementadas ao longo dos *sprints*, bem como os testes realizados. O relatório termina com uma reflexão crítica sobre o trabalho desenvolvido e as aprendizagens obtidas.

1 Caracterização da Entidade de Estágio

1.1 Descrição da Empresa

A *Top Sigma, Lda* (*Top Sigma*, 2025) é uma empresa de tecnologia conhecida por oferecer soluções inovadoras em áreas como o desenvolvimento de *software*, gerir infraestruturas e consultoria em *TI*.

Entre os principais serviços oferecidos pela *Top Sigma* destacam-se:

- *Managed Services Provider* - Gestão de infraestruturas de *TI* (Tecnologias da Informação), cibersegurança, ambientes de *cloud*, *Data Warehouse* e monitorização de sistemas, com suporte proativo e resposta a incidentes.
- Consultoria em Tecnologias de Informação - Apoio especializado nas áreas de *DevOps*, *Cloud Engineering*, *Site Reliability Engineering*, *Compliance*, Cibersegurança e Dados e Inteligência Artificial.

Além disso, também são prestados serviços ao nível do desenvolvimento de videojogos, nomeadamente pela *Top Sigma Studios*. Este departamento, do qual fiz parte no decorrer do estágio, foca-se na criação de narrativas e estrategicamente profundas, como no *State Of Affairs*.

A abordagem da *Top Sigma* permite aos clientes concentrarem-se somente no seu core-business enquanto confiam a gestão tecnológica a uma equipa altamente qualificada.

1.2 Departamento e Equipa onde decorreu o Estágio.

O estágio foi desenvolvido na divisão de Desenvolvimento de Jogos, na *Top Sigma Studios*, uma área dedicada à criação de experiências digitais imersivas. Esta área de trabalho explora o uso do motor de desenvolvimento *Unity* para desenvolver jogos e outras aplicações.

A equipa de cinco membros na qual o estágio decorreu é constituída por profissionais com experiência em programação e criação de jogos. Sob a supervisão do Engenheiro Ricardo Nunes, e em modo *home office*, o estágio proporcionou um ambiente colaborativo e orientado para resultados, permitindo o desenvolvimento de competências técnicas em contextos reais de produção.

O trabalho desenvolvido seguiu os princípios da metodologia ágil *Scrum*, com organização em sprints quinzenais, planeamento e entrega contínua de funcionalidades.

2 Projeto Desenvolvido

2.1 Descrição Geral do Projeto

O projeto desenvolvido durante o período de estágio consistiu na implementação de uma *Vertical Slice* de um jogo de simulação política, com cerca de 156 turnos, dando 60 minutos de jogabilidade. Esta *Vertical Slice* surge já sobre uma base pré-existente e envolveu o desenvolvimento dos principais módulos da jogabilidade. Módulos esses que seriam as propostas de políticas, debates e votação, reações internas por parte das sub-fações e grupos de interesse, eventos dinâmicos e sistema de feedback ao jogador.

- Sistema de Proposta e Votação de Políticas, com validação em vários órgãos do governo (Legislativo, Executivo, Judiciário e Simbólico);
- Sistema de Lealdade e Alinhamento Faccional, com lógica de reação das facções às decisões do jogador;
- Geração de Crises Dinâmicas, com despoletar condicionado por ações anteriores e múltiplos desfechos;
- Gestão Económica e Social, com indicadores como PIB, dívida, sentimento das classes sociais e apoio das forças armadas;
- Interface de Utilizador (UI) para visualização de dados políticos, históricos e económicos;
- Sistema de Feedback com *logs*, relatórios e indicadores de desempenho por turno;
- Persistência de Jogo e carregamento de dados via ficheiros JSON.

Todo o desenvolvimento do jogo foi feito com linguagem C# e o motor Unity, com uma abordagem modular de modo que futuramente fosse fácil a integração de novos conteúdos.

Predominou a metodologia da programação orientada a objetos, também a separação de responsabilidades e reutilização de componentes mediante serviços (*services*), fábricas (*factories*) e controladores de lógica.

2.2 Contexto Narrativo

O jogo decorre no ano de 1953 em plena tensão política, seja pela recente perda do principal governante do partido comunista por cerca de 30 anos, Josef Stalin, como a forte pressão causada pela NATO (*North Atlantic Treaty Organization*) na fronteira da União Soviética.

O jogador assume o papel de *Nikita Khrushchev*, alinhada com a visão reformista, contudo, pode optar pelas ideias militarizadas de *Lavrentiy Beria* ou o equilíbrio de *Georgiy Malenkov* e deve lidar com diversas escolhas políticas, ideológicas e estratégias que influenciarão o futuro do regime.

O *politburo* está dividido entre conservadores, reformistas, moderados, *KGB* (*Komitet Gosudarstvennoy Bezopasnosti*) e o Exército. Ao mesmo tempo, o jogador enfrentará pressões externas da NATO e EUA (Estados Unidos da América) e instabilidade interna, manifestada em protestos da população, tensões nacionalistas no Bloco de Leste e dificuldades económicas pós-guerra.

2.3 Projetos Similares e Soluções Existentes

Para a correta execução e compreensão do jogo que se pretendia elaborar, foram analisados alguns projetos e jogos que partilham características semelhantes. Um dos títulos mais sonantes neste contexto é o jogo *Democracy 4* (*Democracy 4*, 2020) desenvolvido pela Positech Games (*Positech Games*, 1997).

Este é um simulador político no qual o próprio jogador assume o papel de Líder de um país democrático e gere políticas públicas, orçamentos, reações sociais e eleitorais. O jogo tem como base as decisões estratégicas com impacto nas mais variadas áreas, como a educação, economia, saúde ou até mesmo segurança.

Embora o *State of Affairs* esteja inserido num contexto autoritário, a União Soviética pós-Stalin, o que difere de *Democracy 4* que se posiciona num regime mais democrático, estes tendem a partilhar algumas ideias-base, como:

- A gestão de grupos com interesses divergentes;
- A implementação de políticas com impacto político e social;
- A reação dinâmica das entidades afetadas pelas decisões do jogador;
- A estrutura baseada em tomada de decisões com consequências a curto e longo prazo.

Também foram identificados outros projetos semelhantes, um pouco mais distintos daquilo que o *Democracy 4* consegue ser em relação ao *State of Affairs* em termos de ideias de jogo, contudo serviram, também estes, de base para o estudo, entre eles:

- *Suzerain*: que explora uma narrativa política num regime fictício em transição (*Suzerain*, 2020);

- *Tropico*: Focado essencialmente em regimes autoritários numa perspetiva mais económica e satírica (*Tropico 6*, 2019);
- *Crusader Kings III*: Mais virado para a dinastia medieval, contudo, lida com sub-fações, intrigas e gestão política (*Crusader Kings III*, 2020).

3 Bases Teóricas

3.1 Fundamentos de Jogos de Simulação Política

Os jogos de simulação política inserida no grupo de jogos de estratégia cujo objetivo é replicar as iterações dos processos de governação, as decisões adotadas nas mais variadas situações, influência ideológica e gestão de interesses coletivos.

Este tipo de jogabilidade procura replicar as dinâmicas institucionais e ideológicas, permitindo assim que o jogador assuma o papel de líder num universo irreal, enfrentando escolhas que envolvem múltiplas variáveis e consequências.

De acordo com um estudo publicado pela Universidade de Cambridge (Asal et al., 2018), os mini-jogos e simulações iterativas têm demonstrado elevada eficácia no ensino de métodos políticos, pois “requerem a participação ativa dos alunos, permitindo que o conhecimento seja construído e apreendido através da sua própria experiência de aprendizagem” (p. 2). Estes jogos oferecem ao jogador uma compreensão mais profunda das lógicas subjacentes a políticas complexas, ao promoverem ambientes interativos e controlados onde é possível explorar persuasão, conflitos e alianças.

Por sua vez, no estudo efetuado por Marcus Schulzke em “*Video Games and the Simulation of International Conflict*” (Schulzke, 2014), pese embora o foco seja direcionado para jogos simulação de guerra reais, o autor ainda aborda a temática política como ponto fulcral na construção de um jogo de estratégia em particular nos que abordam conflitos, não só simulam eventos históricos, como moldam a perceção do jogador acerca dessas realidades, promovendo assim uma representação ativa de ideologias e equilíbrios de poder.

Enfatizando assim a importância destes jogos naquilo que é a exploração de consequências estratégicas das decisões tomadas.

Por fim, outra abordagem interessante e relevante para o estudo e base teórica do tema foi analisado o artigo “*What do students learn from political simulation games? A mixed-method approach exploring the relation between conceptual and attitudinal changes*” (Oberle et al., 2020), um estudo focado em simulações parlamentares, concluíram que os participantes evidenciam mudanças conceptuais e na atitude

significativas. Os autores referem que os estudantes “compreenderam a relevância dos compromissos na política democrática e perceberam o quão inevitavelmente moroso pode ser o processo de tomada de decisão quando envolve múltiplos interesses divergentes e opostos e se luta por compromissos políticos” (p. 16). Esta capacidade de navegar entre diferentes perspetivas e interesses políticos é um dos principais benefícios das simulações interativas.

Assim, os jogos de simulação política revelam-se particularmente eficazes tanto ao representar temas delicados como na promoção de pensamento estratégico.

Cada jogo de simulação conta com três elementos fulcrais, entre eles:

- Cada decisão do jogador terá impacto no decorrer do jogo, seja na popularidade junto das sub-fações, no equilíbrio do poder ou até mesmo no clima social.
- Há que considerar o leque variado de grupos, sejam eles partidos e militares ou meros cidadãos, em que cada um, lida com as suas prioridades.
- O jogo deve sempre comunicar os resultados das ações do jogador, isto para poder ajustar as suas estratégias e sentir que o mundo reage organicamente às suas decisões.

3.2 Unity e Desenvolvimento de Jogos

No panorama atual do desenvolvimento de jogos, o *Unity* surge como uma das plataformas mais robustas, seja para criar jogos em 2D, 3D ou até mesmo realidade virtual (*Unity*, 2025). Programa este lançado originalmente em 2005 visando tornar acessível o desenvolvimento de jogos. Com o passar dos anos foi ganhando popularidade e notoriedade no ramo do desenvolvimento de jogos e com isto tornando-se uma ferramenta dominante no setor dos videojogos, aplicações educativas, realidade aumentada, etc.

Alguns dos casos práticos em que se evidencia esta transversalidade estão descritos em seguida:

A Volvo, assim como outras marcas para diferentes finalidades, usa Unity para variados testes sob realidade virtual para a avaliação de condução autónoma (*Real-time 3D for enterprise*, 2025).

Também no ramo da construção, empresas como Skanska utilizam Unity para formar operários em ambientes seguros de VR (*Virtual Reality*) (*Virtual Reality Training*, 2025).

Mostrando assim que o Unity desde o seu surgimento até agora conta com uma evolução continua e expansão para as mais variadas áreas, provando que esta ferramenta não é somente utilizada para o desenvolvimento de jogos. O seu sucesso advém, também, de três vetores-chave (*Blog Impulso | Unity, 2025*):

- Sistema de componentes: o *Unity* usa *GameObjects*, que funcionam como “blocos de construção” personalizáveis (*Components*).
- Facilita o trabalho em equipa: com esta estrutura é possível distribuir tarefas eficientemente.
- Linguagem C#: A lógica dos jogos é desenvolvida em C#.

4 Requisitos e Arquitetura do Sistema

4.1 Requisitos Funcionais

Os requisitos descrevem o que o sistema deve fazer, ou seja, são as funcionalidades específicas que o software deve implementar para satisfazer os objetivos do utilizador e os propósitos do sistema (*Requisitos funcionais e não funcionais, 2025*). Portanto, estes representam comportamentos observáveis e interações com o utilizador ou com outros sistemas.

No presente jogo, os requisitos funcionais foram definidos com base nas ações e decisões que o jogador deve poder executar, tendo sempre em conta a complexidade de um sistema político e económico.

Para tal, foram definidos os seguintes pontos fulcrais como requisitos funcionais:

- O jogador pode propor, votar, aprovar e revogar políticas.
- As políticas têm impacto direto ou progressivo no país.
- Deve existir um sistema de facções e sub-facções com comportamentos dinâmicos.
- A lealdade e influência mudam consoante as decisões do jogador.
- Órgãos Governamentais com regras de voto distintas.
- Situações despoletadas devido a condições lógicas ou temporais.
- Classes sociais com sentimento variável conforme as políticas.
- *Feedback* claro mediante notícias, eventos e mudanças estatísticas.
- O jogo deve poder guardar e carregar o estado completo.

Assim, estes requisitos foram definidos para construir uma simulação política o mais realista possível.

4.2 Requisitos Não Funcionais

Diferente dos requisitos funcionais, que descrevem o que o sistema deve fazer, os requisitos não funcionais dizem respeito à forma como esse comportamento deve ser garantido. Como tal, são atributos de qualidade que afetam o desempenho, a fiabilidade, a manutenção, entre outros aspetos críticos da experiência do utilizador (Desenvolvedor, 2023).

Estes requisitos são relevantes em jogos, como o caso deste projeto, uma vez que influenciam diretamente a solidez, escalabilidade e longevidade do software. robustez. Como tal, estes aspetos foram tidos em conta desde as fases iniciais de desenvolvimento.

Modularidade

- A modularidade foi uma prioridade desde o início do projeto. Esta permite dividir o sistema em componentes autónomos, que podem ser desenvolvidos isoladamente. Esta separação facilita tanto a gestão de código como também o trabalho colaborativo (*Modularidade*, 2025).
- No contexto específico deste jogo, optou-se por desenvolver módulos distintos para cada sistema principal, como o caso do *FactionManager*, *EffectManager* e *PolicyManager*.

Escalabilidade

- Num jogo de simulação política, é expectável surgirem necessidades de expansão, seja através da adição de novos países, eventos, políticas ou até mesmo ideologias. Assim, a escalabilidade foi tida como requisito essencial, no sentido de garantir que o sistema possa crescer sem exigir reestruturações profundas.
- Para tal, foi adotada uma abordagem orientada a dados, onde ficheiros JSON externos armazenam a maioria dos conteúdos dinâmicos. Assim, possibilita-se adicionar, por exemplo, uma nova facção ou política somente com alterações no ficheiro de configuração, sem necessidade de reescrever código base. Esta estratégia demonstrou-se eficaz para manter a integridade e estabilidade do sistema ao longo do tempo (*Escalabilidade e desempenho*, 2025).

Manutenção

- A manutenção está intrinsecamente ligada à clareza, organização e documentação do código. Sabendo que a continuidade de um projeto

depende frequentemente da capacidade de compreender e adaptar o trabalho já realizado, foi dada especial atenção à estruturação do código.

- Optou-se pela utilização de práticas como a criação de *factories*, *services* e *managers*, que não só modularizam responsabilidades como tornam a leitura e extensão mais intuitiva. Além disso, a documentação técnica, desenvolvida em paralelo com o código, revelou-se fundamental para facilitar a entrada de novos membros na equipa e para assegurar coerência nas decisões técnicas. A realização de testes E2E, abordados mais adiante no ponto 5.5 contribuiu também para validar funcionalidades e detetar regressões precocemente.

Desempenho

- O desempenho de um jogo não se refere somente à taxa de fotogramas por segundo, mas também ao tempo de carregamento, fluidez em cada transição e capacidade de resposta da interface. Considerando que o jogo trabalha com vários sistemas simultâneos como económicos, políticos e sociais, garantir um desempenho estável era imperativo.
- Foram, por isso, implementadas diversas técnicas de otimização: desde o carregamento assíncrono de dados à reutilização de objetos. A interface gráfica foi concebida para evitar atualizações desnecessárias e sobrecarga de elementos ativos, assegurando assim uma experiência fluida mesmo em cenários mais complexos (matheus, 2024).

Persistência de Dados

- Em jogos de estratégia e simulação, onde as consequências das decisões estendem-se por vários turnos, a persistência é crucial. O sistema precisava de garantir que o jogador pudesse interromper e retomar o jogo sem perda de progresso ou inconsistências nos dados.
- Para tal, foi implementado um mecanismo de serialização modular, em que os dados do país, políticas, lealdade das fações e situação económica são guardados organizadamente em ficheiros JSON. No momento de carregamento, estes são lidos e reconstruídos com base nas *factories* existentes, restaurando o estado exato da sessão anterior. Esta solução revelou-se simples, mas eficaz, para garantir continuidade e integridade (*O que é Chroma DB?*, 2025).

Experiência do Utilizador (UI/UX)

- A interface do utilizador foi pensada com um propósito claro: apresentar informação complexa de forma acessível. Num jogo com múltiplos sistemas interligados, a clareza visual e o *feedback* imediato são determinantes para garantir que o jogador se sinta no controlo da situação.
- A equipa adotou princípios básicos de UI/UX, com destaque para a organização do dossiê informativo, *feedback* visual nas decisões políticas e relatórios de turno que refletem as mudanças no estado do país. Este cuidado com a apresentação contribuiu significativamente para a imersão e para a tomada de decisões mais conscientes por parte do jogador (UXCam, 2025).

4.3 Arquitetura do Sistema

A arquitetura de sistemas é responsável por definir a estrutura, os componentes e as interações de um sistema, moldando a construção e evolução ao longo do tempo. Ou seja, este contém o planeamento e organização da estrutura de um sistema, definindo como os componentes se intercomunicam e interagem entre si e com os utilizadores (Prada, 2024).

Neste projeto, a arquitetura foi planeada com base em princípios de modularidade e encapsulamento, seguindo boas práticas de programação e orientada de modo que fosse mais fácil efetuar a manutenção, evolução e respetiva reutilização de componentes.

Posto isto, cada sistema que completa o jogo, como o caso de políticas, sub-fações, corpos governamentais, entre muitos outros, são encapsulados num módulo próprio com responsabilidades bem definidas.

Estrutura Modular

Cada um dos módulos de jogo, tende a seguir uma estrutura, os quais são normalmente compostas por:

- *Data Classes*: Estas representam as classes orientadas a dados que ficam responsáveis por armazenar e estruturar as informações obtidas por meio de ficheiros JSON no decorrer das fases de inicialização do jogo. As mesmas somente descrevem os atributos necessários para a construção dos objetos. Alguns exemplos destes são *PolicyData*, *FactionData*, *SituationData*, etc.

- *Runtime Classes*: Neste caso refere-se aos objetos que representam entidades vivas no decorrer do jogo, ou seja, possuem um comportamento e interagem entre si. Estes são instanciados com base nas Data Classes, abordadas no ponto anterior, e ficam responsáveis por aplicar lógicas como efeitos, condições e modificações de estado. Exemplos deste são *Policy*, *Faction*, *Situation*, etc.
- *Factories*: Nestas classes o foco é encapsular a lógica de criação de objetos. Classes como *PolicyFactory*, *EffectFactory*, etc. ficam responsáveis por transformar Data Classes em objetos *Runtime*, permitindo assim assegurar maior consistência, a validação dos dados e aplicação de *defaults*.
- *Managers*: Para estas classes o objetivo é acompanhar, atualizar e coordenar o comportamento dos vários objetos. Um exemplo disso é o *PolicyManager* que gere todas as políticas presentes no jogo, fica responsável por gerir a evolução ao longo dos turnos, aprovações, rejeições, entre outros fatores.
- *Services*: Já no caso dos *services*, estes encapsulam lógicas transversais ao sistema, ou seja, funcionalidades que não pertencem somente a um módulo, mas usadas por vários. O *VotingService*, por exemplo, calcula os votos de diferentes sub-fações, ou o *FactionEffectsService* que aplica os efeitos às lealdades.

Justificativa do design modular

A modularidade utilizada no projeto promove uma maior coesão, em que cada módulo é responsável por uma dada funcionalidade, e baixo acoplamento entre os módulos (Ravi, 2022).

Este padrão facilita em tarefas como:

- Manutenção simplificada;
- Testes isolados e reutilização entre projetos;
- Desenvolvimento paralelo.

Ciclo de Vida dos Módulos

- Carregamento, no início do jogo, todos os dados são imediatamente carregados de ficheiros JSON usando a classe *JsonLoader*;
- Criação de Objetos, cujos dados carregados são convertidos em objetos do jogo através das respetivas fábricas (*factory*);
- Gestão e Atualização, durante o jogo, os managers acompanham o estado de cada sistema e coordenam as atualizações no decorrer de cada turno;

- Persistência e Salvaguarda, o sistema permite guardar e respetiva restauração do estado completo do jogo, incluindo políticas ativas, lealdade das sub-fações, situações em curso e *logs* de eventos, etc.

4.3.1 Diagrama de Casos de Utilização

No que toca aos diagramas de casos de utilização em UML, estes têm o objetivo de demonstrar os detalhes dos utilizadores de um dado sistema, conhecidos como atores, e as interações deles. Para tal, usa-se um conjunto de símbolos e conectores especiais (*Diagrama de caso de uso UML*, 2025).

Estes diagramas permitem ajudar equipas a representar:

- Cenários em que um dado sistema interage com pessoas, organizações ou sistemas externos.
- Metas que o sistema ajuda essas entidades a atingir;
- O âmbito do sistema.

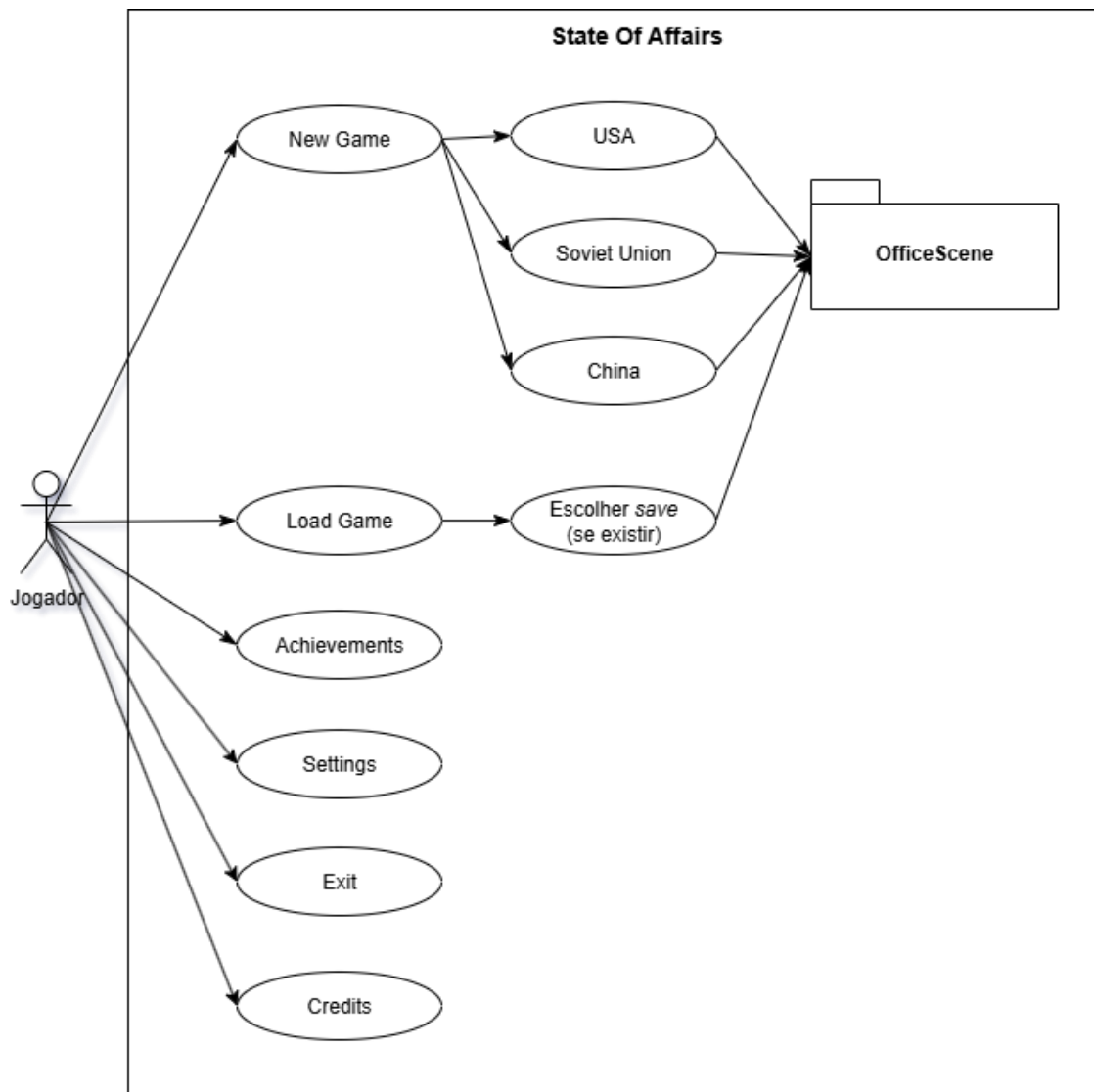
Para o presente projeto, os diagramas foram criados de modo a exemplificar de forma mais clara e superficial a interação do jogador com os principais módulos do jogo, facilitando assim a compreensão das diferentes funcionalidades.

Foram identificadas e representadas três áreas principais, cada uma com as suas ações e ramificações específicas, que no caso trata-se do Menu principal por onde qualquer jogador necessita de passar, em seguida o *OfficeScene* que é desbloqueado imediatamente após e por fim o *Politburo* que representa todo o momento de proposição e votação de políticas:

Menu Inicial

Neste primeiro diagrama está representado o ponto de entrada do jogador no jogo. Através do menu principal, o jogador pode iniciar um novo jogo, carregá-lo previamente guardado ou consultar os *achievements* desbloqueados tal como é possível identificar na figura 1.

Figura 1. Diagrama UML Casos de Utilização Iniciar jogo



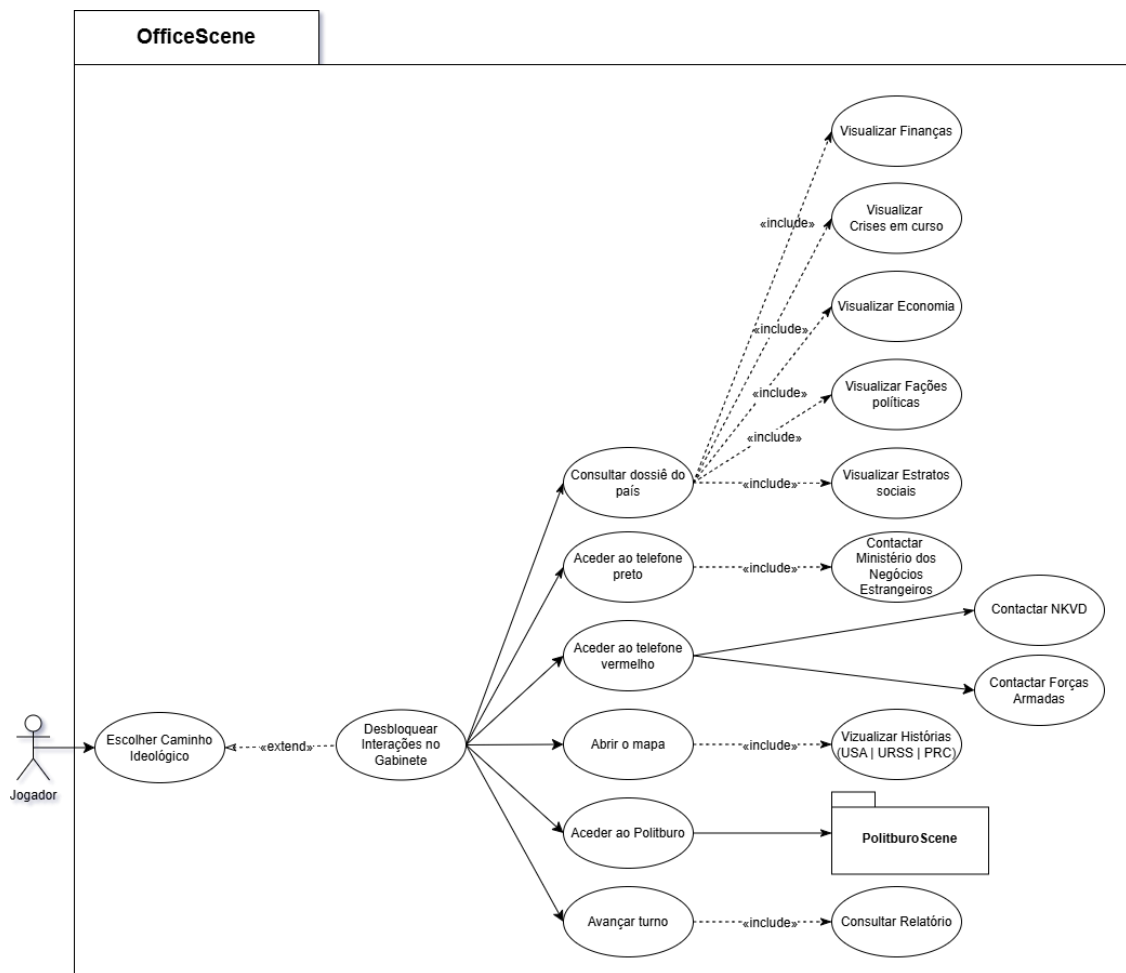
Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

Ao seleccionar um novo jogo, o jogador é direccionado para um menu que permite escolher a narrativa principal que representa um país, e posteriormente entra na *OfficeScene*, onde decorre a maioria da jogabilidade inicial.

OfficeScene

O segundo diagrama representa as interações possíveis no gabinete, após escolha do caminho ideológico. Aqui o jogador pode consultar um dossiê com dados económicos, populacionais e políticos, usar diferentes telefones com finalidades opostas para contactar grupos de interesse ou membros de grande influência. Pode também aceder ao Politburo para propor novas políticas assim como avançar turno e respetivamente visualizar relatórios de turno que contém atualizações assim que se transita de semana.

Figura 2. Diagrama UML Casos de Uso Office Scene



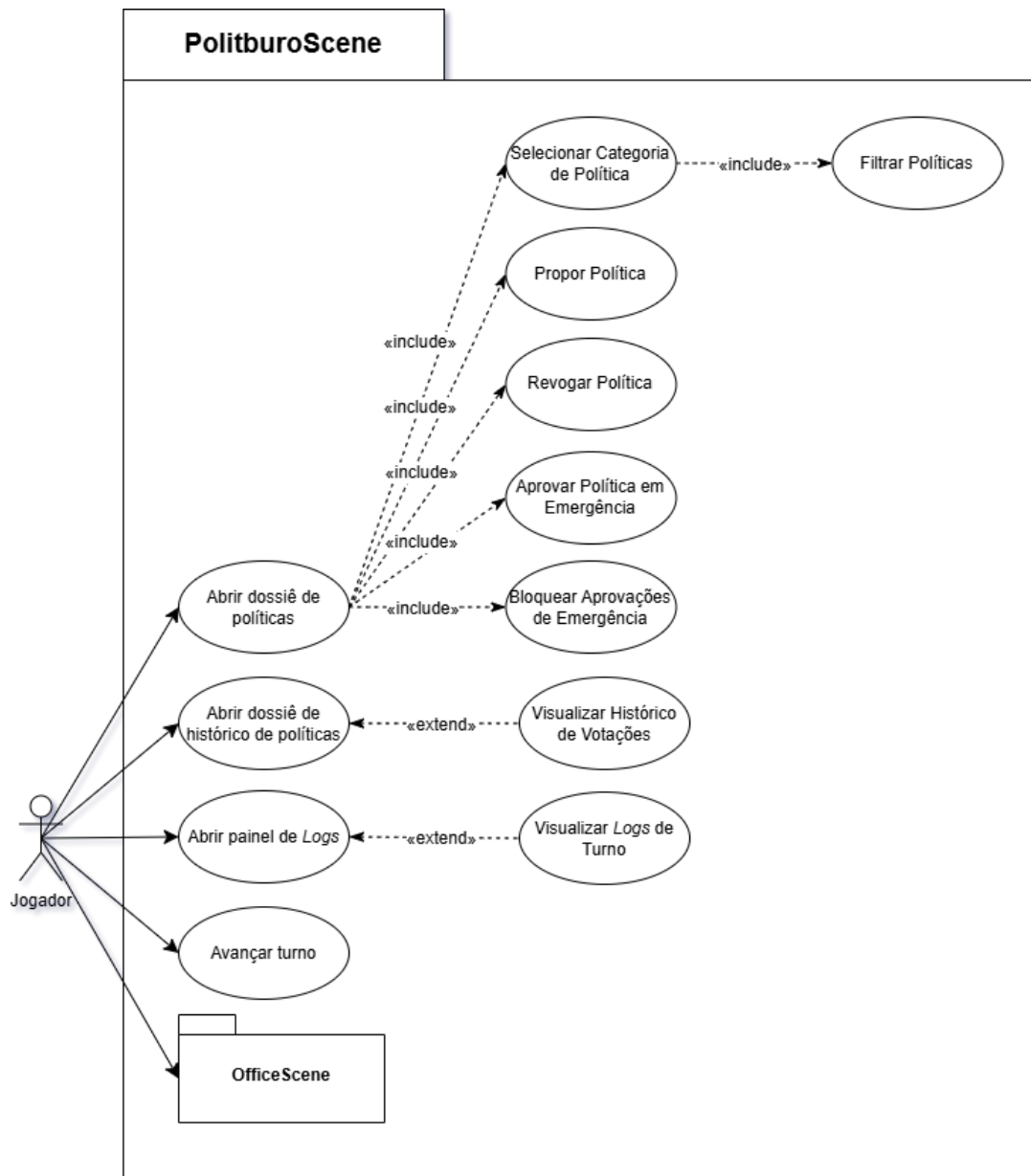
Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

PolitburoScene

O terceiro diagrama representa as ações disponíveis na cena do Politburo, ações essas que podem ser propor políticas, revogar políticas existentes, aceder a diferentes modos de votação como aprovação em emergência ou o bloqueio de aprovações de emergência, pode-se também aceder a dossiê com o histórico de políticas e visualizar mensagens de estado atualizadas com o passar dos turnos.

O utilizador pode também retornar ao *OfficeScene* e avançar o turno. Tudo isso está presente no seguinte diagrama.

Figura 3. Diagrama UML Casos de Uso Politburo Scene



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

4.3.2 Fluxogramas

Fluxogramas representam descrições de processos, sistemas ou algoritmos e são muito úteis em várias áreas para documentar, estudar, planejar, melhorar e comunicar processos que tendem a ser mais complexos por meio de diagramas claros e fáceis de entender (*O que é um fluxograma?*, 2025).

Também aqui são utilizados formas geométricas para definir os tipos de passos de modo a definir o fluxo e sequência.

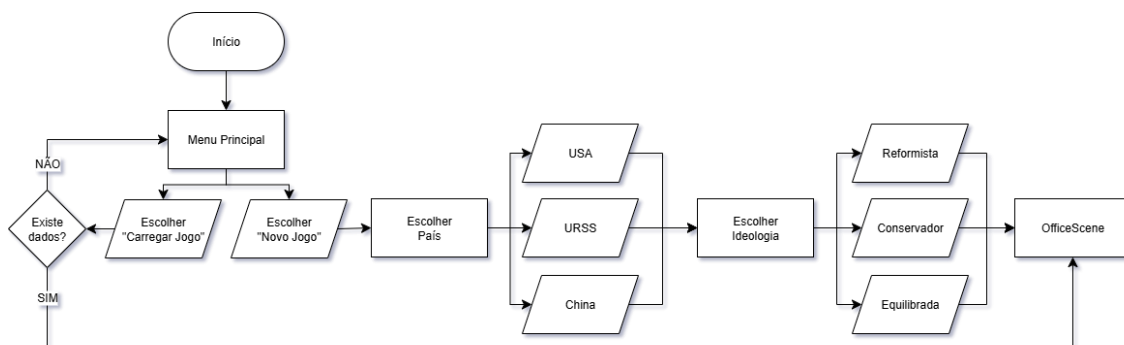
Para complementar a explicação sobre a arquitetura e funcionamento do sistema em geral, desenvolvi três diagramas que representam as principais fases dos percursos do jogador. Com isto será mais fácil visualizar o encandeamento lógico das interações no decorrer do jogo, desde o seu arranque até às ações possíveis nas diferentes cenas.

Menu Inicial

O seguinte diagrama representa todas as opções iniciais, desde o momento é que é dado *play* até à entrada em *OfficeScene*.

Inclui a seleção entre iniciar um novo jogo, carregar algum save anterior, caso já tenha jogado antes, posteriormente passa para a escolha da narrativa, principal, que de momento só estará disponível a *URSS*, a escolha da ideologia a seguir, e, por fim, o início efetivo da jogabilidade com a apresentação do painel de boas-vindas e indicadores do estado inicial do país.

Figura 4. Diagrama de Fluxo do menu inicial

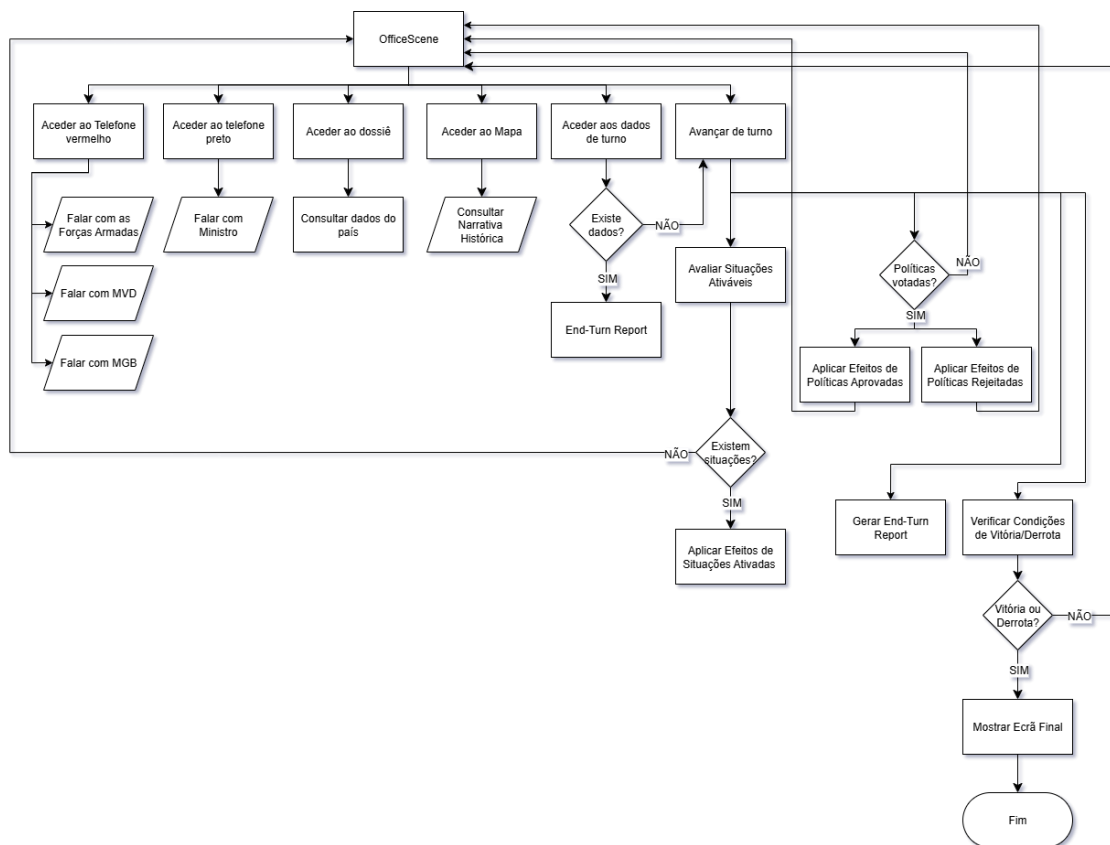


Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

OfficeScene

O segundo diagrama descreve as várias ações disponíveis ao jogador na *OfficeScene*. A partir daqui o jogador pode aceder ao telefone vermelho ou preto, consultar o dossiê do país, visualizar a narrativa histórica no mapa, avançar o turno ou dirigir-se ao *Politburo*. Cada uma destas interações pode desencadear sistemas distintos, como a análise de dados do país que será sempre atualizado ao final de cada turno, como a economia, setores, lealdade das sub-fações e grupos de interesse, etc., até mesmo entrar em contacto com pessoas de cargos importantes com alta influência.

Figura 5. Diagrama de Fluxo do OfficeScene

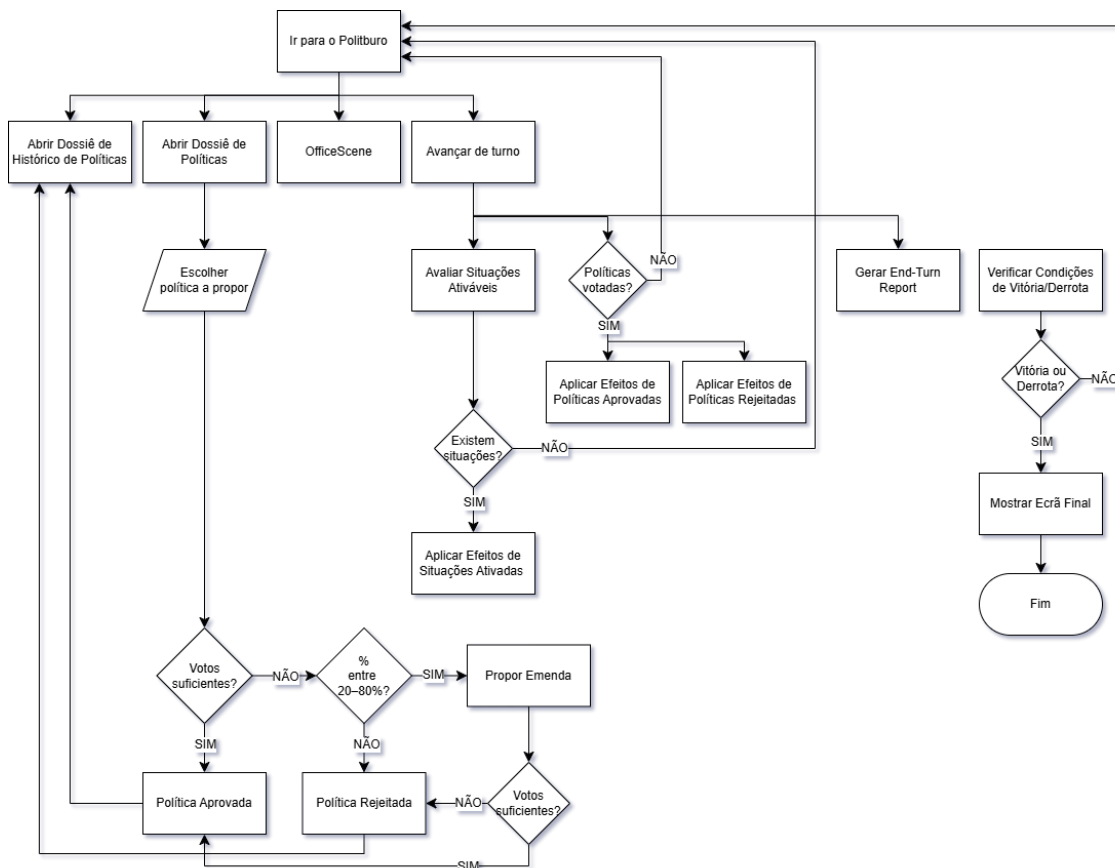


Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

PolitburoScene

Por fim, este diagrama apresenta o funcionamento da *PolitburoScene*, onde se realizam as propostas políticas. Aqui, o jogador pode consultar o histórico de políticas, votar e, em determinados casos, entrar em fase de negociação quando uma proposta não é reprovada ou aprovada de imediato.

Figura 6. Diagrama de Fluxo do Politburo



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

5 Atividades e Trabalho Realizado

5.1 Organização do Trabalho

As atividades do estágio foram desenvolvidas usando metodologias ágeis com uma abordagem *Scrum*, que tinha como principal objetivo permitir uma melhor gestão do trabalho mediante divisão do projeto em ciclos curtos e bem definidos, no caso, os sprints, com entregas contínuas e foco na adaptação e melhorias constantes (Atlassian, 2025b).

Para este estágio, o *Scrum* foi aplicado de forma híbrida, a considerar que a equipa se encontrava remota e multifuncional. Como tal, cada *sprint* tinha uma duração

de duas semanas, sendo sempre iniciado com uma reunião de planeamento, onde o supervisor e os restantes três elementos da equipa de desenvolvimento, no caso todos os Engenheiros Informáticos, incluindo eu, porém ainda em formação, se definiam os objetivos e respetivas tarefas a serem realizadas nesse ciclo.

As tarefas eram distribuídas por diferentes áreas de trabalho, como a criação de novas funcionalidades, a reorganização do código, desenvolvimento de elementos narrativos do jogo, construção de documentação e melhorias na interface do utilizador (UI/UX), cujo trabalho era desenvolvido em colaboração com a equipa de design.

No final de cada sprint, fazíamos uma revisão em conjunto do que havia sido feito. Análise esta que se focava no que foi concluído, nos desafios encontrados e possíveis melhorias para o sprint seguinte.

Para organizar o trabalho, utilizámos o *Jira* (*Jira*, 2025), uma ferramenta de gestão de projetos do ecossistema *Atlassian* (*Atlassian*, 2025a) em conjunto com o *Confluence* (*Confluence*, 2025) para a documentação.

Figura 7. Logo grupo Atlassian



Nota. Adaptado de Atlassian, <https://www.atlassian.com>.

Atualizações Assíncronas Diárias

De modo a acompanhar melhor aquilo que foi o desenvolvimento no dia anterior, foi adotado um modelo de comunicação assíncrona diária, em que cada um partilhava um resumo breve do progresso obtido no dia anterior, do que pretende fazer no presente dia e também possíveis bloqueios que estejam a impedir algum desenvolvimento.

Reuniões de *Check-In* Semanais

Às terças e quintas-feiras, realizavam-se reuniões curtas de aproximadamente 15 minutos via *Microsoft Teams* (*Microsoft Teams*, sem data). Reuniões essas que abordavam:

- Partilhar o progresso individual de cada membro;
- Reavaliar o estado das tarefas em curso;
- Ajustar prioridades conforme necessário;
- Discutir e resolver bloqueios técnicos em grupo.

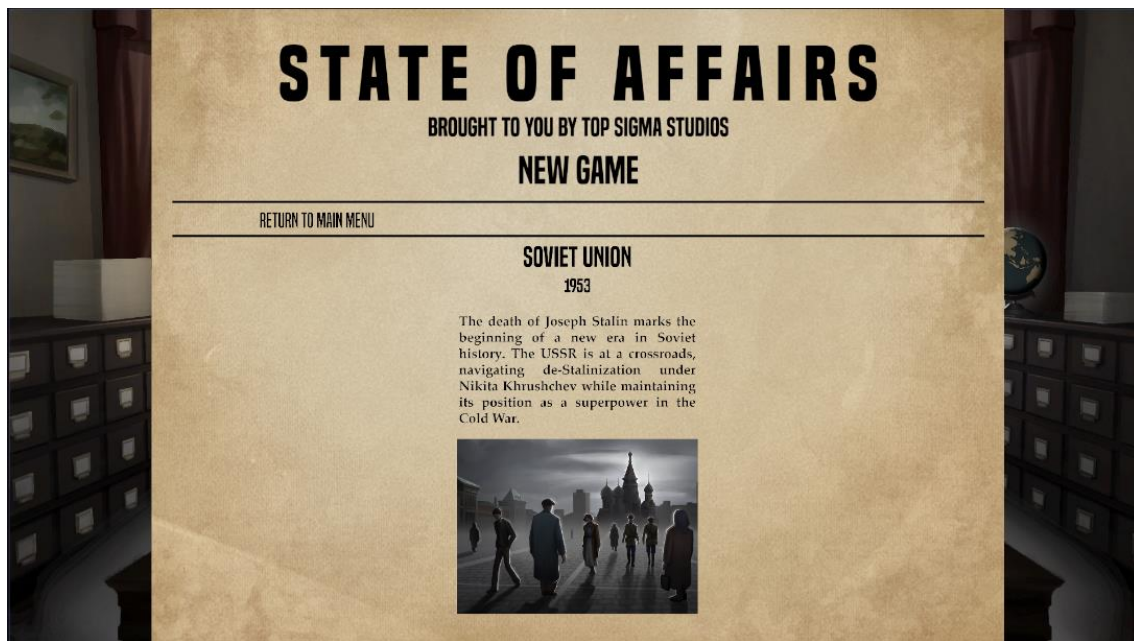
5.2 Desenvolvimento de Funcionalidades

Antes de iniciar o processo de desenvolvimento do jogo, foi importante perceber como iniciar o jogo até então desenvolvido, e, como tal, a figura abaixo exemplifica o menu inicial com as opções de “*New Game*” e “*Load Game*” e “*Achievements*”. Após selecionado um novo jogo é carregado um painel com diferentes eventos históricos a seguir. No momento do desenvolvimento, o único disponível é URSS pós-Stalin, porém, no futuro haverá mais como EUA e China.

Figura 8. Menu principais



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

Figura 9. *Novo jogo URSS*

Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

5.2.1 *Unity Essentials*

Durante os primeiros dias de estágio, pertencente ao sprint inicial, que decorreu na primeira semana, foquei-me na integração ao projeto e na consolidação dos conhecimentos essenciais para contribuir efetivamente para o desenvolvimento do jogo.

Comecei por aprofundar o conhecimento sobre o conceito, narrativa e objetivos do projeto. Fiquei a conhecer a cultura da empresa, metodologias de trabalho e organização dos sprints.

Realizei e completei o curso *Unity Essentials* (Unity Learn, 2025), onde consolidei os fundamentos da *engine Unity*, com foco em:

- Manipulação de *GameObjects*;
- Sistema de física;
- Interfaces de utilizador (UI);
- Conexão com o *script*.

À posterior, iniciei a exploração do repositório do projeto:

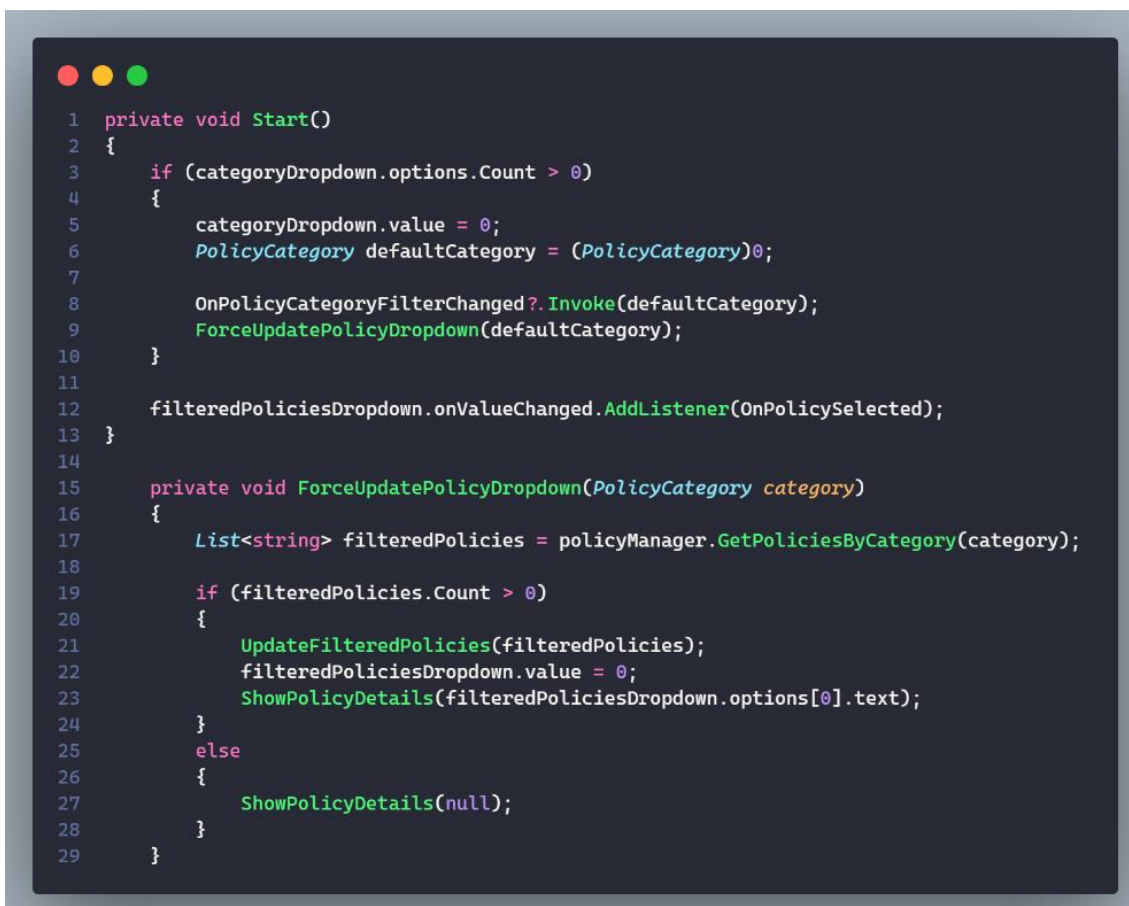
- Analisei a estrutura existente do código;
- Estudei as funcionalidades já implementadas;
- Preparei o ambiente de trabalho para iniciar a programação.

5.2.2 Develop Policy Proposal System

Início da semana seguinte começa o sprint 1, cuja tarefa consistiu em implementar o sistema responsável por permitir que o jogador possa selecionar uma política, visualizar todos os detalhes, desde descrição até efeitos que dela virão em caso de aprovada ou rejeitada.

Inicialmente existia um problema com o programa, em que ao iniciar a respetiva política referente ao primeiro setor económico, que aparece por defeito, não era carregada. Para que a política desse setor fosse exibida no *dropdown* era necessário trocar entre setores para que funcionasse. Como tal, esse problema foi solucionado com recurso ao método *start*, alocado especificamente ao *Unity* para que o *dropdown* exibisse a política inicial ao iniciar o jogo.

Figura 10. Método *Start* e *ForceUpdatePolicyDropdown*



```
1 private void Start()
2 {
3     if (categoryDropdown.options.Count > 0)
4     {
5         categoryDropdown.value = 0;
6         PolicyCategory defaultCategory = (PolicyCategory)0;
7
8         OnPolicyCategoryFilterChanged?.Invoke(defaultCategory);
9         ForceUpdatePolicyDropdown(defaultCategory);
10    }
11
12    filteredPoliciesDropdown.onValueChanged.AddListener(OnPolicySelected);
13 }
14
15 private void ForceUpdatePolicyDropdown(PolicyCategory category)
16 {
17     List<string> filteredPolicies = policyManager.GetPoliciesByCategory(category);
18
19     if (filteredPolicies.Count > 0)
20     {
21         UpdateFilteredPolicies(filteredPolicies);
22         filteredPoliciesDropdown.value = 0;
23         ShowPolicyDetails(filteredPoliciesDropdown.options[0].text);
24     }
25     else
26     {
27         ShowPolicyDetails(null);
28     }
29 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

O método *ForceUpdatePolicyDropdown()* foi criado para forçar a sincronização entre a lista de políticas e a interface, assegurando que os detalhes da primeira política disponível são apresentados corretamente assim que o jogo inicia.

Posteriormente foi desenvolvido o *ShowPolicyDetails()*, função central que exhibe todos os dados relevantes da política selecionada, incluindo nome, descrição, custo, efeitos e aprovações necessárias nos vários órgãos de decisão.

Figura 11. Método *ShowPolicyDetails*

```
1 private void ShowPolicyDetails(string policyName)
2 {
3     List<PolicyData> allPolicies = JsonLoader.LoadPolicies();
4     PolicyData selectedPolicy = allPolicies.FirstOrDefault(p => p.policyName == policyName);
5
6     if (selectedPolicy != null)
7     {
8         policyNameText.text = selectedPolicy.policyName;
9         policyDescriptionText.text = selectedPolicy.description;
10        policyCostText.text = $"Cost: {selectedPolicy.cost}";
11
12        string effectsText = "Enactment Effects:\n";
13        if (selectedPolicy.enactmentEffects.Count > 0)
14        {
15            foreach (EffectData effect in selectedPolicy.enactmentEffects)
16            {
17                effectsText += $"- {effect.effectType}: {effect.effectValue} ({effect.targetGroup}, {effect.durationTurns} turns)\n";
18            }
19        }
20        else
21        {
22            effectsText += "No enactment effects.\n";
23        }
24
25        effectsText += "\nRemoval Effects:\n";
26        if (selectedPolicy.removalEffects.Count > 0)
27        {
28            foreach (EffectData effect in selectedPolicy.removalEffects)
29            {
30                effectsText += $"- {effect.effectType}: {effect.effectValue} ({effect.targetGroup}, {effect.durationTurns} turns)\n";
31            }
32        }
33        else
34        {
35            effectsText += "No removal effects.\n";
36        }
37        effectsText += "\nRejection Effects:\n";
38        if (selectedPolicy.enactmentRejectionEffects.Count > 0)
39        {
40            foreach (EffectData effect in selectedPolicy.enactmentRejectionEffects)
41            {
42                effectsText += $"- {effect.effectType}: {effect.effectValue} ({effect.targetGroup}, {effect.durationTurns} turns)\n";
43            }
44        }
45        else
46        {
47            effectsText += "No rejection effects. \n";
48        }
49
50        policyEffectsText.text = effectsText;
51
52        string approvalsText = "Approvals:\n";
53        foreach (string approval in selectedPolicy.requiredApprovals)
54        {
55            approvalsText += $"- {approval}\n";
56        }
57
58        policyApprovalsText.text = approvalsText;
59        policyDetailsPanel.SetActive(true);
60    }
61    else
62    {
63        policyDetailsPanel.SetActive(false);
64    }
65 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Este sistema contribuiu para ficarmos mais claro para o utilizador o impacto de cada política, cujo resultado está ilustrado a seguir na figura 12 no painel de detalhes.

Figura 12. Detalhes das políticas



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

5.2.3 Develop Political Consequences of Legislative Failure

Terminada a sprint 1 e começando com o sprint 2, no decorrer desta tarefa, o objetivo era permitir que sempre que uma política proposta fosse rejeitada ativasse consequências diretas no país. Este sistema pretende refletir o impacto da instabilidade legislativa na estabilidade política do regime, funcionando assim como uma mecânica penalizadora para todas as más decisões estratégicas.

Como tal, a estabilidade é representada numa fase inicial do projeto com um valor entre 0 e 100, armazenado na classe *Country*, visto que este representa todos os dados de um país.

Figura 13. Estabilidade política

```

1  public float politicalStability = 100f;
2
3  public void DecreasePoliticalStability(float amount, GameUIManager gameUIManager)
4  {
5      politicalStability = Mathf.Clamp(politicalStability - amount, 0, 100);
6      gameUIManager.UpdatePoliticalStabilityUI(politicalStability);
7  }
8
9  public void IncreasePoliticalStability(float amount, GameUIManager gameUIManager)
10 {
11     politicalStability = Mathf.Clamp(politicalStability + amount, 0, 100);
12     gameUIManager.UpdatePoliticalStabilityUI(politicalStability);
13 }

```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Para ser possível implementar de forma contextualizada, ou seja, aprovação, rejeição ou até mesmo remoção de políticas, foi desenvolvido um método *ApplyPolicyStabilityChange()* na classe *CountryManager*, classe esta usada para controlar toda a lógica de um país.

Para ficar mais realista, o método recebe também um contexto que engloba valores importantes sobre o país em questão.

Figura 14. Contexto da estabilidade

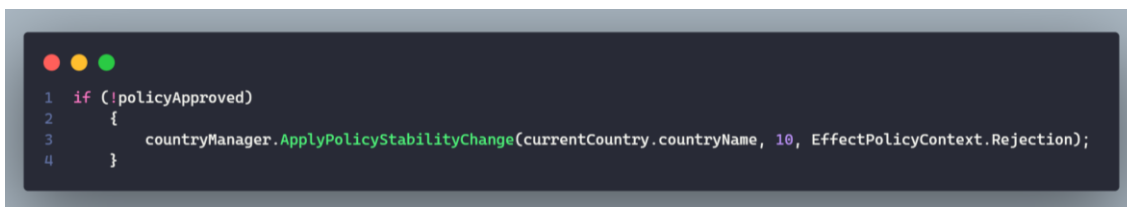
A imagem mostra uma captura de ecrã de um editor de código no Visual Studio Code. O código é escrito em C# e trata-se de um switch statement que recebe um contexto e executa ações diferentes com base nele. O código é o seguinte:

```
1 switch (context)
2 {
3     case EffectPolicyContext.Enactment:
4         currentCountry.IncreasePoliticalStability(effectValue, gameUIManager);
5         break;
6     case EffectPolicyContext.Rejection:
7         currentCountry.DecreasePoliticalStability(effectValue, gameUIManager);
8         break;
9     case EffectPolicyContext.Removal:
10        currentCountry.DecreasePoliticalStability(effectValue * 0.2f, gameUIManager);
11        break;
12 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Este sistema acabou por ser à posterior implementado no fluxo principal de votação de políticas, através do método *VoteOnPolicy()* presente na classe *GameStateManager*. Assim, quando uma política proposta for rejeitada, é disparada uma ação com efeitos negativos que incidem diretamente na estabilidade política.

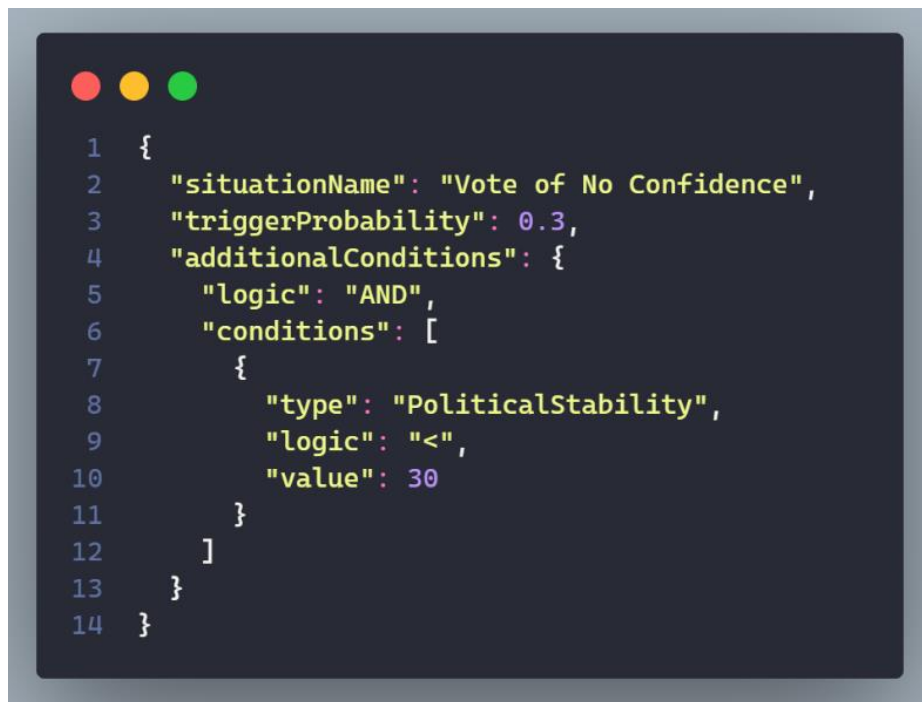
Figura 15. Ativação de efeitos

A imagem mostra uma captura de ecrã de um editor de código no Visual Studio Code. O código é escrito em C# e trata-se de um if statement que verifica se uma política não foi aprovada e, em caso afirmativo, chama o método ApplyPolicyStabilityChange da classe CountryManager com o contexto de rejeição. O código é o seguinte:

```
1 if (!policyApproved)
2 {
3     countryManager.ApplyPolicyStabilityChange(currentCountry.countryName, 10, EffectPolicyContext.Rejection);
4 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

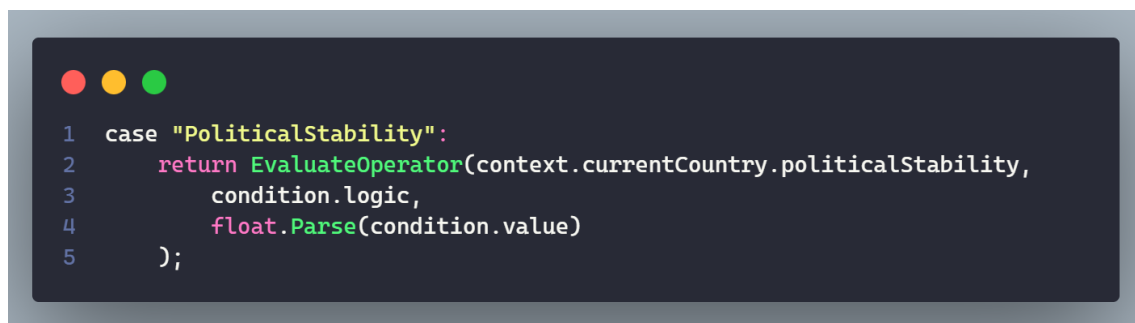
Além disso, e para que toda a implementação fizesse sentido, foi criada uma situação designada por “Vote of No Confidence”, definida no ficheiro *situations.json*, que ao ser ativada logo que os valores da estabilidade política ficam abaixo de 30 afeta diretamente a narrativa e o rumo do próprio jogo.

Figura 16. *Exemplo de situação*

```
1  {
2    "situationName": "Vote of No Confidence",
3    "triggerProbability": 0.3,
4    "additionalConditions": {
5      "logic": "AND",
6      "conditions": [
7        {
8          "type": "PoliticalStability",
9          "logic": "<",
10         "value": 30
11        }
12      ]
13    }
14  }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Já a verificação que desperta a situação definida no *JSON* é através da função *SituationManager* que é usada, também, para ativar diversas.

Figura 17. *Ativação da situação*

```
1  case "PoliticalStability":
2    return EvaluateOperator(context.currentCountry.politicalStability,
3        condition.logic,
4        float.Parse(condition.value)
5    );
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Este sistema foi criado com um propósito, contudo, até então como forma de teste para, de facto, impactar a estabilidade política e trazer consequências reais em níveis baixos, assim como outras situações novas no futuro que poderão surgir.

5.2.4 Create Placeholder UI to Consult Historical Policy Data

Para esta tarefa, o objetivo central seria criar um painel, mais tarde substituído com recurso aos *assets* desenvolvidos pela equipa de *design*, que simulasse um dossiê que ao ser clicado devolvia todas as políticas anteriormente propostas com respetivos resultados e números de votos. Esta funcionalidade permite assim ao utilizador consultar todas as decisões políticas tomadas, promovendo uma maior imersão e noção de continuidade.

A ideia foi criar uma estrutura paginada, acessível na cena *PolitburoScene*. A interface permitia visualizar informação como o título da política, descrição, data de votação, resultado e número de votos.

Objetivos da funcionalidade:

- Registrar políticas aprovadas ou rejeitadas.
- Apresentar os dados numa *interface* paginada.
- Permitir navegação com botões *Next* e *Prev*.
- Garantir a ausência de duplicações por política e turno.

O *script PolicyArchiveUI* fica responsável por instanciar com base nos dados guardados, criar navegação e garantir assim a integridade de toda a lista.

No *Unity* desenvolveu-se um *Prefab* denominado de *PolicyPage_Placeholder*, este será exibido como uma página do dossiê, com campos de texto para cada dado da política proposta.

Figura 18. Método *Initialize*



```
1 public void Initialize(string title, string description, string date, string result, Dictionary<string, int> votes)
2 {
3     titleText.text = title;
4     descriptionText.text = description;
5     dateText.text = date;
6     resultText.text = result;
7     resultText.color = result switch {
8         "Approved" => Color.green,
9         "Rejected" => Color.red,
10        "Ongoing" => Color.yellow,
11        _ => Color.white
12    };
13    votesText.text = FormatVotes(votes);
14 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

De modo a garantir que o histórico das políticas é mantido entre os vários turnos e também cenas, foi utilizada a classe *SceneChangeDataHolder*, que armazena numa lista de objetos *PolicyArchiveData*.

Estes dados são automaticamente carregados no início da cena através do método *RefreshArchiveUI()*.

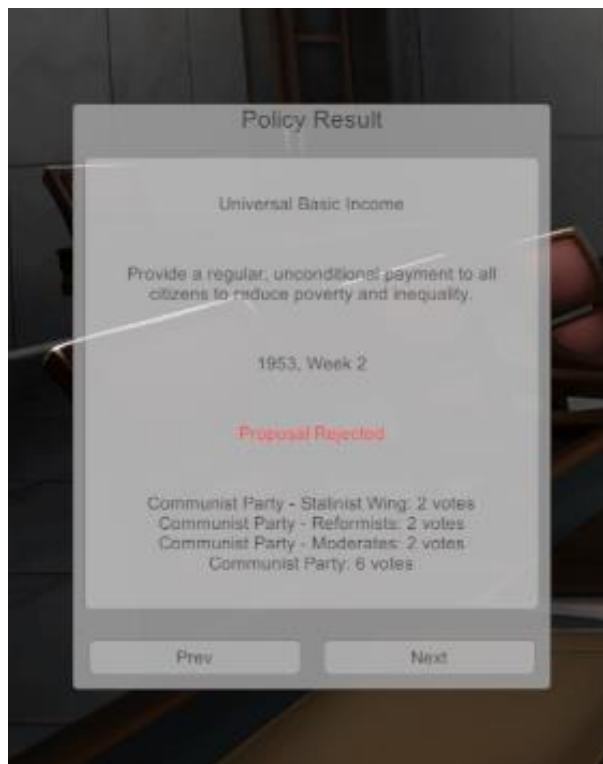
Figura 19. Método *RefreshArchiveUI*

```
1 sceneChangeDataHolder.AddArchivedPolicy(  
2     new PolicyArchiveData(policy.policyName, policy.description, currentDate, result, votes)  
3 );
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Assim, ao longo dos turnos e das várias políticas propostas é possível aceder aos documentos e através dos botões *Next* e *Prev* navegar entre páginas para visualizar todo o histórico. Na figura seguinte é possível verificar o painel aberto após ter sido clicado no dossiê responsável por guardar o histórico de políticas.

Figura 20. Arquivo de políticas votadas



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

5.2.5 Define Industry Sector Activities and Resource Flow

Entretanto, cada país conta com setores económicos importantes para o fluxo financeiro. Como tal, uma simulação do setor industrial foi implementada como um dos pilares centrais da economia soviética em *State of Affairs*. A principal função passa por representar os fluxos produtivos da indústria, desde a produção de energia à transformação de matérias-primas, permitindo gerar recursos estratégicos como maquinaria, veículos, eletricidade ou infraestruturas.

Inicialmente era importante expandir a estrutura do ficheiro *countries.json*, adicionando o setor “*Industry*” com quatro subcategorias:

- *Heavy Manufacturing*
- *Energy Production*
- *Resource Refinement*
- *Construction*

Cada uma destas subcategorias foi configurada com diversos recursos de entrada e de saída, permitindo assim simular cadeias produtivas interligadas.

Para a produção de recursos foi criado um método por um dos colaboradores, o *ProduceResources()* na classe *EconomicSector*. Este método verifica a disponibilidade atual, consome os recursos de entrada necessários e adiciona os recursos de saída ao *stock* nacional do país.

No caso da Indústria, um dos recursos mais importantes que acaba por influenciar os demais setores económicos e subcategorias é a produção de eletricidade, processada primeiro para garantir que todos consigam funcionar com base nos recursos produzidos nesse mesmo turno.

Com isto, é possível implementar o encadeamento de produção de forma lógica, nomeadamente, a subcategoria “*Energy Production*” gera eletricidade, essencial para “*Heavy Manufacturing*” e “*Resource Refinement*”.

A produção de “*Steel*” e “*Machinery*” em setores industriais influencia diretamente o desempenho de outras áreas como Agricultura ou Infraestruturas e também em caso de não existir recursos suficientes, a produção é interrompida e registada no *Debug.Log*.

A visualização deste módulo está presente dentro do *office* no dossiê central. Dossiê esse desenvolvido pela equipa e evoluindo à medida que novos dados provenientes do desenvolvimento de diversas tarefas deveriam ser adicionados lá.

Nele é possível verificar todo o tipo de informações relativas ao país, obtendo assim dados relativos à quantidade de recursos de cada setor económico, tal como o setor industrial presente na seguinte figura.

Figura 21. *Setor Industrial*



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

5.2.6 Develop Debt and Economic Crisis Mechanics

Nesta tarefa, o objetivo seria integrar na simulação económica a dívida pública com aplicação de juros e respetiva ativação de eventos negativos com base em limiares de endividamento. Com isto o jogador é forçado a considerar as consequências das suas decisões económicas se quer manter o jogo a correr.

O país passa a acumular juros sobre a dívida existente no final de cada turno, tendo em conta que historicamente o país atravessa momentos de tensão e como tal o jogador herda, além disso, uma dívida que precisa saber gerir ao longo dos turnos. O cálculo é definido na figura 22.

Figura 22. *Cálculo da dívida nacional*



Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Se a dívida ultrapassar determinados limites, são desencadeadas situações críticas.

Figura 23. Verificação do estado da dívida

```
1 public void CheckDebtThresholds(Country currentCountry, SituationManager situationManager, GameStateContext context)
2 {
3     if (currentCountry.Debt >= 4000)
4     {
5         Debug.Log($"Catastrophic debt level reached in {currentCountry.countryName}: {currentCountry.Debt}. Triggered.");
6         situationManager.CheckForSituations(context.currentYear, context.currentWeek, context);
7     }
8     else if (currentCountry.Debt >= 2000)
9     {
10        Debug.Log($"Critical debt level reached in {currentCountry.countryName}: {currentCountry.Debt}. Negative modifiers applied.");
11        situationManager.CheckForSituations(context.currentYear, context.currentWeek, context);
12    }
13    else
14    {
15        Debug.Log($"Debt level is normal in {currentCountry.countryName}: {currentCountry.Debt}.");
16    }
17 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Na *UI*, ao abrir o dossiê dentro do *office* é possível navegar até à página que contem todos os dados referentes ao país, sendo parte deles fruto do desenvolvimento dos restantes colaboradores, incluindo a dívida e o juro a ela associado, integrados por mim, tal como é possível verificar na figura 24.

Figura 24. Dossiê do office



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

Mediante novos efeitos, como “*Debt*”, “*Expenditure*” e “*InterestRate*”, permitindo assim que políticas e eventos alterem diretamente estes valores. Políticas essas que podem ser “*Austerity*”, “*International Aid*” e “*Restructuring*” que permitem ajudas, porém, a longo prazo poderão ser nocivas.

Figura 25. Exemplo de Política



Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

5.2.7 Implement Legislative Sessions per Turn

Para o sistema legislativo, que decorreu já no sprint 6 e visando ser mais realista, era necessário implementar diversas fases antes de uma dada política ser votada. Como tal, a política proposta passa por uma simulação de votos antes de avançar para uma votação oficial. Esta abordagem permite assim que se preveja o resultado provável da proposta e definir com base nisso o seu rumo, que pode ser aprovação direta, rejeição direta ou então a abertura para alterações.

Esta abordagem segue o seguinte fluxo:

- Semana 1: A política é oficialmente proposta.
- Semana 2: Realiza-se uma simulação de votação:
 - $\geq 80\%$ votos contra e a política é rejeitada;
 - $\geq 50\%$ votos a favor é aprovada de imediato;
 - Entre esses valores surge a possibilidade de emenda, de alterar parâmetros da proposta anterior de modo que possa ser aprovada.
- Semana 3: Em caso de emenda, ocorre a votação final que pode ou não aprovar a política.

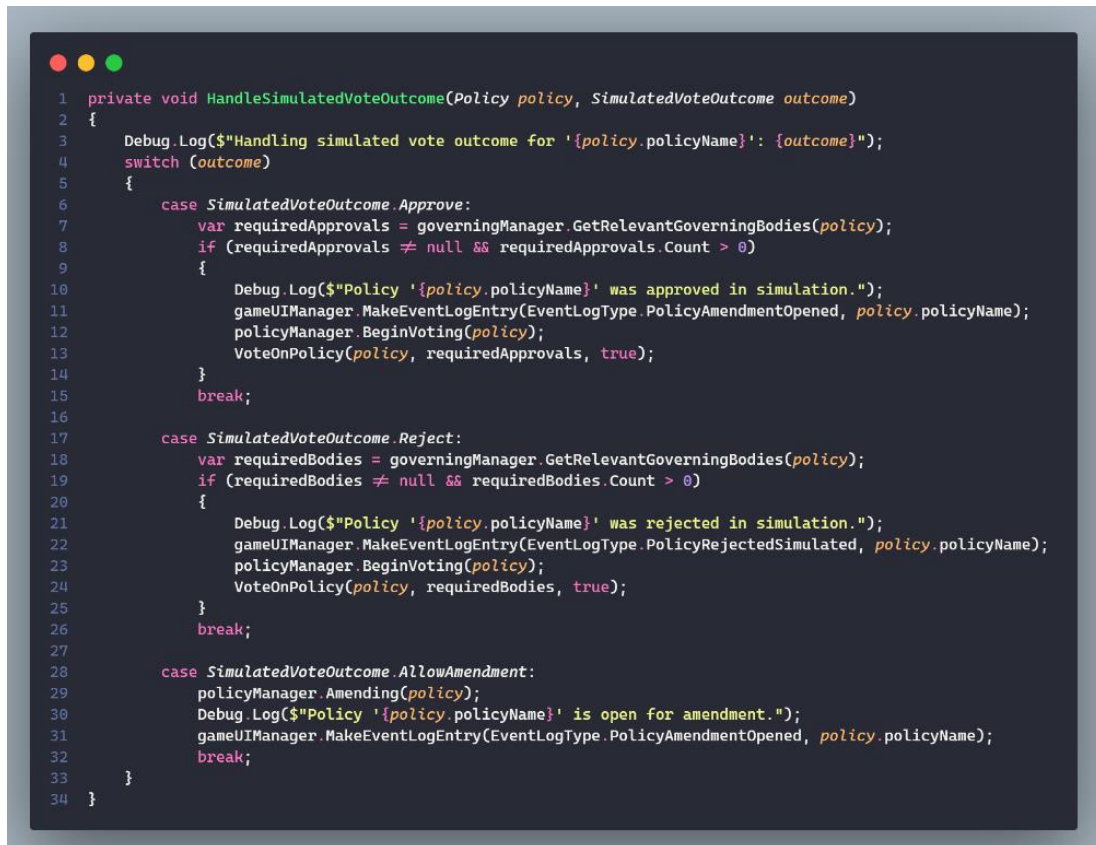
No *script* *GoverningBodyManager.cs* é onde a simulação é concretizada através do método *EvaluateSimulatedVoteOutcome()*. Este realiza a simulação de votos com base nas intenções das sub-fações e determina o desfecho em percentagens e atribui o valor de um *enum* que pode ser “*Approve*”, “*Reject*” ou “*AllowAmendment*”.

Figura 26. Método *EvaluateSimulatedVoteOutcome*

```
1 public SimulatedVoteOutcome EvaluateSimulatedVoteOutcome(Policy policy)
2 {
3     List<Faction> factions = factionManager.factions;
4
5     var (voteResults, _) = SimulateVotes(factions, policy, true);
6     var (totalVotes, votesFor) = CalculateVoteTotals(factions, voteResults);
7
8     float percentageFor = (float)votesFor / totalVotes;
9
10    Debug.Log($"Simulated vote for policy '{policy.policyName}': {votesFor}/
11    {totalVotes} votes for ({percentageFor:P1})");
12
13    if (percentageFor ≥ 0.5f)
14    {
15        return SimulatedVoteOutcome.Approve;
16    }
17    else if (percentageFor < 0.2f)
18    {
19        return SimulatedVoteOutcome.Reject;
20    }
21    else
22    {
23        return SimulatedVoteOutcome.AllowAmendment;
24    }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Após obter o resultado, o método *HandleSimulatedVoteOutcome()* determina o passo seguinte, nomeadamente se a simulação teve resultados positivos, então será enviado para votação final, prevendo assim a aprovação da política. O mesmo para o caso de obter uma simulação com baixa adesão, ou então entra diretamente para *amendment* que permitirá adotar novas medidas relativamente à política proposta.

Figura 27. Método *HandleSimulatedVoteOutcome*


```

1 private void HandleSimulatedVoteOutcome(Policy policy, SimulatedVoteOutcome outcome)
2 {
3     Debug.Log($"Handling simulated vote outcome for '{policy.policyName}': {outcome}");
4     switch (outcome)
5     {
6         case SimulatedVoteOutcome.Approve:
7             var requiredApprovals = governingManager.GetRelevantGoverningBodies(policy);
8             if (requiredApprovals != null && requiredApprovals.Count > 0)
9             {
10                 Debug.Log($"Policy '{policy.policyName}' was approved in simulation.");
11                 gameUIManager.MakeEventLogEntry(EventLogType.PolicyAmendmentOpened, policy.policyName);
12                 policyManager.BeginVoting(policy);
13                 VoteOnPolicy(policy, requiredApprovals, true);
14             }
15             break;
16
17         case SimulatedVoteOutcome.Reject:
18             var requiredBodies = governingManager.GetRelevantGoverningBodies(policy);
19             if (requiredBodies != null && requiredBodies.Count > 0)
20             {
21                 Debug.Log($"Policy '{policy.policyName}' was rejected in simulation.");
22                 gameUIManager.MakeEventLogEntry(EventLogType.PolicyRejectedSimulated, policy.policyName);
23                 policyManager.BeginVoting(policy);
24                 VoteOnPolicy(policy, requiredBodies, true);
25             }
26             break;
27
28         case SimulatedVoteOutcome.AllowAmendment:
29             policyManager.Amending(policy);
30             Debug.Log($"Policy '{policy.policyName}' is open for amendment.");
31             gameUIManager.MakeEventLogEntry(EventLogType.PolicyAmendmentOpened, policy.policyName);
32             break;
33     }
34 }

```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

5.2.8 Implement EventLog Formatting System


De modo que o utilizador consiga visualizar acontecimentos, atualizações de estado de uma política proposta, situações ativas, etc., foi implementado um sistema formatação, armazenamento e visualização de mensagens de Log no painel de interface do jogador. Existia já uma base desenvolvida, contudo, era necessário efetuar melhorias e expandir, e é aí que entra a presente tarefa de refatorar o existente, implementar filtros e facilitar a usabilidade para futuras expansões ao incluir outras mensagens que sejam importantes passar ao utilizador.

A base do sistema assenta em três componentes principais:

- *EventLogEntry*: estrutura que guarda o tipo de evento e a mensagem gerada.
- *EventLogText*: carrega os *templates* a partir de *JSON* e gerar texto com base nos parâmetros recebidos.
- *EventLogUI*: controlador da interface gráfica, exhibe os *logs*, filtra, anima o painel e restaurar entradas salvas.

- Em *EventLogType* são definidos os vários tipos de *logs*, que pode ser expandidos para gerar mensagens e aplicar filtros.

Figura 28. Enum dos tipos de *EventLog*

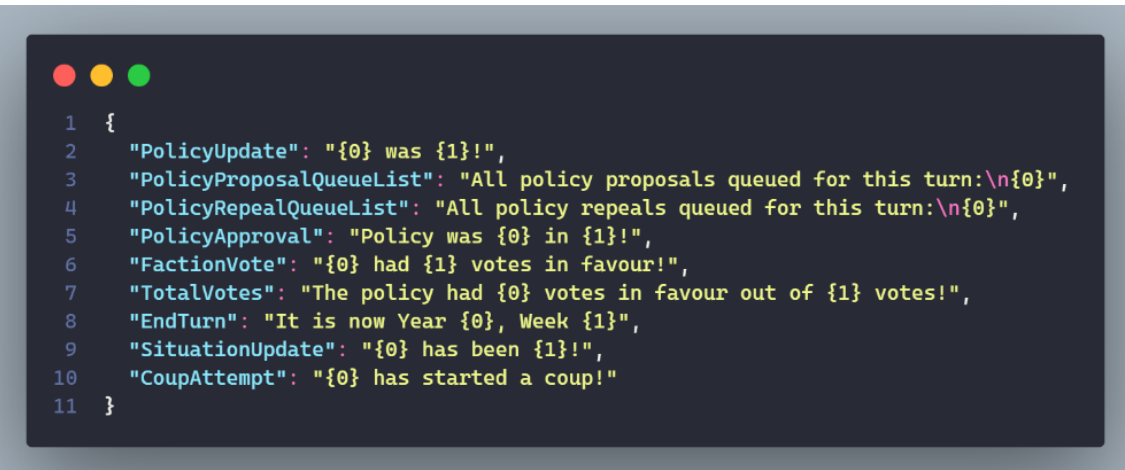


```
1 public enum EventLogType
2 {
3     PolicyApproval,
4     PolicyProposalQueueList,
5     PolicyRepealQueueList,
6     PolicyUpdate,
7     FactionVote,
8     TotalVotes,
9     EndTurn,
10    SituationUpdate,
11    CoupAttempt
12 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

No *eventLogTemplates*, um ficheiro do tipo *JSON* guarda os *templates* de formatação. Permite alterar as mensagens sem tocar em código:

Figura 29. Templates de *EventLog*

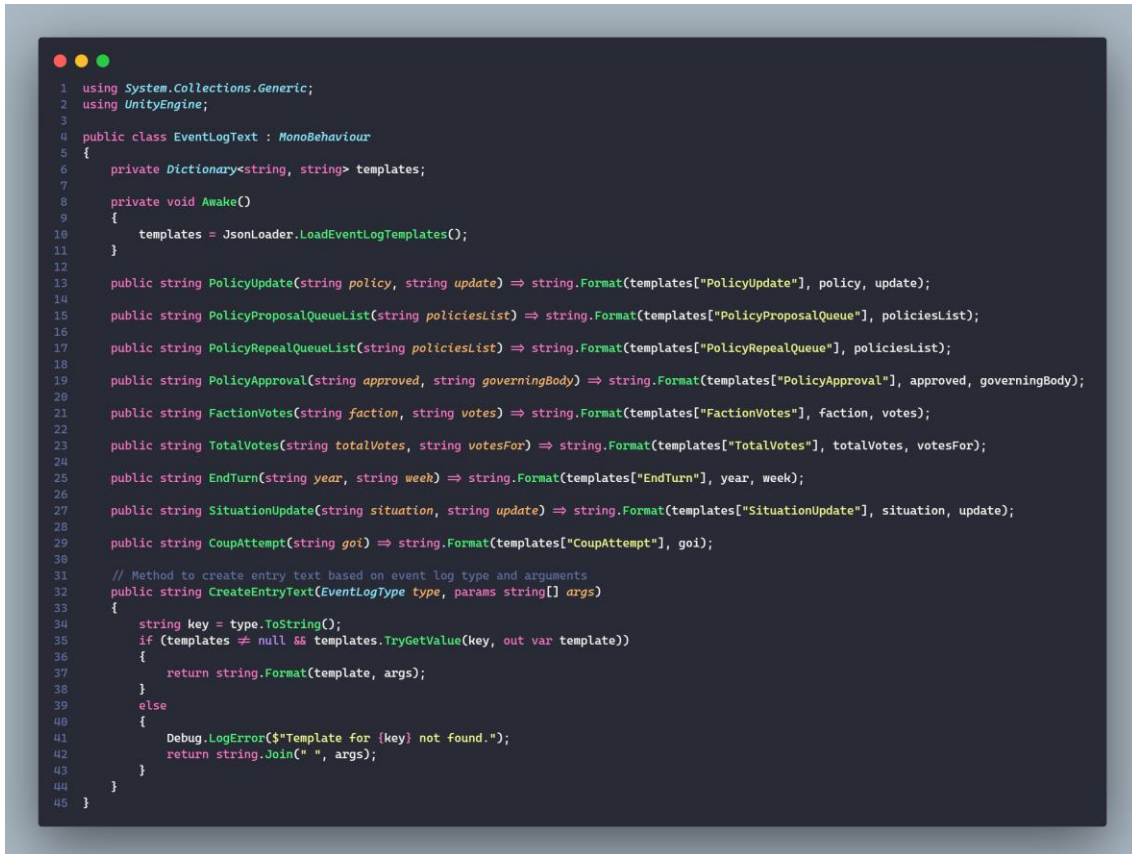


```
1 {
2   "PolicyUpdate": "{0} was {1}!",
3   "PolicyProposalQueueList": "All policy proposals queued for this turn:\n{0}",
4   "PolicyRepealQueueList": "All policy repeals queued for this turn:\n{0}",
5   "PolicyApproval": "Policy was {0} in {1}!",
6   "FactionVote": "{0} had {1} votes in favour!",
7   "TotalVotes": "The policy had {0} votes in favour out of {1} votes!",
8   "EndTurn": "It is now Year {0}, Week {1}",
9   "SituationUpdate": "{0} has been {1}!",
10  "CoupAttempt": "{0} has started a coup!"
11 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Em *EventLogText* carregam-se os *templates* e oferece métodos para cada tipo de evento.

Figura 30. Carregamento dos Logs



```

1  using System.Collections.Generic;
2  using UnityEngine;
3
4  public class EventLogText : MonoBehaviour
5  {
6      private Dictionary<string, string> templates;
7
8      private void Awake()
9      {
10         templates = JsonLoader.LoadEventLogTemplates();
11     }
12
13     public string PolicyUpdate(string policy, string update) => string.Format(templates["PolicyUpdate"], policy, update);
14     public string PolicyProposalQueueList(string policiesList) => string.Format(templates["PolicyProposalQueue"], policiesList);
15     public string PolicyRepealQueueList(string policiesList) => string.Format(templates["PolicyRepealQueue"], policiesList);
16     public string PolicyApproval(string approved, string governingBody) => string.Format(templates["PolicyApproval"], approved, governingBody);
17     public string FactionVotes(string faction, string votes) => string.Format(templates["FactionVotes"], faction, votes);
18     public string TotalVotes(string totalVotes, string votesFor) => string.Format(templates["TotalVotes"], totalVotes, votesFor);
19     public string EndTurn(string year, string week) => string.Format(templates["EndTurn"], year, week);
20     public string SituationUpdate(string situation, string update) => string.Format(templates["SituationUpdate"], situation, update);
21     public string CoupAttempt(string goi) => string.Format(templates["CoupAttempt"], goi);
22
23     // Method to create entry text based on event log type and arguments
24     public string CreateEntryText(EventLogType type, params string[] args)
25     {
26         string key = type.ToString();
27         if (templates != null && templates.TryGetValue(key, out var template))
28         {
29             return string.Format(template, args);
30         }
31         else
32         {
33             Debug.LogError($"Template for {key} not found.");
34             return string.Join(" ", args);
35         }
36     }
37 }

```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

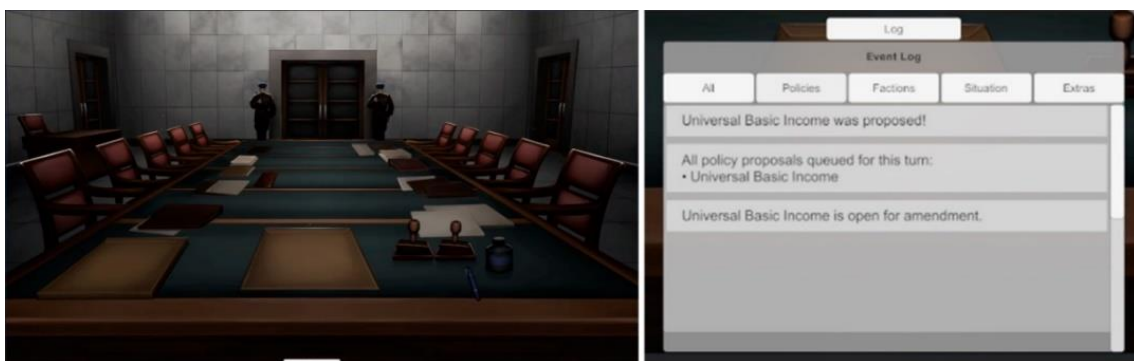
Para a exibição de *logs* e respetiva filtragem, é configurado um painel com abertura ao clicar no botão *Log*, este exhibe todas as mensagens e com a ajuda da configuração de um pequeno painel com diferentes botões de filtragem, permitindo assim que se alterne entre diferentes exibições de modo a ficar mais visível. Filtragem essa que depois é efetuada tal como mostrado na figura 31:

Figura 31. *Filtros de Logs*

```
1 public void ShowAllEntries()
2 {
3     entryObjectList.ForEach(obj => obj.SetActive(true));
4 }
5
6 public void ShowPolicyEntries()
7 {
8     ShowEntriesOfTypes(
9         EventLogType.PolicyApproval,
10        EventLogType.PolicyUpdate,
11        EventLogType.PolicyProposalQueueList,
12        EventLogType.PolicyRepealQueueList
13    );
14 }
15
16 public void ShowFactionEntries()
17 {
18     ShowEntriesOfTypes(EventLogType.FactionVote, EventLogType.TotalVotes);
19 }
20
21 public void ShowSituationEntries()
22 {
23     ShowEntriesOfTypes(EventLogType.SituationUpdate, EventLogType.CoupAttempt);
24 }
25
26 public void ShowExtrasEntries()
27 {
28     ShowEntriesOfTypes(EventLogType.EndTurn);
29 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Posto isto, e como é possível verificar na seguinte figura, os *logs* são exibidos na *UI* do *Politburo* de modo a facilitar a visualização de acontecimentos ao longo dos turnos.

Figura 32. *Logs no Politburo*

Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

5.2.9 Display personalized “Welcome to Office” onboarding note

No decorrer do sprint 7 fora proposta a implementação de um *popup* de boas-vindas que contivesse detalhes importantes como PIB, população, o sentimento de voto das várias sub-fações e a lealdade de *MVD (Ministry of Internal Affairs)*, *MGB (Ministerstvo Gosudarstvennoy Bezopasnosti)* e Forças Armadas. Valores estes que variam dependendo da ideologia escolhida, inspirado na estética de jogos como *Democracy 4*.

Para o desenvolvimento, o painel foi adicionado ao *GameCanvas* da *OfficeScene*. Este contém o título com o nome do Líder, mensagem de boas-vindas e indicadores, como:

- Produto Interno Bruto (PIB);
- População;
- Composição e lealdade do Politburo;
- Lealdade inicial de Grupos de Interesse (*Gol - Group of Interest*).

Foi criado um script *IdeologyWelcomePopupUI* que fica responsável por inicializar e apresentar o painel com base na ideologia escolhida. Atualizar os indicadores económicos e políticos. A apresentação tem como base a estrutura de dados de *IdeologyPathData*, carregada a partir de ficheiros JSON que inclui personagem principal, o nome do caminho, e os valores iniciais de lealdade desenvolvido pelos integrantes da equipa e aproveitado por mim para a minha implementação.

Figura 33. Exemplo de estrutura de dados

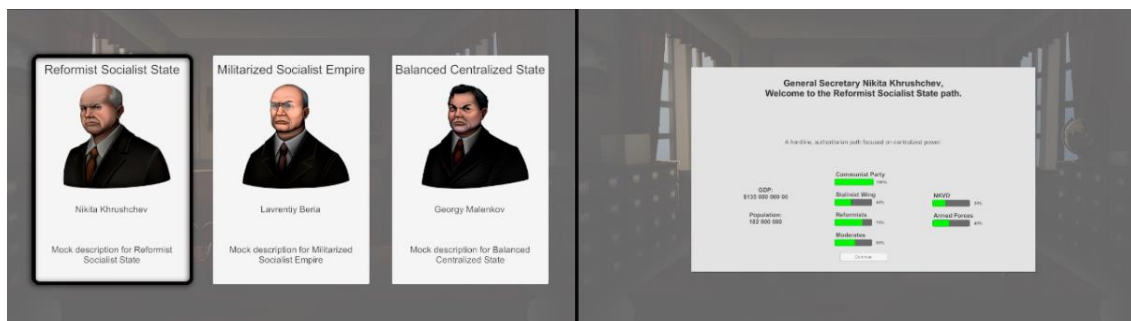


```
1 {
2   "pathKey": "1",
3   "displayName": "Reformist Socialist State",
4   "playerCharacter": "Nikita Khrushchev",
5   "initialFactionLoyalties": {
6     "Communist Party": 100,
7     "Reformists": 70
8   },
9   "initialGoILoyalties": {
10    "Armed Forces of the Union of Soviet Socialist Republics": 40,
11    "NKVD": 30
12  }
13 }
```

Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

A figura anterior ilustra um exemplo do estado Reformista e respetiva estrutura em *countries.json*. Na figura seguinte está apresentado como inicia o jogo, permitindo ao utilizador escolher entre as três ideologias criado pelo colaborador da equipa e, posteriormente, a mensagem de boas-vindas que elaborei.

Figura 34. Escolha ideológica



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

5.3 Documentação Técnica dos Sistemas Principais

No que toca à elaboração da documentação técnica, todo o código implementado ao longo do projeto foi resultado de um esforço colaborativo entre todos os membros da equipa, que me permitiu documentar, de forma estruturada, os principais sistemas desenvolvidos como está representado nos subtópicos seguintes.

5.3.1 Estrutura Central de Países

Para tratar da documentação, as tarefas foram organizadas para serem executadas no decorrer do sprint 3.

Inicialmente foi implementada a classe *Country* como elemento base de cada nação no jogo *State of Affairs*. Classe esta que interliga os restantes componentes, sejam eles políticos ou económicos.

Estrutura e responsabilidade da classe *Country*

A classe *Country* centraliza os dados e sistemas de cada nação, incluindo população, sub-fações, políticas, economia e classes sociais. Atua como ponto de integração entre todos os módulos.

Exemplo de Instância Inicial: URSS (1953)

A primeira instância implementada foi a *URSS* em 1953, com:

- Três sub-fações principais (Conservadores, Reformistas e Moderados), todas no Partido Comunista;
- *Governing bodies* Legislativo, Executivo, Judiciário e Simbólico;
- Um sistema económico centralizado com setores e produção de recursos;
- Dados económicos e políticos alinhados com o contexto histórico do pós-Stalin.

Integração com Outros Sistemas

- O *Country* é chamado diretamente nos principais sistemas do jogo:
- Aplicação de efeitos das políticas e eventos (*EffectManager*);
- Produção económica (*ProduceInAllSectors*);
- Atualização de dívida, despesa e estabilidade política por turno;
- Avaliação de situações emergentes como voto de desconfiança, crise económica ou agitação civil.

5.3.2 Sistema de fações

Foi também desenvolvido o sistema de fações como um dos pilares centrais da simulação política. Cada país é composto por uma ou mais fações políticas, com sub-fações internas que representam ramos ideológicos distintos dentro do mesmo partido dominante.

Estrutura e Responsabilidades

A classe *Faction* representa a entidade principal, contendo:

- Lealdade, influência e poder de voto
- Afinidade com diferentes categorias de políticas
- Uma lista de sub-fações, com comportamentos autónomos

Cada sub-fação tem os seus próprios valores de lealdade, afinidade e influência, permitindo simular disputas internas, mudanças de alinhamento e blocos de poder.

URSS em 1953

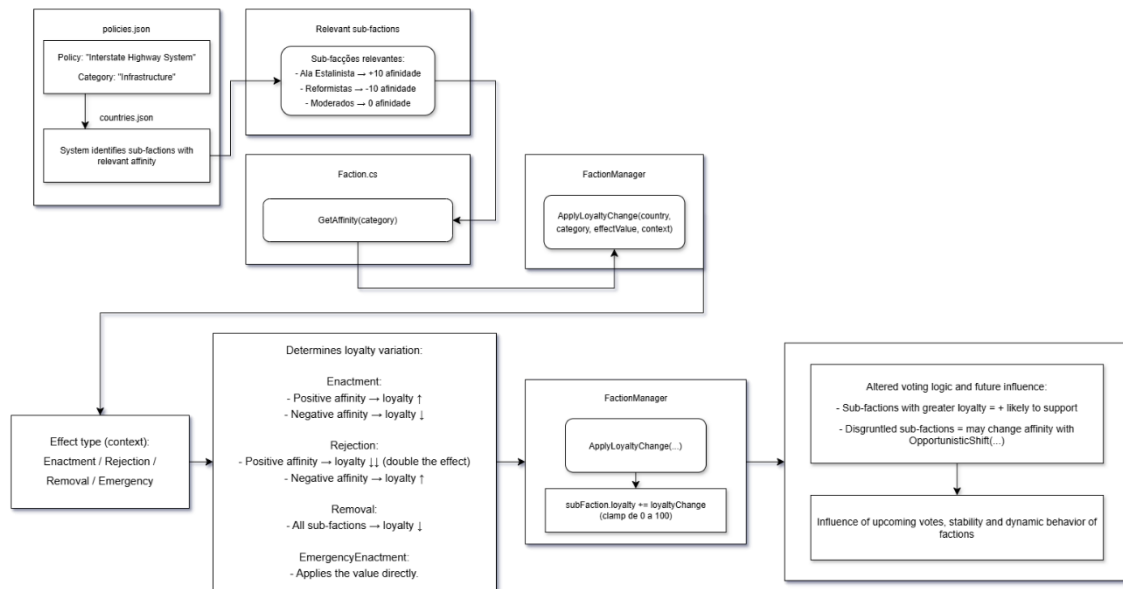
A fação dominante é o Partido Comunista, composto por:

- Conservadores (linha exigente pró-*Stalin* liderada por *Lavrentiy Beria*);
- Reformistas (linha de abertura liderada por *Nikita Khrushchev*);
- Moderados (força de equilíbrio liderada por *Georgiy Malenkov*).

Lógica de Funcionamento

- Avaliam cada política com base nas suas afinidades ideológicas;
- Votam em sessões legislativas com base no seu poder de voto ajustado;
- Alteram a sua lealdade conforme a aprovação ou rejeição de políticas;
- Reagem a eventos do jogo e podem mudar de posição estratégica.

Figura 35. Fluxo de fações



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

Papel nos Corpos Governamentais

As sub-fações compõem os corpos governamentais, como o *Politburo* ou Conselho de Ministros, sendo responsáveis pelas decisões políticas em cada turno. O equilíbrio entre sub-fações define o desfecho de cada votação.

5.3.3 Sistema de Políticas

O sistema de políticas permite ao jogador propor, implementar ou revogar políticas que afetam diretamente a estabilidade, economia e estrutura política do país. O presente sistema foi desenvolvido com uma arquitetura modular, seguindo um *pipeline* claro de criação, aplicação e evolução de cada política ao longo da campanha.

Fases de Vida de uma Política

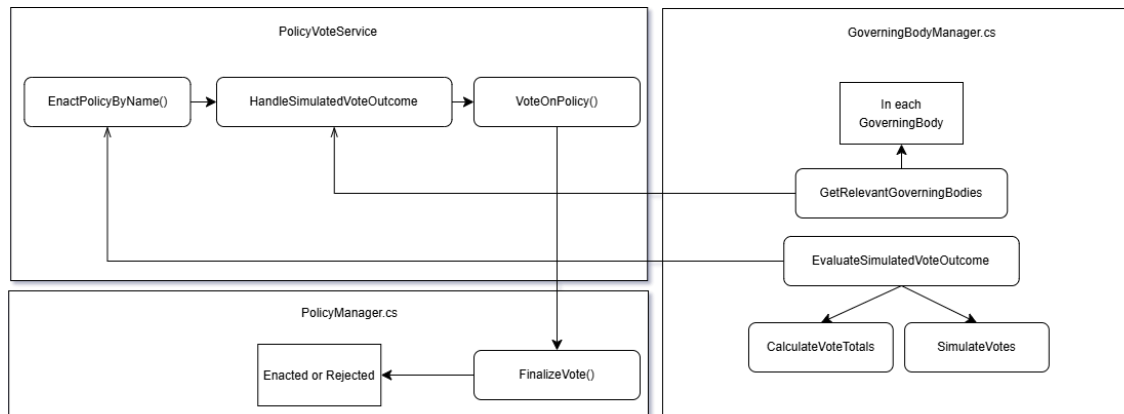
O ciclo de uma política passa pelas seguintes fases:

- Proposta - O jogador escolhe uma política a propor.

- Sessão Legislativa - A política é debatida por sub-facções efetuando uma simulação de votos para definir se este é aprovado, rejeitado ou permite emenda.
- Votação - É executada nos corpos governamentais definidos.
- Resultado - A política é aprovada ou rejeitada.

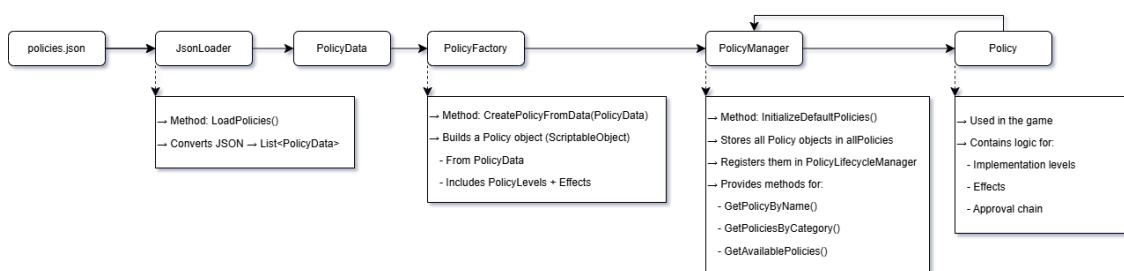
Os tipos de efeitos, estrutura e respetivo funcionamento completo dos efeitos está detalhado na secção 5.3.5 (Sistema Modular de Efeitos).

Figura 36. Fluxo de sessão legislativa



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

Figura 37. Fluxo de políticas



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

Impacto Estratégico no Jogo

Este sistema permite que a cada política sejam desencadeadas consequências imediatas ou prolongadas, afetando:

- Lealdade de sub-facções e grupos de interesse
- Sentimento das classes sociais
- Estabilidade política e económica
- Eventos dinâmicos ou situações emergentes

5.3.4 Sistema de Situações

O Sistema de Situações permite introduzir eventos dinâmicos, dilemas e crises no decorrer da campanha. Estas situações podem afetar diretamente o estado económico, político e social do país, forçando o jogador a adotar decisões estratégicas com consequências a curto ou longo prazo.

Tipos de Situação

- Evento: Situação automática com efeitos diretos.
- Dilema: Apresenta escolhas ao jogador, com consequências associadas.
- Crise: Situação de alta gravidade com impacto sistémico.

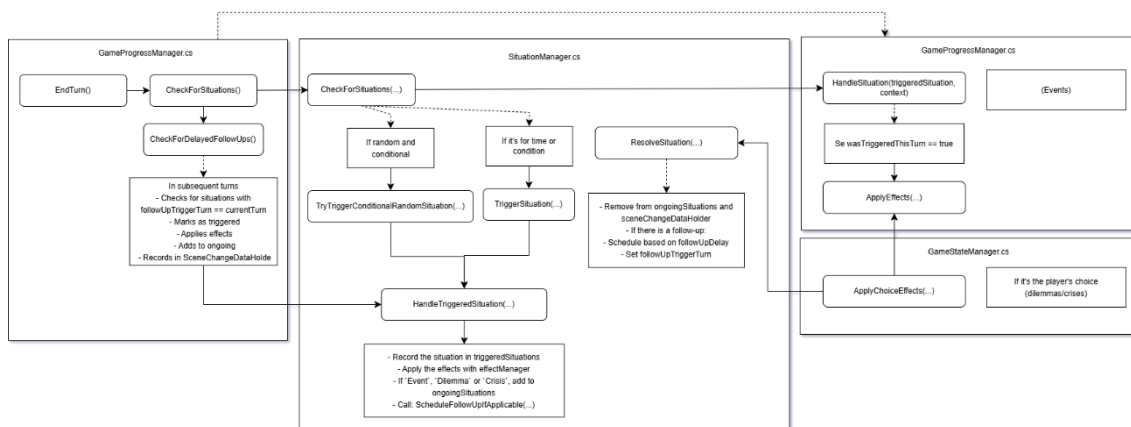
Condições de Ativação

Cada situação pode depender de:

- Ano e semana atual do jogo;
- Probabilidade de ocorrência (0,0 a 1,0);
- Condições lógicas personalizadas.

Ciclo de Vida da Situação

Figura 38. Fluxo de situações



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

Exemplo Prático

Uma situação pode ser definida com lógica composta como:

Figura 39. *Lógica de ativação de situações*



Nota. Elaboração própria com capturas de ecrã do código no Visual Studio Code.

Se for ativada, pode aplicar efeitos como:

- Redução do PIB em -2 durante 5 turnos
- Diminuição da estabilidade política
- Alterações na lealdade de sub-fações associadas

5.3.5 Sistema Modular de Efeitos

Foi concebido para aplicar alterações ao estado do jogo de forma modular, extensível e reativa, permitindo que decisões políticas, eventos ou dilemas tenham consequências mecânicas claras e sustentadas ao longo do tempo.

Objetivo e Aplicações

Este sistema permite modificar variáveis fundamentais do jogo como:

- Produto Interno Bruto (PIB)
- Sentimento das classes sociais
- Lealdade das sub-fações políticas
- Estabilidade política
- Influência de grupos de interesse

Esses efeitos são desencadeados por diferentes sistemas do jogo, nomeadamente:

- Políticas (quando aprovadas, rejeitadas ou revogadas)
- Situações (eventos, dilemas ou crises)
- Decisões emergenciais ou eventos sistémicos

Tipos de Efeito Suportados

O sistema suporta uma variedade de efeitos, definidos no *enum EffectType*:

- *GDP (Gross Domestic Product)* - altera o Produto Interno Bruto
- *VoterSentiment* - afeta a opinião de uma classe social
- *FactionLoyalty* - altera a lealdade das sub-fações (por categoria de política)
- *Go/Loyalty* - afeta a lealdade de grupos de interesse (ex: Forças Armadas, MVD, MGB)
- *PoliticalStability* - modifica a estabilidade política do país

Cada efeito pode ter um grupo-alvo específico (ex: “Classe Média”, “Economia”, “Any”) e pode ser imediato ou prolongado.

Exemplo de Efeito (JSON)

Figura 40. Exemplo de efeitos



Nota. Elaboração própria.

Este efeito representa uma penalização imediata de -40 pontos no Sentimento da Classe Trabalhadora.

Integração com o Ciclo de Jogo

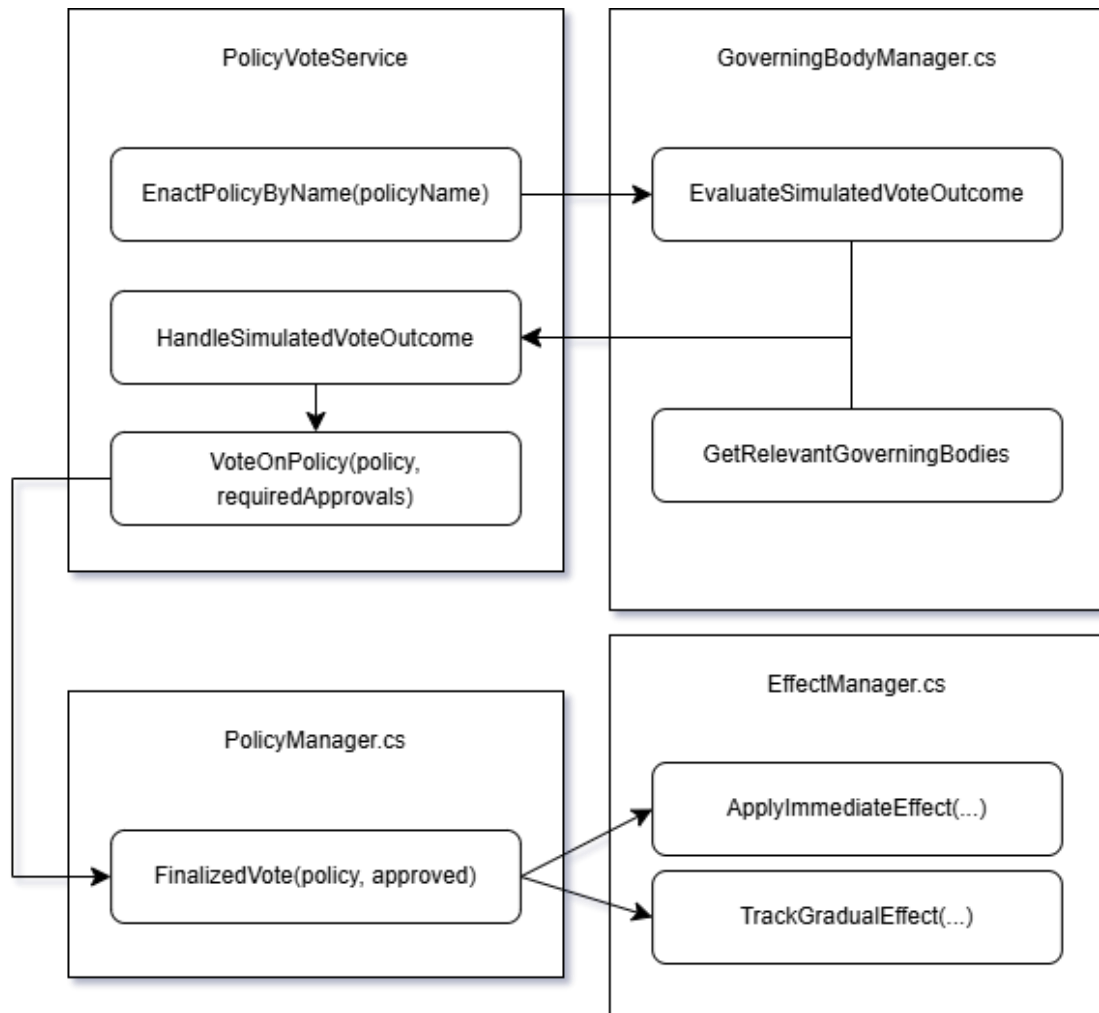
- Durante aprovação de políticas: aplica efeitos definidos em *EnactmentEffects*.
- Em caso de rejeição: efeitos definidos em *EnactmentRejectionEffects*.

- Durante situações e dilemas: efeitos são aplicados automaticamente ou com base nas escolhas do jogador.
- Durante eventos de emergência: efeitos alternativos podem ser ativados (ex: *emergencyEnactmentEffects*).

Fluxo de ativação de efeitos

Políticas

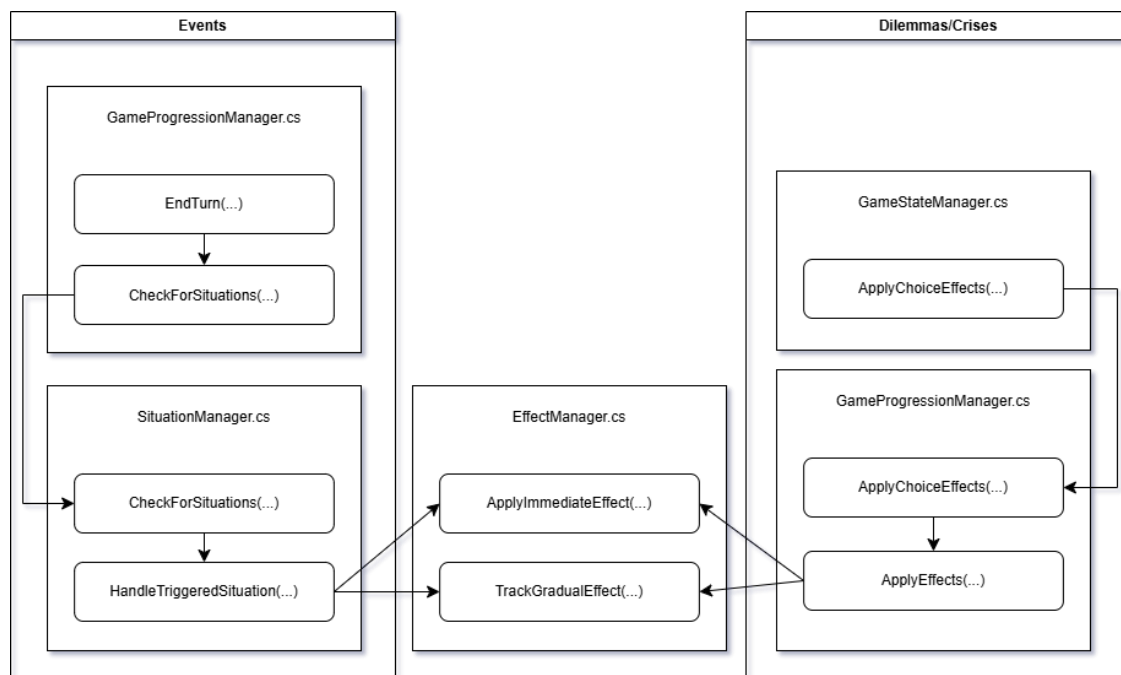
Figura 41. Aplicação de efeitos em políticas



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

Situações (eventos, dilemas, crises)

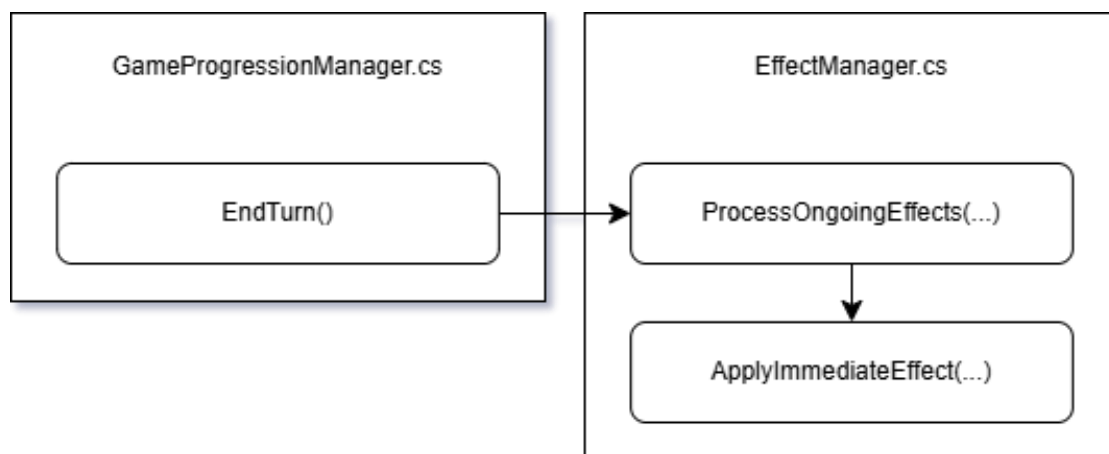
Figura 42. Aplicação de efeitos em situações



Nota. Elaboração própria.

Efeitos Graduais

Figura 43. Aplicação de efeitos graduais



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

5.3.6 Sistema de Classes Sociais (Social Strata)

O sistema de Classes Sociais pretende representar diferentes grupos da população, cada um com posições distintas relativamente ao estado político do país. Estas classes sociais influenciam diretamente o rumo do jogo, afetando o apoio às políticas, a estabilidade social e a reação a eventos.

Objetivos e Papel no Jogo

- Simular a diversidade de opinião pública e a pressão social sobre o governo.
- Tornar as decisões políticas mais significativas, ao afetar diferentes grupos diferenciadamente.
- Criar dinâmicas onde baixa satisfação pode desencadear eventos sociais como agitação civil ou perda de apoio político.

Estrutura Base

Cada país possui uma lista de classes sociais, definidas inicialmente em *countries.json*. As principais classes são:

- Classe Alta (*Wealthy*);
- Classe Média (*Middle Class*);
- Classe Trabalhadora (*Working Class*).
- Classe Militar (*Military*)

Cada uma das classes conta sempre com nome, sentimento relativamente ao governo atual (de 0 a 100) e uma lista de preferência políticas.

Funcionamento

Durante o jogo:

- Políticas aprovadas que correspondem às preferências de uma classe aumentam o seu *sentiment*.
- Políticas rejeitadas, impopulares ou removidas reduzem o *sentiment*.
- Situações dinâmicas e eventos podem também alterar o apoio de cada grupo.

Figura 44. Fluxo dos estados sociais



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

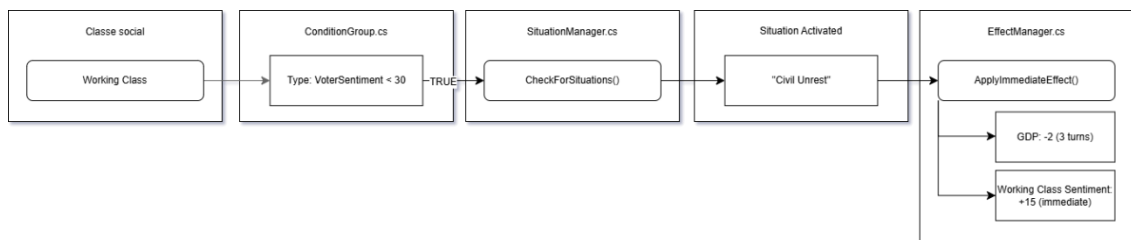
Impacto Narrativo e Estratégico

A insatisfação de uma classe social pode gerar:

- Protestos ou agitação civil;
- Perda de apoio político;
- Reações de *factions* que procuram capitalizar esse descontentamento;

Este sistema obriga o jogador a considerar o impacto de cada decisão e a equilibrar políticas populistas com decisões estratégicas.

Figura 45. Aplicação de efeitos em estados sociais



Nota. Elaboração própria com recurso à ferramenta draw.io (<https://app.diagrams.net>).

5.4 Refatoração e Otimização de Código

No Sprint 4, surgiram tarefas de refatoração de algumas entidades para organizar o código e permitir maior escalabilidade e manutenção no sistema político do jogo, dado que o código-base do projeto foi desenvolvido colaborativamente entre os vários elementos da equipa

5.4.1 Refactor GoverningBody Structure

Um dos módulos mais relevantes a ser reestruturado foi o *GoverningBody*, responsável por representar os diferentes órgãos de governo no processo legislativo.

Para isso, foi adotada uma arquitetura modular organizada da seguinte forma:

- *GoverningBody.cs*: contém a lógica de tempo de execução de um corpo governativo (ex.: *Politburo*, Supremo Soviético). Inclui funcionalidades como lógica de votação, composição de sub-fações, limiares de aprovação e resultados políticos.
- *GoverningBodyData.cs*: estrutura serializável em *JSON*, usada para definir o tipo de órgão, o seu nome, as regras de aprovação e o tempo necessário para a votação. Esta separação permite carregar dados de configuração facilmente e de forma reutilizável.
- *GoverningBodyFactory.cs*: responsável por instanciar objetos *GoverningBody* a partir dos dados definidos em *GoverningBodyData*. A sua função é converter configurações em entidades jogáveis.
- *GoverningBodyManager.cs*: controla todos os órgãos governativos do jogo. Gere o encaminhamento de propostas, monitoriza o progresso de aprovação e garante a coordenação entre diferentes corpos legislativos.

5.4.2 Refactor Effect Structure

Foi, também, iniciada a refatoração do módulo *Effect*, ainda no sprint 4, de modo que este ficasse organizado. A principal motivação que levou a esta tarefa foi a necessidade de tornar o código mais legível, escalável e consistente, à semelhança do que aconteceu com *GoverningBody*.

Numa fase inicial, a estrutura e a respetiva implementação seguiram as boas práticas. Contudo, no sprint 5, que foi especialmente dedicada à refatoração dos módulos, surgiu documentação interna oficial que estabelecia uma estrutura padrão a ser seguida por todos os módulos do jogo. Assim, o trabalho foi revisto e ajustado para cumprir na íntegra essa especificação.

- *Effect.cs*: representa efeitos ativos em tempo de execução. Pode conter lógica de execução e duração.
- *EffectData.cs*: estrutura serializável em *JSON*, usada para definir o tipo de efeito, valor, duração, etc.
- *EffectFactory.cs*: Constrói objetos *Effect* a partir de *EffectData*.

- *EffectManager.cs*: aplica os efeitos ao estado do jogo.
- *EffectPolicyContext.cs*: *Enum* usado para identificar o momento em que o efeito ocorre.
- *EffectType.cs*: *Enum* que define os tipos possíveis de efeitos.

Conforme a documentação, foi criada uma pasta “*Services*”, em que foram extraídos comportamentos complexos dos efeitos que dependiam de múltiplas entidades de jogo. Entre elas estão:

- *FactionEffectService.cs*: aplica efeitos de lealdade a sub-fações com base nas afinidades.
- *GoIEffectService.cs*: aplica efeitos de lealdade a grupos de interesse (*GoI*) por categoria política.
- *GDPEffectService.cs*: atualiza o valor do PIB do país.
- *ProductionEffectService.cs*: altera a capacidade de produção de um setor económico ou *subsector*.

Esta separação em serviços permite que sejam testáveis isoladamente, visto que assim não dependem diretamente do *EffectManager*.

5.4.3 Implement Faction System Architecture

Outro *refactor* efetuado no sprint 5 foi no módulo *Faction*, visando alinhar a sua estrutura com a arquitetura modular, tal como descrito na documentação oficial.

Assim como anteriormente mencionado nos módulos *Effect* e *GoverningBody*, esta abordagem permite a separação clara entre dados, lógica de execução, instanciação e gestão global de entidades.

- *Faction.cs*: representa uma facção em tempo de execução. Armazena lealdade, *subFactions*, etc.
- *FactionData.cs*: estrutura serializável usada para carregar configurações a partir de ficheiros *JSON*.
- *FactionFactory.cs*: Constrói objetos *Faction* com base em *FactionData*.
- *FactionManager.cs*: responsável por gerir todas as sub-fações, inicializar, atualizar e expor dados.

5.5 Testes e Validação

5.5.1 Implementação de Testes End-to-End (E2E)

Na fase final do *Vertical Slice*, realizamos testes *End-to-End (E2E)* de modo a verificar os diferentes caminhos possíveis no jogo, cujo objetivo era garantir que todos os sistemas funcionavam conforme o idealizado, as diferentes sub-fações reagiam corretamente às lógicas do jogador e as condições de vitória e derrota eram justas e coerentes mediante as ações adotadas no decorrer dos testes.

5.5.2 Teste E2E: Caminho Reformista

Testou-se a escolha ideológica Estado Socialista Reformista, em que se assume o controlo de *Nikita Khrushchev* com foco em reformas internas e liberalização do regime. Foram validadas mecânicas como:

1. Inicialização do cenário

Com a seleção da ideologia do Estado Reformista;

2. Proposta de Políticas

Testou-se a submissão de uma política de cariz liberal. Como não havia certeza de qual política seria liberal, algo que poderia ser uma dificuldade presente no utilizador final, ficou anotado como possível melhoria para futuro. Contudo, foi utilizada uma política “*Enforce Bodycams on Police Officers*” visto ser a que mais se enquadrava com a promoção de transparência, direitos civis e responsabilidades das forças armadas. Posto isso, as fações reagiram consoante as suas afinidades, em que os Reformistas demonstraram o apoio desejado, enquanto os Conservadores opuseram-se ativamente. No que toca à lógica de votação, esta funcionou como esperado, prevendo que o Politburo necessitasse de 2/3 para aprovado, ou seja, maioria qualificada o que não aconteceu e a política entrou em fase de emenda antes de ser definitivamente rejeitada, o que permitiu validar esse fluxo e os respetivos efeitos negativos provenientes da política não aprovada.

3. Fase de emenda

Após a política não ter sido aceite, nem aprovada na primeira votação, esta entrou numa fase de negociação para influenciar o voto dos Moderados. Observou-se que essas modificações provocavam alterações nas intenções de voto e indicadores de lealdade, conforme previsto.

4. Crise e reação

Embora o cenário estivesse configurado para, eventualmente, gerar a crise da “Agitação no Bloco de Leste”, esta não se manifestou durante os testes, possivelmente devido ao insucesso na aprovação de políticas reformistas suficientes. Ainda assim, o sistema demonstrou estar funcional nos critérios que antecedem a sua ativação.

5. Feedback de jogo

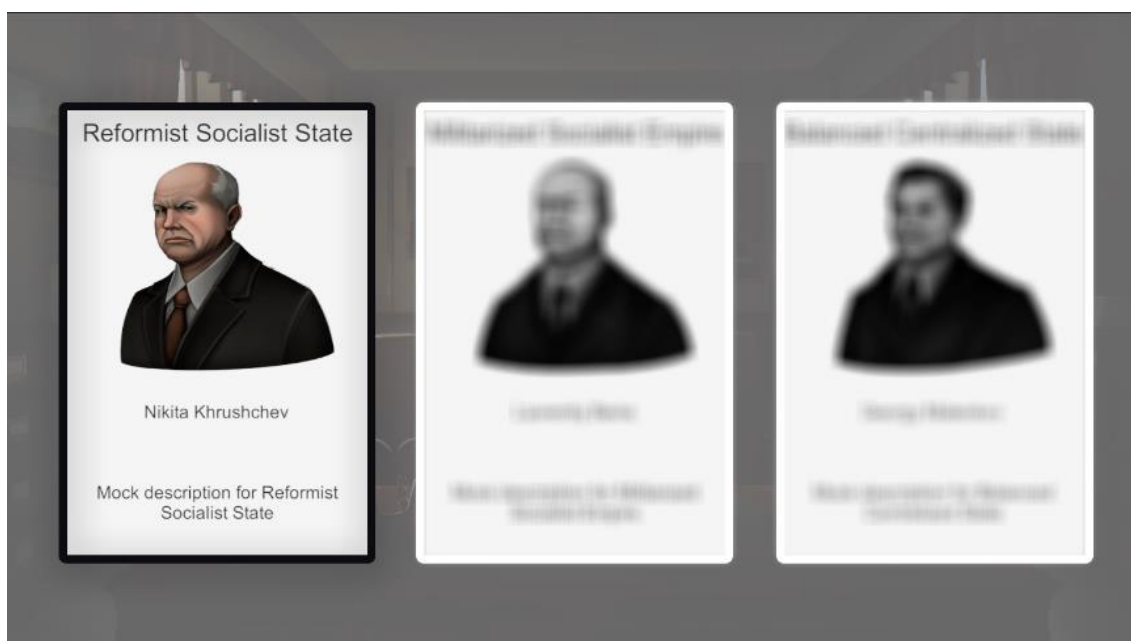
O relatório de fim de turno apresentou corretamente alterações na lealdade das facções, bem como variações nos indicadores económicos e sociais (PIB, dívida, receitas, etc.).

6. Validação final

Não foi possível atingir um estado final conclusivo, seja ele de vitória ou derrota nos testes realizados, visto que até então não havia como concluir o ciclo de jogo.

Este primeiro teste permitiu validar o fluxo principal do caminho reformista, em particular na relação entre as facções e o sistema de votação. Pese embora alguns erros tenham sido notados e nem todos os eventos planeados foram despoletados, os mecanismos centrais comprovaram-se funcionais tanto na lógica de lealdade, os efeitos aplicados e o comportamento do sistema em tempo real.

Figura 46. *Caminho Reformista*



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

Tabela 1. Caso de teste Caminho Reformista

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	STATEOFAFFAIRS-TCS-REF-001
Objetivo:	Verificar o correto funcionamento do jogo ao seguir o caminho ideológico “Estado Reformista”. Validar o comportamento das sub-fações, ciclo de votação, lógica de efeitos, reações a políticas liberais e a possibilidade de gerar a crise “Agitação no Bloco de Leste”.
Autor(es)	Paulo Guimarães
Especificação de Entradas a) Selecionar novo jogo b) Escolher URSS como país c) Selecionar a ideologia “Estado Reformista” d) Propor política de carácter liberal (“ <i>Enforce Bodycams on Police Officers</i> ”) e) Avançar turnos até à possível geração de crise	
Especificação de Saídas a) Reformistas demonstram apoio inicial b) Política entra na agenda e vai à votação c) Política é rejeitada após fase de emenda d) Efeitos negativos aplicados corretamente e) Relatório de fim de turno apresenta alterações nos indicadores f) Jogo permanece estável (sem colapsar), mesmo após falha da política	
Outros Teste realizado em <i>build</i> de QA (<i>Quality Assurance</i>) sem ferramentas de <i>debug</i> Jogo testado até 4/5 turnos completos	
Dependências Política liberal disponível no início Mecânicas de votação e negociação ativas <i>Triggers</i> de crise ativados após políticas reformistas	

Nota. Elaboração própria.

5.5.3 Teste E2E: Caminho Conservador

Para o segundo teste, a ideologia pretendida era o estado Conservador, seguindo as pisadas de *Lavrentiy Beria*, alinhado com as forças armadas num caminho mais autoritário e focado na dominação militar e controlo interno. Para validar o percurso foram efetuados os seguintes:

1. Inicialização do cenário

À semelhança do teste anterior, iniciou-se um novo jogo e definiu-se o caminho conservador como ideologia pretendida. Verificou-se que tínhamos maior lealdade de Conservadores e grupos militares. Por outro lado, os Reformistas mostravam-se fortemente contra, algo já esperado.

2. Proposta de Políticas

Para a proposta política, foi escolhida a *“Increase Military Spending”*, por se enquadrar com o espírito autoritário do caminho. Contudo, e visto que mesmo com o apoio das sub-fações alinhadas, a proposta não fora aprovada, devido à falta de maioria qualificada no Politburo. Como tal, a proposta entrou em fase de negociação. Em seguida, a mesma fora reprovada e provocou consequentemente a queda da lealdade em algumas das sub-fações e grupos de interesse, o que demonstrou que o sistema efetivamente reagia bem à falha estratégicas, porém, com a impossibilidade de aprovar políticas.

3. Fase de emenda

Durante esta fase, testou-se a negociação da proposta, o que alterou alguns valores, nomeadamente na intenção de voto positivamente, confirmando o correto funcionamento da mecânica de influência faccional.

4. Crise e reação

Apesar de o jogo estar preparado para gerar a crise “Provocação da NATO”, relacionada com a escalada militar nas fronteiras, esta não chegou a ser ativada durante os testes. Ainda assim, foram ativadas outras crises menores e verificou-se que a lógica de surgimento e de consequências estava funcional e respetiva ativação de efeitos.

5. Feedback de jogo

O relatório de fim de turno mostrou corretamente os impactos das decisões, tanto a nível da lealdade das facções como nas métricas económicas como PIB, dívida pública, receitas. O rastreador de objetivos estratégicos não mostrou corretamente o progresso relativamente à vitória ou colapso do regime.

6. Validação final

Neste teste, também não foi possível alcançar um estado final conclusivo. A ausência da crise principal, eventualmente as poucas políticas votadas e não aprovadas impediram a observação de um ecrã de vitória ou derrota. Ainda assim, as validações

parciais confirmaram o alinhamento ideológico, as reações das fações e os efeitos aplicados após cada votação ou evento.

Este segundo teste permitiu verificar o rumo adotado no caminho conservador, sobretudo no que toca à dinâmica entre políticas autoritárias, reações internas e possíveis conflitos externos. Apesar de nem todas as condições terem sido totalmente despoletadas, o sistema comportou-se de forma estável e alinhada com a ideologia selecionada.

Figura 47. *Caminho Conservador*



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

Tabela 2. *Caso de teste Caminho Conservador*

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	STATEOFAFFAIRS-TCS-CON-002
Objetivo:	Verificar o funcionamento do jogo ao selecionar a ideologia “Estado Conservador”, validando as reações das fações militarizadas, o processo de votação de políticas autoritárias, a fase de emenda e a preparação para crises externas como a “Provocação da NATO”.
Autor(es)	Paulo Guimarães
Especificação de Entradas a) Iniciar novo jogo b) Escolher URSS como país	

- c) Selecionar a ideologia “Estado Conservador”
- d) Propor a política “*Increase Military Spending*”
- e) Tentar negociar apoio
- f) Avançar turno até à possível crise da NATO

Especificação de Saídas

- a) Conservadores e facções militares demonstram apoio inicial
- b) Reformistas mostram oposição imediata
- c) Política rejeitada na primeira votação, entrando em fase de emenda
- d) Política novamente rejeitada após negociação, aplicando efeitos negativos
- e) Feedback no relatório de turno refletiu corretamente os impactos
- f) Crises menores foram ativadas, mas a crise principal (NATO) não se manifestou

Outros

Teste realizado numa *build* de QA sem uso de comandos de *debug*
Política proposta nas primeiras rondas
Feedback faccional observado no relatório de turno

Dependências

Política autoritária disponível no início
Lógica de facções e votação ativada
Mecânica de emenda e negociação funcional
Triggers de crise NATO ativos (não acionados neste teste)

Nota. Elaboração própria.

5.5.4 Teste E2E: Caminho Equilibrado

Para o terceiro e último teste, foi selecionada uma ideologia Equilibrada, onde o jogador segue as ideias do grupo liderado por *Georgiy Malenkov*, num papel mais moderado, para estabilizar o país sem enveredar por reformas radicais nem repressão externa. Inicialmente terá de ter o apoio por parte dos Moderados e oposição equilibrada das restantes sub-facções.

Posto isto, foram averiguadas as seguintes etapas:

1. Inicialização do cenário

Após iniciar um novo jogo, selecionou-se a ideologia Equilibrada, confirmando-se que *Georgiy Malenkov* era o líder da ideologia jogável, como tal esperava-se que os objetivos estratégicos associados seriam estabilidade política e desenvolvimento económico. As sub-facções Moderadas mostraram alinhamento inicial, enquanto Conservadores e Reformistas mantinham uma postura neutra ou ligeiramente cautelosa, o que seria idealmente previsto.

2. Proposta de Políticas

Foi proposta a política “*Interstate Highway System*”, por ser vista como uma medida neutra de investimento em infraestruturas e desenvolvimento. No entanto, observou-se alguma oposição inesperada de determinadas facções, com a política a ser rejeitada. A rejeição teve impacto nos níveis de lealdade onde os Reformistas demonstraram maior simpatia, os Moderados mantiveram uma posição neutra, e os Conservadores reagiram negativamente. A lógica de votação funcionou corretamente, com a regra da maioria qualificada no Politburo a ser corretamente aplicada.

3. Fase de emenda

Testou-se também a possibilidade de negociação para alterar intenções de voto. Tal como nos testes anteriores, foi possível verificar a atualização dos medidores de lealdade com base nas ações realizadas, validando o funcionamento da lógica de influência.

4. Crise e reação

Este caminho deveria gerar eventualmente a crise “Luta pelo Poder” no *Politburo*, a qual, infelizmente, não foi desencadeada durante este teste. Ainda assim, o sistema demonstrou capacidade de resposta nas fases que precedem esse momento crítico, além de despoletar outras situações nocivas para o estado do país.

5. Feedback de jogo

O relatório de fim de turno mostrou alterações nos valores das facções, em especial nas reações à rejeição da proposta. Indicadores económicos como PIB, dívida e receitas também foram atualizados corretamente. No entanto, o rastreador de objetivos estratégicos não indicou progresso claro, o que poderá estar relacionado com a ausência de medidas aprovadas durante o teste.

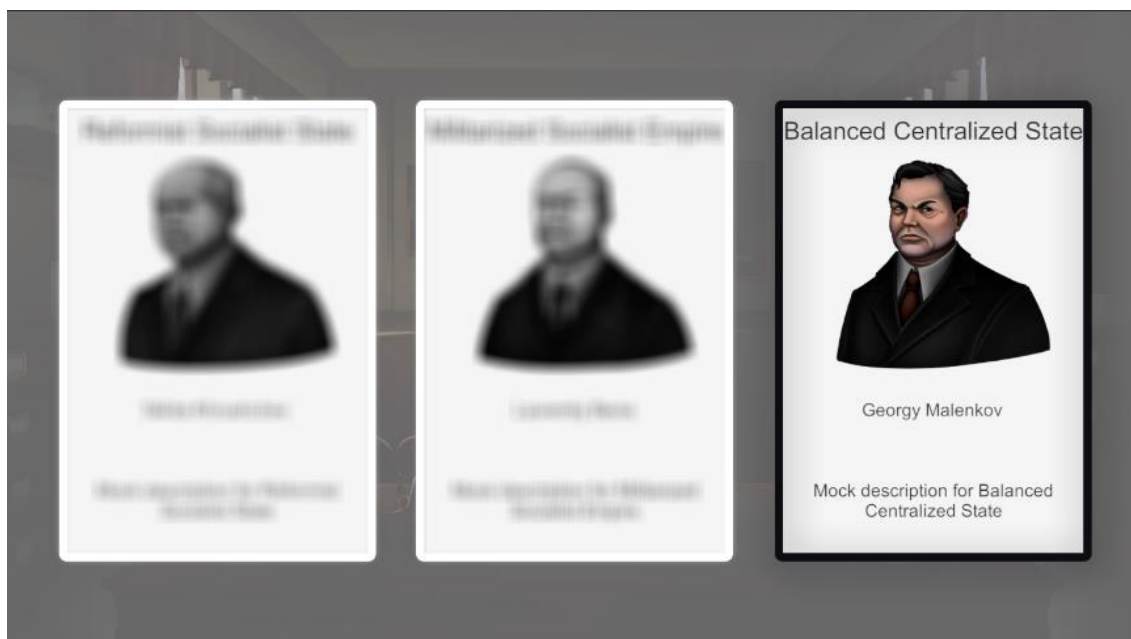
6. Validação final

Tal como nos testes anteriores, não foi possível atingir um estado final de vitória ou derrota. Contudo, foram validadas partes fundamentais da lógica do caminho equilibrado, nomeadamente o comportamento cauteloso das facções extremas, a coerência na aplicação das regras de votação e os efeitos contextuais após a rejeição de políticas.

Este teste permitiu reforçar que, mesmo sem alcançar o estado final ideal, o caminho equilibrado apresenta em termos de lógica e comportamento dinâmico um bom funcionamento. A presença de pequenos desequilíbrios no alinhamento das facções

perante políticas aparentemente moderadas poderá ser um ponto a rever numa fase futura de afinação de valores e narrativa.

Figura 48. *Caminho Equilibrado*



Nota. Elaboração própria com capturas de ecrã do jogo em desenvolvimento.

Tabela 3. *Caso de teste Caminho Equilibrado*

ESPECIFICAÇÃO DE CASO DE TESTE	
Identificador:	STATEOFAFFAIRS-TCS-EQL-003
Objetivo:	Verificar o comportamento do jogo ao seleccionar a ideologia “Estado Equilibrado”, garantindo o alinhamento inicial dos Moderados, a oposição cautelosa das facções extremas, e a possibilidade de ativação da crise interna “Luta pelo Poder no Politburo”. Validar ainda a lógica de votação, negociação, e os efeitos das decisões.
Autor(es)	Paulo Guimarães
Especificação de Entradas a) Iniciar novo jogo b) Seleccionar URSS como país c) Escolher ideologia “Estado Centralizado Equilibrado” d) Propor a política “Interstate Highway System” e) Avaliar resultado e negociar apoio f) Avançar turnos e verificar surgimento de crises	
Especificação de Saídas	

- a) Georgiy Malenkov como líder e moderação nas fações
- b) Política rejeitada e afetações esperadas nas sub-fações (Reformistas, Conservadores, Moderados)
- c) Atualização de lealdade e intenção de voto após negociação
- d) Crise “Luta pelo Poder” não ativada, mas outras crises menores surgem
- e) *Feedback* de fim de turno correto nos indicadores políticos e económicos

Outros

Teste realizado com *build* de QA sem ferramentas de desenvolvimento
 Política proposta nas primeiras rondas
 Sistema de negociação e *feedback* validado
 Objetivos estratégicos não refletiram progresso

Dependências

Sistema de votação e negociação ativo
 Política moderada disponível
 Mecânica de lealdade e rastreio de objetivos ativada
Triggers de crise equilibrada preparados

Nota. Elaboração própria.

6 Cronograma

6.1 Planeamento Inicial

O planeamento do estágio teve como base as 15 semanas propostas. Neste tempo foram inseridas sprints quinzenais de desenvolvimento efetivo, seguidas por uma fase de testes e estabilização.

O plano de desenvolvimento foi o seguinte:

Tabela 4. *Cronograma de estágio*

Sprint	Período	Objetivos Planeados
Sprint 0	5 a 7 de março	Introdução ao projeto, integração com a equipa e exploração da base de código e ferramentas <i>Unity</i> .
Sprint 1	10 a 21 de março	Estudo do código, compreensão da arquitetura base e implementação do sistema de propostas de políticas.
Sprint 2	24 de março a 4 de abril	Desenvolvimento da lógica de sessões legislativas por

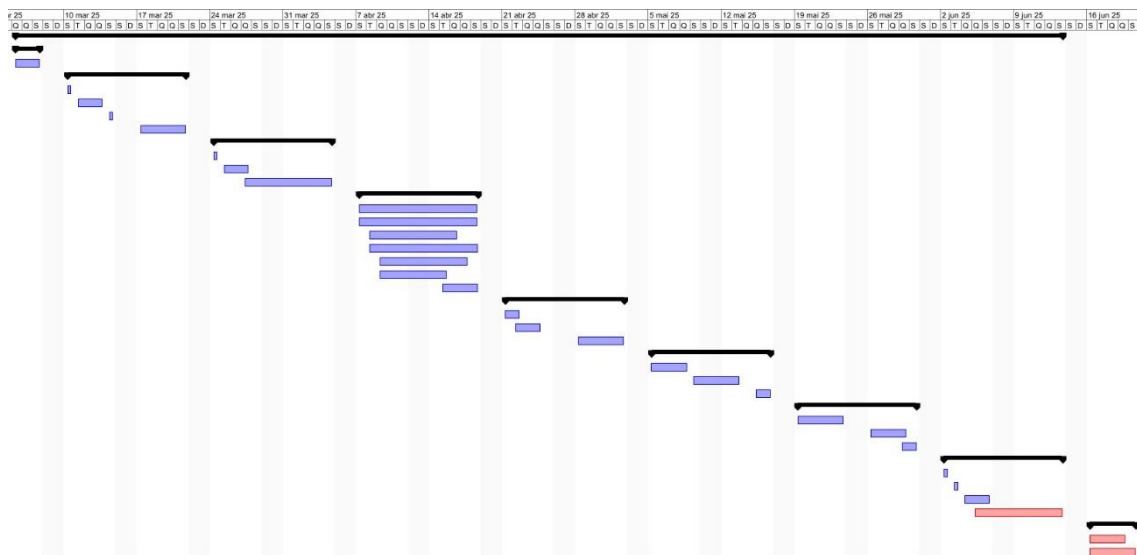
		turnos e simulação de consequências políticas.
Sprint 3	7 a 18 de abril	Elaboração da documentação técnica dos principais sistemas do jogo: país, sub-fações, políticas, situações, efeitos e classes sociais. Criação de <i>UI</i> provisória para arquivo histórico de políticas.
Sprint 4	21 de abril a 2 de maio	Refatoração da estrutura de <i>GoverningBody</i> e <i>Effect</i> . Início do design dos setores industriais e dos fluxos de recursos.
Sprint 5	5 a 16 de maio	Implementação da arquitetura modular de sub-fações e efeitos. Desenvolvimento do sistema de dívida e crises económicas.
Sprint 6	19 a 30 de maio	Integração do sistema formatado de <i>logs</i> (<i>EventLog</i>). Finalização da lógica de sessões legislativas por fases. Início da redação do relatório.
Sprint 7	2 a 13 de junho	Conclusão do relatório final, testes <i>end-to-end</i> e implementação da nota personalizada de <i>onboarding</i> .
Sprint 8	16 a 20 de junho	Conclusão do estágio

Nota. Elaboração própria.

6.2 Execução Real





































A execução real do estágio seguiu de forma geral o planeamento previamente previsto, porém com alguns ligeiros desvios relacionados com feriados, complexidade inesperada de determinadas tarefas e o tempo necessário para testes e refatoração.

Figura 49. *Gráfico de Gantt*



Nota. Elaboração própria com recurso ao ProjectLibre.

Figura 50. Cronograma

		Nome	Duração	Início	Fim
1		State of Affairs	72,875 dias?	05-03-2025 9:00	13-06-2025 17:00
2		Sprint 0	2,875 dias	05-03-2025 9:00	07-03-2025 17:00
3		Introdução ao Projeto	2,875 dias	05-03-2025 9:00	07-03-2025 17:00
4		Sprint 1	9,875 dias?	10-03-2025 9:00	21-03-2025 17:00
5		Studying the code	0,875 dias	10-03-2025 9:00	10-03-2025 17:00
6		Unity Essentials	2,875 dias	11-03-2025 9:00	13-03-2025 17:00
7		Acompanhamento do desenvolvimento	0,875 dias?	14-03-2025 9:00	14-03-2025 17:00
8		Develop Policy Proposal System	4,875 dias?	17-03-2025 9:00	21-03-2025 17:00
9		Sprint 2	9,875 dias?	24-03-2025 9:00	04-04-2025 17:00
10		Studying the code	0,875 dias	24-03-2025 9:00	24-03-2025 17:00
11		Develop Political Consequences of Legislat...	2,875 dias	25-03-2025 9:00	27-03-2025 17:00
12		Develop Legislative Session Turn Logic	6,875 dias?	27-03-2025 9:00	04-04-2025 17:00
13		Sprint 3	10 dias?	07-04-2025 8:00	18-04-2025 17:00
14		Create Country Documentation	9,875 dias	07-04-2025 8:00	18-04-2025 16:00
15		Create Situation Documentation	9,875 dias	07-04-2025 8:00	18-04-2025 16:00
16		Create Policy Documentation	7 dias?	08-04-2025 8:00	16-04-2025 17:00
17		Create Faction Documentation	9 dias?	08-04-2025 8:00	18-04-2025 17:00
18		Create Effect Documentation	7 dias?	09-04-2025 8:00	17-04-2025 17:00
19		Create Social Strata Documentation	5 dias?	09-04-2025 8:00	15-04-2025 17:00
20		Create Placeholder UI to Consult Historical...	4 dias?	15-04-2025 8:00	18-04-2025 17:00
21		Sprint 4	10 dias?	21-04-2025 8:00	02-05-2025 17:00
22		Refactor GoverningBody structure	2 dias?	21-04-2025 8:00	22-04-2025 17:00
23		Refactor Effect structure	3 dias?	22-04-2025 8:00	24-04-2025 17:00
24		Define Industry Sector Activities and Reso...	5 dias?	28-04-2025 8:00	02-05-2025 17:00
25		Sprint 5	10 dias?	05-05-2025 8:00	16-05-2025 17:00
26		Develop Debt and Economic Crisis Mechanic...	4 dias?	05-05-2025 8:00	08-05-2025 17:00
27		Implement Faction System Architecture	3 dias?	09-05-2025 8:00	13-05-2025 17:00
28		Implement Effect System Architecture	2 dias?	15-05-2025 8:00	16-05-2025 17:00
29		Sprint 6	10 dias?	19-05-2025 8:00	30-05-2025 17:00
30		Implement EventLog Formatting System	5 dias?	19-05-2025 8:00	23-05-2025 17:00
31		Implement Legislative Sessions per Turn	4 dias?	26-05-2025 8:00	29-05-2025 17:00
32		Report development	2 dias?	29-05-2025 8:00	30-05-2025 17:00
33		Sprint 7	10 dias?	02-06-2025 8:00	13-06-2025 17:00
34		Report development	1 dia?	02-06-2025 8:00	02-06-2025 17:00
35		E2E Tests	1 dia?	02-06-2025 17:00	03-06-2025 17:00
36		Display personalized "Welcome to Office" ...	3 dias?	03-06-2025 17:00	06-06-2025 17:00
37		Fix Red Phone + ESC key mash UI problems	7 dias?	05-06-2025 8:00	13-06-2025 17:00
38		Sprint 8	5 dias?	16-06-2025 8:00	20-06-2025 17:00
39		Fix Black Phone + ESC key mash UI problems	4 dias?	16-06-2025 8:00	19-06-2025 17:00
40		Report development	5 dias?	16-06-2025 8:00	20-06-2025 17:00

Nota. Elaboração própria com recurso ao ProjectLibre.

6.3 Análise de Desvios e Justificações

No decorrer do estágio, alguns desafios foram surgindo como a complexidade de todo o projeto já desenvolvido até ao momento. O tempo necessário para garantir a consistência entre *UI* e a lógica interna, especialmente na programação do *feedback* ao jogador.

Também a forte necessidade de refatorar código já funcional, de modo a garantir que este seja mais fácil de manusear, garantir manutenção e extensibilidade do sistema no futuro.

Contudo, e apesar de todos os desvios e ajustes, a maioria dos objetivos planeados foi alcançada nos prazos definidos, permitindo assim o correto desenvolvimento do projeto ao longo do estágio.

7 Meios Utilizados

7.1 Meios Humanos (equipa de trabalho)

Todo o projeto desenvolvido no âmbito do estágio curricular, integrado numa equipa de desenvolvimento da *Top Sigma Studios*, mais focada na elaboração de jogos, tanto a nível interno como para outras Entidades como auxílio ou complemento, mas também com forte colaboração da equipa de *design* para integração de *Assets*, e conexão de código com os elementos visuais.

A equipa de trabalho contou com os seguintes intervenientes:

- Estagiário: Paulo Ricardo Figueiredo Pinto Monteiro Guimarães, responsável pelo desenvolvimento do jogo *State of Affairs*, incluindo a lógica das funcionalidades, interfaces, testes e documentação técnica.
- Orientador: Eng.º Ricardo Nunes, que acompanhou todo o progresso do projeto, orientou nas decisões técnicas, organizou todo o calendário de sprints e tarefas a serem executadas e supervisionadas.
- Equipa de Desenvolvedores: Outros elementos da equipa responsáveis pelo desenvolvimento do projeto, no total éramos 3 desenvolvedores ativos diariamente e o Orientador.
- Equipa de *Design*: Também importante mencionar a equipa de *design*, composta por 2 elementos visando criar todos os *assets* e elementos visuais presentes no jogo.

7.2 Meios Materiais (*hardware, software, ferramentas*)

Para a execução de todo o projeto, era necessário material adequado, como tal foram utilizados os seguintes meios, materiais e ferramentas:

Hardware:

- Computador Portátil:
 - *CPU (Central Processing Unit): Intel Core i5-1335U;*
 - *RAM (Random Access Memory): 16 GB DDR4 (Gigabyte Double Data Rate);*
 - *Armazenamento: 512 GB SSD (Solid State Drive);*
 - *GPU (Graphics Processing Unit): Intel Iris Xe;*
 - *Sistema Operativo: Windows 11.*

Software:

- *Unity 2022.3 LTS* - Motor escolhido para o desenvolvimento do jogo
- *Visual Studio Code* - Ambiente de desenvolvimento (*IDE - Integrated Development Environment*) para a programação em C#.
- *GitHub* - Sistema de controlo de versões e repositório remoto.
- *Jira e Confluence* - Programas do *software Atlassian* para gerir sprints, tarefas e toda a execução do projeto em equipa no Jira e documentação no *Confluence*.

8 Problemas e Decisões Tomadas

8.1 Problemas Encontrados

Durante o desenvolvimento do projeto, várias foram os problemas encontrados, a nível técnico e conceptual, que exigiram bastante determinação e foco para os conseguir superar.

O principal desafio deveu-se à complexidade do projeto, a quantidade de *scripts*, a robustez do código até então programado. A linguagem, o motor *Unity* e todas as boas práticas no desenvolvimento de jogos.

Outra dificuldade encontrada foi o próprio tema do jogo, algo bastante interessante, mas igualmente complexo, um sistema político minuciosamente detalhado com dados históricos profundos que se mostrou desafiante de início ao fim do projeto.

8.2 Decisões e Justificações

Para ultrapassar estes problemas, foi essencial abordar diferentes decisões estratégicas e técnicas, justificadas pela necessidade de garantir a robustez, escalabilidade e clareza do projeto.

Documentação de algumas tarefas previamente implementadas e tarefas atribuídas no sentido de documentar várias categorias do jogo como sistemas de políticas completo, sub-fações, efeitos, corpos governamentais, etc.

O estudo do código seguindo o fluxo e implementando *logs* para efeitos de teste e perceber como cada método funcionava na transição de cada classe, permitindo assim perceber a lógica de funcionamento e respetivo fluxo.

Tanto para a compreensão dos termos políticos, dados históricos, linguagem C# foi importante algum estudo por fora assim como um curso para compreender o motor *Unity*.

9 Análise de Resultados

9.1 Avaliação dos Objetivos Atingidos

O principal objetivo para o presente estágio passava pela elaboração de um *Vertical Slice* funcional do jogo *State of Affairs*, que representasse um ciclo completo de jogabilidade política, não finalizado, mas com algum avanço para ser possível averiguar a continuação do desenvolvimento do jogo.

Ao longo do estágio, foram desenvolvidas diversas funcionalidades, tais como:

- Sistema de propostas políticas e votação nos órgãos governamentais;
- Comportamento dinâmico das sub-fações, com reações ideológicas e estratégicas;
- Sistema de registo e visualização de *event logs*, permitindo acompanhar decisões, eventos e alterações no estado do jogo;
- Integração de *assets* para melhorar a interfaces de utilizador;
- Sistema de *feedback* ao jogador com base nas suas decisões.

Além de todo o desenvolvimento técnico, houve também lugar para os objetivos secundários, entre eles:

- Aplicação de boas práticas de programação orientada a objetos, com modularidade do código;

- Documentação das funcionalidades desenvolvidas;
- Integração e respetiva colaboração com a equipa mediante reuniões regulares.

9.2 Impacto Pessoal e Profissional

O presente estágio constituiu uma oportunidade muito importante naquilo que foi a exploração de uma vertente pela qual eu ainda não tivera contacto, o desenvolvimento de jogos. Embora tenha demonstrado muito empenho e dedicação face ao projeto, concluo que o desenvolvimento de jogos, em particular, não se enquadra nos meus interesses profissionais a longo prazo.

Contudo, esta experiência revelou-se extremamente útil em vários aspetos. Em primeiro lugar, permitiu-me consolidar conhecimentos técnicos em programação orientada a objetos, em especial na linguagem C#.

Acima de tudo, este estágio permitiu uma descoberta consciente de preferências profissionais, ajudando-me a perceber com maior clareza às áreas onde me sinto mais motivado e realizado, como o desenvolvimento *web*, com enfoque no *Front-End*.

Posto isto, todas as dificuldades sentidas num projeto desafiante como este obrigou-me a crescer enquanto profissional. Aprendi a lidar com momentos de desmotivação, a solicitar ajuda quando necessário, a refletir sobre o meu progresso e a manter o foco mesmo em tarefas que não me despertavam entusiasmo imediato.

Concluo assim, o presente estágio com impacto positivo no meu percurso enquanto futuro Engenheiro Informático, não por me apontar numa direção exata, mas por proporcionar-me uma ajuda fundamental a eliminar caminhos menos alinhados com os meus objetivos e por me fortalecer com ferramentas, atitudes e experiências que serão, certamente, muito valiosas em qualquer contexto profissional.

Conclusões

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.”

(Albert Einstein)

É esta a sensação que fica após a conclusão do Estágio Profissional, uma mente maior, mais aberta e sensível a novos conteúdos do processo de desenvolvimento de um jogo. Este Estágio profissional permitiu que fosse possível conhecer e experimentar novas técnicas e diversos métodos que certamente constituíram um forte pilar para que daqui em diante seja possível aplicar os mesmos em diferentes áreas de Engenharia Informática e/ou mercado de trabalho de forma autónoma e mais eclética.

Durante esta jornada nem tufo foi fácil. A dificuldade na adaptação dominou todo o período de estágio. As ferramentas, a linguagem, os processos, o projeto já num estado um tanto quanto avançado, a dinâmica de trabalho remoto desde o primeiro dia e até a exigência técnica e criativa envolvida. Surgiram momentos de desmotivação, alimentados por bloqueios técnicos, sensação de sobrecarga e receio de não corresponder às expectativas, a ajuda quase inexistente, obrigando-me a encontrar outras formas de ganhar confiança.

Todo o período de Estágio providenciou experiências marcantes enquanto *Game Developer* que possibilitou o crescimento quer profissionalmente como pessoa com valores e ritmo de trabalho na área. Crescimento este que surge com muito empenho, rigor, humildade e dedicação extrema nesta que foi desafiante viagem pedagógica. A incansável busca por conhecimento, abriu portas para a constante evolução ao nível de investigação, de ação e reflexão perante as adversidades impostas, como sendo uma chave para o desenvolvimento profissional.

Assim, o objetivo estabelecido para o local de estágio foi inteiramente cumprido, nomeadamente o desenvolvimento do jogo na versão de *Vertical Slice* e cumprimento de metas e objetivos, pese embora não tenha cumprido com as minhas expectativas e do supervisor Eng.º Ricardo Nunes. Tudo para proporcionar um correto funcionamento, com o envolvimento para marcar a diferença.

No decorrer do tempo em estágio 100% remoto, foi possível procurar sempre ultrapassar todas as expectativas, dando sempre o máximo, tendo como impulsionadores o Eng.º Ricardo Nunes entre outros excelentes profissionais na área que possibilitassem transpor toda e qualquer fasquia estabelecida.

O desenvolvimento profissional atingido nas áreas desempenhadas persiste em evoluir e mostrar novos caminhos do conhecimento. Contudo, este consta como uma

pequena gota de água num vasto e enorme oceano, existe, ainda, um grande percurso para o crescimento profissional.

A fasquia não parou e continua a subir, as exigências impostas são gradativas e é com a consciência das virtudes e lacunas enquanto Engenheiro Informático que se vão a transpor as elevadas fasquias da área, profissionalmente e académico.

Em suma, o presente Estágio constituiu um leque variado de experiências, revelando assim ser uma fonte inestimável de conhecimento e alargando a visão em áreas de desenvolvimento de jogos e programação.

Por fim, não posso deixar de agradecer à equipa da Top Sigma, em especial ao Eng.º Ricardo Nunes, pelo acompanhamento, desafios e oportunidade de crescimento que me foram proporcionados ao longo deste estágio. A todos os colegas e profissionais que, direta ou indiretamente, contribuíram para este percurso, deixo o meu sincero reconhecimento. Esta experiência marcou profundamente o meu trajeto enquanto estudante e aspirante a engenheiro informático, reforçando o compromisso com o meu desenvolvimento pessoal, técnico e profissional.

Referências bibliográficas

Asal, V., Jahanbani, N., Lee, D., & Ren, J. (2018). Mini-Games for Teaching Political Science Methodology. *PS: Political Science & Politics*, 51(4), 838–841.
<https://doi.org/10.1017/S1049096518000902>

Atlassian. (2025a).

<https://www.atlassian.com/?campaign=9869842058&adgroup=99178949214&targetid=kwd->

[1679236662&matchtype=e&network=g&device=c&device_model=&creative=431899924002&keyword=atlassian&placement=&target=&ds_eid=700000001530700&ds_e1=GOOGLE&gad_source=1&gad_campaignid=9869842058&gclid=Cj0KCQjw0erBBhDTARIsAKO8iqRdxqz06Q8b7ruPTx5h5kxo2liEDQyHsYdzwChcunrz9ST7NRylxnAaApDVEALw_wcB](https://www.atlassian.com/?campaign=9869842058&adgroup=99178949214&targetid=kwd-1679236662&matchtype=e&network=g&device=c&device_model=&creative=431899924002&keyword=atlassian&placement=&target=&ds_eid=700000001530700&ds_e1=GOOGLE&gad_source=1&gad_campaignid=9869842058&gclid=Cj0KCQjw0erBBhDTARIsAKO8iqRdxqz06Q8b7ruPTx5h5kxo2liEDQyHsYdzwChcunrz9ST7NRylxnAaApDVEALw_wcB)

Atlassian. (2025b). *O que é Scrum? (E como começar)*. Atlassian.
<https://www.atlassian.com/br/agile/scrum>

Blog Impulso | Unity: Uma poderosa ferramenta para desenvolvimento de jogos. (2025, maio 12). <https://blog.impulso.team/unity-uma-poderosa-ferramenta-para-desenvolvimento-de-jogos-mfbp/>

Confluence. (2025).

https://www.atlassian.com/br/software/confluence?gclsrc=aw.ds&&campaign=19190483749&adgroup=149977751491&targetid=kwd-366356292898&matchtype=e&network=g&device=c&device_model=&creative=738938281176&keyword=confluence&placement=&target=&ds_eid=700000001542923&ds_e1=GOOGLE&gad_source=1&gad_campaignid=19190483749&gclid=Cj0KCQjw0erBBhDTARIsAKO8iqQ5dIRYyRuZi42I1liV19QnGEwFVU5uGU_NltaqYmyLJ8sdR_gFi10aAsXKEALw_wcB

Crusader Kings III. (2020). <https://www.paradoxinteractive.com/games/crusader-kings-iii/about>

Democracy 4. (2020). <https://www.positech.co.uk/democracy4/>

Desenvolvedor, C. do. (2023, fevereiro 24). Requisitos funcionais e não funcionais: O que são e como identificar? *Blog da Casa do Desenvolvedor*.
<https://blog.casadodesenvolvedor.com.br/requisitos-funcionais-e-nao-funcionais/>

Diagrama de caso de uso UML: O que é, como fazer e exemplos. (2025). Lucidchart.
<https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml>

Escalabilidade e desempenho: Os pilares de uma arquitetura de software de sucesso | AppMaster. (2025). <https://appmaster.io/pt/blog/arquitetura-de-software-de-escalabilidade-e-desempenho>

Jira. (2025).
https://www.atlassian.com/br/software/jira?campaign=19324540241&adgroup=143040538525&targetid=kwd-855725830&matchtype=e&network=g&device=c&device_model=&creative=642069008118&keyword=jira&placement=&target=&ds_eid=700000001558501&ds_e1=GOOGLE&gad_source=1&gad_campaignid=19324540241&gclid=Cj0KCQjw0erBBhDTARIsAKO8iqQu4pRSnbgdCevPU4wIPfMD4iuwEXrmANcBkJWqLTddyfg47e-dBGoaAhzVEALw_wcB

matheus. (2024, janeiro 29). *System Design—Performance, Capacidade e Escalabilidade*. Matheus Fidelis - Engineering Blog.
<https://fidelissauro.dev/performance-capacidade-escalabilidade/>

Microsoft Teams. (2025). <https://www.microsoft.com/pt-pt/microsoft-teams/log-in>

Modularidade: O que é e como aumenta a produtividade empresarial? | Lenovo Portugal. (2025). <https://www.lenovo.com/pt/pt/glossary/modularity/>

O que é Chroma DB? (2025, março 24). IONOS Digital Guide. <https://www.ionos.com/pt-br/digitalguide/servidor/conhecimento/chroma-db/>

O que é um fluxograma? (2025). Lucidchart. <https://www.lucidchart.com/pages/pt/o-que-e-um-fluxograma>

- Oberle, M., Leunig, J., & Ivens, S. (2020). What do students learn from political simulation games? A mixed-method approach exploring the relation between conceptual and attitudinal changes. *European Political Science*, 19(3), 367–386. <https://doi.org/10.1057/s41304-020-00261-2>
- Positech Games*. (1997). <https://www.positech.co.uk/>
- Prada, C. (2024, julho 22). Arquitetura de sistemas: Guia completo da área. [www.euax.com.br](https://www.euax.com.br/2024/07/arquitetura-de-sistemas/). <https://www.euax.com.br/2024/07/arquitetura-de-sistemas/>
- Ravi. (2022, dezembro 29). What is Modularity? *Software Architecture/Design Essentials*. <https://medium.com/software-architecture/what-is-modularity-1480ddb84b51>
- Requisitos funcionais e não funcionais: Guia completo*. (2025). <https://www.mestresdaweb.com.br/tecnologias/requisitos-funcionais-e-nao-funcionais-o-que-sao>
- Schulzke, M. (2014, agosto 1). Video Games and the Simulation of International Conflict. *E-International Relations*. <https://www.e-ir.info/2014/08/01/video-games-and-the-simulation-of-international-conflict/>
- Skanska e OutHere – Aplicativos de segurança de construção em VR | Estudo de caso da Unity*. (2025). Unity. <https://unity.com/case-study/outhere-and-skanska>
- Suzerain*. (2020). Suzerain. <https://www.suzeraingame.com>
- Top Sigma*. (2025). <https://www.top-sigma.com/>
- Tropico 6*. (2019). Kalypso EU. <https://www.kalypsomedia.com/eu/tropico-6-standard-edition>
- Unity*. (2025). <https://unity.com/pt>
- Unity | Real-time 3D for enterprise*. (2025). https://create.unity.com/content-hub?utm_source=chatgpt.com
- Unity Learn*. (2025). Unity Learn. <https://learn.unity.com>
- UXCam*. (2025). *Como fazer Testes de Usabilidade em Websites*. UXCam Blog. <https://uxcam.com/br/blog/como-fazer-testes-de-usabilidade-em-websites>

Glossário

Affinity: Grau de alinhamento entre uma facção e uma política, com impacto na lealdade e nas votações.

Caminho Conservador: Uma das possíveis ideologias no jogo, caracterizada por políticas autoritárias, apoio da NKVD e forças armadas e repressão às reformas.

Caminho Equilibrado: Ideologia centrada na estabilidade e consenso, promovendo um equilíbrio entre facções sem extremismos.

Caminho Reformista: Ideologia voltada para a liberalização política e económica, com apoio dos Reformistas e foco em direitos civis.

Central Processing Unit (CPU): Unidade central de processamento responsável pela execução de instruções e operações lógicas do computador.

Classes Sociais: Grupos populacionais definidos no jogo com base no seu papel socioeconómico, como Classe Trabalhadora, Classe Média e Classe Alta.

Cloud Engineering: Área da engenharia informática dedicada ao desenvolvimento, manutenção e otimização de soluções baseadas em computação na nuvem.

ConditionGroup: Estrutura lógica que permite combinar condições para acionar situações no jogo.

Create Placeholder UI to Consult Historical Policy: Data Funcionalidade desenvolvida para simular uma interface temporária onde o jogador pode consultar o histórico de políticas.

Crise Dinâmica: Evento emergente no jogo gerado com base nas decisões do jogador e nas condições do país, exigindo uma resposta estratégica.

Data Classes: Classes utilizadas para representar estruturas de dados no jogo, geralmente carregadas a partir de ficheiros JSON.

Data Warehouse: Sistema de armazenamento de dados utilizado para análise e geração de relatórios, frequentemente em contextos empresariais.

Define Industry Sector Activities: Funcionalidade que define as atividades económicas dos setores industriais e o fluxo de recursos entre eles.

Develop Debt and Economic Crisis Mechanics: Implementação da lógica de dívida nacional, déficit e crise económica, influenciada por políticas e decisões estratégicas.

Develop Policy Proposal System: Sistema responsável pela proposta, debate e votação de políticas no jogo.

Develop Political Consequences of Legislative Failure: Funcionalidade que define os efeitos e consequências de políticas não aprovadas.

Double Data Rate (DDR): Tipo de memória RAM que realiza duas transferências de dados por ciclo de clock.

Effect: Mecanismo que modifica valores do jogo, podendo ser imediato ou gradual, como alterar o PIB ou a lealdade.

Efeitos Graduais: Efeitos que não são aplicados de forma imediata, mas que se desenrolam ao longo de vários turnos.

Enforce Bodycams on Police Officers: Política que obriga o uso de câmaras corporais por agentes da autoridade para reforçar a transparência.

Eventos Emergentes: Eventos não programados que surgem em resposta às ações do jogador ou à evolução do jogo.

Feedback ao Jogador: Sistema de comunicação de consequências visuais e textuais ao jogador com base nas suas ações.

Formatting System: Sistema que formata e apresenta informação de forma organizada nos logs e relatórios do jogo.

Game Developer: Profissional responsável pelo desenvolvimento técnico de videojogos, incluindo programação e integração de sistemas.

Georgiy Malenkov: Figura política soviética que assumiu temporariamente a liderança após a morte de Staline.

Graphics Processing Unit (GPU): Unidade de processamento gráfico especializada no cálculo de operações relacionadas com imagens, jogos e gráficos.

Gross Domestic Product (GDP): Indicador económico que representa o valor total dos bens e serviços produzidos por um país.

Implement EventLog Formatting System: Sistema responsável por organizar e apresentar os eventos do jogo em formato legível.

Implement Faction System Architecture: Estrutura que organiza a lógica interna das facções, incluindo lealdade, alinhamento e comportamento.

Implement Legislative Sessions per Turn: Mecânica que define sessões legislativas com base no turno atual, permitindo o debate e votação de políticas.

Increase Military Spending: Política que aumenta o orçamento militar, reforçando o apoio das facções autoritárias.

Integrated Development Environment: (IDE) Ambiente de desenvolvimento que integra ferramentas de edição, compilação e depuração de código.

Interstate Highway System: Política de investimento em infraestruturas rodoviárias nacionais, com impacto na economia e opinião pública.

ISPGAYA: Instituto Superior Politécnico Gaya, instituição de ensino superior onde decorre o curso de Engenharia Informática.

Lógica de Lealdade: Sistema que determina o apoio ou oposição de facções e grupos de interesse com base nas ações do jogador.

NKVD: Comissariado do Povo para Assuntos Internos, organismo soviético responsável pela segurança interna e repressão.

Placeholder: UI Interface temporária utilizada durante o desenvolvimento para simular funcionalidades ainda em construção.

Politburo: Órgão central de decisão do Partido Comunista da União Soviética.

Projeto de Engenharia Informática em Contexto Empresarial: Unidade curricular do ISPGAYA que integra um estágio prático numa empresa da área.

ScriptableObject: Tipo especial de objeto em Unity usado para armazenar dados de forma reutilizável e serializável.

Sentiment: Valor numérico (0–100) que representa o apoio de uma classe social ao governo.

Sistema Modular: Estrutura do jogo dividida em componentes independentes que comunicam entre si, facilitando a manutenção e escalabilidade.

Situações e Dilemas: Eventos com múltiplas escolhas possíveis que afetam diretamente o rumo do jogo e as reações das facções.

Sprint: Período onde é definido um conjunto de tarefas a cumprir.

Top Sigma Studios: Divisão da empresa Top Sigma dedicada ao desenvolvimento de videojogos e projetos interativos.

UI/UX: Interface do Utilizador / Experiência do Utilizador, elementos gráficos e de interação que facilitam a utilização do jogo.

Vertical Slice: Secção jogável de um jogo que demonstra funcionalidades centrais e serve como protótipo de apresentação.