

Mineger: Minecraft Modpack Manager

Mini Visão

**Versão *1.0.0***

**Alunos:** *Nicoly Silva Kühn*

*Ruan Gomes do Rosario*

Paranaguá, *2021*

# Declaração do Problema

Muitos jogos atualmente possuem MODs, ou seja, modificações no jogo original que abrangem desde pequenas alterações até a construção de um jogo praticamente novo. O jogo Minecraft se encaixa nesse conjunto de jogos que possuem uma grande variedade de MODs.

Essas modificações, embora incentivadas pelo desenvolvedor, não são conteúdo oficial do jogo, e acabam sendo distribuídas em diversos canais, sendo que cada um tem seu próprio conjunto de ferramentas (APIs, bibliotecas, porções do jogo decompiladas, etc) que permitem a utilização dos MODs.

Uma breve análise dessa situação do ponto de vista de um jogador revela que manter MODs de diversas fontes e diferentes “modloaders” de forma manual é extremamente inconveniente, infuncional e consome o tempo do jogador, com isso se abre espaço para automação; principalmente considerando atualizações, deleções, composições de MODs (modpacks), etc.

Não haver um sistema de organização de MODs afeta diretamente a comunidade de jogadores de Minecraft, já que a falta de?? . Uma solução viável e bem sucedida para esse problema seria a criação de um sistema gerenciador de pacotes que baixe, instale e organize as modificações, separando-as por versões e que apresente ao usuário uma forma rápida e simples de organização de MODs.

# Declaração da Solução ou Produto (Hipótese)

Para resolver o problema apresentado, é necessário uma forma fácil e unificada de baixar, instalar e manter MODs de diversas fontes e formatos, bem como uma maneira de organizar conjuntos de MODs de forma abstrata. Essas são características de um gerenciador de pacotes (package manager, em inglês, e na comunidade UNIX) como o apt, dnf, pacman, yay, portage, etc; que são package managers do Linux. A ideia principal é uma aplicação para desktop com um banco de dados de fontes de modificações (repositórios), que então mostra uma lista dos MODs disponíveis e seus atributos (versões disponíveis, versões compatíveis, dependências, etc), e então baixe os MODs solicitados e os instale corretamente. A aplicação deve também ter acesso ao sistema de arquivos do jogo, para criar perfis de utilização, como conjuntos de MODs que costumam ser usados juntos ou configurações específicas para cada usuário.

Essa solução seria voltada para a comunidade de jogadores de Minecraft que são entusiastas de modificações, que muitas vezes não possuem o conhecimento técnico ou a disponibilidade de tempo para manter e organizar MODs. Essa solução seria um package manager voltado para MODs de minecraft que proporciona uma melhor organização, principalmente se tratando de uma quantidade maior de MODs. Uma alternativa já existente seria o CurseForge, que permite a instalação rápida de MODs por meio de seu cliente; porém carece na organização, que é o foco da presente solução.

# Resumo dos Interessados

|  |  |
| --- | --- |
| **Nome** | **Descrição** |
| A comunidade do jogo Minecraft | Pessoas que frequentemente jogam mas possuem dificuldade de organização |

# Requisitos Funcionais

|  |  |
| --- | --- |
| **Código** | **Requisito** |
| RF01 | Fazer download do mod direto na pasta do jogo para facilitar o acesso à essas modificações |
| RF02 | Desinstalar o mod |
| RF03 | Organizar os mods em perfis de utilização para melhor transição de modpacks |
| RF04 | Organizar perfis por usuários cadastrados para suportar múltiplos jogadores por máquina |
| RF05 | Adicionar a versão do mod |

# Alternativas

Existem ainda CurseForge e MinecraftModUpdater, alternativas gratuitas e ambos mutuamente exclusivos, que servem como gerenciadores de mods e dependências para minecraft, porém essas alternativas não possuem o recurso de organização em pacotes independentes.

# Requisitos não Funcionais

|  |  |  |
| --- | --- | --- |
| **Código** | **Requisito** | **Tipo** |
| RNF01 | Padrão DAO/Repository para camada de persistência | Padrão |
| RNF02 | Padrão DDD (Domain Driven Design) para organização dos pacotes internos | Padrão |
| RNF03 | Funcionamento sobre a plataforma JRE 11 | Plataforma |
| RNF04 | Relatório semanal | Documentação |

# Equipamentos e Tecnologias necessárias (Materiais e métodos)

Para o desenvolvimento dessa solução será necessário: Um banco de dados MySQL para persistência dos dados, JDK 11 para o desenvolvimento da aplicação por meio da linguagem Java 11, um módulo auxiliar em Python ou Java para *web scrapping* para gerar repositórios de teste com base nos sites disponíveis; Para o controle de versão é necessário Git, e para manter o repositório é necessário o Github.