INSTITUTO FEDERAL DE SANTA CATARINA (IFSC)
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MATHEUS D'AMARAL GOMES

PLUGIN DE AUTOCAD PARA AUTOMATIZAÇÃO DE ETAPAS DO DESENVOLVIMENTO DE PROJETOS DE INSTALAÇÕES ELÉTRICAS

ITAJAÍ
2022

INSTITUTO FEDERAL DE SANTA CATARINA (IFSC)
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MATHEUS D'AMARAL GOMES

PLUGIN DE AUTOCAD PARA AUTOMATIZAÇÃO DE ETAPAS DO DESENVOLVIMENTO DE PROJETOS DE INSTALAÇÕES ELÉTRICAS

Relatório final, apresentado ao Instituto Federal de Santa Catarina, campus Itajaí, como parte das exigências para a obtenção do título de Engenheiro Eletricista.

ITAJAÍ
2022

MATHEUS D'AMARAL GOMES

PLUGIN DE AUTOCAD PARA AUTOMATIZAÇÃO DE ETAPAS DO DESENVOLVIMENTO DE PROJETOS DE INSTALAÇÕES ELÉTRICAS

Relatório final, apresentado ao Instituto Federal de Santa Catarina, campus Itajaí, como parte das exigências para a obtenção do título de Engenheiro Eletricista.

Itajaí, 23 de março de 2022.

BANCA EXAMINADORA

_____

Dra. Profa. Ana Elisa Ferreira Schmidt
Chefe do Departamento de Ensino, Pesquisa e Extensão do IFSC Itajaí
Professora Orientadora

_____

Dra. Profa. Fernanda Isabel Marques Argoud
Coordenadora do curso de Engenharia Elétrica do IFSC Itajaí
Avaliadora

_____

Me. Prof. Marcelo dos Santos Coutinho
Professor do IFSC Itajaí
Avaliador

AGRADECIMENTOS

À minha mãe Lilian e familiares próximos, os quais me apoiaram imensamente durante não só o desenvolvimento deste trabalho, mas também em toda minha jornada até o momento, dentro e fora do contexto acadêmico.

À amigos, em especial os que conheci na faculdade e me aproximei cada vez mais durante os últimos tempos, e amigos a centenas e milhares de quilômetros de distância, mas que estão sempre presentes nos melhores e piores momentos.

À colegas de trabalho que me auxiliaram em testes.

E finalmente à minha orientadora Ana Elisa, a qual me orientou neste trabalho desde sua proposição na disciplina de Projeto Integrador III, e que me orienta há anos, como durante a vitória da minha equipe em ambas edições do campeonato de robótica do IFSC.

In memoria di Nielson Brites D'Amaral

RESUMO

Uma das formas de reduzir os riscos impostos pela utilização de energia elétrica em instalações prediais, e ainda aumentar a eficiência da execução das instalações e do sistema elétrico em si, se dá por meio da automatização de projetos de instalações elétricas, o que por sua vez não apenas reduz o tempo de projeto, como também minimiza erros humanos por conta da repetição de tarefas manuais.

Desta forma, este trabalho, por meio do projeto e desenvolvimento de um *plugin* e testes de usabilidade deste, tem como objetivo validar a seguinte hipótese: é viável projetar e desenvolver um *plugin* para AutoCAD dedicado à automatização de etapas da criação de projetos de instalações elétricas?

Os objetivos específicos incluem funcionalidades para: gerar de forma automática quadros de cargas, diagramas unifilares e listas de materiais; e sugerir a quantidade mínima de tomadas e da carga destinada à luminárias por ambiente da instalação. Por fim propõe-se a validação do funcionamento do *plugin* por meio de testes de usabilidade.

Acerca da metodologia empregada, realiza-se uma pesquisa qualitativa, de natureza aplicada, objetivo exploratório-descritivo, método hipotético-dedutivo e adotando como procedimento técnico a pesquisa bibliográfica, documental e de levantamento de dados.

As etapas de desenvolvimento, bem como os resultados, são apresentados e estão divididos entre análise das funcionalidades implementadas e análise dos dados gerados pelos testes, realizados com quatro profissionais com experiência na área, e são apresentados próximos ao término do trabalho. Por fim, conclui-se que a hipótese é confirmada por meio dos resultados e da análise destes, sendo que o *feedback* obtido durante a etapa de testes é positivo.

Palavras-chave: Plugin. Autodesk AutoCAD. Programação. Projetos de Instalações Elétricas. Automatização. Teste de Usabilidade.

ABSTRACT

One way of reducing the risks imposed by the usage of electrical energy in building installations, while also increasing the efficiency of the installation process and the electrical system itself, is through the automation of electrical installation designs, which not only reduces the design time, but also minimizes human-caused errors due to repetitive tasks.

Therefore, this work, through the design and development of a plugin and its usability testing, aims to validate the following hypothesis: is it feasible to design and develop a plugin for AutoCAD dedicated to the automation of sections of the development of electrical installation projects?

The specific objectives include features to: automatically create electrical load schedules, single-line diagrams and materials schedules; and suggest the minimum number of sockets and load assigned to lighting per room. Finally, it's proposed to validate the functioning of the plugin through usability tests.

Regarding the work methodology applied, a qualitative research is carried out, with an applied nature, exploratory-descriptive objective, a hypothetical-deductive method and adopting as technical procedures the bibliographic, documental and data collection research.

The development sections, as well as the results, are presented and divided between the analysis of the implemented features and the analysis of the tests' data, carried out with four professionals with experience in the area, and are presented near the end of the work. At last, it's concluded that the hypothesis is confirmed through the results and their analysis, taking into account that the feedback obtained during this section is positive.

Keywords: Plugin. Autodesk AutoCAD. Programming. Electrical Installation Designs. Automation. Usability Test.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABNT	Associação Brasileira de Normas Técnicas
API	Application Programming Interface
ART	Anotação de Responsabilidade Técnica
BIM	Building Information Management
CAD	Computer Aided Design
CD	Diagrama de Classes
CELESC	Centrais Elétricas de Santa Catarina
CSV	Comma-Separated Values
DLL	Dynamic Link Library
LISP	List Processor
NBR	Norma Brasileira
OOP	Object Oriented Programming
SPA	Sistema de Proteção Contra Descargas Atmosféricas
TCC	Trabalho de Conclusão de Curso
UML	Unified Modeling Language
VCS	Version Control System

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Projetos de instalações elétricas, ou projetos elétricos como são coloquialmente chamados, têm como finalidade representar a transferência de energia elétrica a partir de uma fonte de energia até pontos de utilização, como luminárias, tomadas, motores, etc., tendo como foco, principalmente, a eficácia do sistema e a segurança dos envolvidos (LIMA FILHO, 2011).

A fim de garantir a segurança de pessoas e animais, o desenvolvimento desse tipo de projeto é orientado por normas técnicas, como a Norma Brasileira (NBR) 5410, de instalações elétricas de baixa tensão (ABNT, 2004), e a N-321.0001, de fornecimento de energia elétrica em tensão secundária de distribuição (CELESC, 2019), sendo esta uma norma da principal concessionária de distribuição de energia elétrica da região de Itajaí (SC), as Centrais Elétricas de Santa Catarina (Celesc), enquanto aquela é uma norma nacional, desenvolvida pela Associação Brasileira de Normas Técnicas (ABNT).

Para a aprovação e, posteriormente, a execução de projetos de instalações elétricas são necessários diversos documentos, como: Anotações de Responsabilidade Técnica (ART); memoriais; plantas baixas; esquemas; diagramas; quadros; e detalhes técnicos (CELESC, 2017; LIMA FILHO, 2011).

Atualmente há diversos programas computacionais, também conhecidos como *softwares*, capazes tanto de representar, como calcular e dimensionar parâmetros dos projetos elétricos. Alguns dos softwares mais conhecidos e utilizados atualmente são o AutoCAD (AUTODESK, 2022), o Revit (AUTODESK, 2022) e o QiElétrico (ALTOQI, 2022).

Além dos programas computacionais mencionados, os quais são programas *standalones* (programas que não necessitam de softwares complementares), também há programas denominados *plugins,* ou módulos de extensão, os quais adicionam novas funcionalidades a programas computacionais já existentes, complementando seu funcionamento. Exemplos de *plugins* utilizados para o desenvolvimento de projetos de instalações elétricas incluem o AditivoCAD 3 (ADITIVOCAD, 2022) e o PRO-Elétrica (MULTIPLUS, 2022).

Através de uma experiência profissional em um escritório de engenharia, o autor deste trabalho pôde perceber como, ainda que utilizando programas

computacionais, diversas etapas do desenvolvimento de projetos de instalações elétricas, e também de outras disciplinas como preventivo e hidrossanitário, ainda são realizadas de maneira manual, além de serem muitas vezes repetitivas. Sendo assim, o autor percebeu espaço para o desenvolvimento de uma ferramenta computacional que possa auxiliar na automatização de etapas de projetos de instalações elétricas.

Automação, segundo Billings (1997, p. 18), “é o processo que controla uma função ou tarefa sem intervenção humana.”. A organização *Institute of Electrical and Electronic Engineers* (IEEE) (2011, p. 1) complementa o conceito de automação afirmando que esta enfatiza a eficiência, a produtividade e a confiabilidade. Por mais que conceitualmente similares, tem-se que automação e automatização são termos distintos, onde neste é necessária intervenção humana, enquanto naquele o processo é realizado sem a necessidade de tal intervenção (DA LUZ e KUIAWINSKI, 2006).

Ainda que contando com funcionalidades para automatização de determinadas tarefas, os *plugins* disponíveis no mercado são relativamente genéricos, no sentido de que a simbologia utilizada diverge do padrão utilizado pela empresa no desenvolvendo os projetos de instalações elétrica, e pode não levar em consideração, durante o cálculo e dimensionamento de parâmetros dos projetos, as normas técnicas regionais. Outro ponto a se considerar é o preço destas ferramentas, as quais são comumente inacessíveis para estudantes e projetistas iniciantes.

Desta forma, utilizou-se a matéria de Projeto Integrador III (lecionada no 9° semestre do curso de Engenharia Elétrica do IFSC campus Itajaí, e que tem como objetivo instigar o aluno a desenvolver um projeto que integre diferentes disciplinas vistas no curso até o momento), para desenvolver o protótipo de um *plugin* para AutoCAD, em forma de *proof of concept* (ver seção 2.3.1), a fim de automatizar tarefas referentes ao desenvolvimento de projetos de instalações elétricas.

Esse protótipo, o qual é referido neste relatório de trabalho como versão inicial do *plugin*, constituiu em gerar de forma automatizada o quadro de cargas e o diagrama unifilar de projetos de instalações elétricas prediais. O protótipo foi bem sucedido, sendo o *plugin* capaz de realizar as funções definidas.

Tendo em vista o sucesso do protótipo, este trabalho de conclusão tem como objetivo dar continuidade ao desenvolvimento do *plugin*, otimizando-o e adicionando a ele novas funcionalidades, além da realização de testes de usabilidade com

profissionais com experiência na área, a fim de validar, entre outras coisas, sua eficiência.

Pretende-se com o desenvolvimento deste Trabalho de Conclusão de Curso (TCC) responder a seguinte hipótese: É viável projetar e desenvolver um plugin para AutoCAD dedicado à automatização de etapas da criação de projetos de instalações elétricas?

1.2 Justificativa

De acordo com The World Bank (2022), em 2019 cerca de 90,10 % da população mundial, aproximadamente 6,950 bilhões de pessoas (UNITED NATIONS, 2019, p. 5-6), tinha acesso à energia elétrica. Ainda que uma parcela tão grande da população mundial tivesse acesso à eletricidade, a dificuldade em compreender seu funcionamento, o qual é inatamente abstrato, faz com que seus riscos à integridade humana passem despercebidos em certas situações, como é o caso de instalações elétricas prediais mal projetadas, e, ainda mais grave, quando instalações nem sequer são projetadas, sendo apenas executadas com base no empirismo.

De acordo com um estudo realizado pela empresa Soul em maio de 2014, a pedido do Procobre (Instituto Brasileiro de Cobre) (2014, p. 19-21), 75 % das unidades familiares novas, nas regiões Sudeste e Nordeste do Brasil, não possuíam projeto elétrico na construção, o que representa uma informalidade na execução das instalações elétricas.

Além de questões relacionadas à segurança, também é válido mencionar a complexidade de projetos de instalações elétricas. O desenvolvimento de um projeto de instalação elétrica pode levar de alguns dias a até anos, isso porque é um processo que envolve diversas corporações, como construtoras e distribuidoras de energia elétrica, além do próprio escritório de engenharia. Este tipo de projeto também envolve a criação de diversos documentos, como esquemáticos, tabelas e diagramas (LIMA FILHO, 2011).

De tal maneira, este trabalho se mostra relevante uma vez que este, através da criação e desenvolvimento de um *plugin* para AutoCAD, pretende amenizar ambos os problemas discutidos. As funcionalidades que envolvem orientações extraídas das normas técnicas e a redução da repetição manual em determinadas tarefas reduzem a quantidade de erros no projeto, aumentando a segurança da instalação, enquanto a automatização de determinadas tarefas aumenta a eficiência do desenvolvimento, por

sua vez contribuindo para o trabalho do projetista, e consequentemente para a empresa.

O *plugin* proposto também tem seu valor na comunidade acadêmica, uma vez que pode ser utilizado por docentes em disciplinas como projetos de instalações elétricas a fim de demonstrar determinados conceitos e procedimentos quando se faz necessário um foco maior no resultado. Por fim, considera-se que o trabalho está alinhado com os valores do IFSC, em especial referente à inovação tecnológica e ao desenvolvimento de produtos e processos que possam ser transferidos para empresas do arranjo local, assim como para à comunidade acadêmica.

1.3 Objetivos

1.3.1 Gerais

Comprovar a viabilidade do projeto e desenvolvimento de um *plugin* para o AutoCAD, cujas funcionalidades permitam automatizar etapas de projetos de instalações elétricas, agilizar a criação de tabelas e diagramas necessários para a aprovação e execução do projeto e evitar erros causados pelo projetista por conta de repetição.

1.3.2 Específicos

Os seguintes pontos são abordados durante este trabalho de conclusão de curso:

- Desenvolvimento do modelo conceitual orientado a objetos da primeira versão do *plugin*, em formato de diagrama *Unified Modeling Language* (UML), de forma a otimizar este;
- Reestruturação das classes, parâmetros e métodos desenvolvidos na versão inicial do *plugin*, de acordo com o modelo conceitual;
- Adição de novas funcionalidades, como:
    - Geração de listas de materiais;
    - Geração de sugestão de quantidade mínima de tomadas e carga prevista para iluminação por ambiente, de acordo com a NBR 5410 (ABNT, 2004);
- Processo de validação e verificação do *plugin* a partir de testes de usabilidade, compreendendo as seguintes etapas:

- Definição do plano dos testes;
- Definição do local dos testes;
- Definição e recrutamento dos participantes;
- Preparação dos materiais para testes;
- Condução dos testes;
- Coleta, análise e apresentação dos dados.

2 REFERENCIAL TEÓRICO

Este capítulo tem como objetivo apresentar ao leitor os diferentes conceitos, tecnologias e termos técnicos utilizados neste trabalho.

2.1 Projetos de Instalações Elétricas

O objetivo dos projetos de instalações elétricas pode ser definido como:

> ...garantir a transferência de energia desde uma fonte, em geral a rede de distribuição da concessionária ou geradores particulares, até os pontos de utilização (pontos de luz, tomadas, motores etc.). Para que isso se faça de maneira segura e eficaz é necessário que o projeto seja elaborado, observando as prescrições das diversas normas técnicas aplicáveis. (LIMA FILHO, 2011, p. vi).

Como mencionado acima, projetos de instalações elétricas devem seguir normas técnicas para sua eficácia e segurança dos envolvidos. Entre as principais normas encontram-se a norma NBR 5410, de instalações elétricas em baixa tensão (ABNT, 2004), elaborada pela Associação Brasileira de Normas Técnicas (ABNT), e as normas da concessionária local, que no caso da região de Santa Catarina, onde este projeto foi desenvolvido, são as normas da Celesc, como a norma N-321.0001 de fornecimento de energia elétrica em tensão secundária de distribuição (CELESC, 2019) e a norma NT-03 de fornecimento de energia elétrica a edifícios de uso coletivo (CELESC, 1997).

2.1.1 Planta Baixa de Instalações Elétricas

São esquemáticos que apresentam o projeto arquitetônico e estrutural de um empreendimento, além de componentes elétricos, como tomadas, luminárias,

interruptores, quadros de distribuição, etc. (RASHID, ALGEELANI, *et al.*, 2021), além de suas respectivas conexões (eletrodutos e fiação).

2.1.2 Quadro de Cargas

Quadros de cargas, conforme Lima Filho (2011, p. 15) afirma, são tabelas que contém informações do projeto da instalação, como a divisão dos circuitos terminais e suas respectivas cargas, tipo, tensão e dimensionamento das seções dos condutores e disjuntores. Quadros de cargas comumente representam os parâmetros de um quadro de distribuição, também conhecido como quadro de disjuntores.

A finalidade destas tabelas abrange tanto o dimensionamento de componentes quanto a execução das instalações elétricas, além de permitir o entendimento da divisão dos circuitos e seus respectivos parâmetros, enquanto naquela, a tabela auxilia na definição dos parâmetros de proteção geral (seção do cabo de entrada e disjuntor geral) do quadro de distribuição e no cálculo da demanda total da edificação. A Figura 1 apresenta o modelo de um quadro de cargas baseado em um apartamento de três dormitórios fictício.

Figura 1 - Exemplo de um quadro de cargas.

QD15 - Diferenciado Final 01

Circuito	Descrição	Esquema	Tensão (V)	Pot. Total (W)	Fases	Pot. - R (W)	Pot. - S (W)	Pot. - T (W)	Seção (mm²)	Disj. (A)	DDR (30 mA)
1	Tomadas - Cozinha	F+N+T	220	1500	R	1500			4.0	20	
2	Tomada - Triturador	F+N+T	220	1000	S		1000		4.0	20	
3	Tomadas - Área de Serviço	F+N+T	220	1200	T			1200	4.0	20	
4	Tomadas - Living	F+N+T	220	800	R	800			2.5	16	
5	Tomadas - Suíte Master	F+N+T	220	800	S		800		2.5	16	40
6	Tomadas - Suíte 01	F+N+T	220	700	T			700	2.5	16	
7	Tomadas - Suíte 02	F+N+T	220	700	R	700			2.5	16	
8	Chuveiro	F+N+T	220	5500	S		5500		6.0	32	
9	Chuveiro	F+N+T	220	5500	T			5500	6.0	32	
10	Chuveiro	F+N+T	220	5500	R	5500			6.0	32	
11	Iluminação	F+N+T	220	720	S		720		2.5	16	25
12	Iluminação	F+N+T	220	600	T			600	2.5	16	
13	Ar Condicionado	F+N+T	220	1800	R	1800			4.0	20	
14	Ar Condicionado	F+N+T	220	1800	S		1800		4.0	20	
15	Ar Condicionado	F+N+T	220	1200	T			1200	2.5	16	25
16	Ar Condicionado	F+N+T	220	900	R	900			2.5	16	
17	Ar Condicionado	F+N+T	220	900	S		900		2.5	16	
	Carga Reserva	3F+N+T	220/380	4500	R+S+T	1500	1500	1500			-
Total		3F+N+T	220/380	35020	R+S+T	12700	12220	10700	16	63	-

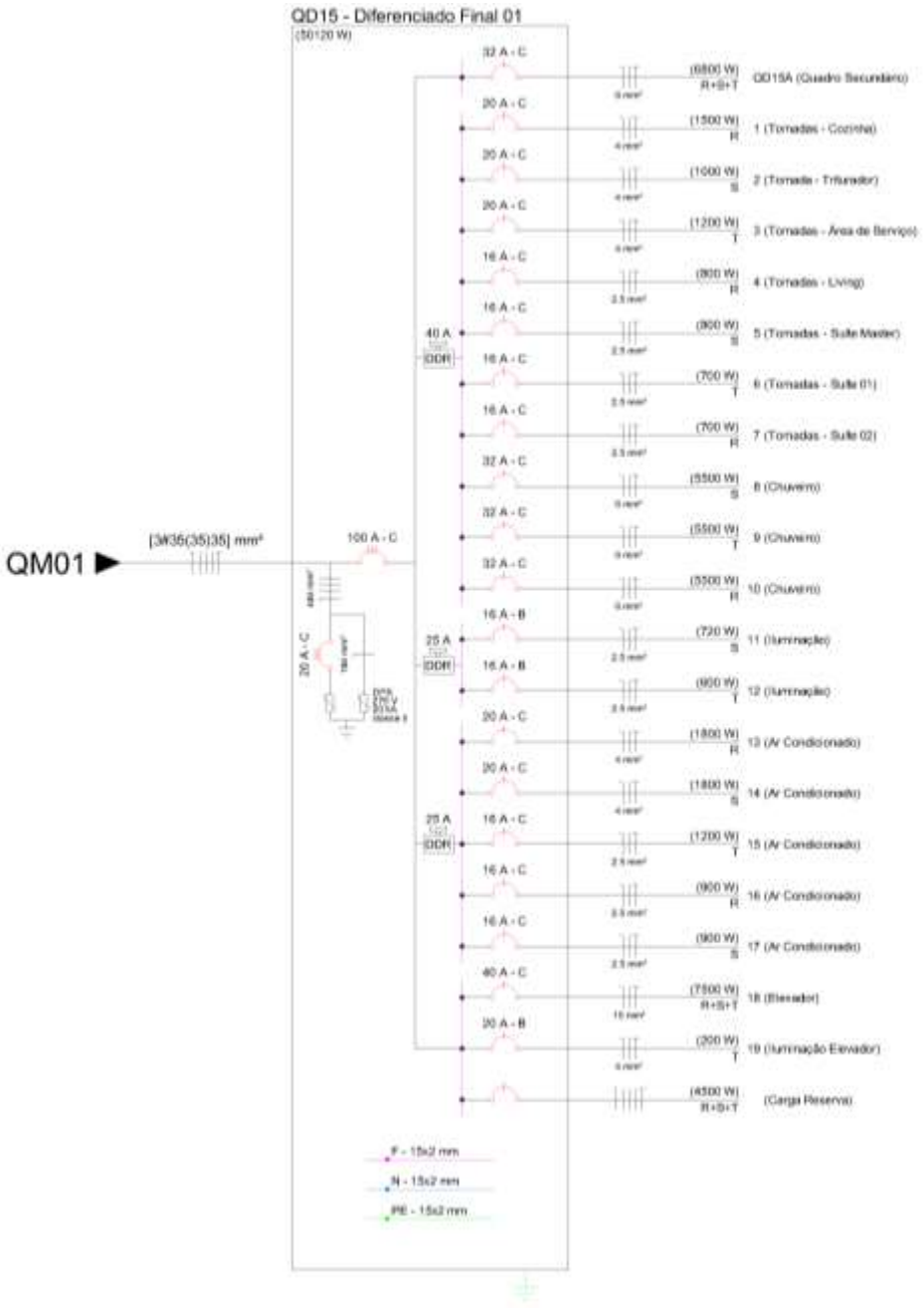
Fonte: Autor (2022).

2.1.3 Diagrama Unifilar

Outra forma de representar os componentes de um quadro elétrico é por meio de diagramas unifilares. O uso deste tipo de diagrama evita a repetição da representação de circuitos idênticos, no caso de conexões bifásicas e trifásicas, fazendo com que sejam mais práticos de representar e compreender (ZHOU, 2019).

As informações apresentadas em quadros de cargas e diagramas unifilares são similares, sendo a maior diferença a forma utilizada para apresentar os valores. Diagramas unifilares apresentam os parâmetros dos circuitos por meio de símbolos e textos, e tem como diferencial a possibilidade de visualizar as conexões dos circuitos. A Figura 2 abaixo apresenta o mesmo quadro elétrico da Figura 1 da seção 2.1.2, porém agora em formato de diagrama unifilar.

Figura 2 - Exemplo de um diagrama unifilar.



Fonte: Autor (2022).

2.1.4 Lista de Materiais

Listas de materiais apresentam informações como quantidade, tipo, tamanho, entre outras informações necessárias para aquisição de materiais para a execução de obras de instalações elétricas (ARIJELOYE; AKINRADEWO, 2016). Estudos também demonstram que a correta utilização de listas de materiais está atrelada à diminuição de desperdício de materiais em obras da construção civil (ADENAIYA e ADEJUGBAGBE, 2020).

2.1.5 Previsão de Carga

A fim de prever a quantidade mínima de carga demandada por tomadas e iluminação em um ambiente específico, utiliza-se a NBR 5410 (ABNT, 2004), a qual apresenta métodos de calcular a quantidade de pontos de tomada e a carga mínima para iluminação por ambiente, isso de acordo com sua área, perímetro e tipo.

2.2 Conceitos Computacionais

2.2.1 *Plugin*

Plugins são módulos de extensão que tem como função estender as capacidades de um certo *software*, permitindo, por exemplo, a customização da interface ou a adição de novas funcionalidades, não presentes nativamente no software (GARRIDO, *et al.*, 2019; ELLIS, TORCELLINI e CRAWLEY, 2008).

2.2.2 Programação Orientada a Objetos

A programação orientada a objetos, também conhecida como *Object Oriented Programming* (OOP), se baseia na existência de objetos, os quais possuem parâmetros, ou atributos, e métodos, ou funções (MOORE, 2020).

2.2.3 Diagrama UML

Diagramas do tipo UML podem ser utilizados para representar a relação entre classes, parâmetros e métodos de um programa computacional, sem a necessidade de conhecer uma linguagem computacional específica.

De acordo com Gosala, Chowdhuri, *et al*, (2021, p. 1), “diagramas UML são muito vantajosos para pesquisadores, desenvolvedores de *software* e acadêmicos”,

auxiliando a “estudar, analisar, documentar, projetar e desenvolver qualquer software eficientemente”.

2.2.3.1 Diagrama de Classes

Diagrama de classes é:

> ...um tipo de diagrama UML que ajuda a expressar as classes e relações entre as classes. Diagramas de classes (CD) UML são usados para projetar e ilustrar a estrutura do *software*. São ferramentas muito importantes para engenheiros entenderem a estrutura básica de um sistema. (GOSALA, *et al.*, 2021, p. 1-2)

2.2.4 IDE

Integrated Development Environment (IDE), ou ambiente de desenvolvimento integrado, segundo Dutta, Sethi e Khatri (2014, p. 1), “...é uma aplicação que provê facilidades para programadores para desenvolvimento de *softwares*, como preenchimento e correção de códigos, edição e gerenciamento do código fonte, testagem automática, etc.”.

2.2.5 API

Interfaces de Programação de Aplicativos, tradução do termo em inglês *Application Programming Interfaces* (APIs), permitem a interação entre diferentes programas computacionais, onde os serviços e funções de um podem ser acessados pelo outro (SHELBY, *et al.*, 2019).

2.2.6 CSV

São arquivos de texto que podem ser utilizados para armazenar estruturas tabulares simples (MÄS, *et al.*, 2018) para uso em editores externos, como Microsoft Excel (MICROSOFT, 2022) e Google Planilhas (GOOGLE, 2022).

2.2.7 DLL

Segundo a Microsoft, empresa desenvolvedora da tecnologia, *Dynamic Link Library* (DLL):

> ...é uma biblioteca que contém código e dados que podem ser usados por mais de um programa ao mesmo tempo. Por exemplo, em Windows sistemas operacionais, a DLL Comdlg32 executa funções relacionadas à caixa de diálogo comuns. Cada programa pode usar a funcionalidade contida nesta DLL para implementar uma caixa de diálogo Abrir. Ele ajuda a promover a reutilização de código e o uso eficiente de memória. (MICROSOFT, 2021)

2.2.8 Depuração

Elçi et al (2018, p. 195-198) explicam que *debugging*, denominado depuração em português, é o processo de encontrar erros na execução e a causa de incoerências no código. A depuração pode ser realizada, por exemplo, utilizando o método de execução passo-a-passo, onde as linhas do código são executadas uma a uma de forma sequencial, possibilitando a visualização de alterações à medida que ocorrem.

2.2.9 BIM

De acordo com a Autodesk (2022), empresa responsável pela criação do software Revit, entre outros, a Modelagem da Informação da Construção (BIM) "...integra dados estruturados e multidisciplinares para produzir uma representação digital de um recurso em todo seu ciclo de vida, desde o planejamento e o projeto até a construção e as operações". Vale ressaltar que BIM não é um *software*, e sim uma forma de gerar e manter fontes de informações ricas em dados, como destaca Sayary e Omar (2021).

2.2.10 *Softwares* e Plataformas

2.2.10.1 StarUML

StarUML (MKLABS, 2022) é um software de modelagem voltado para a criação de diagramas UML. Segundo a empresa desenvolvedora, a MKLabs, o software possui funcionalidades como suporte multiplataforma, compatibilidade com UML 2.x, entre outros, e tem como foco profissionais e instituições de ensino.

2.2.10.2 Visual Studio Community

De acordo com a Microsoft (2022), empresa criadora do software em questão, o Visual Studio é “um IDE gratuito, completo e extensível para a criação de aplicativos modernos para Android, iOS e Windows, bem como aplicativos Web e serviços de nuvem.”.

2.2.10.3 GitHub

A plataforma GitHub (GITHUB, 2022) se trata de um site onde é possível hospedar e desenvolver códigos, e que possui funcionalidades que permitem trabalho de forma colaborativa, automação, segurança, etc., além de possuir uma comunidade vasta.

Entre as funcionalidades específicas relacionadas ao desenvolvimento de códigos tem-se a possibilidade de criar *branches* no projeto, o que traduzindo de forma literal significa criar ramificações no projeto, as quais são versões de teste do código, e que, caso não deem certo, podem ser revertidas, desta forma não alterando a versão principal, e funcional, do programa (RAM, 2013). Caso o *branch* seja validado, pode-se utilizar a função *commit* para mesclar o *branch* com o código principal.

Além de sua plataforma online, IDEs como o Visual Studio permitem a integração do GitHub através do próprio *software* de programação, permitindo, por exemplo, a criação de novos *branches* e *commits*.

2.2.10.4 AutoCAD

O Autodesk AutoCAD é “um software CAD que arquitetos, engenheiros e profissionais da construção utilizam para criar desenhos 2D e 3D precisos” (AUTODESK, 2022).

2.2.10.4.1 CAD

Ferramentas do tipo *Computer Aided Design* (CAD), como o nome implica, auxiliam por meio de computação o desenvolvimento de projetos, principalmente de áreas como arquitetura e engenharia, mais especificamente civil, elétrica e mecânica, diminuindo o tempo de criação de projetos, ao mesmo tempo em que a qualidade destes aumenta (SILVA; LAMOUNIER; CARDOSO, 2006).

2.2.10.4.2 *Model Space*

Model Space, no AutoCAD, como o nome em inglês indica, é o espaço de modelagem do *software*, onde é possível criar e editar objetos em 2D e 3D (AUTODESK, 2020).

2.2.10.4.3 Blocos

No AutoCAD, blocos são coleções de objetos primitivos como linhas, arcos, círculos, etc. agrupados em um único objeto nomeado. Tais blocos podem ser utilizados para representar, por exemplo, mobília em projetos arquitetônicos (OMURA, 2009), peças metálicas em projetos mecânicos e componentes elétricos e eletrônicos em projetos elétricos.

No software em questão, a diferença entre blocos e grupos é o fato de que cada bloco possui seu próprio *block definition*, o qual funciona como uma espécie de modelo de referência. Enquanto cópias de grupos, ou agrupamentos de objetos primitivos, são em essência objetos únicos, os quais podem ser alterados sem que um influencie no outro, cópias de blocos são, até certo ponto, idênticas, e no momento em que o bloco tem seu *block definition* editado, todas suas cópias, ou instâncias, presentes no desenho são atualizadas.

2.2.10.4.4 Blocos Dinâmicos

Foi mencionado acima que instâncias de um mesmo bloco são idênticas, porém apenas até certo ponto, isto porque, blocos podem ser dinâmicos. Blocos dinâmicos possuem regras e restrições impostas pelo usuário, que ajudam a controlar a aparência de blocos em um desenho (AUTODESK, 2020).

Um exemplo de modificação que transforma um bloco em bloco dinâmico é a utilização de atributos. Atributos são parâmetros que podem ser adicionados a blocos, permitindo que o usuário edite seus valores, seja de forma direta ou indireta. Os atributos são representados em todas as instâncias de um bloco, porém seu valor é variável e único para cada uma delas, podendo ser modificado pelo projetista no campo das propriedades do bloco.

Um bloco representando uma tomada, por exemplo, pode possuir um atributo denominado “CIRCUITO”, e que tem seu valor representado na tela ao lado do símbolo do ponto de tomada, a fim de diferenciar os circuitos do projeto, como é

apresentado na Figura 3 abaixo, onde é possível observar o número 1 no canto superior esquerdo.

Figura 3 - Bloco dinâmico representando um ponto de tomada.

Fonte: Autor (2021).

Um bloco também pode ser dinamizado ao se adicionar um comando de visibilidade a ele, o que permite um único bloco possuir diferentes representações visuais. As vantagens de ser ter um único bloco representando múltiplos objetos incluem facilidade de uso, otimização computacional e padronização.

2.2.10.4.5 *Macro*

Macros ajudam a automatizar tarefas repetitivas a fim de economizar tempo e acelerar o processo de desenho (AUTODESK, 2014); o que é feito por meio da gravação de comandos e, posteriormente, pela reprodução destes quantas vezes for necessário.

2.2.10.4.6 Visual LISP

Visual LISP é uma linguagem de programação, baseada na família *List Processing* (LISP), utilizada pelo software AutoCAD para a criação de novas funcionalidades para o *software*, por sua vez utilizadas para a automatização de tarefas (AUTODESK, 2022).

2.2.10.5 Microsoft Excel

O Excel (MICROSOFT, 2022) é uma das ferramentas do pacote Office da Microsoft, disponível em diversas plataformas como computadores, dispositivos móveis e navegadores *web*, capaz de realizar, entre outras funções, cálculo, organização e análise de dados, além da geração de gráficos.

2.2.10.6 Google Planilhas

O Google Planilhas (GOOGLE, 2022), ou Google Sheets como é o nome original, possui funcionalidades muito similares as do Microsoft Excel, sendo as maiores diferenças o fato de que o Google Planilhas pode ser acessado de forma gratuita por meio de uma conta do Gmail, também da Google, porém apenas por meio de um navegador *web*, além da falta de algumas funcionalidades e fórmulas avançadas.

2.2.10.7 Google Forms

Esta é outra plataforma para navegador *web*, também da Google (GOOGLE, 2022), focada na criação de formulários, permitindo a coleta e análise dos dados de forma prática.

2.3 Demais Conceitos

2.3.1 *Proof of Concept*

Proof of concept, ou prova de conceito como é comumente traduzido do inglês, consiste no desenvolvimento do protótipo de um produto, o qual acaba minimizando riscos, uma vez que questões como a viabilidade do projeto e possíveis problemas podem ser encontrados ainda nesta versão de testes (LAUKAT, 2020).

2.3.2 Usabilidade

Hassenzahl e Burmester (2021, p. 202-204) explicam que a usabilidade promove o projeto de sistemas interativos de forma que seus usuários alcancem seus objetivos de maneira efetiva e eficiente, tendo como foco o uso intuitivo.

2.3.2.1 Teste de Usabilidade

Teste de usabilidade é uma ferramenta de pesquisa onde uma amostra de pessoas, baseada em uma população de interesse, testam se determinados produtos fazem jus aos critérios de usabilidade (RUBIN; CHISNELL, 2008, p. 21-26).

2.4 Trabalhos Relacionados

2.4.1 AditivoCAD 3

O AditivoCAD 3 (ADITIVOCAD, 2022) é um plugin para AutoCAD com funcionalidades focadas no desenvolvimento de projetos de arquitetura, de instalações elétricas e de hidrossanitário, divididos em três módulos que podem ser adquiridos separadamente.

O módulo de elétrica possui funcionalidades como inserção de tomadas, luminárias, eletrodutos e fiação elétrica, além da geração de quadros de cargas, diagramas unifilares, listas de materiais e legendas, entre outras.

O *plugin* proposto neste trabalho se diferencia do AditivoCAD 3 Elétrico por ser um *plugin* especializado, com foco nos padrões representativos do projeto de uma determinada empresa e nas normas técnicas regionais, sem a necessidade de configurações adicionais, sendo realizados os cálculos e dimensionamentos necessários com bases nessas.

2.4.2 PRO-Elétrica

PRO-Elétrica (MULTIPLUS, 2022) é um *software* para projetos elétricos que, segundo a empresa, “detalha e dimensiona as instalações elétricas em baixa tensão, Sistema de Proteção Contra Descargas Atmosféricas (SPDA), instalação de placas fotovoltaicas *On-Grid*, cabeamento estruturado, loteamento com locação de postes e iluminação e automação residencial.”.

Um dos pré-requisitos deste *software* é ter instalado no computador do usuário algum outro *software* CAD, da lista de *softwares* compatíveis disponibilizada pelos desenvolvedores, fazendo com que, pelo entendimento do autor deste trabalho, esse possa ser considerado um *plugin*.

O *plugin* proposto neste trabalho se diferencia do PRO-Elétrica de forma similar à apresentada na seção 2.4.1.

2.4.3 Revit MEP

A versão *Mechanical, Electrical and Plumbing* (MEP) do software Revit (AUTODESK, 2022) tem como foco, como a sigla indica, a modelagem de projetos mecânicos, elétricos e de encanamento. Sendo um *software* que possibilita o uso do

método BIM (ver seção 2.2.9), este é capaz de representar projetos com um alto nível de detalhamento enquanto auxilia na execução deste.

Por ser um *software* complexo e que implementa a plataforma BIM, o Revit costuma ser utilizado em projetos de grandes dimensões e/ou alta complexidade, onde uma representação mais detalhada do projeto e o uso de funcionalidades para, por exemplo, encontrar problemas de compatibilização entre disciplinas (elétrico, preventivo, hidrossanitário, etc.) compensam ao facilitar e garantir uma maior confiabilidade na hora da execução do projeto.

Desta forma o *plugin* proposto pretende ser utilizado em paralelo com *softwares* mais robustos como o Revit, mais especificamente em projetos de menor complexidade.

2.4.4 QiElétrico

O módulo QiElétrico do software QiBuilder (ALTOQI, 2022), assim como os programas apresentados nas seções 2.4.1 e 2.4.2, apresenta funcionalidades específicas para o desenvolvimento de projetos elétricos, como lançamento de eletrodutos, geração de listas de materiais e dimensionamento, além de que, assim como o software Revit apresentado acima, permite a integração do método BIM.

Assim como descrito na seção 2.4.3 em relação ao *software* Revit, o QiElétrico também é uma ferramenta complexa e comumente utilizada em projetos que envolvem uma maior complexidade. Desta forma o *plugin* proposto pretende ser utilizado em paralelo com *softwares* como o QiElétrico, e não o substituir.

2.4.5 Plugin para AutoCAD para Cálculo de Quantitativos de Insumos e Auxílio no Projeto Orçamentário

Da Silva Jr. (2017) também desenvolve em seu TCC, do curso de Engenharia Civil, um *plugin* para AutoCAD com funcionalidades voltadas para o cálculo de quantitativos de insumos e auxílio no projeto orçamentário de obras civis.

O *plugin* de Da Silva Jr. se diferencia do proposto neste trabalho por ter foco no aspecto de projeto civil, ao contrário do presente plugin, que foca no projeto elétrico, ainda que também esteja inserido no contexto de projetos da construção civil.

Já quanto à metodologia, este trabalho de conclusão se diferencia do mencionado nesta seção no processo de validação do programa desenvolvido,

realizado por meio de testes de usabilidade, desta forma também caracterizando este trabalho como uma pesquisa.

3 METODOLOGIA

3.1 Pesquisa

A pesquisa tem caráter qualitativo, que segundo Gerhardt e Silveira (2009, p. 31-34) se baseia na análise de dados não-métricos, onde o pesquisador é o instrumento de coleta de dados e é enfatizado o raciocínio intuitivo.

3.2 Natureza

Pesquisa de natureza aplicada, onde o foco é a aplicação prática e a solução de problemas (GERHARDT; SILVEIRA, 2009).

3.3 Objetivo

O objetivo da pesquisa pode ser definido como exploratório-descritivo. Exploratório por envolver levantamento bibliográfico, documental e entrevistas, gerando um produto passível de investigação com procedimentos sistematizados, e descritivo por levantar opiniões, estando a pesquisa preocupada com a atuação prática (GIL, 2008, p. 27-29).

3.4 Método

O método utilizado é o hipotético-dedutivo, no qual a existência de um problema leva à criação de uma hipótese, a qual é validada ou refutada a partir de testes empíricos (GIL, 2008, p. 49-59).

3.5 Procedimentos Técnicos

Os procedimentos técnicos do projeto são dados por meio de pesquisa bibliográfica (artigos, livros, jornais, etc.), pesquisa documental (como normas e leis) e por meio de levantamento de campo, onde ocorre a interrogação direta de pessoas de um grupo predeterminado, baseado no problema, e permite o estudo de opiniões e atitudes (GIL, 2008).

3.6 Etapas do Desenvolvimento do Trabalho

O desenvolvimento deste trabalho é seccionado em quatro partes principais, as quais são apresentadas nas subseções abaixo.

3.6.1 Levantamento Bibliográfico e Documental

A fim de aprofundar o conhecimento técnico de conceitos, tecnologias e termos técnicos relacionados ao trabalho, são realizadas pesquisas bibliográficas, nas quais são estudados principalmente artigos e livros de trabalhos pertinentes ao tema e de preferência atuais, além de pesquisas documentais, as quais envolvem o estudo de documentos como normas e leis. O levantamento é apresentado no capítulo 2 - Referencial Teórico.

3.6.2 Desenvolvimento do Plugin

O desenvolvimento do plugin, o qual não partiu do zero por conta do protótipo desenvolvido previamente, como mencionado no capítulo 1 - Introdução, é apresentado na seção 4, e é constituído pelo aprimoramento do protótipo desenvolvido na disciplina de PI3, nos seguintes aspectos: pela otimização desta por meio do desenvolvimento de um diagrama de classes orientado a objeto e pelo projeto e desenvolvimento das novas funcionalidades, conforme descrito na seção 1.3.2 de objetivos específicos.

3.6.3 Testes de Usabilidade

A divisão das etapas relacionadas aos testes de usabilidade é baseada no manual de testes de usabilidades de Jeff Rubin e Dana Chisnell (2008), onde ocorre a definição do plano de teste; definição do local dos testes; definição e recrutamento dos participantes; preparação dos materiais para a condução dos testes; condução da sessão de testes; coleta dos dados; análise dos dados e, por fim, apresentação dos resultados e conclusão. Informações sobre os testes de usabilidade são apresentadas na seção 5.

3.6.4 Análise dos Resultados Obtidos e Conclusão

A última etapa do trabalho é constituída pela análise dos resultados obtidos, no desenvolvimento do *plugin* e durante os testes de usabilidade, assim como a conclusão do projeto, onde a comparação dos resultados com os objetivos propostos permitem a validação da hipótese proposta.

4 DESENVOLVIMENTO DO *PLUGIN*

Foi mencionado anteriormente, em especial no capítulo 1 - Introdução, como o presente trabalho dá continuação à versão inicial do *plugin* desenvolvida em Projeto Integrador III, matéria lecionada no 9° período do curso de engenharia elétrica do IFSC campus Itajaí. A fim de melhor nortear o leitor quanto ao desenvolvimento do *plugin*, é descrito nesta seção não só o desenvolvimento das novas funcionalidades, e sim de todas as etapas, incluindo as funcionalidades já previamente desenvolvidas, englobando também a concepção do plugin e a criação dos blocos dinâmicos no AutoCAD, necessários para seu correto funcionamento.

Deste modo, define-se as seguintes subseções conforme o momento em que foi desenvolvido:

Quadro 1 - Divisão das etapas do desenvolvimento do *plugin.*

Versão do *Plugin*	**Etapa do Desenvolvimento**
Prova de conceito (Projeto Integrador III)	4.1 - Criação do *Plugin*
	4.2 - Criação dos Blocos Dinâmicos
	4.3 - Obtenção dos Parâmetros do Plugin
	4.4 - Geração Automatizada do Quadro de Cargas
	4.5 - Geração Automatizada do Diagrama Unifilar
Versão final (TCC)	4.6 - Criação do Diagrama de Classes
	4.7 - Geração Automatizada da Lista de Materiais
	4.8 - Sugestão da Quantidade de Tomadas e Carga Prevista para Iluminação por Ambiente

Fonte: Autor (2022).

A seguir, cada uma das etapas de desenvolvimento do plugin, citadas no Quadro 1, são apresentadas em detalhes.

4.1 Criação do *Plugin*

Há várias formas de aprimorar funcionalidades já existentes ou criar novas funcionalidades para o AutoCAD, sendo as mais comuns por meio de *macros*, Visual LISPs e *plugins*. A criação de *macros* e Visual LISPs pode ser feita por meio do próprio *software*, o qual disponibiliza funcionalidades específicas para tal, como o gravador de *macros* (AUTODESK, 2019), o qual grava os comandos utilizados pelo usuário para que possam ser reproduzidos posteriormente de forma automatizada, e uma IDE embutida para o desenvolvimento de Visual LISPs, contando inclusive com a possibilidade de depurar os códigos programados (AUTODESK, 2022). No entanto,

as funcionalidades oferecidas pelo AutoCAD para criação de macros e de Visual LISPs são limitadas, fazendo com que o desenvolvimento de funcionalidades complexas seja um trabalho árduo, e é aí que *plugins* se destacam.

Como mencionado, *plugins* para AutoCAD, desenvolvidos por meio de APIs, permitem a programação de funcionalidades complexas. Este tipo de *software* é comumente programado em uma de duas linguagens computacionais para as quais a Autodesk oferece APIs: o C# e o VB.NET (AUTODESK, 2022). Ainda que similares e baseadas na mesma plataforma, a .NET da Microsoft (MICROSOFT, 2022), o C# (lido como C *Sharp*) e o VB (Visual Basic) possuem certas diferenças relevantes para o desenvolvimento deste trabalho, as quais são destacadas no Quadro 2.

Quadro 2 - Comparativo entre as linguagens de programação VB.NET e C#.

VB.NET	**C#**
Se pronuncia como Visual Basic .NET, o qual é uma funcionalidade e versão atualizada do Class Visual Basic 6.0.	Se pronuncia como linguagem "*C SHARP*", e pertence à família C.
Também é utilizado no desenvolvimento de várias aplicações funcionando no *framework* .NET.	É utilizado para criar uma variedade de aplicações que funcionam no *framework* .NET.
Ambas as linguagens são funcionalmente semelhantes.	Ambas as linguagens são funcionalmente semelhantes.
É uma linguagem não sensível a letras maiúsculas e minúsculas. Por exemplo, "*Hello*" e "*hello*" são o mesmo.	É uma linguagem sensível a letras maiúsculas e minúsculas. Por exemplo, "*Hello*" e "*hello*" não são o mesmo.
VB.NET suporta manuseio de erros estruturados e não estruturados.	Suporta apenas o manuseio de erro estruturado.
Eventos são automaticamente vinculados.	Eventos não são possíveis em C#.
Declaração e definição são diferentes em ambos.	Declaração e definição são diferentes em ambos.
Utiliza inglês simples para definição de estrutura. Dim x As Integer Public x As Integer = 10	Utiliza estrutura de programação simples, como C, Java, Python, C++, etc. int x; int x = 10;
Cada declaração não termina com um ponto e vírgula.	Cada declaração termina com um ponto e vírgula.

Fonte: JavaTpoint (2021).

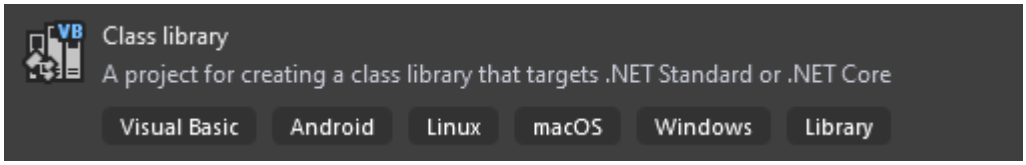
A escolha da linguagem a ser utilizada para o desenvolvimento do plugin deste trabalho está baseada em dois fatores: 1) a maneira de definir a estrutura do código;

e 2) a quantidade de material encontrado na documentação da linguagem para a finalidade desejada, assim como a quantidade de material encontrado online em fóruns e afins. De tal modo, escolhe-se a linguagem VB.NET para a programação do *plugin*.

Quanto ao IDE utilizado, escolhe-se o Microsoft Visual Studio Community 2019 (MICROSOFT, 2022) pelos seguintes fatores: além de ser gratuito, o que determina a escolha do IDE neste trabalho é o fato dessa versão ser completa, ou seja, sem limite de funcionalidades, muitas das quais vêm a ser úteis durante o desenvolvimento do *plugin*, como: a instalação das APIs necessárias; depuração do código durante testes dentro do AutoCAD e integração da plataforma GitHub (GITHUB, 2022).

O tipo de novo projeto selecionado durante a primeira inicialização do Visual Studio é o *Class* Library, conforme Figura 4, o qual suporta classes do tipo VB.NET, linguagem escolhida para a programação, e que permite a criação de arquivos do tipo *Dynamic Link Library* (.dll), que são utilizados para agregar funções e bibliotecas adicionais sem embuti-las no próprio programa (MICROSOFT, 2021). O AutoCAD suporta arquivos do tipo DLL para agregação de novas funcionalidades através de *plugins*.

Figura 4 - Novo projeto do tipo *Class Library*.

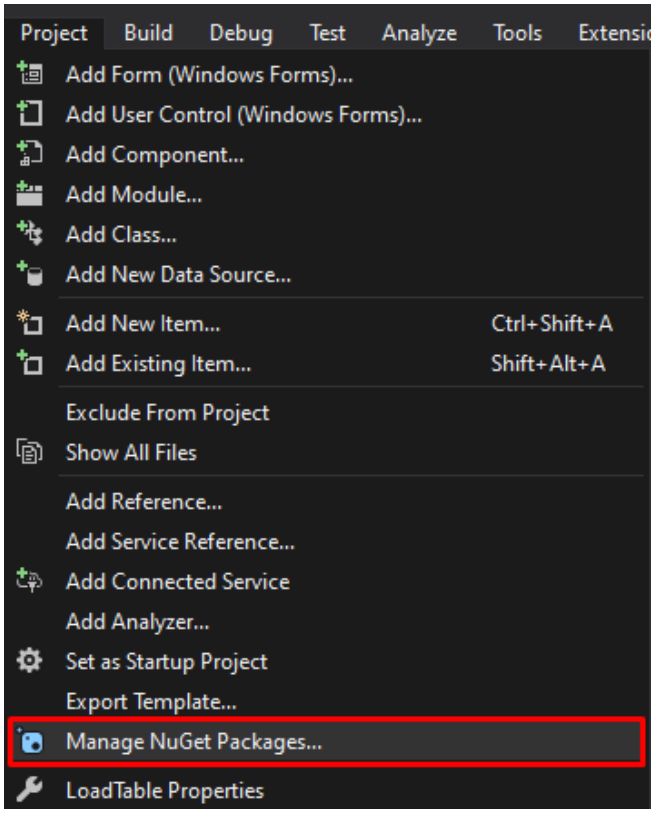


Fonte: Visual Studio (2021).

Uma das funcionalidades do Visual Studio que facilita a criação de *plugins* é a de gerenciamento de APIs, denominada NuGet Packages (MICROSOFT, 2022). Esta função, encontrada na aba Projetos como mostra a Figura 5, auxilia na obtenção de APIs, possibilitando ao usuário a instalação destas através da própria IDE.

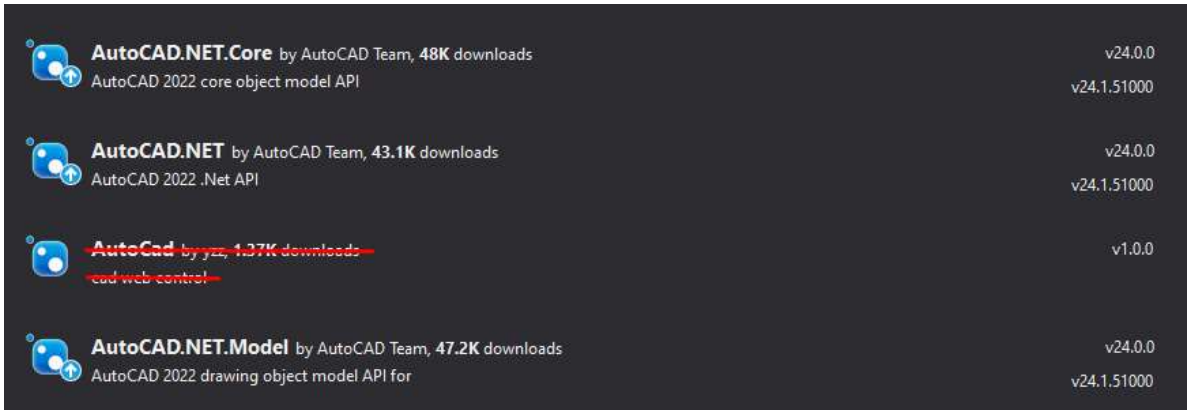
As APIs do AutoCAD utilizadas neste trabalho são a AutoCAD.NET, AutoCAD.NET.Core e AutoCAD.NET.Model, conforme mostra a Figura 6, as quais suportam o uso da linguagem VB.NET. Vale ressaltar que a versão do AutoCAD utilizada durante este projeto é a 2021, e que se deve atentar às versões das APIs instaladas, que neste caso tiveram que ser alteradas para uma versão mais antiga no momento da instalação, isso por meio da própria ferramenta de gerenciamento de APIs.

Figura 5 – Escolha do gerenciador de APIs NuGet Packages através do Visal Studio.



Fonte: Visual Studio (2021).

Figura 6 - APIs utilizadas neste trabalho.

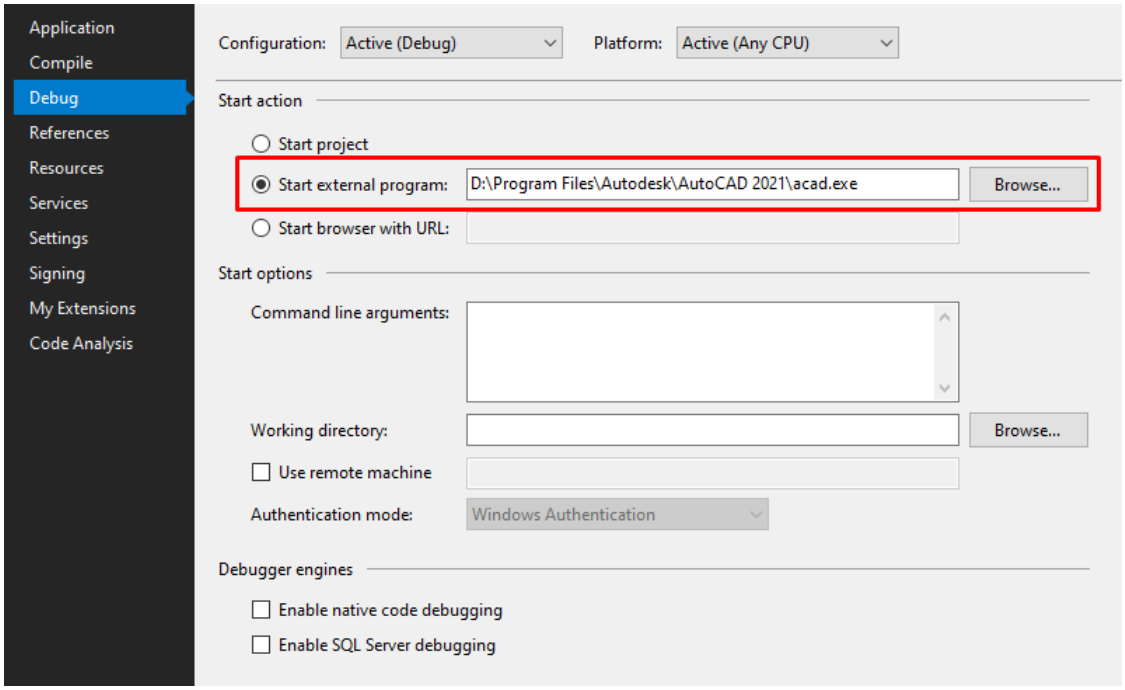


Fonte: Visual Studio (2021).

A fim de possibilitar a depuração, ou *debugging* como é o termo em inglês, de um código em desenvolvimento, pode-se adicionar uma referência a um programa externo ao Visual Studio, no qual o código será testado. No caso do *plugin* sendo desenvolvido, configura-se o executável como sendo o .exe do AutoCAD, chamado de acad.exe e presente na pasta de instalação do *software*. Essa configuração pode ser encontrada na última opção da aba Projetos, em “Nome do Projeto” *Properties* do Visual Studio.

Para testar o código, basta clicar no botão “*Start*” na porção superior da IDE, caso o executável do AutoCAD já tenha sido atrelado ao Visual Studio, conforme explicado no parágrafo anterior. Uma vez iniciado o AutoCAD, e tendo selecionado um projeto, é utilizado o comando “*NETLOAD*” para inicializar o *plugin*, selecionando-o (arquivo com extensão .dll) na pasta em que foi salvo o projeto do Visual Studio, conforme mostra a Figura 7.

Figura 7 - Configuração para uso do AutoCAD para depuração do código.



Fonte: Visual Studio (2021).

Por fim, ocorre também por meio da IDE a criação de um repositório do código na plataforma GitHub, para controle de versão durante o desenvolvimento do projeto, o qual possui uma aba dedicada dentro do Visual Studio. O GitHub permite a criação de *branches*, reduzindo o risco de cometer erros difíceis de serem revertidos, bem como o controle de versão do programa, dentre outras funcionalidades relacionadas a gerenciamento do desenvolvimento de projetos de *software* (GITHUB, 2022).

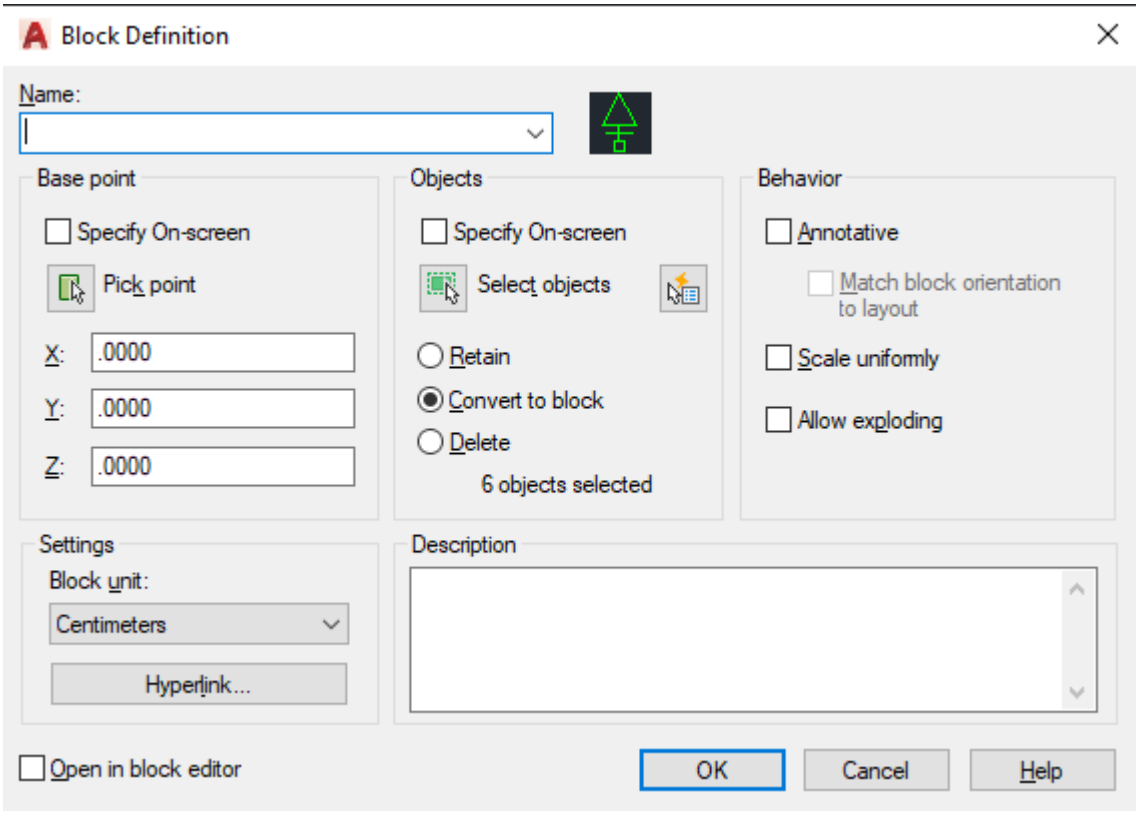
A versão completa do código desenvolvido para o *plugin* é apresentada na subseção 6.1, dentro da seção Resultados.

4.2 Criação dos Blocos Dinâmicos

A criação de um bloco dinâmico no AutoCAD se dá por meio da criação de um bloco convencional (definido na seção 2.2.10.4.3), o que por sua vez pode ser feito desenhando um objeto, composto por formas primitivas, selecionando estas e

utilizando o comando “*BLOCK*”, o que abre uma janela para definição dos parâmetros iniciais do bloco, como nome e ponto de inserção, conforme mostra a Figura 8.

Figura 8 – Interface do AutoCAD para o *block definition*.

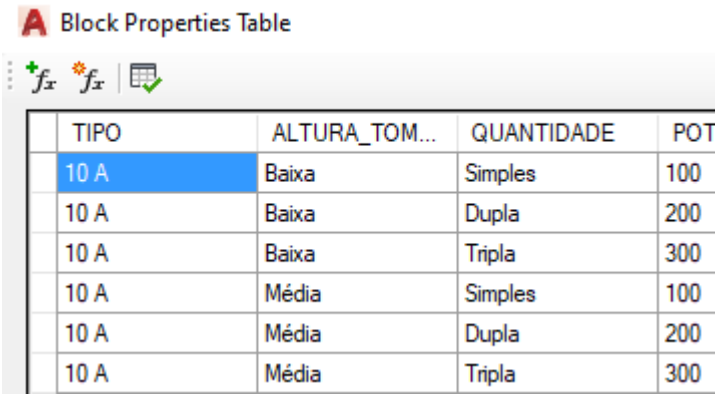


Fonte: Visual Studio (2022).

Selecionando então o bloco previamente criado, clicando com o botão direito e selecionando a opção “*Block Editor*”, o usuário é apresentado à uma nova interface, onde é possível utilizar comandos específicos para a edição de blocos. É o caso do comando “*BLOCKTABLE* (*BTABLE*)”, o qual gera uma tabela em que as colunas representam atributos criados pelo usuário, enquanto as linhas representam os valores padrões de cada visibilidade do bloco. No caso de uma tomada, por exemplo, pode-se ter atributos como: tipo (10 A, 20 A, chuveiro, iluminação de emergência, etc.); altura (baixa, média, alta); quantidade (simples, dupla, tripla), entre outros, e que, quando utilizados em conjunto com diferentes visibilidades, permite uma escolha simplificada dos diferentes objetos. A Figura 9 apresenta a tabela gerada pelo comando “*BLOCKTABLE*”.

A partir do discutido, são desenvolvidos neste trabalho blocos dinâmicos para tomadas, luminárias e interruptores, os quais são apresentados na seção 6.1.2.1.

Figura 9 – Exemplo de tabela gerada pelo comando *block table* com atributos definidos pelo usuário.

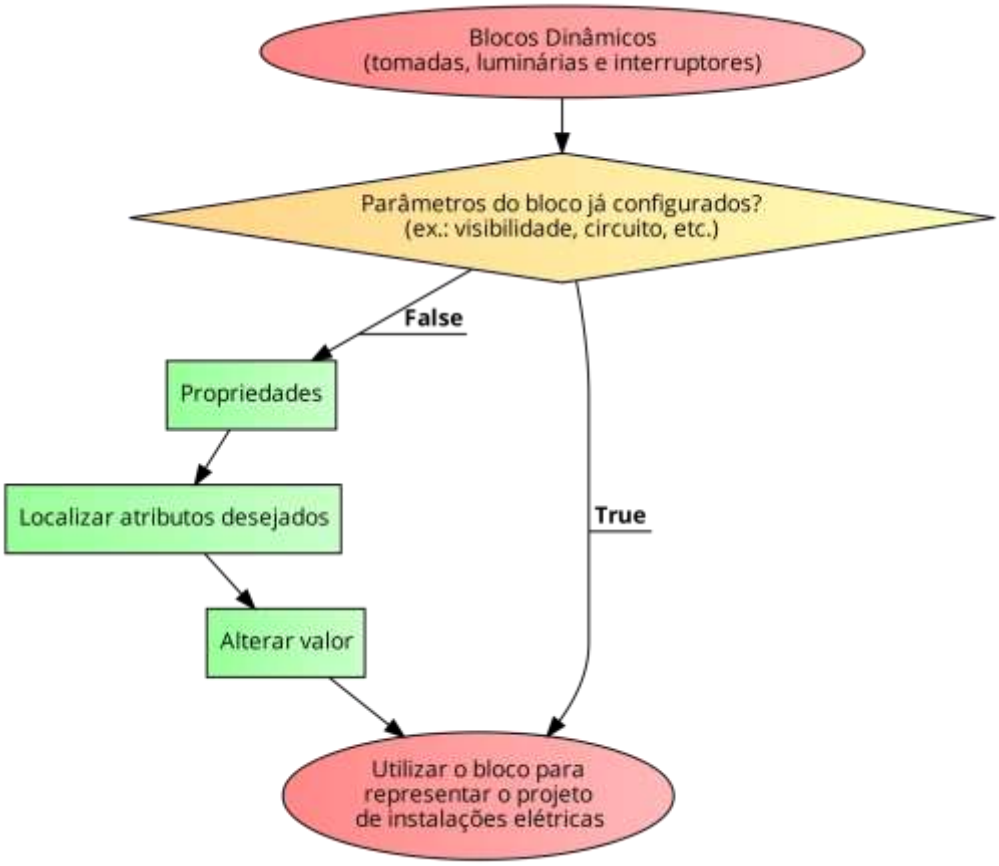


TIPO	ALTURA_TOM...	QUANTIDADE	POT
10 A	Baixa	Simples	100
10 A	Baixa	Dupla	200
10 A	Baixa	Tripla	300
10 A	Média	Simples	100
10 A	Média	Dupla	200
10 A	Média	Tripla	300

Fonte: Autor (2021).

A Figura 10 abaixo apresenta o fluxograma representando a utilização dos blocos dinâmicos para representação de projetos de instalações elétricos.

Figura 10 - Fluxograma da utilização dos blocos dinâmicos.



Fonte: Autor (2022).

4.3 Obtenção dos Parâmetros do Plugin

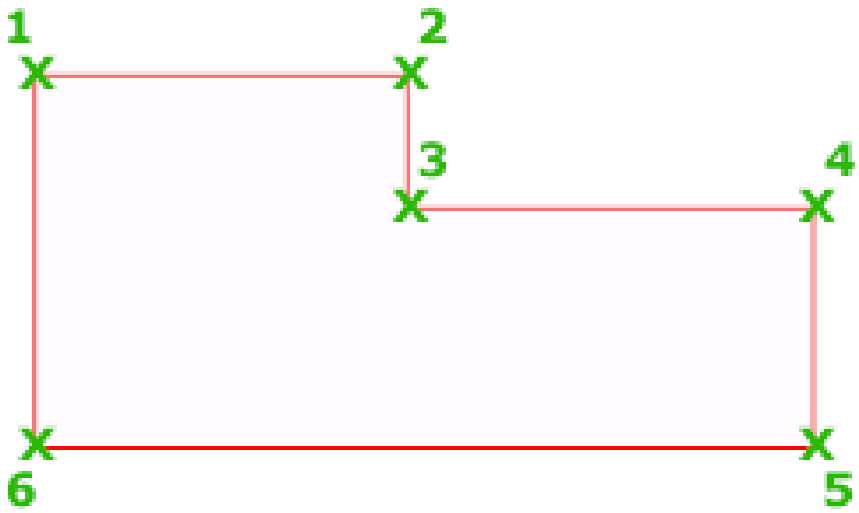
A obtenção dos parâmetros, ou informações, dos circuitos é fundamental para a montagem das tabelas e diagramas das demais etapas. Com informações como:

carga; seção dos cabos e disjuntores dos circuitos é possível dimensionar determinados componentes da instalação, como o disjuntor geral e o tamanho do barramento do quadro de distribuição, ambos definidos em normas, no caso a NBR 5410 (ABNT, 2004) e a normas NT-0 (CELESC, 1997), respectivamente.

A lógica utilizada para obtenção das informações para alimentar o plugin a partir da planta elétrica realiza uma busca por todos os objetos desejados dentro de uma área, como por exemplo, a parte interna de uma casa ou apartamento, onde os objetos especificados, no caso pontos de tomadas e luminárias, retornam seus parâmetros a fim de possibilitar o cálculo dos demais parâmetros necessários para a montagem das tabelas e diagramas.

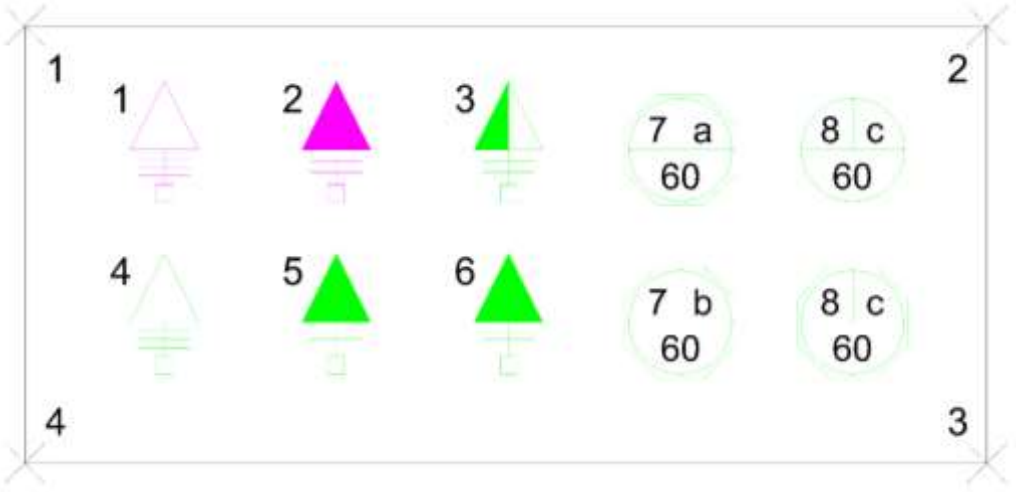
A área, mencionada no parágrafo anterior, pode ser definida pelo usuário por meio do comando nativo do AutoCAD “*PLINE*”, o qual permite a criação de um polígono, composto por linhas e/ou arcos, como no exemplo da Figura 11. Os vértices do polígono selecionado, destacados por meio do símbolo “X” nas figuras abaixo, são então utilizados para criar uma seleção em área ao redor dos blocos desejados, como mostra o exemplo da Figura 12, área essa que o *plugin* utiliza para procurar por atributos.

Figura 11 - Polígono desenhado com o uso do comando *PLINE*.



Fonte: Autor (2021).

Figura 12 - Exemplo de polígono, do tipo *PLINE*, envolvendo blocos dinâmicos.



Fonte: Autor (2021).

O Apêndice A apresenta o código utilizado para a filtragem dos blocos dinâmicos, a obtenção dos parâmetros armazenados nos atributos dos blocos dinâmicos e a criação dos objetos da classe Circuito.

O código mencionado acima começa com a filtragem do vetor de blocos “objID”, obtido previamente por meio da seleção de uma área e por meio de uma condição de seleção, onde são obtidos apenas os objetos do tipo bloco, removendo objetos como linhas e textos, os quais podem, por exemplo, representar paredes e legendas de cômodos, respectivamente. Após esta filtragem inicial, onde também são excluídos da seleção blocos que não possuem atributos, ocorre a leitura de todos os atributos em todos os blocos selecionados, e caso esses correspondam a nomes predefinidos (CIRCUITO, POTÊNCIA, CONEXÃO, SEÇÃO e DISJUNTOR), ocorre a criação de um objeto da classe Circuito, conforme a segunda metade do código no Apêndice A, onde é verificado se já existe um objeto, na lista de objetos de classe Circuito, com número de circuito igual ao sendo lido. Na Figura 13 abaixo é apresentado um *snippet*, ou trecho, do código utilizado para obtenção dos parâmetros:

Figura 13 - *Snippet* do código para obtenção dos parâmetros dos circuitos.

```
Private Function CriaCircuitos
    For Each bloco As BlockReference In blocos
      For Each objID As ObjectId In ac 'Para cada bloco selecionado…
        Case atributo
          Case "CIRCUITO"
            numeroDeAtributosCompativeis += 1
            numero = atributo.TextString
            Exit Select
          Case "POTÊNCIA"
            numeroDeAtributosCompativeis += 1
            Double.TryParse(atributo.TextString, potencia)
```

```
                Exit Select
            Case "CONEXÃO"
                numeroDeAtributosCompativeis += 1
                Integer.TryParse(atributo.TextString, conexão)
                Exit Select
            Case "SEÇÃO"
                numeroDeAtributosCompativeis += 1
                Double.TryParse(atributo.TextString, secao)
                Exit Select
            Case "DISJUNTOR"
                numeroDeAtributosCompativeis += 1
                Integer.TryParse(atributo.TextString, disjuntor)
                Exit Select
        End Select
    Next
  Next
End function
```

Fonte: Autor (2022).

Esta verificação dos circuitos é realizada na medida em que os blocos são lidos. Caso o circuito esteja sendo visto pela primeira vez, cria-se um novo objeto Circuito que é adicionado à lista, considerando o valor de sua carga como o valor total da carga daquele circuito. Caso o circuito já tenha sido visto, o que é verificado percorrendo a lista de circuitos e checando seus parâmetros até que seja encontrado uma combinação, adiciona-se o valor de sua carga à carga do objeto Circuito já criado. Os valores de seção e disjuntor, idealmente, devem coincidir entre componentes de um mesmo circuito, porém, se não for o caso, os valores de maior grandeza são considerados para que não haja subdimensionamento da proteção.

Como a lista de circuitos é populada conforme os blocos são lidos, os circuitos acabam não ficando organizados de forma crescente. Por conta disto, é realizada a ordenação da lista de circuitos baseados em sua numeração.

Pode-se dizer que a obtenção dos parâmetros obtidos através dos blocos dinâmicos está finalizada, porém, há ainda parâmetros que podem ser obtidos de forma indireta, como é o caso da quantidade de cabos por circuito, sua tensão e a(s) fase(s) destinada(s) à sua alimentação.

Os dois primeiros casos mencionados no fim do último parágrafo, no caso a quantidade de cabos e a tensão, são obtidos pelo atributo Conexão presente nos blocos, o qual pode ser monofásico, bifásico ou trifásico, o que por sua vez representa uma distribuição de cabos do tipo F+N+T, 2F+N+T e 3F+N+T, respectivamente, e uma tensão de 220 V para monofásico e 380/220 V para bifásico e trifásico, ao menos na região do vale do Itajaí, para a qual foi projetado o *plugin*.

A distribuição das cargas nas fases deve ser, idealmente, feita de forma a haver um balanceamento no somatório das cargas por fase, porém, para simplicidade do código, faz-se uma distribuição simples, onde os circuitos monofásicos são distribuídos de forma sequencial, começando em R, passando para S e T e voltando a R, onde o ciclo recomeça, considerando que a alimentação da instalação seja trifásica.

4.4 Geração Automatizada do Quadro de Cargas

Após a obtenção dos dados dos circuitos que compõem o sistema de um quadro de distribuição, torna-se possível desenvolver o quadro de cargas do sistema, o qual apresenta as informações obtidas na última seção por meio de uma tabela.

Para a montagem de um quadro de cargas, faz-se necessário entender primeiro sua estrutura. O Quadro 3 apresenta a estrutura de um quadro de cargas genérico, com suas características destacadas por meio dos textos que o compõe.

Quadro 3 - Estrutura de um quadro de cargas.

NOME DO QUADRO DE DISTRIBUIÇÃO											
TÍTULOS CONFORME INFOMAÇÕES DAS COLUNAS ABAIXO											
CIRCUITO	DESCRIÇÃO	ESQUEMA	TENSÃO (V)	POTÊNCIA TOTAL (W)	FASES	POTÊNCIA FASE R (W)	POTÊNCIA FASE S (W)	POTÊNCIA FASE T (W)	SEÇÃO (mm²)	DISJUNTOR (A)	DDR/IDR (30 mA)
TOTAL											

Fonte: Autor (2022).

Definiu-se que as primeiras duas linhas são compostas pelo cabeçalho, onde a primeira apresenta o título do quadro de distribuição, composto geralmente por um nome como “QDXX - nome do quadro”, em que “XX” é o número do quadro, e “nome do quadro” comumente representa a unidade consumidora ou o pavimento. Como

exemplos têm-se “QD12 - Apartamento 101” e “QD07 - Pavimento Lazer (Condomínio)”, no caso de um prédio residencial.

Durante o desenvolvimento do código para geração do quadro de cargas é considerado um nome genérico para o título deste, o qual pode ser modificado posteriormente à criação da tabela.

A segunda, e última, linha que compõe o cabeçalho da tabela é composta por colunas que representam os títulos das diferentes informações obtidas pelo projeto da instalação elétrica, como identificação do circuito, descrição (este que também é representado de forma genérica), esquema de alimentação, tensão, carga total, etc.

O corpo do quadro de cargas é composto pelas informações dos circuitos, cada um (circuito) representado por uma linha. Finalmente tem-se a linha do total, que representa o circuito de entrada do quadro de distribuição, onde aparece, entre outros parâmetros, o somatório da carga de todos os circuitos, total e por fase, além da seção do ramal de ligação e do disjuntor geral do quadro de distribuição, ambos obtidos a partir da tabela 8-A do adendo da norma NT-03 (CELESC, 1999).

Entendendo sua estrutura, pode-se agora utilizar o comando de criação de tabelas do AutoCAD, por meio de funções presentes em sua API, juntamente com os dados dos circuitos obtidos para automatizar a criação dos quadros de cargas. O código para criação da tabela do quadro de cargas é apresentado no Apêndice B. A Figura 14 abaixo apresenta um *snippet* do código para criação da tabela do quadro de cargas.

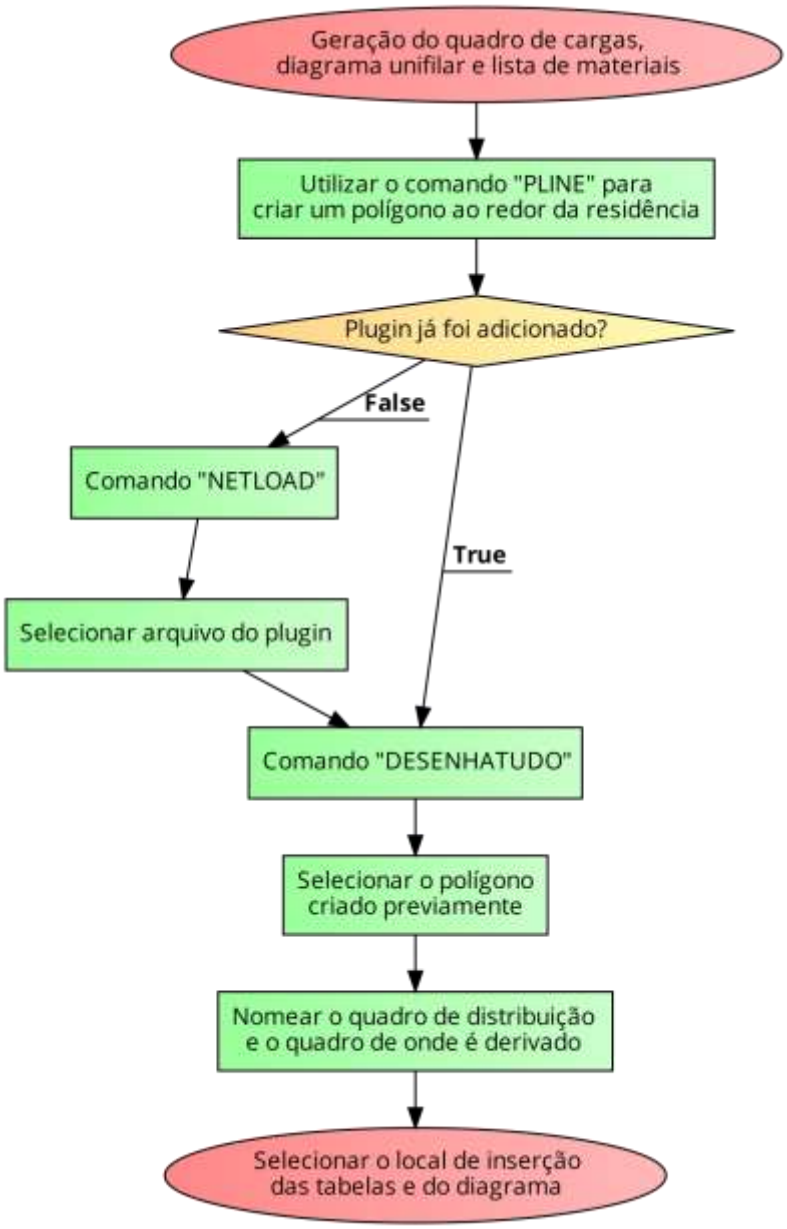
Figura 14 - *Snippet* do código para criação da tabela do quadro de cargas.

```
Private Function MontaQC(posicao As Point3d)
    Dim tabela As Table = New Table() ‘Nova tabela do tipo Table (classe do AutoCAD)
    With tabela
      ‘Formatação da tabela
      ‘Inserção dos parâmetros do projeto de instalações elétricas nas células da tabela
    End with
    Return tabela ‘Retorna a tabela
End Function
```

Fonte: Autor (2022).

Apresenta-se na seção 6.1.2.2 os resultados desta funcionalidade. Já abaixo, na Figura 15, é apresentado o fluxograma representando a utilização do comando “DESENHATUDO”, desenvolvido neste trabalho, para geração do quadro de cargas, do diagrama unifilar e da lista de materiais.

Figura 15 - Fluxograma da utilização da funcionalidade para geração do quadro de cargas, diagrama unifilar e lista de materiais.



Fonte: Autor (2022).

4.5 Geração Automatizada do Diagrama Unifilar

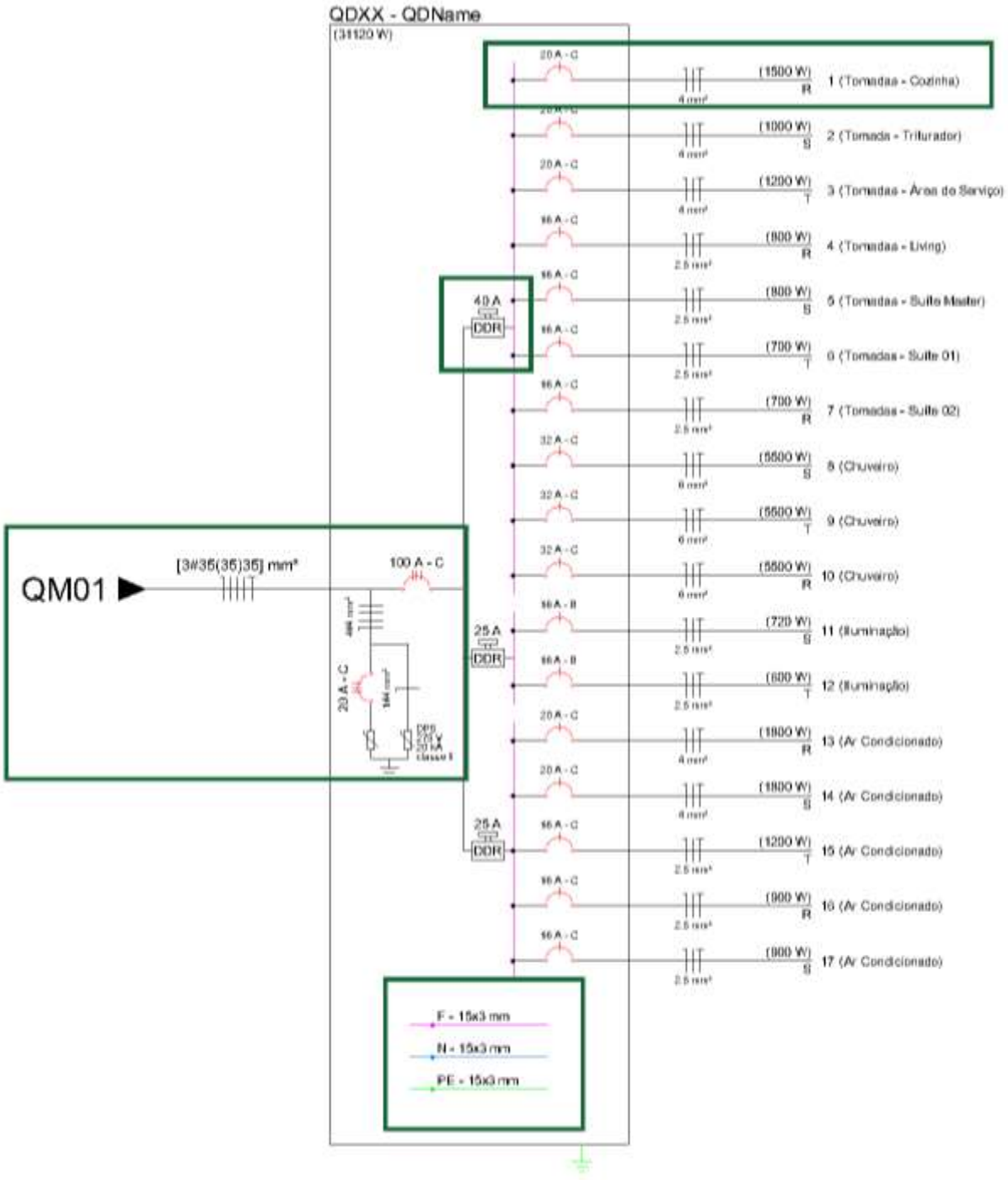
Diagrama unifilares e quadros de cargas apresentam informações similares sobre o projeto, sendo o maior diferencial entre eles o fato de que estes apresentam as informações em uma tabela, enquanto aqueles apresentam as informações por meio de diagramas, composto por símbolos e textos, como mostra a Figura 2 da página 20.

A representação do diagrama foi feita por meio de blocos, textos e formas primitivas, como linhas e polígonos. A utilização de blocos se dá pelo fato de que eles

facilitam a representação de desenhos complexos e/ou símbolos utilizados múltiplas vezes, como os utilizados para a representação dos disjuntores.

O diagrama unifilar pode ser dividido em seções que representam diferentes componentes de um quadro de distribuição. A Figura 16 apresenta a estrutura de um diagrama unifilar, destacando seus componentes principais.

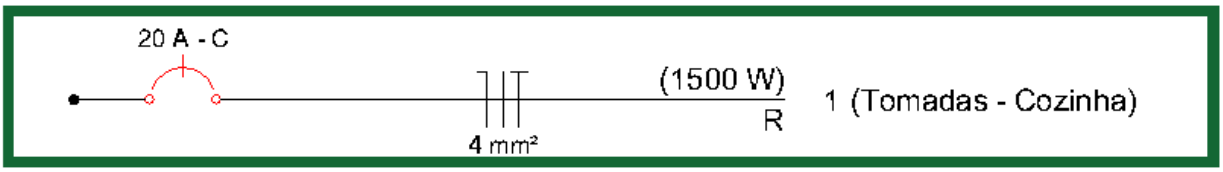
Figura 16 - Exemplo de um diagrama unifilar com os blocos que o compõe destacados em amarelo.



Fonte: Autor (2021).

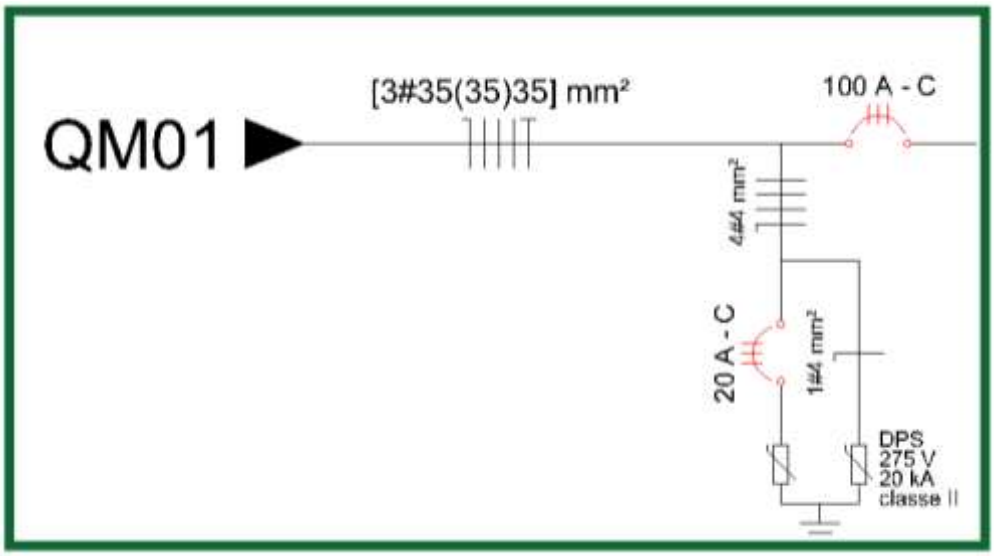
O bloco do circuito, apresentado na Figura 17, possui como atributos os valores do disjuntor, da seção do cabo, da carga, da fase, do número do circuito e sua descrição, sendo que este último é preenchido de forma genérica com o texto “(Descrição)”, sendo apresentado na possível a edição do texto após a construção do diagrama. O circuito principal, apresentado na Figura 18, possui como atributos o disjuntor, a seção e o nome de onde se deriva o quadro de distribuição, sendo que este também é preenchido de forma genérica.

Figura 17 - Bloco do circuito.



Fonte: Autor (2021).

Figura 18 - Bloco dos parâmetros de entrada.

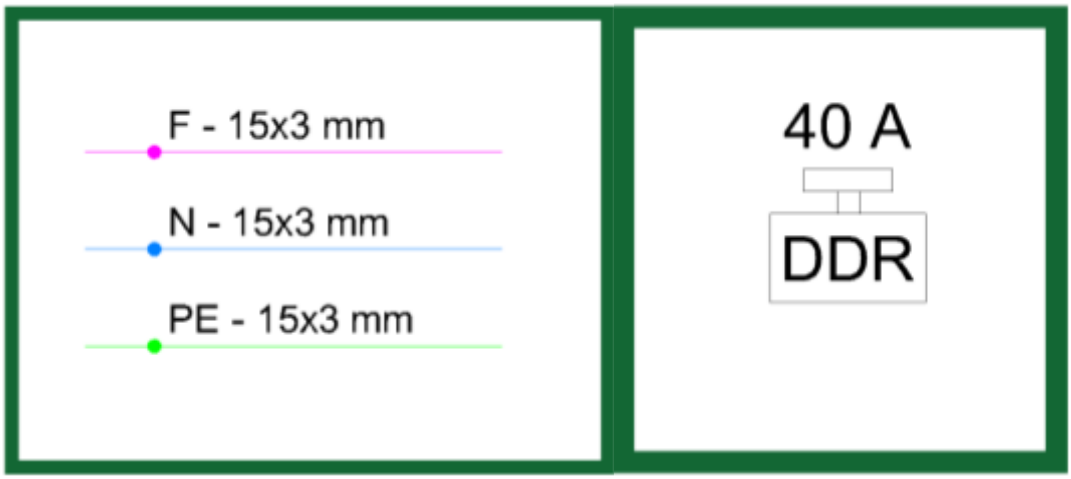


Fonte: Autor (2021).

O bloco do barramento do quadro, apresentado na seção esquerda da Figura 19, tem como atributo o tamanho do barramento, obtido comparando o valor do disjuntor do circuito principal com a tabela de dimensões de barramentos encontradas na norma NT-03 da Celesc (CELESC, 1997). Por fim tem-se o bloco do DR (Dispositivo Diferencial Residual), apresentado na seção direita da Figura 19, e que possui apenas seu valor de corrente nominal como parâmetro alterável. Ainda que programada, a funcionalidade não está implementada no *plugin*, o que se dá pela falta

de tempo para desenvolvimento de uma forma de definir quais circuitos possuem DRs, e quais não possuem.

Figura 19 - Blocos do barramento e DDR, respectivamente.



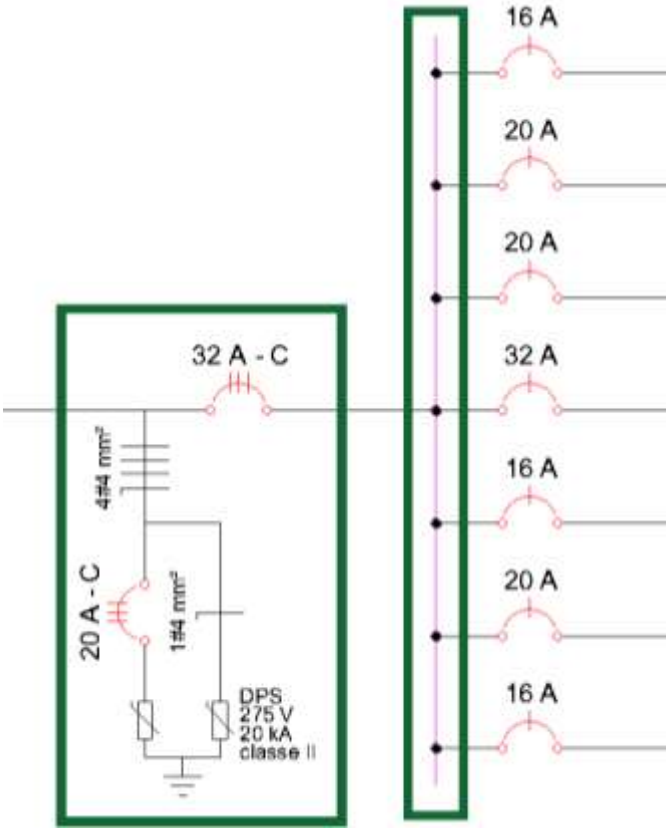
Fonte: Autor (2021).

Os demais componentes do diagrama unifilar foram representados utilizando textos, polígonos e linhas, como é o caso dos textos que representam o título do quadro e a carga total, o retângulo que envolve o quadro e as linhas que representam os barramentos e cabos, respectivamente.

Por fim, a função de desenhar diagramas unifilares, que é um método da classe QD (denominação referente ao Quadro de Distribuição), funciona por meio de coordenadas 2D e distâncias preestabelecidas. Desta forma, pede-se ao usuário selecionar um ponto na tela, no qual o diagrama será desenhado. No ponto selecionado é desenhado o primeiro circuito do QD. Logo abaixo são desenhados os demais circuitos, variando o eixo Y do ponto de inserção dos blocos destes. A posição dos demais componentes do diagrama é definida de acordo com o ponto escolhido pelo usuário, ou ponto de inserção do primeiro circuito, e o ponto de inserção do último circuito, que pode ser obtido facilmente pela quantidade total de circuitos e a distância preestabelecida entre um circuito e outro.

Desta forma, o barramento que conecta os circuitos é desenhado utilizando o comando linha, e que tem como vértices o primeiro e último circuito mais uma margem para cima e para baixo. Já o circuito principal é conectado no ponto médio entre o ponto de inserção do primeiro e último circuito. A Figura 20 abaixo destaca o circuito principal, à esquerda, e o barramento, à direita.

Figura 20 - Circuito principal e barramento destacados em amarelo, da esquerda para a direita.



Fonte: Autor (2021).

O Apêndice C apresenta a função utilizada para desenhar o diagrama unifilar no AutoCAD. Nela é possível observar a utilização de outras funções, as quais tem como objetivo desenhar pequenas seções do diagrama.

Como mencionado na seção acima, a Figura 15 na página 44, apresenta o fluxograma demonstrando a utilização da funcionalidade desta seção.

4.6 Criação do Diagrama de Classes e Reestruturação do Código Inicial

Como descrito no início da seção 4, a criação do diagrama de classes e reestruturação do código inicialmente desenvolvido na unidade curricular de PI III, é a primeira etapa pós criação do protótipo do *plugin*. Esta etapa tem como objetivo abordar problemas decorridos do planejamento da criação das classes do código inicial, como funções em classes indevidas e dificuldade na criação de novas funções por falta de clareza em sua estrutura. É criado então um diagrama de classes UML baseado nas funcionalidades propostas pelo *plugin*, porém não levando em consideração o código em si.

O diagrama é montado com o uso do *software* StarUML (MKLABS, 2022), e é então utilizado como base para reescrever o código do *plugin* orientado a objetos (ver

seção 2.2.2). Os resultados desta etapa, como o diagrama de classes UML criado e os benefícios disto, são abordados na seção 6.1.1 dos resultados.

4.7 Geração Automatizada da Lista de Materiais

Após a reestruturação do código para orientação a objetos, foram desenvolvidas as novas funcionalidades do *plugin*. A funcionalidade de geração de listas de materiais é baseada nos comandos previamente desenvolvidos, onde as informações contidas nos blocos dinâmicos, necessárias para representar graficamente os diferentes componentes elétricos, são utilizadas, possibilitando a montagem de uma tabela populada com as informações da lista de materiais do projeto de instalações elétricas em questão.

O desenvolvimento dessa funcionalidade requer a atualização dos blocos dinâmicos criados na versão inicial do *plugin*, adicionando a eles novos parâmetros esperados de uma lista de materiais, como, por exemplo, a quantidade e/ou dimensão de caixas de passagem, módulos de tomadas e interruptores, tipo de luminária, etc.

Desta maneira, um bloco utilizado para representar um ponto de tomada de 10 A baixa (altura em relação ao solo) e simples (quantidade de módulos unitária) pode apresentar os seguintes parâmetros, como demonstrados na Tabela 1 abaixo:

Tabela 1 - Exemplo dos materiais que compõem um ponto de tomada baixa 10 A simples.

Descrição	Quantidade (un.)
Caixa de passagem 4”x2”	1
Espelho tampa simples	1
Módulo de tomada 10 A	1

Fonte: Autor (2022).

Quanto à como filtrar os blocos que devem ter seus parâmetros adicionados à lista de materiais, pode-se fazer uma varredura por todos os blocos selecionados na seleção em área, mencionada na seção 4.3, procurando por todos os atributos que contenham uma combinação específica de caracteres em seu nome, como por exemplo “LM” (Lista de Materiais).

A criação de uma classe denominada “Materiais” auxilia na verificação de materiais já encontrados e na soma de novas quantias de materiais à lista. Por fim cria-se uma tabela, a qual é adicionada ao *Model Space*, como mencionado na seção 4.4.

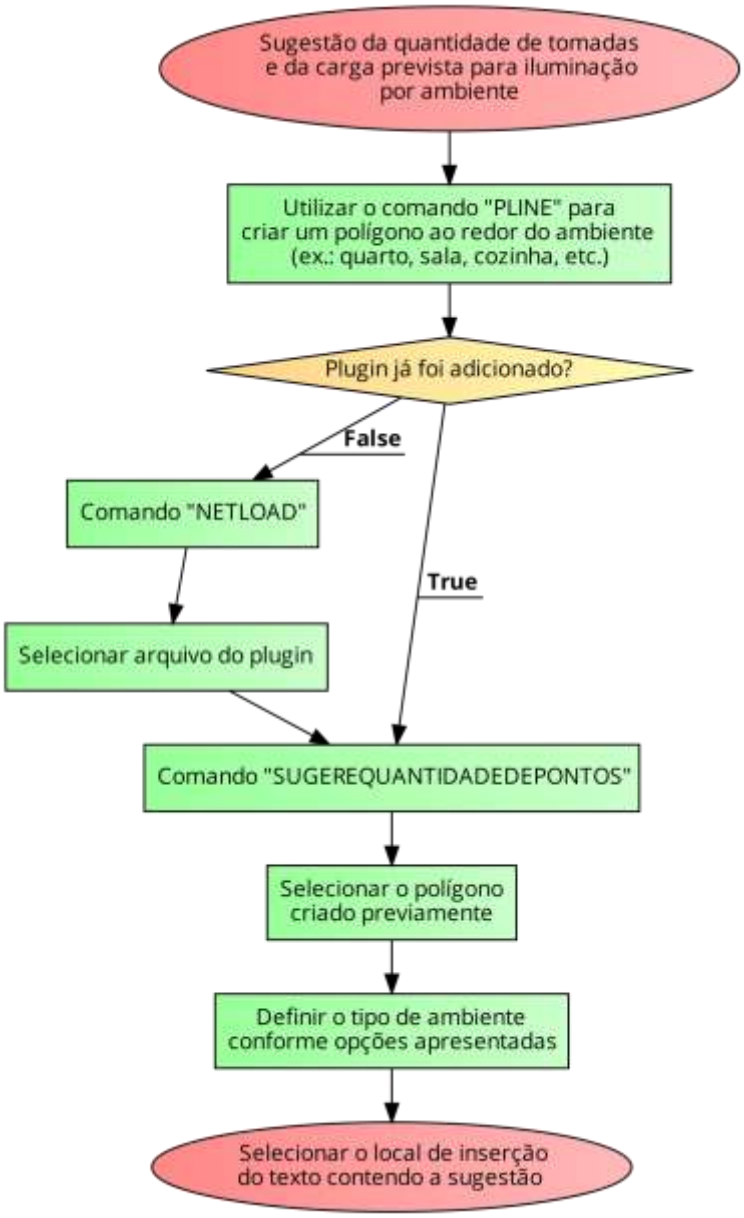
Os resultados desta funcionalidade são apresentados na seção 6.1.2.4 dos resultados. Ressalta-se que a Figura 15, apresentada na página 44, demonstra por meio de um fluxograma a utilização da funcionalidade apresentada nesta seção.

4.8 Sugestão da Quantidade de Tomadas e Carga Prevista para Iluminação por Ambiente

A funcionalidade para apresentar uma sugestão da quantidade de tomadas e carga prevista para iluminação por ambiente se baseia na norma NBR 5410 (ABNT, 2004, p. 190-192). As informações do perímetro e área do ambiente são obtidas por meio de um polígono desenhado pelo usuário, o qual deve envolver o ambiente desejado, método similar ao apresentado na seção 4.3. A funcionalidade também pergunta ao usuário qual o tipo de local (“seco”, como salas e quartos, ou “molhado”, como cozinhas e banheiros), uma vez que isto afeta o cálculo.

O Apêndice D apresenta parte do código utilizado para a criação da funcionalidade em questão. Os resultados são apresentados na seção 6.1.2.5. Já abaixo, na Figura 21, apresenta-se o fluxograma da utilização da funcionalidade para sugestão de quantidade de pontos de tomadas e carga prevista para iluminação por ambiente.

Figura 21 - Fluxograma da utilização da funcionalidade para sugestão de quantidade de pontos de tomadas e carga prevista para iluminação por ambiente.



Fonte: Autor (2022).

4.9 Exportação das Tabelas em Formato CSV (Não Implementada)

A funcionalidade para exportar as tabelas, no caso o quadro de cargas e a lista de materiais, em formato CSV tem como objetivo permitir ao usuário a edição dessas por meio de programas computacionais especializados na criação e edição de tabelas, como o Microsoft Excel e o Google Planilhas, caso o projetista deseje alterar alguma informação de forma mais prática.

Existe no AutoCAD uma função para exportação de tabelas em formato CSV, porém o intuito desta nova funcionalidade é de facilitar seu uso, integrando-a às

demais funcionalidades criadas até então, uma vez que esses programas apresentam uma maior facilidade de uso e mais funcionalidades para a manipulação de tabelas do que as tabelas encontradas no AutoCAD.

Essa integração se daria por meio do uso das APIs do AutoCAD, permitindo o uso da função em conjunto com as demais funcionalidades de geração de tabelas. Porém, não é encontrado, durante a realização deste trabalho, uma forma de utilizar a função em questão por meio da API do AutoCAD.

Uma alternativa para isso é o desenvolvimento de uma funcionalidade para gerar o arquivo CSV do zero, criando um documento novo e salvando as informações obtidas pelas demais funcionalidades a ele. Contudo, esta solução demanda tempo adicional de desenvolvimento e de teste junto às demais funcionalidades do plugin, e por este motivo é decidido não implementar a exportação de tabelas em CSV na versão atual do plugin.

5 TESTES DE USABILIDADE

As seguintes subseções descrevem as etapas do processo de definição dos procedimentos utilizados para realização dos testes de usabilidade.

5.1 Definição do Plano de Testes

5.1.1 Objetivos

Os testes propostos possuem os seguintes objetivos:

- Avaliar a eficiência e eficácia das funcionalidades implementadas;
- Avaliar a facilidade de uso do *plugin*;
- Identificar erros;
- Obter sugestões de novas funcionalidades.

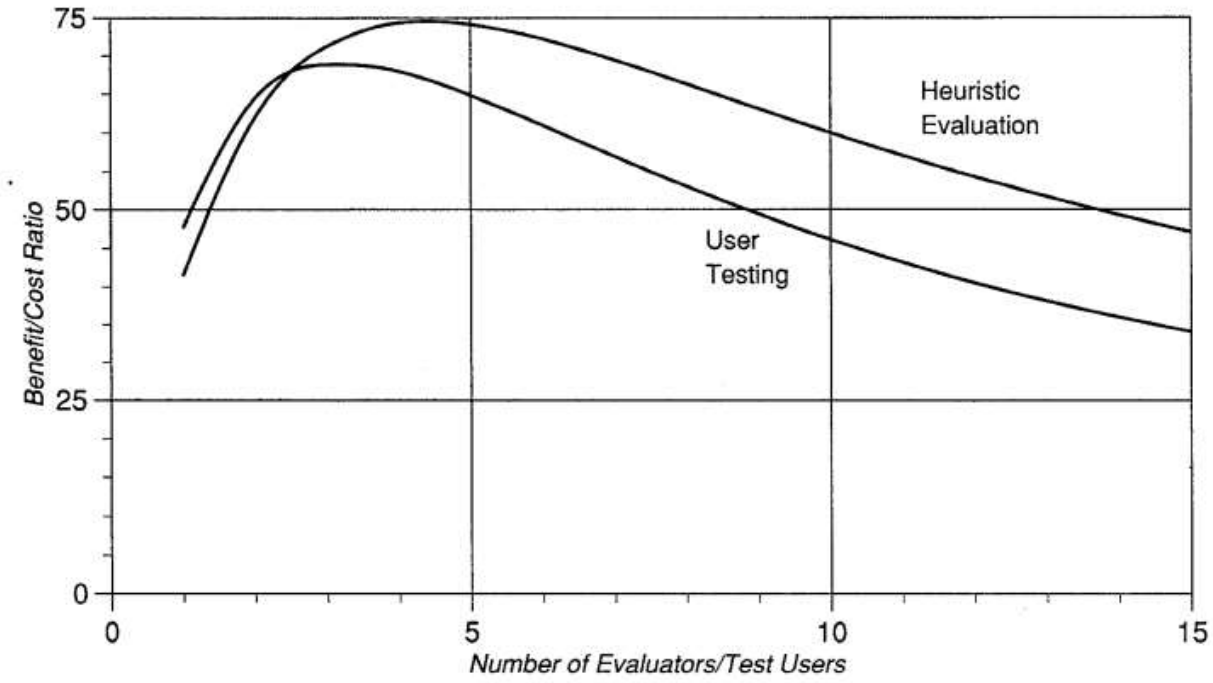
5.1.2 Definição e Recrutamento dos Participantes

A definição do perfil dos participantes dos testes é crucial para a coleta de dados válidos para o estudo. Levando em consideração o problema abordado neste trabalho, define-se o perfil dos participantes como profissionais que trabalham desenvolvendo projetos de instalações elétricas.

Quanto à quantidade de participantes recrutados, leva-se em consideração o estudo de Nielsen e Landauer (1993, p. 206-213), o qual apresenta a relação entre benefícios/custos e a quantidade de participantes em testes de usabilidade, conforme

a Figura 22. Nela é possível observar que o número ótimo de participantes, quando tratando de benefícios e custos de uma sessão de testes de usabilidade, é entre 4 e 5. Nielsen e Landauer também concluem que um número tão pequeno de avaliações é válido ao considerarem que, uma vez que descobertos os primeiros problemas e serem aplicadas as devidas alterações para correção destes, o projeto será alterado e novos testes são cabíveis, processo este que pode ser repetido diversas vezes, sugerindo então que as primeiras instâncias de um projeto sejam avaliadas de maneira menos minuciosa (NIELSEN; LANDAUER, 1993).

Figura 22 - Relação entre benefícios/custos e a quantidade de participantes em testes de usabilidade.



Fonte: Nielsen e Landauer (1993, p. 212).

Sendo assim, levando em consideração o estudo de Nielsen e Landauer (1993) e o fato de que este é o primeiro grupo de teste focal do *plugin*, propõe-se o recrutamento de apenas quatro participantes, com perfis resumido no Quadro 4. Porém, salienta-se que os instrumentos de testes criados permitem testes de usabilidade futuros com um número maior de participantes, desta forma, não invalidando a metodologia utilizada neste trabalho.

Quadro 4 - Informações sobre os participantes dos testes de usabilidade.

Avaliador	Formação Profissional	Cargo	Tempo de Experiência em Projetos de Instalações Elétricas
Participante 1	Superior Incompleto (Engenharia Civil)	Projetista	< 3 anos
Participante 2	Superior Incompleto (Engenharia Civil)	Auxiliar de Projetista	< 6 meses
Participante 3	Superior Incompleto (Engenharia Civil)	Auxiliar de Projetista	< 1 ano
Participante 4	Superior Completo (Engenharia Elétrica e Engenharia Civil	Engenheiro Projetista	> 10 anos

Fonte: Autor (2022).

Salienta-se que, por questões como conflito de interesses, o autor deste projeto participou da etapa de testes apenas como moderador.

5.1.3 Definição do Local dos Testes

A definição do local de testes engloba diversos fatores, como por exemplo a facilidade de deslocamento do moderador, participantes e equipamentos, a disponibilidade de um ambiente que permita a condução dos testes e espaço suficiente para assistentes, caso seja necessário o auxílio para, por exemplo, cronometrar o tempo de alguma tarefa ou configurar algum equipamento.

A princípio planejou-se realizar os testes de maneira remota por conta do estado atual de pandemia decorrente do COVID-19, porém, por conta da flexibilização dos decretos, definição da aplicação do teste como moderada e possíveis complicações na configuração de uma máquina virtual, tem-se que os testes foram realizados de forma presencial. Aproveitando o perfil do usuário, o qual é apresentado na seção 5.1.2, utiliza-se um escritório de engenharia (MK ENGENHARIA, 2022) para a realização da sessão de testes do *plugin*. Salienta-se que a utilização deste local foi propriamente autorizada por meio de conversas prévias com o proprietário da empresa.

Um resumo com as demais informações sobre o local dos testes pode ser encontrado no

Quadro 5 abaixo.

Quadro 5 - Resumo da definição do local dos testes.

Itens	Descrição
Local	Escritório de engenharia
Equipamen-tos	Computador com sistema operacional Windows 10
Programas computacio-nais	AutoCAD 2019 e *plugin*
Documentos	Questionário pós testes https://docs.google.com/forms/d/1LARohK9UhtuPZzkM4AW2hnkbHGystElOmvq2Pbz4qh4/viewanalytics
Moderador	Matheus D'Amaral (autor do trabalho)
Assistentes	Nenhum

Fonte: Autor (2022).

5.2 Questões

O Quadro 6 apresenta as questões apresentadas aos participantes após os testes, por meio do Google Forms e divididas em seções referentes às diferentes funcionalidades do *plugin*.

Quadro 6 - Questões dos testes de usabilidade.

Categoria	Questão
Feedback quanto à funcionalidade de sugestão de quantidade de pontos de tomadas e carga prevista para iluminação.	Durante o teste do *plugin* a funcionalidade de sugestão de quantidade de pontos de tomadas e carga destinada à iluminação facilitou o seu trabalho? Se sim, destaque os pontos positivos desta funcionalidade. Se não, o que você sugere para torná-la mais útil? (resposta livre)
	Indique em quais tipos de projeto você utilizaria a funcionalidade de sugestão de quantidade de pontos de tomadas e carga destinada à iluminação: (seleção múltipla: residencial unifamiliar; residencial multifamiliar; comercial; industrial; ou nenhum)

Feedback quanto à funcionalidade de blocos de tomadas, luminárias e interruptores.	Comente sobre a facilidade ou dificuldade que você teve em alternar entre as diferentes visibilidades dos blocos (ex.: trocar de tomada baixa 10 A simples para tomada média 20 A dupla). (resposta livre)
Feedback quanto à funcionalidade de Geração do quadro de cargas, do diagrama unifilar e da lista de materiais.	Destaque os pontos positivos e/ou negativos da funcionalidade de geração de tabelas e diagramas do projeto. Comente, por exemplo, sobre a facilidade de uso e resultados obtidos. (resposta livre)
	O que você mudaria ou adicionaria à essas funcionalidades? (resposta livre)
Feedback geral quanto às funcionalidades testadas.	Você se deparou com algum erro durante o teste? Se sim, mencione brevemente o problema. (resposta livre)
	Você percebeu uma redução no tempo de desenvolvimento do projeto utilizando as novas funcionalidades? (sim ou não)
	O que você achou da curva de aprendizado das funcionalidades testadas? (escala de muito difícil a muito fácil)
	Quais novas funcionalidades você sugere para serem acrescentadas ao plugin? (resposta livre)
	Você considerou o manual de usuário suficiente para o aprendizado das funcionalidades testadas? Indique pontos que poderiam ser melhorados. (resposta livre) Questão desconsiderada por não ter sido utilizado o manual, decorrente do tempo limitado disponível para aplicação dos testes, porém compensado pela aplicação moderada destes.
	Caso deseje, utilize este campo para deixar comentários adicionais relacionados às funcionalidades testadas. (resposta livre opcional)

Fonte: Autor (2022).

5.3 Roteiro dos testes

No Quadro 7 abaixo é apresentado o roteiro utilizado para a realização dos testes, baseado no desenvolvimento de projetos de instalações elétricas, permitindo o teste de todas as funcionalidades implementadas.

Quadro 7 - Roteiro dos testes.

Etapa	Instruções
1	Utilize o comando PLINE para desenhar um polígono ao redor do perímetro de cada ambiente.
2	Utilize o comando SUGEREQUANTIDADEPONTOS, selecione um dos polígonos criados e digite as opções conforme requisitado na tela. O comando criará um texto com a sugestão da quantidade mínima de tomadas e luminárias do ambiente, o qual pode ser colocado em qualquer lugar do *Model Space.*
3	Utilize os blocos fornecidos para o desenvolvimento do projeto elétrico. Ao terminar de inserir os blocos, atribua a eles números de circuitos.
4	Finalizando a representação dos componentes elétricos, desenhe um polígono ao redor de toda a casa, novamente utilizando o comando PLINE.
5	Utilize então o comando DESENHATUDO, clique no polígono criado na etapa 4, e insira as informações requisitadas na tela. Por fim escolha onde deseja inserir o quadro de cargas, o diagrama unifilar e a lista de materiais.

Fonte: Autor (2022).

5.4 Preparação dos Materiais para Testes

Os materiais criados e utilizados neste trabalho para os testes de usabilidade são apresentados nas subseções abaixo.

5.4.1 Cenário de Testes

A fim de testar as funcionalidades do *plugin*, foi desenvolvida uma planta baixa arquitetônica de uma casa genérica e fictícia dentro do AutoCAD, apresentada na Figura 23, na qual são realizadas as simulações do desenvolvimento de projetos de instalações elétricas. Também estão inclusas neste arquivo, que contém a planta baixa, as instruções contidas no roteiro de testes apresentado no Quadro 7 acima, caso os participantes requeiram a leitura do passo a passo, assim como os blocos dinâmicos necessários para representação dos componentes elétricos e correto funcionamento das demais funcionalidades do *plugin*, os quais são apresentados na Figura 23 e Figura 24, respectivamente.

Figura 23 - Planta baixa e instruções utilizadas durante os testes de usabilidade.

Fonte: Autor (2022).

Figura 24 - Blocos dinâmicos disponíveis durante os testes de usabilidade.

Fonte: Autor (2022).

5.4.2 Questionário Pós Testes

Como mencionado previamente, o questionário pós testes para coleta de *feedback* por parte dos participantes foi desenvolvido na plataforma Google Forms, e a Figura 25 apresenta um trecho da página do formulário. As questões apresentadas aos participantes são as mesmas apresentadas no Quadro 6, na página 55, e o formulário na íntegra pode ser acessado, para fins de visualização, através deste endereço eletrônico:

https://docs.google.com/forms/d/1LARohK9UhtuPZzkM4AW2hnkbHGystElOmvq2Pbz4qh4/viewanalytics.

Figura 25 - Trecho do formulário pós testes, desenvolvido no Google Forms.

Fonte: Autor (2022).

5.5 Condução dos Testes

A sessão de testes é conduzida de forma moderada, ou seja, há engajamento, quando necessário, entre moderador e participantes durante os testes.

5.6 Coleta dos Dados

A coleta dos dados dos testes ocorre por meio do questionário pós testes, gerenciado por meio da ferramenta Google Forms.

5.7 Análise dos Dados

A análise dos dados se baseia nos objetivos propostos pelos testes, conforme seção 5.1.1. A fim de facilitar a análise, os dados são agrupados da seguinte forma, como sugerido por Rubin e Chisnell (2008, p. 254-267):

- Resultados de questões com respostas limitadas (ex.: múltipla escolha) são somados, a fim de visualizar tendências;
- Resultados de questões com respostas livres são agrupados baseados no quesito similaridade, como por exemplo o agrupamento por respostas positivas e negativas.

5.8 Apresentação dos Dados

Os resultados das análises dos dados coletados são apresentados na seção 6.2.

6 RESULTADOS

6.1 O *Plugin*

Para o *plugin* desenvolvido neste trabalho foram implementados os seguintes métodos, conforme o Quadro 8:

Quadro 8 - Métodos criados para o *plugin*.

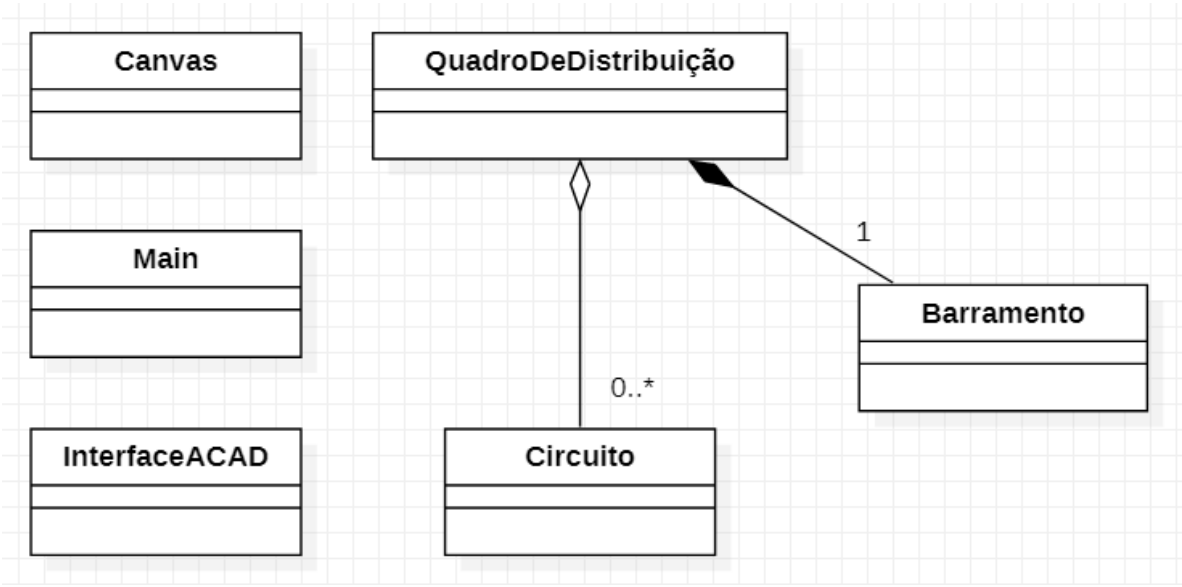
Comando	Descrição
DESENHATUDO	Desenha, a partir do projeto de instalações elétricas, o quadro de cargas, o diagrama unifilar e a lista de materiais.
SUGEREQUANTIDADEDEPONTOS	Gera, de acordo com o ambiente, uma sugestão relatando a quantidade mínima de pontos de tomadas e a carga mínima prevista para iluminação.

Fonte: Autor (2022).

6.1.1 Diagrama UML e Reestruturação do Código para Orientação a Objeto

O diagrama de classes UML desenvolvido neste trabalho, para a reestruturação da versão inicial do código, pode ser observado no Apêndice E. A Figura 26 abaixo apresenta resumidamente a relação entre as classes desenvolvidas.

Figura 26 – Relação entre classes da versão inicial do *plugin* desenvolvido em PI III.



Fonte: Autor (2022).

Abaixo, na Tabela 2, é apresentada uma comparação das métricas geradas pelo IDE entre a versão original do código, desenvolvida em Projeto Integrador III como prova de conceito, e a versão final do código orientado a objeto, já com a nova estrutura de classes e novas funcionalidades implementadas. É possível acessar o código completo do *plugin* desenvolvido neste trabalho através do endereço eletrônico: https://github.com/mattdamaral/TCC.

Tabela 2 - Comparativo entre a versão inicial do *plugin* e a versão final (dados do Visual Studio).

Dado	Versão Inicial	Versão Final
Índice de Facilidade de Manutenção	71	76
Complexidade Ciclomática	298	291
Profundidade de Herança	3	3
Acoplamento de Classe	78	88
Linhas de Código-Fonte	106	106
Linhas de Código Executável	870	802

Fonte: Autor (2022).

A explicação completa de cada dado analisado pode ser encontrada na página sobre valores de métricas de código no site da Microsoft (2021), porém é válido destacar dois dos resultados. O primeiro é o índice de facilidade de manutenção, o qual, por meio de um valor de 0 a 100, define a facilidade relativa de manter o código, onde:

- 0 a 9: Baixa facilidade de manutenção;
- 10 a 19: Manutenção moderada; e
- 20 a 100: Fácil manutenção.

Ainda que apresentando valores similares, é possível perceber um aumento na facilidade na manutenção do código na versão final.

Outro ponto válido de destaque é o referente à quantidade de linhas de código executável. É notável que, apesar da adição de duas novas funcionalidades, obtém-se uma redução na quantidade de linhas executáveis, novamente confirmando uma otimização do código.

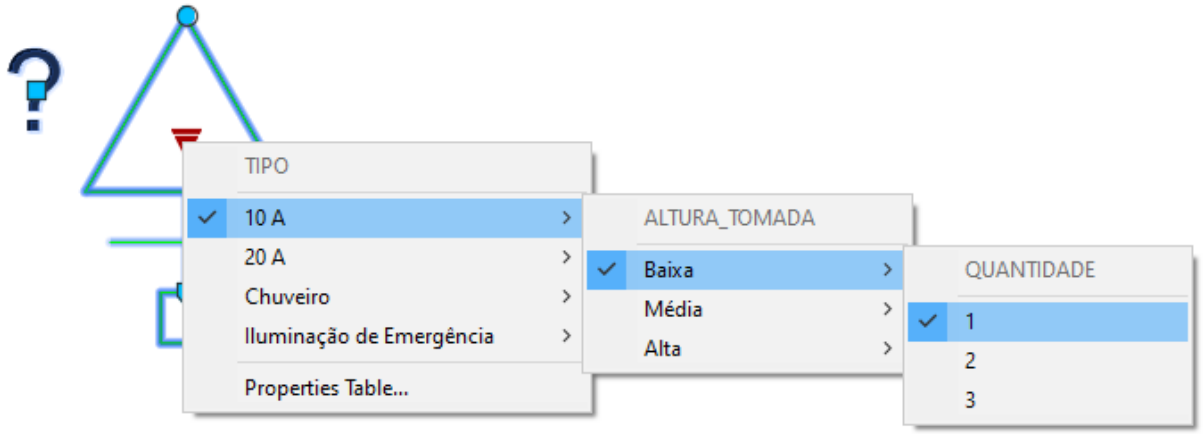
Percebe-se a partir destes resultados que o desenvolvimento de um diagrama de classes orientado a objeto aumenta a organização do código desenvolvido, possibilitando um código mais otimizado e facilitando futuras manutenções.

6.1.2 Funcionalidades do Plugin

6.1.2.1 Blocos Dinâmicos

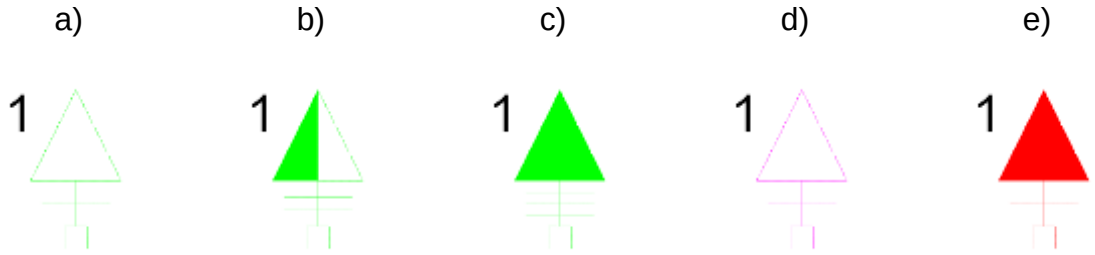
Como mencionado na seção do desenvolvimento dos blocos dinâmicos (seção 4.2), são desenvolvidos blocos de tomadas, luminárias e interruptores. A Figura 27 apresenta um bloco de tomada com diferentes visibilidades atribuídas a si, permitindo diferentes simbologias e valores padrão em um único bloco dinâmico, as quais (simbologias) são controladas por meio do parâmetro *block table*. Já a Figura 28 apresenta algumas das diferentes visibilidades do bloco dinâmico da tomada mencionada, onde a) representa uma tomada baixa de 10 A, b) representa duas tomadas médias de 10 A, c) representa três tomadas altas de 10 A, d) representa uma tomada baixa de 20 A e, por fim, e) representa uma tomada alta para iluminação de emergência.

Figura 27 – Diferentes opções de tomadas em único bloco dinâmico.



Fonte: Autor (2021).

Figura 28 – Exemplos de diferentes visibilidades contidas no bloco dinâmico de tomada.



Fonte: Autor (2021).

A Figura 29 mostra os 6 blocos dinâmicos criados para a representação de projetos de instalações elétricas e uso das funcionalidades do *plugin* propostas neste trabalho, sendo um de interruptor, um de luminária e quatro de tomadas, conforme destacado abaixo. No total, entre os 6 blocos dinâmicos, são criadas 158 visibilidades diferentes. Salienta-se também que todos os blocos possuem parâmetros predefinidos de carga, seção, disjuntor e conexão, os quais podem ser facilmente alterados por meio da janela de propriedades.

Figura 29 –Blocos dinâmicos criados para o *plugin* e algumas de suas diferentes visibilidades.

Fonte: Autor (2022).

6.1.2.2 Quadro de Cargas

Com o intuito de comparar os resultados gerados durante a geração do quadro de cargas e do diagrama unifilar, foi desenvolvida uma planta baixa de uma casa fictícia para testes do *plugin* durante seu desenvolvimento, a qual também foi utilizada nos testes de usabilidade. A planta baixa em questão é apresentada na Figura 23 da seção 5.4.1, e é novamente apresentada na Figura 30 abaixo, porém agora com os pontos elétricos já lançados. Destaca-se que os eletrodutos e fiação não são representados, uma vez que não são necessários para o funcionamento das funcionalidades do *plugin*, além de que dificultaria a visualização dos demais componentes.

Figura 30 - Planta baixa de instalações elétricas, utilizada nos testes de usabilidade, já com os pontos elétricos lançados.

Fonte: Autor (2022).

Utilizando o comando denominado “DESENHATUDO”, o qual foi desenvolvido neste trabalho, são gerados o quadro de cargas, o diagrama unifilar e a lista de materiais. O quadro de cargas é apresentado na Figura 31.

Figura 31 - Quadro de cargas gerado pelo *plugin*.

Quadro de Cargas											
Circuito	Descrição	Esquema	Tensão (V)	Potência (W)	Fases	R (W)	S (W)	T (W)	Seção (mm²)	Disjuntor	DDR (30 mA)
1	??	F+N+T	220	2000	R	2000	0	0	4	20	-
2	??	F+N+T	220	1000	S	0	1000	0	4	20	-
3	??	F+N+T	220	2400	T	0	0	2400	4	20	-
4	??	F+N+T	220	1200	R	1200	0	0	2.5	16	-
5	??	F+N+T	220	5500	S	0	5500	0	6	32	-
6	??	F+N+T	220	380	T	0	0	380	2.5	16	-
7	??	F+N+T	220	300	R	300	0	0	2.5	16	-
Total		3F+N+T	380/220	12780	R+S+T	3500	6500	2780	10	40	-

Fonte: Autor (2022)

Cabe aqui pontuar que na versão atual do plugin falta acrescentar a descrição dos circuitos e a da definição dos DDRs/IDRs. Uma possível solução para permitir o

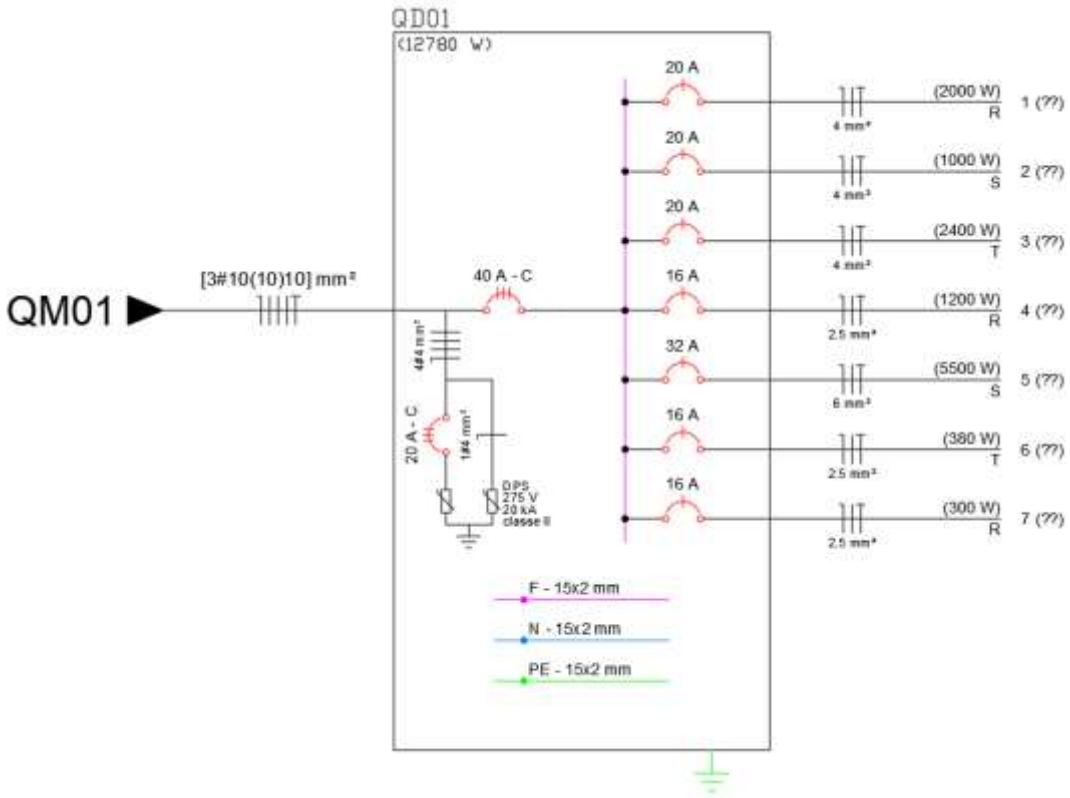
controle desses parâmetros é a adição de novos atributos aos blocos dinâmicos criados, o que permitirá o controle direto desses parâmetros. Quanto aos demais parâmetros, como: potência; seção e disjuntor, considera-se que esses foram calculados ou dimensionados de maneira esperada, podendo salientar, por exemplo, a correta divisão dos circuitos nas diferentes fases, o dimensionamento da proteção de entrada e a identificação dos esquemas, tensões e fases.

6.1.2.3 Diagrama Unifilar

O diagrama unifilar gerado pelo *plugin*, apresentado na Figura 32, apresenta resultados similares aos do quadro de cargas, onde a descrição e os DDRs/IDRs não são representados e os demais parâmetros apresentam o funcionamento esperado.

Nota-se a partir destes resultados preliminares, ainda não considerando os testes de usabilidade, que os problemas encontrados na geração do quadro de cargas e no diagrama unifilar puderam ser resolvidos simultaneamente através do uso do *plugin*.

Figura 32 - Diagrama unifilar gerado pelo *plugin*.



Fonte: Autor (2022).

6.1.2.4 Lista de Materiais

A partir do observado na Figura 33, em relação aos valores esperados, pode-se concluir que a lista de materiais é a funcionalidade mais eficaz até o momento, automatizando completamente, e corretamente, uma tarefa demorada e altamente passível de erros humanos.

Figura 33 - Lista de materiais gerada pelo *plugin* a partir do projeto da planta baixa apresentada na Figura 30.

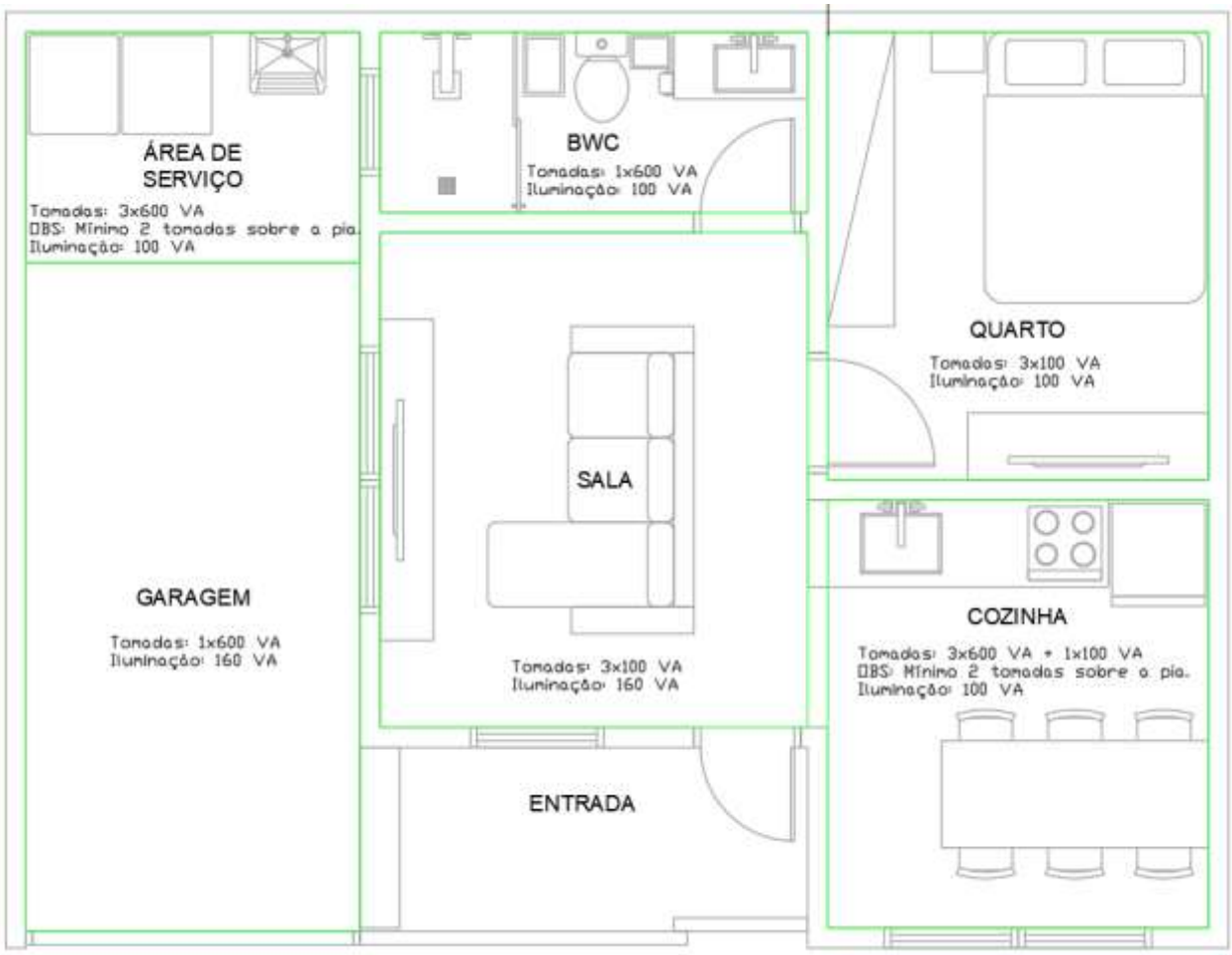
Lista de Materiais		
Material	Quantidade	Unidade
ARANDELA	5	un
CAIXA 4X2	31	un
CAIXA OCTOGONAL	7	un
ESPELHO DUPLO	5	un
ESPELHO FURO	1	un
ESPELHO SIMPLES	20	un
LUMINÁRIA TETO	7	un
MODULO INTERRUPTOR COMUM	6	un
MODULO INTERRUPTOR PARALELO	2	un
MODULO TOMADA 10A	12	un
MODULO TOMADA 20A	10	un

Fonte: Autor (2022).

6.1.2.5 Sugestão da Quantidade de Pontos de Tomadas e Carga Prevista para Iluminação

A precisão desta funcionalidade é validada pela testagem de diferentes áreas e perímetros pré-determinados, comparando os resultados obtidos com uma planilha com valores esperados, também baseada na norma NBR 5410 (ABNT, 2004). A sugestão gerada, em forma de texto, pode ser adicionada a qualquer local do *Model Space*. Abaixo, na Figura 34, é apresentado o resultado do uso da funcionalidade em questão.

Figura 34 - Sugestão de quantidade de tomadas e carga prevista para iluminação gerada pelo *plugin*.



Fonte: Autor (2022).

6.1.2.6 Exportação das Tabelas em Formato CSV

Como mencionado na seção 4.9, a implementação da funcionalidade para exportação das tabelas geradas pelo *plugin* em formato CSV, mais especificamente o quadro de cargas e o diagrama unifilar, apesar de prevista na proposta o TCC, não foi possível de ser realizada dentro do escopo de tempo de desenvolvimento deste trabalho.

6.2 Testes de Usabilidade

Nesta seção são apresentados os resultados referentes aos dados obtidos por meio dos testes de usabilidade.

6.2.1 Resultados Gerais

O Quadro 9 apresenta um resumo sobre a conclusão e a curva de aprendizado dos participantes dos testes de usabilidade. Neste é possível observar que todos os participantes conseguiram concluir os testes, passando por todas as etapas destes, e que houve uma unanimidade quanto à curva de aprendizado, a qual se mostrou muito fácil (em uma escala de muito difícil a muito fácil). É notável pelos resultados unânimes que o *plugin* é intuitivo.

Quadro 9 - Informações gerais sobre os testes de usabilidade do *plugin*.

Participante	Concluiu os Testes	Curva de Aprendizado
Participante 1	Sim	Muito fácil
Participante 2	Sim	Muito fácil
Participante 3	Sim	Muito fácil
Participante 4	Sim	Muito fácil

Fonte: Autor (2022).

6.2.2 *Feedback* sobre a Sugestão da Quantidade de Tomadas e Carga Prevista para Iluminação por Ambiente

No Quadro 10 abaixo é apresentada uma relação entre as respostas obtidas, referentes à facilidade do uso e vantagens da funcionalidade de sugestões, e objetivos esperados. Ao correlacionar e analisar as respostas é possível observar um ganho de eficiência.

Quadro 10 - Relação entre respostas e objetivos esperados para a sugestão da quantidade de tomadas e carga prevista para iluminação por ambiente.

Resposta	Objetivo Alcançado	Frequência
Facilitou o trabalho	Eficiência	3
Aumentou a produtividade	Eficácia e Eficiência	1

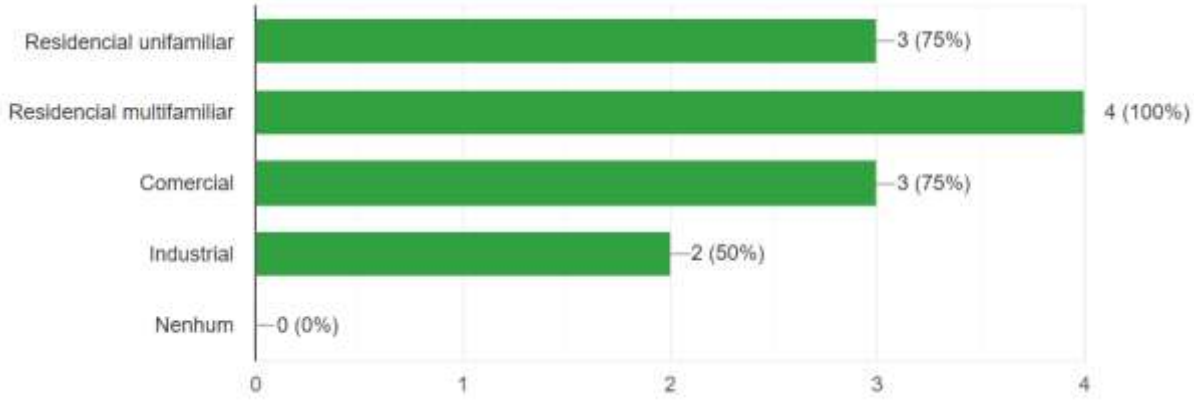
Fonte: Autor (2022).

Quanto aos tipos de projeto em que a funcionalidade pode ser útil, esperava-se em um primeiro momento uma maior quantidade de respostas indicando projetos residências, tanto unifamiliar quanto multifamiliar, uma vez que segundo a NBR 5410 a previsão de cargas é aplicável a:

> ...locais utilizados como habitação, fixa ou temporária, compreendendo as unidades residenciais como um todo e, no caso de hotéis, motéis, flats, apart-hotéis, casas de repouso, condomínios, alojamentos e similares, as acomodações destinadas aos hóspedes, aos internos e a servir de moradia a trabalhadores do estabelecimento. (2004, p. 182)

Entretanto foram observadas múltiplas respostas indicando seu uso em empreendimentos comerciais e industriais, como apresentado na Figura 35, o que pode indicar um possível uso errôneo da funcionalidade. Uma possível solução para isto seria a implementação de um aviso indicando projetos recomendados para uso da funcionalidade, evitando o uso da funcionalidade em ambientes não aplicáveis.

Figura 35 – Respostas quanto ao tipo de projeto onde a sugestão da quantidade de tomadas e carga prevista para iluminação por ambiente se mostraria útil.



Fonte: Autor (2022).

6.2.3 *Feedback* sobre blocos dinâmicos

A utilização dos blocos dinâmicos, segundo participantes dos testes, apresentaram os seguintes resultados qualitativos:

- Fácil compreensão;
- Fácil utilização;
- Evita a inserção de outros blocos no desenho.

Desta forma, os blocos dinâmicos facilitam o desenvolvimento de projetos, otimizando estes ao agrupar diversos símbolos comumente utilizados em uma quantidade pequena de blocos dinâmicos.

6.2.4 *Feedback* sobre Geração das Tabelas e Diagrama

Em relação a funcionalidade de geração das tabelas e diagrama, os resultados obtidos a partir das respostas referentes à facilidade de uso foram similares aos

obtidos na seção 6.2.2, com pontos válidos de menção sendo a eliminação de repetições e a evitação de erros humanos.

Já referente às mudanças sugeridas para estas funcionalidades, têm-se as seguintes sugestões por parte dos participantes:

- Alterar o valor padrão do disjuntor da luminária, o qual não é contabilizado corretamente na geração do quadro de cargas e diagrama unifilar;
- Atrelar as descrições do quadro de cargas ao diagrama unifilar, ou vice-versa, permitindo com que a alteração realizada em um reflita no outro;
- Apresentar uma pré-visualização das tabelas e do diagrama antes de inseri-los, evitando a sobreposição destes.

Estas sugestões se mostram valiosas no processo de aperfeiçoamento do *plugin*, podendo ser utilizadas no desenvolvimento de trabalhos futuros baseados neste trabalho.

CONCLUSÃO

No início deste trabalho de conclusão de curso é constatado que a falta de projetos de instalações elétricas, em unidades residenciais novas, apresenta um risco para a população, e que o complexo processo de desenvolvimento destes dificulta a situação. Desta maneira, faz-se necessário o estudo de uma forma de aumentar a eficiência e eficácia deste processo. Para tanto, na seção 1.3 foram definidos objetivos gerais e específicos para guiar o desenvolvimento deste trabalho.

Quanto a estes, tem-se que o desenvolvimento do modelo conceitual orientado a objetos, e a reestruturação da versão inicial do *plugin* baseada nesse, foram bem-sucedidos, comprovando a eficácia do planejamento do código, conforme resultados apresentados na seção 6.

Em relação ao objetivo específico de implementação de novas funcionalidades, conforme apresentado na seção 1.3.2, foram implementadas a geração de listas de materiais e sugestão de quantidade de tomadas e carga prevista para iluminação por ambiente, o que, juntamente com as funcionalidades de geração do quadro de cargas e do diagrama unifilar permitem um ganho de produtividade no desenvolvimento de projetos de instalações elétricas. Salienta-se que a adição das novas funcionalidades não ocorre de forma integral, uma vez que certas limitações de tempo dificultaram a integração da funcionalidade para exportação de tabelas, em formato CSV, ao *plugin*, conforme mencionado na seção 6.1.2.6.

Este ganho é confirmado por meio da análise dos resultados dos testes de usabilidade apresentados na seção 6.2, último dos objetivos específicos propostos. Por meio dos testes realizados com quatro profissionais da área, foi possível obter dados fundamentais para a determinação da eficiência da ferramenta, evidenciando também sua eficácia.

De tal maneira, o problema, da dificuldade referente ao desenvolvimento de projetos de instalações elétricas, pode ser solucionado por meio do projeto e desenvolvimento de ferramentas complementares à *softwares* já existentes. Analisando outros problemas similares, conclui-se que esta solução pode ser aplicada à outras disciplinas, também no âmbito de projetos de obras civis, como é o caso de projetos preventivos e hidrossanitários.

Sendo assim, constata-se que o objetivo de comprovar a viabilidade do projeto e desenvolvimento de um *plugin* para AutoCAD, para a automatização de etapas do desenvolvimento de projetos de instalações elétricas, foi alcançado, conforme análise dos resultados do projeto, extraídos do desenvolvimento e testes da ferramenta computacional proposta. Com isso, resposta à hipótese **“É viável projetar e desenvolver um plugin para AutoCAD dedicado à automatização de etapas da criação de projetos de instalações elétricas?” é afirmativa**, conforme demonstrado no corpo deste TCC.

CONSIDERAÇÕES FINAIS

Ressalta-se que a metodologia da pesquisa teve caráter qualitativo, natureza aplicada, método hipotético-dedutivo e tem como referencial teórico material bibliográfico, documental e de levantamento de campo, conforme seção 3.

No decorrer da pesquisa, deparamo-nos com certas limitações, como a falta de dados quantitativos para uma comparação mais aprofundada entre o método convencional e o método onde é utilizado o *plugin*, isto no que se diz respeito ao desenvolvimento de projetos de instalações elétricas. Também é notada uma dificuldade em encontrar material bibliográfico para auxílio no desenvolvimento de *plugins* para AutoCAD.

Por fim, propõe-se como possíveis trabalhos futuros:

- A atualização do *plugin*, a fim de resolver os problemas e implementar as sugestões que surgem durante os testes de usabilidade;
- A comprovação do aumento de eficiência gerado pela solução proposta por meio de testes quantitativos, comparando o tempo de desenvolvimento de projetos de instalações elétricas com e sem o uso de *plugins* customizados, por sua vez, possibilitando uma análise do tempo necessário para desenvolvimento da ferramenta e o tempo economizado nos projetos;
- E a proposição de novas soluções para os problemas discutidos, realizando ao final uma comparação dos métodos.

REFERÊNCIAS

ABNT. **NBR 5410**. 2008. ed. Rio de Janeiro: [s.n.], 2004.

ADENAIYA, O. A.; ADEJUGBAGBE, J. A. **Impacts of Waste Management of Construction Industry in Nigeria**. International Conference on Covd-19 Pandemic: An Enabled TVET Towards Sustainable Survival and Recovery of Global Economics. [S.l.]: [s.n.]. 2020. p. 8-12.

ADITIVOCAD. AditivoCAD 3. **AditivoCAD.**, 2022. Disponivel em: <https://www.aditivocad.com/aditivocad3.php>. Acesso em: 18 Dezembro 2021.

ALTOQI. QiElétrico. **QiBuilder**, 2022. Disponivel em: <https://hotsite.altoqi.com.br/qibuilder/qieletrico/>. Acesso em: 18 Dezembro 2021.

ARIJELOYE, B. T.; AKINRADEWO, F. O. Assessment of Materials Management on Building Projects in Ondo State, Nigeria. **World Scientifc News**, Akure, p. 177, 2016.

AUTODESK. Usar macros para automatizar o trabalho. **Autodesk**, 2014. Disponivel em: <https://knowledge.autodesk.com/pt-br/search-result/caas/sfdcarticles/sfdcarticles/PTB/Use-macros-to-automate-work.html>. Acesso em: 20 Março 2022.

AUTODESK. Ready, Set, Action! Learn How to Create Action Macros in AutoCAD. **AutoCAD**, 2019. Disponivel em: <https://blogs.autodesk.com/autocad/action-macros-in-autocad/>. Acesso em: 20 Março 2022.

AUTODESK. About Dynamic Blocks. **Autodesk Knowledge Network**, 2020. Disponivel em: <https://knowledge.autodesk.com/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/AutoCAD-Core/files/GUID-3C2FB982-3AF6-437B-987F-4EDF81EA0662-htm.html>. Acesso em: 30 Dezembro 2021.

AUTODESK. About Model Space and Paper Space. **Autodesk Knowledge Network**, 2020. Disponivel em: <https://knowledge.autodesk.com/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/AutoCAD-Core/files/GUID-990538B6-DDA1-4190-BCC0-BB5BA94C9879-htm.html>. Acesso em: 30 Dezembro 2021.

AUTODESK. About Getting Started with Visual LISP (Visual LISP IDE). **Autodesk**, 2022. Disponivel em: <https://help.autodesk.com/view/OARX/2022/ENU>. Acesso em: 20 Março 2022.

AUTODESK. AutoCAD. **Autodesk**, 2022. Disponivel em: <https://www.autodesk.com.br/products/autocad/overview>. Acesso em: 12 Fevereiro 2022.

AUTODESK. AutoCAD ObjectARX SDK Developer Center. **Autodesk**, 2022. Disponivel em: <https://www.autodesk.com/developer-network/platform-technologies/autocad/objectarx>. Acesso em: 20 Março 2022.

AUTODESK. BIM. **Autodesk**, 2022. Disponivel em: <https://www.autodesk.com.br/solutions/bim>. Acesso em: 13 Fevereiro 2022.

AUTODESK. Revit. **Autodesk**, 2022. Disponivel em: <https://www.autodesk.com.br/products/revit/mep>. Acesso em: 18 Dezembro 2021.

BILLINGS, C. E. **Aviation Automation:** The Search for a Human-Centered Approach. Mahwah: Lawrence Erlbaum Associates, Inc, 1997.

CELESC. **NT-03:** Fornecimento de Energia Elétrica à Edifícios de Uso Coletivo. Florianópolis: [s.n.], 1997.

CELESC. **Adendo NT-03:** Fornecimento de Energia Elétrica a Edifícios de Uso Coletivo. Florianópolis: [s.n.], 1999.

CELESC. **Sistema PEP - Projeto Elétrico de Particulares**. Florianópolis: [s.n.], 2017.

CELESC. **N.321-0001:** Fornecimento de Energia Elétrica em Tensão Secundária de Distribuição. Florianópolis: Celesc, 2019.

DA LUZ, G. B.; KUIAWINSKI, D. L. Mecanização, Autonomação e Automação - Uma Revisão Conceitual e Crítica. **XIII SIMPEP**, Bauru, 6-8 Novembro 2006.

DA SILVA JR., P. T. D. S. **Desenvolvimento de um Plugin para AutoCAD®, para Cálculo de Quantitativo de Insumos e Auxílio no Projeto Orçamentário**. CESMAC. Maceió. 2017.

DUTTA, M.; SETHI, K. K.; KHATRI, A. Web Based Integrated Development Environment. **International Journal of Innovative Technology and Exploring Engineering (IJITEE)**, Bhopal, Março 2014.

ELÇI, A. et al. **Smart Computing Paradigms:** New Progress and Challenges. Singapore: Springer Nature Singapore Pte Ltd, v. 2, 2018.

ELLIS, P. G.; TORCELLINI, P. A.; CRAWLEY, D. B. **Energy Design Plugin**. IBPSA-USA SimBuild 2008 Conference. Berkeley: National Renewable Energy Laboratory. 2008. p. An Energyplus Plugin for Sketchup.

GARRIDO, F. et al. **On the Way for Materializing cMOOC Requirements:** An Experience Dealign with Plugins on Moodle Platform. VIII Congresso Brasileiro de Informática na Educação (CBIE 2019). Salvador: [s.n.]. 2019. p. 567.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de Pesquisa**. 1ª. ed. Porto Alegre: UFRGS, 2009.

GIL, A. C. **Métodos e Técnicas de Pesquisa Social**. 6ª. ed. São Paulo: [s.n.], 2008.

GITHUB. GitHub. **GitHub**, 2022. Disponivel em: <https://github.com>. Acesso em: 19 Fevereiro 2022.

GOLDBERG, K. **What is Automation**. Berkeley, p. 1. 2011.

GOOGLE. Forms. **Google**, 2022. Disponivel em: <https://workspace.google.com/intl/pt-BR/products/forms>. Acesso em: 5 Março 2022.

GOOGLE. Sheets. **Google**, 2022. Disponivel em: <https://www.google.com/sheets/about/>. Acesso em: 20 Fevereiro 2022.

GOSALA, B. et al. Automatic Classification of UML Class Diagrams Using Deep Learning Technique: Convolutional Neural Network. **Applied Sciences**, Basel, p. 1, November 2021.

HASSENZAHL, M.; BURMESTER, M.; KOLLER, F. User Experience Is All There Is. **De Gruyter**, Oldenbourg, 2021. 202-204.

JAVATPOINT. VB.NET vs C#. **JavaTpoint**, 2021. Disponivel em: <https://www.javatpoint.com/vb-net-vs-c-sharp>. Acesso em: 19 Fevereiro 2022.

LAUKAT, T. Software Vendor Selection Using Proof of Concepts. **Seminar IT - Management in the Digital Age**, Wedel, 2020. 3-4.

LIMA FILHO, D. L. **Projetos de Instalações Elétricas Prediais**. 12. ed. São Paulo: Érica, 2011.

MÄS, S. et al. **Generic Schema Descriptions for Comma-Separated Values Files of**. International Conference on Geographic Information Science. Lund: AGILE. 2018. p. 1.

MICROSOFT. O que é uma DLL. **Docs**, 2021. Disponivel em: <https://docs.microsoft.com/pt-br/troubleshoot/windows-client/deployment/dynamic-link-library>. Acesso em: 2022 Março 2022.

MICROSOFT. Valores de Métricas de Código. **Microsoft**, 2021. Disponivel em: <https://docs.microsoft.com/pt-br/visualstudio/code-quality/code-metrics-values?view=vs-2022>. Acesso em: 12 Março 2022.

MICROSOFT. Desenvolva aplicativos.NET. **Visual Studio**, 2022. Disponivel em: <https://visualstudio.microsoft.com/pt-br/vs/features/net-development/>. Acesso em: 20 Março 2022.

MICROSOFT. Microsoft Excel. **Microsoft 365**, 2022. Disponivel em: <https://www.microsoft.com/pt-br/microsoft-365/excel>. Acesso em: 19 Fevereiro 2022.

MICROSOFT. Quickstart: Install and use a package in Visual Studio (Windows only). **Docs**, 2022. Disponivel em: <https://docs.microsoft.com/en-us/nuget/quickstart/install-and-use-a-package-in-visual-studio>. Acesso em: 20 Março 2022.

MICROSOFT. Visual Studio Community. **Visual Studio**, Redmond, 2022. Disponivel em: <https://visualstudio.microsoft.com/pt-br/vs/community/>. Acesso em: 29 Janeiro 2022.

MK ENGENHARIA. Escritório de engenharia, Balneário Camboriú, 25 Fevereiro 2022. Disponivel em: <https://mkengenharia.eng.br/sobre-engenharia.html>. Acesso em: 26 Fevereiro 2022.

MKLABS. StarUML. **StarUML**, 2022. Disponivel em: <https://staruml.io>. Acesso em: 19 Fevereiro 2022.

MOORE, B. **Object Oriented vs Functional Programming - Library Instruction in a Bite-Sized Functional Model**. Brick & Click. Maryville: Northwest Missouri State University. 2020. p. 5-10.

MULTIPLUS. PRO-Elétrica. **Multiplus**, 2022. Disponivel em: <https://multiplus.com/software/pro-eletrica/index.html>. Acesso em: 18 Dezembro 2021.

NIELSEN, J.; LANDAUER, T. K. **A Mathematical Model of the Finding of Usability Problems**. INTERCHI. Morristown: [s.n.]. 1993. p. 206-213.

OMURA, G. Mastering AutoCAD 2010 and AutoCAD LT 2010. In: OMURA, G. **Mastering AutoCAD 2010 and AutoCAD LT 2010**. 1. ed. Indianapolis: Wiley Publishin, Inc, 2009. Cap. 18.

PROCOBRE. **Panorama da Situação das Instalações Elétricas Prediais no Brasil**. International Copper Association Brazil. São Paulo, p. 19-21. 2014.

RAM, K. **Git Can Facilitate Greater Reproducibility and Increased Transparency in Science**. BioMed Central. Berkeley, p. 1-7. 2013.

RASHID, M. et al. **Indoor Electrical Installation Design Layout Using IOT**. 2021 International Congress of Advanced Technology and Engineering (ICOTEN). Taiz: IEEE. 2021. p. 1.

RUBIN, J.; CHISNELL, D. **Handbook of Usability Testing**. 2nd. ed. Indianapolis: Wiley Publishing, Inc, 2008.

SAYARY, S. E.; OMAR, O. Designing a BIM Energy-Consumption Template to Calculate and Achieve a Net-Zero-Energy House. **Solar Energy**, 1 March 2021. 315.

SHELBY, P. et al. **BrAPI - An Application Programming Interface for Plant Breeding Applications**. [S.l.]: Bioinformatics, v. 35, 2019.

SILVA, A. L. D.; LAMOUNIER, E.; CARDOSO, A. A CAD System to Develop Electrical Installation. **12th International Conference on Geometry and Graphics**, Salvador, 6-10 Agosto 2006. 2.

THE WORLD BANK. Access of Electricity (% of population). **World Bank**, 2022. Disponivel em: <https://data.worldbank.org/indicator/EG.ELC.ACCS.ZS>. Acesso em: 11 Março 2022.

UNITED NATIONS. **World Population Prospects 2019**. New York: United Nations, 2019.

ZHOU, M. **A Tool for Generating Electrical Single Line Diagram in BIM**. University of South California. California, p. 2-3. 2019.

Apêndice A - Código para Criação e Organização dos Circuitos

```
‘Link para o código completo: https://github.com/mattdamaral/TCC

Private Function CriaCircuitos(blocos As List(Of BlockReference), trans As Transaction)
        Dim circuitos As New List(Of Circuito) ‘Lista de objetos da classe Circuito
        For Each bloco As BlockReference In blocos
            Dim ac As AttributeCollection = bloco.AttributeCollection
            Dim numeroDeAtributosCompativeis As Integer = 0
            'Parâmetros de criação de um circuito
            Dim numero As String = ""
            Dim potencia As Double = 0
            Dim conexão As Integer = 1
            Dim secao As Double = 0
            Dim disjuntor As Integer = 0
            For Each objID As ObjectId In ac ‘Para cada bloco selecionado…
                Dim atributo As AttributeReference = CType(trans.GetObject(objID, OpenMode.ForRead),
AttributeReference)  'Armazena os atributos temporariamente em modo de leitura
                Select Case atributo.Tag ‘Checa se o atributo é compatível com um dos Cases abaixo
                    ‘Caso seja compatível, adiciona 1 ao contador de atributos compatíveis e armazena seu
valor
                    Case "CIRCUITO"
                        numeroDeAtributosCompativeis += 1
                        numero = atributo.TextString
                        Exit Select
                    Case "POTÊNCIA"
                        numeroDeAtributosCompativeis += 1
                        Double.TryParse(atributo.TextString, potencia)
                        Exit Select
                    Case "CONEXÃO"
                        numeroDeAtributosCompativeis += 1
                        Integer.TryParse(atributo.TextString, conexão)
                        Exit Select
                    Case "SEÇÃO"
                        numeroDeAtributosCompativeis += 1
                        Double.TryParse(atributo.TextString, secao)
                        Exit Select
                    Case "DISJUNTOR"
                        numeroDeAtributosCompativeis += 1
                        Integer.TryParse(atributo.TextString, disjuntor)
                        Exit Select
                End Select
            Next
            ‘Caso o bloco possua 5 atributos compatíveis, cria ou adiciona o circuito
            If numeroDeAtributosCompativeis = 5 Then
                If circuitos.Count > 0 Then 'Checa se já existe circuitos na lista de circuitos
                    Dim circuitoJaExiste = False
                    For i = 0 To (circuitos.Count - 1) 'Checa se o número do bloco é igual a um dos existentes
                        If numero = circuitos(i).GetNumero() Then 'Se já existe circuitos na lista de circuitos, e
o número do bloco é igual a um dos existentes, adiciona/modifica os parâmetros
                            circuitoJaExiste = True
                            circuitos(i).AdicionaAPotenciaTotal(potencia)
                            If circuitos(i).GetSecao < secao Then
                                circuitos(i).SetSecao(secao)
                            End If
                            If circuitos(i).GetDisjuntor < disjuntor Then
                                circuitos(i).SetDisjuntor(disjuntor)
                            End If
                            GoTo goto_01
```

```
                End If
            Next
            If circuitoJaExiste = False Then 'Se já existe circuitos na lista de circuitos, porém o
número do bloco difere dos existentes, cria um novo
                circuitos.Add(New Circuito(numero, conexão, potencia, secao, disjuntor))
            End If
        Else 'Se a lista de circuitos está vazia, cria um novo
            circuitos.Add(New Circuito(numero, conexão, potencia, secao, disjuntor))
        End If
    End If
goto_01:
End Function
```

Apêndice B - Código para Criação da Tabela do Quadro de Cargas

```
‘Link para o código completo: https://github.com/mattdamaral/TCC

Private Function MontaQC(posicao As Point3d)
    Dim tabela As Table = New Table() ‘Nova tabela do tipo Table (classe do AutoCAD)
    With tabela
        Dim headerRows As Integer = 2 'Quantidade de linhas para o cabeçalho
        Dim totalRows As Integer = 1 'Quantidade de linhas para o total
        Dim rows As Integer = circuitos.Count + headerRows + totalRows 'Quantidade de linhas totais (cabeçalho + conteúdo + total)
        Dim columnTextList() As String = {"Circuito", "Descrição", "Esquema", "Tensão (V)", "Potência (W)", "Fases", "R (W)", "S (W)", "T (W)", "Seção (mm²)", "Disjuntor", "DDR (30 mA)"} 'Conteúdo das colunas / segunda linha do cabeçalho
        Dim columns As Integer = UBound(columnTextList) - LBound(columnTextList) + 1 'Quantidade de colunas totais (baseado no tamanho do array do conteúdo das colunas)
        ‘Formatação da tabela (quantidade total de linhas e colunas, altura e largura das linhas e colunas, posição, etc.
        .SetSize(rows, columns)
        .SetRowHeight(16)
        .SetColumnWidth(100)
        .Position = posicao
        ‘Formatação do texto e conteúdo da primeira linha (título da tabela)
        .SetTextHeight(0, 0, 8)
        .SetAlignment(0, 0, CellAlignment.MiddleCenter)
        .SetTextString(0, 0, "Quadro de Cargas")
        ‘Formatação do texto e conteúdo da segunda linha (títulos das colunas)
        For indexColumn = 0 To (columns - 1)
            .SetTextHeight(1, indexColumn, 8)
            .SetAlignment(1, indexColumn, CellAlignment.MiddleCenter)
            .SetTextString(1, indexColumn, columnTextList(indexColumn))
        Next
        ‘Formatação do texto e conteúdo do corpo da tabela (informações dos circuitos)
        For indexRow = headerRows To (rows - totalRows - 1)
            For indexColumn = 0 To (columns - 1)
                .SetTextHeight(indexRow, indexColumn, 8)
                .SetAlignment(indexRow, indexColumn, CellAlignment.MiddleCenter)
                Dim circRow = indexRow - headerRows
                Select Case indexColumn
                    Case 0
                        .SetTextString(indexRow, indexColumn, circuitos(circRow).GetNumero().ToString())
                    Case 1
                        .SetTextString(indexRow, indexColumn, "??")
                        .SetColumnWidth(indexColumn, 300)
                    ‘Seguem Cases 2 até 10 (omitido no apêndice por redundância e redução de páginas)
                    ‘…
                    Case 11
                        .SetTextString(indexRow, indexColumn, "-")
                End Select
            Next
        Next
        ‘Formatação do texto e conteúdo da última linha da tabela (informações da alimentação do quadro de distribuição)
        For indexColumn = 0 To (columns - 1)
            .SetTextHeight(rows - totalRows, indexColumn, 8)
            .SetAlignment(rows - totalRows, indexColumn, CellAlignment.MiddleCenter)
        Next
        .SetTextString(rows - totalRows, 0, "Total")
        Dim range As CellRange = CellRange.Create(tabela, rows - totalRows, 0, rows - totalRows, 1)
```

```
            .MergeCells(range)
            .SetTextString(rows - totalRows, 2, esquema.ToString)
            .SetTextString(rows - totalRows, 3, tensao.ToString)
            .SetTextString(rows - totalRows, 4, potencia_total.ToString)
            .SetTextString(rows - totalRows, 5, fases.ToString)
            .SetTextString(rows - totalRows, 6, potencia_r.ToString)
            .SetTextString(rows - totalRows, 7, potencia_s.ToString)
            .SetTextString(rows - totalRows, 8, potencia_t.ToString)
            .SetTextString(rows - totalRows, 9, secao.ToString)
            .SetTextString(rows - totalRows, 10, disjuntor.ToString)
            .SetTextString(rows - totalRows, 11, "-")
            tabela.GenerateLayout()
        End With
        Return tabela ‘Retorna a tabela
End Function
```

Apêndice C - Código para Criação do Diagrama Unifilar

```
‘Link para o código completo: https://github.com/mattdamaral/TCC

Public Sub DesenhaDU(posicao As Point3d)
    'Caso o número de circuitos seja pequeno, define uma altura mínima para inserção do barramento
    Dim posicaoInferior As Double
    If circuitos.Count < 5 Then
        posicaoInferior = 5
    Else
        posicaoInferior = circuitos.Count
    End If
    'Muda a layer atual para a layer do diagrama unifilar
    Call Canvas.TrocaLayer("MD - Diagrama Unifilar")
    'Desenha o barramento que conecta os circuitos
    DesenhaLinha(New Point3d(posicao.X, posicao.Y + 20, posicao.Z), New Point3d(posicao.X, (posicao.Y - (circuitos.Count - 1) * 60) - 20, posicao.Z), 6)
    'Desenha os circuitos
    For i = 0 To (circuitos.Count - 1)
        circuitos(i).Desenha(New Point3d(posicao.X, posicao.Y + (i * (-60)), posicao.Z))
    Next
    'Desenha os textos do nome do QD e da potência total, respectivamente
    DesenhaTexto(nome, New Point3d((posicao.X - 200), (posicao.Y + 65), 0), 7, 15) 'Texto da Descrição
    DesenhaTexto("(" & potencia_total.ToString & " W)", New Point3d((posicao.X - 195), (posicao.Y + 45), 0), 7, 10) 'Texto da Potência Total
    'Desenha o retângulo que envolve o diagrama unifilar
    Dim frame As New List(Of Point2d)
    frame.Add(New Point2d(posicao.X - 200, posicao.Y + 60))
    frame.Add(New Point2d(posicao.X + 125, posicao.Y + 60))
    frame.Add(New Point2d(posicao.X + 125, posicao.Y - (60 * (posicaoInferior - 1) + 200)))
    frame.Add(New Point2d(posicao.X - 200, posicao.Y - (60 * (posicaoInferior - 1) + 200)))
    frame.Add(New Point2d(posicao.X - 200, posicao.Y + 60))
    DesenhaPolyline(frame, 0, 0, 0, True, "Continuous")
    'Desenha o detalhe do Terra
    DesenhaTerra(New Point3d(posicao.X + 75, posicao.Y - (60 * (posicaoInferior - 1) + 200), 0))
    'Desenha o barramento
    barramento.DesenhaBarramento(New Point3d(posicao.X, posicao.Y - (60 * (posicaoInferior - 1)) - 70, posicao.Z))
    'Desenha a linha que conecta o barramento dos circuitos ao ramal de entrada (coluna onde ficariam os DRs)
    DesenhaLinha(New Point3d(posicao.X, (posicao.Y + (posicao.Y - (circuitos.Count - 1) * 60)) / 2, posicao.Z), New Point3d(posicao.X - 55, (posicao.Y + (posicao.Y - (circuitos.Count - 1) * 60)) / 2, posicao.Z), 7)
    'Desenha o ramal de entrada
    DesenhaRamalEntrada(New Point3d(posicao.X - 55, (posicao.Y + (posicao.Y - (circuitos.Count - 1) * 60)) / 2, posicao.Z))
End Sub
```

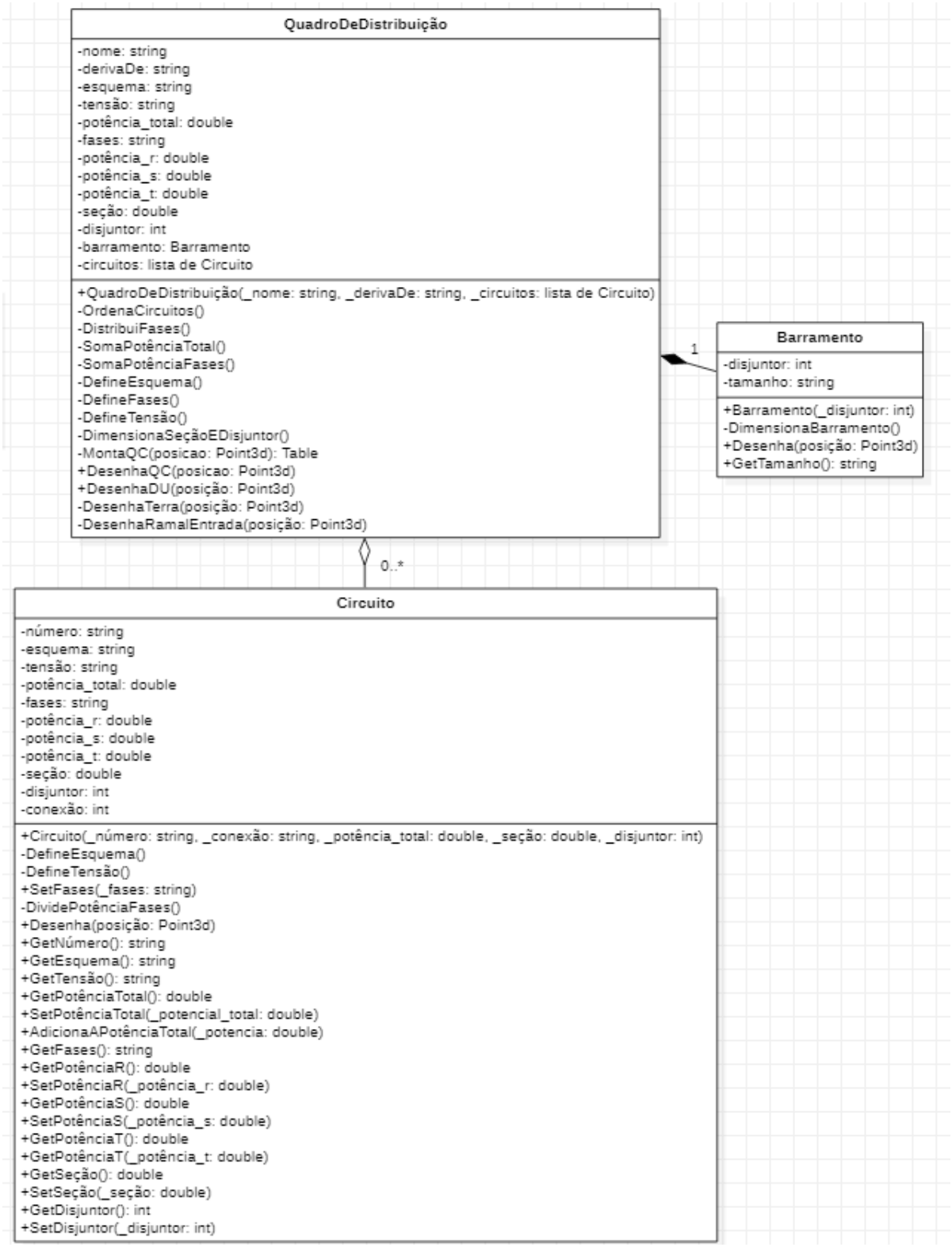
Apêndice D - Código para Sugestão da Quantidade de Luminárias e Tomadas

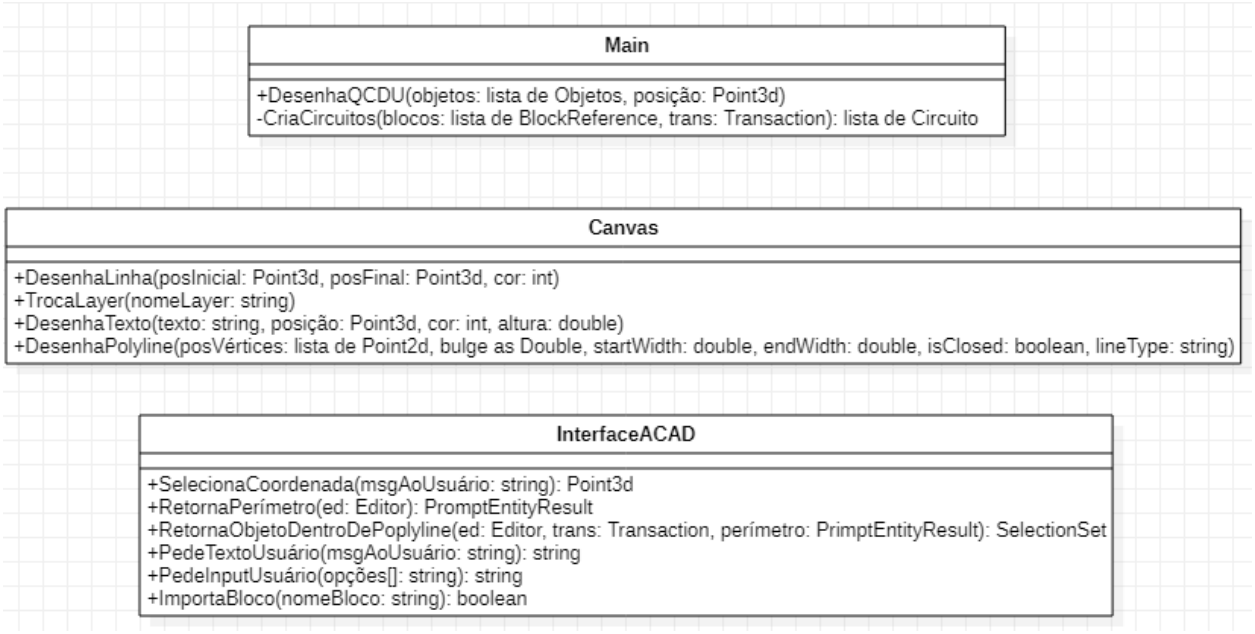
```
‘Link para o código completo: https://github.com/mattdamaral/TCC

Private Function SugereLuminarias(area As Double) 'Função que sugere a potência mínima d
luminárias
    Dim potencia As Integer
    If area <= 6 Then 'Condições baseadas na seção 9.5.2.1 da NBR 5410
        potencia = 100
    Else
        potencia = 100 + Math.Floor(((area - 6) / 4)) * 60
    End If
    Return ("Iluminação: " + potencia.ToString + " VA") 'Retorna um texto com a potência mínima
para iluminação
End Function

Private Function SugereTomadas(perimetro As Double) 'Função que sugere a quantidade e
potência mínima para tomadas
    Dim qtdTomadas
    If PerguntaSeAmbienteSeco() = True Then 'Condições baseadas na seção 9.5.2.2 da NBR 5410
        qtdTomadas = Math.Ceiling(perimetro / 5)
        Return ("Tomadas: " + qtdTomadas.ToString + "x100 VA")
    Else
        If PerguntaSeCozinhaOuAnalogo() = True Then
            Dim obs As String = vbNewLine + "OBS: Mínimo 2 tomadas sobre a pia."
            If perimetro <= 7 Then
                qtdTomadas = 2
                Return ("Tomadas: " + qtdTomadas.ToString + "x600 VA") + obs
            Else
                qtdTomadas = Math.Ceiling(perimetro / 3.5)
                If qtdTomadas <= 3 Then
                    Return ("Tomadas: " + qtdTomadas.ToString + "x600 VA") + obs
                ElseIf qtdTomadas <= 6 Then
                    Return ("Tomadas: 3x600 VA + " + (qtdTomadas - 3).ToString + "x100 VA") + obs
                Else
                    Return ("Tomadas: 2x600 VA + " + (qtdTomadas - 2).ToString + "x100 VA") + obs
                End If
            End If
        Else
            qtdTomadas = 1
            Return ("Tomadas: " + qtdTomadas.ToString + "x600 VA")
        End If
    End If
End Function
```

Apêndice E - Diagrama UML da Versão Inicial do Código Atualizada




Main
+DesenhaQCDU(objetos: lista de Objetos, posição: Point3d)
-CriaCircuitos(blocos: lista de BlockReference, trans: Transaction): lista de Circuito
Canvas
+DesenhaLinha(posInicial: Point3d, posFinal: Point3d, cor: int)
+TrocaLayer(nomeLayer: string)
+DesenhaTexto(texto: string, posição: Point3d, cor: int, altura: double)
+DesenhaPolyline(posVértices: lista de Point2d, bulge as Double, startWidth: double, endWidth: double, isClosed: boolean, lineType: string)
InterfaceACAD
+SelecionaCoordenada(msgAoUsuário: string): Point3d
+RetornaPerímetro(ed: Editor): PromptEntityResult
+RetornaObjetoDentroDePoplyline(ed: Editor, trans: Transaction, perímetro: PrimptEntityResult): SelectionSet
+PedeTextoUsuário(msgAoUsuário: string): string
+PedeInputUsuário(opções[]: string): string
+ImportaBloco(nomeBloco: string): boolean