Universidade do Minho
Escola de Engenharia

Bruno Souza Muniz

Plataforma BIM para execução de verificações automáticas no âmbito do licenciamento municipal

dezembro de 2023

Plataforma BIM para execução de verificações automáticas no âmbito do licenciamento municipal

Bruno Souza Muniz

UMinho | 2023

Universidade do Minho
Escola de Engenharia

Bruno Souza Muniz

Plataforma BIM para execução de verificações automáticas no âmbito do licenciamento municipal

Dissertação de Mestrado
Mestrado em Construção e Reabilitação Sustentáveis

Trabalho efetuado sob a orientação do(a)
Professor Miguel Ângelo Dias Azenha
E coorientação do
Doutor José Luís Duarte Granja

dezembro de 2023

AGRADECIMENTOS

Agradeço primeiro aos meus pais, Cláudio e Selene, que apesar das dificuldades inerentes à responsabilidade de instruir e guiar alguém pelo mundo, sempre tiveram coragem de encarar suas inseguranças para me dar a chance de desenvolver a força para lidar com meus próprios desafios. Também agradeço ao meu irmão, André, que na partilha dos seus entusiasmos me impede de sentir só.

Ao meu amigo André, por ter redefinido o meu conceito de amizade. Assim como à Rita, que tem me ajudado a redefinir tantos outros conceitos.

Ao Hector e todos os demais amigos e familiares que a vida me trouxe, que independentemente da distância partilham comigo um pouco das ideias e sentimentos que há muito ando a despejar por aí.

Ao Professor Miguel Azenha e ao Doutor José Granja pela confiança e partilha nestes últimos meses que têm, sem dúvida, sido os mais desafiantes e esclarecedores que me lembro.

Finalmente a todas as demais pessoas que foram parte essencial de todas as coisas boas da minha caminhada até aqui.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Plataforma BIM para execução de verificações automáticas no âmbito do licenciamento municipal

RESUMO

O licenciamento para a construção é o instrumento legal que atesta que os edifícios que serão construídos atendem a requisitos mínimos de desempenho em fatores como segurança, salubridade e sustentabilidade. Os processos para emissão de licenças, por outro lado, são caracterizados pela complexidade e heterogeneidade, além de atrasos generalizados nos prazos. Muito desta realidade se deve à fragmentação do quadro regulamentar, além da baixa utilização de recursos tecnológicos nas análises destes processos, que ainda se apoiam largamente na análise humana.

As tecnologias BIM permitem desenvolver modelos de edificações que podem ser processados através de ferramentas computacionais e, consequentemente, realizar análises complexas de edificações em tempos inferiores aos utilizados nas análises humanas. Neste contexto, existem algumas frentes de investigação a nível nacional e internacional que visam usufruir dos benefícios do BIM para otimizar os processos de licenciamento, sobretudo através da substituição da análise manual de projetos 2D pela análise automatizada de modelos de informação.

A tentativa de automatizar o processo de licenciamento implica em diversos desafios, principalmente devido à necessidade de se desenvolver regulamentos interpretáveis por máquina, o que é dificultado pelo extenso corpo regulamentar utilizado nos processos de licenciamento e a falta de pessoal qualificado em programação. Desta maneira, os métodos que proporcionam maneiras de se criar regulamentos interpretáveis por máquina a partir de linguagem de programação visual têm ganhado relevância por permitir a criação destes regulamentos pelos técnicos das câmaras, geralmente pouco qualificados em programação.

Esta dissertação visa apresentar o desenvolvimento de uma plataforma protótipo, desenvolvida com ferramentas de código aberto, para a criação de regulamentos interpretáveis por máquina através de linguagem visual baseada em blocos e para a execução de verificações de conformidade a partir de modelos IFC.

Palavras chave: BIM, Verificação de conformidade, Licenciamento Digital, IFC, IfcOpenShell

BIM platform to execute automated compliance checks in building permit processes

ABSTRACT

Building permits are legal instruments that certify that the buildings to be constructed meet minimum performance requirements in factors such as safety, health, and sustainability. The processes for issuing permits, on the other hand, are characterized by their complexity and heterogeneity, as well as widespread delays. Much of this situation is caused by the fragmentation of the regulatory framework, as well as the low use of technological resources in the analysis of these processes, which still rely largely on human analysis.

BIM technologies enable the development of models that can be processed using computational tools and, as a result, complex building analysis to be carried out in less time than human analysis. In this context, there are a number of research initiatives at national and international level that aim to take advantage of the benefits of BIM to optimize building permit processes, especially by replacing the manual analysis of 2D projects with the automated analysis of information models.

The attempt to automate the building permit process involves a number of challenges, mainly due to the need to develop machine-interpretable regulations, which is made difficult by the extensive regulatory body used in building permit processes and the lack of staff qualified in programming. In this way, methods that provide ways to create machine-interpretable regulations using visual programming languages have gained relevance, as they allow the creation of these regulations by the municipality's technicians, who are usually not qualified in programming.

This dissertation aims to present the development of a prototype platform, developed with open source tools, for the creation of machine-interpretable regulations through block-based visual language and for the execution of compliance checks based on IFC models.

Key-words: BIM, Compliance Check, Digital Building Permit, IFC, IfcOpenShell

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

Lista de Abreviaturas, Siglas e Acrónimos

AEC	Arquitectura, Engenharia e Construção
AISC	American Institute of Steel Construction
API	Application Programming Interface
BCF	BIM Collaboration Format
BERA	Building Environment Rule and Analysis Language
BIM	Building Information Modeling
BIMRL	BIM Rule Language
B-REP	Boundary Representation
bsDD	buildingSMART Data Dictionary
CDE	Ambiente Comum de Dados
CHEK	Change Toolkit for Digital Building Permit
CSG	Constructive Solid Geometry
DRF	Django REST Framework
DSL	Domain Specific Languages
IDM	Information Delivery Manual
IFC	Industry Foundation Classes
INE	Instituto Nacional de Estatística
LPO	Lógica de Primeira Ordem
MVC	Model-View-Controller
MVD	Model View Definition
ORM	Object Relational Mapping
PDM	Plano Diretor Municipal
PMOT	Planos Municipais de Ordenamento do Território
RASE	Requirement, Applicabilities, Selection and Exceptions
RGEU	Regulamento Geral das Edificações Urbanas
RJUE	Regime Jurídico da Urbanização e Edificação
RPDML	Regulamento do Plano Diretor Muncipal de Lisboa
SASE	Standards Analysis, Synthesis, and Expression
STEP	Standard for the Exchange of Product model data
UML	Unified Modeling Language
VBQL	Visual BIM Query Language
VCCL	Visual Code Checking Language
XML	Extensible Markup Language

1 Introdução

1.1 Enquadramento geral

O licenciamento para a construção é o instrumento legal que atesta que os edifícios que serão construídos atendem a requisitos mínimos de desempenho em diversos âmbitos, a exemplo da segurança, salubridade, qualidade e funcionalidade das instalações, desempenhos térmico, acústico e energético, entre outros fatores (Pedro et al., 2011). Além disto, muitas diretivas da Comissão Europeia acabam por ter implementações nacionais na forma dos regulamentos para emissão de licenças de construção, de forma que os processos de licenciamento sofrem alterações com alguma frequência para alinhar-se a prioridades europeias como a sustentabilidade (Green Deal), segurança e bem-estar (Build4people) também em linha com a Renovation Wave.

Apesar do recente desenvolvimento de tecnologias, que têm promovido a digitalização e automatização de processos em diversos setores, os fluxos para a emissão de licenças de construção ainda estão apoiados em processos manuais de análise com recurso a projetos e documentos em 2D, o que contribui para que estes processos sejam demorados, confusos, ineficientes e estejam sujeitos a diversos erros provenientes da análise humana (Malsane et al., 2015). Entretanto, esse atraso na adoção de novas tecnologias possui correlação com desafios notórios no setor da construção, que é caracterizado, de uma forma geral, pela demora na implementação de recursos tecnológicos e pela mão de obra pouco especializada e pouco capaz de lidar com os recentes avanços.

A adoção da metodologia BIM, através da digitalização dos processos, está a proporcionar uma profunda revolução na indústria da construção. O potencial criado pela partilha de informações em um ambiente colaborativo com suporte a novas tecnologias, que estão em constante evolução, tem gerado um grande estímulo para a adoção do BIM a todos os níveis e por todos os intervenientes da indústria (Aguiar et al., 2020). Isto pode ser sentido na pressão normativa e no suporte regulamentar que são alguns dos fatores preponderantes para a difusão do BIM mundialmente (Shehzad et al., 2020), já que a otimização advinda da adoção dessa nova metodologia tem grande potencial de gerar economias no âmbito público. Também é possível identificar uma grande quantidade de frentes de investigação, com abordagem de tópicos como verificação regulamentar, ontologias e computação móvel (Zhao, 2017).

Entre as tecnologias BIM em desenvolvimento, há também um grande esforço para a promoção de soluções abertas para desenvolvimento de fluxos de trabalho com recurso a BIM, conhecido como

openBIM , liderado maioritariamente pela entidade sem fins lucrativos buildingSMART, que é responsável por diversas iniciativas para promover a interoperabilidade no desenvolvimento de projetos através do BIM, a exemplo do formato aberto IFC, para partilha de informações de edificações durante o ciclo de vida do projeto. Também é importante destacar a evolução de ferramentas abertas para criação de aplicações, como o IfcOpenShell, que tem facilitado e tornado acessível o desenvolvimento de soluções para otimizar diversos processos na indústria.

Em linha com estes desenvolvimentos, há um esforço significativo para otimizar os processos de licenciamento através da aplicação de várias dessas tecnologias, principalmente através da tentativa de automatização das análises de projetos, o que é possível através da implementação de verificações de conformidade a partir de modelos BIM. Além disso, o termo mais amplo "licenciamento municipal" é utilizado neste trabalho, apesar do foco ser no licenciamento urbanístico. Finalmente, este trabalho pretende colaborar com este esforço a partir da apresentação do desenvolvimento de uma plataforma protótipo, baseada em ferramentas de código aberto, para a criação de regulamentos digitais para a execução de verificações de conformidade a partir de ficheiros IFC.

1.2 Objetivos e metodologia

1.2.1 Objetivo geral

O objetivo geral é desenvolver um protótipo, com ferramentas de código aberto, que permita a criação de regulamentos interpretáveis por máquina em linguagem visual e execução de verificações de conformidade a partir de ficheiros IFC.

1.2.2 Objetivos específicos

O trabalho apresentado pretende desenvolver os seguintes objetivos específicos :

- Verificar as publicações mais recentes e os trabalhos mais significativos no âmbito do licenciamento municipal, do uso de tecnologias BIM e suas aplicações para o licenciamento municipal e dos métodos utilizados para criar regulamentos interpretáveis por máquina;
- Promover a análise de algumas cláusulas de regulamentos portugueses e implementá-las em um formato interpretável por máquina a partir do desenvolvimento de um modelo de entidades do licenciamento;

- Desenvolver uma plataforma web, com o emprego de tecnologias de formato aberto, a fim de executar verificações automáticas de conformidade e permitir a criação de regulamentos interpretáveis por máquina a partir do uso de uma linguagem de programação visual;
- Validar os regulamentos criados na plataforma desenvolvida a partir da análise de um modelo IFC;
- Verificar as lacunas e dificuldades existentes na estrutura como um todo e nos seus componentes, a fim de determinar quais os pontos chave e quais os potenciais de desenvolvimento futuro.

1.2.3 Metodologia adotada

A execução deste trabalho ocorreu através do desenvolvimento de duas tarefas de alto nível:

1. Desenvolvimento de modelo de entidades do licenciamento e ferramentas para processamento geométrico;
2. Desenvolvimento de um protótipo de plataforma para criação de regulamentos digitais e execução de verificações de conformidade;

A primeira tarefa é essencial para se estabelecer todas as classes e métodos que serão utilizados para a execução das verificações de conformidade e foi desenvolvida a partir dos seguintes passos:

a) Análise de cláusulas regulamentares;
b) Obtenção de modelo conceptual de entidades e propriedades do licenciamento;
c) Mapeamento entre modelo conceptual e IFC e implementação do modelo de entidades do licenciamento;
d) Criação de ferramentas baseadas em classes de alto nível para processamento geométrico;
e) Validação do modelo de entidades a partir de verificação de modelo IFC;

A segunda tarefa envolve a criação do protótipo de plataforma que permite criar regulamentos digitais a partir de uma linguagem de programação visual capaz de gerar códigos que implementam as classes desenvolvidas na tarefa 1, além de executar as verificações e visualizar relatórios. Desta maneira, a tarefa foi implementada a partir dos seguintes passos:

a) Definição de casos de uso;
b) Desenvolvimento dos serviços backend e APIs;
c) Desenvolvimento dos serviços frontend;

d) Criação da linguagem de programação baseada em blocos e implementação do editor de regras de verificação;
e) Implementação do visualizador;
f) Validação da plataforma através da verificação de modelo IFC de teste.

1.3 Estrutura da dissertação

A presente dissertação possui 5 capítulos principais, sendo este o primeiro, que apresenta o enquadramento do tema a ser abordado, sua motivação, objetivos e definição dos principais aspetos da metodologia adotada.

O capítulo 2 é o estado da arte, que contém a revisão dos trabalhos mais relevantes acerca do licenciamento municipal e das tecnologias BIM que podem ser utilizadas para promover sua otimização, principalmente a partir da digitalização do processo de verificação de conformidade. O capítulo também apresenta as iniciativas mais relevantes de implementação e investigação, assim como os métodos utilizados para promover a digitalização dos regulamentos do licenciamento a partir da sua interpretação e implementação em um formato interpretável por máquina.

O capítulo 3 apresenta o desenvolvimento da primeira tarefa de alto nível, com início no desenvolvimento do modelo de entidades do licenciamento. Em sequência são apresentados os pontos mais importantes da implementação das ferramentas nível para processamento geométrico e finalmente é apresentada a validação do modelo de entidades.

O quarto capítulo apresenta o desenvolvimento da segunda tarefa de alto nível, com foco inicial nos casos de uso estabelecidos e na arquitetura global desenvolvida para a plataforma. Seguidamente, os componentes da arquitetura são abordados. O fim do capítulo apresenta o processo de verificação estabelecido e a validação da plataforma.

O quinto capítulo apresenta as conclusões do trabalho, com abordagem dos principais desafios e a discussão dos potenciais desenvolvimentos futuros.

2 Digitalização do processo de licenciamento

2.1 Licenciamento municipal

2.1.1 Enquadramento

O setor da construção possui caráter estratégico no desenvolvimento de cada país devido ao impacte das edificações nas condições de saúde e conforto da população, à sua relevância económica e à sua relação com o desenvolvimento sustentável (Pedro et al., 2009), o que evidencia a necessidade de planear o desenvolvimento e desempenho das edificações num espaço urbano. Assim, as operações urbanísticas requerem aprovação do Estado a fim de garantir seu desempenho em função de critérios de segurança, saúde, eficiência energética, sustentabilidade e acessibilidade (Pedro et al., 2011) através de um processo pautado na análise de informações sobre a edificação em função de parâmetros estabelecidos em regulamentos de construção.

2.1.2 Processo

Os procedimentos aplicados às operações urbanísticas são variados, com a isenção da necessidade de licenças em alguns casos, a exemplo das alterações interiores que não impliquem modificações de estruturas resistentes, cérceas, fachadas ou formas dos telhados, assim como obras de edificação ou demolição que possuam pouca relevância urbanística (Decreto-Lei n.º 555/99, de 16 de dezembro | DR, 1999). Ainda assim, muitos casos necessitam de licenças para o início da obra, de forma que as principais operações urbanísticas são:

- Obras de construção;
- Obras de reconstrução;
- Obras de ampliação;
- Obras de alteração;

Os processos de licenciamento são operacionalizados nas Câmaras Municipais e sua forma legal é determinada pelo Regime Jurídico da Urbanização e Edificação (RJUE). Segundo Oliveira et al. (2016), o procedimento de licenciamento de operações urbanísticas é composto por 5 fases: (i) iniciativa, com a apresentação do requerimento nas plataformas da Câmara; (ii) saneamento e apreciação liminar, onde são avaliadas possíveis inconsistências e irregularidades nos pedidos de licenciamento que possam interferir no seguimento do processo; (iii) instrução, onde ocorre a consulta às entidades

externas (p.ex. Direção Geral do Património Cultural e Segurança Social) e apreciação dos projetos; (iv) constitutiva, onde ocorre a deliberação final de deferimento do pedido de licenciamento e (v) fase integrativa da eficácia, onde ocorre a emissão do alvará (Figura 2).

A apreciação dos projetos, que ocorre na instrução, é um processo usualmente baseado na análise de ficheiros DWF ou até mesmo projetos de papel, de forma a verificar a conformidade dos aspetos técnicos da operação urbanística em face a uma série de requisitos regulamentares. Esse traço pouco tecnológico deste processo implica em pouca eficiência, morosidade e tendência a erros (Noardo et al., 2021).

Figura 1: Procedimento de licenciamento de operações urbanísticas. Adaptado de Oliveira et al. (2016).

O Decreto-Lei nº 136/2014, que alterou o texto do artigo 8º do RJUE, determinou que a tramitação do processo de licenciamento ocorra através de uma plataforma eletrónica, nomeadamente os processos de entrega de requerimentos e comunicações, consulta pelos interessados do estado nos procedimentos, submissão dos procedimentos à consulta por entidades externas ao município, obtenção automática de comprovativos de requerimentos, comunicações e deferimentos. Contudo, o mesmo decreto determina que na inexistência ou indisponibilidade de recursos informáticos, os procedimentos podem recorrer a outros recursos digitais ou com recurso a papel (Decreto-Lei n.º 136/2014, de 9 de setembro do Ministério do Ambiente, Ordenamento do Território e Energia, 2014). Esta permissão ocorre em função da incapacidade de algumas autarquias de digitalizar completamente o processo, seja pela escassez de recursos materiais e humanos, seja pelas dificuldades advindas de exigir a adoção de ferramentas tecnológicas específicas.

A recente iniciativa de simplificação dos processos administrativos materializa na Proposta de Lei nº. 77/XV, entre diversas medidas que versam sobre os processos de licenciamento, algumas que tangem o âmbito da digitalização, com uma alteração legislativa abrangente, a incluir o RJUE e o Regulamento Geral das edificações Urbanas (RGEU). Neste âmbito, há destaque para a criação de uma "Plataforma Eletrónica dos Procedimentos Urbanísticos" que permita, entre algumas funcionalidades:

- Apresentar pedidos online;

– Submeter pedidos em formato BIM, com automatização da verificação do cumprimento dos planos aplicáveis.

Além disso, a proposta prevê que a plataforma deverá ser obrigatoriamente utilizada pelos municípios a partir de 5/1/26 (Proposta de Lei 77/XV/1, 2023). Estas medidas implicam num grande desafio tendo em perspetiva o estado de desenvolvimento dos processos digitais de licenciamento, não somente a nível nacional, mas também mundial, já que os diversos casos de implementação ainda possuem diversas limitações (Shahi et al., 2019).

Em relação aos prazos para a deliberação final, o RJUE estabelece, salvo algumas exceções, 45 dias para obras de construção, contados a partir do momento da obtenção de toda documentação necessária para instrução do processo ou da emissão dos pareceres de entidades externas (Decreto-Lei n.º 555/99, de 16 de dezembro | DR, 1999). Já o prazo estipulado para a análise dos projetos é de 30 dias, o que mostra a relevância deste processo dentro do fluxo para emissão de licenciamentos. Ainda assim, ao analisar os tempos realmente aplicados para a obtenção de uma licença, há uma grande discrepância entre o que é estabelecido legalmente e o que é praticado. Um estudo realizado pela Ordem dos Arquitetos da Secção Regional de Lisboa e Vale do Tejo indica a ocorrência de atrasos de até dois anos nos processos de licenciamento, além de caracterizar estes atrasos como um problema transversal a todas as autarquias do país. O estudo também indica a falta de interpretação uniforme, pelas autarquias, das regras em vigor[1].

2.1.3 Corpo regulamentar

O processo de licenciamento apoia-se numa série de requisitos existente num vasto quadro regulamentar, que indica parâmetros, em diversos domínios, que não podem ser ultrapassados pela operação urbanística. O quadro regulamentar português foi pouco alterado desde a década de 1990. Maioritariamente, as alterações que ocorreram nestes documentos foram consequência da necessidade de proporcionar um caráter mais prático aos processos de licenciamento, da falta de técnicos nas estruturas municipais e dos longos tempos levados para execução dos processos. Estas alterações promoveram medidas como a redução do controle público e a delegação de responsabilidades para os particulares (Pedro et al., 2011). Além disso, grande parte destas alterações foram resultado da necessidade de implementar Diretivas europeias. Ainda assim, a necessidade do

[1] https://expresso.pt/economia/2021-07-24-Construcao.-90-dos-licenciamentos-fora-de-prazo-bad4af23

controle dos processos de Reabilitação do parque edificado requer a existência de mecanismos para o estabelecimento de requisitos mais adequados a este tipo de edificação, já que as exigências atuais ainda tornam difíceis a conformidade com as exigências em processos de reabilitação (Pedro et al., 2009).

O quadro regulamentar português conta com mais de 45 regulamentos e documentos que atribuem requisitos para as edificações (Branco Pedro et al., 2009). A nível da edificação, o Regulamento Geral das Edificações Urbanas (RGEU) é o principal regulamento nacional e possui uma estrutura muito estável desde que entrou em vigor. O documento possui provisões a estabelecer as diretrizes técnicas em diversos domínios da edificação, sejam estruturais, a exemplo das fundações, paredes, coberturas e escadas, assim como parâmetros relativos às dimensões espaciais, ao conforto e à integração do edifício no espaço urbano, de forma a assegurar condições de segurança, salubridade e estética adequadas à sua utilização e às funções que deve exercer. O RGEU também indica a necessidade de adequação das edificações tanto aos planos municipais, quanto a outras disposições legais que estejam a cargo da administração municipal (Decreto-Lei n.º 38382, de 7 de agosto do Ministério das Obras Públicas - Gabinete do Ministro, 1951).

Ao nível dos municípios, os principais regulamentos que tratam sobre os requisitos das edificações são o Plano Diretor Municipal (PDM) e o Regulamento Municipal de Urbanização e Edificação (RMUE). Os planos diretores municipais são de desenvolvimento obrigatório, à exceção da existência de planos intermunicipais, e têm por objetivo ser instrumento para a aplicação das estratégias de gestão territorial a alto nível, sendo o principal entre os Planos Municipais para Ordenamento do Território (PMOT), e servindo de referência para o desenvolvimento dos demais planos municipais, os Planos de Urbanização e os Planos de Pormenor (Decreto-Lei n.º 80/2015, de 14 de maio do Ministério do Ambiente, Ordenamento do Território e Energia, 2015). O estabelecimento da organização do concelho nos planos diretores ocorre através da classificação e qualificação do solo através de zonas (Figura 2). Já os Regulamentos Municipais de Urbanização e Edificação estabelecem conceitos técnicos no âmbito do urbanismo e da edificação, sendo complementares aos demais planos municipais, mas também discorrem sobre regras técnicas e procedimentais não previstas expressamente no RJUE, a fim de proporcionar maior clareza e transparência à atuação municipal.

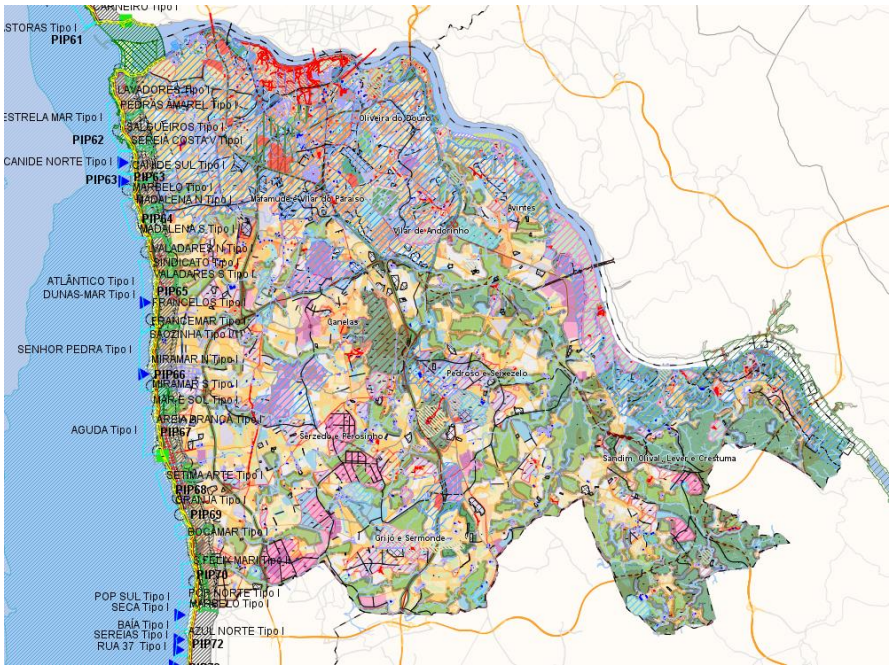

Figura 2: Zonas estabelecidas pelo Plano Diretor Municipal de Vila nova de Gaia. Fonte: Geoportal Gaiurb. Acesso 14/07/2023.

Esta diversidade de documentos torna a estrutura regulamentar portuguesa complexa e fragmentada, o que contribui negativamente para a clareza dos processos de licenciamento (Pedro et al., 2009). Além disso, este quadro dificulta a automatização da verificação de conformidade, já que é um processo que envolve a codificação dos regulamentos num formato interpretável por máquina, o que maioritariamente deve ser executado por equipas especializadas tanto nos regulamentos quanto em sistemas da informação. Assim, a codificação de um quadro regulamentar com as características mencionadas, nos moldes aplicados atualmente, iria requerer equipes especializadas em S.I. para a codificação e manutenção deste processo em cada Câmara, o que tornaria o processo muito oneroso do ponto de vista de recursos humanos.

Além destes regulamentos, é preciso também mencionar a ficha de elementos estatísticos do INE, que tem caráter obrigatório na instrução do processo e possui informações geométricas e de ocupação das edificações a fim de proporcionar dados para análises estatísticas, e os quadros sinópticos que identificam de forma objetiva as principais características espaciais sobre o terreno e o edifício a ser construído (Arias Vanegas, 2022).

2.2 Licenciamento municipal com recurso a BIM

2.2.1 Building Information Modelling e enquadramento normativo

Os projetos no setor da construção são conhecidos pela necessidade de interação de diversos intervenientes durante seu ciclo de vida. Entretanto, a falta de planeamento, lenta adoção de novas tecnologias e baixa especialização que caracterizam os trabalhos no setor implicam em projetos desorganizados, que sofrem com atrasos, erros, custos elevados e problemas de comunicação e partilha de informação (Eichler et al., 2023).

O *Building Information Modelling* (BIM) é uma metodologia que prevê o desenvolvimento de um ambiente colaborativo de partilha da informação durante todo o ciclo de vida de uma edificação. Isto é possível através da criação de modelos de informação que representam uma edificação parcialmente ou integralmente, com nível de informação controlado e adequado para o uso pretendido (C. M. Eastman, 2011). Estes modelos contêm informações geométrica e não geométrica, organizadas numa estrutura de dados, com representações de diversas naturezas, a exemplo de propriedades físicas, atributos e classificações de elementos. Além disso, são criados dentro de uma lógica paramétrica e orientada a objetos, de tal forma que seus elementos constituintes são estabelecidos através de um conjunto de entidades e relações, definidas em diferentes níveis de abstração, vinculadas por regras definidas em parâmetros (Cerovsek, 2011). Uma das consequências mais facilmente observáveis advindas disto é a alteração simultânea de elementos nas diversas vistas numa aplicação BIM.

No ciclo de vida de um projeto com recurso a BIM, os modelos desenvolvidos precisam de ser partilhados e modificados com vista a usos diferentes e em diferentes fases, a exemplo de projetos arquitetónicos, que servem de base para desenvolvimento dos projetos de especialidades e análises energéticas numa primeira fase, mas podem servir futuramente para o desenvolvimento e acompanhamento de planos de manutenção. Desta forma, este ciclo de vida é caracterizado por uma constante de partilha de informações sobre os modelos gerados.

Em relação à normalização, os principais documentos a nível internacional são:

- Série EN ISO 19650 *Building Information Modelling* (BIM): série de normas para a organização e gestão da informação dentro da indústria da construção. Fornece orientações para a criação, distribuição, e utilização de informação digital, incluindo BIM. As normas da série EN ISO 19650 foram concebidas para melhorar a eficiência e eficácia do processo de construção, facilitando o intercâmbio e utilização de informação digital entre as várias partes envolvidas num projeto de construção, incluindo proprietários, projetistas, empreiteiros, e gestores de instalações;

- EN ISO 29481-1:2017 *Building information models – Information delivery manual – Part 1: Methodology and format*: esta norma especifica a informação requerida por todos os processos contidos no ciclo de vida das instalações com o intuito de facilitar a interoperabilidade entre aplicações;
- EN ISO 12006-3: 2022 *Building construction – Organization of information about construction works - Part 3: Framework for object-oriented information*: esta norma especifica um modelo para o desenvolvimento de dicionários no âmbito da construção. O modelo pode ser ampliado, através de uma lógica direcionada a objetos, permitindo o desenvolvimento de ontologias, taxonomias, meronomias, léxicos e thesaurus a partir do seu conteúdo.

Além disso, é importante considerar algumas das iniciativas a nível nacional, como:

- Comissão Técnica de Normalização BIM – CT 197: entidade delegada pelo Instituto Português da Qualidade (IPQ) para desenvolvimento de normalização em BIM como: sistemas de classificação, modelação da informação e de processos ao longo do ciclo de vida dos empreendimentos. Tem o Guia de Contratação BIM como principal trabalho publicado até o momento;
- buildingSMART Portugal: associação sem fins lucrativos portuguesa, integrada por agentes em diversas esferas do setor da construção, que visa promover a interoperabilidade no BIM através do desenvolvimento e suporte de formatos abertos para a troca de informações em todos os processos da indústria da construção durante seu ciclo de vida;
- BIM nas autarquias: documento que tem por intuito sensibilizar as autarquias para o BIM através da sistematização, de forma concisa e acessível, dos diversos tópicos envolvidos na matéria, como a normalização, principais definições, desafios, além do potencial de retorno advindo da sua implementação ao nível das autarquias.

2.2.2 Ferramentas, plataformas e ambientes BIM

Maioritariamente, a metodologia BIM é apoiada pela utilização de soluções tecnológicas para implementação dos seus fluxos e princípios. Isto recai na utilização de diversos tipos de aplicações, com diferentes funcionalidades, específicas ou não para BIM, que integram os fluxos dos projetos em diferentes fases (Figura 3).

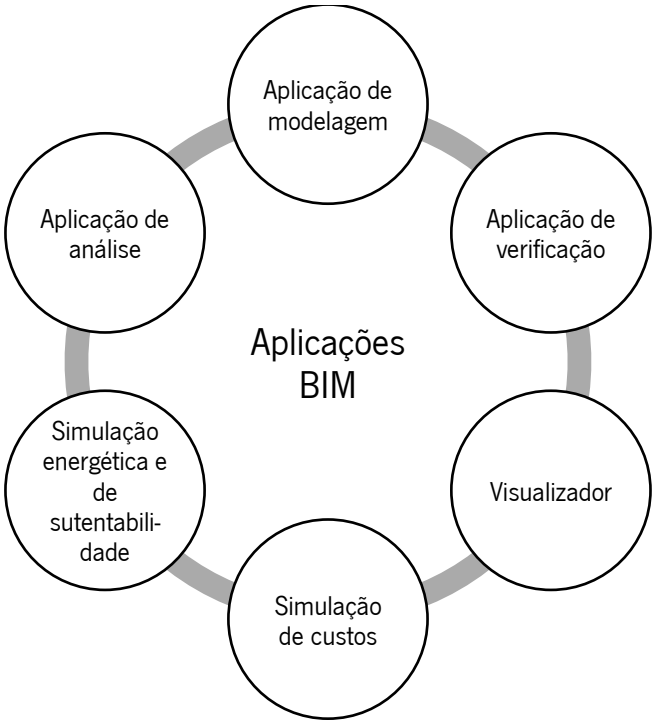

Figura 3: Aplicações BIM de diferentes funcionalidades. Adaptado de (Eichler et al., 2023).

Segundo Eastman (2011), as aplicações BIM podem ser consideradas dentro da seguinte hierarquia:

- Ferramenta BIM: uma aplicação que possui uma finalidade específica. Dentro desta categoria estariam as aplicações que desenvolvem funções específicas, como deteção de colisões, análise energética, visualização e extração de quantidades;
- Plataforma BIM: aplicações que geram produtos para usos múltiplos. Normalmente consistem em aplicações de projeto que desenvolvem modelos de edfificação que podem ser utilizados em ferramentas BIM com diferentes funções. A maior parte das plataformas, porém, também incorporam algumas ferramentas;
- Ambientes BIM: ecossistema integrado de aplicações (ferramentas e plataformas) com a finalidade de gerir um ou mais fluxos da informação dentro de uma organização. O âmbito da informação nos ambientes BIM já ultrapassa os empregados em modelos de edificação, com troca de dados relativos a imagens, vídeos, emails e outros tipos de informação.

Atualmente, as aplicações mais populares para criação de modelos e execução de análises são proprietárias e estes modelos são armazenados em formatos fechados. Este quadro força a utilização de soluções para partilha das informações num projeto em ecossistemas proprietários, o que contribui negativamente com a livre partilha de informações, mas que pode ser contornada através da utilização

de modelos de dados abertos em conjunto com interfaces de programação de aplicações, as APIs (Laakso & Kiviniemi, 2012).

Também é preciso referenciar as implementações de um ambiente comum de dados (*common data environment* - CDE), que prevê a recolha, a gestão e a disseminação de containers de informação em processos BIM (International Organization for Standardization, 2019) já que, apesar de não estar restrito a implementações em forma de aplicações, o conceito do CDE é largamente beneficiado através da sua implementação em aplicações locais ou através de plataformas web, o que facilita a colaboração entre os intervenientes, inclusive com a integração de diversas aplicações utilizadas pelas diversas equipes no ciclo de vida de uma edificação (Eichler et al., 2023).

2.2.3 Interoperabilidade, buildingSmart e openBIM

A capacidade de troca de informações entre aplicações é denominada Interoperabilidade, sendo esta um fator chave para o desenvolvimento de projetos com recurso a BIM (C. M. Eastman, 2011). Esta capacidade tem sido garantida através do desenvolvimento de formatos de troca de informação, baseados em modelos de dados abrangentes, que são adotados pela indústria.

De forma a promover a interoperabilidade na indústria, a *Industry Alliance for Interoperability* (IAI) foi criada dentro da Autodesk em 1994. Dois anos a pós sua criação, a organização tornou-se aberta e sem fins lucrativos, modificando o seu objetivo para a publicação de um modelo de dados voltado para o ciclo de vida de construções, baseado em tecnologias ISO-STEP, denominado *Industry Foundation Classes* (IFC). Esta organização em 2005 foi renomeada, tornando-se buildingSMART. Atualmente, além de desenvolver e gerir o IFC, é responsável pelo protocolo de comunicação *BIM Collaboration Format* (BCF), pelo Manual de Entrega de Informações (IDM), pelo buildingSMART Data Dictionary (bSDD) e outras soluções para a transformação tecnológica e colaboração na indústria AEC num fluxo de trabalhos digital.

Alinhado com os objetivos da buildingSMART o conceito do openBIM visa garantir a colaboração entre os intervenientes num ciclo de vida da edificação através do desenvolvimento de soluções abertas para a execução de tarefas nos processos BIM e para a interoperabilidade. O caráter aberto destas soluções impulsiona a acessibilidade, usabilidade, gestão e sustentabilidade dos dados digitais, além de facilitar a criação de ferramentas BIM, já que o acesso livre à informação contida nos modelos é propício para a análise através de ferramentas de programação, como a linguagem Python e suas bibliotecas.

2.2.4 Industry Foundation Classes – IFC

O IFC é um modelo de dados e um formato que estabelece uma abrangente série de classes, baseada nas necessidades de troca de informação presentes nos projetos do setor da construção, como meio de estabelecer um formato padrão de interoperabilidade e favorecer a colaboração entre os intervenientes no ciclo de vida de um projeto (Laakso & Kiviniemi, 2012).

O modelo de dados do IFC é definido através da linguagem EXPRESS, que possui lógica orientada a objetos, de forma que sua estrutura é composta de entidades representativas tanto de elementos físicos como de relações, propriedades e outros conceitos (C. M. Eastman, 2011). Esses elementos são estruturados a partir de uma arquitetura em diversas camadas, de forma que os elementos instituídos nas camadas de mais baixo nível servem de recurso ou referência para criação de elementos das camadas superiores, através de herança. As camadas da arquitetura do modelo de dados do IFC podem ser observadas na Figura 4, sendo estas:

Camada de recursos: camada de mais baixo nível que contém a definição de todos os recursos que serão utilizados na criação de elementos das camadas superiores, a exemplo da geometria, topologia, materiais, medidas e propriedades. As definições nesta camada não possuem um identificador global único, já que estas definições não devem ser utilizadas independentemente.

Camada central: camada superior à camada de recursos, que contém o modelo do kernel e os modelos de extensão central, com as definições mais gerais de entidades. A partir desta camada, todas as entidades passam a ter um identificador global único, e definições opcionais de proprietário e histórico;

Camada de interoperabilidade: camada superior à camada central, em que as entidades já representam definições mais especializadas de produtos, processos ou recursos. As definições deste nível são tipicamente utilizadas para troca de definições entre diferentes especializações e partilha de informações da construção;

Camada de domínio: camada de mais alto nível e maior abstração. As definições de entidades nesta camada são as mais especializadas e normalmente são utilizadas para troca de informação no âmbito da mesma especialização.

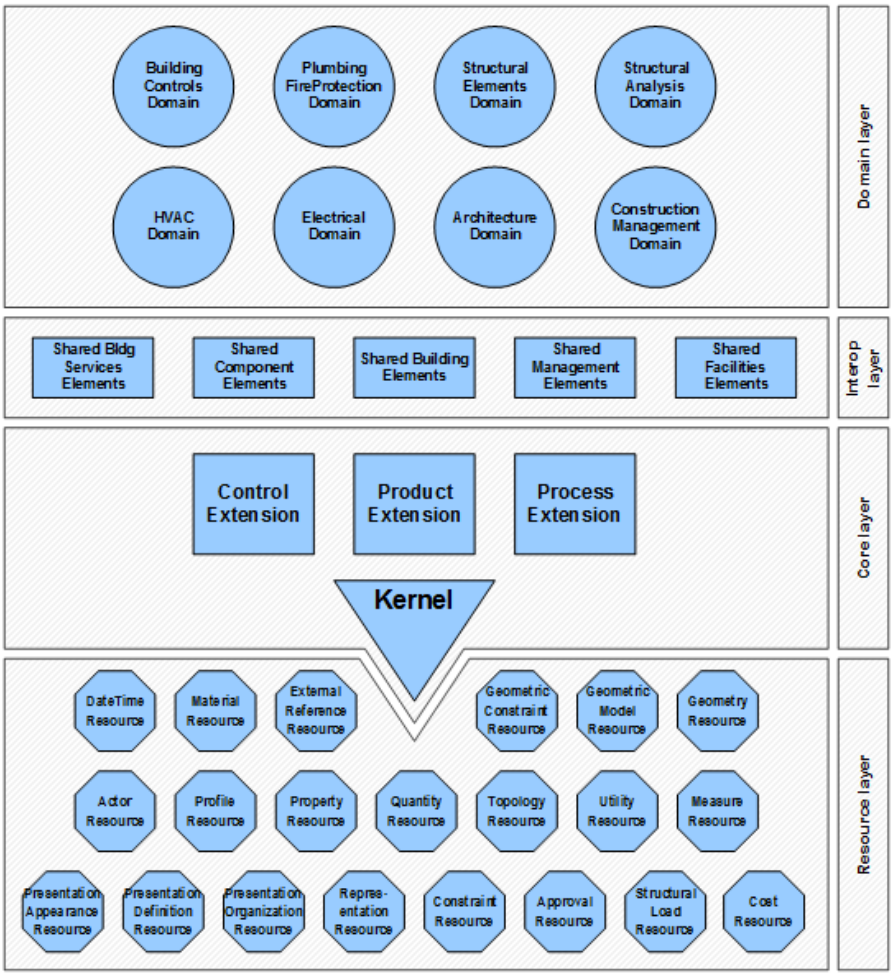

Figura 4: Arquitetura do modelo de dados IFC com representação das camadas conceptuais[2].

A estrutura hierárquica do IFC implica que os objetos mais especializados estejam nas extremidades de uma árvore hierárquica que representa as relações entre entidades. Assim, a definição da estrutura de uma parede, por exemplo, herda propriedades e relações de entidades que serviram como construtores básicos, como é possível observar na Figura 5. A herança do IfcRoot na definição de um elemento impõe a existência das propriedades que definem nome, histórico, identificador global e descrição, já a herança do IfcObject determina a existência de uma relação IsDefinedBy, que associa a entidade a grupos de propriedades (C. M. Eastman, 2011).

A série de propriedades e relações existentes numa determinada entidade proporciona o acesso a todas as informações existentes num elemento e em todos os elementos relacionados, o que viabiliza a consulta de diversas informações de um elemento através das suas relações.

[2] https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/

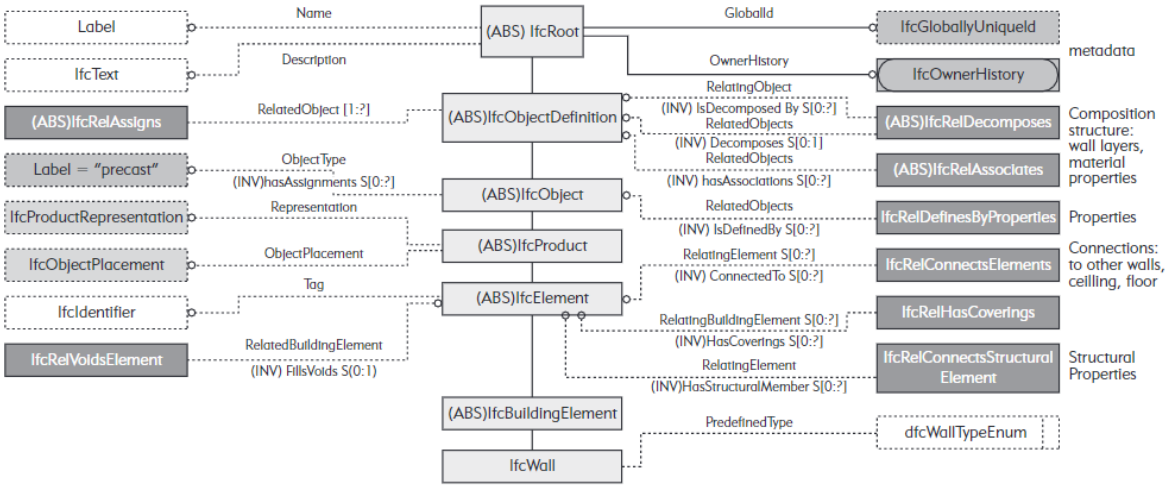

Figura 5: Estrutura do IFC para determinação de um IfcWall (Eastman et al., 2008).

O modelo de dados IFC pode ser implementado em alguns formatos, a exemplo do XML. Entretanto o formato largamente utilizado até o momento é o STEP[3], sendo este o formato mais compacto utilizado que pode ser lido como texto. Neste formato, cada linha possui a representação de uma entidade, que é numerada, além dos valores das suas propriedades e referências a outras entidades no modelo (Figura 6).

```
#32970= IFCWALL('1xbDTrVOf4090BjkM1J0x2',#42,'Parede b\X2\00E1\X0\sica:Retaining -
180mm Concrete:357379',$,'Parede b\X2\00E1\X0\sica:Retaining - 180mm Concrete',#
32941,#32966,'357379',.NOTDEFINED.);
#32973= IFCWALLTYPE('1xbDTrVOf4090BjkM1J0YC',#42,'Parede b\X2\00E1\X0
\sica:Retaining - 180mm Concrete',$,$,(#32976),$,'356941',$,.STANDARD.);
#32974= IFCPROPERTYSINGLEVALUE('ThermalTransmittance',
$,IFCTHERMALTRANSMITTANCEMEASURE(3.61111111111111),$);
#32975= IFCPROPERTYSINGLEVALUE('IsExternal',$,IFCBOOLEAN(.T.),$);
```

Figura 6: Representação de algumas entidades num ficheiro IFC STEP.

A representação geométrica é de extrema importância em modelos BIM, já que através da análise de dados geométricos dos elementos é possível extrair informações de interesse para usos diversos, como a análise de desempenho energético, através da verificação de volumes de ambientes e deteção de colisão entre objetos em processos de compatibilização de projetos (C. M. Eastman, 2011). Essa capacidade de obter informação geométrica de modelos também permite verificar a conformidade das edificações que estão modeladas com muitos requisitos dos regulamentos de licenciamento. A representação de modelos sólidos no IFC (Figura 7) pode ser dividida em três categorias:

[3] https://technical.buildingsmart.org/standards/ifc/ifc-formats/

- **Boundary representation (B-Rep):** nesta representação, os sólidos são definidos através de uma descrição completa da superfície limitante de um sólido, composta por vértices, arestas e faces. Esta representação implica uma maior complexidade da geometria;
- **Operações de varrimento:** as operações de varrimento definem um sólido a partir do percurso de um elemento bidimensional por um caminho;
- **Constructive Solid Geometry (CSG):** a construção de sólidos ocorre através de operações com sólidos básicos. Assim, combinações de operações de união, intersecção e diferença entre sólidos básicos, que podem ser representadas numa estrutura hierarquizada, definem a geometria de sólidos, que são consequentemente mais complexos. A representação em CSG não dispõe de uma representação explícita das superfícies limitantes do sólido (Tsuzuki et al., 2007).

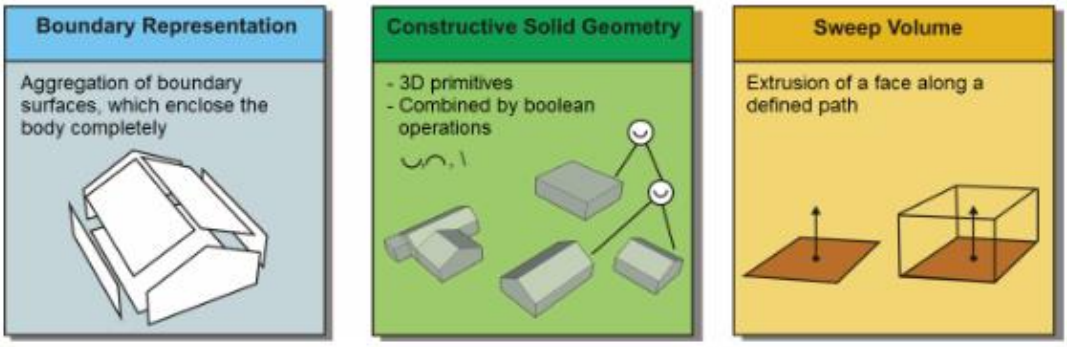

Figura 7: Categorias de métodos mais utilizados na representação de sólidos (Donkers, 2013).

A leitura de informações geométricas num ficheiro IFC que possui geometrias definidas por operações de varrimento ou por CSG requer que as operações de definição dos sólidos sejam executadas para o conhecimento das geometrias dos elementos. Já ficheiros que são construídos com B-Rep possuem uma complexidade maior para implementação da geometria dos elementos, que podem ser calculados de várias formas para representar um sólido. Porém, a sua definição explícita de geometria torna a sua representação em visualizadores mais expedita. Na maior parte dos casos práticos, a geometria dos elementos no IFC é definida através de operações de varrimento e de CSG. Por outro lado, a definição explícita de elementos representados em B-Rep permite mais facilmente as análises de volume e verificações de colisão entre elementos (Donkers, 2013).

Apesar do modelo de dados do IFC dar suporte a informações em todo o ciclo de vida e domínios envolvidos numa edificação, normalmente a troca de dados entre os atores de um projeto ocorre através de conjuntos de informações mais especializadas. O Information Delivery Manual (IDM) tem por

objetivo definir a informação requerida para os fluxos existentes num projeto com recurso a BIM, tendo a metodologia para sua elaboração sido descrita na ISO 29481. Dentro deste contexto, a utilização de todo o modelo do IFC nos fluxos de um projeto incidiria na transmissão de informação desnecessária ou redundante, de forma que surge a necessidade de definir um subconjunto do modelo IFC próprio às necessidades de troca de informação num determinado processo, o que é implementado através do conceito do Model View Definition (MVD).

2.3 O licenciamento digital

Os avanços tecnológicos recentes estimularam o desenvolvimento de investigação para propiciar formas mais eficientes de se desenvolver processos na indústria AEC. Uma das frentes de investigação que tem se desenvolvido mais nos últimos anos está relacionada à digitalização dos processos de licenciamento, devido ao potencial de economia de tempo na execução dos processos a partir da automatização parcial das verificações de conformidade, já que estes processos ocorrem de forma manual na maioria dos casos. A digitalização nesses processos pode ocorrer em diversos níveis, podendo se caracterizar somente pela digitalização dos documentos e dos fluxos de trabalho numa plataforma. Contudo, a evolução das tecnologias e ferramentas BIM proporciona, através do desenvolvimento de modelos da informação, a possibilidade de automatizar parcialmente a verificação de conformidade, sendo um dos campos mais promissores (Figura 8).

Figura 8: Níveis da digitalização do processo de licenciamento. Adaptado de Shahi et al. (2019).

A automatização da verificação de conformidade tem por objetivo substituir os processos de análise manuais de projetos em 2D por processos integralmente ou parcialmente automáticos, através da execução de verificações de modelos BIM em aplicações através de um conjunto de regras criado através da implementação dos regulamentos num formato interpretável por máquina (C. Eastman et al., 2009). Um fluxo proposto para a automatização desse processo pode ser visto na Figura 9. O sucesso desta automatização torna a deliberação dos processos de licenciamento mais ágil,

transparente e menos suscetível a erros, consequentemente traduzindo-se em economia para o erário público e claras vantagens para a sociedade. A automatização da verificação também pode permitir, potencialmente, a análise de requisitos das edificações que hoje, em Portugal, têm conformidades atestadas pelos projetistas e técnicos dos municípios.

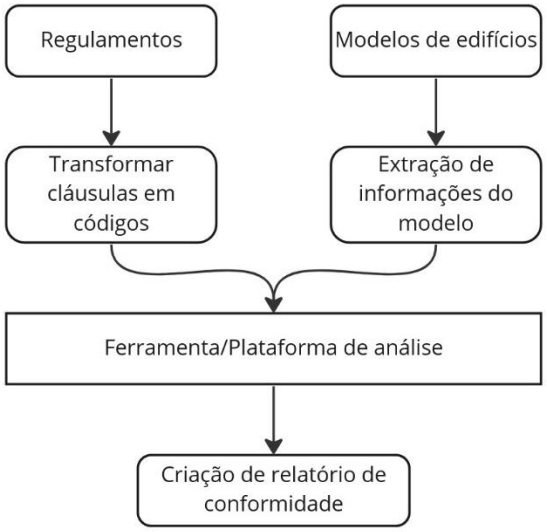

Figura 9: Fluxo de licenciamento digital com recurso a BIM. Adaptado de Shahi et al. (2019).

2.3.1 Casos de implementação e investigação do licenciamento digital

A implementação de sistemas digitais de licenciamento municipal com recurso a BIM possui alguns exemplos de sucesso, de forma que alguns dos mais relevantes são os de Singapura, Finlândia, Estados Unidos (Shahi et al., 2019). Além disso, o tema tem sido largamente investigado, o que resulta numa série de experimentos, iniciativas e pilotos desenvolvidos nas últimas décadas. Entretanto, devido à abrangência de conceitos necessários, agregados às implementações neste âmbito, grande parte das soluções propostas apresentam-se fragmentadas (Noardo et al., 2021), o que por vezes limita sua capacidade de implementação em distintos contextos, assim como não fornece uma solução integrada para os desafios impostos. Finalmente, alguns projetos recentes visam propor soluções mais abrangentes de forma a mitigar esta fragmentação, com abordagens e âmbitos mais abrangentes.

2.3.1.1 Iniciativas de implementação

Singapura é pioneira no desenvolvimento de soluções para a digitalização do licenciamento. O sistema digital de troca de informações sobre a construção e o mercado imobiliário, denominado CORENET, foi lançado em 1995. Após a implementação do sistema de submissão de desenhos 2D digitais dos projetos em 2002 e de modelos BIM em 2011, o tempo de processamento para a emissão de licenças

foi reduzido em 75%. Em 2015, o uso de BIM para projetos de construção passou a ser obrigatório. Hoje, o CORENET também conta com um sistema de verificação regulamentar, chamado FORNAX, que consiste numa biblioteca de objetos que expande as informações contidas no modelo de dados IFC, a fim de introduzir informações necessárias para a verificação dos regulamentos (C. Eastman et al., 2009). As verificações do FORNAX abrangem parte do código de segurança contra incêndio, assim como algumas verificações relacionadas ao plano de desenvolvimento urbano e aos códigos de edificações (Shahi et al., 2019).

Já o verificador de modelos Solibri Model Checker (SMC) foi desenvolvido originalmente na Finlândia e possui a capacidade de executar verificações de conformidade automáticas em modelos IFC. Diferentemente do sistema FORNAX, é possível inserir as regras que serão utilizadas para verificação. Em relação à abrangência das verificações regulamentares automáticas no caso finlandês, o código de segurança contra incêndio e os códigos de acessibilidade podem ser parcialmente verificados (Preidel & Borrmann, 2015).

Assim como o Solibri, o Jotne EDModelChecker possui capacidades de codificação de regras, sendo estas em formato EXPRESS, de forma que já foi utilizado para verificação em algumas iniciativas, a exemplo da promovida pelo *National Office of Building Technology and Administration* (BE) na Noruega (C. Eastman et al., 2009).

O Departamento de Energia dos Estados Unidos (DOE) possui duas aplicações web, denominadas COMcheckweb e REScheckweb, que executam a verificação dos edifícios em função do código internacional de conservação de energia (IECC) e das normas ASHRAE. Estas aplicações estão disponíveis para uso gratuito online, de forma a dar suporte no desenvolvimento de projetos em conformidade com estes regulamentos (Shahi et al., 2019). Contudo, este sistema ainda não possui suporte à submissão e verificação de modelos BIM.

Em 2020, ocorreu a implementação na cidade de Salvador, no Brasil, da plataforma Metropolis, para licenciamento de edifícios novos multifamiliares com a verificação automática de alguns parâmetros estabelecidos em códigos de edificação através da análise de modelos BIM. Apesar de inovador, a plataforma ainda exige, além do ficheiro IFC, a submissão de projetos em PDF e no formato proprietário RVT («Análise da plataforma de verificação de regras de salvador, A Metropolis», 2022).

Entretanto, um aspeto relevante a ser apontado é o desenvolvimento de um sistema de classificação de forma a dar suporte às necessidades de informação advindas dos requisitos regulamentares que não

são suportadas pelos modelos de dados BIM utilizados. Este sistema é baseado na OMNICLASS e trata de elementos como espaços, níveis e elementos construtivos (Torreão & Novais, 2020).

2.3.1.2 Iniciativas a nível de projetos e redes de investigação

As iniciativas para desenvolvimento de ferramentas tecnológicas para execução de verificações automáticas de conformidade incluem a investigação de formas de se automatizar o processo de verificação dos projetos que é executado nos processos de licenciamento da construção.

Devido à complexidade intrínseca ao processo de transição dos processos de licenciamento vigentes para um formato digital, diversos pesquisadores e intervenientes com trabalhos nesta área e áreas interligadas fundaram uma rede de colaboração, no início de 2020, denominada European Network for Digital Building Permits - EUnet4DBP (Noardo et al., 2020).

O projeto europeu Change toolkit for digital building permit (CHEK)[4] tem o objetivo geral de permitir o desenvolvimento e adoção de métodos digitais para o licenciamento de obras a partir de modelos de dados representativos do ambiente construído ao nível do edifício e da cidade. Este objetivo é dividido em frentes de trabalho que visam estabelecer o processo a ser adotado em licenciamentos digitais, os meios para treinar os intervenientes no processo e adoção das novas tecnologias. O consórcio do projeto é composto por 19 entidades, entre instituições de pesquisa, autarquias, projetistas, empresas de software e entidades normativas. Entre os impactos esperados pelo projeto estão a redução do tempo de processamento para emissão de licenças em 30% e a redução em 80% dos casos em que ocorrem diversas submissões para complementar documentos de requerimento de licenças.

O projeto ACCORD[5] visa digitalizar os processos de licenciamento utilizando BIM e outras tecnologias para melhorar a eficiência e qualidade dos processos de conceção e construção, consequentemente apoiando o desenvolvimento de um parque edificado mais sustentável. Assim, o foco é desenvolver soluções técnicas para automatizar a verificação da conformidade dos edifícios, incluindo requisitos de ordem ambiental (emissão de CO2, análise do ciclo de vida e economia circular) e criar uma estrutura semântica para processos de licenciamento digital de edifícios com base em formatos abertos e neutros para troca de informações. O projeto tem por objetivo demonstrar estas soluções em vários países (Reino Unido, Finlândia, Estónia, Alemanha, Espanha), além de desenvolver APIs,

[4] https://chekdbp.eu/

[5] https://cris.vtt.fi/en/projects/automated-compliance-checks-for-construction-renovation-or-demoli

proporcionando maior acesso às soluções criadas. O objetivo final é criar um ecossistema de licenciamento escalável, durável e flexível.

Já o projeto DigiChecks tem por objetivo propor um framework digital composto de quatro passos a fim de superar os desafios existentes na implementação de um processo digital de licenciamento. Estes quatro passos envolvem: a criação de uma ontologia voltada para os licenciamentos, a fim de homogeneizar a linguagem utilizada para este domínio; o desenvolvimento de ferramentas para modelação dos processos de licenciamento baseado em padrões utilizados pelo *Object Management Group* (OMG); a disponibilização de codificação de regras a ser utilizadas em verificações automáticas de conformidade; e a integração de soluções através de uma API aberta. O projeto tem fim previsto para maio de 2025 e conta com 13 parceiros entre empresas de construção, entidades de pesquisa e ensino, entidades de certificação e empresas especializadas em tecnologias da informação.

Também podemos destacar o projecto BRISE[6], que visa melhorar a eficiência e sustentabilidade dos processos de licenciamento de construção em Viena, com recurso a tecnologias como BIM, realidade aumentada, robótica, e inteligência artificial. O projeto envolve a colaboração entre a Cidade de Viena, a Universidade de Tecnologia de Viena, especialistas em BIM, e a Ordem dos Engenheiros Civis da Áustria, e é financiado pela iniciativa Urban Innovative Actions (UIA) da União Europeia. O BRISE visa reduzir, até 50%, o tempo de processamento dos pedidos de licenças de construção e tornar os projetos de construção mais transparentes e compreensíveis para os cidadãos. O projeto visa também fornecer importantes informações e estudos de caso para o desenvolvimento futuro dos processos administrativos digitais e para melhorar a eficiência, a facilidade de utilização e a sustentabilidade do Governo em Viena. Espera-se que o BRISE seja implementado até ao segundo trimestre de 2023.

Além do envolvimento no âmbito do projeto CHEK, a investigação acerca da digitalização do processo de licenciamento em Portugal conta com outros exemplos. O LicA é uma aplicação, apresentada em trabalho publicado por Martins & Monteiro (2013), para executar verificações automáticas de sistemas de abastecimento de água a partir de modelos BIM em face a uma base de dados de requisitos criada a partir dos regulamentos portugueses. Já Santos (2021), em sua dissertação de mestrado, propôs um protótipo de plataforma para execução, em estudos de caso, da aplicação de artigos do Regulamento Geral das Edificações Urbanas e dos regulamentos municipais de Vila nova de Gaia, de forma a mostrar a viabilidade da implementação de fluxos de verificação automática regulamentar num

[6] https://digitales.wien.gv.at/projekt/brisevienna/

contexto português a partir da análise de modelos IFC. A implementação dos regulamentos foi realizada em linguagem de programação. Finalmente, o trabalho apresentou uma análise do potencial de digitalização do Plano Diretor Municipal de Vila Nova de Gaia, que determinou que mais de 40% do regulamento é codificável através da implementação de algoritmos de programação procedimental.

2.3.2 Interpretação regulamentar para obtenção de regras de verificação

Para viabilizar verificações automáticas de conformidade a partir de modelos BIM, num fluxo digital para emissão de licenças, é necessário implementar os regulamentos de edificação num formato digital que permita a sua leitura por rotinas de programação, que posteriormente irão executar verificações baseadas em processamento de geometria e análise de informações alfanuméricas (J. M. Lee, 2010). Estes regulamentos, portanto, em seu formato digital, devem ser claros a fim de estabelecer de forma inequívoca os parâmetros que deverão ser utilizados para a verificação de cada regra. Entretanto, as regras existentes nestes regulamentos possuem frequentemente um caráter ambíguo e subjetivo, o que deriva frequentemente da necessidade de se estabelecer uma análise casuística das operações urbanísticas, o que impossibilita uma simples conversão dessas regras num formato interpretável por máquina.

Perante o exposto, a automatização do processo de análise de projetos deve ser parcial, com foco na verificação de requisitos que possam ser clarificados e tornados objetivos através de um processo de interpretação humana da linguagem natural, sendo este pautado numa análise do corpo regulamentar para estabelecer quais as regras que são passíveis de automatização (Soliman-Junior et al., 2021). Também é importante considerar que em alguns casos a edição dos regulamentos existentes pode ser fortuita para se esclarecer alguns requisitos e facilitar sua digitalização.

Entre os diversos métodos empregados para implementar os requisitos regulamentares em verificações automáticas, muitos baseiam-se em mecanismos proprietários, implementados em codificação rígida e específicas a um domínio, o que implica em soluções de difícil manutenção e inflexíveis para se adaptar aos diversos tipos de regulamentos existentes (Nawari, 2019). Outros métodos, entretanto, adotam uma abordagem aberta, baseada na interpretação da linguagem natural através de representações lógicas que podem posteriormente ser implementadas em rotinas de programação.

As tabelas de decisão, apresentadas por Fenves (1966), são uma das tentativas iniciais mais relevantes para representar os requisitos regulamentares de forma objetiva e flexível. Estas tabelas

consistem numa representação tabular de uma série de condições que devem ser verificadas para a tomada de uma determinada ação. Num trabalho mais recente, Tan et al. (2010), propuseram a utilização destas tabelas de decisão para permitir a verificação de conformidade de edifícios face a regulamentos relativos à envolvente de edifícios. Os pacotes de regras, como denominados no trabalho, consistem na representação das diversas cláusulas dos regulamentos em grupos de tabelas de decisão que contêm a lógica contida nestas regras, incluindo a possibilidade de representação de relações entre cláusulas. Entretanto, esta abordagem possui algumas limitações para a representação de regras que determinam relações entre objetos na edificação, além de representar de forma muito simples as interdependências que podem existir nos regulamentos.

A fim de formalizar a especificação do American Institute of Steel Construction (AISC), Nyman et al. (1973) expandiram a utilização das tabelas de decisão, como suporte à restruturação do documento, através de um modelo de quatro níveis. O modelo lógico utilizado neste trabalho serviu para o desenvolvimento da metodologia Standards Analysis, Synthesis, and Expression (SASE), que por sua vez considera a utilização das tabelas de decisão de forma a representar a lógica dos regulamentos ao nível das provisões (Figura 10). Desta forma os regulamentos são apresentados através de um modelo hierarquizado e interconectado com o intuito de clarificar os requisitos neles existentes.

		1	2	3	4	5	6	7	8	9	10
1	Clearance < 6'	T	T	T	T	T	T	T	T	T	F
2	Sign const. = closed	T	T	F	F	T	T	F	F	.	.
3	Bldg. type = 1 or 2	T	-	T	-	T	-	T	-	F	.
4	Bldg. type = 3	-	T	-	T	-	T	-	T	F	.
5	Height > 35'	.	F	.	.	-	T	+	+	.	.
6	Height > 50'	F	-	.	.	T	.	+	+	.	.
7	Height > 60'	-	-	.	F	.	.	+	T	.	.
8	Height > 100'	-	-	F	-	.	.	T	.	.	.
1	Height acceptable	X	X	X	X						
2	Height not acceptable					X	X	X	X	X	X

Figura 10: Tabela de decisão para representação de verificação de altura (S. J. Fenves et al., 1987).

O trabalho de Lee (2010) apresenta a utilização de tabelas paramétricas para representar os requisitos regulamentares relativos à circulação. A determinação dos parâmetros a ser utilizados derivou de uma análise do Guia de Projeto dos Tribunais dos EUA, de forma que o autor comenta que a generalização destes parâmetros para a verificação de regras de circulação de outros regulamentos ainda carece de

mais testes. Assim, a aplicação de forma generalizada de tabelas de parâmetros para outros tipos de regras, além das de circulação, envolve um trabalho de análise de todos os demais tipos de regras a se verificar de forma a estabelecer quais são os parâmetros fundamentais para a sua verificação, além da possível necessidade de atualização destes consoante mudanças na forma de se verificar as referidas regras.

Através da utilização de gráficos conceptuais e sua fundamentação em lógica de primeira ordem (LPO), Solihin & Eastman (2016) propuseram uma abordagem para interpretar os requisitos de códigos de edificação de forma a definir claramente quais as entidades e conceitos necessários para verificar determinadas regras. A abordagem já utiliza uma referência direta ao modelo IFC na determinação das regras. Entretanto, os autores comentam que há a necessidade frequente de um pré-processamento da lógica presente no texto original a exemplo dos casos em que há mais de uma regra por cláusula, o que requer a separação do texto em regras "atómicas".

Nawari (2019) propôs uma metodologia generalizada adaptativa (GAF) para automatizar integral ou parcialmente o processo de verificação regulamentar. Esta metodologia é baseada numa representação das regras de verificação orientada a objetos, que podem lidar tanto com dados objetivos como com dados ambíguos e transformar regulamentos em expressões computadorizáveis com recurso a um algoritmo de classificação e conceptualização das regras em linguagem natural. No seu trabalho, ele utiliza a implementação do modelo IFC na linguagem de marcação XML e utiliza *Language Integrated Query* (LINQ) para o mapeamento das entidades expressas nas cláusulas dos regulamentos com o IFC, o que acaba por se traduzir numa aplicação em linguagem de programação C#.

A partir da premissa de que todas as soluções para implementação de verificações automáticas de conformidade necessitam de um mapeamento entre os conceitos empregados nos regulamentos e as entidades no modelo BIM, Zhang & El-Gohary (2020) propuseram uma abordagem baseada em *machine-learning* que mapeia os conceitos e relações de regulamentos a elementos e relações do modelo IFC. Os dados utilizados para treinar o modelo de mapeamento foram obtidos a partir do "*2009 international Building Code"* e do "*2015 International Building Code Ammendments".* Os resultados foram 76% de acerto no mapeamento de conceitos e 78% no acerto de mapeamento de relações.

Alguns métodos partem do princípio de que o modelo IFC não possui capacidade para representar todas as entidades e conceitos necessários para se verificar a conformidade face aos regulamentos.

Pauwels et al. (2011) comentam que a linguagem EXPRESS possui limitações para a representação de alguns conceitos, de forma que propuseram a utilização de linguagens de regra, como a N3Logic, para realizar verificações em modelos IFC, convertidos numa representação com recurso a *Resource Description Framework* (RDF) e Ontologias. Existem também outras abordagens que seguem o uso de ontologia e *web semantics* para a representação de regulamentos (Bouzidi et al., 2013; Yurchyshyna & Zarli, 2009). Porém, alguns autores comentam que estes métodos possuem pouca expressividade para representar algumas das partes dos regulamentos de edificações (Nawari, 2019; Z. Zhang et al., 2023).

Hjelseth & Nisbet, (2011) desenvolveram um método de marcação semântica para classificar as cláusulas dos regulamentos a partir da utilização de quatro operadores, representados na Figura 11: Requirement [R]; Applies [A]; Select [S]; e Exception [E], de forma que o método é conhecido como RASE. O operador Requirement estabelece as condições que deverão ser atendidas; já o operador Applies define o objeto, seja este um aspeto ou entidade ao qual a verificação será aplicada; o operador Select é utilizado para determinar condicionantes para a aplicação dos requisitos, caso estes não sejam aplicados a todos os elementos estipulados; e o operador Exception que identifica condicionantes para a não aplicação dos requisitos. Uma das vantagens do método RASE é a simplicidade de sua aplicação para marcar os textos dos regulamentos. Ainda assim, isto requer que o texto seja bem estruturado, o que não é a realidade da maior parte dos regulamentos utilizados no processo de licenciamento.

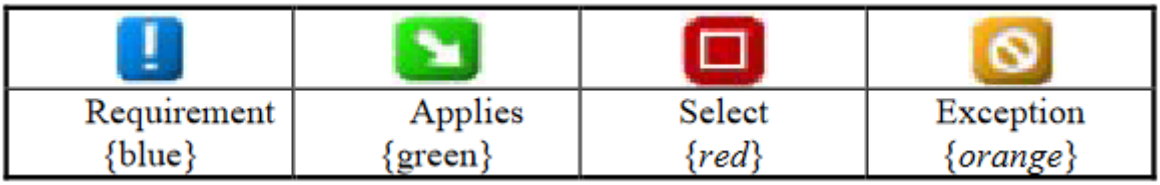

Figura 11: Operadores do método RASE (Hjelseth & Nisbet, 2011).

Com o objetivo de colmatar os problemas da aplicação do método RASE, Ilal e Gunaydin (2017) desenvolveram um método para representar regulamentos de edificação que fossem independentes de sistemas de verificação, mitigasse os problemas advindos das ambiguidades e redundâncias existentes nas cláusulas, tivesse abrangência na representação de informação e tivesse fácil manutenção. A implementação deste método se dá através da adaptação dos operadores utilizados no método RASE numa reestruturação dos regulamentos desenvolvida em quatro níveis:

- Nível de domínio: modelação dos conceitos implementados no regulamento original com seus atributos e relações;

- Nível da regra: utilização do método RASE para representação das regras com os conceitos implementados no nível de domínio;
- Nível do grupo de regras: agrupamento de conceitos e relações entre as cláusulas;
- Nível de gestão: conceptualização e organização do regulamento.

Um procedimento importante para a organização e perceção do potencial de transformação de regulamentos para um formato interpretável por máquina é a classificação das suas cláusulas. Macit et al. (2013) desenvolveram um estudo em que analisaram o código de edificações da cidade turca de Izmir, sendo que, a partir da classificação das cláusulas, identificaram que 79% têm capacidade de tornarem-se regras interpretáveis por computador. Já Olsson et al. (2018) avaliaram os requisitos presentes no plano de desenvolvimentos sueco, em que foi verificado que 83% do conteúdo incluia requisitos quantitativos, a exemplo de medidas e distâncias, de forma que isto demonstra um bom potencial para a digitalização destes códigos. Finalmente, a partir da análise dos requisitos de informação presentes no processo de licenciamento em Tirol do Sul, na Itália, Plazza et al. (2019) Identificaram que 91% dos requisitos eram objetivos e quantitativos, apesar de 63% dos requisitos necessitarem de cálculo ou derivação, não sendo autocontidos. Estes exemplos demonstram um potencial para a implementação dos regulamentos em formato interpretável por máquina. Entretanto, a reformulação da estrutura dos códigos é um fator importante no desenvolvimento de regulamentos digitais.

2.3.3 Implementação de regras para verificação automática de conformidade

O processo de interpretação dos regulamentos permite a transposição dos requisitos regulamentares numa estrutura lógica, o que facilita a sua implementação para as rotinas de verificação. Entretanto, na maior parte dos casos, o produto da interpretação é intermediário, o que requer sua implementação numa linguagem de programação para a execução das verificações (Pauwels et al., 2011). Assim, existem algumas iniciativas que visam o desenvolvimento de linguagens de programação e de consulta especializadas, focadas na análise, processamento e obtenção de informação de modelos BIM, o que permite a implementação dos requisitos regulamentares interpretados ou a direta interpretação destes requisitos num formato a ser executado.

Alguns destes estudos utilizaram estas capacidades de verificação através de tabelas paramétricas existentes no software Solibri Model Checker (SMC). Soliman-Junior et al. (2021), a partir de uma análise dos regulamentos para edificações do setor de saúde no Reino Unido, verificaram que

aproximadamente metade dos requisitos regulamentares estabelecidos eram passíveis de representação numa regra lógica, todavia não puderam ser incorporados no Solibri, devido à falta de flexibilidade proporcionada pelos seus parâmetros programáveis. Assim, devido às limitações da abordagem baseada em tabelas de parâmetros, a capacidade de verificação de regras não é abrangente. Além disso, a falta de transparência na verificação do software pode levar a resultados falsos e inconsistentes.

As abordagens baseadas na criação de linguagens de domínio específico (*Domain Specific Languages - DSL*) visam obter a flexibilidade existente nas linguagens de programação clássicas, mas com a sintaxe e funcionalidades voltadas para a análise, consulta e verificações de modelos BIM. Uma das grandes vantagens da utilização deste tipo de linguagem está relacionada com o maior nível de abstração que tem em relação às linguagens de uso geral, já que sua sintaxe é desenvolvida a partir da terminologia utilizada no domínio. Isso garante que um usuário de uma DSL possa focar-se na lógica da implementação que se deseja desenvolver ao invés da sintaxe da linguagem (Langlois et al., 2007). Outra questão é que a estrutura lógica presente nas cláusulas regulamentares acaba por ensejar uma implementação que seja capaz de transpor esse tipo de estrutura, o que recai na utilização de abordagens baseadas em linguagens de programação (Solihin et al., 2019).

As DSLs podem ser categorizadas em internas ou externas em função da sua forma de implementação (Cunningham, 2008). As linguagens internas utilizam da capacidade de interpretação de uma linguagem já existente para expandir e modificar a sua sintaxe para um domínio específico, de forma que possuem as vantagens inerentes à existência de uma maior comunidade de desenvolvimento, bem como a presença de mais ferramentas e recursos para desenvolvimento, por exemplo, a aplicação FORNAX™, para verificação automática de conformidades, utiliza funções de script baseadas em Lua (Solihin et al., 2019). As linguagens externas possuem novas sintaxes e a necessidade de desenvolvimento de um interpretador específico, o que lhes garante a capacidade de serem mais específicas a um determinado domínio, mas normalmente sofrem pela falta de recursos que facilitam sua implementação, a exemplo do suporte de interfaces de desenvolvimento e comunidades.

A BIM *Rule Language* (BIMRL) é uma DSL externa, que utiliza as capacidades da SQL, com uma sintaxe própria (Figura 12), e tem por objetivo realizar consultas e verificações a partir de uma estrutura de dados relacional que contém as informações carregadas a partir de modelos IFC. Esta estrutura consegue tratar as informações alfanuméricas e geométricas através duma implementação

baseada numa representação em estrela, utilizada usualmente para tratar dados em *Data Warehouses.* A BIMRL também possui a capacidade de processamento e realização de algumas operações geométricas, assim como a possibilidade da criação de novas geometrias para a verificação, o que facilita a análise de casos onde, por exemplo, é necessário verificar a interseção de objetos do modelo com planos descritos em regulamento (Dimyadi, 2016; Solihin, 2015). Assim como outras DSLs, a linguagem sofre pela falta de suporte e comunidade, além de não ser de fácil acesso para a sua implementação (Solihin et al., 2019). Além disso, a necessidade de se transpor os dados do modelo IFC para um outro formato implica numa possível perda de informações, além de um maior trabalho para implementar suporte para futuras versões do formato.

```
CHECK IfcSpace S, BIMRL_TOPO_FACE F
WHERE F.ElementId = S.ElementId
AND F.Type = 'BODY'
AND F.Orientation in ('BOTTOM','SIDE')
COLLECT S.ElementId SpaceId,S.Name SpaceNo,S.LongName SpaceName,
  F.Polygon Geom,PROPERTY(S,Height,BaseQuantities) SpaceHeight;
EVALUATE SmallestRectangularEdge(Geom) OUTPUT ?SmallestEdge;
ACTION WHEN ?SmallestEdge < 1000 {PRINT RESULT};
```

Figura 12: Representação de implementação da BIM *Rule Language* (Dimyadi, 2016).

Assim como a BIMRL, a *Building Environment Rule and Analysis Language* (BERA) necessita da transposição das informações contidas num modelo IFC para um outro formato, o BERA *Object Model* (BOM). Entretanto, as capacidades de associação de informações estáticas e dinâmicas ao BOM permitem mitigar parcialmente o desafio relativo à distinção de conceitos entre os regulamentos e o modelo IFC. A linguagem implementa a *dot-notation* para o acesso a informações do modelo, o que contribui para um nível de abstração superior e facilita a sua utilização. Apesar da sintaxe da linguagem ser aberta, o que possibilita a implementação em diversos interpretadores, a aplicação mais conhecida da BERA é implementada em Java através de um plug-in do Solibri Model Checker, o que limita a sua utilização a um software proprietário (J. K. Lee, 2011).

A linguagem KBimCode é uma DSL que, assim como a linguagem BERA, utiliza um modelo de objetos como base para representar as entidades contidas no *Korean Building Act* e consequentemente transformar suas regras num formato interpretável por máquina. A proposta prevê que a linguagem seja intermediária, com definição das especificações e do léxico, que foi desenvolvido a partir da definição de cláusulas atômicas dos regulamentos, de forma a que possa ser implementada posteriormente em linguagens formais para execução das verificações (Park et al., 2016). Uma dessas

implementações foi realizada em linguagem Python no software KBimAssess, que utiliza o IFC como modelo BIM para verificação, de forma que um dos pontos críticos foi o mapeamento necessário entre as entidades, propriedades e métodos de verificação existente na linguagem com os existentes no IFC (Song et al., 2019). O resultado é um regulamento interpretável por máquina onde as verificações estão formalizadas em código Python embebidos num ficheiro XML (Figura 13).

```
<CcCode>
  <Id>101</Id>
  <ParentId>1</ParentId>
  <Number>공간객체의 유무</Number>
  <Desc>모든 영역에 공간객체가 모델링되어 있어야 한다.</Desc>
  <Script>
  <![CDATA[def Check():
for storey in SELECT('storey'):
    undef_spaces = storey.SELECT('undefined space')
    if undef_spaces.COUNT() == 0:
        storey.SUCCESS('모든 영역에 공간객체가 모델링되어 있습니다.')
    else:
        for undefined_space in undef_spaces:
            area = undefined_space.SELECT('area').UNIT('m2').NUMBER()
            if area > 1:
                undefined_space.FAIL('공간객체 없음 (면적: ' + str(area) + 'm2)')
            else:
                undefined_space.WARNING('공간객체 없음 (면적: ' + str(area) + 'm2)')]]>
  </Script>
</CcCode>
<CcCode>
```

Figura 13: Exemplo da KBimCode para representação de regra do Korean Building Act[7].

As implementações de regulamentos digitais baseadas em linguagens de *scripting* apresentam um desafio, já que os técnicos especializados nos regulamentos municipais usualmente não possuem a especialização necessária para a implementação de linguagens a este nível de abstração, o que gera a necessidade de técnicos especializados em programação e mais uma barreira na implementação destas soluções (Kim et al., 2019).

Neste contexto, as linguagens de programação visual, por serem baseadas em símbolos e representações gráficas de conceitos da programação, apresentam um equilíbrio entre a capacidade de execução de códigos flexíveis e a facilidade de utilização (Solihin & Eastman, 2016).

Existem alguns exemplos da aplicação de linguagens visuais para a consulta e análise de modelos BIM, mas poucas possuem o foco na verificação automática de conformidades. Wülfing et al. (2014) desenvolveram uma linguagem de consulta a modelos BIM, denominada *Visual* BIM *Query Language* (VBQL). A partir de uma abordagem baseada em blocos (Figura 14), desenvolvida a partir da extinta iniciativa OpenBlocks, a linguagem era capaz de realizar a extração de informações de modelos IFC, assim como a verificação de algumas condições, retornando resultados booleanos. Os

[7] https://www.inno-lab.co.kr/Home/en/product-KBim_Assess-Lite.html

desenvolvimentos da linguagem e do protótipo foram maioritariamente baseados em Java. Além disso, os recursos utilizados para a criação do editor de código e do visualizador já não possuem mais suporte para desenvolvimento. Outra iniciativa relevante nesta área está presente no trabalho de Daum & Borrmann (2015), que apresenta uma sintaxe gráfica para implementação de um sistema de consulta a modelos BIM.

Figura 14: Editor de código para realização da consulta a modelos BIM (Wülfing et al., 2014).

A linguagem *Visual Code Checking Language* (VCCL) foi proposta no estudo de Preidel & Borrmann (2015), sendo uma linguagem baseada num fluxo composto de linhas e nós. O objetivo do estudo era provar o conceito da aplicação de linguagens visuais para execução de verificações de conformidade, o que foi executado através da implementação de uma aplicação em conjunto com a empresa de software Nemetscheck (Figura 15). Finalmente, a partir da análise dos objetos, métodos e referências presentes nos regulamentos do *Korean Building Act*, o trabalho de Kim et al. (2019) apresenta o desenvolvimento de uma linguagem visual capaz de gerar automaticamente scripts de KBimCode e consequentemente executar as verificações de conformidade.

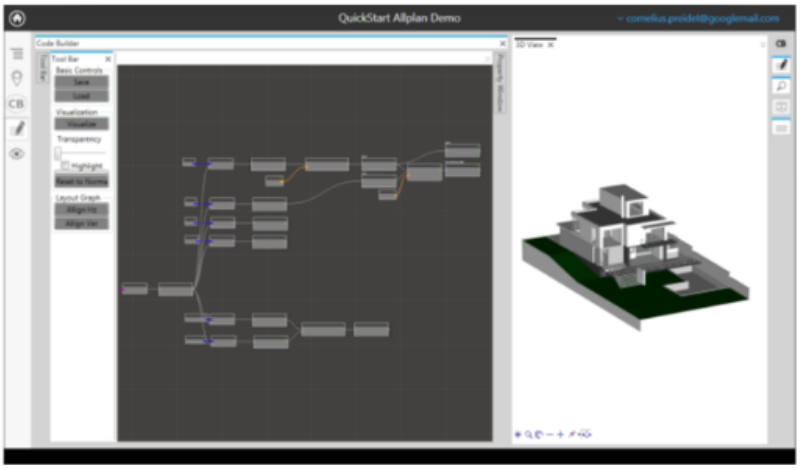

Figura 15: Interface do utilizador do CodeBuilder (Preidel & Borrmann, 2015).

2.4 Recursos para desenvolvimento de ferramentas e plataformas openBIM

2.4.1 Enquadramento

O emprego de modelos de informação no setor da construção tem o potencial expandido através da utilização de recursos voltados aos sistemas da informação. Assim, é possível usufruir de uma série de linguagens e suas bibliotecas para criar aplicações voltadas à otimização e automatização de variados processos. Linguagens de programação muito populares para o desenvolvimento de análises de dados, a exemplo do Python, estão a ser largamente utilizadas para realizar o processamento de modelos BIM de forma a gerar aplicações independentes e complementares a plataformas através de APIs. Além disso, o openBIM tende a impulsionar este quadro a partir da promoção do modelo de dados aberto IFC, que tornou-se padrão para a interoperabilidade no setor.

2.4.2 Análise e processamento de modelos IFC

A análise de ficheiros IFC parte da interpretação das informações do modelo, que usualmente estão implementadas em formato STEP, a partir do modelo de dados definido em linguagem EXPRESS. Assim, as ferramentas de interpretação são capazes de identificar dados válidos e inválidos, assim como tokens relevantes dentro de um campo de dados de forma a representá-los de uma forma normalizada (Loshin, 2010). Em termo práticos, a interpretação dos ficheiros IFC permite o acesso à informação contida no modelo a partir de um maior nível de abstração, o que facilita a análise e o processamento dos seus dados.

Apesar da relevância de ferramentas de interpretação, como o IFC-SDK e o xBIM, as ferramentas mais populares atualmente são o Web-ifc e o IfcOpenShell. Estas ferramentas não somente possuem a capacidade de interpretar os ficheiros IFC, como também agregam outros métodos que possibilitam a sua análise e processamento a partir das informações do modelo.

O Web-ifc é uma biblioteca JavaScript que possui um módulo desenvolvido originalmente em C++, que utiliza os recursos do WebAssembly para interpretar modelos IFC num ambiente web a partir de código compilado. Isso resulta em tempos de carregamento do ficheiro muito menores comparativamente ao carregamento feito exclusivamente com JavaScript, que é uma linguagem interpretada. O desenvolvimento desta biblioteca faz parte do projeto IFC.js, de forma que é maioritariamente utilizado para o carregamento da geometria dos modelos para posterior visualização em bibliotecas de visualização de elementos 3D, como o Three.Js. Assim, a biblioteca é utilizada como motor de interpretação e obtenção de geometria para os visualizadores do IFC.js assim como visualizadores oriundos de outras iniciativas, como o Xeokit[8].

Já o IfcOpenShell é um conjunto de bibliotecas de código aberto para a manipulação de ficheiros IFC, o que permite o desenvolvimento muito flexível de aplicações. O módulo principal de interpretação é desenvolvido em C++, mas possui uma interface de programação em Python, de tal forma que as capacidades de interpretação são largamente expandidas pela utilização da sua vasta quantidade de bibliotecas para processamento de dados disponíveis. Outra mais-valia é a utilização da *dot-notation* utilizada na linguagem Python para aceder às propriedades e relações dos elementos do modelo, fornecendo uma camada superior de abstração além de tornar a escrita de código para tratamento de informações facilitada pela extensa documentação do IFC disponibilizada pela BuildingSmart [9].

Devido à considerável comunidade de desenvolvimento, composta por 151 contribuidores do repositório principal do GitHub, o IfcOpenShell possui uma larga escala de recursos e integrações, a exemplo de bibliotecas para comparação de alterações em ficheiros IFC, deteção de colisões e conversão para modelos de cidade (CityJSON), assim como a integração com o software de código aberto Blender para o desenvolvimento de um fluxo aberto de criação e edição de modelos IFC, contrário às práticas vigentes que são maioritariamente dependentes de softwares proprietários.

[8] https://github.com/IFCjs/web-ifc
[9] https://github.com/IfcOpenShell/IfcOpenShell

Entre os diversos módulos proporcionados pelo IfcOpensShell, o Selector destaca-se por utilizar as capacidades da biblioteca Lark para promover consultas complexas a ficheiros IFC através de comandos concisos e uma linguagem de consulta própria. Assim, a extração de informações que, de forma usual requereria a execução de loops e operadores condicionais, pode ser executada com poucas linhas de código (Figura 16).

```
# Open IFC file
ifc_file = ifcopenshell.open('backend\check_module\ifcfile.ifc')

# Initiate selector
selector = Selector()

# Get all slabs in the storey 'Pavimento1'
slabs = selector.parse(ifc_file,'@.IfcBuildingStorey[Name="Pavimento1"] & .IfcSlab')
```

Figura 16: Consulta executada em ficheiro IFC através do módulo Selector.

2.4.3 Trimesh

Trimesh é uma biblioteca Python de código aberto que proporciona uma série de classes e métodos para a criação, carregamento, visualização e processamento de malhas triangulares. Uma das grandes vantagens desta biblioteca reside na abstração da sintaxe implementada, já que a execução de operações geométricas é baseada em métodos intuitivos, o que facilita a sua aprendizagem e implementação.

Entre as funcionalidades da biblioteca, as que se destacam do ponto de vista da sua implementação, para o processamento geométrico a partir de modelos IFC, são[10]:

- Cálculos de área, volume, centro de massa e momento de inércia;

- Cálculo de seções transversais;

- Execução de operações booleanas em malhas;

- Voxelização (Figura 17).

[10] https://github.com/mikedh/trimesh

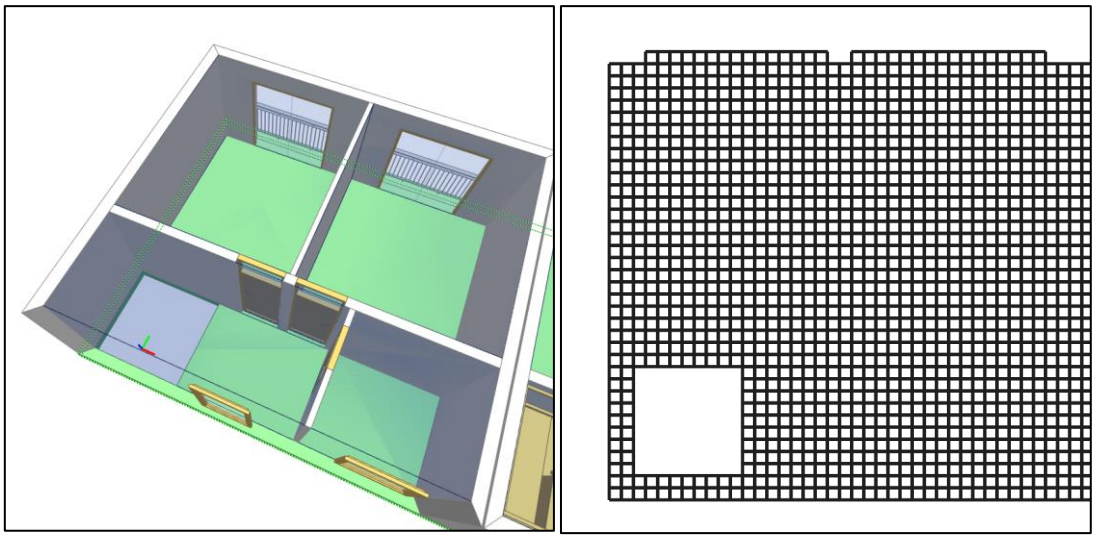

Figura 17: À esquerda: modelo IFC com seleção de laje, em verde, no software BIMVision. À direita: voxelização da laje executada pelo Trimesh.

2.4.4 Ferramentas de desenvolvimento web

As aplicações web são aquelas desenvolvidas com recurso a linguagens suportadas por navegadores web, por onde estas aplicações são acedidas. A grande vantagem das aplicações web em relação às aplicações nativas, que são acedidas através do ambiente local de um determinado sistema, é o alcance proporcionado pelo seu acesso através de redes web, o que propicia que possam usufruir das funcionalidades da aplicação através de diversos aparelhos, sejam estes computadores ou aparelhos móveis, desde que ligados à internet. Essa flexibilidade de acesso é vantajosa para o desenvolvimento de trabalhos colaborativos (Al-Fedaghi, 2011), em que ocorrem trocas e controle de informação, como é o caso dos processos desenvolvidos com o uso do BIM.

O desenvolvimento moderno de aplicações web é composto de três componentes imprescindíveis: um servidor, uma base de dados e linguagens de programação voltadas ao desenvolvimento web (Nurminen et al., 2008), que precisam de suporte dos navegadores. Assim, é usual a distinção entre duas grandes especialidades no desenvolvimento web: aquela voltada a aspetos relacionados à criação da interface e de elementos gráficos, comumente chamada de frontend; e a voltada para aspetos relacionados ao servidor e a base de dados, comumente chamada de backend. Nestas especializações, algumas linguagens que se destacam são o HTML, o JavaScript e CSS para o desenvolvimento do frontend e o PHP, Python e Java para o desenvolvimento do backend. Além disso, a utilização de

linguagens de programação para desenvolvimento web é usualmente realizada através de frameworks, que são soluções para facilitar e estruturar o desenvolvimento de aplicações. Na Figura 18 é possível verificar linguagens usualmente utilizadas para desenvolvimento web e frameworks utilizadas para estas linguagens entre parênteses.

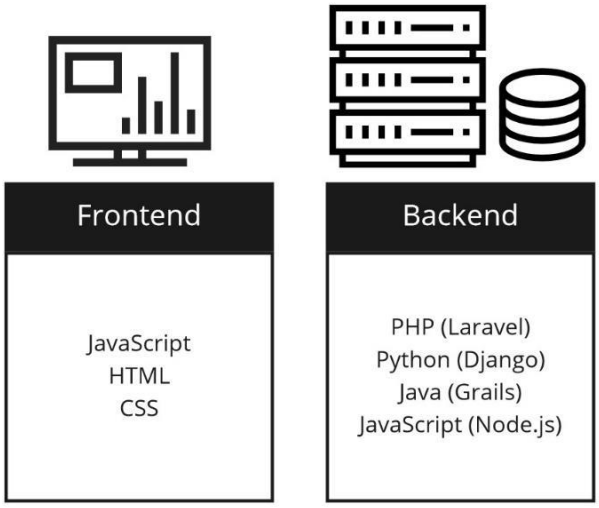

Figura 18: Linguagens e frameworks (entre parênteses) normalmente utilizadas para desenvolvimento frontend e backend.

Neste âmbito, destacam-se alguns recursos, entre bibliotecas e frameworks, que podem ser utilizados para o desenvolvimento web nas duas especialidades (frontend e backend):

Django:

Django é um framework para desenvolvimento web, de código aberto, criado a partir da necessidade de um rápido desenvolvimento orientado a base de dados (Forcier et al., 2008). Assim, aliado ao fato de ser desenvolvido em linguagem Python, a sua implementação permite focar o desenvolvimento na estrutura de dados e nos fluxos propostos, ao invés de aspetos como a sintaxe ou outros pormenores da linguagem.

Um importante aspeto na utilização do Django é a implementação do padrão *Model-View-Controller* (MVC), que implementa a modularidade a partir de três camadas: modelo, responsável pelo acesso e definição da estrutura da base de dados; vista, responsável por determinar quais e como os dados serão apresentados; controlador, que determina qual vista será selecionada a partir de uma determinada requisição (Holovaty & Kaplan-Moss, 2009).

Devido à simplicidade da definição dos controladores, os componentes mais enfatizados na implementação de um projeto em Django são os modelos e as vistas. O framework apoia-se na

utilização de uma técnica para converter dados definidos numa base de dados relacional com uma abordagem direcionada a objetos (*Object Relational Mapping* – ORM), o que facilita não somente a criação da estrutura de dados a ser utilizada, mas também as capacidades de consulta aos dados, já que a base para a sintaxe de consulta é a *dot-notation,* utilizada na linguagem Python para aceder aos atributos e métodos de um objeto (Ghimire, 2020).

Django REST Framework (DRF):

Representational State Transfer (REST) é um estilo de arquitetura de software que tem por objetivo induzir propriedades como usabilidade, simplicidade, escalabilidade e extensibilidade (Li & Chou, 2011). Desta forma, o modelo de desenvolvimento web moderno utiliza esta arquitetura para o desenvolvimento de interfaces de programação de aplicações (APIs) que permitem a comunicação entre seus serviços, sendo estas denominadas APIs REST (Masse, 2011).

Devido a vantagens como o suporte nativo à linguagem JavaScript, a sua larga adoção e a sintaxe familiar, o formato *JavaScript Object Notation* (JSON) é comummente utilizado para estruturar as trocas de dados em APIs REST. A sintaxe desse formato é basicamente composta por pares de nome/valor, com capacidade de representação de quatro tipos de dados primários (número, texto, booleano, e nulos) e dois tipos de dados estruturados (objetos e vetores) (Fonseca & Simões, 2007).

O Django REST Framework (DRF) permite a criação de APIs REST para aplicações web desenvolvidas com Django, de forma que toma vantagem da simplicidade do desenvolvimento da arquitetura dos modelos para desenvolver a arquitetura das interfaces de programação, com serialização dos dados em JSON, que pode ser definida com flexibilidade ou facilmente a partir de classes pré-definidas.[11] Além disso, a integração do DRF nas aplicações desenvolvidas em Django permite sobretudo a integração com frameworks para desenvolvimento frontend, onde ocorrerá um fluxo de troca de informações com o backend através das APIs.

React:

O framework de código-aberto React foi criado por engenheiros no Facebook de forma a facilitar o desenvolvimento de interfaces do utilizador complexas e com dados dinâmicos. A modularidade é central no desenvolvimento de interfaces a partir do React, de tal forma que a estrutura da aplicação é fundamentalmente constituída por componentes, o que também acaba por influir positivamente em

[11] https://www.django-rest-framework.org/

aspetos como a sua manutenibilidade. Na sua forma mais simples, o React pode ser considerado como a vista numa arquitetura MVC, consequentemente permitindo uma integração mais fluida com frameworks como o Django (Gackenheimer, 2015).

Além de ser baseado em JavaScript, o framework aprsenta o formato JavaScript XML (JSX), que permite a integração da sintaxe do HTML em ficheiros JavaScript, o que promove um desenvolvimento mais ágil e claro de aplicações, que podem ser construídas a partir da integração de diversas bibliotecas, a exemplo de bibliotecas para visualização de modelos BIM. Outro quesito importante deriva da implementação do princípio da modularidade no React, o que beneficia a criação de componentes com pouca interdependência (*loosely coupled),* de forma a favorecer a substituição de componentes dentro da estrutura da aplicação.

Xeokit:

Criado pela Xeolabs, o Xeokit é uma biblioteca JavaScript que permite a criação de visualizadores para modelos BIM. O desenvolvimento com a biblioteca é facilitado tanto pela quantidade de métodos já desenvolvidos que ampliam as capacidades básicas de visualização, a exemplo dos métodos de seleção, omissão e colorização de elementos, como pela sua abrangente documentação. Apesar de ainda não existir implementação uma implementação oficial do Xeokit para o React, o desenvolvimento de um componente de visualização a partir da biblioteca é possível[12].

A biblioteca é gratuita e a sua licença de utilização, Affero GPL v3, aplica o princípio de código aberto do *copyleft,* de tal modo que qualquer software que utiliza esta biblioteca deve ser disponibilizado também com o código aberto. Outra característica do Xeokit é o foco na utilização de um formato de modelo próprio, o XKT. A visualização de modelos IFC também é suportada a partir de um carregador baseado em web-ifc, assim como há a disponibilização, pela própria Xeolabs, de um conversor de IFC para XKT, que pode ser integrado tanto a nível do frontend quanto do backend.

Uma das alternativas mais relevantes ao Xeokit é o visualizador do IFC.js, que possui carregamentos muito mais rápidos para ficheiros IFC, além de possuir licença MIT, o que o torna muito mais permissivo à sua utilização e modificação, já que não impõe a adoção de uma licença aberta para a aplicação criada. Ainda assim, a documentação e a série de métodos disponíveis para este visualizador

[12] https://github.com/xeokit/xeokit-sdk

ainda é pequena se comparada ao Xeokit, o que faz com que o desenvolvimento com este seja mais simplificado.

2.4.5 Programação visual e biblioteca Blockly

Uma linguagem de programação visual tem por princípio permitir a criação de aplicações através da manipulação visual. Normalmente estas linguagens são baseadas nos seguintes elementos:

- Blocos (p.ex. Scratch[13]);
- Fluxos baseados na conexão de nós (p.ex. Dynamo[14]);
- Ícones e representações não-textuais (p.ex. Kodu[15]).

O objetivo de fornecer soluções para o desenvolvimento de linguagens de programação visual e interfaces do utilizador para a programação desses blocos pode ser encontrado em algumas iniciativas já extintas, a exemplo do OpenBlocks[16] e do ScriptBlocks[17]. Blockly é uma biblioteca de código aberto, que surgiu com o intuito de substituir o uso do OpenBlocks no projeto AppInventor. O foco da biblioteca é adicionar programação visual baseada em blocos numa determinada aplicação. Este tipo de programação visual é costumeiramente utilizado para facilitar e agilizar o ensino de linguagens tradicionais de programação. Desta forma, a partir da conexão de blocos, num formato de quebra-cabeças (Figura 19) é possível criar estruturas lógicas com utilização de recursos como operadores condicionais, ciclos, declaração de variáveis, entre outros, que são depois convertidos em scripts em outras linguagens como Python, Javascript, PHP, Lua ou Dart (Adi & Kitagawa, 2019). Também é possível criar blocos customizados e configurar os geradores de código, de forma a expandir o conjunto de blocos disponível para usos mais complexos[18].

[13] https://scratch.mit.edu/
[14] https://dynamobim.org/
[15] https://www.kodugamelab.com/
[16] https://web.mit.edu/mitstep/openblocks.html
[17] https://code.google.com/archive/p/scriptblocks/
[18] https://developers.google.com/blockly/guides/create-custom-blocks/generating-code

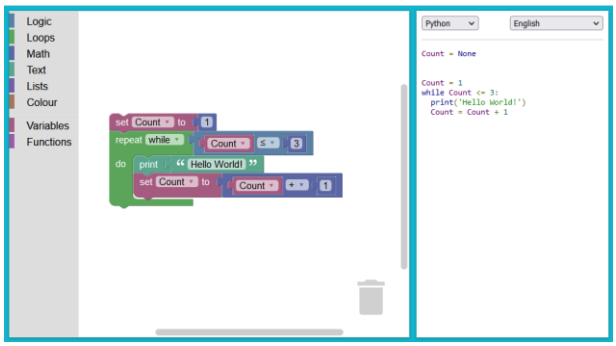

Figura 19: Criador de código baseado em Blockly. Fonte: Developers.google.

3 Interpretação e verificação automática de conformidade

3.1 Enquadramento

As cláusulas dos regulamentos apresentam requisitos atribuídos a uma série de condições aplicáveis. Tanto os requisitos, como as condições aplicáveis, são baseados numa série de conceitos advindos de diversas matérias relacionadas ao processo de licenciamento, a exemplo da arquitetura, engenharia e urbanismo. Além disso, alguns conceitos utilizados são específicos do licenciamento, de forma que, ao se executar uma verificação para aprovação da execução de uma determinada operação urbanística, termos como "alinhamento" ou "altura do edifício" possuem definições próprias e com distinções relevantes às empregadas nas matérias anteriormente mencionadas.

Os modelos BIM, por outro lado, foram criados e desenvolvidos sob um ponto de vista da abrangência de incorporação dos seus conceitos para diversos domínios, de forma a possibilitar a sua utilização para desenvolvimentos de projetos de arquitetura, construção, gestão de ativos, entre outros domínios específicos, através da implementação de camadas especializadas de objetos. Assim, é possível desenvolver modelos que representam, em variados níveis de informação, elementos como paredes, portas, ambientes e áreas.

Desta forma, um dos desafios na implementação de verificações baseadas em modelos BIM para o licenciamento é o mapeamento entre as entidades utilizadas nos regulamentos e os modelos de dados utilizados para representação dos edifícios, principalmente devido à frequente inexistência de uma mapeamento direto. Logo, a abordagem utilizada neste trabalho envolve o desenvolvimento de um modelo de objetos, baseado numa análise de algumas cláusulas de regulamentos portugueses, de forma a executar este mapeamento. Este modelo, desta forma, disponibiliza uma série de classes para a construção de objetos que facilita a execução das verificações de conformidade, já que estas ocorrem a partir da invocação de atributos e métodos que aplicam os conceitos específicos ao domínio do licenciamento.

Outra preocupação deste trabalho foi basear as verificações na utilização do modelo aberto IFC, o que é essencial para uma real implementação de sistemas de verificação a ser utilizados publicamente, já que não obrigam à utilização de nenhuma aplicação específica para a criação dos modelos.

A utilização do IFC, por outro lado, suscitou a tentativa de executar algumas das verificações a partir de um processamento puramente geométrico dos elementos, já que o formato aberto permite editar

alguns parâmetros relativos às quantidades que podem implicar em resultados falsos que não condizem com a geometria do modelo. Esta tentativa acabou por gerar um desafio adicional, que culminou no desenvolvimento de um módulo para processamento da geometria dos modelos.

O trabalho para desenvolvimento dessas ferramentas tem por intuito permitir a execução de verificações, assim como a edição e criação de regulamentos interpretáveis por máquina num ambiente web, o que ocorreu através da implementação de uma plataforma, apresentada no capítulo seguinte. Assim, o trabalho desenvolvido favorece a escalabilidade em alguns níveis, tanto pela definição de um modelo de entidades do licenciamento, que é expansível, assim como pela permissão da criação e edição de regulamentos interpretáveis por máquina. Este cuidado também se deu devido à extensão do corpo regulamentar utilizado no processo de licenciamento português, o que implica na necessidade de programação de muitas regras a ser verificadas.

O processo para o desenvolvimento deste modelo de entidades é apresentado neste capítulo e consistiu nas seguintes etapas:

I. Definição de cláusulas a ser avaliadas;
II. Análise inicial das cláusulas através da metodologia RASE;
III. Obtenção de modelo de entidades do licenciamento;
IV. Criação de ferramentas de processamento geométrico;
V. Validação do modelo de entidades e das ferramentas através da execução de verificações.

Finalmente, há uma discussão sobre os requisitos de informação necessários para a execução das verificações, assim como sobre os desafios da execução de verificações baseadas em processamento das informações geométricas do modelo IFC.

3.2 Criação do modelo de entidades do licenciamento

3.2.1 Seleção de cláusulas

A escolha pela definição de um modelo de entidades, baseado em classes de objetos, para a execução das verificações, foi baseada na diferença entre os conceitos empregados nos regulamentos (p.ex. cozinhas, quartos, varandas) e as entidades existentes no modelo de dados IFC. Esta diferença usualmente impõe a necessidade de um mapeamento entre regulamento e modelo IFC de forma a codificar as verificações.

Entretanto, a definição de um modelo de entidades dentro da lógica de um desenvolvimento orientado a objetos, permite implementar o mapemaneto de forma encapsulada em classes reutilizáveis, o que simplifica a instanciação de objetos criados a partir destas classes e a execução das operações necessárias à verificação. Além disso, a centralização da definição das correlações em classes facilita a expansão do modelo de entidades e consequentemente torna o método de verificação mais escalável.

O modelo de entidades, portanto, deve ser definido a partir da análise dos regulamentos a fim de obter os conceitos necessários para execução da verificação. Ao considerar os objetivos e âmbito deste trabalho, optou-se pela análise de algumas cláusulas em documentos que fossem representativos de alguns tipos de verificação encontrados no corpo regulamentar, de tal forma que os documentos escolhidos foram o RGEU, devido à sua abrangência nacional e por possuir verificações relativas a dimensões e áreas do edificado, assim como os regulamentos e planos diretores municipais de Vila Nova de Gaia e de Lisboa, que possuem foco em verificações no domínio do urbanismo.

A escolha das regras analisadas também foi pautada no seu potencial de codificação e na capacidade de obter as informações a ser verificadas maioritariamente de um modelo de edifício, já que alguns parâmetros estabelecidos, principalmente nos planos de ordenamento do território, necessitam de informação do espaço urbano, o que envolve a análise de modelos de informação relativos a Sistemas de Informação Geográfica (SIG), o que não está enquadrado no âmbito deste trabalho. Entretanto, de forma a avaliar a capacidade de verificar parâmetros urbanísticos através de modelos de edifícios, principalmente devido à importância destes parâmetros em comparação às questões internas dos edifícios, que normalmente não são o foco da verificação nas câmaras, algumas cláusulas neste âmbito foram escolhidas. Além disso, foram privilegiadas cláusulas que dispusessem de requisitos explícitos e que pudessem ser convertidos em parâmetros codificáveis.

Desta forma, a seguir são apresentadas as cláusulas analisadas neste trabalho, assim como comentários que contêm informações gerais sobre a motivação para a escolha de cada uma:

RGEU - Artigo 46.º, § 7: "*Os degraus das escadas das edificações para habitação colectiva terão a largura (cobertor) mínima de 0,25 m e a altura (espelho) máxima de 0,193 m. No entanto, nos edifícios de três, quatro ou cinco pisos e sempre que não seja instalado ascensor, a largura (cobertor) mínima será de 0,280m e a altura (espelho) máxima será de 0,175m.*"

A escolha desta cláusula foi determinada por tratar-se de uma verificação relativa às dimensões de elementos internos à construção.

RGEU - Artigo 65.º, § 1: "*A altura mínima, piso a piso, em edificações destinadas à habitação é de 2,70m (27m), não podendo ser o pé-direito livre mínimo inferior a 2,40 m (24m).*"

RGEU - Artigo 65.º, § 2: "*Excepcionalmente, em vestíbulos, corredores, instalações sanitárias, despensas e arrecadações será admissível que o pé-direito se reduza ao mínimo de 2,20m (22m).*"

Esta cláusula foi escolhida por lidar com parâmetros de altura que são internos à edificação.

RGEU - Artigo 66.º, § 1: "*Os compartimentos de habitação não poderão ser em número e área inferiores aos indicados no quadro seguinte:*

	número de compartimentos por fogo							
	2 T0	3 T1	4 T2	5 T3	6 T4	7 T5	8 T6	Mais de 8 Tx>6
	áreas em metros quadrados							
Quarto casal	___	10,5	10,5	10,5	10,5	10,5	10,5	10,5
Quarto duplo	___	___	9	9	9	9	9	
Quarto duplo	___	___	___	9	9	9	9	
Quarto duplo	___	___	___	___	___	9	9	restantes quartos 9m2
Quarto simples	___	___	___	___	6,5	6,5	6,5	6,5
Quarto simples	___	___	___	___	___	___	6,5	6,5
Sala	10	10	12	12	12	16	16	16
Cozinha	6	6	6	6	6	6	6	6
Suplemento de área obrigatório.....	6	4	6	8	8	8	10	(x + 4)m2 (x= n.º de quartos)

"

Apesar das questões internas à edificação não serem usualmente verificadas, esta cláusula foi escolhida pela oportunidade de demonstrar as capacidades da verificação não somente a nível do município, mas ao nível dos procedimentos antes da submissão dos projetos, o que pode ser benéfico para os projetistas.

RGEU - Artigo 67.º, § 1: "*As áreas brutas dos fogos terão os seguintes valores mínimos:*

	Tipo de fogo							
	T0	T1	T2	T3	T4	T5	T6	Tx>6
área bruta em metros quadrados	35	52	72	91	105	122	134	1,6 x Ah

"

A escolha desta cláusula foi determinada por lidar com verificações de dimensões e o conceito do fogo, utilizado no âmbito do licenciamento, e que pode se refletir em particularidades de implementação interessantes de discutir e avaliar neste trabalho.

PDM de Vila Nova de Gaia - Artigo 24.º, § 1, a): "As edificações devem ter uma cércea máxima 2 pisos e uma altura máxima de 6,5m, salvo instalações técnicas devidamente justificadas"

Apesar de ser relativa a instalações adstritas a explorações agrícolas, pecuárias e florestais, a cláusula foi escolhida por demonstrar uma das formas com a qual os requisitos de altura podem ser apresentados e, consequentemente, verificados.

PDM de Vila Nova de Gaia - Artigo 42.º, § 1: "*As novas construções principais implantar-se-ão dentro de uma faixa de 35m, confinante com o espaço público, sem prejuízo do previsto nos números seguintes."*

A escolha desta cláusula ocorreu por tratar-se de uma questão relativa ao posicionamento do edifício na parcela, o que é uma característica relevante ao analisar a integração do edifício no ambiente urbano.

RPDM - Artigo 42.º, § 7, e): "*Índice de edificabilidade, em parcelas com uma profundidade superior a 14 metros e/ou com uma área de lote ou parcela superior a 130m2:*

i) 1,0 em lote ou parcela com área inferior a 150m2;

ii) ii) 0,7 em lote ou parcela com área igual ou superior a 150m2, sendo sempre permitido um mínimo de 150m2 de superfície de pavimento."

A escolha desta cláusula foi determinada pela sua relevância e relação com questões urbanísticas.

RPDML - Artigo 43.º, § 1: "*Sem prejuízo do disposto nos números seguintes, a profundidade máxima das empenas, sem considerar as varandas e os corpos balançados, é de 15 metros, com exceção dos estabelecimentos hoteleiros e equipamentos de utilização coletiva, cuja empena pode atingir os 18 metros.*"

Esta cláusula foi escolhida por tratar de fator importante na integração urbanística.

3.2.2 Análise de cláusulas e obtenção de modelo conceptual

O processo de desenvolvimento do modelo de entidades partiu preliminarmente da implementação da metodologia RASE para as cláusulas escolhidas, isto possibilitou o levantamento inicial dos objetos e propriedades necessárias para a execução das verificações e a sistematização dos valores utilizados. Desta forma, o primeiro passo da implementação da metodologia partiu da marcação dos textos originais das cláusulas, a fim de obter as frases métricas, o que pode ser verificado na Tabela 1, que apresenta a marcação para uma das cláusulas analisadas. Além disso, a marcação executada para as demais cláusulas pode ser verificado no Anexo I.

Tabela 1: Marcação através da metodologia RASE.

Cláusula	**Texto**
RGEU, Artigo 46.º 7	<R>Os <a>degraus das escadas das edificações para <s>habitação colectiva </s>terão a <r>largura (cobertor) mínima de 0,25 m </r>e a <r>altura (espelho) máxima de 0,193 m</r>. <E>No entanto, <s>nos edifícios de três, quatro ou cinco pisos</s> e <s>sempre que não seja instalado ascensor</s>, a <r>largura (cobertor) mínima será de 0,280m</r> e a <r>altura (espelho) máxima será de 0,175m</r>.</E></R>

Após o processo de marcação, ocorreu o processo de sistematização das frases métricas, que é exemplificado na Tabela 2 e pode ser encontrado para as demais cláusulas no Anexo II. O principal objetivo deste processo foi extrair elementos importantes para a composição de um modelo conceptual, como objetos e propriedades. Entretanto, a metodologia possui algumas limitações, principalmente em situações onde os elementos não estão explícitos no texto, o que envolve frequentemente a necessidade de recorrer a outras partes do regulamento. Além disso, a metodologia também possui limitações para tratar situações mais complexas, a exemplo de situações em que os valores são outros objetos, o que ocorre no caso de distância entre elementos, ou propriedades de outros objetos. Desta forma, frequentemente é necessário fazer adaptações na sistematização ou promover análises posteriores para colmatar as deficiências dos resultados.

Tabela 2: Sistematização das frases métricas obtidas através da marcação RASE.

Frase métrica	**Tipo**	**Objeto**	**Propriedade**	**Comparação**	**Valor**	**Un**
degraus das escadas das edificações	aplicabilidade	escada	tipo	=	True	
habitação	seleção	edifício	categoria	=	habitação	

colectiva					coletiva	
largura (cobertor) mínima de 0,25 m	requisito	escada	largura do degrau	>=	0,25	m
altura (espelho) máxima de 0,193 m	requisito	escada	altura do degrau	<=	0,193	m
nos edifícios de três, quatro ou cinco pisos	seleção	edifício	número de pisos	=	3 4 5	
sempre que não seja instalado ascensor	seleção	ascensor	tipo	=	False	
largura (cobertor) mínima será de 0,280m	requisito	escada	largura do degrau	>=	0,28	m
altura (espelho) máxima será de 0,175m	requisito	escada	altura do degrau	<=	0,175	m

A partir das informações obtidas, foi possível estabelecer uma versão conceptual do modelo de entidades a ser utilizado (Figura 20), modelado num diagrama de classes UML. Isto permitiu esboçar os primeiros requisitos de informação e estabelecer de forma concreta as relações entre as entidades necessárias a ser implementadas.

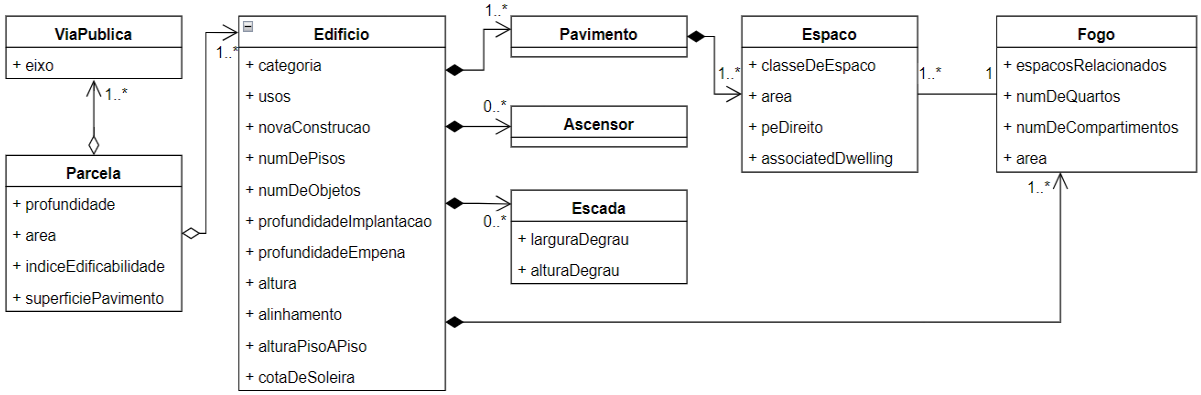

Figura 20: Modelo conceptual de entidades desenvolvido.

A partir da implementação do modelo de conceptual numa série de classes em Python, foi possível determinar questões cruciais, nomeadamente a sintaxe a desenvolver para executar as verificações, o que foi ajudado por uma implementação tipificada (Figura 21), que permite a o acesso a todos os atributos e classes a diversos níveis. O processo de desenvolvimento destas verificações foi pautado,

não somente nas informações sistematizadas levantadas a partir da marcação RASE, mas também da análise do texto original, já que um dos objetivos é permitir que o usuário possa criar novas verificações sem a necessidade de fazer o processo de interpretação como o feito para o desenvolvimento do modelo conceptual.

```
class Building(PermitObject):

    def __init__(self,data,model):

        super().__init__(data,model)

        self.category: Optional[object] = self.get_property("PermitCheck","Category")
        self.uses: Optional[object] = self.get_property("PermitCheck","Uses")
        self.isNewConstruction: Optional[object] = self.get_property("PermitCheck","IsNewConstruction")
        self.elevators: List[Elevator] = self.get_connection('IfcTransportElement[PredefinedType = "ELEVATOR"]', Elevator)
        self.buildingStoreys: List[BuildingStorey] = self.get_connection('IfcBuildingStorey', BuildingStorey)
        self.stairs: List[Stair] = self.get_connection('IfcStair', Stair)
        self.building_mesh = self.get_building_mesh()
        self.thresholdLevel = self.model.select('.IfcBuildingStorey[classification.Identification="En_95_05"]')[0].Elevation
```

Figura 21: Implementação das classes com recurso a tipos.

A implementação também demonstrou a necessidade da criação de classes específicas para o tratamento dos dados que seriam carregados no modelo. Assim, foi criada uma classe “PermitObject”, a ser herdada por todos os elementos, e que estabelece métodos básicos para associação de entidades do licenciamento e o modelo IFC, além dar acesso direto a este por cada entidade do modelo, o que permite executar consultas ao IFC diretamente pela entidade do licenciamento. Outra classe criada foi a “PermitModel”, de onde deriva a criação de todos os objetos baseados nas classes do modelo de entidades.

3.2.3 Mapeamento entre modelo de entidades do licenciamento e IFC

Após o desenvolvimento e implementação do modelo de entidades, foi necessário estabelecer o mapeamento com o modelo de dados IFC. Esta etapa necessitou de alguns cuidados, de forma a manter as relações de pertença, existente em elementos como pavimentos e ambientes, assim como de perceber quais informações poderiam vir diretamente de propriedades alfanuméricas e quais teriam de ser derivadas com processamento de geometria ou com a implementação de métodos.

Como mencionado anteriormente, com frequência o mapeamento não ocorre diretamente. Assim, em alguns casos o mapeamento foi simples, através da definição de atributos como “classeDeEspaço”, que permite determinar uma relação entre espaços do IFC e diferentes categorias de espaço existentes em regulamentos, p.ex. cozinhas e quartos duplos. Para outros casos, porém, o mapeamento não pôde

ser desenvolvido desta forma, de tal maneira que gerou a necessidade de avaliar a conversão de alguns destes atributos para métodos mais complexos a ser desenvolvidos nas respetivas classes.

A determinação de classes foi utilizada como recurso não somente para a distinção entre os compartimentos da habitação, mas também para a definição de alguns elementos que são específicos do processo de licenciamento, como parcelas e fogos. Neste âmbito, houve o cuidado de avaliar a capacidade de utilização do sistema SECClasS para a classificação dos objetos do IFC. Assim, todas as classes necessárias para a execução das verificações foram obtidas a fim de avaliar uma possível associação com as classes existentes. Em alguns casos, não havia classes que fossem adequadas às entidades requeridas. Nestes casos, novas classes foram criadas na tentativa de colmatar esta deficiência, porém, já que esta avaliação não está entre os objetos centrais deste trabalho, uma efetiva proposta de classes como estas no sistema deve ser avaliada com mais cuidado.

A Tabela 3 apresenta o mapeamento entre as classes obtidas para a execução das verificações segundo o modelo implementado e o sistema SECClasS. Devido à inexistência de classes no sistema que fossem adequadas para representar as entidades do licenciamento, algumas novas classes tiveram que ser propostas no âmbito deste trabalho e estão apresentadas na terceira coluna da tabela.

Tabela 3: Mapeamento entre classes exigidas para verificação e classes SECClasS.

Entidades	**SECClasS**	**Classes propostas**
Vestíbulo	SL_40_65_94 - Vestíbulos	–
Corredor	SL_90_10_36 - Corredores	–
Instalação sanitária	SL_35_80 – Espaços sanitários	–
Despensa	SL_90_50_46 – Copas e despensas de alimentos	–
Arrecadação	–	SL_90_50_39 - Arrecadações
Sala	SL_45_10_49 – Salas de estar ; SL_45_10_22 – Salas de jantar domésticas	–
Cozinha	SL_45_10_23 – Cozinhas domésticas	–
Quarto casal	–	SL_45_10_10 – Quarto casal

Quarto duplo	–	SL_45_10_11 – Quarto duplo
Quarto simples	–	SL_45_10_07 – Quarto simples
Varanda	SL_45_10_06 - Varandas	
Parcela Cadastral	–	SL_22_05 – Parcela Cadastral
Via Pública	–	SL_22_10 – Via Pública
Arruamento	–	SL_22_20_05 – Arruamento
Fogo	–	SL_22_15 – Fogo
Cota de soleira	–	En_95_05

A determinação destas classes foi, então, um dos passos necessários na atribuição das correlações, de onde derivam os requisitos de modelação necessários para executar as verificações. Um aspeto importante considerado no mapeamento foi a exequibilidade dos requisitos estabelecidos. Isto implica não somente avaliar a estrutura de dados do IFC, mas também a capacidade dos softwares disponíveis para a exportação de modelos IFC com as informações requeridas.

Desta forma, esta avaliação foi centralizada nas capacidades do modelo IFC de representar as entidades necessárias e das capacidades de exportação para IFC dos softwares de modelagem mais utilizados. Estes fatores demonstraram empecilhos relevantes para o estabelecimento de um modelo IFC adequado à verificação. Isto pode ser exemplificado pela capacidade do IFC de representar agrupamentos de estruturas espaciais como “IfcSpaces” e “IfcBuildingStoreys” consoante o atributo “CompositionType”, o que não é possível de executar através dos softwares de modelagem mais populares. Algumas destas capacidades, portanto, poderiam ser de grande valia para a criação de modelos para licenciamento, de forma a facilitar a modelagem de elementos como os fogos de um edifício e seus respetivos compartimentos.

A possibilidade de criação do IFC através de recursos como o BlenderBIM foi avaliada, devido à relativa complexidade de modelação no estado atual deste software e devido à sua pouca adoção, os requisitos para os casos avaliados seriam pouco acessíveis, além de direcionar à necessidade de utilização de um software específico, o que é incompatível com os princípios da Administração Pública. Desta forma, os requisitos obtidos refletem algumas adequações necessárias devido a estas limitações, sendo este um interessante tema a ser investigado.

Uma tarefa crucial no estabelecimento das correlações foi a coleta e análise das definições de alguns elementos componentes do modelo de entidades, o que envolveu uma pesquisa aos regulamentos e a materiais externos para a obtenção de informações que pudessem ser suficientes para a atribuição de uma correta associação ou para a criação de métodos que fossem devidamente representativos das entidades ou propriedades a ser implementadas. Esta tarefa apresentou um considerável desafio, principalmente devido à tentativa de obter algumas das informações do modelo de entidades a partir de processamento geométrico.

Para exemplificar esta questão, podemos avaliar a simples obtenção da profundidade de elementos, a exemplo da profundidade de uma parcela ou de um edifício. As dimensões de um paralelepípedo normalmente são denominadas por comprimento, largura e altura, o que permite facilmente obtê-las se considerarmos que suas faces estão normais ou paralelas aos eixos. Entretanto, o termo profundidade é comumente empregado para representar, em situações reais, uma das dimensões de objetos paralelepipédicos que possuem uma frente determinada, sendo então a dimensão perpendicular ao plano estabelecido pela face frontal do objeto. Assim, no momento de determinar a profundidade de um determinado objeto, é preciso primeiro determinar a sua frente. Isto foi tratado pela criação de uma função para cálculo da profundidade de objetos a partir de sua *bounding box* orientada e da determinação de outro objeto que é utilizado para determinar a frente, que pode ser o arruamento fronteiro, no caso de parcelas e edifícios. O algoritmo implementado nesta função pode ser verificado na Figura 22.

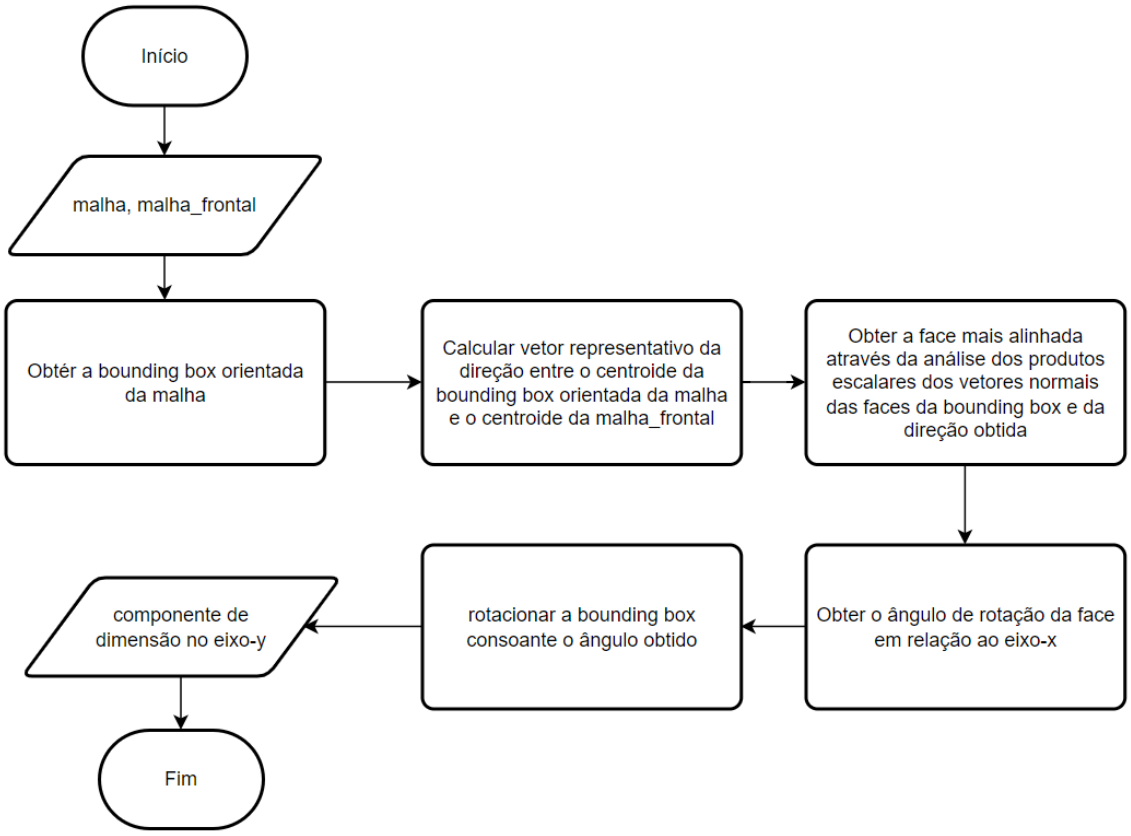

Figura 22: Algoritmo para o cálculo da profundidade de um objeto.

Desta forma, a criação de diversas outras funções foi necessária para implementar não somente conceitos básicos, mas também elementos que possuem definições mais complexas e próprias do licenciamento, como o índice de edificabilidade. Este processo permitiu então estabelecer quais seriam as propriedades extraídas do IFC, o que implicou na diminuição da necessidade de criação de propriedades de projeto. Além disso, algumas propriedades do modelo de entidades, como "numDePisos", também puderam ser facilmente substituídas pela utilização de métodos nativos do Python, como len().

A Tabela 4 apresenta o mapeamento das entidades e propriedades do modelo de entidades com o modelo IFC, onde "PP" são designações das propriedades de projeto que precisam ser criadas, enquanto "NP" são propriedades nativas da estrutura do IFC. Por outro lado, as propriedades obtidas através de métodos são apresentadas na Tabela 5, assim como na discussão dos exemplos utilizados para a validação das classes criadas.

Tabela 4: Mapeamento entre modelo de entidades e IFC.

Modelo de entidades	**Modelo IFC**
Parcela	IfcSpace com classe “SL_22_05”
ViaPublica	IfcSpace com classe “SL_22_10”
Edifício	IfcBuilding
+ categoria	PP: Category
+ usos	PP: Uses
+ novaConstrução	PP: IsNewConstruction
+ cotaSoleira	Propriedade Elevation do IfcBuildingStorey com classe “En_95_05”
Ascensor	IfcTransportElement.ELEVATOR
Escada	IfcStair
+ larguraDegrau	NP: TreadLength
+ alturaDegrau	NP: RiserHeight
Pavimento	IfcBuildingStorey
Fogo	IfcSpace com classe “SL_22_15”
+ referência	PP: DwellingReference
Espaco	IfcSpace
+ classeDeEspaco	Classificação
+ fogoAssociado	PP: AssociatedDwelling

Os resultados obtidos para o mapeamento, com consideração das cláusulas analisadas, não se convertem em requisitos robustos, com a necessidade de criação de apenas 5 propriedades de projeto. Isto deve-se maioritariamente à utilização do processamento geométrico, que permitiu calcular diversas propriedades do modelo de entidades. A Tabela 5 apresenta as propriedades que foram derivadas e qual foi a abordagem utilizada para o seu cálculo, maioritariamente apoiado nas definições consultadas em documentos. O cálculo executado para algumas propriedades também foi pautado em sua facilidade de execução, de forma que ainda precisam ser aprimorados para uma implementação real.

Tabela 5: Descrição da abordagem utilizada para obter métodos.

Parcela	
+ profundidade	A profundidade de elementos, como definida anteriormente, foi definida a partir da profundidade da bounding box orientada do elemento com definição do arruamento como malha frontal.
+ area	Como a área da parcela foi determinada a partir de um IfcSpace, a área foi estabelecida como a área da seção transversal na altura do seu centroide.
+ indiceEdificabilidade	O índice de edificabilidade foi obtido a partir do cálculo da soma da superfície de pavimento para todos os pavimentos do edifício sobre a área da parcela.
+ superficePavimento	A superfície de pavimento foi obtida a partir da soma da área bruta de todos os pavimentos.
ViaPublica	
+ eixo	Por facilidade de cálculo, o eixo da geometria foi definido a partir do seu centroide, de forma que os cálculos da distância até este ponto tiveram ortogonalidade forçada.
Edifício	
+ numDePisos	Estas três propriedades foram obtidas através da contagem de elementos da respetiva classe através do método len()
+ numDeHabitacoes	
+ numDeObjetos	
+ profundidadeImplantacao	A profundidade de implantação foi considerada como a distância ortogonal máxima entre o alinhamento e os elementos do edifício.
+ profundidadeEmpena	A profundidade da empena foi calculada como a profundidade da boundig box orientada das paredes externas do edifício, de forma a desconsiderar as varandas e corpos balançados.
+ altura	A altura foi considerada como a distância entre a maior coordenada entre todos os elementos contidos no IfcBuilding, à exceção de IfcChimney, e o atributo cota de soleira.
+ alinhamento	O alinhamento foi considerado como a linha formada pelos pontos mais próximos entre a projeção da parcela na maior elevação da vía pública e a

	projeção da via pública.
+ alturaPisoAPiso	A altura piso a piso foi considerada como a distância mínima entre as elevações dos pisos, numa interação entre pavimento n e n+1 feita numa lista dos pavimentos ordenada por sua elevação.
Pavimento	
+ areaBruta	A área bruta do pavimento foi considerada como a área de todos os espaços de um pavimento, mais as áreas das projeções das paredes internas e externas ao nível do piso, sem considerar as áreas das varandas.
Fogo	
+ numDeQuartos	Estas duas propriedades foram obtidas através da contagem de elementos da respetiva classe através do método len()
+ numDeCompartimentos	
+ area	A área do fogo foi considerada como a área da seção transversal do espaço considerado como fogo, na altura do plano definido pelo seu centroide.
Espaco	
+ area	A área do espaço foi considerada como a área da seção transversal do espaço considerado como fogo, na altura do plano definido pelo seu centroide.
+ peDireito	O pé direito foi considerado como a altura do espaço modelado.

3.3 Verificação de conformidades a partir de modelos IFC

3.3.1 Criação de ferramentas para processamento geométrico

O processamento geométrico a partir de ficheiros IFC, com a utilização do IfcOpenShell, é um processo relativamente complexo, que recorre à criação de geometrias consoante as configurações determinadas pelo usuário. Sua configuração de base permite gerar objetos B-Rep relacionados à biblioteca OpenCascade ou converter praticamente todas as representações geométricas do IFC em malhas triangulares, o que permite aplicar algoritmos e bibliotecas de processamento de malhas para o processamento da geometria.

O processo de verificação de conformidade mais simples usualmente parte da obtenção das informações do modelo do edifício, existentes em grupos de propriedades e quantidades para a comparação com valores provisionais. Entretanto, as provisões regulamentares quase sempre são mais

complexas do que isto, o que requer a capacidade de processamento geométrico para a obtenção das informações necessárias. Outras vantagens da implementação de capacidades de processamento geométrico são: dificultar a burla através da edição de quantidades em ficheiros IFC; diminuição da quantidade de propriedades exigidas nos requisitos de informação.

Outra capacidade importante é a de consultar as informações do IFC. O IfcOpenShell, em sua forma mais básica permite a obtenção de elementos através de seu tipo e possui alguns módulos com funções para a obtenção de grupos de propriedades e quantidades. Porém, a utilização da classe Selector, para a realização de consultas com recurso à biblioteca Lark, proporciona uma capacidade de realizar consultas muito mais complexas e flexíveis, o que favorece muito a capacidade de executar verificações com uma sintaxe mais simples.

Desta forma, uma das tarefas executadas neste trabalho foi a de simplificar a execução de verificações a partir da consulta a ficheiros IFC e de processamento geométrico a partir das malhas através da integração das ferramentas de consulta e do processamento geométrico. Para isso, foram criadas ferramentas baseadas em duas classes: a primeira, denominada “CheckModel”, tem por objetivo centralizar o carregamento de ficheiros, carregamento de geometria e consulta a ficheiros num único objeto; a segunda, “CheckElement”, tem por objetivo convergir as propriedades dos elementos do IfcOpenShell com as informações de malhas geométricas carregadas num objeto Trimesh. A comparação entre a execução convencional para processamento geométrico e a versão com a implementação proposta pode ser vista na Figura 23, onde é possível observar que a versão com a implementação proposta é mais concisa e simples para obter informações.

Um aspeto importante considerado na criação destas classes foi a centralização da criação da geometria. Isto possibilitou melhorias no desempenho derivadas da utilização do “GeometryIterator” do IfcOpenShell, que permite a utilização dos recursos de multiprocessamento do Python, além de colmatar os problemas advindos da criação de geometria de forma repetida a cada necessidade de processamento da malha de um elemento específico.

```python
import ifcopenshell
import ifcopenshell.geom
import ifcopenshell.util.shape


# carregamento do ficheiro IFC
ficheiro_ifc = ifcopenshell.open('model.ifc')

# seleção de um espaço com nome "Quarto-01"
espacos = ifc_file.by_type('IfcSpace')

quarto = None

for espaco in espacos:
    if espaco.Name == "Quarto-01":
        quarto = espaco

# Determinação das configurações para criação de geometria
settings = ifcopenshell.geom.settings()

# Criação da geometria
shape = ifcopenshell.geom.create_shape(settings, quarto)

# Obtenção do volume
volume = ifcopenshell.util.shape.get_volume(shape)
```

```python
from model.apiv4 import *

# Carregamento do ficheiro IFC
meu_modelo = CheckModel('model.ifc')

# Seleção do quarto
quarto = meu_modelo.select('.IfcSpace[Name="Quarto-01"]')[0]

# Obtenção do volume
volume = quarto.mesh.volume
```

Figura 23: À esquerda: forma tradicional para processamento geométrico. À direita: forma proposta.

Da centralização do processo de criação de geometria foi gerado um dicionário com as malhas criadas, com chaves relativas ao id global dos elementos, desta forma a integração com o mecanismo de seleção permite fornecer não somente os elementos IFC que se adequam à consulta, mas também suas malhas, para a execução do processamento geométrico, que podem ser acedidos através do atributo “mesh”. Assim, os elementos gerados na consulta possuem todos os métodos de um elemento IFC, assim como todos os métodos para processamento de geometria disponibilizados pelo Trimesh. Este processo de seleção pode ser verificado na Figura 24

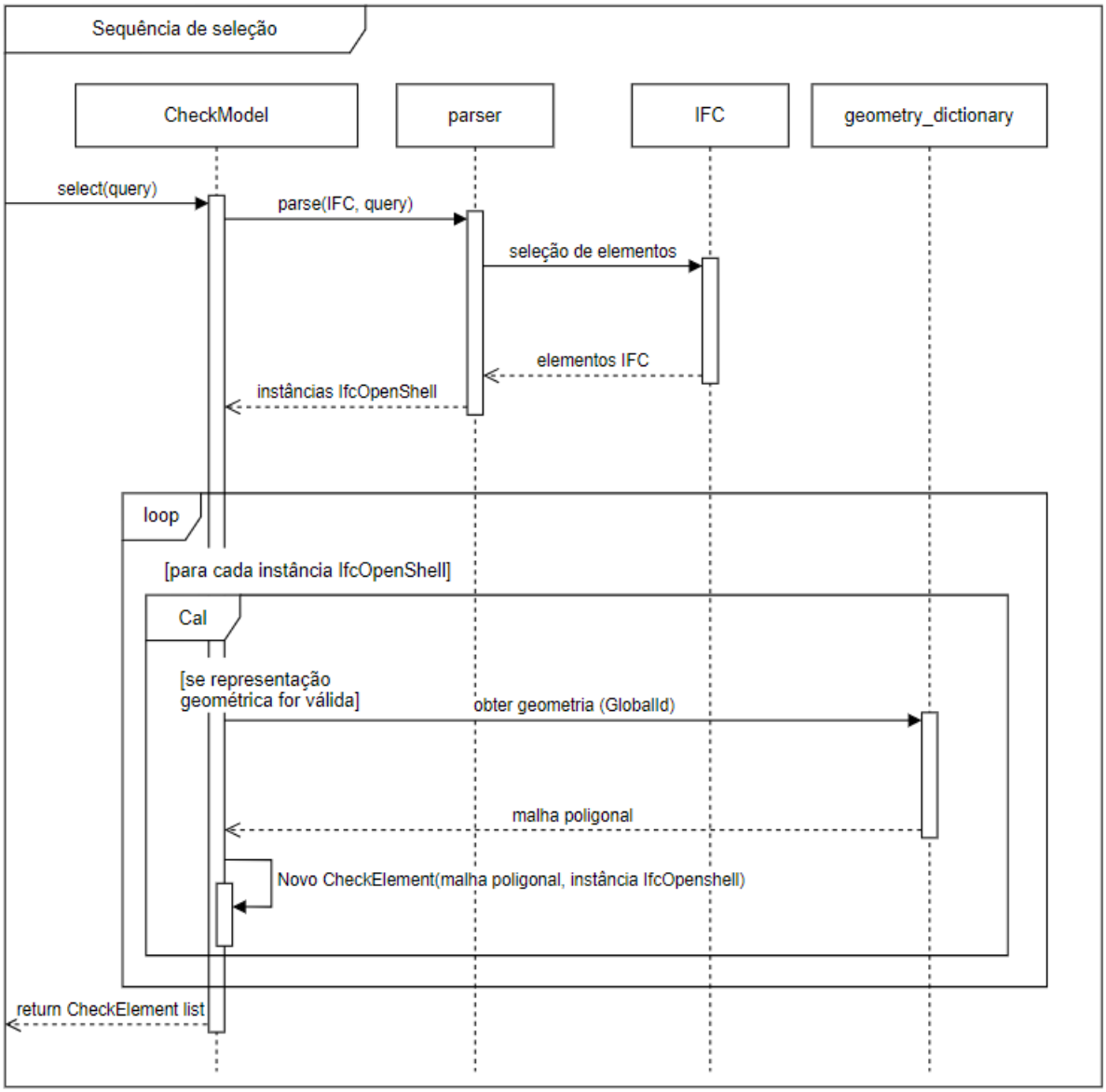

Figura 24: Sequência de consulta e criação dos objetos “CheckElement”

3.3.2 Implementação das verificações em Python

A fim de validar o modelo de entidades do licenciamento desenvolvido, as cláusulas analisadas foram implementadas em código Python e as verificações executadas a partir de um modelo IFC de teste, que possui representação de fogos com diferentes características, com base nos requisitos de informação definidos. O modelo IFC foi desenvolvido no Revit, versão 2024 e exportado para o formato IFC com o exportador nativo de versão mais atualizada à época do desenvolvimento deste trabalho. Para a classificação dos objetos do no Revit, foi utilizado o Classification Manager do Revit Interoperability

tools, a partir da tabela do SECClasS modificada com as novas classes criadas, o que facilitou o processo de modelagem.

O edifício modelado (Figura 25) possui 4 pavimentos, sendo um abaixo do nível do solo e 3 acima. Em cada um dos pavimentos acima do solo há um apartamento com diferentes características, sendo dois T1 e um T0.

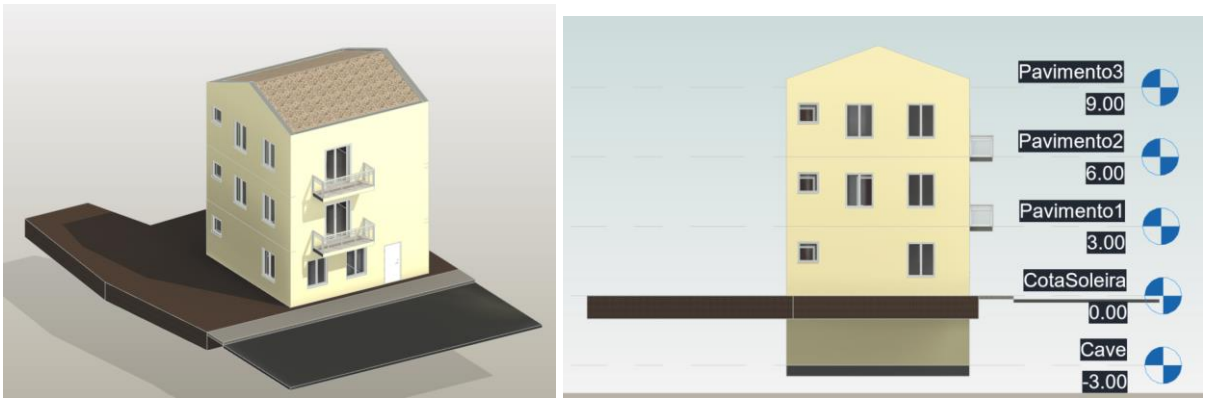

Figura 25: Modelo simples desenvolvido para a validação das verificações.

As diferentes características dos apartamentos podem ser vistas nas Figura 26 e 27, de forma que, apesar de possuírem a mesma área bruta (desconsiderando o valor da área da varanda), possuem diferentes compartimentações e áreas para a mesma classe de espaço.

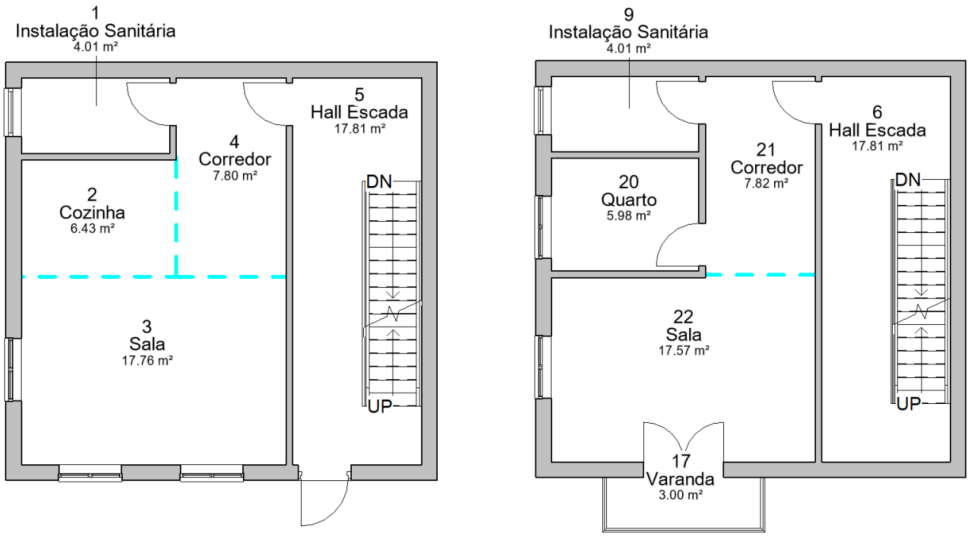

Figura 26: Apartamentos do rés-do-chão e do pavimento 1.

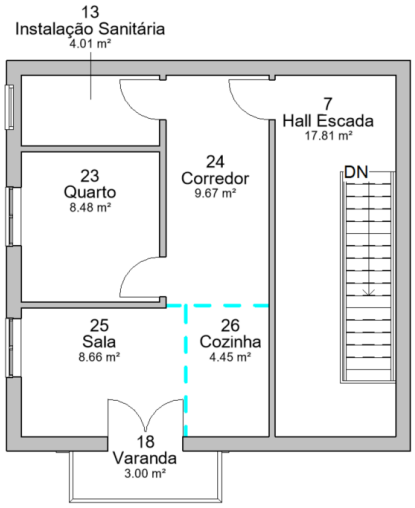

Figura 27: Apartamento do pavimento 2.

De forma a validar os resultados obtidos com as classes e métodos criados, alguns valores foram obtidos nos códigos de verificação e comparados com os valores obtidos através do visualizador de ficheiros IFC BIMVision, onde é possível executar medições de áreas, distâncias e volumes, além da seleção e filtragem de múltiplos itens.

Os códigos de três verificações referentes às cláusulas analisadas podem ser vistos a seguir, seguidos da validação e avaliação dos resultados obtidos.

3.3.2.1 Regulamento Geral das Edificações Urbanas, Artigo 67.º, parágrafo 1

O primeiro exemplo é relativo às áreas mínimas dos fogos consoante o seu tipo. De um modo geral, a execução das verificações tomou como princípio a iteração através dos elementos a ser verificados. Desta forma, foi preciso iterar entre todos os edifícios em todas as parcelas de forma a verificar todos os fogos. Como pode ser visto na Figura 28, primeiro foi identificado o número de quartos para que seja possível determinar qual o tipo de fogo, o que determina qual a área mínima requerida. Após a identificação, ocorre uma série de condicionantes "se" para verificar corretamente a área do fogo.

Neste primeiro exemplo, também é possível perceber que apesar da sintaxe da linguagem Python, os objetos referenciados são os do modelo de entidades, o que implica numa escrita com um nível superior de abstração e consequentemente facilita a escrita do código. Para a validação, foram introduzidos alguns códigos para impressão dos valores e de informações textuais para designar o fogo que estava a ser avaliado, assim como o número de quartos calculado.

```
# ------------------------------------------------------------------
# RGEU, Artigo 67.º 1
# ------------------------------------------------------------------

for parcel in my_permit.parcels:
    for building in parcel.buildings:

        for dwelling in building.dwellings:

            num_of_bedrooms = 0

            for space in dwelling.relatedSpaces:

                if space.objectClass in [room_types["quartoCasal"],room_types
                ["quartoSimples"],room_types["quartoDuplo"]]:

                    num_of_bedrooms +=1

            if num_of_bedrooms == 0:

                print(dwelling.gross_area() >= 35)

            if num_of_bedrooms == 1:

                print(dwelling.gross_area() >= 52)
```

Figura 28: Código para verificação do artigo 67.°, parágrafo 1, do RGEU, com modelo de entidades implementado.

Os resultados obtidos na verificação em comparação com a verificação visual auxiliada pelo software BIMVision podem ser verificadas na Figura 29. É possível ver que o cálculo executado corresponde ao valor identificado pelo software, entretanto, a automatização do processo permitiu obter os valores de forma muito mais rápida.

```
################################################
O fogo avaliado: AP1
Número de quartos calculado: 0
Área do fogo: 42.78
Área requerida: 35 m2
O fogo cumpre os requisitos
################################################
O fogo avaliado: AP3
Número de quartos calculado: 1
Área do fogo: 42.78
Área requerida: 52 m2
O fogo não cumpre os requisitos
################################################
O fogo avaliado: AP2
Número de quartos calculado: 1
Área do fogo: 42.78
Área requerida: 52 m2
O fogo não cumpre os requisitos
```

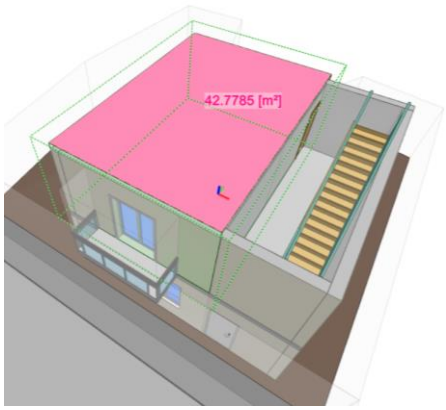

Figura 29: À esquerda: resultado da verificação. À direita: valores obtidos com o BIMVision.

3.3.2.2 Regulamento do Plano Diretor Municipal de Lisboa, Artigo 43.°, parágrafo 1

O segundo exemplo escolhido trata da profundidade de empena do edifício, que é um parâmetro importante na verificação da integração do edifício com o espaço urbano. Esta verificação também mostra a avaliação da aplicabilidade a partir do atributo "categoria", de forma que edifícios que possuam categorias diferentes de "estabelecimento hoteleiro" e "equipamento coletivo" devem possuir profundidade de empena menor ou igual a 15 metros. Outro ponto importante de destacar nesta verificação é o fato da profundidade estar relativa a um referencial, que neste caso é determinado como a rua da frente, isso permite que independente do norte verdadeiro, ou do edifício ser de gaveto, com fachada para dois arruamentos, ele sempre seja avaliado da forma correta. O código para a verificação desta regra está apresentado na Figura 30.

```
# ------------------------------------------------------------------
# RPDML Artigo 43.º 1
# ------------------------------------------------------------------

'''Sem prejuízo do disposto nos números seguintes, a profundidade máxima das
empenas, sem considerar as varandas e os corpos balançados, é de 15 metros, com
exceção dos estabelecimentos hoteleiros e equipamentos de utilização coletiva,
cuja empena pode atingir os 18 metros.'''

for parcel in my_permit.parcels:
    for building in parcel.buildings:
        if building.category in ["estabelecimento hoteleiro", "equipamento
        coletivo"]:
            print(building.depth(parcel.frontStreet) <= 18)
        else:
            print(building.depth(parcel.frontStreet) <= 15)
```

Figura 30: Código para verificação do artigo 43.°, parágrafo 1, do RPDML, com modelo de entidades implementado.

Os resultados para esta verificação, assim como os valores para comparação obtidos no BIMVision, a partir do cálculo da distância entre o plano da parede da fachada frontal e o plano da parede da fachada tardoz, podem ser vistos na Figura 31.

```
##################################################
Categoria do edifício: habitação coletiva
Profundidade de empena: 7.9
A empena cumpre os requisitos
```

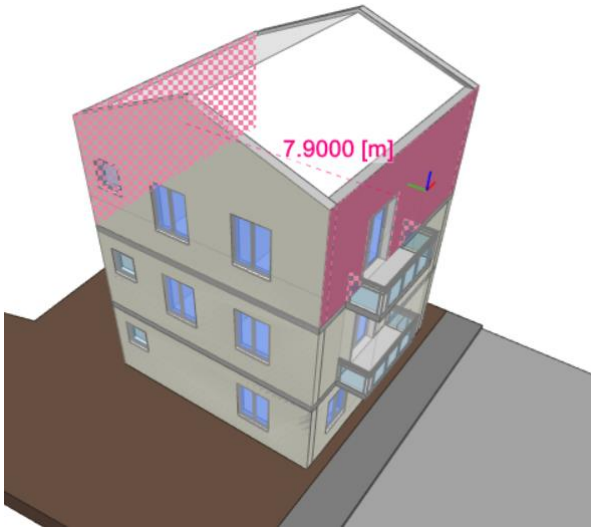

Figura 31: À esquerda: resultado da verificação. À direita: valores obtidos com o BIMVision.

3.3.2.3 Plano Diretor Municipal de Vila Nova de Gaia, Artigo 42.°, parágrafo 1

O terceiro exemplo é referente à faixa de implantação do edifício, que também é um tipo de verificação relativo à urbanização e, já que este parâmetro foi implementado num método da classe da parcela, sua verificação, como pode ser vista na Figura 32, possui uma sintaxe bem simples.

```python
# ------------------------------------------------------------------
# PDM de Vila Nova de Gaia, Artigo 42.º 1
# ------------------------------------------------------------------

'''As novas construções principais implantar-se-ão dentro de uma faixa de 35m,
confinante com o espaço público, sem prejuízo do previsto nos números seguintes.
'''

for parcel in my_permit.parcels:
    print(parcel.building_implantation_range() <= 35)
```

Figura 32: Código para verificação do artigo 42.°, parágrafo 1, do PDM de Vila Nova de Gaia, com modelo de entidades implementado.

Como pode ser visto na Figura 33, os resultados e valores obtidos para esta verificação estão de acordo com os valores encontrados através do BIMVision. Para a obtenção dos valores no software, a faixa foi determinada como a distância entre o plano delimitador do espaço público e o plano da fachada tardoz do edifício.

```
##################################################
Faixa de implantação: 8.3
A implantação está dentro da faixa regulamentar
```

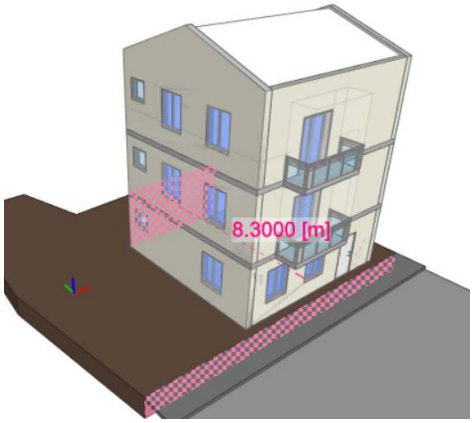

Figura 33: À esquerda: resultado da verificação. À direita: valores obtidos com o BIMVision.

4 Plataforma de verificação: conceção, implementação e demonstração

4.1 Enquadramento

Este capítulo apresenta o desenvolvimento da plataforma protótipo, seus principais componentes e a linguagem de programação visual desenvolvida. Desta forma, primeiro, é apresentada uma visão global da arquitetura estabelecida para o protótipo, o que é seguido pela apresentação dos seus principais componentes, entre eles, a linguagem de programação visual. Ao fim, é apresentado um caso de aplicação, onde o fluxo final para criação de regulamentos digitais e execução de verificações de conformidade na plataforma é demonstrado com utilização do modelo IFC apresentado no capítulo 3.

4.2 Arquitetura do protótipo

4.2.1 Visão global da arquitetura desenvolvida

Como abordado no capítulo 2, o desenvolvimento de aplicações web exige a utilização de recursos para o desenvolvimento em duas camadas, comumente denominadas frontend e backend. Apesar da existência de diversas soluções para o desenvolvimento simplificado de aplicações web, como por exemplo o Streamlit, estas não possuem a flexibilidade necessária para a implementação de soluções que envolvam fluxos de informação mais complexos. Logo, a implementação de um editor de regras de verificação em linguagem de programação visual e os fluxos necessários para a execução de verificações de conformidade em ficheiros IFC estimulou a adoção de soluções que pudessem proporcionar a flexibilidade necessária e ainda assim tivessem uma rápida curva de aprendizagem.

O primeiro passo para a escolha deste recurso, foi a definição dos casos de uso (Figura 34) a ser implementados na plataforma. Assim, os casos de uso obrigatórios determinados foram:

a) Criação de regras para verificação através de linguagem de programação visual, que implementa as classes do modelo de entidades apresentado no capítulo 3 e que devem ser armazenada em um formato interpretável por máquina;
b) Executar verificações a partir de modelos IFC;
c) Visualizar relatório de verificação;
d) Visualizar modelo IFC verificado;
e) Capacidade de se registar e realizar login na plataforma.

Além dos casos de uso obrigatórios, outros casos de uso opcionais foram estabelecidos, como a capacidade de realizar interações com o modelo IFC, p. ex., medições e observar os resultados da verificação a partir de destaques visual de objetos.

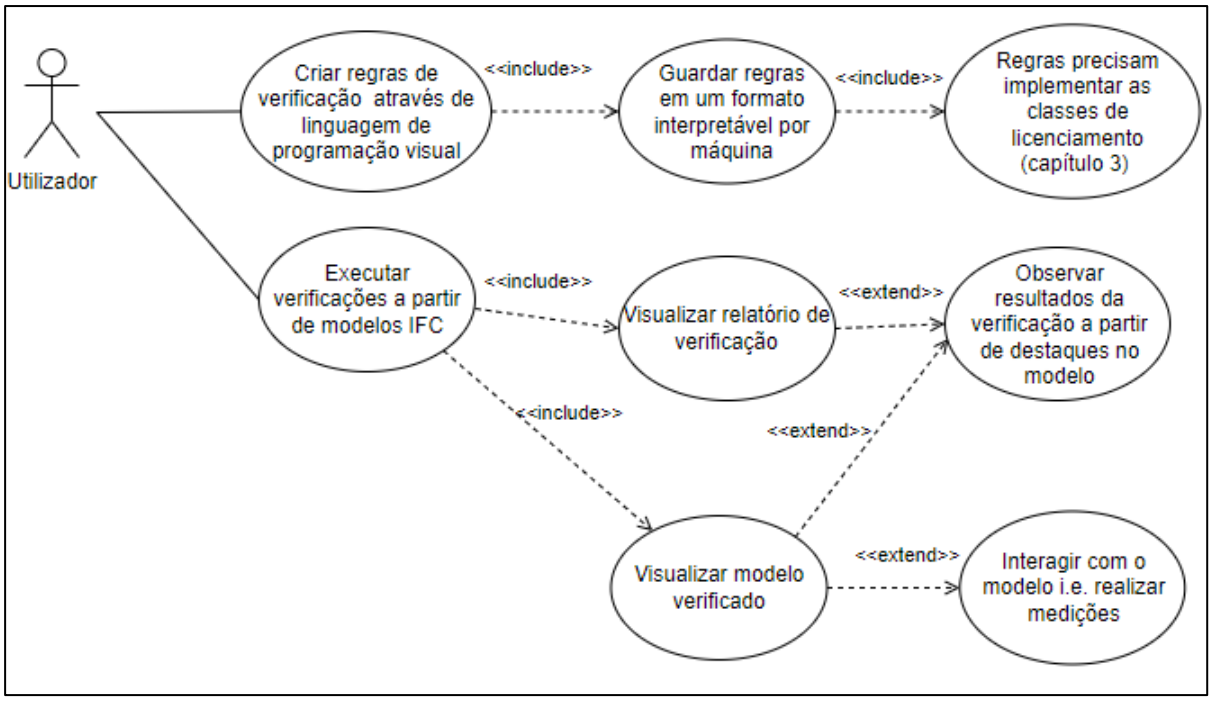

Figura 34: Casos de uso elencados.

A determinação dos casos de uso permitiu avaliar quais eram as ferramentas necessárias para desenvolver a plataforma, o que também foi determinado pela sua facilidade de implementação. Desta maneira, foram escolhidas ferramentas para o desenvolvimento frontend e backend, além das demais ferramentas necessárias para realizar os fluxos de informação entre estas duas camadas.

A escolha do Django como *framework* para o desenvolvimento da maior parte dos serviços backend foi motivada maioritariamente por dois fatores:

a) Capacidade de gestão das bases de dados simplificada a partir do desenvolvimento de classe em linguagem Python;
b) Integração com as ferramentas desenvolvidas para verificação de modelos IFC, apresentadas no capítulo 3.

Para a implementação dos serviços de frontend a escolha foi pela biblioteca React, que se devem aos seguintes fatores:

a) Facilidade de desenvolvimento de aplicações web modulares baseada em componentes;

b) Simplificação do desenvolvimento de páginas dinâmicas a partir de recursos de gestão de estado de componentes;

c) Possibilidade de integração com bibliotecas como o Xeokit, para visualização de modelos BIM, e com o Blockly, para criação e utilização da linguagem de programação visual.

O fluxo entre as camadas da aplicação foram realizadas através do estabelecimento de uma API, desenvolvida através da biblioteca Django REST Framework. Já ao nível do frontend, a biblioteca Redux permitiu centralizar o acesso às APIs e aos estados dos componentes da plataforma.

A definição dessas ferramentas permitiu estabelecer a arquitetura da plataforma, que pode ser vista de forma simplificada na Figura 35. É possível observar as duas principais camadas de implementação, com suas respetivas soluções e demonstração simplificada do fluxo implementado, que ocorre através da API estabelecida ao nível do backend e é acedida através de requisições, de forma a retornar valores em JSON que são então apresentados no frontend.

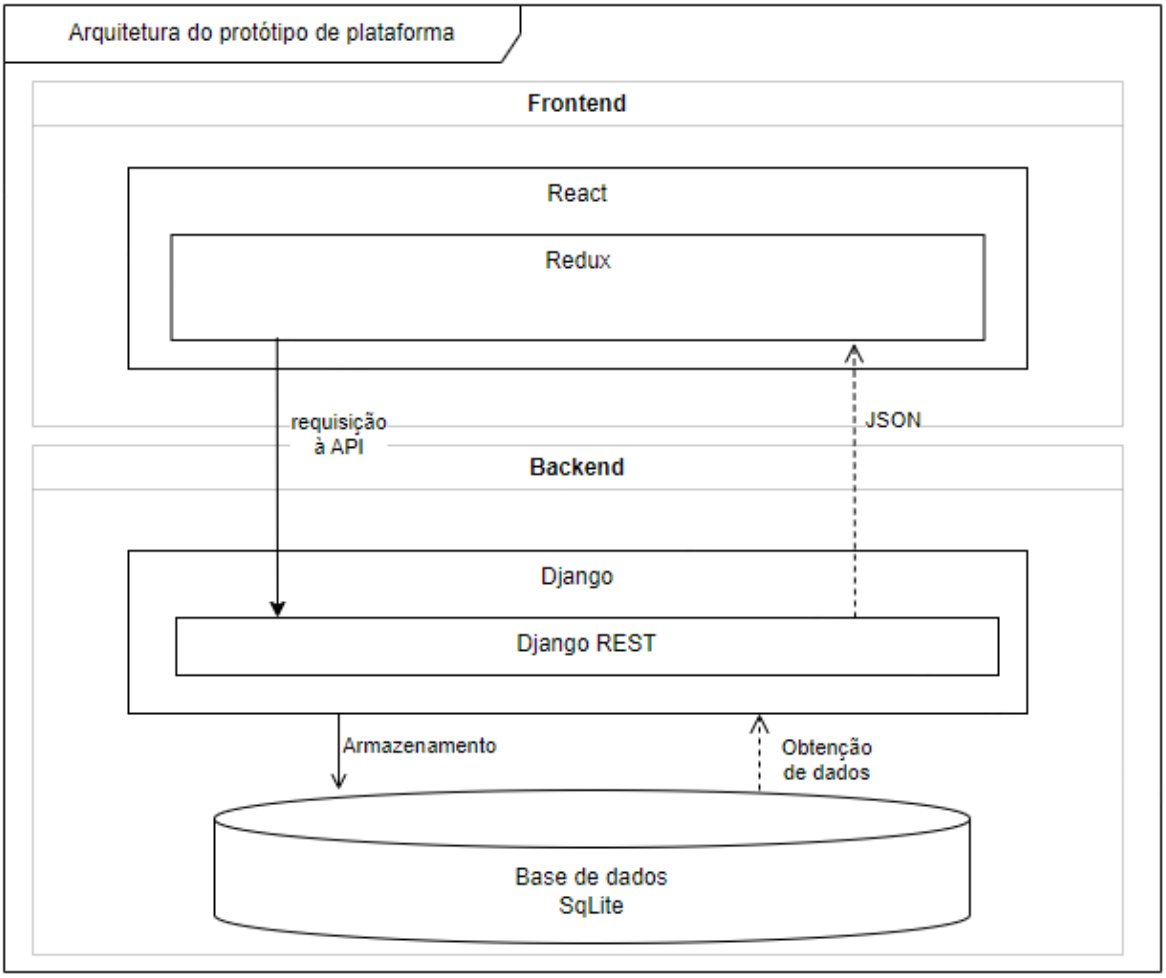

Figura 35: Arquitetura simplificada do protótipo.

Todos estes elementos serão aprofundados nas próximas secções, a começar pelos serviços de backend, com o desenvolvimento dos modelos do Django e implementação da API. Depois serão abordados os serviços de frontend, com uma breve explicação da estrutura de componentes desenvolvida e a integração com as bibliotecas para visualização dos modelos IFC. Posteriormente, há um aprofundamento na criação dos blocos desenvolvidos para este protótipo e da implementação do editor de regras de verificação. Finalmente, é mostrado alguns dos códigos das cláusulas analisadas no capítulo anterior desenvolvidos a partir da linguagem visual assim como imagens dos relatórios obtidos na plataforma.

4.2.2 Serviços backend e APIs

Um projeto em Django é comummente composto de aplicações modulares, de forma a implementar uma separação de responsabilidades. Assim as seguintes aplicações foram implementadas:

- **Verificação (AP1):** aplicação para lidar com o controle da execução de verificações e relatórios gerados;
- **Regulamento Digital (AP2) :** aplicação que lida com o armazenamento das regras criadas em um formato interpretável por máquina;
- **Módulo de verificação (AP3)**: módulo que permite executar regras de verificação a partir do modelo de entidades do licenciamento e demais ferramentas desenvolvidas no capítulo 3;

O desenvolvimento da estrutura da base de dados foi definida para as aplicações Verificação (AP1) e Regulamento digital (AP2), a partir do estabelecimento de modelos do Django, relativos à base de dados, que são demonstrados na Figura 36. Somente estas duas aplicações acedem e armazenam informações na base de dados. O modelo de Verificação (AP1) possui somente uma classe, com atributos para definição do modelo IFC a ser verificado, a ser apresentado no visualizador e para controle do tempo em que a verificação foi executada. Adicionalmente, a classe possui dois métodos:

- **get_rules():** permite obter todas as regras dos regulamentos associados à verificação e retornar o conjunto de regras a ser avaliada na execução da verificação;
- **run_verification():** permite acionar o Módulo de verificação (AP3) de forma a avaliar todas as regras obtidas pelo método “get_rules()” a partir do modelo IFC armazenado no atributo “ifc_file”.

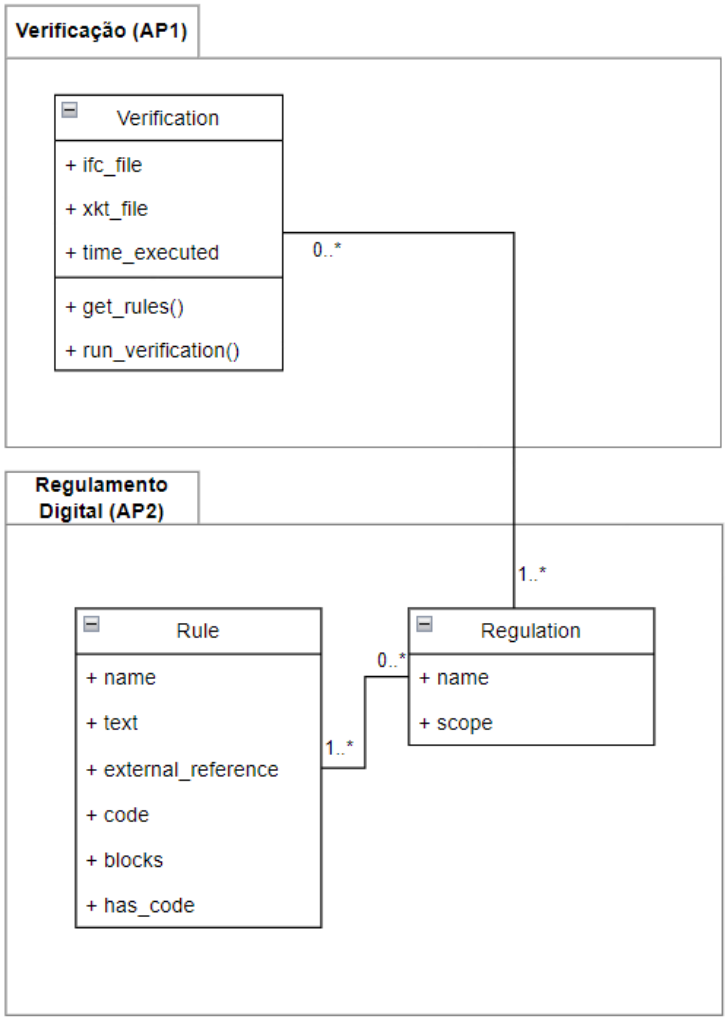

Figura 36: Modelos do Django desenvolvidos para estabelecimento da estrutura de base de dados.

O modelo do Regulamento digital (AP2) é composto por duas classes:

a) **Rule:** classe que possui atributos para a definição de características básicas das regras, como nome, texto da regra no regulamento original, assim como referência à cláusula de onde a regra foi obtida. Além disso, a classe possui o atributo "code", que armazena o código Python, e o atributo "blocks", que guarda informação dos blocos utilizados para gerar o código Python.

b) **Regulation:** classe que determina um grupo de regras através de um nome e um escopo, que faz referência ao âmbito de aplicação do regulamento, p. ex. municipal ou nacional. Portanto, esta classe estabelece o regulamento interpretável por máquina.

O Módulo de verificação (AP3) é responsável por ser chamado pela aplicação de Verificação (AP1) de forma a executar as regras de verificação, consoante o modelo de entidades do licenciamento estabelecido no capítulo 3. Ao final da avaliação das regras é gerado um relatório, em formato JSON,

com algumas informações sobre o resultado da execução. Desta forma, este módulo é composto de duas classes (Figura 37):

- **ComplianceCheck:** classe que importa as classes do licenciamento e executa os códigos da regra para avaliar o modelo IFC submetido.
- **Report:** classe utilizada para armazenar os relatórios de verificação, que são gerados no formato JSON, para ser transmitidos através da API.

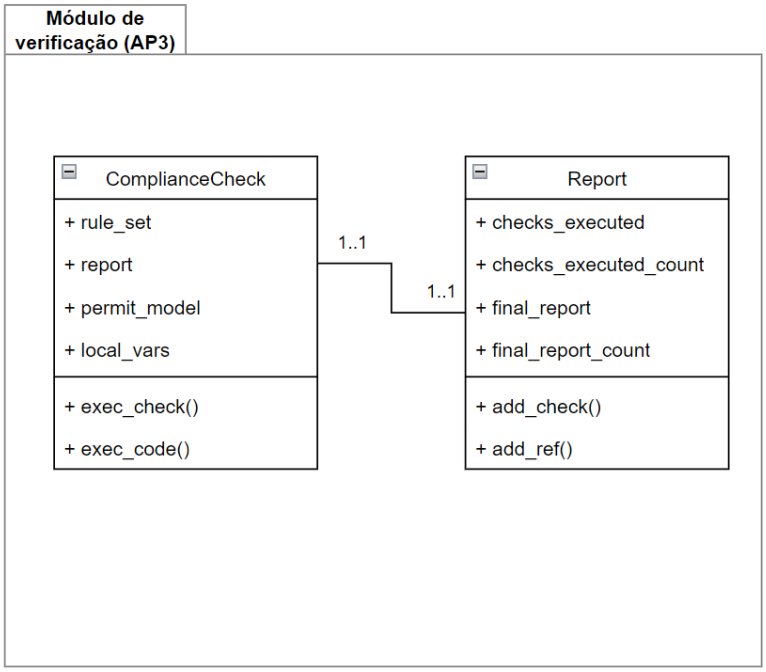

Figura 37: Estrutura do módulo de verificação.

Como a plataforma é controlada pelo utilizador na camada de frontend, é preciso estabelecer um fluxo de informação entre o frontend e backend e isto é feito através do estabelecimento de uma API ao nível do backend. Esta API permite a requisição e o envio de informação dos componentes do frontend às aplicações do backend, o que é executado através de um fluxo de informações no formato JSON. A conversão da informação da base de dados para o formato JSON é executada por serializadores. Desta forma, foram implementados serializadores para a aplicação Verificação (AP1) e Regulamento digital (AP2). O Módulo de verificação (AP3) não possui acesso à base de dados e somente é requisitado através da aplicação Verificação (AP1), de tal maneira que não possui serializador.

Desta maneira, a arquitetura final para os serviços de backend pode ser vista na Figura 38, e resume todos os aspetos abordados anteriormente.

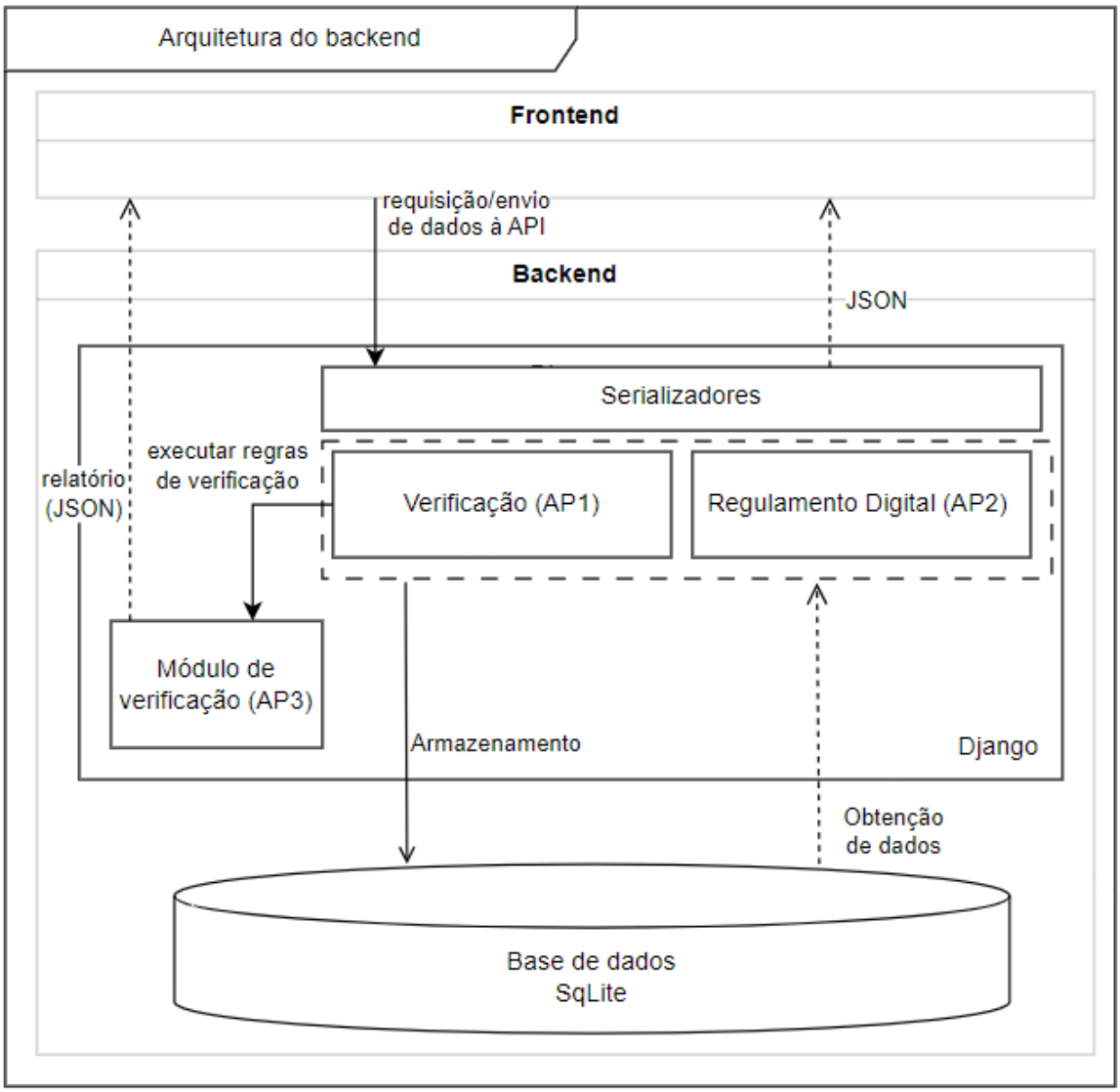

Figura 38: Arquitetura do backend.

4.2.3 Serviços frontend

O mapa da plataforma foi pensado para ser simples e por isso foram planeadas as seguintes páginas:

I. Regulamento digital: página onde são criados e editados os regulamentos digitais a partir de um editor desenvolvido a partir da biblioteca Blockly;
II. Painel de verificação: página onde é possível criar verificações a ser executadas, inserir os ficheiros IFC a verificar e a ver informações sobre as demais verificações executadas;

III. Relatório: página onde é possível executar as verificações, ver os relatórios gerados e inspecionar o IFC avaliado.

A partir deste mapa e dos requisitos estabelecidos nos casos de uso, foi estabelecido um projeto React., de forma que os componentes da plataforma podem ser caracterizados em 3 principais tipos:

- **Vistas:** pasta que contem as páginas implementadas.
- **Contexto:** contém os principais componentes de gestão global de estado através do Redux, além da implementação do controle da API.
- **Componentes gerais:** diretório principal dos componentes que integram as vistas da aplicação, desde a barra de navegação a modais. Aqui também foram implementadas as subpastas que contêm os componentes do visualizador BIM e todos os componentes necessários para a implementação do editor de regras de verificação e da linguagem em blocos desenvolvida.

Estes componentes também estabelecem a arquitetura do frontend, que pode ser vista na Figura 39. Assim, foi dado destaque aos componentes das vistas, que são os principais na implementação da plataforma. É possível observar que estes componentes são responsáveis por executar requisições e envios de dados à API por intermédio do contexto. Estes fluxos que determinam a possibilidade de se criar códigos através do editor de regras de verificação, que reside no componente de regulamentos e enviar as informações dos códigos gerados para ser armazenados em um formato interpretável por máquina na base de dados. Também é desta forma que ocorre a requisição para a execução de verificações, com o retorno de um relatório em formato JSON, que pode ser visualizado na página de relatório.

A estrutura final dos componentes da plataforma pode ser verificada no anexo III.

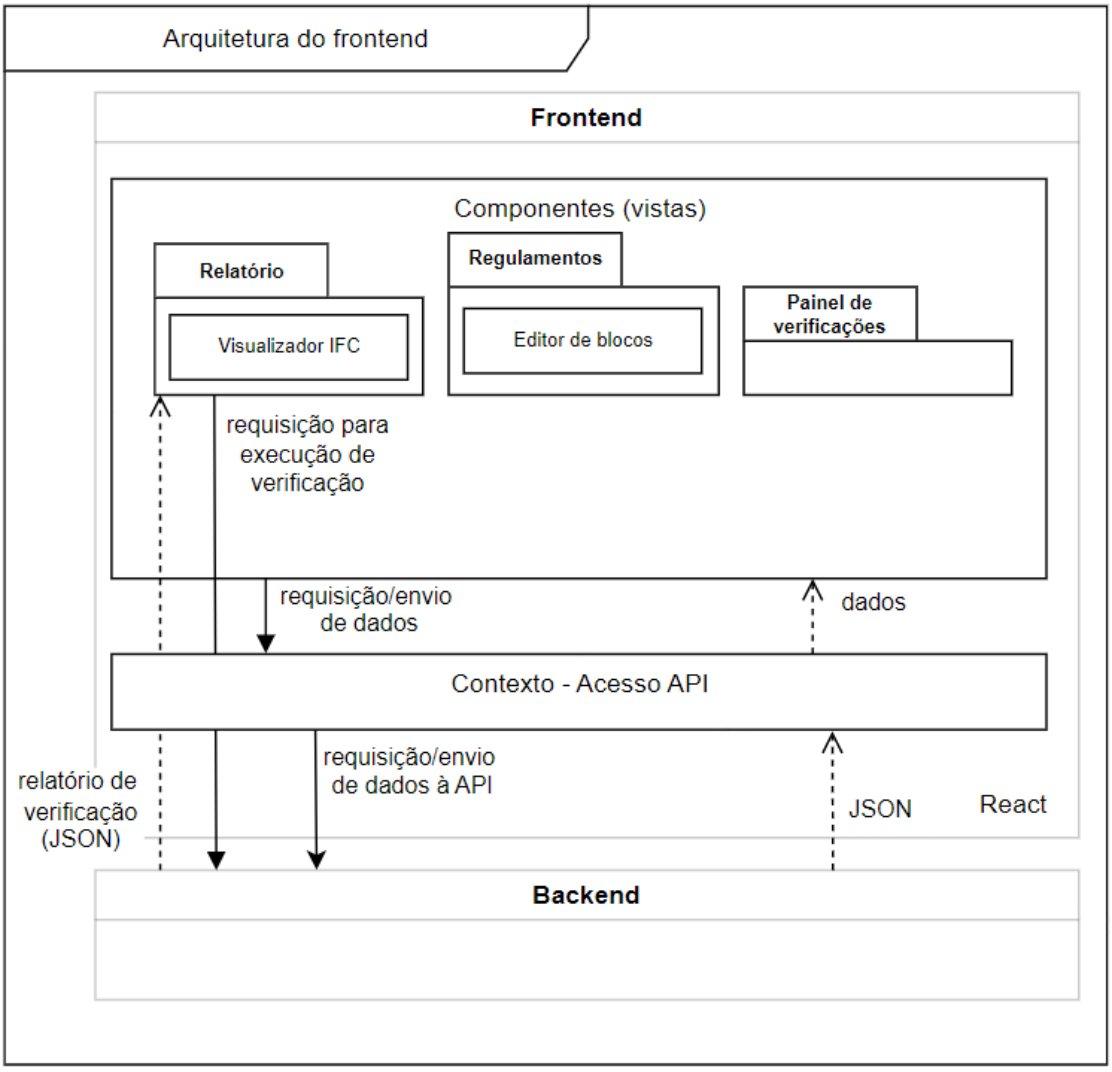

Figura 39: Arquitetura do frontend implementada no protótipo.

4.3 Criação da linguagem de programação baseada em blocos e implementação do editor de regras de verificação

Como apresentado no capítulo 2, a biblioteca Blockly permite o desenvolvimento de linguagens de programação visual em blocos capaz de gerar código Python. Isto é possível a partir da disponibilização de uma série de ferramentas que facilitam desde a criação de blocos customizados até a implementação de um editor. Desta forma, os procedimentos adotados para a disponibilização da capacidade de criação de regras de verificação a partir de uma linguagem visual no protótipo foram:

i. Criação de blocos customizados;

ii. Desenvolvimento de geradores de código Python adaptados ao modelo de entidades do licenciamento;

iii. Implementação do editor no protótipo.

A estratégia para a execução destes procedimentos foi também baseada na exequibilidade da tarefa, já que foi considerado que a demonstração do fluxo e meios de implementar esta solução são a peça chave deste trabalho. Deste modo, as próximas subsecções apresentam os pormenores da execução destes procedimentos.

4.3.1 Blocos customizados

A criação dos blocos partiu da análise do modelo de entidades do licenciamento conforme desenvolvimento demonstrado no capítulo anterior, para identificar a melhor maneira de disponibilizar elementos visuais que pudessem codificar as verificações de forma intuitiva. Isto envolveu a necessidade de conduzir esta criação a partir das boas práticas para desenvolvimento de linguagens visuais, o que é pautado pela definição de alguns aspetos primordiais que guiarão o desenvolvimento do vocabulário da linguagem (i.e. os blocos a ser disponibilizados) e os aspetos visuais e a sintaxe a ser implementada. Desta maneira os principais aspetos considerados foram:

Características dos utilizadores da linguagem: como apresentado anteriormente, os principais utilizadores da linguagem visual para a criação de regras de verificação são os técnicos nas câmaras, que usualmente não possuem muita familiaridade com linguagens de programação. Desta forma, os blocos desenvolvidos tiveram de adotar um nível de abstração que permitisse flexibilidade para execução de diversos tipos de verificação, mas que não adotasse toda a complexidade comumente existente nas linguagens de programação em script.

Âmbito: um dos objetivos foi definir a menor quantidade de blocos necessários para a execução das verificações analisadas, já que a simplicidade de utilização também passa pela limitação de opções disponíveis ao utilizador.

Visual: um dos objetivos foi estabelecer uma paleta de cores de forma a representar as categorias de blocos desenvolvidos. Além disso, a utilização de ícones para melhor representar a função dos blocos foi priorizada. É expetável que a utilização destes recursos visuais favoreça um uso mais intuitivo do utilizador.

A partir da definição destes aspetos, a primeira tarefa executada foi a definição do vocabulário da linguagem. Optou-se por disponibilizar blocos relativos ao modelo de licenciamento desenvolvido, de forma que cada entidade pudesse ser convertida num bloco codificável, logo disponível para utilizar na

criação de regras. Esta abordagem é interessante pois materializa o conceito de design direcionado a objetos que foi utilizado para a definição do modelo de entidades no capítulo anterior e facilita a sua utilização. Esta abordagem também facilita a implementação futura de um fluxo de criação automática de blocos a partir da definição de um novo modelo de entidades ou da expansão do modelo existente.

Um fator central na definição destes blocos foi ter em consideração que a sua implementação é restrita pelo código que pretende-se desenvolver. Isto implica que o design deve refletir a lógica do código a ser gerado por cada bloco e pela conexão entre blocos. Neste trabalho, isto implicou em pensar no código utilizado para executar as verificações, demonstrado no capítulo anterior, e transferir o fluxo de implementação utilizado para um nível maior de abstração, ou seja, um formato mais simplificado a ser desenvolvido com os blocos.

Deste modo, foram desenvolvidos 3 tipos de blocos relativos ao modelo de licenciamento:

Blocos de loop: estes blocos, demonstrados na Figura 40, executam um loop entre todas as instâncias de uma determinada classe. Esta solução foi desenvolvida de forma a facilitar o processo de percorrer as todas as entidades, o que usualmente ocorre em processos de análise e obtenções de informação. Entretanto, esta abordagem implica que os blocos de nível inferior estejam associados aos blocos dos níveis superiores para que o loop ocorra corretamente. Logo, o bloco de edifício deve estar incorporado num bloco de parcela para ser avaliado. Esta lógica também está associada a limitações do modelo de entidades, de tal maneira que futuras melhorias podem resultar num vocabulário de blocos mais eficiente e intuitivo;

Figura 40: Blocos de loop.

Blocos de propriedade: estes blocos, como exemplificado na Figura 41, representam as propriedades das entidades do licenciamento e podem ser utilizados tanto para a verificação de valores

regulamentares, quanto para o estabelecimento de condições aplicáveis à verificação. Estes blocos foram desenvolvidos em 3 subtipos: propriedades de categoria; propriedades numéricas e propriedades booleanas. Esta estratégia foi pensada a facilitar o desenvolvimento automático destes blocos futuramente, a partir da definição dos tipos de propriedades no modelo de licenciamento. Na prática, estes blocos funcionam através da seleção da propriedade que se pretende utilizar na regra;

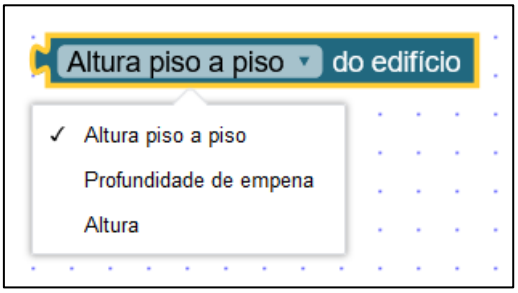

Figura 41: Exemplo de bloco de propriedades.

Blocos de elementos relacionados: os blocos de elementos relacionados servem para possibilitar a execução de operações numa lista de entidades "filhas" de uma determinada entidade utilizada em um bloco de loop. Isto significa que ao utilizar o bloco "pavimentos do edifício", representado na Figura 42, é possível obter a lista de todos os pavimentos referentes a um determinado edifício. Um dos benefícios da implementação deste bloco para as cláusulas analisadas é determinar o número de elementos da lista de forma a obter dados como o número de pavimentos, compartimentos, entre outros que são agrupados em entidades de nível hierárquico superior, entretanto estes blocos também permitem uma implementação mais flexível de loops entre as entidades do modelo, proporcionando uma maior flexibilidade no desenvolvimento de regras mais complexas.

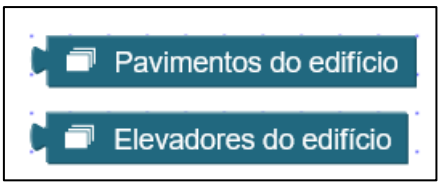

Figura 42: Exemplos de blocos de entidades relacionados.

Ao todo foram criados 28 blocos referentes ao modelo do licenciamento, que podem ser conectados de variadas formas de modo a criar regras. Esta conjunto de possibilidades gerado a partir do rearranjo dos blocos disponibilizados é um dos fatores que implica na escalabilidade da abordagem implementada, de modo que blocos utilizados numa verificação também podem ser utilizadas noutras verificações. Os blocos do modelo representam as entidades do modelo de licenciamento, desta forma,

foi necessário desenvolver blocos para representar aos operações, valores e fluxos que podem ser desenvolvidos com estas classes.

Assim, foram estabelecidas 4 categorias de blocos, adicionais para além da categoria de blocos do modelo. Esta categorização teve como finalidade organizar os tipos de blocos disponibilizados, facilitar o desenvolvimento de regras e facilitar uma expansão organizada dos blocos. As categorias e suas descrições são apresentadas em seguida:

Lógica: Blocos de estrutura condicional "se" e operadores lógicos, como pode ser visto na Figura 43. Com estes blocos é possível definir condições de aplicabilidade para determinada verificação, assim como determinar as comparações a ser executadas nas verificações.

Figura 43: Blocos da categoria lógica.

Verificação: Blocos para estabelecimento das condições a ser verificadas. Possui bloco "verificar", que pode ser visto a ser utilizado na verificação da área bruta de um determinado fogo na Figura 44. Este bloco é um dos principais componentes da verificação.

Figura 44: Bloco de verificação, a estabelecer a verificação da área bruta de um fogo.

Valores: Blocos referentes a valores numéricos, textuais e de classes a ser utilizados como parâmetros de condições e verificações (Figura 45). Com estes blocos é possível estabelecer os valores de classes de compartimento a ser comparados para estabelecimento de requisitos e condições de aplicabilidade.

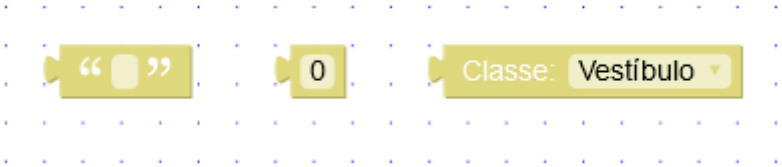

Figura 45: Blocos de valores.

Matemática: Blocos para operação matemática e contagem de elementos (Figura 46). Com esse bloco é possível contar a quantidade de elementos filhos de algum bloco de loop, assim como definir operações aritméticas a ser executadas.

Figura 46: Blocos da categoria matemática.

Finalmente, houve o cuidado de estabelecer duas paletas de cores (Figura 47) para a utilização nos blocos das categorias e dos blocos do modelo. Este fator é relevante no desenvolvimento de linguagens visuais de bloco, pois indicam com maior precisão as funções de cada bloco, já que representam as diferentes categorias e a hierarquia da categoria do modelo, o que torna mais fácil a utilização dos blocos por pessoal não especializado em programação.

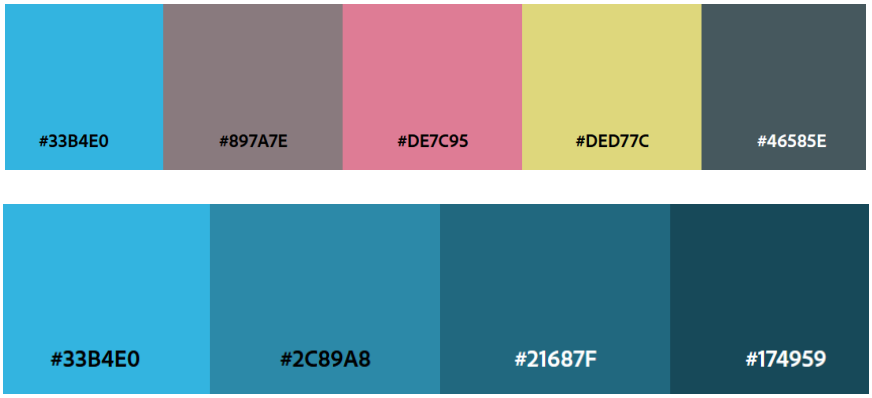

Figura 47: Paletas de cores adotadas.

4.3.2 Geradores de código

Para implementar os blocos customizados, é necessário criar geradores de código que obtêm os valores e declarações associadas a um determinado bloco e transformam em código referente a uma linguagem de programação, que no caso deste trabalho é o Python. A biblioteca Blockly possui um módulo gerador de código Python por defeito. Grande parte das funcionalidades deste blocos são relativas apenas a alguns pormenores da sintaxe da linguagem, como indentação e a prioridade dos elementos integrantes de uma declaração. Portanto, os códigos desenvolvidos para os blocos customizados tiveram em sua grande maioria de ser implementados. Isto, entretanto, proporcionou a flexibilidade ideal para criar os códigos que fossem adequados ao modelo de licenciamento

desenvolvido, o que é um requisito fundamental para a execução das verificações a partir do código gerado.

Um vez que o Blockly já possui alguns blocos por defeito, que foram utilizados em categorias como valores, lógica e matemática, a grande parte do trabalho de implementação de geradores de código foi centrada nos blocos referentes ao modelo de entidades do licenciamento. Desta maneira, já que os blocos do modelo foram desenvolvidos em 3 tipos, é possível estabelecer o método utilizado para a geração de código para cada um dos tipos desenvolvidos, o que é apresentado a seguir.

Blocos de loop: os blocos de loop são definidos pelo loop entre uma lista de todos os elementos de uma determinada classe de elementos. O desenvolvimento do gerador de blocos de loop implementa o fluxo apresentado na Figura 48, envolvendo a obtenção dos blocos associados e, caso estes blocos existam, o seu código é executado no loop definido pelo bloco principal. Caso o bloco principal esteja vazio, é adicionado um pass ao seu loop para mitigar erros de código Python.

Na notação utilizada no algoritmo, os elementos entre parêntesis retos são *placeholders* que são definidos consoante o bloco, seguindo as regras estabelecidas na classe de licenciamento estabelecida no capítulo 3.

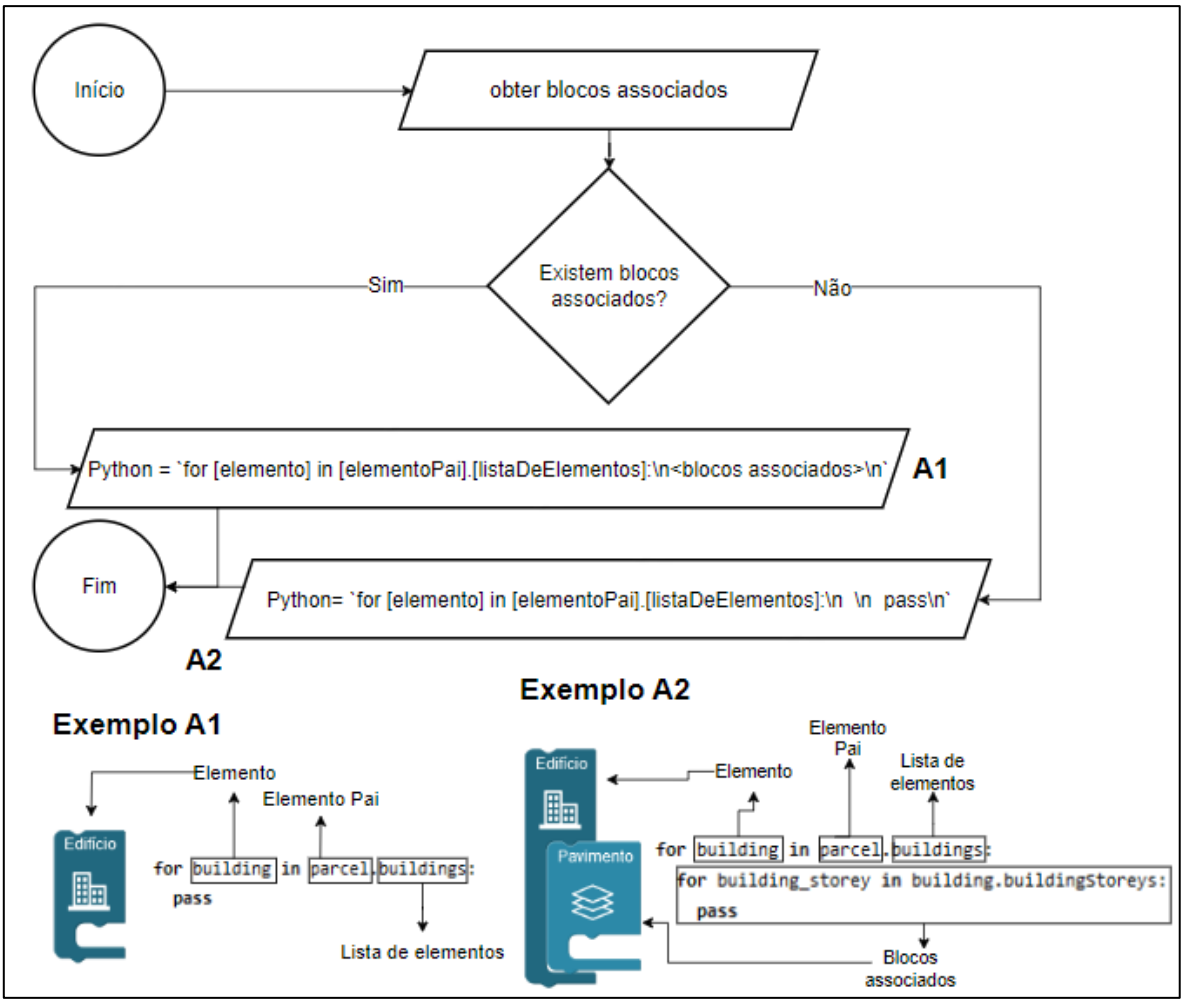

Figura 48: Algoritmo de gerador de código para blocos em loop.

Blocos de propriedades: os blocos de propriedade implementam o algoritmo apresentado na Figura 49, onde o código Python gerado é constituído por um objeto e seu atributo ou método, consoante a propriedade selecionada no bloco.

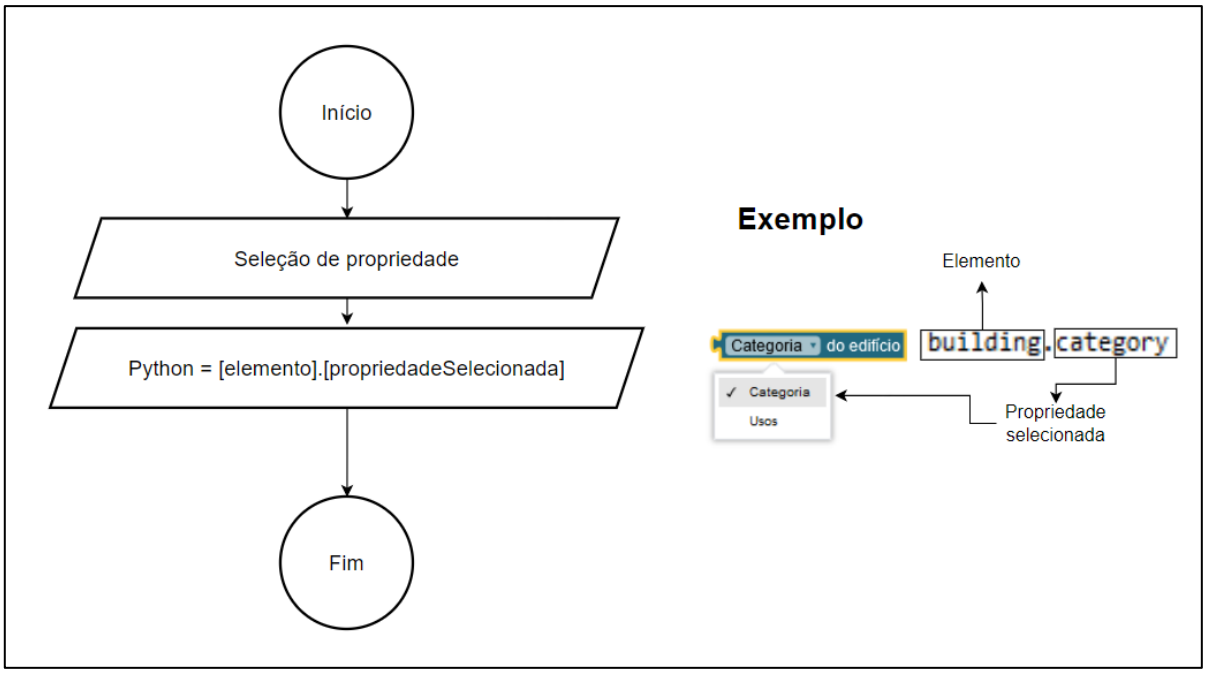

Figura 49: Algoritmo para gerador de código para blocos de propriedades.

Blocos de elementos relacionados: os blocos de elementos relacionados seguem um fluxo simples, como demonstrado na Figura 48, que envolve simplesmente a atribuição da lista de elementos ao elemento pai através da sintaxe da linguagem Python.

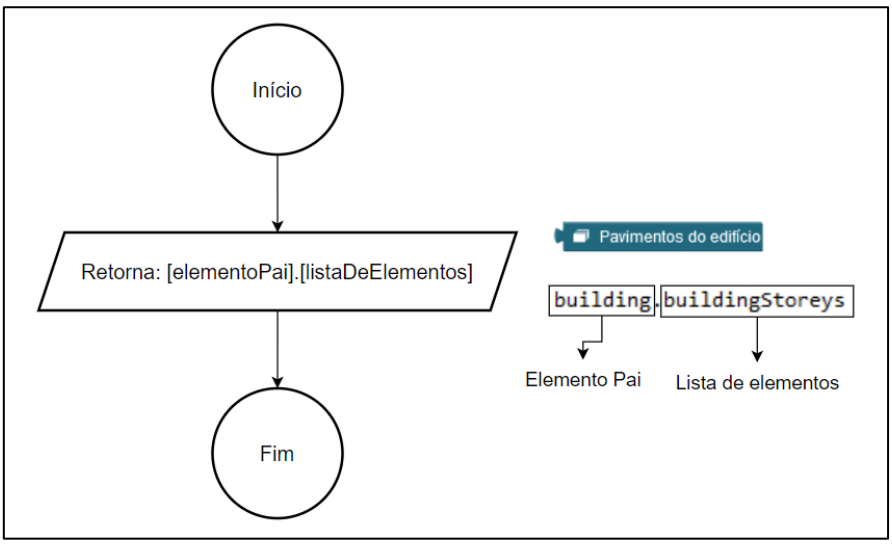

Figura 50: Algoritmo para gerador de blocos de elementos relacionados.

A partir dos geradores de código desenvolvidos com estes algoritmos, é possível criar verificações que implementam código Python, com utilização das classes de licenciamento apresentadas no capítulo 3, de forma a executar todas as verificações analisadas. Além disso, é possível executar diversas outras

combinações com estes blocos, o que potencialmente permite criar outras verificações, o que proporciona escalabilidade e flexibilidade.

4.3.3 Editor de regras de verificação

Para utilizar os blocos desenvolvidos para criar regras de verificação, foi preciso implementar um editor que permitisse a utilização da linguagem visual em blocos, o que foi consideravelmente facilitado pela biblioteca Blockly, de forma que os principais elementos necessários à sua definição foram as configurações visuais e de comportamento do editor, assim como a "caixa de ferramentas" a ser disponibilizada. A versão final do editor pode ser vista na Figura 51, e apresenta os seguintes elementos: (a) categoria de blocos disponível; (b) zona de construção de regras de verificação; (c) botão para salvar regras; (d) botão para mostrar código gerado.

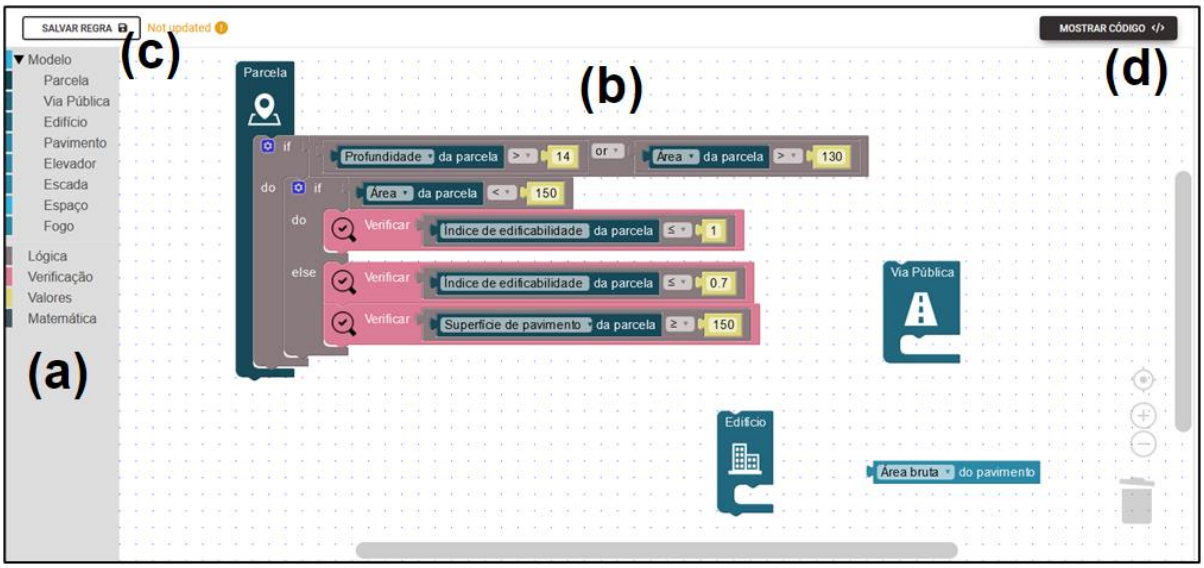

Figura 51: Editor de regra de verificação implementado.

4.4 Implementação do visualizador

Como abordado na seção 4.2, a visualização do modelo IFC verificado foi considerada essencial para o protótipo, já que a sua visualização é importante antes da verificação, de forma a executar confirmações do ficheiro que será verificado, assim como depois, de forma a assistir a visualização do relatório.

Um dos desafios encontrados para a implementação do visualizador foi a escolha de uma biblioteca que fosse estável e que dispusesse de métodos capazes de executar as funções desejadas. A

disponibilidade de documentação e a integração com o React também foram prontos cruciais. Deste modo, a escolha, pelas suas vantagens em todos os parâmetros mencionados, foi o Xeokit. O visualizador do IFC.js também foi testado e, apesar de possuir um grande potencial para implementações futuras, devido à sua rapidez no carregamento de ficheiros e qualidade do motor gráfico, a documentação ainda é inferior à do Xeokit, além de ser bastante recente e possuir problemas para a implementação no React dos fluxos estabelecidos para o desenvolvimento deste protótipo.

A estratégia para a implementação do visualizador foi a criação de uma classe para a determinação dos atributos e métodos que definem configurações e funcionalidades (Figura 52). Isto possibilitou o desenvolvimento de um sistema modular o que, integrado ao uso de um middleware, permitiu o acesso ao visualizador por todos os componentes da plataforma.

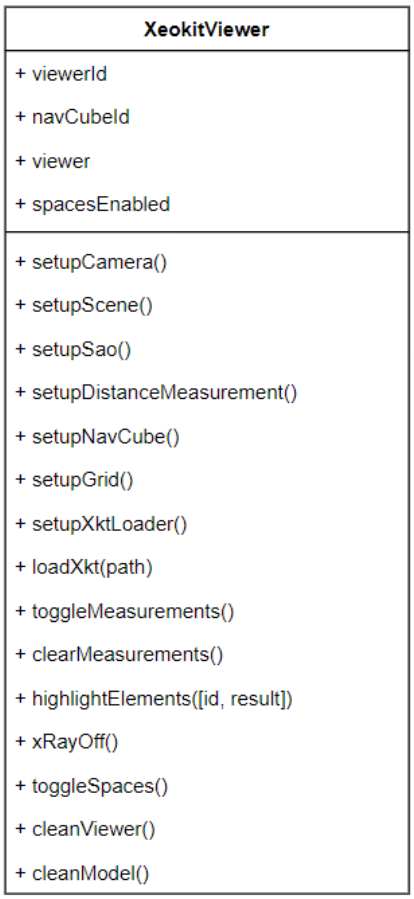

Figura 52: Estrutura da classe do visualizador implementada.

A partir dos métodos desenvolvidos é possível executar medições, como apresentado na Figura 53, ativar espaços e renovar a visualização de elementos apagados. Um método em particular, denominado "highlightElements()" foi desenvolvido de forma a permitir o destaque de elementos

consoante o seu identificador global e um valor booleano, de tal maneira que foi implementada para permitir a visualização dos resultados de algumas verificações no relatório.

Uma das desvantagens da utilização do Xeokit reside na demora para visualização de modelos IFC diretamente, de forma que o fluxo mais apropriado para mitigar este problema envolve a conversão dos IFCs para o seu formato XKT, que é executado de forma muito mais rápida. Este fluxo foi então implementado no carregamento de ficheiros, onde os ficheiros IFC são convertidos e armazenados no formato XKT. Assim, após o carregamento e conversão, os ficheiros armazenados podem ser visualizados rapidamente todas as vezes sem a necessidade da execução de uma nova conversão.

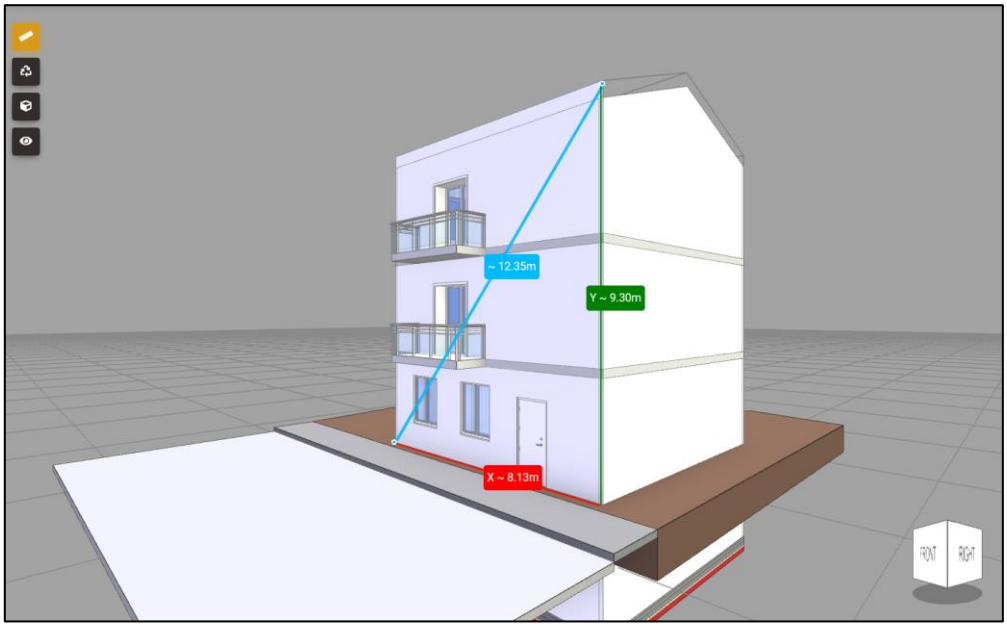

Figura 53: Visualizador implementado.

4.5 Páginas da plataforma e fluxo de utilização

A partir do desenvolvimento e integração dos componentes que foram apresentados anteriormente, foi possível implementar a plataforma, que consiste nas 3 páginas principais planeadas, que são apresentadas a seguir com descrição dos seus elementos de layout principais.

A primeira página é denominada “Regulamentos digitais”, onde é possível criar e editar regulamentos interpretáveis por máquina a partir da criação de regras em uma linguagem de programação visual baseada em blocos. Os principais elementos da página podem ser vistos na Figura 54 e são:

a) Seletor e criador de regulamentos digitais: possibilita criar e selecionar regulamentos para criar/editar regras para verificação;

b) Lista de regras: apresenta a lista de regras do regulamento selecionado. É possível selecionar uma regra para editar ou criar código para ela, assim como criar novas regras.

c) Informações da regra: apresenta o nome e o texto da regra selecionada.

d) Editor de regras: permite editar e criar códigos para a regra selecionada.

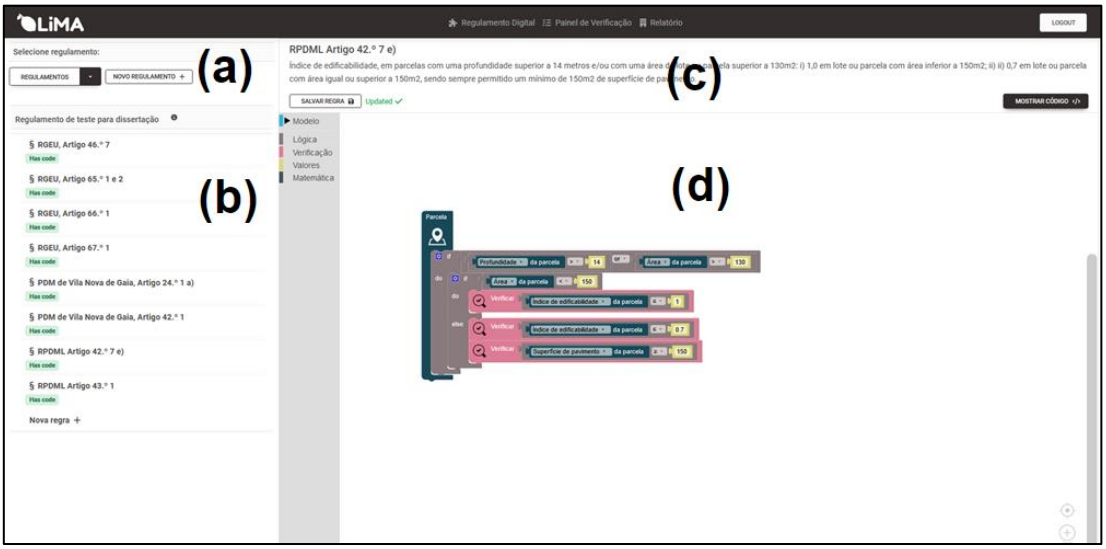

Figura 54: Página de regulamentos digitais.

A segunda página da plataforma é denominada "Painel de verificação" (Figura 55), onde é possível criar novas verificações para ser executadas, assim como carregar os ficheiros IFC que serão verificados. Os principais componentes desta página são:

a) Botão de nova verificação: permite criar novas verificações a partir dos regulamentos que se deseja verificar.

b) Tabela de verificações: mostra as verificações criadas e permite carregar ficheiros para as verificações.

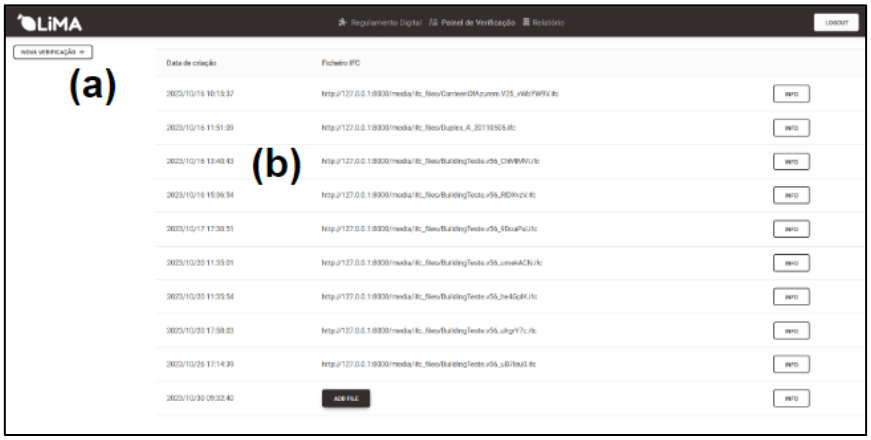

Figura 55: Página painel de verificação.

A última página é denominada "Relatório" e permite a execução das verificações criadas no painel de verificações e a visualização dos modelos IFC e relatórios de verificação. Os principais elementos da página podem ser vistos na Figura 56 e são:

a) Seletor de verificações: é possível selecionar a verificação que se presente executar;
b) Informações: apresenta informações gerais sobre a verificação selecionada;
c) Botão de execução: permite executar a verificação selecionada;
d) Relatório de verificação: apresenta o resultado da verificação para todas as regras avaliadas, que consiste no resultado da conformidade do modelo IFC e de algumas informações de cada objeto verificado;
e) Visualizador de modelos IFC: permite a visualização antes da execução da verificação, para confirmar o modelo a ser verificado, e permite avaliar algumas informações depois da execução a partir de medidas e do destaque de alguns elementos.

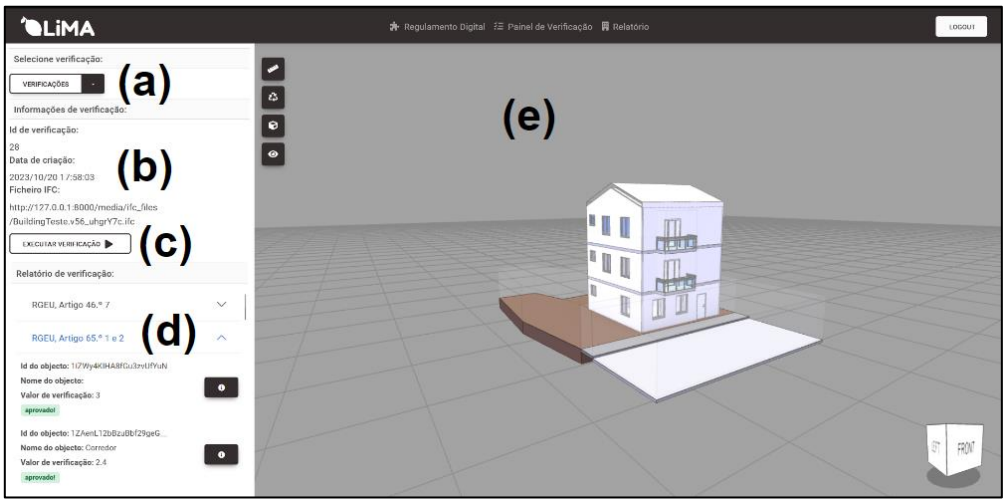

Figura 56: Página Relatório.

A partir do desenvolvimento dos componentes e das páginas foi possível estabelecer o fluxo final de utilização da plataforma protótipo (Figura 57). Os dois primeiros passos são referentes à (1) criação de um novo regulamento e à (2) codificação de suas regras, o que acontece a nível da página "Regulamentos" com utilização da linguagem de programação visual no editor implementado. Ao fim da codificação das regras, o seu código Python está armazenado, graças à estrutura da base de dados demonstrada na seção 4.2.2. Seguidamente, na página "Painel de verificações", é preciso (3) criar uma nova verificação, o que gera um registo de verificação na base de dados, que serve para determinar os regulamentos que serão verificados. Então, (4) o ficheiro IFC que será verificado precisa ser carregado, o que irá associá-lo ao registo de verificação criado no passo anterior. Finalmente, na

página “Relatório”, é possível (5) executar a verificação a partir do registo criado e (6) visualizar os relatórios da verificação e inspecionar o modelo IFC.

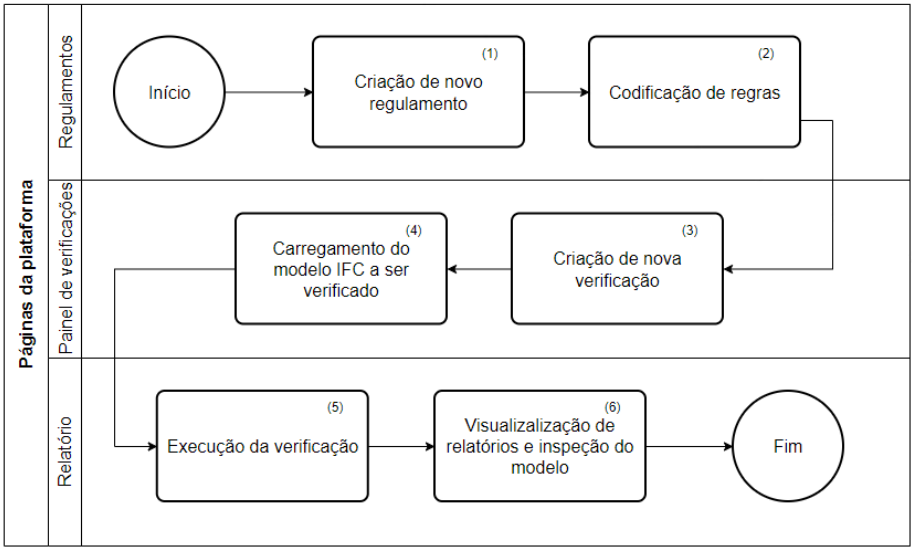

Figura 57: Fluxo de verificações final implementado na plataforma protótipo

4.6 Aplicação da plataforma para verificação de conformidade

A fim de validar a plataforma, o fluxo apresentado na seção 4.5 é aplicado com a criação de um regulamento que contém as regras utilizadas na validação do capítulo 3 e a verificação do modelo IFC de teste também demonstrado no capítulo 3.

4.6.1 Criação de regulamento e regras

De forma a promover a validação da plataforma, foi criado um regulamento de teste (Figura 58) , de forma a simplificar a operação realizada.

Figura 58: criação de regulamento de teste para validação da plataforma.

O passo posterior à criação dos regulamentos na plataforma é a codificação das regras, o que foi feito e mostrado nas subseções seguintes, com referências ao texto original das cláusulas e comentários sobre os pormenores da codificação executada. Estas próximas subseções são focadas somente na criação dos códigos das regras a partir da linguagem de blocos desenvolvida, já que os pormenores de baixo nível do código gerado pelos blocos foram apresentados no capítulo 3, com apresentação das abordagens usadas para implementação dos cálculos e do mapeamento com o IFC.

4.6.1.1 Codificação do RGEU, Artigo 67.º, parágrafo 1

O texto original deste artigo, como visto no capítulo 3, dispõe sobre requisitos acerca da área bruta dos fogos consoante o tipo de fogo, apresentados através na Tabela 6:

Tabela 6: Requisitos mínimos para área bruta dos fogos. Fonte: RGEU.

	Tipo de fogo							
	T0	T1	T2	T3	T4	T5	T6	Tx > 6
área bruta em metros quadrados.....	35	52	72	91	105	122	134	1,6 x Ah

Para o desenvolvimento através dos blocos, foi necessário utilizar os blocos de loop para percorrer todas as instâncias de fogos presentes no modelo e seus elementos superiores. Posteriormente, foi estabelecido um bloco de condição de forma a determinar verificações consoante o número de quartos do fogo. Desta forma, a regra codificada em blocos pode ser vista na Figura 59.

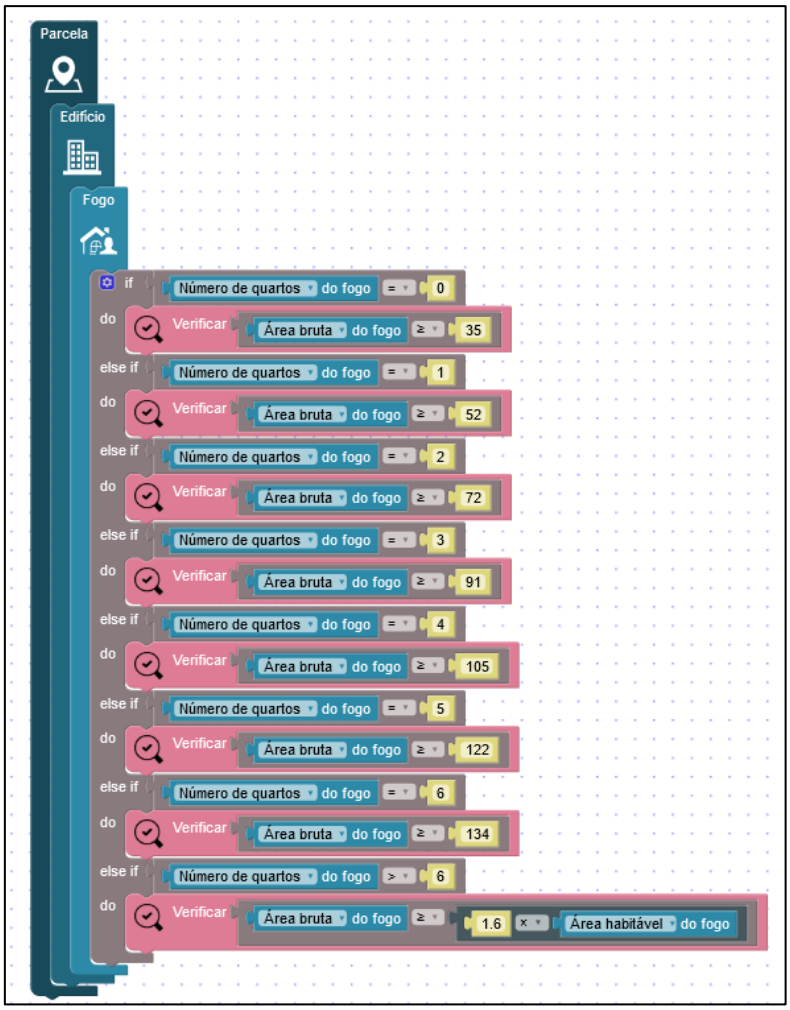

Figura 59: Regra em blocos para o RGEU, Artigo 67.º , parágrafo 1.

4.6.1.2 Codificação do RPDML Artigo 43.º, parágrafo 1

O texto original do Regulamento do Plano Diretor Municipal de Lisboa é:

"Sem prejuízo do disposto nos números seguintes, a profundidade máxima das empenas, sem considerar as varandas e os corpos balançados, é de 15 metros, com exceção dos estabelecimentos hoteleiros e equipamentos de utilização coletiva, cuja empena pode atingir os 18 metros."

Para a codificação desta regra, foi preciso avaliar através de uma condição "se" a categoria do edifício, que eram "estabelecimento hoteleiro" ou "equipamento coletivo", através de valores de texto para verificar as suas categorias. De forma a proporcionar maior flexibilidade na verificação, a implementação de todas as categorias possíveis como um valor selecionável é recomendada para implementações futura, de forma a dirimir erros do usuário. Assim, após o estabelecimento da condição, os blocos para a verificação de empena forma anexados, de forma que, em caso positivo, a

profundidade deverá ser menor ou igual a 18 metros e, em caso negativo, deverá ser menor ou igual a 15 metros.

Neste caso, já que a propriedade profundidade de empena já foi determinada no modelo de entidades, como apresentado no capítulo anterior, como a desconsiderar as varandas e os corpos balançados, o bloco mantém este método de cálculo, de forma que não preciso realizar nenhuma operação adicional. Além disso, um aviso foi implementado ao passar o cursor do rato pelo bloco a avisar este pormenor, como pode ser visto na Figura 60.

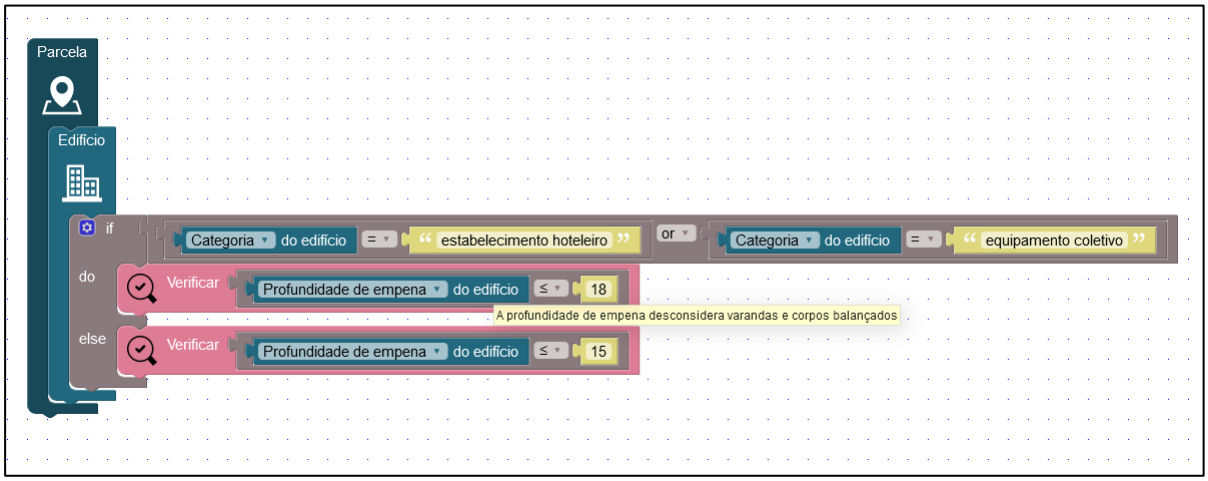

Figura 60: Regra em blocos para o RPDML Artigo 43.º, parágrafo 1.

4.6.1.3 Codificação do PDM de Vila Nova de Gaia, Artigo 42.º, paragrafo 1

O texto original do Plano Diretor de Vila Nova de Gaia determina que:

"As novas construções principais implantar-se-ão dentro de uma faixa de 35m, confinante com o espaço público, sem prejuízo do previsto nos números seguintes."

A codificação desta cláusula passou simplesmente por atribuir um bloco de verificação a um bloco de loop da parcela com o requisito da faixa de implantação, como pode ser visto na Figura 61. Neste caso, também é importante notar que o texto original determina que esta verificação não prejudica as regras dos parágrafos seguintes do artigo, o que implica em nenhuma codificação extra, já que numa implementação abrangente do regulamento, todas as verificações aplicáveis serão avaliadas.

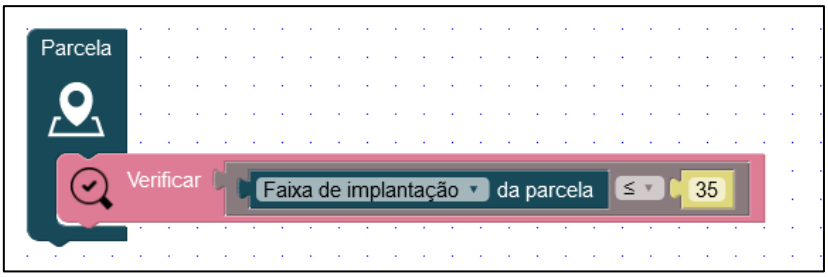

Figura 61:Regra em blocos para o PDM de Vila Nova de Gaia, Artigo 42.º, paragrafo 1.

4.6.2 Criação de verificação e carregamento de ficheiro IFC

Para realizar uma verificação, é necessário primeiro criar uma, o que foi feito na página "Painel de verificação" com seleção do regulamento de teste criado para ser verificado (Figura 62).

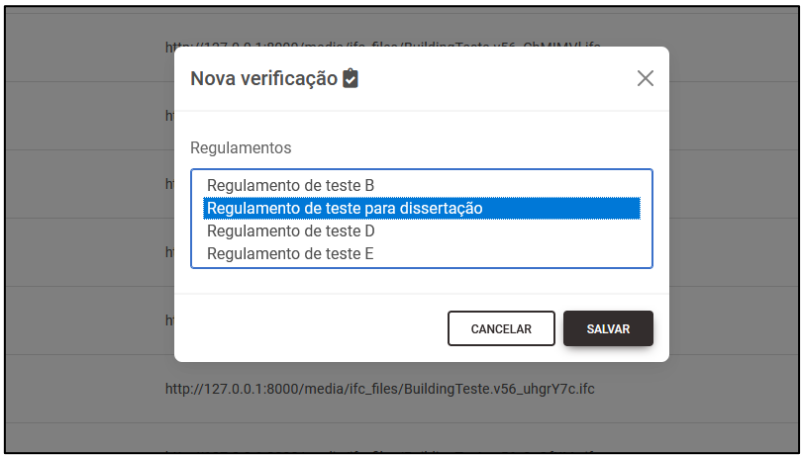

Figura 62: Criação de uma nova verificação.

Após a criação da verificação, é preciso carregar o ficheiro IFC, que será então convertido para o formato XKT, como demonstrado na Figura 63, e poderá então ser salvo. Assim, a verificação será atualizada com o modelo IFC a ser verificado e com o ficheiro XKT para visualização.

Figura 63: Carregamento e conversão de ficheiro IFC.

4.6.3 Execução de verificação e visualização dos resultados

O passo final para a verificação é executá-la na página de relatório, o que pode ser feito primeiro a partir da seleção da verificação que se deseja executar no painel à esquerda, destacado na Figura 64. Logo após a seleção, é possível verificar as informações básicas sobre a verificação e visualizar o modelo IFC a ser verificado. Após a confirmação das informações da verificação, ela pode ser executada a partir do botão "executar verificação".

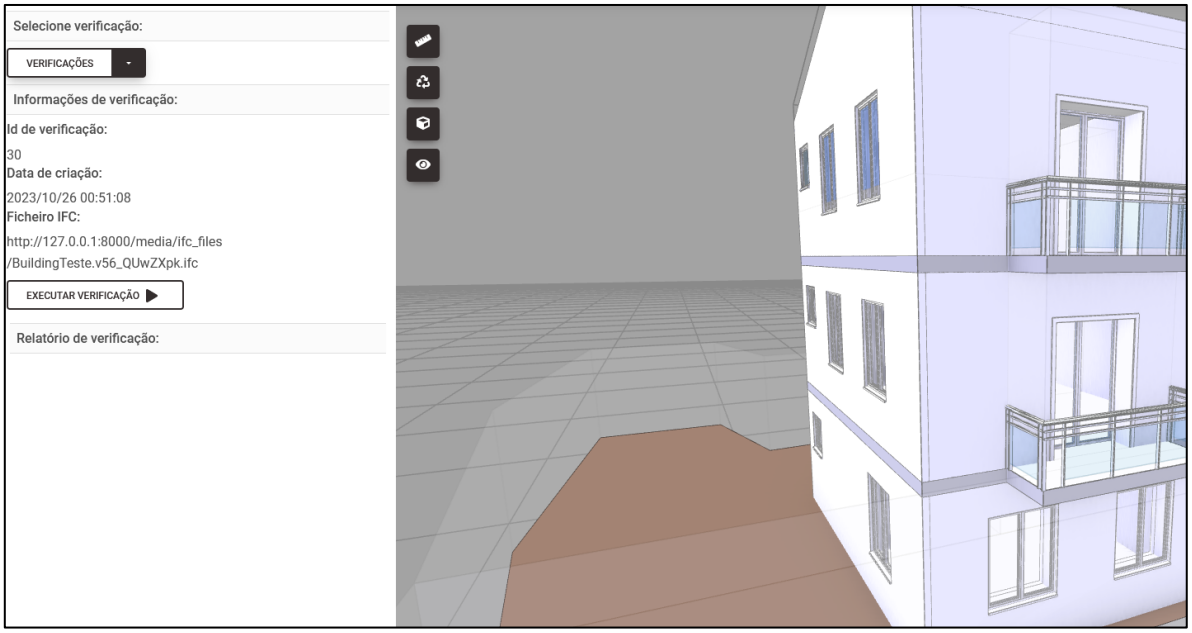

Figura 64: Informações sobre verificação selecionada.

Posteriormente à execução da verificação, o painel esquerdo é atualizado com os resultados da verificação, agrupados pelas regras verificadas. Como abordado anteriormente, em algumas regras a visualização do modelo IFC é interativa consoante o resultado da verificação, deste modo o resultado da verificação do RGEU, Artigo 67, paragrafo 1, demonstra que somente o fogo "AP1" está conforme com o regulamento e o relatório é complementado através do destaque do elemento na vista, como pode ser observado na Figura 65.

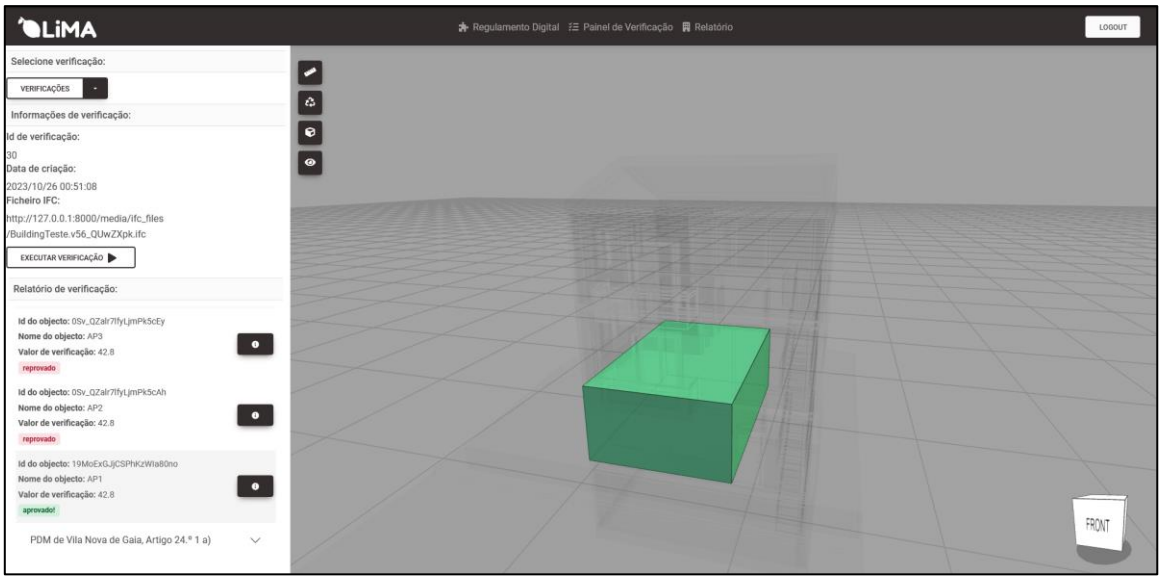

Figura 65: Relatório de verificação assistido por visualizador.

Já para a verificação das empenas, referente ao RPDML, Artigo 43º, parágrafo 1, foi possível verificar que a profundidade de empena estava correta, além de utilizar os recursos de medição do visualizador de forma a assistir o processo de verificação, como pode ser visto na Figura 66.

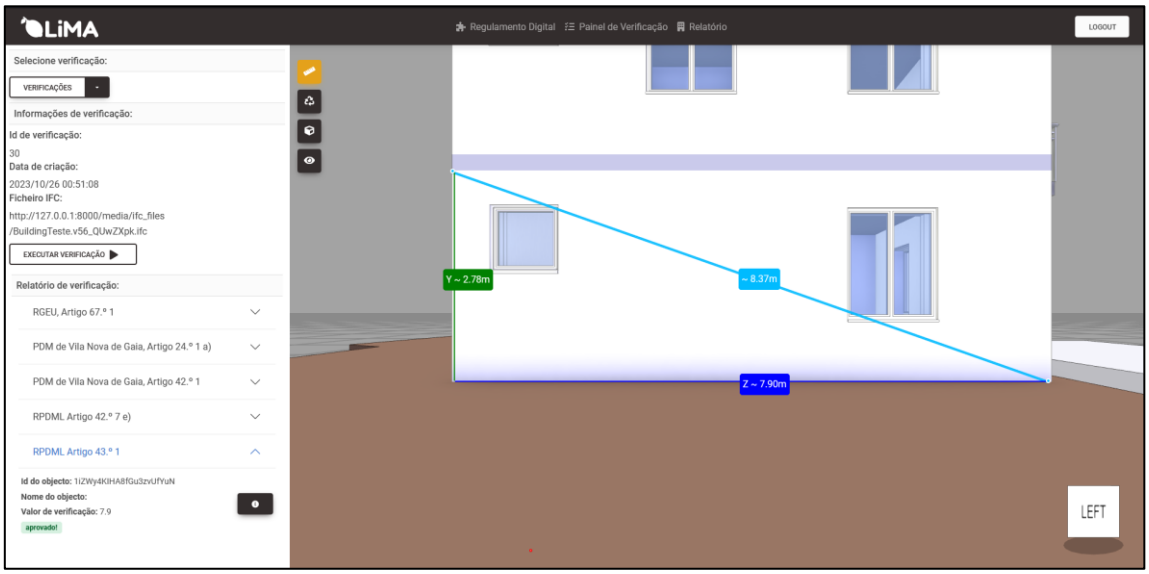

Figura 66: Verificação assistida através de ferramenta de medição.

Por último, a verificação da faixa de implantação, referente ao PDM de Vila Nova de Gaia, Artigo 42.º, parágrafo 1, demonstrou que o edifício representado no modelo IFC estava conforme, como pode ser visto na Figura 67.

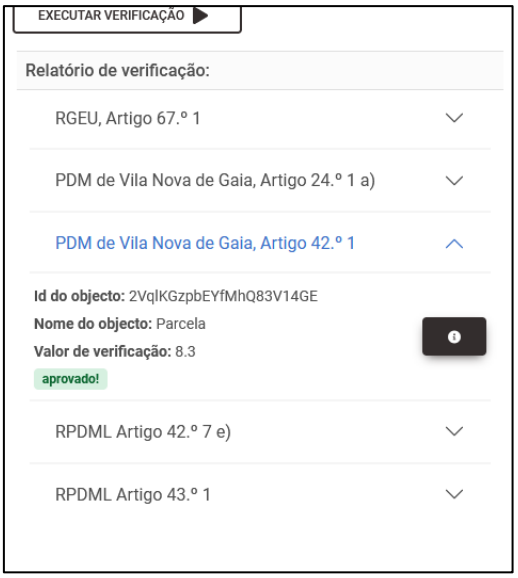

Figura 67: Relatório de verificação do protótipo.

5 Conclusões

5.1 Conclusões gerais

Este trabalho apresentou o desenvolvimento de uma plataforma protótipo que permite executar verificações de conformidade através de regulamentos interpretáveis por máquina que são criados através de uma linguagem de programação visual baseada em blocos. Este processo baseado em linguagem visual facilita a criação e edição destes regulamentos por pessoal não qualificado em programação, o que potencialmente permite que os técnicos nas câmaras possam criar e editar seus próprios regulamentos em um formato interpretável por máquina.

O processo para desenvolvimento da plataforma envolveu diversas atividades e desafios, a começar pelo estabelecimento de uma forma de traduzir os conceitos utilizados no licenciamento para a execução de verificação de conformidades em modelos IFC. Para isto, foi estabelecido um modelo de entidades em Python, que implementa os conceitos do licenciamento obtidos de algumas cláusulas analisadas através do método RASE e os correlaciona com o modelo IFC. Desta maneira, este modelo de entidades proporciona um formato interpretável por máquina dos conceitos utilizados, que podem então ser utilizados para criar os regulamentos em um formato interpretável por máquina e executar as verificações. Detetou-se que utilização do método RASE pode ajudar a obter os conceitos utlizados no licenciamento e desenvolver modelos conceptuais. Entretanto, a efetiva implementação destes modelos envolve um mapeamento com o modelo IFC, o que em diversas vezes não é feito através de uma relação 1:1.

Esta abordagem de desenvolvimento de um modelo de entidades para lidar com a diferença de conceitos é similar a algumas das abordagens que criam linguagens específicas de domínios e às abordagens que utilizam ontologias para este mesmo fim. Entretanto, este trabalho privilegiou a simplicidade do conceito a partir da implementação de uma série de classes na linguagem Python.

Para lidar com alguns dos desafios decorrentes das correlações foram utilizados recursos de processamento geométrico para implementar conceitos que não existem no modelo IFC, como foi feito para o simples conceito de profundidade demonstrado no capítulo 3. Desta forma, foi constatado que este processo deve ocorrer de forma muito cuidadosa e pode ser beneficiada pela consulta a especialistas nas Câmaras, de forma a traduzir corretamente os conceitos para parâmetros puramente geométricos. Além disso, foi possível verificar que a incapacidade dos softwares de criar modelos IFC

com todas as suas capacidades, p. ex. grupos de espaços e espaços parciais, limita as possíveis associações no mapeamento dos elementos do modelo de entidades do licenciamento com o modelo IFC, o que requer que uma efetiva aplicação seja feita a partir duma avaliação de como melhor executar estas correlações face as limitações de modelação. Além disso, a utilização de sistemas de classificação pode ajudar a estabelecer correlações, desde que os sistemas estejam preparados para representar os conceitos do licenciamento, o que deve ser melhor investigado.

Nos desenvolvimentos vinculados à plataforma, um dos grandes desafios foi a aprendizagem dos diversos recursos necessários para sua implementação, o que também influenciou a escolha por um âmbito de funcionalidades mais conciso e portanto exequível no tempo disponível para desenvolvimento do trabalho. Isto, entretanto, implica que uma implementação efetiva da plataforma requer um aprimoramento dos componentes desenvolvidos, principalmente para permitir uma melhor escalabilidade e controle de qualidade.

A linguagem de programação visual criada com ajuda da biblioteca Blockly permite a execução das verificações a partir da criação de código Python que implementa o modelo de entidades demonstrado no capítulo 3. Desta forma, este foi um dos principais fatores que influenciou as escolhas de conceção dos blocos desenvolvidos. A criação de métodos para a consulta destes modelos de entidades pode ajudar a conceber linguagens de bloco mais concisas e que não requeiram a execução baseada em séries de loops.

É importante estabelecer que o principal contributo deste trabalho é abordar o desenvolvimento de uma protótipo funcional, que pode servir de base para implementações de plataformas de verificação com codificação acessível a pessoal não qualificado em programação. O protótipo então pode ter seus componentes expandidos e aprimorados, já que foi desenvolvido através de ferramentas e de uma arquitetura escalável. Além disso, o protótipo pode servir de base para implementações além do âmbito do licenciamento, o que requer principalmente alteração do modelo de entidades demonstrado no capítulo 3 e consequentemente da linguagem de programação visual demonstrada no capítulo 4.

5.2 Desenvolvimentos futuros

O tempo para desenvolvimento da plataforma restringiu o desenvolvimento dos componentes e ferramentas desenvolvidos de uma forma global, de forma que segue uma lista de aprimoramentos

que podem ser implementados para futuros trabalhos, de maneira a proporcionar uma solução mais completa, poderosa e estável:

a) Desenvolvimento de um modelo de entidades do licenciamento a partir de uma análise abrangente do corpo regulamentar;
b) Desenvolvimento de métodos para consulta dos modelos de entidades do licenciamento;
c) Implementar rotinas para criação de blocos de forma automática a partir do modelo de entidades do licenciamento;
d) Aprimoramento das ferramentas baseadas em classes de alto nível para processamento geométrico;
e) Criação de protocolos para a geração de relatórios de verificação e visualização de resultados a partir de modelos IFC;
f) Inserção de modelos da cidade, por exemplo o CityGML, para ampliar a capacidade de executar verificações de regulamentos do ordenamento urbano;
g) Criação da possibilidade de confirmação e aprovação de relatórios pelos técnicos, com estabelecimento de um fluxo semiautomático de verificação de conformidade assistido por visualizadores;
h) Criação de testes automatizados e controle de erros.

6 Referências

Adi, P. D. P., & Kitagawa, A. (2019). A Review of the Blockly Programming on M5Stack Board and MQTT Based for Programming Education. *2019 IEEE 11th International Conference on Engineering Education (ICEED)*, 102–107. https://doi.org/10.1109/ICEED47294.2019.8994922

Al-Fedaghi, S. (2011). *Developing Web Applications*.

Análise da plataforma de verificação de regras de salvador, A Metropolis. (2022). *4º congresso português de 'Building Information Modelling' vol. 2 - ptBIM*, 178–187. https://doi.org/10.21814/uminho.ed.77.15

Arias Vanegas, J. S. (2022). *Automatic rule verification for digital building permits* [masterThesis]. https://repositorium.sdum.uminho.pt/handle/1822/83764

Bouzidi, K. R., Faron-Zucker, C., Fies, B., Corby, O., & Nhan, L.-T. (2013). *Towards a Semantic-based Approach for Modeling Regulatory Documents in Building Industry* (arXiv:1302.4811). arXiv. https://doi.org/10.48550/arXiv.1302.4811

Branco Pedro, J., Meijer, F. M., & Visscher, H. J. (2009). *The Portuguese Building Regulation System: A Critical Review*. http://repositorio.lnec.pt:8080/xmlui/handle/123456789/16726

Cerovsek, T. (2011). A review and outlook for a «Building Information Model» (BIM): A multi-standpoint framework for technological development. *Advanced Engineering Informatics*, *25*(2), 224–244. https://doi.org/10.1016/j.aei.2010.06.003

Cunningham, H. C. (2008). A little language for surveys: Constructing an internal DSL in Ruby. *Proceedings of the 46th Annual Southeast Regional Conference on XX*, 282–287. https://doi.org/10.1145/1593105.1593181

Daum, S., & Borrmann, A. (2015). *Simplifying the Analysis of Building Information Models Using tQL4BIM and vQL4BIM*.

Decreto-Lei n.º 80/2015, de 14 de maio do Ministério do Ambiente, Ordenamento do Território e Energia, Diário da República n.º 93/2015, Série I (2015). www.dre.pt

Decreto-Lei n.º 136/2014, de 9 de setembro do Ministério do Ambiente, Ordenamento do Território e Energia, Diário da República n.º 173/2014, Série I (2014). www.dre.pt

Decreto-Lei n.º 555/99, de 16 de dezembro | DR, (1999). www.dre.pt

Dimyadi, J. (2016). *Integrating the BIM Rule Language into Compliant Design Audit Processes*.

Donkers, S. (2013). *Automatic generation of CityGML LoD3 building models from IFC models*.

Eastman, C., Lee, J., Jeong, Y., & Lee, J. (2009). Automatic rule-based checking of building designs. *Automation in Construction*, *18*(8), 1011–1033. https://doi.org/10.1016/j.autcon.2009.07.002

Eastman, C. M. (Ed.). (2011). *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors* (2. ed). Wiley.

Eichler, C. C., Schranz, C., Krischmann, T., & Urban, H. (2023). *BIMcert Handbook 2023*.

Fenves, S. (1966). Tabular Decision Logic for Structural Design. *Journal of the Structural Division 92*, 473–490.

Fenves, S. J., Wright, R. N., Stahl, F. I., & Reed, K. A. (1987). *Introduction to SASE: Standards analysis, synthesis, and expression* (NBS IR 87-3513; 0 ed., p. NBS IR 87-3513). National Bureau of Standards. https://doi.org/10.6028/NBS.IR.87-3513

Fonseca, R., & Simões, A. (2007, fevereiro). *Alternativas ao XML: YAML e JSON*. https://repositorium.sdum.uminho.pt/handle/1822/6230

Forcier, J., Bissex, P., & Chun, W. J. (2008). *Python Web Development with Django*. Addison-Wesley Professional.

Gackenheimer, C. (2015). *Introduction to React*. Apress.

Ghimire, D. (2020). *Comparative study on Python web frameworks: Flask and Django* [Metropolia University of Applied Sciences]. http://www.theseus.fi/handle/10024/339796

Hjelseth, E., & Nisbet, N. (2011). *CAPTURING NORMATIVE CONSTRAINTS BY USE OF THE SEMANTIC MARK-UP RASE METHODOLOGY* (pp. 26–28).

Holovaty, A., & Kaplan-Moss, J. (2009). *The Definitive Guide to Django: Web Development Done Right*. Apress.

International Organization for Standardization. (2019). *ISO 19650-1:2018*. https://www.iso.org/standard/68078.html

Kim, H., Lee, J. K., Shin, J., & Choi, J. (2019). Visual language approach to representing KBimCode-based Korea building code sentences for automated rule checking. *Journal of Computational Design and Engineering*, *6*(2), 143–148. https://doi.org/10.1016/J.JCDE.2018.08.002

Laakso, M., & Kiviniemi, A. O. (2012). *The IFC standard: A review of History, development, and standardization, Information Technology*. http://www.itcon.org

Langlois, B., Jitia, C.-E., & Jouenne, E. (2007). *DSL Classification*. https://www.semanticscholar.org/paper/DSL-Classification-Langlois-Jitia/61134c1ce75a8985e36df559db2442cda8595f7b

Lee, J. K. (2011). *Building environment rule and analysis (BERA) language and its application for evaluating building circulation and spatial program*. https://smartech.gatech.edu/handle/1853/39482

Lee, J. M. (2010). *Automated checking of building requirements on circulation over a range of design phases*. https://smartech.gatech.edu/handle/1853/34802

Li, L., & Chou, W. (2011). Design and Describe REST API without Violating REST: A Petri Net Based Approach. *2011 IEEE International Conference on Web Services*, 508–515. https://doi.org/10.1109/ICWS.2011.54

Loshin, D. (2010). *Master Data Management*. Morgan Kaufmann.

Macit, S., Ilal, M. E., Günaydın, H. M., İlal, M. E., Günaydın, H. M., & Suter, G. (2013). İzmir municipality housing and zoning code analysis and representation for compliance checking. *20th Workshop of the European Group for Intelligent Computing in Engineering*. https://www.researchgate.net/publication/259763143

Malsane, S., Matthews, J., Lockley, S., Love, P. E. D., & Greenwood, D. (2015). Development of an object model for automated compliance checking. *Automation in Construction*, *49*(PA), 51–58. https://doi.org/10.1016/j.autcon.2014.10.004

Martins, J. P., & Monteiro, A. (2013). LicA: A BIM based automated code-checking application for water distribution systems. *Automation in Construction*, *29*, 12–23. https://doi.org/10.1016/j.autcon.2012.08.008

Masse, M. (2011). *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O'Reilly Media, Inc.

Nawari, N. O. (2019). A Generalized Adaptive Framework (GAF) for automating code compliance checking. *Buildings*, *9*(4). https://doi.org/10.3390/buildings9040086

Noardo, F., Ellul, C., Harrie, L., Overland, I., Shariat, M., Ohori, K. A., & Stoter, J. (2020). Opportunities and challenges for GeoBIM in Europe: Developing a building permits use-case to raise awareness and examine technical interoperability challenges. *Journal of Spatial Science*, *65*(2), 209–233. https://doi.org/10.1080/14498596.2019.1627253

Noardo, F., Guler, D., Fauth, J., Malacarne, G., Ventura, S. M., Azenha, M., Olsson, O., & Senger, L. (2021). *Unveiling the actual progress of Digital Building Permit: Getting awareness through a critical state of the art review*. https://www.bimacademy.global/insights/infrastructure/the-golden-thread-of-information-putting-the-hacki

Nurminen, J. K., Wikman, J., Kokkinen, H., Muilu, P., & Futurice, M. H. (2008). *Mobile Web Application Development Stack*. https://doi.org/10.1109/ccnc08.2007.291

Nyman, D. J., Fenves, S. J., & Wright, R. N. (1973). Restructuring Study of the AISC Specification. *Civil Engineering Studies SRS-393*. https://hdl.handle.net/2142/13806

Oliveira, F., Neves, M. J., & Lopes, D. (2016). *Regime Jurídico da Urbanização e Edificação—Comentado* (4a ed.). Almedina. https://www.wook.pt/livro/regime-juridico-da-urbanizacao-e-edificacao-dulce-lopes/18883703

Olsson, P. O., Axelsson, J., Hooper, M., & Harrie, L. (2018). Automation of building permission by integration of BIM and geospatial data. *ISPRS International Journal of Geo-Information*, *7*(8). https://doi.org/10.3390/ijgi7080307

Park, S., Lee, Y.-C., & Lee, J.-K. (2016). Definition of a domain-specific language for Korean building act sentences as an explicit computable form. *Journal of Information Technology in Construction*, *21*, 422–433. Scopus.

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van De Walle, R., & Van Campenhout, J. (2011). A semantic rule checking environment for building performance checking. *Automation in Construction*, *20*(5), 506–518. https://doi.org/10.1016/j.autcon.2010.11.017

Pedro, J. B., Meijer, F., & Visscher, H. (2011). *Comparison of building permit procedures in European in European Union countries Cova da Moura Rehabilitation Needs Assessment View project Building Regulations for Construction Works in Existing Buildings View project Comparison of building permit procedures in European Union countries*. https://www.researchgate.net/publication/257527312

Plazza, D., Röck, M., Malacarne, G., Passer, A., Marcher, C., & Matt, D. T. (2019). BIM for public authorities: Basic research for the standardized implementation of BIM in the building permit process. *IOP Conference Series: Earth and Environmental Science*, *323*(1). https://doi.org/10.1088/1755-1315/323/1/012102

Preidel, C., & Borrmann, A. (2015). Automated code compliance checking based on a visual language and building information modeling. *32nd International Symposium on Automation and Robotics in Construction and Mining: Connected to the Future, Proceedings*. https://doi.org/10.22260/ISARC2015/0033

Proposta de Lei 77/XV/1, DAR II série A n.º 215, 2023.05.02, da 1.ª SL da XV Leg (pág. 17-60] (2023). www.parlamento.pt

Decreto-Lei n.º 38382, de 7 de agosto do Ministério das Obras Públicas—Gabinete do Ministro, Diário do Governo n.º 166/1951, 1º Suplemento, Série I (1951). www.dre.pt

Santos, M. F. de S. (2021). *Metodologias BIM para verificação regulamentar em contexto de licenciamento municipal: Proposta, implementação e aplicação* [masterThesis]. https://repositorium.sdum.uminho.pt/handle/1822/76900

Shahi, K., McCabe, B. Y., & Shahi, A. (2019). Framework for Automated Model-Based e-Permitting System for Municipal Jurisdictions. *Journal of Management in Engineering*, *35*(6), 04019025. https://doi.org/10.1061/(asce)me.1943-5479.0000712

Shehzad, H. M. F., Ibrahim, R. B., Yusof, A. F., khaidzir, K. A. M., Shawkat, S., & Ahmad, S. (2020). Recent developments of BIM adoption based on categorization, identification and factors: A systematic literature review. *International Journal of Construction Management*. https://doi.org/10.1080/15623599.2020.1837719

Solihin, W. (2015). *A simplified BIM data representation using a relational database schema for an efficient rule checking system and its associated rule checking language* [Georgia Institute of Technology]. https://www.semanticscholar.org/paper/A-simplified-BIM-data-representation-using-a-schema-Solihin/1b62897fa23d7d9f204b72d265c916c9f19eaa8c

Solihin, W., Dimyadi, J., & Lee, Y.-C. (2019). In Search of Open and Practical Language-Driven BIM-Based Automated Rule Checking Systems. Em I. Mutis & T. Hartmann (Eds.), *Advances in Informatics and Computing in Civil and Construction Engineering* (pp. 577–584). Springer International Publishing. https://doi.org/10.1007/978-3-030-00220-6_69

Solihin, W., & Eastman, C. (2016). *A knowledge representation approach in BIM rule requirement analysis using the conceptual graph*. *21*(CIB W78 2015 Special track on Compliance Checking), 371.

Soliman-Junior, J., Tzortzopoulos, P., Baldauf, J. P., Pedo, B., Kagioglou, M., Formoso, C. T., & Humphreys, J. (2021). Automated compliance checking in healthcare building design. *Automation in Construction*, *129*. https://doi.org/10.1016/j.autcon.2021.103822

Song, J., Kim, J., & Lee, J.-K. (2019). *Converting KBImCode into an executable code for the automated design rule checking system*.

Tan, X., Hammad, A., & Fazio, P. (2010). Automated Code Compliance Checking for Building Envelope Design. *Journal of Computing in Civil Engineering*, *24*(2), 203–211. https://doi.org/10.1061/(ASCE)0887-3801(2010)24:2(203)

Torreão, R., & Novais, L. (2020). *Manual de Boas Práticas Plataforma Metropolis*. https://cdn-sedur.metropolis.solutions/docs/Manual%20de%20Boas%20Pr%C3%A1ticas%20-%20Plataforma%20Metropolis.pdf

Tsuzuki, M. D. S. G., Takase, F. K., Garcia, M. A. S., & Martins, T. D. C. (2007). Converting CSG models into meshed B-Rep models using euler operators and propagation based marching cubes. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *29*(4), 337–344. https://doi.org/10.1590/S1678-58782007000400001

Wülfing, A., Windisch, R., & Scherer, R. (2014). A visual BIM query language. Em A. Mahdavi, B. Martens, & R. Scherer (Eds.), *eWork and eBusiness in Architecture, Engineering and Construction* (pp. 157–164). CRC Press. https://doi.org/10.1201/b17396-30

Yurchyshyna, A., & Zarli, A. (2009). An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. *Automation in Construction*, *18*(8), 1084–1098. https://doi.org/10.1016/j.autcon.2009.07.008

Zhang, R., & El-Gohary, N. (2020). A Machine-Learning Approach for Semantic Matching of Building Codes and Building Information Models (BIMs) for Supporting Automated Code Checking. *Sustainable Civil Infrastructures*, 64–73. https://doi.org/10.1007/978-3-030-34216-6_5

Zhang, Z., Nisbet, N., Ma, L., & Broyd, T. (2023). Capabilities of rule representations for automated compliance checking in healthcare buildings. *Automation in Construction*, *146*, 104688. https://doi.org/10.1016/j.autcon.2022.104688

Zhao, X. (2017). A scientometric review of global BIM research: Analysis and visualization. *Automation in Construction*, *80*, 37–47. https://doi.org/10.1016/J.AUTCON.2017.04.002

ANEXO I: Marcação RASE

<table>
<tr><th>Cláusula</th><th>Texto</th></tr>
<tr><td>RGEU, Artigo 46.º 7</td><td><R>Os <a>degraus das escadas das edificações para <s>habitação colectiva </s>terão a <r>largura (cobertor) mínima de 0,25 m </r>e a <r>altura (espelho) máxima de 0,193 m</r>. <E>No entanto, <s>nos edifícios de três, quatro ou cinco pisos</s> e <s>sempre que não seja instalado ascensor</s>, a <r>largura (cobertor) mínima será de 0,280m</r> e a <r>altura (espelho) máxima será de 0,175m</r>.</E></R></td></tr>
<tr><td>RGEU, Artigo 65.º 1</td><td><R>A <r>altura mínima</r>, <a>piso a piso, em <s>edificações destinadas à habitação</s> <r>é de 2,70m (27m)</r>, <r>não podendo ser o pé-direito livre mínimo inferior a 2,40 m (24m)</r>.</R></td></tr>
<tr><td>RGEU, Artigo 65.º 2</td><td><E>Excepcionalmente, em <a>vestíbulos, <a>corredores, <a>instalações sanitárias, <a>despensas e <a>arrecadações <r>será admissível que o pé-direito se reduza ao mínimo de 2,20m (22m)</r>.</E></td></tr>
<tr><td>RGEU, Artigo 66.º 1</td><td><R>Os compartimentos de habitação não poderão ser em número e área inferiores aos indicados no quadro seguinte:
<table>
<tr><th></th><th colspan="8">número de compartimentos por fogo</th></tr>
<tr><th></th><th><r>2</r>
<a>T0</th><th><r>3</r>
<a>T1</th><th><r>4</r>
<a>T2</th><th><r>5</r>
<a>T3</th><th><r>6</r>
<a>T4</th><th><r>7</r>
<a>T5</th><th><r>8</r>
<a>T6</th><th><r>Mais de 8</r>
<a>Tx > 6</th></tr>
<tr><th></th><th colspan="8">áreas em metros quadrados</th></tr>
<tr><td><a>Quarto casal....</td><td>—</td><td><r>10,5</r></td><td><r>10,5</r></td><td><r>10,5</r></td><td><r>10,5</r></td><td><r>10,5</r></td><td><r>10,5</r></td><td><r>10,5</r></td></tr>
<tr><td><a>Quarto duplo...</td><td>—</td><td>—</td><td><r>9</r></td><td><r>9</r></td><td><r>9</r></td><td><r>9</r></td><td><r>9</r></td><td></td></tr>
<tr><td><a>Quarto duplo...</td><td>—</td><td>—</td><td>—</td><td><r>9</r></td><td><r>9</r></td><td><r>9</r></td><td><r>9</r></td><td></td></tr>
<tr><td><a>Quarto duplo...</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td><r>9</r></td><td><r>9</r></td><td><r>restantes quartos 9m2</r></td></tr>
<tr><td><a>Quarto simples</td><td>—</td><td>—</td><td>—</td><td>—</td><td><r>6,5</r></td><td><r>6,5</r></td><td><r>6,5</r></td><td><r>6,5</r></td></tr>
<tr><td><a>Quarto simples</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td><r>6,5</r></td><td><r>6,5</r></td></tr>
<tr><td><a>Sala.....</td><td><r>10</r></td><td><r>10</r></td><td><r>12</r></td><td><r>12</r></td><td><r>12</r></td><td><r>16</r></td><td><r>16</r></td><td><r>16</r></td></tr>
<tr><td><a>Cozinha...</td><td><r>6</r></td><td><r>6</r></td><td><r>6</r></td><td><r>6</r></td><td><r>6</r></td><td><r>6</r></td><td><r>6</r></td><td><r>6</r></td></tr>
<tr><td><a>Suplemento de área obrigatório..</td><td><r>6</r></td><td><r>4</r></td><td><r>6</r></td><td><r>8</r></td><td><r>8</r></td><td><r>8</r></td><td><r>10</r></td><td><r>(x+4)m2</r> (x= n.º de quartos)</td></tr>
</table>
</R></td></tr>
<tr><td>RGEU, Artigo 67.º 1</td><td><R>As áreas brutas dos fogos terão os seguintes valores mínimos:
<table>
<tr><th></th><th colspan="8">Tipo de fogo</th></tr>
<tr><th></th><th><a>T0</th><th><a>T1</th><th><a>T2</th><th><a>T3</th><th><a>T4</th><th><a>T5</th><th><a>T6</th><th><a>Tx > 6</th></tr>
<tr><td>área bruta em metros quadrados.....</td><td><r>35</r></td><td><r>52</r></td><td><r>72</r></td><td><r>91</r></td><td><r>105</r></td><td><r>122</r></td><td><r>134</r></td><td><r>1,6 x Ah</r></td></tr>
</table></td></tr>
</table>

	</R>
PDM de Vila Nova de Gaia, Artigo 24.º 1 a)	<R>As <a>instalações directamente adstritas às explorações agrícolas, pecuárias ou florestais, devem respeitar as seguintes condições de edificabilidade: a) As <r>edificações devem ter uma cércea máxima 2 pisos </r>e uma <r>altura máxima de 6,5m</r>, <E>salvo <a>instalações técnicas <r>devidamente justificadas</r>;</E></R>
PDM de Vila Nova de Gaia, Artigo 42.º 1	<R>As <a>novas construções principais <r>implantar-se-ão dentro de uma faixa de 35m, confinante com o espaço público</r>, sem prejuízo do previsto nos números seguintes.</R>
RPDML Artigo 42.º 7 e)	<R><r>Índice de edificabilidade</r>, em <a>parcelas com uma profundidade superior a 14 metros e/ou com <a>uma área de lote ou parcela superior a 130m2: i) <r>1,0</r> em <s>lote ou parcela com área inferior a 150m2</s>; ii) <r>0,7 </r>em <s>lote ou parcela com área igual ou superior a 150m2</s>, sendo sempre permitido um <r>mínimo de 150m2 de superfície de pavimento.</r></R>
RPDML Artigo 43.º 1	<R>Sem prejuízo do disposto nos números seguintes, a <r>profundidade máxima das empenas</r>, <E>sem considerar as <a>varandas e os <a>corpos balançados</E>, é de <r>15 metros</r>, com <E>exceção dos <a>estabelecimentos hoteleiros e <a>equipamentos de utilização coletiva, cuja <r>empena pode atingir os 18 metros</r></E>.</R>

ANEXO II: Sistematização das frases métricas através do método RASE

RGEU, Artigo 46.º 7

Frase métrica	Tipo	Objeto	Propriedade	Comparação	Alvo	Unidade
degraus das escadas das edificações	aplicabilidade	escada	tipo	=	True	
habitação colectiva	seleção	edifício	categoria	=	habitação coletiva	
largura (cobertor) mínima de 0,25 m	requisito	escada	largura do degrau	>=	0,25	m
altura (espelho) máxima de 0,193 m	requisito	escada	altura do degrau	<=	0,193	m
nos edifícios de três, quatro ou cinco pisos	seleção	edifício	número de pisos	=	3	
					4	
					5	
sempre que não seja instalado ascensor	seleção	ascensor	tipo	=	False	
largura (cobertor) mínima será de 0,280m	requisito	escada	largura do degrau	>=	0,28	m
altura (espelho) máxima será de 0,175m	requisito	escada	altura do degrau	<=	0,175	m

RGEU, Artigo 65.º 1 e 2

Frase métrica	Tipo	Objeto	Propriedade	Comparação	Alvo	Unidade
piso a piso	aplicabilidade	piso (pavimento)	tipo	=	True	
edificações destinadas a habitação	seleção	edificação	categoria	=	habitação coletiva	
					moradia unifamiliar	

altura mínima é de 2,70m	requisito	piso (pavimento)	altura piso a piso	>=	2,7	m
não podendo ser o pé-direito livre mínimo inferior a 2,4 m	requisito	espaço	pé-direito	>=	2,4	m
vestíbulos	aplicabilidade	espaço	classe	=	vestíbulo	
corredores	aplicabilidade	espaço	classe	=	corredor	
instalações sanitárias	aplicabilidade	espaço	classe	=	instalação sanitária	
despensas	aplicabilidade	espaço	classe	=	despensa	
arrecadações	aplicabilidade	espaço	classe	=	arrecadação	
será admissível que o pé-direito se reduza ao mínimo de 2,20m	requisito	espaço	pé-direito	>=	2,2	m

RGEU, Artigo 66.º 1

Frase métrica	Tipo	Objeto	Propriedade	Comparação	Alvo	Unidade
T0	aplicabilidade	fogo	número de quartos	=	0	
2	requisito	fogo	número de compartimentos	=	2	
Sala	aplicabilidade	espaço	classe	=	sala	
10	requisito	espaço	área	>=	10,0	m^2
Cozinha	aplicabilidade	espaço	classe	=	cozinha	
6	requisito	espaço	área	>=	6,0	m^2
Suplemento de área obrigatório	aplicabilidade	espaço	suplemento de área	=	True	
6	requisito	espaço	área	>=	6,0	m^2
T1	aplicabilidade	fogo	número de quartos	=	1	
3	requisito	fogo	número de	=	3	

			compartimentos			
Quarto casal	aplicabilidade	espaço	classe	=	quarto casal	
10,5	requisito	espaço	área	>=	10,5	m²
10	requisito	espaço	área	>=	10,0	m²
6	requisito	espaço	área	>=	6,0	m²
4	requisito	espaço	área	>=	4,0	m²
T2	aplicabilidade	fogo	número de quartos	=	2	
4	requisito	fogo	número de compartimentos	=	4	
Quarto duplo	aplicabilidade	espaço	classe	=	quarto duplo	
10,5	requisito	espaço	área	>=	10,5	m²
9	requisito	espaço	área	>=	9,0	m²
12	requisito	espaço	área	>=	12,0	m²
6	requisito	espaço	área	>=	6,0	m²
6	requisito	espaço	área	>=	6,0	m²
T3	aplicabilidade	fogo	número de quartos	=	3	
5	requisito	fogo	número de compartimentos	=	5	
10,5	requisito	espaço	área	>=	10,5	m²
9	requisito	espaço	área	>=	9,0	m²
9	requisito	espaço	área	>=	9,0	m²
12	requisito	espaço	área	>=	12,0	m²
6	requisito	espaço	área	>=	6,0	m²
8	requisito	espaço	área	>=	8,0	m²
T4	aplicabilidade	fogo	número de quartos	=	4	
6	requisito	fogo	número de compartimentos	=	6	
10,5	requisito	espaço	área	>=	10,5	m²

9	requisito	espaço	área	>=	9,0	m²
9	requisito	espaço	área	>=	9,0	m²
Quarto simples	aplicabilidade	espaço	classe	=	quarto simples	
6,5	requisito	espaço	área	>=	6,5	m²
12	requisito	espaço	área	>=	12,0	m²
6	requisito	espaço	área	>=	6,0	m²
8	requisito	espaço	área	>=	8,0	m²
T5	aplicabilidade	fogo	número de quartos	=	5	
7	requisito	fogo	número de compartimentos	=	7	
10,5	requisito	espaço	área	>=	10,5	m²
9	requisito	espaço	área	>=	9,0	m²
9	requisito	espaço	área	>=	9,0	m²
9	requisito	espaço	área	>=	9,0	m²
6,5	requisito	espaço	área	>=	6,5	m²
16	requisito	espaço	área	>=	16,0	m²
6	requisito	espaço	área	>=	6,0	m²
8	requisito	espaço	área	>=	8,0	m²
T6	aplicabilidade	fogo	número de quartos	=	6	
8	requisito	fogo	número de compartimentos	=	8	
10,5	requisito	espaço	área	>=	10,5	m²
9	requisito	espaço	área	>=	9,0	m²
9	requisito	espaço	área	>=	9,0	m²
9	requisito	espaço	área	>=	9,0	m²
6,5	requisito	espaço	área	>=	6,5	m²
6,5	requisito	espaço	área	>=	6,5	m²
16	requisito	espaço	área	>=	16,0	m²

6	requisito	espaço	área	>=	6,0	m²
10	requisito	espaço	área	>=	10,0	m²
T > 6	aplicabilidade	fogo	número de quartos	>	6	
Mais de 8	requisito	fogo	número de compartimentos	>	8	
10,5	requisito	espaço	área	>=	10,5	m²
restantes quartos 9m2	requisito	espaço	área	>=	9,0	m²
6,5	requisito	espaço	área	>=	6,5	m²
6,5	requisito	espaço	área	>=	6,5	m²
16	requisito	espaço	área	>=	16,0	m²
6	requisito	espaço	área	>=	6,0	m²
(x+4)m2	requisito	espaço	área	>=	número de quartos + 4,0	m²

RGEU, Artigo 67.º 1

Frase métrica	Tipo	Objeto	Propriedade	Comparação	Alvo	Unidade
T0	aplicabilidade	fogo	número de quartos	=	0	
35	requisito	fogo	área	>=	35,0	m²
T1	aplicabilidade	fogo	número de quartos	=	1	
52	requisito	fogo	área	>=	52,0	m²
T2	aplicabilidade	fogo	número de quartos	=	2	
72	requisito	fogo	área	>=	72,0	m²
T3	aplicabilidade	fogo	número de quartos	=	3	
91	requisito	fogo	área	>=	91,0	m²
T4	aplicabilidade	fogo	número de quartos	=	4	
105	requisito	fogo	área	>=	105,0	m²
T5	aplicabilidade	fogo	número de quartos	=	5	

122	requisito	fogo	área	>=	122,0	m²
T6	aplicabilidade	fogo	número de quartos	=	6	
134	requisito	fogo	área	>=	134,0	m²
Tx>6	aplicabilidade	fogo	número de quartos	>	6	
1,6xAh	requisito	fogo	área	>=	1,6 x Área habitável	m²

PDM de Vila Nova de Gaia, Artigo 24.º 1 a)

Frase métrica	Tipo	Objeto	Propriedade	Comparação	Alvo	Unidade
instalações directamente adstritas às explorações agrícolas, pecuárias ou florestais	aplicabilidade	edifício	usos	inclui	exploração agrícola	
					exploração pecuária	
					exploração florestal	
edificações devem ter uma cércea máxima 2 pisos	requisito	edifício	número de pisos	<=	2	
altura máxima de 6,5m	requisito	edifício	altura	<=	6,5	m
instalações técnicas	aplicabilidade	edifício	usos	inclui	instalação técnica	
devidamente justificadas	requisito	análise técnica				

PDM de Vila Nova de Gaia, Artigo 42.º 1

Frase métrica	Tipo	Objeto	Propriedade	Comparação	Alvo	Unidade
novas construções	aplicabilidade	edificação	nova construção	=	True	
implantar-se-ão dentro de uma faixa de 35m, confinante com o espaço público	requisito	edificação	implantação	contida em	faixa de 35m, confinante com o espaço público	

RPDML Artigo 42.º 7 e)

Frase métrica	Tipo	Objeto	Propriedade	Comparação	Alvo	Unidade
parcelas com uma profundidade superior a 14 metros	aplicabilidade	parcela	profundidade	>=	14,0	m
uma área de lote ou parcela superior a 130m2	aplicabilidade	parcela	área	>	130,0	m^2
lote ou parcela com área inferior a 150m2	seleção	parcela	área	<	150,0	m^2
Índice de edificabilidade... 1,0	requisito	parcela	índice de edificabilidade	<=	1,0	m^2/m^2
lote ou parcela com área igual ou superior a 150m2	seleção	parcela	área	>	150,0	m^2
Índice de edificabilidade... 0,7	requisito	parcela	índice de edificabilidade	<=	0,7	m^2/m^2
mínimo de 150m2 de superfície de pavimento.	requisito	parcela	superfície de pavimento	>=	150	m^2

RPDML Artigo 43.º 1

Frase métrica	Tipo	Objeto	Propriedade	Comparação	Alvo	Unidade
varandas	aplicabilidade	espaço	classe	=	varanda	
corpos balançados	aplicabilidade	elemento construtivo	é componente de corpo balançado	=	True	
profundidade máxima das empenas ... 15 metros	requisito	empena	profundidade	<=	15,0	m
estabelecimentos hoteleiros	aplicabilidade	edificação	categoria	=	estabelecimento hoteleiro	
equipamentos de utilização coletiva	aplicabilidade	edificação	categoria	=	equipamento de utilização coletiva	

empena pode atingir 18 metros	requisito	empena	profundidade	<=	18,0	m

Anexo III: Estrutura de componentes do frontend