Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Infraplanning: Uma plataforma web para desenvolvimento em PDDL

Autor: Antonio Ruan Moura Barreto
Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF
2023

Antonio Ruan Moura Barreto

Infraplanning: Uma plataforma web para desenvolvimento em PDDL

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF

2023

Antonio Ruan Moura Barreto
Infraplanning: Uma plataforma web para desenvolvimento em PDDL/ Antonio Ruan Moura Barreto. – Brasília, DF, 2023-
45 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Bruno César Ribas

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2023.

1. SAT. 2. PDDL. I. Prof. Dr. Bruno César Ribas. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Infraplanning: Uma plataforma web para desenvolvimento em PDDL

CDU 02:141:005.6

Antonio Ruan Moura Barreto

Infraplanning: Uma plataforma web para desenvolvimento em PDDL

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 18 de dezembro de 2023:

Prof. Dr. Bruno César Ribas
Orientador

Prof. Dr. John Lenon Cardoso Gardenghi
Convidado 1

Prof. Dr. Renato Coral Sampaio
Convidado 2

Brasília, DF
2023

Resumo

Este estudo tem como objetivo simplificar o acesso a planejadores automatizados para a linguagem PDDL (*Problem Domain Description Language*) por meio de um ambiente web interativo. A linguagem PDDL oferece uma descrição detalhada de problemas, e a utilização de um planejador possibilita a geração de um plano de ações para resolver o problema descrito. A escolha do planejador é de extrema importância para o desempenho de projetos desenvolvidos em PDDL. O modelo proposto visa disponibilizar uma plataforma online na qual os usuários podem criar projetos em PDDL, explorar diversas opções de planejadores para comparar seu desempenho e obter um plano de ações que atenda às necessidades específicas do problema em questão, sempre que viável. Este ambiente web interativo oferece uma experiência amigável e acessível, proporcionando aos usuários uma interface intuitiva para desenvolver, testar e comparar soluções para problemas expressos em PDDL.

Palavras-chave: SAT. Planejamento automatizado. PDDL.

Abstract

This study aims to streamline access to automated planners for the PDDL (Problem Domain Description Language) through an interactive web-based environment. PDDL offers a detailed description of problems, and employing a planner enables the generation of action plans to address the described problem. The choice of planner significantly impacts the performance of projects developed in PDDL. The proposed model seeks to provide an online platform where users can create projects in PDDL, explore various planner options to compare their performance, and obtain an action plan that meets specific problem requirements whenever feasible. This interactive web environment offers a user-friendly and accessible interface, empowering users to develop, test, and compare solutions for problems expressed in PDDL

Key-words: SAT. Automated planning. PDDL.

Lista de abreviaturas e siglas

SAT	Satisfatibilidade Booleana
PDDL	Planning Domain Definition Language
FNC	Forma Normal Conjuntiva

Sumário

Introdução

Na área de planejamento automatizado, a linguagem PDDL (*Problem Domain Description Language*) oferece uma abordagem simples e intuitiva para resolver problemas. No entanto, a execução de planejadores em projetos baseados em PDDL pode apresentar desafios, tanto para novos usuários devido à dificuldade na configuração e execução dos planejadores, quanto para usuários experientes, já que cada planejador possui suas peculiaridades e o usuário precisa testar vários para avaliar seu desempenho em projetos específicos. Nesse contexto, o presente trabalho porpõe o Infraplanning: um editor de código fonte em uma interface web integrada a um conjunto de máquinas. Essa abordagem visa garantir disponibilidade e oferecer uma variedade de planejadores, enriquecendo a experiência do usuário. O corpo do trabalho está apresentado no Anexo A.

Anexos

ANEXO A – Artigo

Infraplanning: Uma plataforma web para desenvolvimento em PDDL

Antonio Ruan Moura Barreto Orientador: Prof. Dr. Bruno César Ribas

Resumo

Este estudo tem como objetivo simplificar o acesso a planejadores automatizados para a linguagem PDDL (*Problem Domain Description Language*) por meio de um ambiente web interativo. A linguagem PDDL oferece uma descrição detalhada de problemas, e a utilização de um planejador possibilita a geração de um plano de ações para resolver o problema descrito. A escolha do planejador é de extrema importância para o desempenho de projetos desenvolvidos em PDDL. O modelo proposto visa disponibilizar uma plataforma online na qual os usuários podem criar projetos em PDDL, explorar diversas opções de planejadores para comparar seu desempenho e obter um plano de ações que atenda às necessidades específicas do problema em questão, sempre que viável. Este ambiente web interativo oferece uma experiência amigável e acessível, proporcionando aos usuários uma interface intuitiva para desenvolver, testar e comparar soluções para problemas expressos em PDDL.

Palavras-chave: SAT. Planejamento automatizado. PDDL.

Abstract

This study aims to streamline access to automated planners for the PDDL (Problem Domain Description Language) through an interactive web-based environment. PDDL offers a detailed description of problems, and employing a planner enables the generation of action plans to address the described problem. The choice of planner significantly impacts the performance of projects developed in PDDL. The proposed model seeks to provide an online platform where users can create projects in PDDL, explore various planner options to compare their performance, and obtain an action plan that meets specific problem requirements whenever feasible. This interactive web environment offers a user-friendly and accessible interface, empowering users to develop, test, and compare solutions for problems expressed in PDDL.

***Keywords**: SAT. Automated planning. PDDL.*

1 Introdução

Os planejadores automatizados são ferramentas essenciais para a resolução de problemas complexos, exigindo anotações sobre quais ações devem ser tomadas para atingir determinados objetivos ou em quais circunstâncias executar ações compostas (Ghallab et al., 1998). A linguagem PDDL (*Planning Domain Definition Language*) foi desenvolvida para formalizar essas anotações e fornecer uma representação precisa e estruturada dos problemas de planejamento.

No entanto, a diversidade de planejadores disponíveis, cada um com suas próprias configurações e instalações específicas, pode ser intimidadora para usuários que não estão familiarizados com essa área. A complexidade e a falta de padronização podem dificultar o uso eficiente e eficaz dessas ferramentas.

Diante desse contexto, este trabalho tem como objetivo a criação de uma plataforma online que ofereça uma ampla variedade de planejadores disponíveis, proporcionando aos usuários uma interface padronizada e intuitiva para executar planejadores e desenvolver problemas em PDDL.

A criação dessa plataforma online com uma ampla gama de planejadores disponíveis beneficiará tanto os usuários iniciantes, que terão acesso facilitado a ferramentas de planejamento automatizado, quanto os usuários experientes, que poderão explorar diferentes abordagens e comparar o desempenho dos planejadores em seus problemas específicos.

Além disso, é essencial que essa plataforma seja capaz de operar em um ambiente de laboratório com vários servidores, distribuindo as tarefas entre as máquinas disponíveis. Isso garantirá uma execução eficiente das tarefas de planejamento, otimizando o desempenho e a escalabilidade da plataforma em um laboratório com infraestrutura distribuída.

Espera-se que essa plataforma online contribua para a disseminação e a utilização mais ampla do planejamento automatizado, proporcionando aos usuários uma maneira fácil e acessível de desenvolver e resolver problemas em PDDL.

Este texto explora a praticidade e alguns conceitos fundamentais da linguagem PDDL em discussão. Além disso, apresenta trabalhos correlatos que oferecem um editor de código para PDDL, fornecendo uma maneira simplificada de executar os projetos criados pelos usuários. Por fim, propõe um modelo para o sistema do Infraplanning e discute os resultados obtidos por meio da implementação desse sistema, incluindo comparações com outros trabalhos correlatos.

2 PDDL

A linguagem PDDL (*Planning Domain Definition Language*) tem como objetivo expressar a "física" de um domínio, ou seja, descrever os predicados existentes, as ações possíveis, a estrutura das ações compostas e os efeitos das ações (Ghallab et al., 1998). Essa notação possibilita que os sistemas de planejamento interpretem e forneçam um plano que resolva o problema de forma eficiente.

Para demonstrar as vantagens de utilizar uma linguagem de planejamento automatizado como a linguagem PDDL, vamos primeiro utilizar somente satisfatibilidade Booleana para resolver um problema no qual um cozinheiro novato decide abrir sua própria hamburgueria, porém com um baixo orçamento ele precisa atender todos seus clientes

com a pouca variedade de ingredientes que conseguiu comprar. Os ingredientes serão representados por variáveis, aqui está a lista de ingredientes a disposição do cozinheiro:

- P: Pão normal
- B: Pão brioche
- C: Carne bovina
- V: Carne vegetariana
- Q: Queijo
- A: Alface
- M: Mostarda
- K: Ketchup

A montagem de um hambúrguer é relativamente simples, precisamos de um tipo de pão, carne bovina ou carne vegetariana como proteína, queijo ou alface para acompanhamento e mostrada ou ketchup como molho. Nada impede o cozinheiro de colocar ambas as opções de pão, proteína, acompanhamento ou molho, a menos que o cliente deixe isso explicito na sua preferência.

Este problema será representado por uma fórmula Booleana na Forma Normal Conjuntiva (FNC), uma fórmula formada por conjunções de cláusulas, onde essas cláusulas por sua vez podem ser uma disjunção de literais ou simplesmente um literal. Uma fórmula Booleana é considerada satisfatível se houver alguma atribuição dos valores 0 e 1 às suas variáveis que resulte em um valor de avaliação igual a 1 (Cormen et al., 2009). No contexto do nosso exemplo, a Equação 1 representa os requisitos necessários para a preparação do nosso hambúrguer.

$$(P \vee B) \wedge (C \vee V) \wedge (Q \vee A) \wedge (M \vee K) \tag{1}$$

Para este cenário de satisfatibilidade Booleana o cozinheiro precisa atender três clientes, todos com restrições alimentares específicas, e com um estoque limitado de ingredientes. Com a inclusão desses clientes, o número de variáveis triplica, e tanto a escassez de ingredientes como as restrições alimentares exigem a adição de cláusulas adicionais para resolver o problema.

Das restrições alimentares de cada um dos três clientes, os dois primeiros não podem comer pão normal e evitam a combinação de pão brioche com alface (representado pelas Equações 2 e 3). O terceiro cliente, por sua vez, não aprecia a junção de pão brioche com alface, mas não tem restrições em relação aos tipos de pães (como visto na Equação 4). Na cozinha, alguns ingredientes estão em quantidade limitada: há apenas um pão normal e dois queijos disponíveis, enquanto os demais ingredientes não possuem restrições (Equação 5).

$$\neg P_1 \wedge B_1 \wedge (C_1 \vee V_1) \wedge Q_1 \wedge \neg A_1 \wedge (M_1 \vee K_1) \tag{2}$$

$$\neg P_2 \wedge B_2 \wedge (C_2 \vee V_2) \wedge Q_2 \wedge \neg A_2 \wedge (M_2 \vee K_2) \tag{3}$$

$$(P_3 \vee B_3) \wedge (C_3 \vee V_3) \wedge (Q_3 \vee A_3) \wedge (M_3 \vee K_3) \wedge (\neg B_3 \vee \neg A_3) \tag{4}$$

$$(\neg Q_1 \vee \neg Q_2 \vee \neg Q_3) \wedge (\neg P_1 \vee \neg P_2) \wedge (\neg P_1 \vee \neg P_3) \wedge (\neg P_2 \vee \neg P_3) \tag{5}$$

Após resolver as equações acima, as variáveis Booleanas positivas identificadas foram: B_1, C_1, Q_1, M_1, B_2, C_2, Q_2, M_2, P_3, C_3, A_3, M_3. Com base nisso, foi possível criar dois hambúrgueres de pão brioche, carne, queijo e mostarda para os dois primeiros clientes, e um hambúrguer de pão normal, carne, alface e mostarda para o terceiro cliente.

Esse exemplo ilustra como um problema aparentemente simples se torna complexo com a inclusão de novos parâmetros, ampliando consideravelmente o espaço de combinações possíveis. A seguir descreveremos este mesmo problema do cozinheiro em PDDL, porém com um objetivo diferente a ser almejado pelo cozinheiro.

O objetivo é demostrar a notação utilizada na linguagem PDDL, e como ela permite representar problemas de planejamento de uma forma concisa e mais clara do que a forma normal conjuntiva demonstrada nas Equações de 1 a 5. Além disso, a linguagem PDDL é amplamente utilizada e suportada por várias ferramentas e sistemas de planejamento automatizado, o que facilita a implementação e a interação com solucionadores de problemas específicos.

2.1 Domínio

A física de um domínio, mencionada pela definição de Ghallab et al. (1998), pode ser formalmente representada por um domínio em PDDL. Essa representação abrange as ações possíveis que podem ser realizadas dentro do escopo do problema em questão. Além das ações, a parte do domínio também especifica os requisitos necessários para a execução de cada ação, bem como os efeitos resultantes quando uma ação é concluída com sucesso, e os predicados que definem as propriedades dos objetos.

No domínio são utilizados diferentes blocos, cada bloco desempenha um papel essencial na descrição precisa e completa do problema, permitindo que um planejador PDDL identifique uma sequência de ações que leva à resolução do objetivo proposto. A seguir, será detalhada a função de cada bloco e sua importância na representação e resolução do problema do cozinheiro em PDDL.

2.1.1 Requisitos

O trecho de código apresentado no Código 1 inicia abrindo o bloco *define*, delimitando o escopo do domínio. Em seguida, o nome do domínio é definido, seguido pelos requisitos necessários para sua execução. O conjunto de requisitos de um domínio permite que um planejador determine prontamente se é viável lidar com o referido domínio (Ghallab et al., 1998).

Código 1 – Requisitos para o problema do cozinheiro

```
1 (define
2     (domain hamburguer)
3     (:requirements :strips :typing :negative-preconditions)
```

Para o nosso exemplo são necessários os seguintes requisitos:

- ***strips*:** Permite a utilização de efeitos para uma ação (as ações serão abordadas em breve).

- ***typing*:** Permite a utilização de categorização para objetos. A categorização se assemelha a classes e subclasses na programação orientada a objetos.
- ***negative-preconditions*:** Incorporada na versão 2.1 do PDDL, possibilita a utilização do operador de negação nas pré-condições, um recurso que será abordado posteriormente.

2.1.2 Tipos

Os tipos são responsáveis por definir os objetos manipulados no domínio. Utilizando uma analogia à programação orientada a objetos, podemos estabelecer uma classe base denominada "ingrediente", na qual são definidas subclasses para representar os diferentes tipos de ingredientes pertinentes ao problema do cozinheiro, tais como pão, carne, acompanhamento e molho. Esses tipos estão descritos em PDDL no Código 2.

Código 2 – Tipos para o problema do cozinheiro

```
1     (:types
2         pao carne acompanhamento molho - ingrediente)
```

2.1.3 Predicados

O campo de predicados consiste em uma lista de declarações de predicados, utilizando a sintaxe de lista tipada para declarar os argumentos de cada um (Ghallab et al., 1998). No contexto do nosso exemplo, são necessários apenas dois predicados. O predicado "ingrediente-consumido" é utilizado para indicar a indisponibilidade de um ingrediente específico, enquanto o predicado "hamburguer-pronto" é utilizado para sinalizar que um determinado hambúrguer está pronto. Esses predicados descritos em PDDL podem ser observados no Código 3.

Código 3 – Predicados para o problema do cozinheiro

```
1     (:predicates
2         (ingrediente-consumido ?i - ingrediente)
3         (hamburguer-pronto ?p - pao ?c - carne ?a - acompanhamento ?m -
              molho))
```

2.1.4 Ação

Uma ação em PDDL permite definir a interação entre as entidades do domínio, os requisitos necessários para executar a ação e como o estado do mundo é modificado por meio dessas ações. É por meio das ações que um planejador em PDDL pode determinar uma sequência de ações que leva a um objetivo específico a partir de um estado inicial.

No contexto do problema do cozinheiro, podemos começar com duas ações simples. A primeira ação é a de preparar o hambúrguer, que requer apenas a disponibilidade dos ingredientes e resulta em um hambúrguer pronto, além do consumo dos ingredientes. A segunda ação é a de comprar ingredientes, que só é realizada quando há falta de algum ingrediente e resulta na compra do ingrediente necessário para o preparo. Essas duas ações podem ser observadas no Código 4, junto com o fechamento do bloco *define* que foi aberto no Código 1.

Código 4 – Ações para o problema do cozinheiro

```
    (:action fazer-hamburguer
        :parameters (?p - pao ?c - carne ?a - acompanhamento ?m - molho)
        :precondition (and
            (not (ingrediente-consumido ?p))
            (not (ingrediente-consumido ?c))
            (not (ingrediente-consumido ?a))
            (not (ingrediente-consumido ?m))
        )
        :effect (and
            (ingrediente-consumido ?p)
            (ingrediente-consumido ?c)
            (ingrediente-consumido ?a)
            (ingrediente-consumido ?m)
            (hamburguer-pronto ?p ?c ?a ?m)
        )
    )

    (:action comprar-ingrediente
      :parameters (?i - ingrediente)
      :precondition (ingrediente-consumido ?i)
      :effect (not (ingrediente-consumido ?i))
    )
)
```

2.2 Problema

Avançando no detalhamento dos blocos em PDDL, adentramos à etapa de formulação do problema. O problema define o objetivo que o planejador busca resolver, conforme discutido por Ghallab et al. (1998). Nesta fase, concebemos uma instância do domínio previamente demonstrado, especificando os recursos à nossa disposição e delineando claramente nosso objetivo final. Essa etapa é fundamental para a concepção de um planejamento com as ações necessárias para alcançar o objetivo almejado. Em alguns cenários, é viável otimizar o plano gerado ao definir uma métrica a ser maximizada, permitindo uma abordagem mais refinada e eficiente na resolução do problema em questão.

2.2.1 Objetos

No Código 5, é utilizada novamente a expressão *define*, desta vez para estabelecer o escopo do problema. Em seguida, o problema é nomeado e referenciado ao domínio correspondente. Outro campo apresentado é o campo *objects*, no qual declaramos todos os objetos que fazem parte desse domínio, especificando seus tipos.

Código 5 – Objetos para o problema do cozinheiro

```
(define
    (problem hamburguer001)
    (:domain hamburguer)
    (:objects
      pao-normal - pao
      brioche - pao
      bovina - carne
      vegetariana - carne
      queijo - acompanhamento
      alface - acompanhamento
```

```
11        mostarda - molho
12        ketchup - molho)
```

2.2.2 Init

Nesta seção, descrevemos o estado inicial do problema. Como um domínio pode ter vários problemas, cada problema descreve um estado inicial específico para uma determinada instância. No Código 6, é declarado um estado inicial no qual apenas o molho mostarda está indisponível para o preparo do hambúrguer.

Código 6 – Inicialização do problema do cozinheiro

```
1     (:init
2       (ingrediente-consumido mostarda)
3     )
```

2.2.3 Objetivo

Nesta etapa, é declarado o objetivo desejado que o planejador deve alcançar durante a resolução da instância do problema. Após encontrar uma sequência de ações que leva ao objetivo, o planejador retorna as ações executadas para concluir a instância do problema.

No Código 7, é apresentado o objetivo de preparar dois tipos de hambúrgueres. O primeiro objetivo é preparar um hambúrguer com pão brioche, carne bovina, alface e mostarda. O segundo objetivo é preparar um hambúrguer com pão brioche, carne vegetariana, alface e ketchup.

Código 7 – Objetivo para o problema do cozinheiro

```
1     (:goal (and
2       (hamburguer-pronto brioche bovina alface mostarda)
3       (hamburguer-pronto brioche vegetariana alface ketchup)
4     ))
5 )
```

Esses objetivos específicos fornecem uma direção clara para o planejador em termos dos ingredientes e condimentos necessários para alcançar cada tipo de hambúrguer. A partir dessas informações, o planejador pode identificar as ações apropriadas a serem executadas e determinar a sequência de passos necessários para atingir os objetivos propostos.

2.3 Planejador

A linguagem PDDL possui uma variedade de planejadores que são capazes de analisar as ações disponíveis no domínio, levando em consideração suas pré-condições. O objetivo desses planejadores é construir um plano que atenda aos requisitos específicos do problema. Eles podem utilizar algoritmos de busca e técnicas de planejamento para explorar o espaço de estados e encontrar uma sequência de ações que leva ao estado objetivo desejado.

2.3.1 Julia Planners

Embora não seja um planejador em si, a linguagem de programação Julia oferece uma variedade de pacotes que podem ser aplicados em diversos domínios de planejamento automatizado. Julia combina conhecimentos de diversos campos da ciência da computação e da ciência computacional para criar uma nova abordagem para a computação numérica (Bezanson et al., 2017).

Dentre os pacotes disponíveis na linguagem Julia, destacam-se dois na área de planejamento. O PDDL.jl é uma interface extensível de interpretador e compilador para planejamento de IA rápido e flexível (Zhi-Xuan, 2022). Com o PDDL.jl, é possível interpretar ou compilar projetos em PDDL para um formato adequado aos algoritmos de planejamento.

Outro pacote relevante para PDDL é o SymbolicPlanners.jl (Zhi-Xuan, 2023). Ao ser combinado com o PDDL.jl, o SymbolicPlanners.jl permite resolver problemas descritos em PDDL, fornecendo uma variedade de heurísticas e especificações. Algumas dessas especificações incluem minimizar o número de ações para atingir o objetivo, minimizar ou maximizar uma métrica ao alcançar o objetivo e adicionar restrições de estados.

2.3.2 Fast Downward

Fast Downward é um sistema de planejamento clássico baseado em busca heurística (Helmert, 2006). Ele utiliza uma combinação de técnicas, como busca em largura, busca em profundidade e busca heurística informada, para resolver problemas de planejamento de forma eficiente. O sistema é altamente modular e flexível, o que permite sua configuração e adaptação para diferentes domínios e requisitos específicos.

O Fast Downward oferece uma variedade de configurações prontas para uso, que podem ser acessadas através de apelidos específicos. Por exemplo, a configuração apelidada como “seq-sat-lama-2011” foi a configuração utilizada na participação do sistema na Competição Internacional de Planejamento realizada em 2011. Essas configurações prontas demonstram a versatilidade do Fast Downward e sua capacidade de enfrentar diferentes desafios de planejamento.

2.3.3 Madagascar

O sistema de planejamento Madagascar implementa várias inovações no planejamento com SAT, incluindo heurísticas SAT especializadas para planejamento e estruturas de dados que suportam a resolução paralelizada de SAT com instâncias de problemas muito grandes (Rintanen, 2014). O Madagascar utiliza um resolvedor SAT próprio e implementa diversas heurísticas para otimizar o desempenho do planejador em determinadas instâncias de problema.

O Madagascar oferece três configurações principais: M, que utiliza SAT com a heurística VSIDS (*Variable State Independent Decaying Sum*) para resolver problemas de planejamento; Mp, com paralelização de busca para melhorar o desempenho; e MpC, que combina paralelização de busca e compactação eficiente de planos para escalabilidade em problemas de grande porte. Cada configuração apresenta vantagens específicas em termos de resolução de problemas de planejamento.

3 Trabalhos Correlatos

No âmbito do desenvolvimento de projetos PDDL em plataformas Web, dois trabalhos proeminentes merecem destaque: o Planning Domains e o Web Planner. O Planning Domains destaca-se como uma ferramenta ágil na definição e formalização de domínios para o planejamento automatizado. Por outro lado, o Web Planner, um software proprietário concebido por alunos da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), representa uma notável contribuição nesse cenário. Essas plataformas oferecem recursos para a criação e manipulação de problemas e domínios em PDDL diretamente online, permitindo aos usuários desenvolverem e testarem modelos de planejamento de forma acessível e prática. Nesta seção, examinaremos em detalhes esses dois trabalhos, explorando sua influência e características específica para o desenvolvimento de projetos em PDDL em plataformas Web.

3.1 Planning Domains

O Planning Domains é uma plataforma online desenvolvida por Christian Muise, professor assistente na *Queen's University* em Kingston, Canadá. O objetivo principal é fornecer um conjunto de recursos, repositórios e ferramentas para que os pesquisadores descubram e desenvolvam problemas de planejamento, além de ser um meio de divulgação para outras comunidades não familiarizadas com a tecnologia de planejamento (Muise, 2016).

3.1.1 Arquitetura

A arquitetura do Planning Domains foi projetada para fornecer uma solução completa e flexível para a realização de planejamento, ela é dividida entre três componentes que se complementam. O primeiro componente é uma API abrangente que oferece uma variedade de problemas de planejamento, incluindo aqueles utilizados na Competição Internacional de Planejamento. Essa API permite aos usuários acessar e utilizar diferentes cenários de planejamento para fins de teste e pesquisa.

O segundo componente consiste em um planejador em nuvem dedicado a proporcionar uma plataforma acessível para resolver problemas de planejamento. Com essa solução, os usuários podem executar seus algoritmos de maneira rápida e fácil, sem a necessidade de configurações adicionais. A simplicidade é garantida, pois o usuário interage apenas com o seu projeto em PDDL, sem se preocupar com detalhes de configuração. Isso torna o processo de execução dos algoritmos mais eficiente e conveniente.

O terceiro componente é o editor PDDL, que é uma ferramenta essencial para criar e editar problemas de planejamento. O editor oferece recursos básicos, como destaque de sintaxe e preenchimento automático, para facilitar a escrita correta do código PDDL. Além disso, o editor do Planning Domains possui recursos adicionais que simplificam o trabalho com PDDL, como extensões que ampliam suas funcionalidades e a possibilidade de adicionar uma URL para um planejador externo de PDDL, caso o usuário prefira outro planejador que não o do próprio Planning Domains. Com isso os usuários podem testar diferentes algoritmos e estratégias de resolução para otimizar seus planos.

Para exemplificar a utilidade das extensões, o editor possui a extensão Torchlight (Hoffmann, 2011). Ela oferece recursos de análise avançada de problemas de planejamento,

previsão de desempenho do planejador por meio de aprendizado de máquina com base nas características geradas, além de reformular automaticamente do domínio usando as características geradas como orientação para reformulação. Ele permite aos usuários examinar a estrutura do problema, identificar possíveis falhas na formulação do domínio e da instância. Essa ferramenta é especialmente valiosa durante a fase de depuração e refinamento de modelos PDDL.

Devido à natureza intensiva em recursos de memória e processamento dos problemas de planejamento, o Planning Domains estabelece limites de dez segundos de processamento e quinhentos megabytes de memória principal para o uso de seu planejador interno. No entanto, é importante ressaltar que domínios complexos ou mal otimizados podem facilmente ultrapassar esses limites. Para contornar essa restrição, é necessário recorrer a um planejador externo, que ofereça maior flexibilidade em termos de recursos de processamento e memória disponíveis.

Outro problema importante a ser relatado é a possível indisponibilidade de algumas extensões no Planning Domains. Isso ocorre porque algumas extensões são hospedadas por terceiros ou pelo próprio criador da extensão, em vez de estarem diretamente incorporadas à plataforma do Planning Domains. Essa dependência de terceiros pode afetar a disponibilidade e a atualização das extensões.

É importante ressaltar que todos os componentes mencionados são softwares proprietários. No entanto, o Planning Domains disponibiliza um repositório público que oferece algumas ferramentas específicas para interagir com esses componentes. Embora os componentes principais sejam proprietários, a disponibilidade dessas ferramentas adicionais amplia as possibilidades de interação e customização para atender às necessidades dos usuários.

3.1.2 Extensão para VS Code

Há contribuições externas significativas feitas por terceiros que estendem as funcionalidades do Planning Domains. Um desses exemplos é a extensão para o editor VS Code desenvolvida por Jan Dolejsi, chamada PDDL. Esta extensão transforma o VS Code em um ambiente excelente para modelagem de domínios de planejamento, equiparando a linguagem PDDL ao suporte oferecido a linguagens proeminentes como C#, Python ou JavaScript (Dolejsi, 2023).

Esta extensão integra várias funcionalidades com o Planning Domains. Por exemplo, permite a navegação direta e exibição de domínios e problemas da API do Planning Domains, além do uso de sessões. As sessões do Planning Domains permitem que os usuários salvem e continuem seu trabalho em PDDL, e essa extensão utiliza essa funcionalidade para salvar e carregar sessões diretamente no VS Code.

Outra integração importante ocorre durante a execução do planejador. Entre as opções de planejadores disponíveis, a extensão inclui o planejador do Planning Domains pronto para uso. Além disso, os usuários podem integrar outros planejadores a essa extensão, seja por meio de um arquivo binário ou fornecendo um link para o planejador.

Essas integrações externas ressaltam a extensibilidade e versatilidade do ecossistema do Planning Domains, oferecendo a oportunidade para contribuições e personalizações por parte da comunidade. A capacidade de integrar diferentes ferramentas e funcionalidades permite que o Planning Domains atenda a diversas demandas e se adapte às necessidades

específicas dos usuários, tornando-se um ambiente versátil e flexível para o desenvolvimento e execução de projetos em PDDL.

3.2 Web Planner

O Web Planner é um *software* proprietário desenvolvido por alunos da Pontifícia Universidade Católica do Rio Grande do Sul, com objetivo de simplificar o processo de configuração necessário para executar planejadores, ao mesmo tempo em que fornece visualizações para entender melhor como as diferenças de domínio e as heurísticas podem impactar o desempenho do planejador (Magnaguagno et al., 2017).

Assim como o Planning Domains, o Web Planner traz uma plataforma online com funcionalidades que agregam no desenvolvimento de domínio em PDDL para novos usuários ou para usuários experientes. Tanto o Web Planner quanto o Planning Domains oferecem recursos de visualização que permitem aos usuários compreender como as características do domínio e as heurísticas podem afetar o desempenho do planejador.

Dos recursos oferecidos pelo Web Planner há dois tipos de visualização, uma focada no espaço de estados explorado pelo planejador e outra na execução do primeiro plano encontrado. Uma das visualizações é a heurística que utiliza layouts de árvore para representar o espaço de estados explorado durante o processo de planejamento, assim oferecendo uma maneira intuitiva de compreender como as heurísticas influenciam o processo de planejamento, permitindo a identificação de padrões e a análise de resultados de diferentes heurísticas utilizadas. As informações detalhadas sobre cada estado são exibidas em uma dica de ferramenta ao passar o cursor sobre o estado que é representado como um nó. A visualização também destaca as arestas que levam do estado inicial ao objetivo em casos de planejamento bem-sucedido.

Outra visualização de dados oferecida é a visualização Dovetail, ilustrada na Figura 1. A visualização Dovetail é uma metáfora visual implementada para mostrar como os predicados mudam durante a execução do plano Magnaguagno et al. (2017). Na representação visual proposta, cada predicado contido nos efeitos das ações é retratado como uma linha, onde uma peça de quebra-cabeça ilustra o estado Booleano do próprio predicado. Por sua vez, os estados inicial e objetivo, juntamente com as ações, são dispostos como colunas nessa estrutura visual.

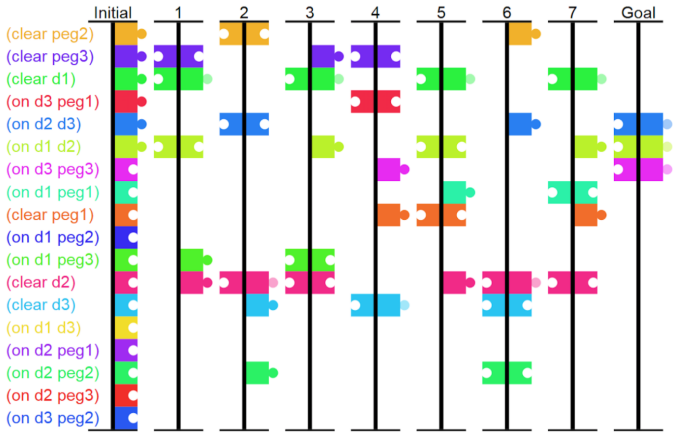

Figura 1 – Visualização Dovetail para o problema da torre de hanoi com três discos. Fonte: Magnaguagno et al. (2017)

O Editor do Web Planner, além de trazer algumas funcionalidades presentes no Planning Domains como o preenchimento automático e o destaque da sintaxe PDDL, traz também a implementação de modelos para blocos da linguagem PDDL mostrando como o bloco é definido. O Editor é dividido em 3 colunas, uma para a edição do domínio, outra para edição do problema e uma terceira para a visualização do plano gerado. Essa capacidade de visualização permite uma melhor compreensão do processo, tornando mais fácil para os usuários entenderem a relação entre os dados de entrada e os resultados obtidos.

A plataforma é desenvolvida para auxiliar o usuário em duas tarefas: a descrição correta do domínio e do problema, e a identificação dos detalhes que afetam o desempenho do planejamento. Para isso é fornecida uma arquitetura com dois serviços, como mostrada na Figura 2. O Navegador do usuário é responsável por enviar o projeto em PDDL para o servidor, que por sua vez executa um planejador, e retorna o plano e dados sobre o plano para que o navegador consiga reproduzir as visualizações dos dados.

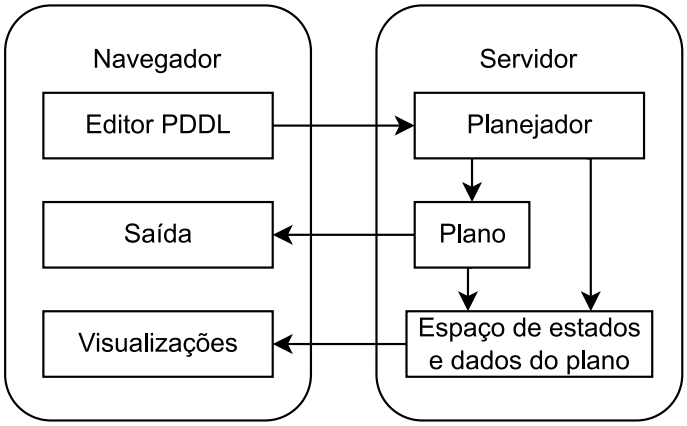

Figura 2 – Arquitetura do Web Planners. Fonte: Magnaguagno et al. (2017)

Mesmo que o planejador não retorne um plano para a instancia do problema, o usuário consegue visualizar o espaço de estados explorados pelo planejador. Para que aconteça essa comunicação entre os serviços os dados são transferidos no formato JSON.

O editor online de PDDL e da extensão Dovetail foi previamente mantida de forma gratuita no Heroku, uma plataforma de hospedagem em nuvem. No entanto, a partir de 28 de agosto de 2022, o Heroku deixou de ser gratuito, resultando na interrupção dos serviços oferecidos pelo Web Planner. Essa situação torna-se mais preocupante devido à natureza proprietária do código fonte do Web Planner. Isso significa que, para acessar novamente esses serviços, dependemos exclusivamente da decisão do proprietário em realocar o serviço para outra plataforma de hospedagem ou contrate algum plano do Heroku.

4 Modelo Proposto

Este segmento destina-se a documentar o trabalho realizado para conceber uma plataforma web, denominada Infraplanning, destinada a facilitar o desenvolvimento de projetos na linguagem PDDL e a execução de planejadores para a mesma. O foco principal deste segmento abrange os métodos e tecnologias empregados ao longo do desenvolvimento da plataforma.

4.1 Requisitos

A fim de iniciar esta análise, é essencial definir os perfis de usuários-alvo do Infraplanning, considerando seus diferentes níveis de familiaridade com a linguagem PDDL, bem como os objetivos específicos de cada categoria de usuário. A plataforma foi desenvolvida para atender a três perfis distintos:

- **Usuário leigo:** usuário com pouca ou nenhuma experiência na linguagem PDDL e seus planejadores. Esse usuário demanda uma interface intuitiva capaz de realizar a execução simplificada de um planejador em poucos passos, além de listar configurações prontas para cada planejador, independentemente da escolha do planejador.
- **Usuário avançado:** usuário que já desenvolveu alguns projetos em PDDL e já utilizou de planejadores para executar seu problema. Este usuário busca uma ampla variedade de opções de planejadores e configurações para comparar o desempenho de cada configuração em seu projeto PDDL.
- **Usuário administrador:** usuário que é dispensável o conhecimento com a linguagem PDDL, onde seu papel é administrar a aplicação. Seu objetivo principal é gerenciar como os demais usuários utilizam as máquinas que hospedam a plataforma Infraplanning.

Com os usuários definidos e suas respectivas necessidades identificadas, foi possível elicitar os requisitos que a plataforma Infraplanning precisa atender. Esses requisitos foram abordados como épicos, funcionalidades e histórias de usuário, seguindo uma metodologia comum no desenvolvimento ágil. Épicos, funcionalidades e histórias são formas de expressar as necessidades do usuário e os benefícios implícitos, mas em diferentes níveis de abstração (Leffingwell, 2011).

O Quadro 1 apresenta todos os requisitos identificados por rótulos. Os rótulos dos requisitos seguem uma sigla única, seguida de um número para distinguir entre diferentes tipos de requisitos. As siglas utilizadas são: **E** para Épico, **F** para Funcionalidade e **US** para História de Usuário. Em seguida, esses requisitos são listados com uma breve descrição.

Quadro 1 – Hierarquia dos requisitos

Épico	Funcionalidade	História de Usuário
E1	F1	US1
		US2
		US3
		US4
		US5
	F2	US6
		US7
E2	F3	US8
		US9
	F4	US10
		US11
		US12
		US13
		US14

- **E1:** Desenvolvimento do Projeto PDDL
- **E2:** Administração
- **F1:** Criação do Projeto
- **F2:** Persistência do Projeto
- **F3:** Gerenciamento das Máquinas
- **F4:** Gerenciamento dos Usuários
- **US1:** Como usuário, quero acessar um editor de código-fonte para editar meu projeto.
- **US2:** Como usuário, quero selecionar a máquina, planejador e sua configuração para executar meu projeto, buscando a melhor performance.
- **US3:** Como usuário, quero configurar limites de memória e tempo para interromper a execução do planejador caso exceda algum desses limites.
- **US4:** Como usuário, quero receber informações do planejador em execução para saber o que está acontecendo.
- **US5:** Como usuário, quero cancelar a execução de um planejador para corrigir um erro inesperado.
- **US6:** Como usuário, quero salvar meu projeto para continuar trabalhando posteriormente.
- **US7:** Como usuário, quero carregar projetos salvos para continuar de onde parei.
- **US8:** Como professor, quero habilitar uma máquina para uso dos usuários em seus planejamentos.
- **US9:** Como professor, quero desabilitar uma máquina para preservar seus recursos.
- **US10:** Como professor, quero criar grupos de usuários para fornecer diferentes limites de memória e tempo de processamento dos que já existem.
- **US11:** Como professor, quero visualizar os grupos de usuários existentes para verificar quantos usuários existem nos grupos e seus limites de memória e tempo de processamento.
- **US12:** Como professor, quero modificar os limites de memória e tempo de processamento de um grupo para corrigir alguma inconformidade.
- **US13:** Como professor, quero apagar um grupo de usuário para remover grupos criados por engano ou sem usuários.
- **US14:** Como professor, quero mover um usuário de grupo para alterar seus limites de memória e tempo de processamento.

4.2 Infraestrutura

O levantamento e análise dos requisitos permitem não apenas compreender as funcionalidades essenciais do sistema, mas também mapear os requisitos relacionados à infraestrutura. Esse processo auxilia na identificação das necessidades específicas de hardware, software e tecnologias necessárias para sustentar as operações do sistema.

Para garantir uma alta disponibilidade do sistema, é recomendável implementá-lo em uma infraestrutura composta por várias máquinas capazes de executar planejadores. Com esse objetivo, foi desenvolvido o servidor denominado Trabalhador, cuja principal finalidade é realizar a execução dos planejadores.

É possível criar várias replicas do servidor Trabalhador em máquinas distintas. Para gerenciar essas replicas, foi implementado o servidor Mestre, responsável por atribuir tarefas e coletar respostas do servidor Trabalhador. Toda a comunicação com o servidor Trabalhador é realizada exclusivamente pelo servidor Mestre.

Adicionalmente, foi concebido uma interface Web com uma usabilidade simples e intuitiva. Embora a distribuição de todos esses serviços sejam essenciais para o funcionamento do Infraplanning, o projeto pode ser executado tanto em um sistema distribuído entre vários computadores quanto em um sistema centralizado em uma única máquina. Dito isso, a recomendação da distribuição dos servidores trabalhadores se deve ao alto consumo de recursos por parte do planejado, assim otimizando o desempenho do sistema como um todo. A Figura 3 demonstra a organização dos serviços desenvolvidos.

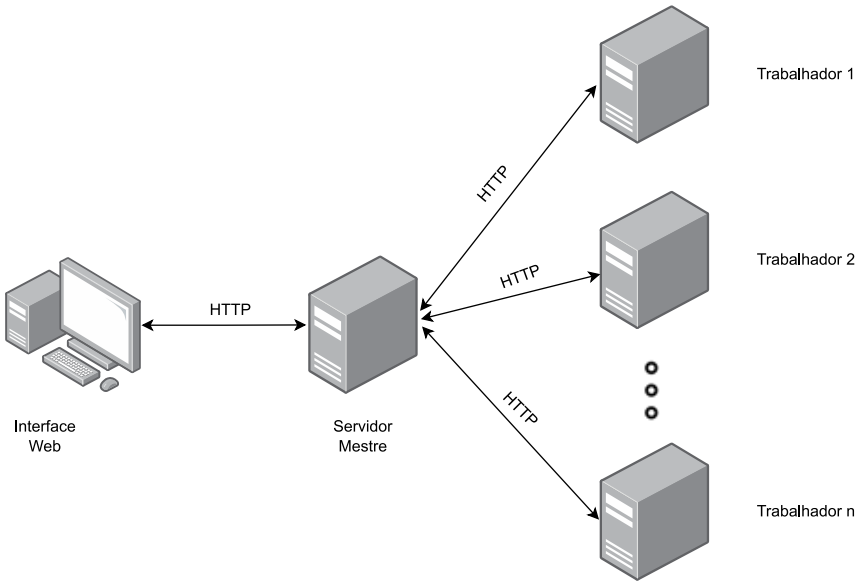

Figura 3 – Diagrama de infraestrutura

A comunicação entre os serviços é estabelecida por meio do protocolo HTTP, no qual os dados são transmitidos no corpo das requisições e respostas em formato JSON, seguindo a mesma abordagem utilizada na plataforma Web Planner. Essa estratégia permite a transferência eficiente e estruturada das informações, garantindo a integridade e compatibilidade dos dados compartilhados entre os diferentes componentes do sistema.

Nas próximas subseções, aprofundaremos nos serviços desenvolvidos para o Infraplanning, detalhando os desafios enfrentados desde a concepção inicial até a implementação

e otimização das funcionalidades. Todo o desenvolvimento foi feito mirando a escalabilidade do sistema, a eficiência na execução dos planejadores e a criação de uma interface de usuário intuitiva.

4.3 Interface Web

A interface Web serve como o principal ponto de interação entre o usuário e o sistema, desempenhando um papel crucial na experiência do usuário. É essencial que essa interface seja intuitiva e de fácil utilização, proporcionando uma navegação fluida e agradável ao usuário. Além disso, é fundamental que seja capaz de lidar com situações inesperadas ou falhas de forma eficaz, garantindo uma recuperação suave e transparente diante de eventuais erros.

Outro objetivo da interface web é o desempenho, por isso a escolha do Vue.js (You, 2023), um *framework* de JavaScript conhecido por sua sintaxe simples e uma boa curva de aprendizado. Além disso, o Vue.js utiliza um sistema de reatividade que atualiza automaticamente a interface do usuário sempre que os dados subjacentes são modificados. Além de possuir um conhecimento prévio da ferramenta, outro aspecto que influenciou a escolha do Vue.js para este trabalho foi a presença de uma comunidade ativa e um ecossistema de pacotes prontos para uso, o que facilita a adição de recursos ao projeto.

Para o gerenciamento desses pacotes, foi utilizado o Yarn (McKenzier, 2023). Desenvolvido pelo Facebook, o Yarn tem como objetivo melhorar a velocidade, segurança e confiabilidade no gerenciamento de dependências. A alta velocidade do Yarn para a instalação de pacotes é alcançada por meio do armazenamento em *cache* e da instalação em paralelo.

Um dos pacotes utilizados neste trabalho foi o Vuetify (Leider and Leider, 2023), uma biblioteca para Vue que disponibiliza uma extensa seleção de componentes prontos para uso, acelerando o processo de desenvolvimento. Tanto o Vue.js quanto o Vuetify são de código aberto e possuem licença MIT. O Yarn também é de código aberto, porém com licença BSD 2, que permite a livre utilização do gerenciador de pacotes, desde que a declaração de direitos autorais seja incluída na documentação ou nos arquivos distribuídos.

Para cumprir a história de usuário **US1** utilizamos o ACE Editor, um editor de código fonte desenvolvido pela empresa Ajax.org. O ACE Editor oferece recursos úteis para o propósito deste trabalho. Embora não tenha suporte nativo à linguagem PDDL, é possível realizar configurações para destacar a sintaxe PDDL, oferecer preenchimento automático de palavras reservadas e palavras já escritas, além de realçar erros para depuração e correção de problemas. Essas funcionalidades do ACE Editor facilitam a escrita e o desenvolvimento do código PDDL, contribuindo para a eficiência e a qualidade do trabalho.

4.4 Servidores Mestre e Trabalhador

Esta subseção destina-se a abordar tanto o servidor mestre quanto o servidor trabalhador. A razão para esta abordagem abrangente está na interdependência de ambos para a realização coordenada de diversas funcionalidades, além da escolha compartilhada das tecnologias empregadas em ambos os serviços.

Tanto o servidor mestre quanto o trabalhador utilizam o FastAPI (Ramírez, 2023), um framework de sintaxe simples para a linguagem Python, compatível com a versão 3.7 ou

posterior. A escolha desse framework foi motivada por suas características, como facilidade de uso, alto desempenho, boa integração com o ecossistema Python e familiaridade prévia com o framework. Esses aspectos agilizam o desenvolvimento da plataforma e facilitam a integração de novos membros ao projeto.

4.4.1 Autenticação

A necessidade de autenticação de usuários na aplicação surgiu em função da existência de diferentes permissões atribuídas a esses usuários. Um dos principais objetivos do Infraplanning é facilitar a execução de planejadores nas máquinas que hospedam o servidor trabalhador. Aproveitando o acesso prévio dos usuários do Infraplanning a essas máquinas, a autenticação é realizada diretamente no nível do sistema operacional, não precisando assim salvar a senha do usuário na aplicação Infraplanning.

Aqueles usuários do Infraplanning que não possuem acesso direto às máquinas ainda podem utilizar o sistema como usuários convidados. No entanto, há limitações no uso dos planejadores, restringidos a 1 Gigabyte de memória RAM e 1 minuto de processamento.

Para viabilizar a autenticação entre as requisições da interface Web e o servidor mestre, foi empregado o uso de JSON Web Tokens (JWT). Este formato compacto de representação de declarações é especialmente projetado para ambientes com recursos limitados, como cabeçalhos de autorização HTTP e parâmetros de consulta URI (Jones et al., 2015).

O uso do JWT elimina a necessidade de armazenar as credenciais do usuário na interface Web, melhorando a segurança da aplicação. O token inclui informações sobre o tempo de expiração, configurado para 2 horas a partir da emissão do token no Infraplanning. Após a expiração, o usuário precisa autenticar-se novamente na aplicação. A Figura 4 ilustra as interações realizada pelo sistema para a autenticação do usuário.

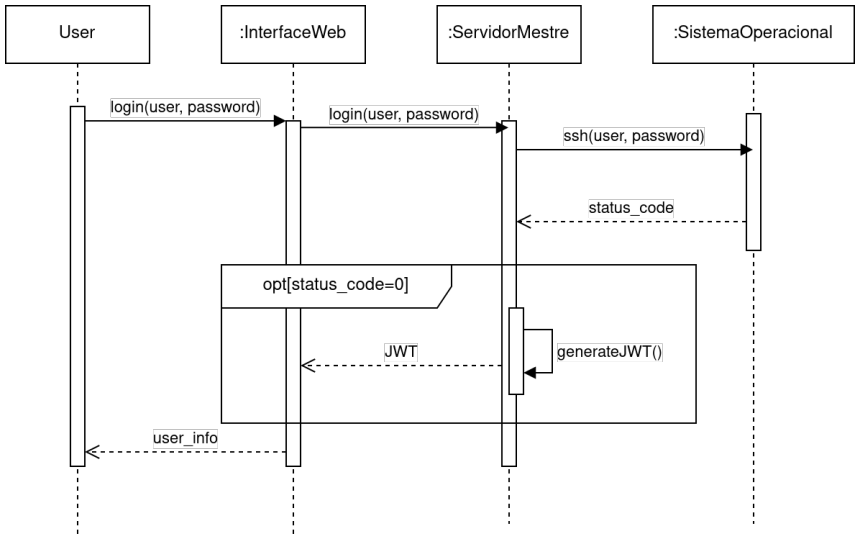

Figura 4 – Diagrama de sequência para autenticação

É importante destacar que as credenciais do usuário não são enviadas por meio do JWT, já que suas informações são codificadas, não criptografadas. Contudo, o token é

assinado com uma chave privada exclusiva do Infraplanning, garantindo assim a validade e autenticidade do token.

4.4.2 Protocolos de comunicação

As histórias de usuário **US4** e **US05** demonstram situações em que o usuário necessita interagir diretamente com o planejador. É essencial que essas interações sejam realizadas em tempo real, simultaneamente à execução do planejador, para garantir o controle e monitoramento contínuos do processo de planejamento. Essas exigências levantaram um desafio significativo: estabelecer uma troca de mensagens fluida entre a interface Web e o servidor trabalhador durante a execução do planejador.

Embora o protocolo HTTP seja amplamente adotado em serviços da Web, oferecendo métodos de requisição específicos para diversas ações do usuário e códigos de resposta definidos para várias situações, seu modelo de requisição-resposta, que permite apenas uma requisição e uma resposta por conexão, torna complexa a implementação das histórias de usuário mencionadas.

Para contornar essa limitação, optamos pelo protocolo WebSocket, uma tecnologia de comunicação bidirecional e em tempo real que opera por meio de um canal único e persistente. O WebSocket viabiliza uma conexão contínua e interativa entre o cliente e o servidor, permitindo a troca de mensagens em ambas as direções sem a necessidade de iniciar uma nova conexão. A Figura 5 ilustra um diagrama de sequências para um usuário que cancela a execução do planejador.

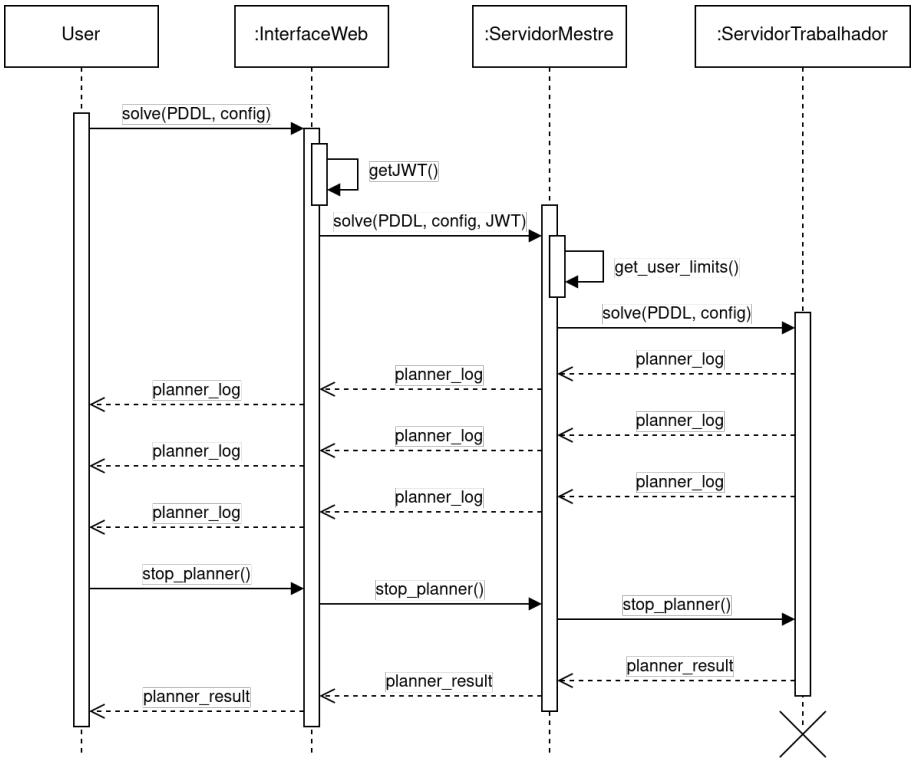

Figura 5 – Diagrama de sequência cancelamento do planejador

4.4.3 Armazenamento do projeto PDDL

Visando a autenticação do usuário do Infraplanning com as credenciais do sistema operacional das máquinas que abrigam o servidor trabalhador, e com o intuito de atender às histórias de usuário **US6** e **US7**, decidiu-se salvar o projeto PDDL diretamente no diretório *home* do próprio usuário do sistema operacional.

Para que o Infraplanning possa realizar a leitura e escrita nos arquivos salvos no diretório *home* do usuário, assim que o usuário realiza o login na aplicação é criada uma pasta denominada ".infraplanning" com permissões de escrita e leitura para todos os usuários pertencentes ao grupo "esw". A Figura 6 ilustra como é feita a gravação e leitura dos arquivos.

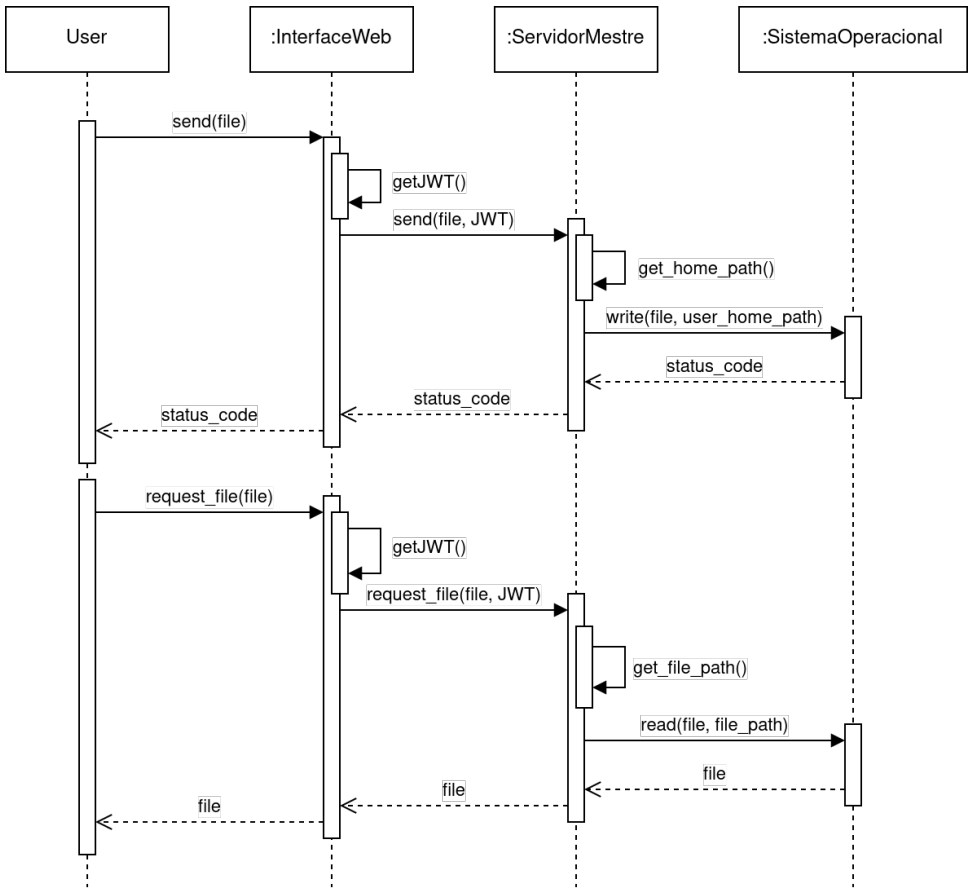

Figura 6 – Diagrama de sequência para gravação e leitura de arquivos

Entretanto, essa abordagem gera um problema significativo ao expor uma vulnerabilidade, permitindo que outros usuários do grupo "esw" possam acessar informações sensíveis ou, até mesmo, injetar códigos maliciosos com o intuito de prejudicar os usuários-alvo.

4.4.4 Padronização dos planejadores

O Infraplanning oferece suporte a três planejadores de PDDL: Julia Planners (2.3.1), Fast Downward (2.3.2) e Madagascar (2.3.3). Todos esses planejadores de PDDL são

desenvolvidos por equipes diferentes e, como resultado, não seguem um padrão único. Eles adotam opções de execução diferentes, como heurísticas e formatos de saída de resultados.

Isso apresenta um desafio significativo para o servidor trabalhador que interage com esses planejadores, pois precisam lidar com várias ferramentas que possuem peculiaridades distintas, buscando obter um resultado consistente e padronizado.

Para superar esse desafio, o servidor trabalhador faz o uso de interface, uma técnica da Programação Orientada a Objetos (veja Figura 7). A interface possui somente métodos públicos e abstratos, estabelecendo um contrato que as classes que a implementam devem seguir. Essa abstração permite que o trabalhador trate os diferentes planejadores de forma uniforme, simplificando a integração e a estabelecendo um resultado padrão para cada planejador.

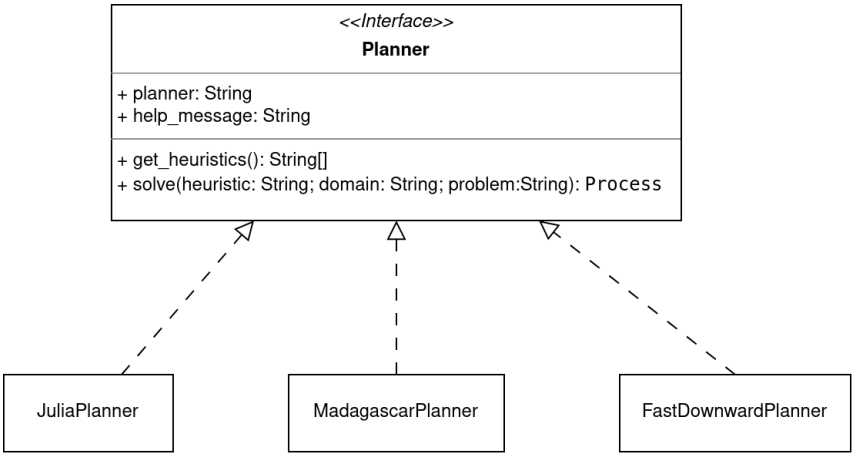

Figura 7 – UML da interface para planejadores

4.4.5 Arquivo de configuração

O funcionamento adequado do servidor mestre depende de um arquivo de configuração em formato JSON denominado *config.json*, exemplificado no Código 8. Este arquivo *config.json* contém uma chave denominada *workers*, que representa uma lista de servidores trabalhadores. Cada servidor trabalhador é identificado pelo *hostname* da máquina onde está hospedado, a porta em que está configurado para escutar e um status que indica se o servidor mestre deve utilizá-lo para executar um planejador.

Código 8 – Arquivo config.json para o servidor mestre

```
1  {
2      "server": {
3          "host": "0.0.0.0",
4          "port": 8000
5      },
6      "workers": [
7          {
8              "worker": "cm1",
9              "port": 8001,
10             "status": true
11         },
12         {
13             "worker": "cm2",
14             "port": 8001,
```

```
15              "status": true
16          },
17          {
18              "worker": "cm3",
19              "port": 8001,
20              "status": true
21          },
22          {
23              "worker": "cm4",
24              "port": 8001,
25              "status": true
26          },
27          {
28              "worker": "cm5",
29              "port": 8001,
30              "status": true
31          }
32      ]
33  }
```

Para listar os trabalhadores, o servidor mestre cria N *threads*, onde N é a quantidade de trabalhadores registrados no arquivo *workers.json*. Cada *thread* é responsável por estabelecer a comunicação com um trabalhador específico e obter as informações necessárias. Um tempo limite de dois segundos é estabelecido para cada requisição, garantindo uma resposta ágil.

4.5 Banco de dados

O funcionamento adequado do Infraplanning depende da persistência dos dados relacionados aos limites de tempo de execução e de memória para cada usuário. Para gerenciar esses limites de maneira eficiente e considerando a quantidade de usuários na aplicação, foram introduzidos grupos de usuários. Essa estrutura permite que o administrador ajuste os limites para todos os membros de um grupo simultaneamente. Caso seja necessário modificar os limites de um usuário específico, é possível criar um novo grupo com configurações atualizadas e mover o usuário para este grupo.

Para assegurar a persistência dos dados, foi adotado o SQLite (Hipp, 2023), um Sistema Gerenciador de Banco de Dados Relacional (SGBDR) conhecido por sua leveza, simplicidade e facilidade de uso. O SQLite é uma opção robusta em muitos cenários, especialmente em aplicações como o Infraplanning, que priorizam simplicidade, portabilidade e não requerem alta escalabilidade ou complexidade em operações de banco de dados. O diagrama lógico de dados representando o banco de dados criado para o Infraplannig pode ser visto na Figura 8.

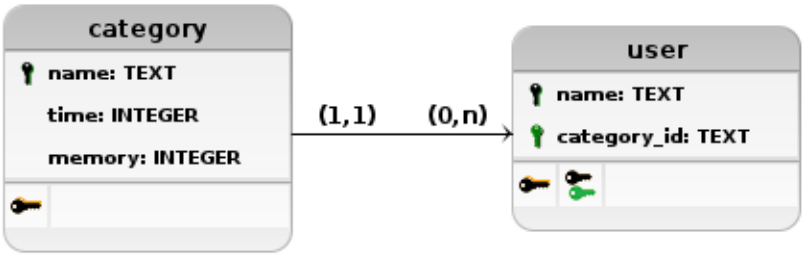

Figura 8 – Diagrama Lógico de Dados

O SQLite opera diretamente com um arquivo de banco de dados, dispensando a necessidade de um servidor separado. Esse arquivo de banco de dados está localizado junto aos arquivos do servidor mestre. Na inicialização do servidor mestre, caso o arquivo do banco de dados não exista, um novo é criado.

Além disso, sempre que o servidor mestre inicializa um novo banco de dados, este é preenchido com informações cruciais para o funcionamento do Infraplanning. Entre essas informações, são populados três grupos: o grupo professor com permissão para 24 horas de processamento e 32GB de memória; o grupo aluno com acesso a 30 minutos de processamento e 16GB de memória; e o grupo visitante com um minuto de processamento e 1GB de memória. Essas configurações predefinidas garantem o funcionamento adequado da aplicação para diferentes tipos de usuários.

5 Resultados

O código fonte desenvolvido, incluindo o histórico de versionamento, instruções de execução e requisitos necessários, estão armazenados em um repositório do GitHub[1]. Além disso, a aplicação Infraplanning[2] está disponível publicamente e pronta para uso.

É importante salientar que foram implementadas tarefas para atender a todos os requisitos detalhados na seção 4.1, oferecendo suporte aos planejadores: Julia Planners, Fast Downward e Madagascar. Para cada planejador é possível passar parâmetros de execução via interface Web. Por exemplo, o planejador Julia Planners aceita três heurísticas como parâmetro: HADD, HMAX e FF.

O planejador Fast Downward possui uma série de configurações prontas que podem ser escolhidas na interface Web, sendo elas: lama, lama-first, seq-opt-bjolp, seq-opt-fdss-1, seq-opt-fdss-2, seq-opt-lmcut, seq-opt-merge-and-shrink, seq-sat-fd-autotune-1, seq-sat-fd-autotune-2, seq-sat-fdss-1, seq-sat-fdss-2, seq-sat-fdss-2014, seq-sat-fdss-2018 e seq-sat-lama-2011. No caso do planejador Madagascar, é possível escolher entre as três configurações principais: M, Mp e MpC.

As próximas subseções detalharão as particularidades do ambiente no qual a aplicação está implantada, além de realizar uma comparação entre a solução desenvolvida e os trabalhos correlatos: Planning Domains e o Web Planner.

[1] Disponível em <https://github.com/UnB-SAT/tcc-antonio-ruan-infraplanning>
[2] Disponível em <https://plan-editor.naquadah.com.br>

5.1 Ambiente

Atualmente, o Infraplanning está integrado em um conjunto de maquinas localizadas na Universidade de Brasília, no campus Gama, onde conta com um total de 11 máquinas contribuindo com seu funcionamento. O Quadro 2 enumera todas essas máquinas, suas configurações e os serviços específicos que cada máquina hospeda.

Quadro 2 – Máquinas integradas no Infraplanning

Hostname	Processador	Memória RAM (GB)	Servidores
prestigio	Intel i5-4200U	4	Web
cm1	Intel i7-8700	16	Trabalhador
cm2	Intel i7-8700	16	Trabalhador
cm3	Intel i7-8700	16	Trabalhador
cm4	Intel i7-8700	16	Trabalhador
cm5	Intel i7-8700	16	Trabalhador
pos1	Intel i7-9700	16	Trabalhador
pos2	Intel i7-9700	16	Trabalhador
gpu1	Ryzen 7 2700	32	Trabalhador
gpu2	Ryzen 7 2700	32	Mestre e Trabalhador
gpu3	Ryzen 7 2700	32	Trabalhador

Por meio dessa infraestrutura robusta e bem-organizada, o Infraplanning está apto a atender às exigências de desempenho e escalabilidade, proporcionando uma experiência eficaz e ágil para os usuários. O planejamento e a configuração meticulosos refletem o compromisso em oferecer um ambiente confiável e estável para o desenvolvimento e execução de projetos utilizando a linguagem PDDL.

As máquinas listadas podem ser acessadas remotamente por meio do protocolo SSH por alunos da disciplina de Tópicos Especiais em Engenharia de Software, oferecida pela Universidade de Brasília e ministrada pelo Prof. Dr. Bruno César Ribas. Dentro desta disciplina, são estudados os Fundamentos Lógicos da Inteligência Artificial, e uma das linguagens estudadas é a linguagem PDDL.

A implementação da plataforma Infraplanning neste ambiente proporciona uma forma ágil e prática de desenvolver projetos em PDDL e executar um planejador, em comparação com a interface de linha de comandos disponível pelo protocolo SSH. O sistema de gerenciamento de usuários integrado ao sistema operacional permite que os alunos criem um projeto na interface web e prossigam com o trabalho remotamente via SSH, ou vice-versa. Isso proporciona maior flexibilidade no desenvolvimento de projetos em PDDL.

5.2 Análise Comparativa com Trabalhos Correlatos

A plataforma Infraplanning coexiste com o editor do Planning Domains e a plataforma Web Planner. O Quadro 3 apresenta um comparativo detalhado entre o Infraplanning e o Planning Domains, realçando suas características, funcionalidades e diferenças, oferecendo uma visão abrangente das semelhanças e diferenças entre eles. Infelizmente, devido à indisponibilidade da plataforma Web Planner, não foi possível realizar comparações com esta última.

Quadro 3 – Quadro comparativo entre Infraplanning e Planning Domains

	Infraplanning	Planning Domains
Quantidade de planejadores padrão	3	7
Permite configurar planejadores de terceiros?	Não	Sim
Possui visualização dos logs em tempo real do planejador?	Sim	Não
Permite cancelar o planejador enquanto esta sendo executado?	Sim	Não
Permite personalizar os limites de execução do planejador?	Sim	Não
Tempo de processamento máximo por planejamento (segundos)	60	10
Memória máxima por planejamento (MegaBytes)	1024	500
Possui visualização gráfica do plano gerado?	Não	Sim

As interfaces do editor do Planning Domains (consulte a Figura 9) e da plataforma Web Planner (consulte a Figura 10) serviram de inspiração para a concepção da interface da plataforma Infraplanning. De acordo com Magnaguagno et al., visualizar e editar o domínio e o problema em PDDL simultaneamente melhora a compreensão de que eles estão sendo usados juntos, reduzindo o esforço do usuário. No entanto, essa visualização se torna inviável à medida que o número de arquivos que o usuário está editando aumenta. Por isso, a interface do Planning Domains é a que mais se assemelha à interface do Infraplanning, conforme mostrada na Figura 11.

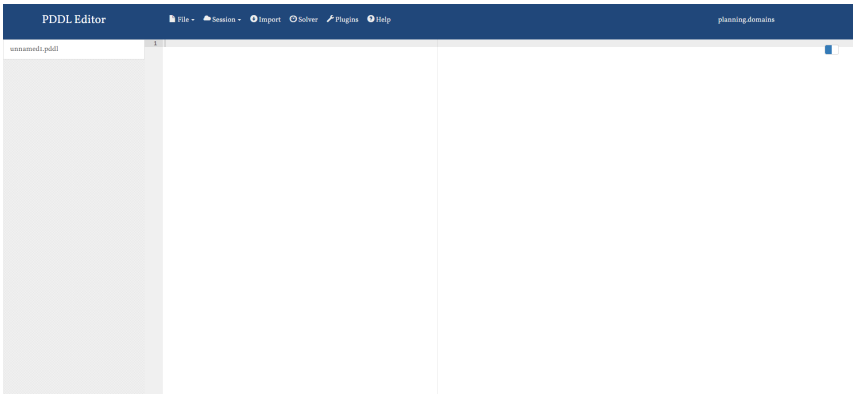

Figura 9 – Interface do editor do Planning Domains. Fonte: Página do editor do Planning Domains[3]

[3] Disponível em: <`https://editor.planning.domains/`> Acesso em dez. 2023

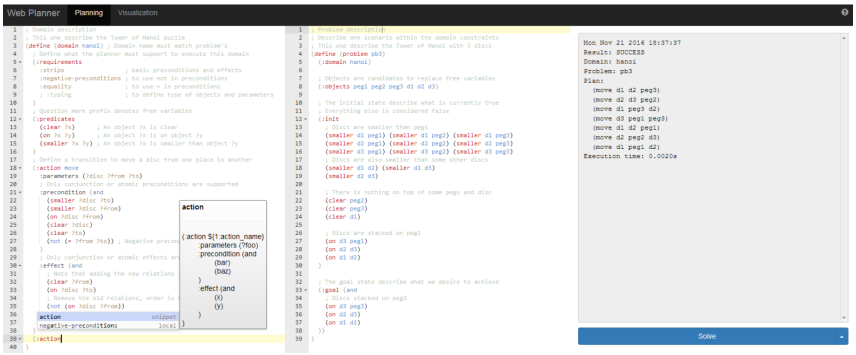

Figura 10 – Interface do editor do Web Planner. Fonte: Magnaguagno et al. (2017)

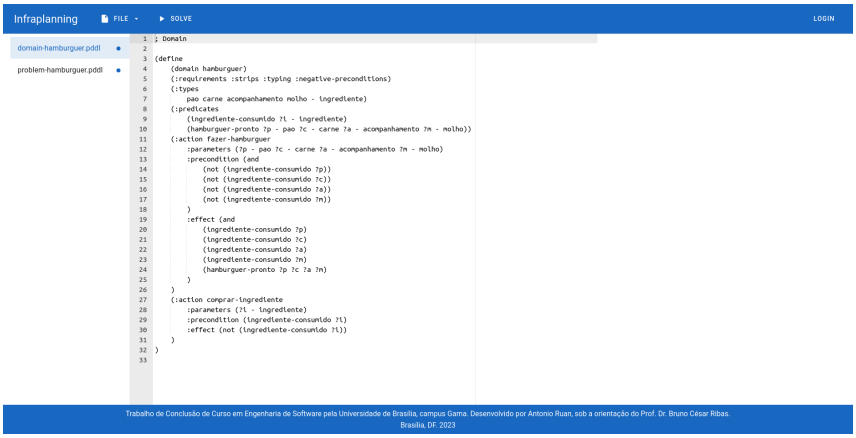

Figura 11 – Interface web do Infraplanning para um usuário convidado.

Por padrão, o projeto PDDL descrito na Seção 2 é carregado, o que ajuda os usuários que estão aprendendo sobre a linguagem PDDL a compreender a sintaxe e como utilizar a linguagem para resolver um problema real. A compreensão desse exemplo pode aumentar a autoconfiança dos estudantes, incentivando-os a explorar mais e aprofundar seu conhecimento na linguagem.

6 Conclusão

O desenvolvimento da plataforma Infraplanning representa um avanço significativo ao proporcionar um ambiente interativo e facilitador para a elaboração e execução de projetos em PDDL (*Planning Domain Definition Language*) e a execução de planejadores. A flexibilidade e a usabilidade do Infraplanning desempenharam um papel essencial ao atender às demandas dos usuários, especialmente no âmbito educacional, permitindo que estudantes e pesquisadores tenham acesso simplificado a planejadores automatizados para a resolução de problemas em Inteligência Artificial (IA).

No entanto, é importante salientar a necessidade de contínuo aprimoramento para acompanhar as demandas e avanços na área de planejamento automatizado. Sugerem-se melhorias para o editor de código PDDL, como destaque de sintaxe e autocompletar o código em PDDL, assemelhando-se ao que já é implementado no Planning Domain e no Web Planner. Ademais, a integração da resposta do planejador com o editor, apontando

os pontos no código que possam conter erros, é outro aspecto a ser considerado em trabalhos futuros, auxiliando na detecção e correção de possíveis falhas e, por conseguinte, aprimorando a qualidade do desenvolvimento.

Outras sugestões para futuras contribuições incluem melhorias na flexibilidade de instalação e uso do Infraplanning. A conteinerização dos serviços do Infraplanning permitiria uma instalação rápida e independente do sistema operacional da máquina que hospeda o serviço. A disponibilização de uma rota HTTP para execução de planejadores, atualmente realizada pelo protocolo Web Socket, facilitaria a utilização do sistema por terceiros, dispensando a necessidade de acessar a interface web.

Adicionalmente, uma recomendação para abordar a vulnerabilidade na pasta ".infraplanning" (detalhada na subseção 4.4.3) seria a utilização de grupos subordinados. Nesse cenário, as permissões de escrita e leitura para o diretório ".infraplanning" de cada usuário seriam atribuídas a um grupo subordinado específico. O Infraplanning, por sua vez, abrangeria todos os grupos subordinados atribuídos pelos usuários, proporcionando um controle mais preciso sobre as permissões de acesso.

Referências

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Dolejsi, J. (2023). Planning domain description language support. `https://github.com/jan-dolejsi/vscode-pddl`.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wikins, D. (1998). *PDDL - The Planning Domain Definition Language*. Yale Center for Computational Vision and Control.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.

Hipp, R. D. (2023). SQLite. `https://www.sqlite.org/index.html`. Acessado: 25 de novembro de 2023.

Hoffmann, J. (2011). Analyzing search topology without running any search: On the connection between causal graphs and h. *Journal of Artificial Intelligence Research*, 41:155–229.

Jones, M. B., Bradley, J., and Sakimura, N. (2015). JSON Web Token (JWT). RFC 7519.

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley Professional, 1st edition.

Leider, J. and Leider, H. (2023). Vuetify. `https://vuetifyjs.com`. Acessado: 08 de julho de 2023.

Magnaguagno, M. C., Pereira, R. F., Móre, M. D., and Meneguzzi, F. R. (2017). Web planner: a tool to develop classical planning domains and visualize heuristic state-space search. In *2017 Workshop on User Interfaces and Scheduling and Planning (UISP@ICAPS)*.

McKenzier, S. (2023). Yarn. `https://yarnpkg.com`. Acessado: 08 de julho de 2023.

Muise, C. (2016). Planning.Domains. In *The 26th International Conference on Automated Planning and Scheduling - Demonstrations*.

Ramírez, S. (2023). FastAPI. `https://fastapi.tiangolo.com`. Acessado: 08 de julho de 2023.

Rintanen, J. (2014). Madagascar : Scalable planning with sat.

You, E. (2023). Vue.js. `https://vuejs.org`. Acessado: 08 de julho de 2023.

Zhi-Xuan, T. (2022). *PDDL. jl: An Extensible Interpreter and Compiler Interface for Fast and Flexible AI Planning*. PhD thesis, Massachusetts Institute of Technology.

Zhi-Xuan, T. (2023). SymbolicPlanners.jl. https://github.com/JuliaPlanners/SymbolicPlanners.jl.