UNIVERSIDADE CEUB – TAGUATINGA CAMPUS II

BACHARELADO CIÊNCIA DA COMPUTAÇÃO

Estudo de Caso

Análise Técnica do Estudo de Caso

|  |  |  |
| --- | --- | --- |
| Turma: | CC - Matutino | |
| Nome do squad: | Van Preta | |
| Versão da entrega: | (X) Primeira parcial ( ) Segunda parcial ( ) Final | |
| Nomes dos alunos: | MATRÍCULA | NOME COMPLETO |
| 22454089 | Eduardo Cabral Nunes |
| 22452139 | Felipe Amorim Monteiro |
| 22450586 | Gabriel Ferreira Costa |
| 22452647 | Leonardo Victor Lima Soares |
| 22453629 | Pedro Ivo Gonçalves Pinheiro Coelho |
|  |  |
| BANCA EXAMINADORA | Orientador: Denys Alves da Silvia | ASSINATURA: |
|  | Grupos: | ASSINATURA: |

**SUMÁRIO**

[TÍTULO 4](#_heading=h.4d34og8)

[RESUMO 4](#_heading=h.2s8eyo1)

[INTRODUÇÃO 5](#_heading=h.17dp8vu)

[ANÁLISE TÉCNICA 6](#_heading=h.3rdcrjn)

[1.Fundamentos da Programação Orientada a Objetos 6](#_heading=h.26in1rg)

[2.Arquitetura de Minecraft: Estrutura e Implementação 7](#_heading=h.lnxbz9)

[3.Modularidade e Extensibilidade 8](#_heading=h.35nkun2)

[3.1 O Mod Aether e a Implementação do Portal 9](#_heading=h.l8e2henjut97)

[4. Impacto da Abordagem POO no Sucesso do Jogo 10](#_heading=h.1ksv4uv)

[5. Contribuições da Comunidade e a Evolução dos Mods 11](#_heading=h.44sinio)

[6. Comparação com Outras Abordagens de Desenvolvimento 12](#_heading=h.2jxsxqh)

[CONCLUSÃO 1](#_heading=h.z337ya)

[REFERÊNCIAS BIBLIOGRÁFICAS 1](#_heading=h.3j2qqm3)

**EDUARDO CABRAL NUNES**

**FELIPE AMORIM MONTEIRO**

**GABRIEL FERREIRA COSTA**

**LEONARDO VICTOR LIMA SOARES**

**PEDRO IVO GONÇALVES PINHEIRO COELHO**

# TÍTULO

Minecraft: Uma Abordagem de Programação Orientada a Objetos no Desenvolvimento de Games e a Evolução dos Mods – O Caso do Mod Aether

UNICEUB TAGUATINGA CAMPUS II

1ºSEMESTRE DE 2025

# RESUMO

Este estudo de caso analisa a aplicação dos conceitos da Programação Orientada a Objetos (POO) no desenvolvimento de Minecraft, um dos jogos mais vendidos da história. A investigação aborda os fundamentos da POO, a estrutura modular do código, a implementação dos elementos do jogo, como blocos, entidades e mecânicas, e a influência dessa abordagem na expansão e manutenção do software. Além disso, discute brevemente métodos específicos de POO aplicados pela Mojang Studios, como a utilização de diagramas UML e referências ao guia SWEBOK. O estudo destaca como a modularidade e a extensibilidade inerentes à POO permitiram a criação de um ambiente dinâmico e personalizável, impulsionando a comunidade de modding e contribuindo diretamente para o sucesso do jogo.

**Palavras-chave:** Programação Orientada a Objetos; Modularidade; Extensibilidade; Minecraft; Desenvolvimento de Jogos; UML; SWEBOK.

# INTRODUÇÃO

O desenvolvimento de softwares complexos demanda metodologias eficazes que promovam organização, manutenção e escalabilidade. Nesse cenário, a Programação Orientada a Objetos (POO) destaca-se como uma metodologia essencial para o desenvolvimento de aplicações robustas e flexíveis. Silva (2018, p. 12) destaca que a POO facilita a criação de componentes autônomos, o que favorece a reutilização e manutenção simplificada do código.

Minecraft, um dos jogos mais populares do mundo, lançado oficialmente pela Mojang Studios, utiliza fortemente o paradigma de POO. Desenvolvido com base em blocos e entidades implementados como objetos independentes com propriedades e métodos específicos, o jogo facilita expansões e personalizações através da criação de mods pela comunidade. Pereira (2015, p. 45) afirma que essa arquitetura modular é fundamental para o sucesso e a longevidade do jogo.

Ao longo deste estudo, serão analisados os fundamentos da POO e sua implementação em Minecraft, destacando como essa estrutura contribuiu para a evolução dos games e para a consolidação do jogo como um fenômeno global. A análise baseia-se em estudos acadêmicos, documentação técnica e literatura especializada, utilizando as diretrizes da ABNT NBR 6023:2018 para referências bibliográficas e da ABNT NBR 10520:2023 para citações.

# ANÁLISE TÉCNICA

## 1.Fundamentos da Programação Orientada a Objetos

A Programação Orientada a Objetos é fundamentada em quatro pilares principais:

**1.1 Encapsulamento**

Trata-se da técnica de ocultar os dados internos dos objetos, permitindo o acesso apenas por meio de métodos públicos. Essa prática garante a integridade dos dados e reduz a dependência entre os componentes do sistema (SILVA, 2018, p. 15).

**1.2 Herança**

Permite a criação de novas classes com base em classes existentes, promovendo a reutilização de código e a criação de hierarquias de objetos. A herança facilita a implementação de comportamentos comuns a diversos elementos (PEREIRA, 2015, p. 47).

**1.3 Polimorfismo**

Refere-se à capacidade dos objetos de responderem de forma distinta a uma mesma mensagem ou método, dependendo do contexto. Isso possibilita a implementação de métodos genéricos que podem ser adaptados conforme a necessidade do sistema (MOREIRA, 2017, p. 62).

**1.4 Abstração**

Consiste na identificação dos aspectos essenciais de um objeto, ignorando detalhes supérfluos. A abstração permite a modelagem de conceitos complexos de maneira simplificada e organizada (SILVA, 2018, p. 19).

Esses fundamentos da programação orientada a objetos fornecem uma base sólida que permite a implementação de uma arquitetura modular eficiente, como será analisado detalhadamente na seção seguinte sobre a arquitetura de Minecraft.

## 

## 2.Arquitetura de Minecraft: Estrutura e Implementação

Minecraft foi desenvolvido em Java, linguagem que adota fortemente o paradigma orientado a objetos. Cada elemento do jogo – desde os blocos que compõem o ambiente até as entidades que interagem com o jogador – é implementado como um objeto, permitindo a definição de propriedades específicas (por exemplo, resistência, textura) e comportamentos (como interação e destruição) (MINECRAFT, 2011).

A estrutura modular do código de Minecraft se baseia na criação de classes que representam os diversos componentes do jogo. Por exemplo, a classe “Bloco” pode ser estendida para criar subclasses como “Pedra”, “Terra” ou “Madeira”, cada uma com atributos e métodos particulares. Essa abordagem favorece a manutenção do código, uma vez que alterações em uma classe base se propagam para suas subclasses, reduzindo a redundância e facilitando atualizações (PEREIRA, 2015, p. 52).

Além disso, a utilização de interfaces e classes abstratas permite a definição de contratos para o comportamento dos objetos, garantindo que todas as entidades do jogo implementem métodos essenciais para a interação com o ambiente. Conforme MOREIRA (2017, p. 68), essa prática é crucial para a integração e a comunicação entre os diferentes módulos do sistema.

A utilização de diagramas UML pela Mojang Studios contribuiu significativamente para a clareza e comunicação entre os desenvolvedores, permitindo uma visualização eficaz das interações e dependências entre as classes e módulos do sistema. Além disso, o uso das boas práticas recomendadas pelo guia SWEBOK auxiliou na gestão e evolução do código ao longo do desenvolvimento do jogo.

Diante dessa arquitetura claramente orientada a objetos, Minecraft conseguiu implementar uma estrutura modular que facilita não apenas a manutenção, mas também a extensibilidade do jogo, como será detalhado a seguir.

## 

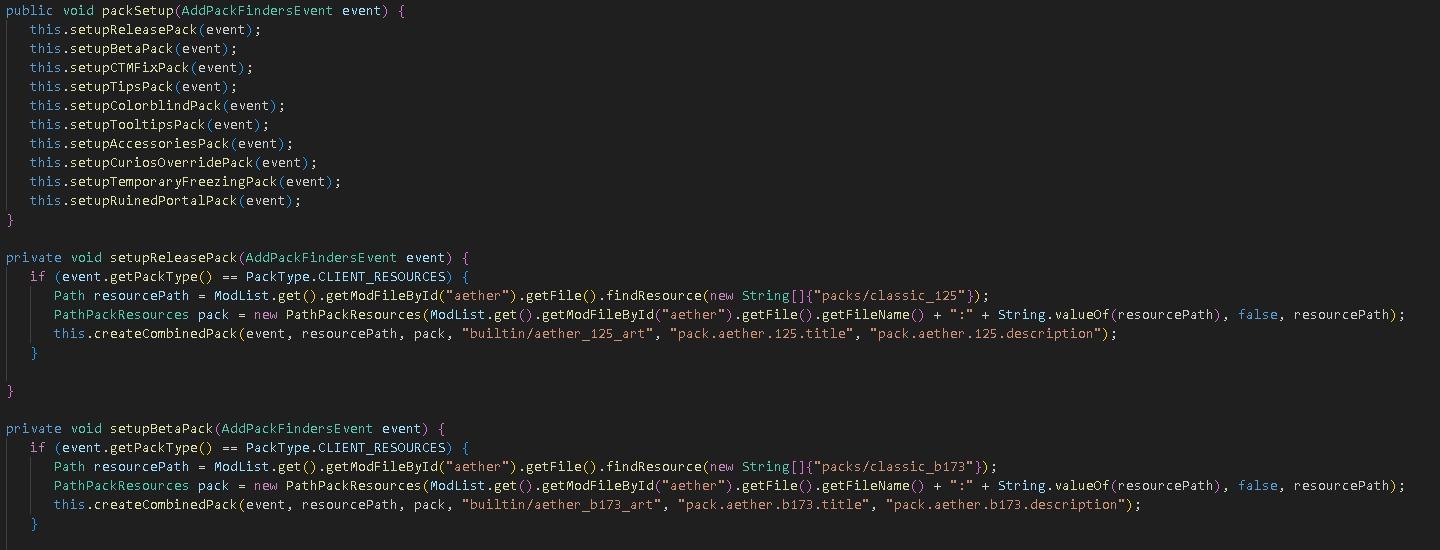
## 3.Modularidade e Extensibilidade

Um dos maiores trunfos da implementação de Minecraft é a modularidade do seu código. Essa característica permite que cada parte do jogo seja desenvolvida, testada e atualizada de forma independente. A modularidade, consequência direta dos conceitos de encapsulamento e abstração, possibilita a adição de novas funcionalidades sem a necessidade de reescrever grandes porções do sistema (SILVA, 2018, p. 23).

Essa arquitetura modular é particularmente visível na forma como o jogo lida com os mods – modificações feitas pela comunidade para adicionar ou alterar funcionalidades. Os mods podem interagir com o sistema base através de interfaces bem definidas, o que reduz o risco de conflitos e torna o processo de modificação mais seguro e eficiente. Segundo estudos recentes, a capacidade de personalização do jogo é um fator determinante para o seu sucesso, pois cria um ecossistema dinâmico de inovação colaborativa (MOREIRA, 2017, p. 75).

#### 3.1 O Mod Aether e a Implementação do Portal

O mod Aether é um dos exemplos mais famosos de como a modularidade e os princípios da POO facilitam a expansão de Minecraft. Esse mod introduz uma nova dimensão – o Aether – que se caracteriza por ambientes celestiais, novas mecânicas de jogo e desafios inéditos. A peça central dessa modificação é o portal do Aether, cuja implementação serve de exemplo para ilustrar a eficácia da POO.



*Observação:* A imagem acima ilustra o trecho do código que implementa a criação e ativação do portal do Aether, demonstrando como os conceitos da POO – especialmente a herança e a implementação de interfaces – são utilizados para integrar essa funcionalidade ao sistema base de Minecraft.

## 

## 4. Impacto da Abordagem POO no Sucesso do Jogo

A utilização dos princípios da POO proporcionou benefícios como a facilidade na manutenção, capacidade de expansão e incentivo à inovação pela comunidade. A modularidade permitiu atualizações rápidas e eficientes, contribuindo diretamente para o sucesso e longevidade de Minecraft.

**4.1 Engajamento da Comunidade**

A abertura para a criação de mods, decorrente de uma arquitetura orientada a objetos robustos, fomentou uma comunidade ativa e inovadora. Essa interação colaborativa não só enriqueceu o jogo, mas também prolongou seu ciclo de vida comercial e cultural (MOREIRA, 2017, p. 80).

Tais benefícios ressaltam a importância de uma boa arquitetura de software no contexto do desenvolvimento de jogos e demonstram como a POO pode ser determinante para o sucesso de um produto digital.

## 

## 5. Contribuições da Comunidade e a Evolução dos Mods

Contribuições da Comunidade e a Evolução dos Mods A arquitetura modular baseada em POO foi crucial para que Minecraft se tornasse uma plataforma vibrante de desenvolvimento colaborativo. A comunidade contribuiu significativamente através de mods, ampliando as possibilidades de jogabilidade e prolongando o ciclo de vida comercial do jogo.

## 

## 6. Comparação com Outras Abordagens de Desenvolvimento

Comparação com Outras Abordagens de Desenvolvimento Comparando-se com abordagens estruturadas ou procedurais, a POO oferece uma maior facilidade para lidar com complexidade, permitindo subdividir sistemas em componentes gerenciáveis. Esse aspecto estratégico foi essencial para atender às demandas crescentes e complexas do Minecraft ao longo do tempo.

**7. Desafios e Perspectivas Futuras**

Apesar dos inúmeros benefícios, a aplicação da POO em um sistema de grande porte como Minecraft também enfrenta desafios. Entre eles, destaca-se a complexidade do gerenciamento de interdependências entre objetos e a necessidade de uma arquitetura bem definida para evitar a proliferação de códigos redundantes. Tais desafios demandam o desenvolvimento contínuo de boas práticas e a adoção de padrões de design que promovam a eficiência e a legibilidade do código (SILVA, 2018, p. 40).

No cenário atual, com a evolução constante das tecnologias e das linguagens de programação, perspectivas futuras apontam para a integração de novas metodologias, como a programação funcional, que podem complementar e enriquecer a abordagem orientada a objetos. A convergência de paradigmas promete oferecer soluções ainda mais robustas para o desenvolvimento de softwares complexos, ampliando o horizonte de possibilidades para a indústria de games (MOREIRA, 2017, p. 92).

Em suma, a análise dos desafios enfrentados por Minecraft ilustra que, embora a POO seja uma ferramenta poderosa, a evolução contínua das demandas tecnológicas exige uma postura adaptativa e inovadora por parte dos desenvolvedores.

# CONCLUSÃO

O estudo de caso apresentado evidencia que a aplicação dos conceitos da Programação Orientada a Objetos foi decisiva para o sucesso de Minecraft. A estrutura modular, baseada em princípios como encapsulamento, herança, polimorfismo e abstração, possibilitou não apenas a criação de um jogo robusto e de fácil manutenção, mas também estimulou a participação ativa da comunidade por meio da criação de mods. Como resultado, Minecraft consolidou-se como um fenômeno global, demonstrando a relevância e a eficácia do paradigma POO no desenvolvimento de softwares complexos.

A análise técnica evidencia que a escolha da POO foi estratégica para atender às demandas de inovação, escalabilidade e adaptação do jogo, proporcionando um ambiente propício para a experimentação e a evolução contínua. Dessa forma, o estudo contribui para o entendimento dos benefícios de uma arquitetura orientada a objetos e serve como referência para futuros projetos na indústria de games e outras áreas que demandam alta modularidade e flexibilidade.

# 

# REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 6023:2018 – Informação e documentação: referências – elaboração*. Rio de Janeiro: ABNT, 2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 10520:2023 – Informação e documentação: citações em documentos*. Rio de Janeiro: ABNT, 2023.

MOREIRA, Carlos. *Estruturas de Dados e Algoritmos em Java*. São Paulo: Editora Beta, 2017.

PEREIRA, Ana. *Desenvolvimento de Software com Java*. Rio de Janeiro: Editora Alfa, 2015.

SILVA, João. *Programação Orientada a Objetos: Fundamentos, Princípios e Aplicações*. Belo Horizonte: Ciência Moderna, 2018.

MINECRAFT. Mojang Studios. Disponível em:<https://www.minecraft.net>. Acesso em: 21 fev. 2025.