DESENVOLVIMENTO DE APLICATIVOS EDUCACIONAIS PARA QUÍMICA: FERRAMENTAS PARA O APRENDIZADO DE CONFIGURAÇÃO ELETRÔNICA E GEOMETRIA MOLECULAR

ÉLISSON MICHAEL FERNANDES MEIRELLES ARAÚJO

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE – UENF

CAMPOS DOS GOYTACAZES – RJ

AGOSTO – 2019

DESENVOLVIMENTO DE APLICATIVOS EDUCACIONAIS PARA QUÍMICA: FERRAMENTAS PARA O APRENDIZADO DE CONFIGURAÇÃO ELETRÔNICA E GEOMETRIA MOLECULAR

ÉLISSON MICHAEL FERNANDES MEIRELLES ARAÚJO

Tese apresentada ao Centro de Ciência e Tecnologia, da Universidade Estadual do Norte Fluminense Darcy Ribeiro, como parte das exigências para obtenção de título de Doutor em Engenharia e Ciência dos Materiais.

Orientador: Prof. D.Sc. Angelus Giuseppe Pereira da Silva

CAMPOS DOS GOYTACAZES – RJ

AGOSTO – 2019

AGOSTO – 2019

DESENVOLVIMENTO DE APLICATIVOS EDUCACIONAIS PARA QUÍMICA: FERRAMENTAS PARA O APRENDIZADO DE CONFIGURAÇÃO ELETRÔNICA E GEOMETRIA MOLECULAR

ÉLISSON MICHAEL FERNANDES MEIRELLES ARAÚJO

Tese apresentada ao Centro de Ciência e Tecnologia, da Universidade Estadual do Norte Fluminense Darcy Ribeiro, como parte das exigências para obtenção de título de Doutor em Engenharia e Ciência dos Materiais.

Aprovada em ___/___/_____.

Comissão Examinadora:

_____

Prof. Ítalo de Oliveira Matias (D.Sc., Sistemas Computacionais) UCAM

_____

Prof. Leonard Barreto Moreira (D.Sc., Modelagem Computacional) UCAM

_____

Prof.ª Annabell Del Real Tamariz (D.Sc., Engenharia Elétrica) UENF

_____

Prof. Fabrício Barros Gonçalves (D.Sc., Engenharia de Sistemas e Computação) IFF

_____

Prof. Angelus Giuseppe Pereira da Silva (D.Sc., Ciências Técnicas) UENF
Orientador

RESUMO

Dispositivos eletrônicos estão cada vez mais econômicos e acessíveis, o que por sua vez, facilita a disseminação de tecnologias da informação como dispositivos de massa. Dentro desse cenário, o desenvolvimento de recursos que permitam o aproveitamento desses dispositivos como ferramentas é um passo natural e necessário, para explorar novas formas de assimilação de conhecimento. O objetivo geral deste trabalho é o desenvolvimento de aplicativos relacionados a conteúdos de química. Especificamente, os objetivos são o desenvolvimento de dois aplicativos, denominados “Configuração Eletrônica” e “Geometria Molecular”. O primeiro permite ao usuário fazer a distribuição eletrônica de qualquer átomo da tabela periódica. O segundo aplicativo aborda conceitos acerca de ligações covalentes, hibridização e a geometria envolvida em cada tipo de ligação, oferecendo recursos gráficos que permitem a visualização 3D da configuração geométrica da estrutura. Os aplicativos foram desenvolvidos em C#, usando o paradigma da programação OO e uso do motor de jogo Unity®. As aplicações possuem versões para Android® e Windows®, e podem ser executadas em dispositivos móveis e *desktops*, respectivamente. Estão disponíveis gratuitamente para *download* nas lojas virtuais da Google® e Microsoft®. A publicação foi sucedida da elaboração de formulários de avaliação que pudessem validar as ferramentas de forma quantitativa. Os usuários receberam notificações, com links para os questionários, na tela inicial de cada aplicativo e manifestaram sua concordância com 15 afirmações, assinalando um número em uma escala de 1-5, sendo que o valor 1 significava “discordância plena” e o valor 5 “concordância plena”. Um total de 115 formulários foram preenchidos e foi calculado a média de cada afirmação, assim como uma média total, por aplicativo. Ambos os aplicativos ficaram com uma média igual ou superior a 4 em todas as perguntas, permitindo-se afirmar que foram bem aceitos. A média total do aplicativo Geometria Molecular foi 4,34 e o do Configuração Eletrônica foi 4,49. Relatórios da *Google Play* informaram que houve um total de mais de 5.370 instalações dos aplicativos. Em 29/08/19, existiam mais de 1.136 dispositivos com pelo menos um dos softwares instalados.

Palavras chave: Aplicativo, Química, Configuração Eletrônica, Geometria Molecular

ABSTRACT

Electronic devices are becoming increasingly available and affordable, which in turn facilitates the spread of information technologies as mass devices. Within this scenario, the development of resources that allow the use of these devices as tools is a natural and necessary step to explore new ways of knowledge assimilation. The general objective of this work is the development of applications related to chemistry content. Specifically, the objectives are the development of two applications, called “Configuração Eletrônica” and “Geometria Molecular”. The first allows the user to electronically distribute any atom in the periodic table. The second application addresses concepts about covalent bonding, hybridization, and the geometry involved in each bonding type, providing graphical features that enable 3D visualization of the geometric configuration of the structure. The applications were developed in C# using OO programming paradigm and use of the Unity® game engine. Both have versions for Android® and Windows®, and can run on mobile devices and desktops, respectively. They are available free for download at Google® Play and Microsoft® stores. The publication was followed with the elaboration of evaluation forms that could validate the tools quantitatively. Users received notifications, with links to the questionnaires, on each app's home screen and agreed with 15 statements, marking a number on a scale of 1-5, with a value of 1 meaning “full disagreement” and a value of 5 "full agreement". A total of 115 forms were completed and the average of each statement was calculated, as well as a total average per application. Both applications scored 4 or more on all questions, allowing us to state that they were well accepted. The total average of the “Geometria Molecular” application was 4.34 and of “Configuração Eletrônica” was 4.49. Reports from Google Play informed that there was a total of over 5,370 downloads. On 08/29/19, there were over 1,136 devices with at least one of the softwares installed.

Keywords: Software, Chemistry, Electron Configuration, Molecular Geometry

LISTA DE FIGURAS

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

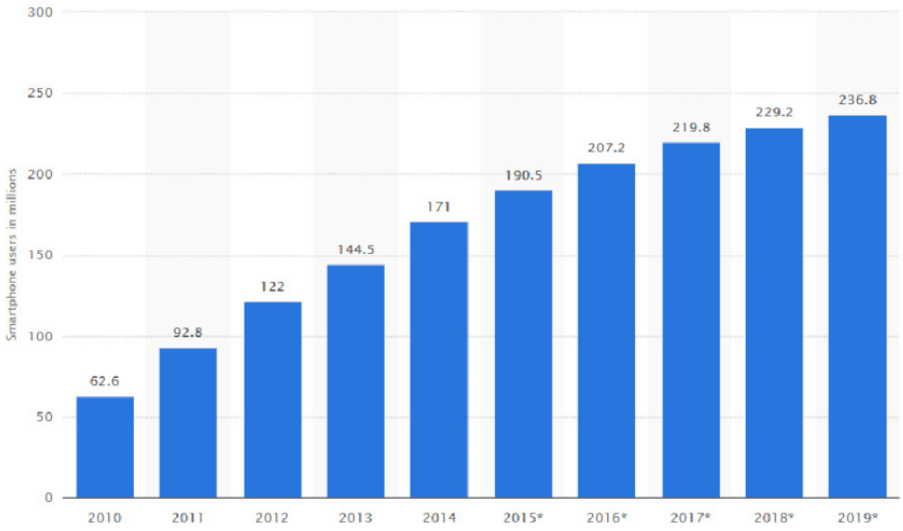
O estudo de alguns conteúdos de química que envolve abstrações geométricas é complexo, pela dificuldade que os professores têm de descrever o assunto com os recursos limitados de sala de aula, ao mesmo tempo em que alunos têm obstáculos ao interpretar a descrição limitada oferecida pelo professor. Diante desse cenário, é importante que o professor procure recursos para facilitar o aprendizado e tornar as aulas mais dinâmicas. Um recurso poderoso atualmente é o smartphone, que oferece oportunidades de inovação no processo educacional. (NETO; DA FONSECA, 2013)

Uma dessas oportunidades é o paradigma educacional chamado aprendizagem móvel. Ele permite que qualquer pessoa possa acessar informações e materiais de aprendizagem de qualquer lugar e a qualquer hora. Por isso é muito mais flexível do que o aprendizado por computadores desktop tradicionais. (DEMIRBILEK, 2010)

Esse paradigma tem se tornado cada vez mais atraente porque a tecnologia da computação tem sofrido um rápido processo de evolução, sem precedentes na história. Dispositivos eletrônicos estão cada vez mais econômicos e acessíveis, o que por sua vez, facilita a disseminação de tecnologias da informação como dispositivos de massa. (COSTA et al., 2013)

Um dos principais motivos responsáveis pelo aumento exponencial dos dispositivos móveis é a multifuncionalidade, gerada pela quantidade de sensores que carregam, desde câmeras fotográficas até navegadores web e GPS. Além de ser multifuncional, o fato da tecnologia em si ter evoluído rapidamente agiu como catalisador para esse aumento de adoção. Os processadores e a capacidade de memória dos smartphones vêm aumentando e com isso eles ganham sistemas operacionais e aplicativos mais poderosos (KASINATHAN; MUSTAPHA; SAMSUDIN, 2016). A Figura 1 mostra o aumento da adoção através de um gráfico de total de usuários (em milhões) de smartphones nos Estados Unidos.

Figura 1 Total de *smartphones* por ano (EUA)

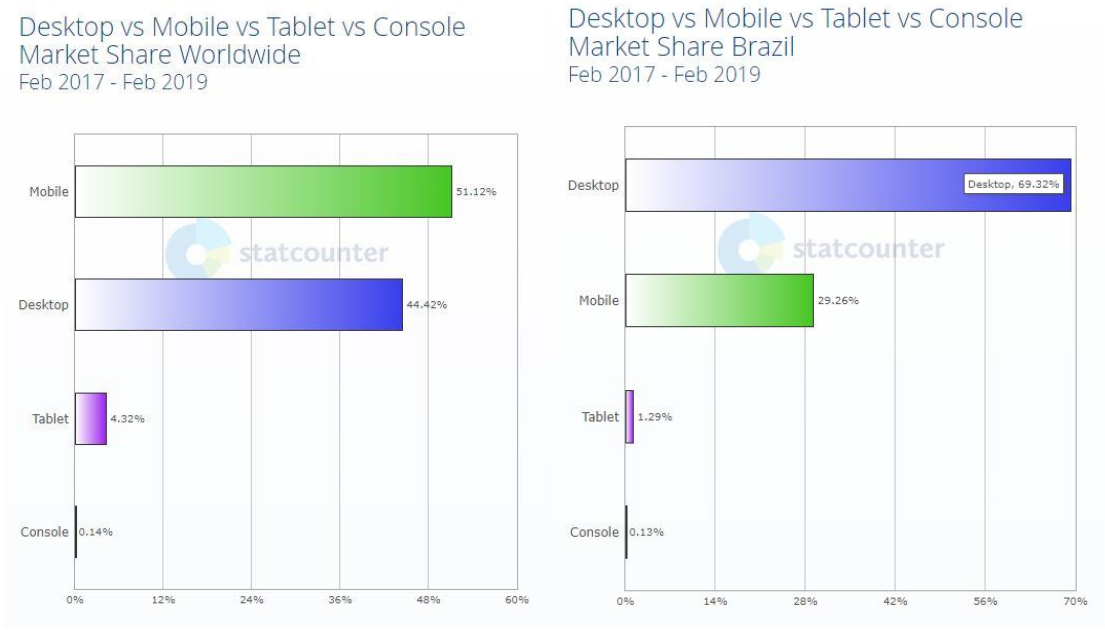


Fonte: (©STATISTA, 2016 Apud KASINATHAN; MUSTAPHA; SAMSUDIN, 2016)

O serviço StatCounter conta com um código de rastreamento instalado em mais de 2 milhões de sites globalmente, os quais gravam mensalmente aproximadamente 10 bilhões de visualizações de páginas dos usuários desses sites. Para cada acesso são analisados o tipo de dispositivo, o sistema operacional, localização, resolução, entre outras informações. Em setembro de 2015, a taxa de amostra do serviço consistia de 16,3 bilhões de visualizações de páginas, o Brasil estava entre as dez maiores amostras individuais, com 583.688.596 registros. (©STATCOUNTER, 2015)

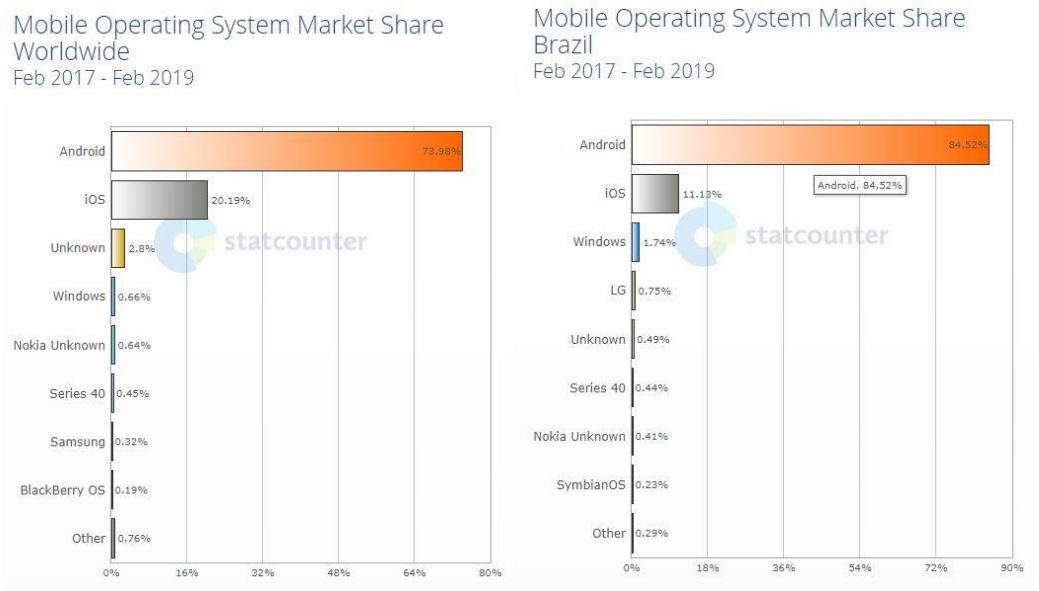
A Figura 1 mostra que o total de acessos por dispositivos móveis (*tablets* e *smartphones*) supera o de computadores (*desktops* e *notebooks*) no mundo, mas ainda representa aproximadamente 30% das visualizações de páginas no Brasil, nos últimos três anos. No mesmo período, a Figura 2 exibe um gráfico com os sistemas operacionais utilizados nos smartphones. O sistema operacional *Android*® (da *Google*®) representa a maior parte, tanto mundialmente (73,98%), quanto no Brasil (84,52%).

Figura 2 Total de Visualizações de Páginas por Tipo de Dispositivo



Fonte: (©STATCOUNTER, 2019b, a)

Figura 3 Porcentagem de Visualizações de Acordo com Sistema Operacional Móvel



Fonte: (©STATCOUNTER, 2019d, c)

Diante desse cenário, aplicativos que auxiliem no ensino-aprendizado de alguns conceitos abstratos de química podem se mostrar muito úteis na educação ao ser capazes de alcançarem uma gama tão grande de usuários.

1.1 – Objetivos

O objetivo principal desse trabalho é o desenvolvimento de software em forma de aplicativos para dispositivos móveis (celulares e tablets) que abordem os seguintes conceitos de química: distribuição eletrônica, ligações covalentes e hibridização.

São objetivos específicos do projeto:

- Desenvolver softwares intuitivos e simples;
- Colocar à disposição de alunos e professores uma ferramenta dinâmica e gráfica que pode auxiliar os processos de apresentação e compreensão de como se faz a distribuição eletrônica com o diagrama de Pauling.
- Implementar recursos que permitam a visualização de geometrias moleculares pelos usuários em todos os ângulos;
- Substituir o processo de desenho das geometrias moleculares no quadro negro bidimensional e páginas de livros pela visualização de objetos tridimensional no software;
- Criar modelos 3D que permitam os usuários visualizarem orbitais híbridos;
- Permitir ao usuário construir moléculas a partir de um átomo central e rotacionar os modelos criados;
- Utilizar a teoria da repulsão dos pares de elétrons da camada de valência (VSEPR) no aplicativo para demonstrar seu efeito sobre a geometria das moléculas;
- Publicar os aplicativos nas lojas de aplicativos para facilitar e simplificar sua distribuição;
- Avaliar os softwares através da aplicação de questionários.

1.2 – Justificativa

Na sociedade hoje, todos estão aprendendo a se comunicar e integrar o humano e tecnológico. Torna-se fundamental conectar o ensino com a vida do aluno, chegando a ele de todas formas possíveis. (STEIN et al., 2019)

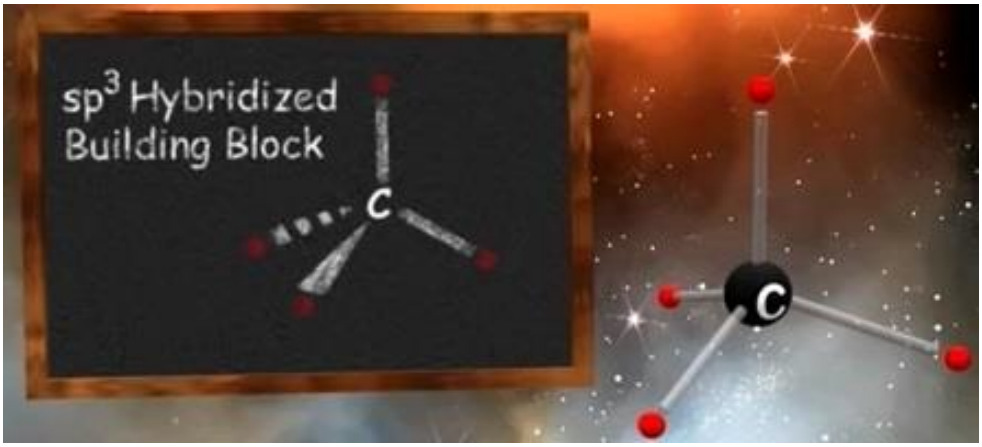
O uso de softwares, como ferramentas de auxílio e aprimoramento do processo de ensino-aprendizagem, tem sido muito bem recebido nos últimos anos. Mais do que isso, eles têm sido vistos como instrumento de motivação e estímulo na construção do conhecimento. Os aplicativos podem ajudar a prender a atenção do aluno, estimular o trabalho em grupo e a interação com o professor. (NOGUEIRA et al., 2008)

Hoje, educar os jovens com eletrônicos se torna mais simples, porque os eletrônicos são o primeiro contato das crianças com o mundo digital. Esse contato proporciona a construção de alguns aspectos cognitivos como memória, atenção, criatividade, entre outros. (GROSS, 2008 *apud* PORTUGAL, 2009)

Muitos professores atualmente buscam recursos para deixar suas aulas de química mais dinâmicas, pois vêem necessidade de encontrar meios que facilitem o processo de ensino-aprendizagem. Um exemplo são os jogos didáticos, usados por professores como atividade lúdica, para tornar suas aulas mais agradáveis. (COSTA et al., 2013)

A Figura 4 permite visualizarmos a diferença entre representar uma estrutura molecular usando um desenho bidimensional e um tridimensional. Observe como pode ser mais difícil para o aluno abstrair a geometria da molécula apenas a partir da imagem no quadro negro (a esquerda na figura).

Figura 4 Comparação entre Geometria Molecular em 2D e 3D



Os professores desenham linhas tracejadas para representar algo que está atrás do quadro. Fonte: Adaptado de (SPONHOLTZ PRODUCTIONS, 2017)

DEMIRBILEK, (2010) conduziu uma pesquisa com 113 professores em oito paises, onde 76% dos participantes responderam estarem interessados no uso de dispositivos móveis e jogos educacionais em suas atividades de ensino.

1.3 – Ineditismo

A pesquisa realizada identificou oportunidade de desenvolvimento de aplicativos com funcionalidades inexistentes e de melhorias nas funções encontradas nos aplicativos disponíveis.

A pesquisa foi conduzida nas lojas de aplicativos online da Google, Microsoft e Apple. Entende-se que não há necessidade de pesquisar todos os softwares existentes, limitando-se aos aplicativos para dispositivos móveis, os quais são o foco e objetivo principal desse trabalho. Softwares de química famosos como, por exemplo, o ACD/ChemSketch[1], que permite a construção de moléculas complexas, não estão diretamente relacionados com o escopo desse projeto porque estão disponíveis apenas para computadores *deskop*.

Os termos de busca utilizados nas lojas de aplicativos foram:

- “Configuração Eletrônica” e “Distribuição Eletrônica”;
- “Electron Configuration”;
- Pauling e “Diagrama de Pauling”;
- “Geometria Molecular”;
- “Molecular Geometry” e “Molecular Structure”;
- VSEPR e “Valence Shell Electron Pair Repulsion”.

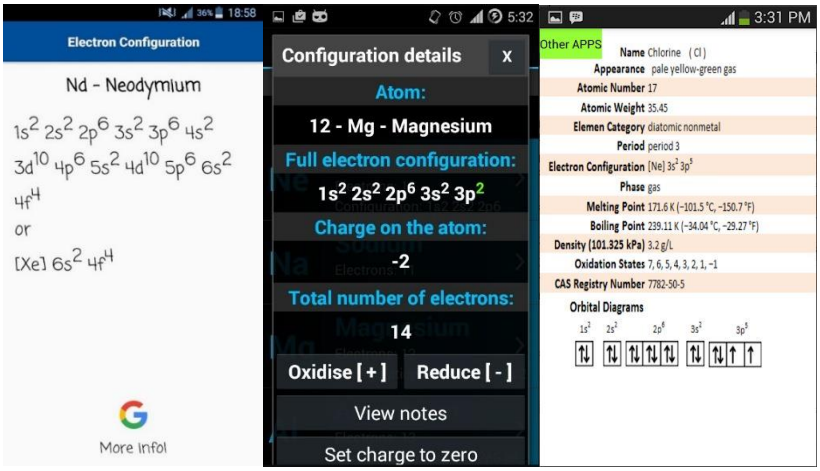
Os resultados retornados sobre os termos relacionados à Configuração Eletrônica, são em sua maioria aplicativos que se limitam a exibição, em formato de texto, da configuração eletrônica do elemento (átomo) pesquisado. Os aplicativos disponíveis, que possuem essa funcionalidade são: Electron Configuration (Figura 5), Electron Config Lite (Figura 5), Periodic Table (Figura 5), Electron (NOVELLINO, 2018),

[1] https://www.acdlabs.com/resources/freeware/chemsketch/

InterÁtomo (SOUSA, 2018), o não lançado Construa Seu Átomo! (LABORATÓRIO DIDÁTICO DE QUÍMICA - LADQUIM, 2018), Tabela Periódica dos Elementos - Modern PTE (DALUZ, 2018), Naked Chem (Figura 6) e Building Atoms, Ions and Isotopes (Figura 7).

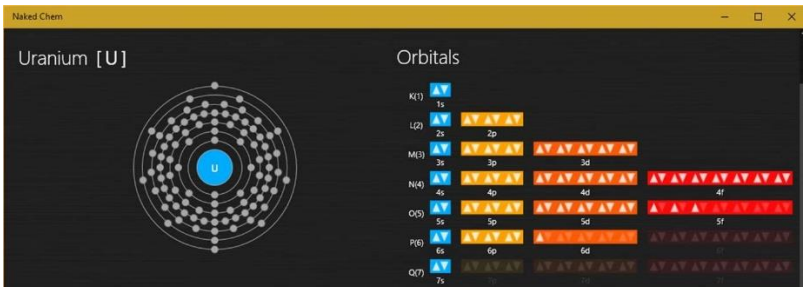
Alguns aplicativos, como o Electronic Structure 4.0 (SCIENCEHIGH, 2016) em inglês, usam elementos de jogos (gamificação) relacionados à configuração eletrônica, mas com uma abordagem diferente da utilizada em sala de aula no brasil. No aplicativo, o usuário clica nas camadas para distribuir elétrons, mas o aplicativo não usa ou menciona Diagrama de Pauling, por exemplo, nem introduz o conceito de *spin* (regra de *Hund*), como abordado no aplicativo desenvolvido nesse projeto.

Figura 5 Electron Configuration, Electron Config Lite e Periodic Table (Google Store)



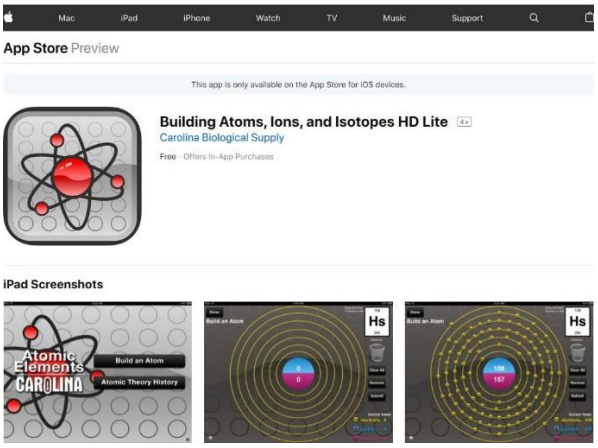
Fonte: (GUY, 2018), (YUKOD SOFTWARE, 2015) e (57DERAJAT, 2019)

Figura 6 Naked Chem (Microsoft Store)



Fonte: (CUZIAC, 2016)

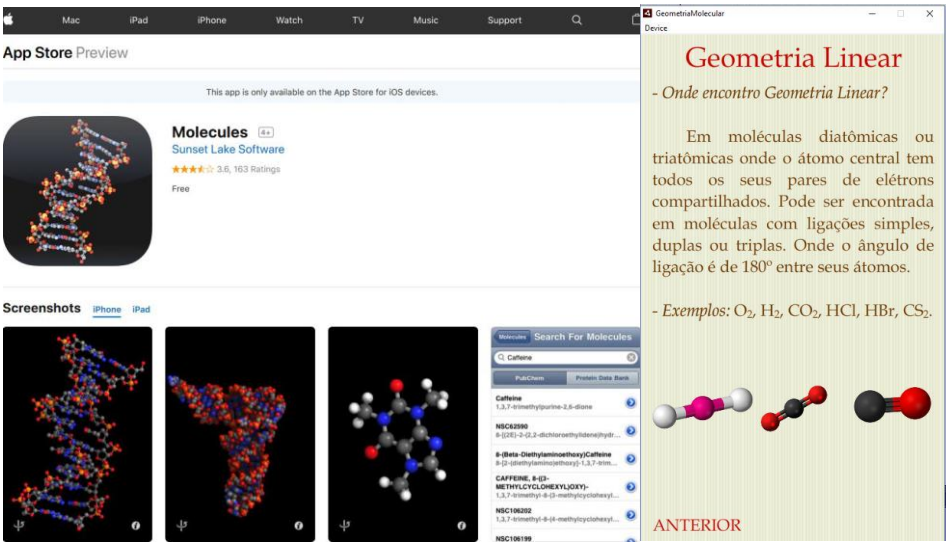
Figura 7 Building Atoms, Ions and Isotopes (Apple Store)



Fonte: (SUPPLY, 2011)

Os resultados encontrados sobre os termos relacionados à Geometria Molecular, são em sua maioria aplicativos que se limitam a exibir moléculas em três dimensões. O usuário pesquisa por compostos químicos e os visualiza em 3D. Outros servem apenas como material de estudo, apresentando a teoria e algumas imagens. Finalmente, existem alguns aplicativos que têm o objetivo apenas de facilitar o desenho de moléculas em duas ou três dimensões, não fazendo nenhum tipo de validação da estrutura desenhada. Os aplicativos disponíveis que possuem essas funcionalidades são: Molecules (Figura 8), Ball & Stick (Figura 9), Chemistry: Geometries (Figura 10), Molecular 3D (Figura 10), Shapes of Molecules (Figura 10), LDMD Química: Aprendendo Geometria Molecular (Figura 8), Chem 101 (Figura 9), Electron Orbitals (JOHNSON, 2018) e Moléculas (GALEMBECK, 2013).

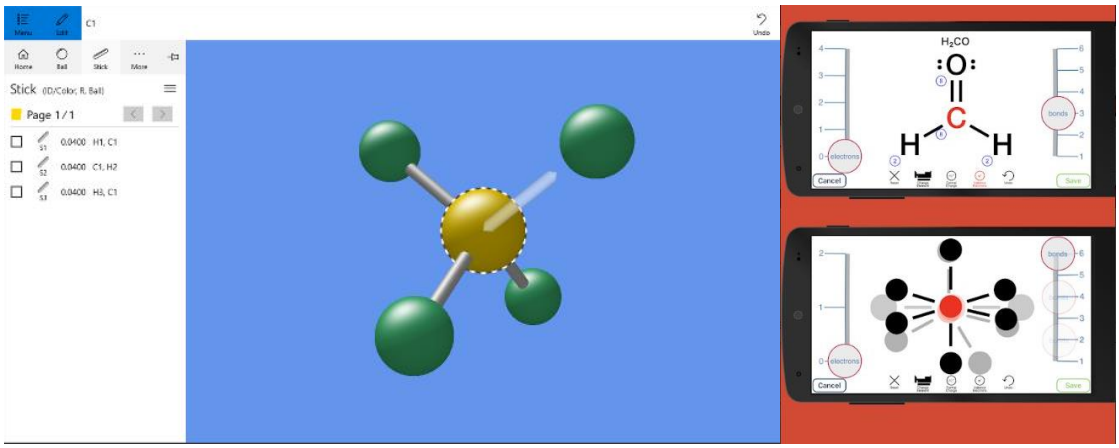
Figura 8 Molecules (Apple Store) e Aprendendo Geometria Molecular (Google Store)



Fonte: (LAKE, 2011) e (LDMD - UFSJ, 2018)

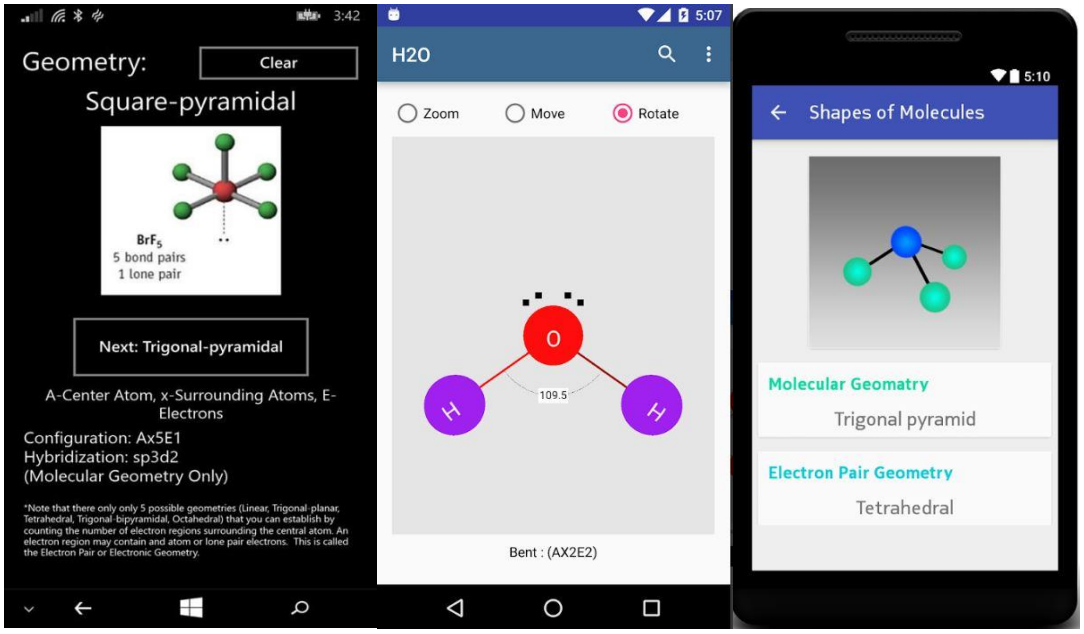
Um aplicativo que é uma exceção é o Molecular Constructor (TEPLUKHIN, 2018). Ele permite que o usuário construa molécula de forma interativa, como um dos objetivos desse trabalho, mas o aplicativo usa uma teoria diferente, da utilizada nesse trabalho, para calcular a forma geometria das moléculas. Além disso, a língua é uma possível barreira na utilização desse software por estudantes nacionais.

Figura 9 Ball & Stick (Microsoft Store) e Chem 101 (Google Store)



Fonte: (FOURELEM, 2016) e (101 EDU, 2019)

Figura 10 Chemistry: Geometries (Microsoft Store), Molecular 3D e Shapes of Molecules (Google Store)



Fonte: (HEBERT, 2014), (D SOLUTIONS, 2016) e (MOBI, 2015)

1.4 – Organização

O presente capítulo descreve a introdução do trabalho desenvolvido, detalhando sua justificativa, seus objetivos gerais e específicos, ineditismo e organização dos capítulos.

O capítulo 2 descreve os fundamentos teóricos das componentes principais deste trabalho, explicando os conceitos necessários de química como: distribuição eletrônica, geometria molecular, ligações covalentes e hibridização. Também explica conceitos utilizados no desenvolvimento do software como: fundamentos do paradigma da programação orientada a objetos, alguns processos de engenharia de software que foram adotados, diagramas da notação gráfica UML e uma introdução aos motores de jogos (*game engines*).

No capítulo 3, as ferramentas e métodos que foram utilizadas no desenvolvimento dos aplicativos são explicados em detalhes. É explicado como as ferramentas de modelagem *Paint 3D, Blender e* o motor de jogo *Unity* foram utilizados. Para isso é importante conhecer a arquitetura de um projeto no *Unity* e seus principais

componentes. Esse capítulo também descreve em detalhes o os métodos e processo de implementação de cada etapa do sistema, começando pelo processo de prototipação, seguido por aspectos no desenvolvimento comuns à ambos os aplicativos e depois aspectos específicos de cada um. O *design* de cada projeto é abordado individualmente e discutido com uso de diagramas de classe e diagramas de sequência da linguagem UML. Ao final, desse capítulo, é abordado também o processo de elaboração e divulgação dos questionários de avaliação dos softwares desenvolvidos.

Os resultados e conclusões são expostos nos capítulos 4 e 5, respectivamente, em que são exibidas algumas imagens dos aplicativos em suas versões finais. São apresentados e discutidos os resultados obtidos dos questionários de avaliação e relatórios das lojas virtuais onde os aplicativos foram publicados. Algumas limitações são descritas e ilustradas, assim como algumas sugestões para melhorias dos sistemas e trabalhos futuros.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Esse capítulo tem como objetivo explicar os principais conceitos teóricos necessários para compreensão desse trabalho.

Os conceitos serão apresentados separadamente entre Computação e Química, através de uma revisão bibliográfica.

2.1 – Ciência da Computação

Nesse subcapítulo serão abordados os principais componentes teóricos relacionados à ciência da computação, requeridos para compreensão da parte técnica relacionada ao desenvolvimento de software desse projeto.

É esperado que o leitor já possua um conhecimento prévio dos blocos básicos de condicionais e laços de repetição em qualquer linguagem de programação baseada no paradigma procedural. Matéria que é parte do conteúdo programático de qualquer curso de engenharia, segundo a portaria INEP. (SHITSUKA et al., 2019)

2.1.1 – Programação Orientada a Objetos (POO)

Todo mundo que começa a programar conhece primeiro os fundamentos básicos e a sintaxe inicial de alguma linguagem de programação. Aprende a transformar um algoritmo em código de uma linguagem e logo descobre que existe uma grande diferença entre escrever algumas linhas de código e projetar, criar e manter uma aplicação mais complexa. Nesse momento, a orientação a objetos pode ajudar, mas antes é preciso conhecer o vocabulário, a terminologia e as palavras que usamos quando falamos sobre uma aplicação orientada a objetos. (ALLARDICE, 2012a)

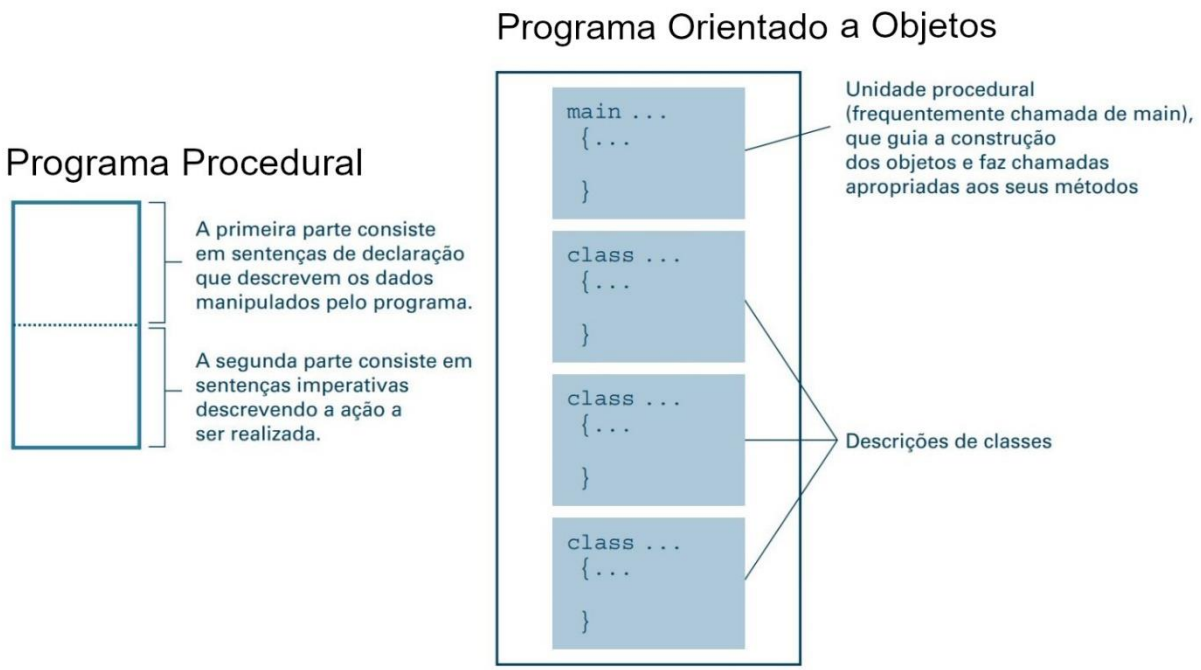
Programação orientação a objetos (POO) é um paradigma, um conjunto de ideias, uma maneira de programar que resolve muitos problemas encontrados no

paradigma da programação procedural, também chamado de paradigma da programação imperativa. (BROOKSHEAR; SMITH; DENNIS BRYLOW, 2013)

Programação procedural comumente se refere a código escrito como um grande procedimento. O código é escrito direto, passo a passo, todo em um mesmo local. O código pode conter algumas funções ou sub-rotinas, para tentar tornar a manutenção mais simples, mas o programa não passa de um grande pedaço de código, frequentemente com toda as variáveis definidas em uma parte e toda a lógica em outra. (ALLARDICE, 2012a)

Conforme os programas foram se tornando cada vez maiores e mais complexos, a abordagem procedural (imperativa) se tornou cada vez mais difícil de se manter. O principal problema enfrentado é que não existe uma forma simples de ligar dados e funcionalidades. No paradigma orientado a objetos estamos o tempo todo agrupando em classes o comportamento e as informações, deixando os atributos perto dos métodos que as utilizam (METZ, 2013). A Figura 11 mostra, lado a lado, como seria dividido um programa procedural e um orientado a objeto.

Figura 11 Diferença entre um Programa Procedural e Orientado a Objetos



Fonte: Adaptado de (BROOKSHEAR; SMITH; DENNIS BRYLOW, 2013)

Existem outros paradigmas além do procedural e orientado a objetos, como os paradigmas da programação funcional e programação lógica. Esses paradigmas costumam ser mais populares no mundo acadêmico ou na resolução de problemas bem específicos de computação. Na prática, o mercado de desenvolvimento de software atualmente usa o paradigma da orientação a objetos e linguagens orientadas a objetos como: C++, C#, Java, PHP, Python, Ruby e muitas outras. (BROOKSHEAR; SMITH; DENNIS BRYLOW, 2013)

Os aplicativos desse projeto foram desenvolvidos com a linguagem de programação orientada a objetos C#.

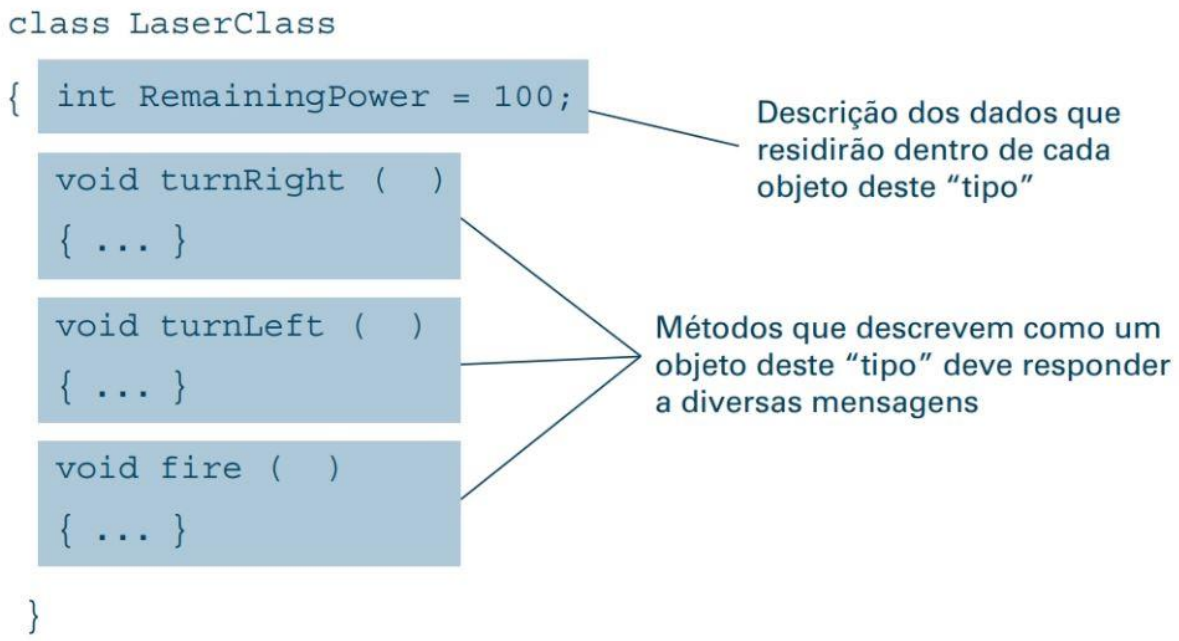
2.1.1.1 – Objetos e Classes

Em uma linguagem orientada a objetos, um grande programa procedural seria dividido em objetos independentes. Quase como se tivéssemos vários miniprogramas, cada um contendo seus próprios dados e lógica, representando diferentes partes da aplicação. Esses objetos se comunicam entre si e eles representam a maneira como você conversaria ou pensaria sobre o problema que você está tentando resolver. Aproximando o raciocínio que temos sobre o mundo real e o nosso programa (ALLARDICE, 2012h).

Objetos e classes estão diretamente ligados uns aos outros. Não podemos falar de um sem mencionar o outro. Em POO usamos classes para criar objetos. Objetos de uma determinada classe, são instâncias dessa classe. (SOUZA, 2013)

Uma linguagem orientada a objeto permite que você crie classes que fornecem um modelo para construção de uma coleção de objetos semelhantes. Uma classe define os métodos (comportamento) e atributos (variáveis). Esses métodos são chamados em resposta a mensagens que o objeto recebe. Um mesmo método pode ser implementado por vários objetos diferentes. É responsabilidade da linguagem encontrar e chamar o método no objeto correto para qualquer mensagem enviada (METZ, 2013). A Figura 12 define uma classe, seu nome é “LaserClass”, objetos dessa classe possuem o atributo “RemainingPower” e respondem as mensagens turnRight, turnLeft e fire.

Figura 12 Exemplo de Definição de uma Classe em POO



Fonte: (BROOKSHEAR; SMITH; BRYLOW, 2013)

Objetos contêm características, propriedades que os descrevem. Um objeto pode conter outros objetos, mas ainda é possível ver a separação entre eles. Objetos podem ter as mesmas características, mas cada objeto é diferente, eles têm sua própria identidade. Os atributos ou estados são independentes entre os objetos, alterar o atributo de um objeto não influência nenhum outro. Objetos também possuem comportamento, geralmente está relacionado com o que o objeto pode fazer e é específico ao tipo do objeto. (ALLARDICE, 2012d)

A palavra classe vem da taxonomia da biologia. Todos os seres vivos de uma mesma classe biológica têm uma série de atributos e comportamentos em comum, mas não são iguais, podem variar nos valores desses atributos e como realizam esses comportamentos. A classe descreve o que o objeto será, mas não é o objeto. Ela pode ser descrita como uma definição ou diagrama do objeto que representa. A maioria das linguagens de programação já possuem algumas classes úteis para datas, vetores e *strings*. de modo que, você não precise começar seus programas escrevendo as mesmas classes básicas todas as vezes. Essas classes costumam ser escritas e agrupadas em bibliotecas. (ALLARDICE, 2012b)

2.1.1.2 – Conceitos Fundamentais em Orientação a Objetos

Existem quatro conceitos fundamentais em linguagens orientadas a objeto: Abstração, Encapsulamento, Herança e Polimorfismo. É importante mantê-los em mente durante o desenvolvimento de classes em POO. Essa terminologia será utilizada nos capítulos seguintes.

Abstração significa que nós focaremos nos aspectos essenciais e gerais de algo, ao invés de em um exemplo específico. Quando abstraímos, automaticamente descartamos o que não é importante ou relevante. Abstrair significa criar um conceito ou ideia que é completamente separado de qualquer instância em específico. Isso é importante em POO, pois é exatamente o que é feito quando se cria uma classe. (ALLARDICE, 2012c)

Em POO agrupamos atributos e comportamento em uma mesma classe, mas também há interesse em se restringir o acesso ao funcionamento interno dessa classe ou qualquer objeto baseado nessa classe. Nos referimos a encapsulamento quando falamos em ocultar informações ou esconder dados. O princípio é que um objeto não deve revelar nada sobre si mesmo, exceto o que for absolutamente necessário para que outras partes da aplicação funcionem. (BROOKSHEAR; SMITH; BRYLOW, 2013)

Por esse motivo nós escondemos um atributo, para controlar o acesso a um dado, de forma que seja acessível apenas dentro do objeto. Isso é importante porque diminui as dependências entre classes no sistema. Se todas as partes da sua aplicação interagem com uma classe apenas através de um método, você só precisa se preocupar com esse método quando estiver alterando essa classe. Sem medo de que uma alteração em outras propriedades dessa classe faça com que outras partes do sistema parem de funcionar. Uma analogia seria com uma “caixa preta”, o funcionamento interno dessa caixa é escondido, podemos interagir com essa caixa apenas pela sua interface, os botões de entradas e saídas. (ALLARDICE, 2012e)

Pense no código de um jogo de corrida, por exemplo, não queremos que outras partes do jogo alterem o atributo velocidade de um veículo diretamente. Queremos que a velocidade seja alterada através de uma interface, o pedal. O pedal ativa o comportamento acelerar, esse por sua vez influencia no atributo velocidade do

veículo. Esse método faria parte desse objeto e apenas ele poderia acessar o atributo velocidade.

Herança é primeiramente uma boa forma de reutilização de código. Com ela podemos criar uma classe, mas ao invés de escrevê-la completamente do início, nós podemos baseá-la em uma classe existente. Quando usamos herança, dizemos que a classe que herda é chamada de subclasse (ou classe filha) e a classe da qual ela herdou chamamos de superclasse (ou classe pai). A classe filha automaticamente tem tudo que a classe pai tem, todos os atributos e métodos, sem que se precise escrever qualquer código. Dessa forma, é necessário escrever apenas os novos atributos e/ou métodos que a classe filha precisa ter.(ALLARDICE, 2012f)

Herança não é importante apenas para reutilização de código, mas também por nos permitir o uso de polimorfismo, o último dos quatro conceitos fundamentais de POO.

O termo polimorfismo é originário do grego e significa "muitas formas". Polimorfismo é o estado de ter muitas formas. Biólogos usam essa palavra, pois uma espécie pode ter várias formas. No contexto de POO, polimorfismo se refere à habilidade de diferentes objetos responderem a uma mesma mensagem. Os objetos que enviam a mensagem não precisam conhecer ou se importar com a classe do receptor, receptores fornecem sua versão específica de comportamento. Uma única mensagem, portanto, tem muitas (*poli*) formas (*morphos*). (METZ, 2013)

Métodos polimórficos são intercambiáveis do ponto de vista do objeto que envia a mensagem. Qualquer objeto que implementa um método polimórfico pode ser substituído por qualquer outro, o objeto que envia a mensagem não precisa saber ou se importar com essa substituição. Esse conceito é o mais complexo entre os quatro conceitos fundamentais, mas é muito poderoso, pois permite que que um algoritmo execute automaticamente o comportamento correto mesmo quando objetos forem de diferentes tipos, ou seja, tenha diferentes “formas”. (BROOKSHEAR; SMITH; BRYLOW, 2013)

Um exemplo prático de polimorfismo é o método de somar. Em várias linguagens, o comportamento dele depende dos tipos de parâmetros que recebe. Se os

parâmetros forem números inteiros, por exemplo, somar adicionaria numericamente esses valores, 2 + 2 retornaria 4. Se os parâmetros forem *strings*, por exemplo, somar iria concatenar esses valores, “2” + “2” retornaria “22”. (ALLARDICE, 2012g)

Quando uma classe filha reimplementa um método da classe pai, diz-se que ela sobrescreve (*override,* em inglês) o método. (METZ, 2013)

2.1.2 – *Unified Modeling Language* (UML)

Unified Modeling Language, linguagem de modelagem unificada, em português, é uma família de notações gráficas, que ajuda na descrição e no projeto de sistemas de software, principalmente os que são construídos usando o paradigma da orientação a objetos. Equipes de desenvolvimento de software fazem uso de linguagens gráficas, como a UML, porque as linguagens de programação não estão em um nível de abstração alto o suficiente para facilitar discussões sobre um projeto. (FOWLER et al., 2005)

Fowler (2005) explica que existem três cenários principais onde UML é usado:

1. Como **esboço**, para transmitir aspectos de um sistema durante seu **desenvolvimento** ou para entender um sistema que já existe (**engenharia reversa**);
2. Como **projeto,** desenvolvido por um projetista em detalhes de modo que uma equipe de programadores seja responsável apenas pela codificação;
3. Como **linguagem de programação**, nesse ambiente os desenvolvedores desenham diagramas UML que são compilados diretamente para código executável.

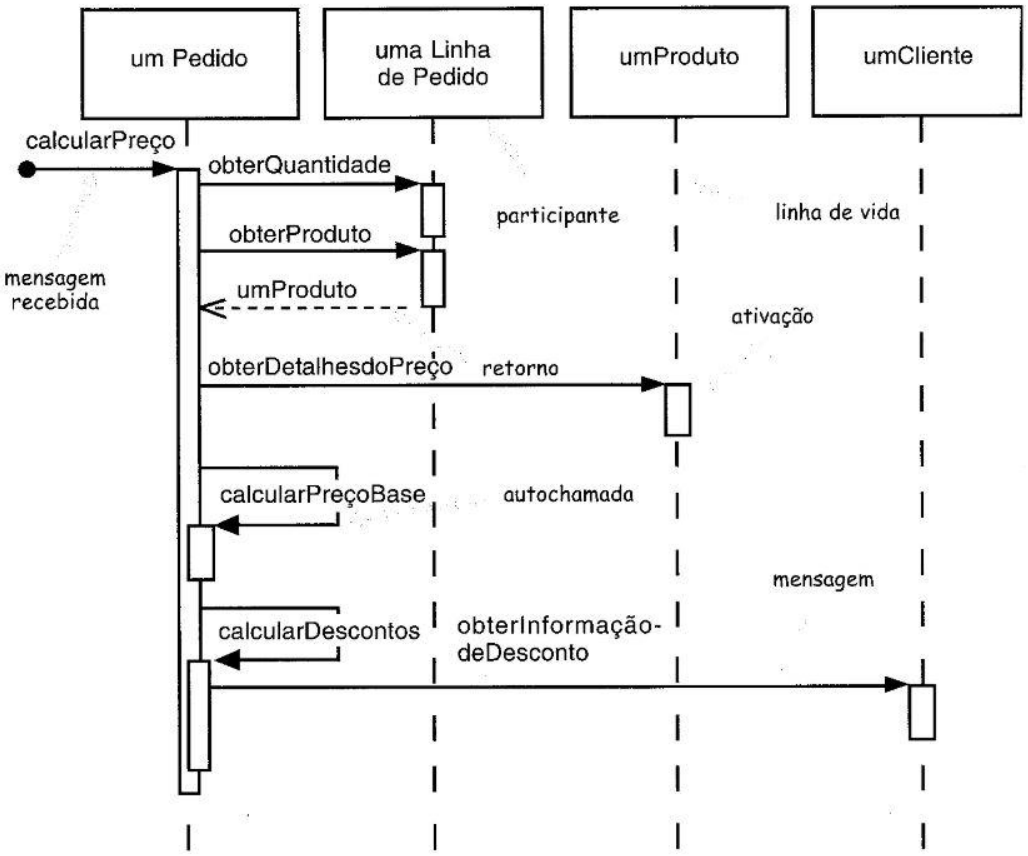
No Capítulo 4, os aspectos principais dos aplicativos desenvolvidos nesse trabalho serão abordados através de dois diagramas UML: o diagrama de sequência e o diagrama de classes. Essa ferramenta facilitará a compreensão de aspectos mais complexos do sistema, sem necessitar do leitor a compreensão de código na linguagem de programação C#, a qual seria a alternativa para transmitir esse conteúdo. Em

outras palavras, a UML será usada nesse trabalho como esboço para ajudar no entendimento de código já existente. Por isso, apenas as classes sobre as quais vale a pena mencionar, terão seu funcionamento explicado.

2.1.2.1 – Diagrama de Sequência

O diagrama de sequência é o diagrama de interação mais comum da UML. Ele descreve como grupos de objetos colaboram em algum comportamento, geralmente de um único cenário. Nele é possível visualizar quais mensagens são trocadas ao longo do tempo. A Figura 13 exibe um exemplo de diagrama de sequência. (METZ, 2013)

Figura 13 Exemplo de Diagrama de Sequência

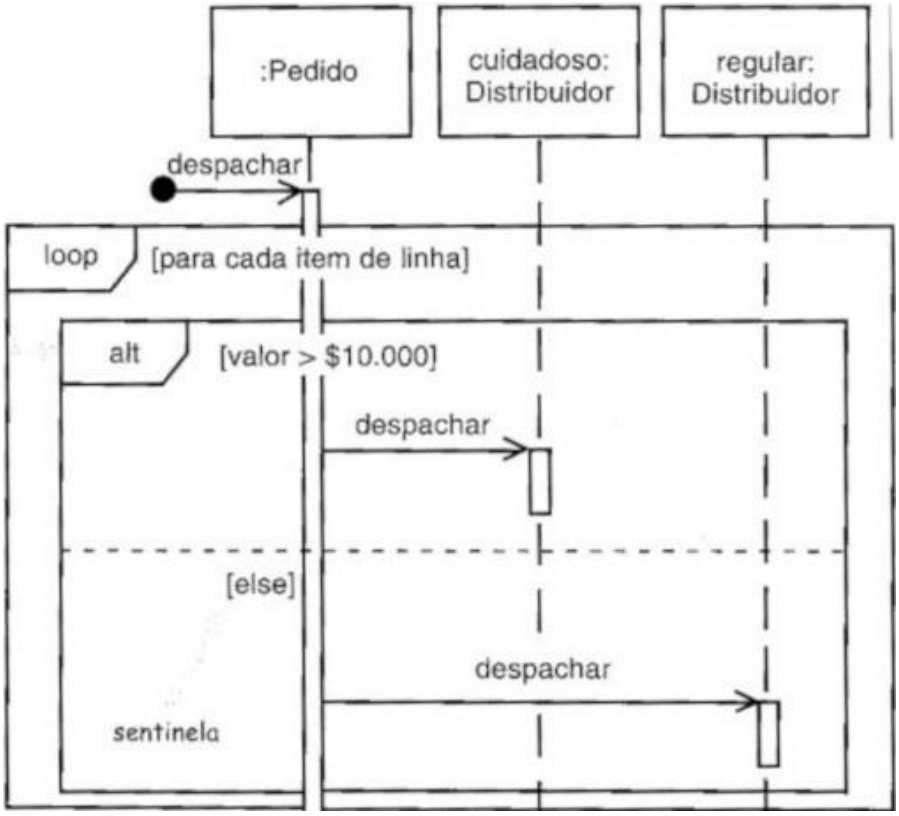


Fonte: (FOWLER et al., 2005)

Cada participante é desenhado com uma linha vertical (que representa o tempo), as setas representam quais mensagens são enviadas, mostrando quem é o remetente e quem é o destinatário. A leitura deve ocorrer de cima para baixo. A barra de ativação mostra quando um participante está ativo na interação e são opcionais na UML. Se houver algum parâmetro, ele pode ser escrito entre parêntesis, logo após a mensagem. (FOWLER et al., 2005)(METZ, 2013)

Também é possível ilustrar estruturas de controle, como laços de repetição e condicionais, através de quadros de interação, conforme ilustrado na Figura 14. O operando *loop* se refere aos laços de repetição como, por exemplo, o *for* e o *while.* O operando alt simboliza a condicional *if* das linguagens de programação. (FOWLER et al., 2005)

Figura 14 Quadros de Interação

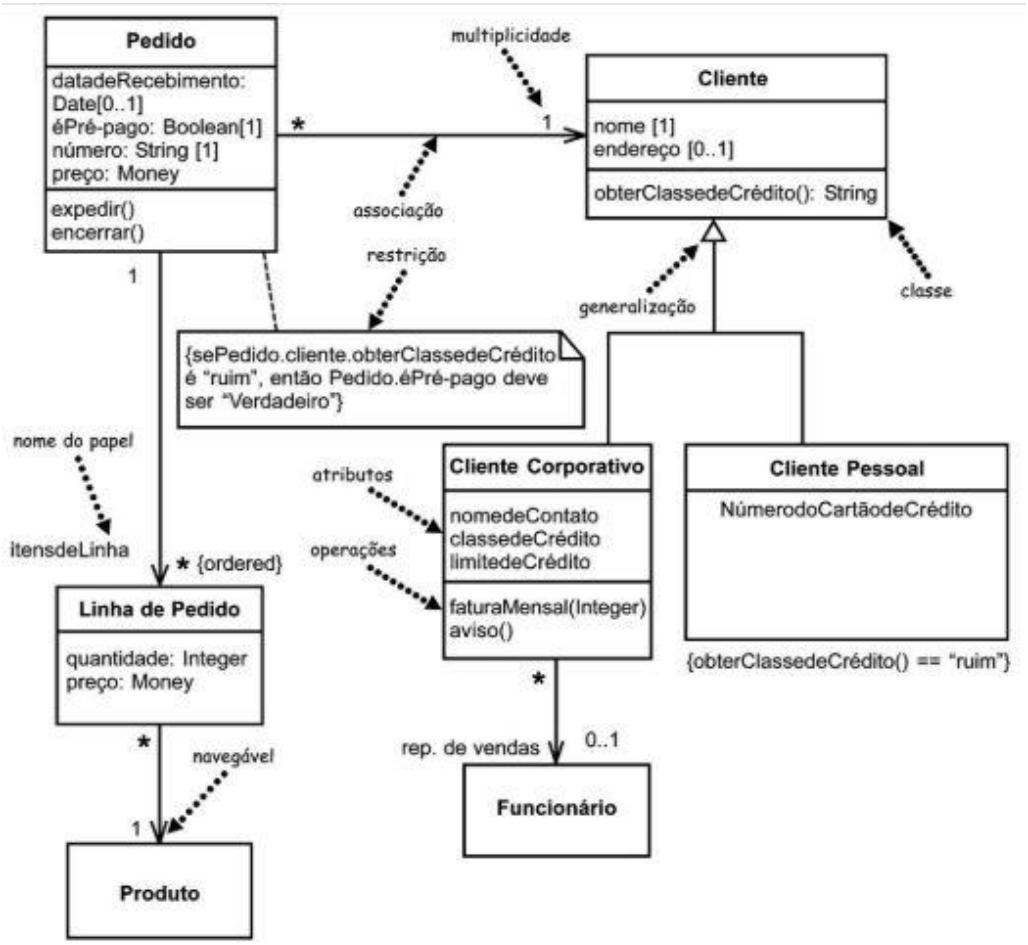


Fonte: (FOWLER et al., 2005)

2.1.2.2 – Diagrama de Classes

O diagrama de classes é amplamente utilizado para descrever os tipos de relações existentes entre os objetos presentes no sistema. Ele também exibe os atributos e métodos de cada classe. A Figura 15 mostra um exemplo de diagrama de classes, considerado simples, pelo autor do livro. As classes são representadas por caixas divididas em três partes: o nome da classe, suas propriedades e operações. Os nomes dos dados e comportamento podem estar acompanhados de um marcador de visibilidade: **+**, **-** ou # que representam público, privado ou protegido, respectivamente. Ao lado das propriedades é descrito o tipo do campo e ao lado dos métodos é descrito o tipo de retorno, quando houver um. O sublinhado é usado para representar operações e atributos de classe. (FOWLER et al., 2005)

Figura 15 Exemplo de Diagrama de Classes



Fonte: (FOWLER et al., 2005)

Entre as classes aparecem linhas, que representam associações e são direcionadas da classe origem para classe de destino. Associações contém multiplicidade, que são indicações de quantos objetos podem preencher a propriedade. As multiplicidades mais comuns são: Um para um, um para muitos e muitos para muitos. Muitos também pode ser representado pela letra **N** ou por um asterisco (*****). (FOWLER et al., 2005)

2.1.3 – Motores de Jogo (*Game Engines*)

O surgimento dos motores de jogo fez com que a tarefa de criação de jogos se tornasse mais fácil. A partir delas os desenvolvedores podem criar, e lançar, seus jogos mais rapidamente para várias plataformas diferentes como, por exemplo, PCs, consoles e *smartphones*.

Esse projeto usou uma *game engine* chamada *Unity* para desenvolver os aplicativos. Antes de conhecer a ferramenta (no capítulo seguinte) é importante compreender o que é um motor de jogo. É importante salientar que o termo jogo é usado constantemente quando o assunto é motor de jogo, mas o leitor deve entender que um jogo é um software e que o termo jogo poderia ser substituído por software em todas as ocorrências a seguir.

Um motor de jogo é uma ferramenta complexa, o que torna difícil sua descrição em poucas palavras. Ele consiste em pacote de funcionalidade que te permite criar jogos multiplataformas 2D e 3D a partir de uma interface unificada. O motor provê uma série de ferramentas comuns, de forma a permitir que seus usuários possam focar na criação de jogos sem ter que “reinventar a roda”. Seus jogos podem ser exportados, com menos esforço, para uma grande quantidade de plataformas, incluindo *desktop* (Linux, macOS, Windows), *mobile* (Android, iOS) e *web* (HTML5). (LINIETSKY; MANZUR, 2014)

Um motor de jogo consiste em um *software* ou conjunto de bibliotecas capazes de reunir e processar todos os aspectos de um jogo em tempo real. Incluindo motor gráfico, motor de física, animações, suporte para sons, inteligência artificial, gerenciamento de arquivos, programação em código, entre outros. São esses motores que

hoje permitem que uma equipe muito menor de pessoas crie um jogo que muitos anos atrás precisaria de muito mais tempo ou muito mais pessoas por trás de seu desenvolvimento. (DIAS, 2016)

O termo *game engine* surgiu na década de 90, em referência aos jogos em primeira pessoa como o Doom, da empresa id Software. Doom foi desenvolvido com uma separação bem definida entre seu núcleo de software principal (o sistema de *rendering,* o sistema detecção de colisões e o sistema de áudio) e os arquivos de arte, mapas e regras de jogo. Os desenvolvedores observaram a importância dessa separação, pois o tempo que levariam para criar jogos seria reduzido se reutilizassem parte do núcleo de software principal que já estava pronto. (GREGORY, 2009)

Antes dessa época, cada jogo era apenas um software feito para ser executado somente em um hardware específico. Conforme a indústria de games cresceu e ficou cada vez mais complexa, foi necessário segmentar cada aspecto do jogo em módulos bem específicos. (GREGORY, 2009)

Motores de Jogo, oferecem componentes reutilizáveis que podem ser manipulados para desenvolvimento de jogos, simuladores ou aplicativos que envolvam cenas com objetos 3D que interagem entre si e sofrem mudanças de acordo com interação do usuário. (WARD, 2008)

Desenvolvedores escolhem utilizar um motor gráfico já existente, devido ao custo-benefício em relação ao tempo e custo que se levaria para desenvolver o próprio motor para sua aplicação. Um ponto negativo é que, quanto mais trabalho o motor faz, menor costuma ser a flexibilidade para os desenvolvedores que o utilizam. (WARD, 2008)

Podemos fazer uma analogia com o desenvolvimento de aplicações web. Atualmente os desenvolvedores escolhem fazer uso de um *framework* desenvolvido na linguagem de programação que pretendem utilizar. Por exemplo, se a linguagem de programação pretendida for *Python,* uma escolha possível seria o *framework* Django, se a linguagem for *Ruby*, escolhas possíveis para framework são *Ruby on Rails* e Sinatra para desenvolvimento da aplicação web. De maneira semelhante, se a decisão for pela linguagem PHP, é improvável que o desenvolvedor escreva a aplicação

do começo absoluto. No caso do PHP, diversos *frameworks* para desenvolvimento de sites estão disponíveis, como *CodeIgniter* ou *Laravel*. (MILETTO; BERTAGNOLLI, 2014)

Frameworks reúnem bibliotecas com funções que resolvem aspectos comuns a todos os sites, como processamento de requisições HTTP, trabalho com o protocolo HTML, envio de formulários, etc. Dessa forma, os desenvolvedores podem focar no que faz a aplicação ser única, ao invés de escreverem código para lidar com algo comum a toda aplicação web. Esse foco acelera o desenvolvimento e permite que mais trabalho possa ser feito em menos tempo. (MOLINARI, 2012)

2.1.4 – Modelos de Processo

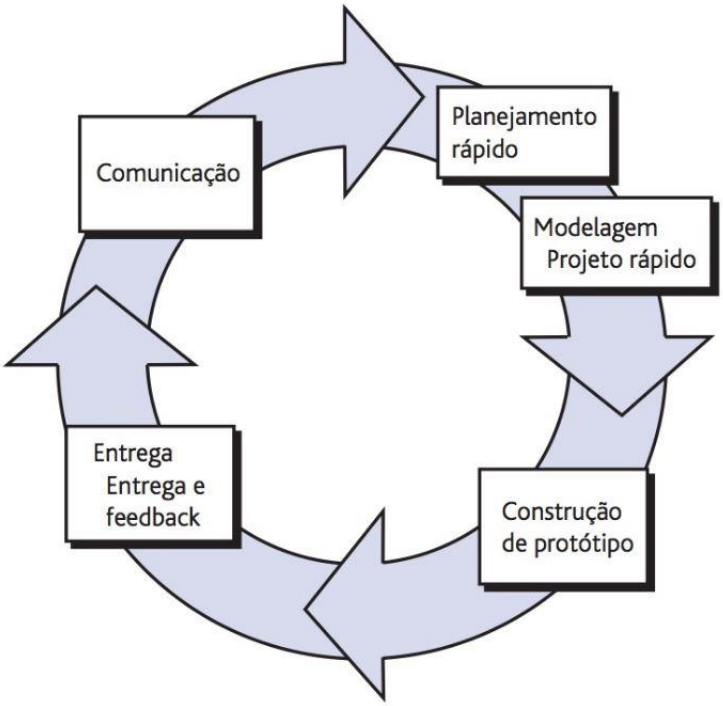
Esse trabalho foi desenvolvido seguindo uma adaptação de roteiros propostos por dois modelos de processo tradicionais de engenharia de software: a Prototipação e o Modelo Incremental.

2.1.4.1 – Prototipação

A prototipação geralmente é utilizada como uma técnica que pode ser implementada no contexto de qualquer modelo de processo de engenharia de software, embora possa ser utilizada como um modelo de processo isolado. O paradigma da prototipação ajuda os envolvidos no projeto a compreender melhor o que está para ser construído ao se concentrar em desenvolver um projeto rápido que possa servir como uma representação dos aspectos do software que serão visíveis para os usuários. (PRESSMAN; MAXIM, 2016)

Brooks (apud PRESSMAN, 2016) diz que a finalidade do protótipo é servir como mecanismo para identificar melhor os requisitos de software quando eles ainda estão obscuros. Portanto, ele sugere que se jogue fora esse “primeiro sistema”, afirmando que quase sempre a qualidade fica comprometida ao se tentar transformar um protótipo grosseiro em um produto final. A Figura 16 mostra o roteiro que foi seguido para desenvolvimento dos primeiros protótipos dos aplicativos.

Figura 16 O paradigma da prototipação



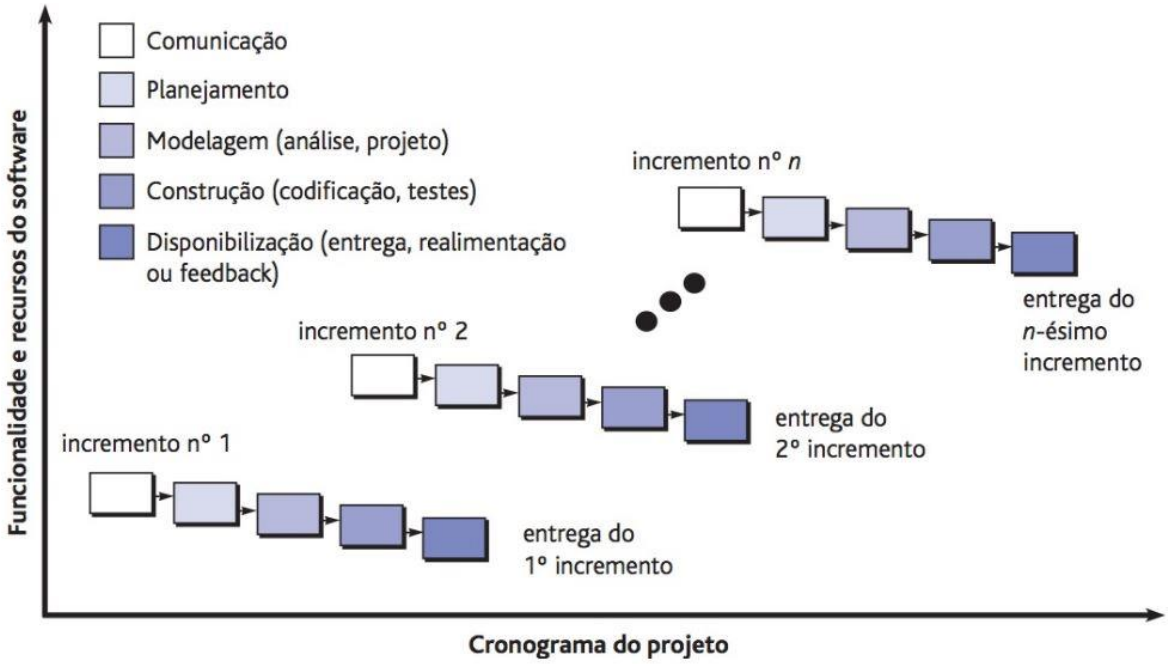
Fonte: (PRESSMAN; MAXIM, 2016)

2.1.4.2 – Modelo de Processo Incremental

Uma vez que os requisitos iniciais do software ficaram razoavelmente bem definidos, foi possível seguir o roteiro proposto pelo modelo de processo incremental. Nesse modelo, o escopo geral do trabalho ainda não está completo, o que impede que um processo puramente linear seja seguido. Por isso, trabalha-se no fornecimento de um conjunto funcional por vez, que irão expandir as funcionalidades do software a cada versão (incremento). (PRESSMAN; MAXIM, 2016)

A Figura 17 ilustra as sequências lineares que produzem incrementos entregáveis de software, no modelo incremental. Os aplicativos desenvolvidos nesse trabalho recebem as novas funcionalidades, propostas através do feedback dos usuários, seguindo esse fluxo.

Figura 17 O modelo incremental



Fonte: (PRESSMAN; MAXIM, 2016)

2.2 – Química

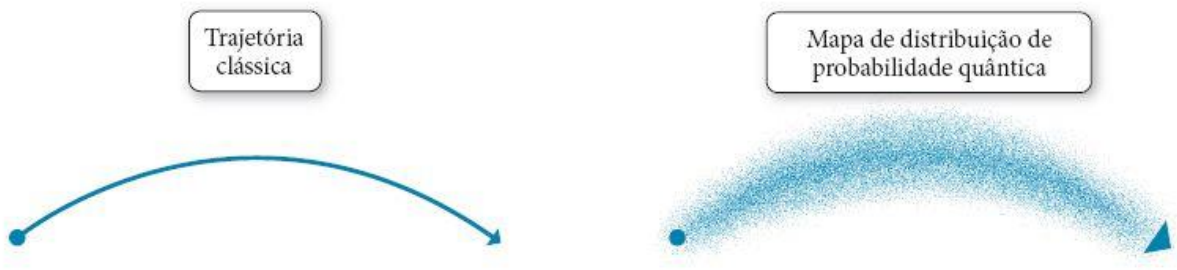
Nesse subcapítulo serão abordados os principais componentes teóricos relacionados à química, necessários para se compreender as partes de configuração eletrônica, geometria molecular, ligações covalentes e hibridização desse projeto.

2.2.1 – Conceitos Fundamentais

Os átomos consistem de elétrons em volta de um núcleo muito pequeno, composto por prótons e nêutrons. Os elétrons possuem carga elétrica com sinal negativo, enquanto os prótons possuem sinal positivo. Os nêutrons são eletricamente neutros. O número de prótons é chamado de número atômico e é usado para caracterizar os elementos químicos. Um átomo eletricamente neutro tem o número de elétrons igual ao seu número atômico. (CALLISTER; RETHWISCH, 2007)

O elétron possuí características tanto de uma onda como de uma partícula, por isso não podemos saber simultaneamente sua posição e sua velocidade, isso faz com que seja impossível traçar sua trajetória, como poderíamos fazer segundo a física clássica. Assim sendo a posição de um elétron é considerada como sendo a probabilidade de o elétron estar em vários locais ao redor do núcleo. Isso significa que a posição dele é descrita por uma distribuição de probabilidades ou nuvem eletrônica. Em outras palavras, na mecânica quântica, as trajetórias são substituídas por mapas de distribuição de probabilidade. O mapa mostra onde um elétron tem probabilidade de ser encontrado sob um dado conjunto de condições. A Figura 18 mostra uma comparação entre uma trajetória segundo a mecânica clássica e a mecânica quântica. (TRO, 2017)

Figura 18 Trajetória Clássica e Distribuição Probabilística



Na mecânica clássica, a posição e velocidade determinam o percurso. Na mecânica quântica não podemos calcular trajetórias determinísticas, é preciso pensar em termos de probabilidades. (TRO, 2017)

A posição e velocidade do elétron são propriedades complementares, se soubermos uma com precisão, a outra se torna indeterminada. A velocidade está diretamente relacionada com a energia e, portanto, a posição e a energia são propriedades complementares, quanto mais soubermos a respeito de uma, menos sabemos a respeito da outra. Os mapas de probabilidade a seguir são descritos para os estados do elétron com energias bem definidas, mas não para suas posições. Em outras palavras, podemos especificar a energia com precisão, mas não sua localização em um determinado instante. Dessa forma, a posição dos elétrons é descrita em termos de um orbital, que representa a distribuição espacial dos elétrons atômicos. Isso é importante para compreensão do compartilhamento de elétrons entre átomos formando ligações covalentes. (TRO, 2017; VOLLHARDT; SCHORE, 2013)

2.2.2 – Distribuição Eletrônica

As representações gráficas dos orbitais correspondem a funções de onda, possíveis soluções da equação de Schroedinger, que estão fora do escopo desse trabalho. Cada orbital em um átomo é caracterizado pelos três primeiros, de quatro parâmetros conhecidos como números quânticos, que estão inter-relacionados: (CALLISTER; RETHWISCH, 2007)

1. *n*, o número quântico principal;
2. *l,* o número quântico de momento angular (azimutal);
3. m_l, o número quântico de magnético;
4. m_s, *o número quântico de spin;*

O número quântico principal determina o tamanho e energia global de um orbital. Está relacionado a distância de um elétron ao núcleo. Ele pode assumir os valores n = 1, 2, 4, ... e assim por diante. Pode ocorrer dessas camadas serem designadas por letras K, L, M, N, entre outras, como pode ser visto na Figura 19. (CALLISTER; RETHWISCH, 2007; TRO, 2017)

O número quântico de momento angular designa uma subcamada, é um número inteiro que determina a forma do orbital. Pode assumir os valores l = 0, 1, 2, 3 e as subcamadas serem designadas pelas letras s, p, d, f, respectivamente. A Figura 19 também mostra esses valores. (TRO, 2017)

Figura 19 Estados Eletrônicos de Algumas Camadas

Principal Quantum Number n	*Shell Designation*	*Subshells*	*Number of States*	*Number of Electrons*	
				Per Subshell	*Per Shell*
1	*K*	*s*	1	2	2
2	*L*	*s*	1	2	8
		p	3	6	
3	*M*	*s*	1	2	18
		p	3	6	
		d	5	10	
4	*N*	*s*	1	2	32
		p	3	6	
		d	5	10	
		f	7	14	

Fonte: (CALLISTER; RETHWISCH, 2007)

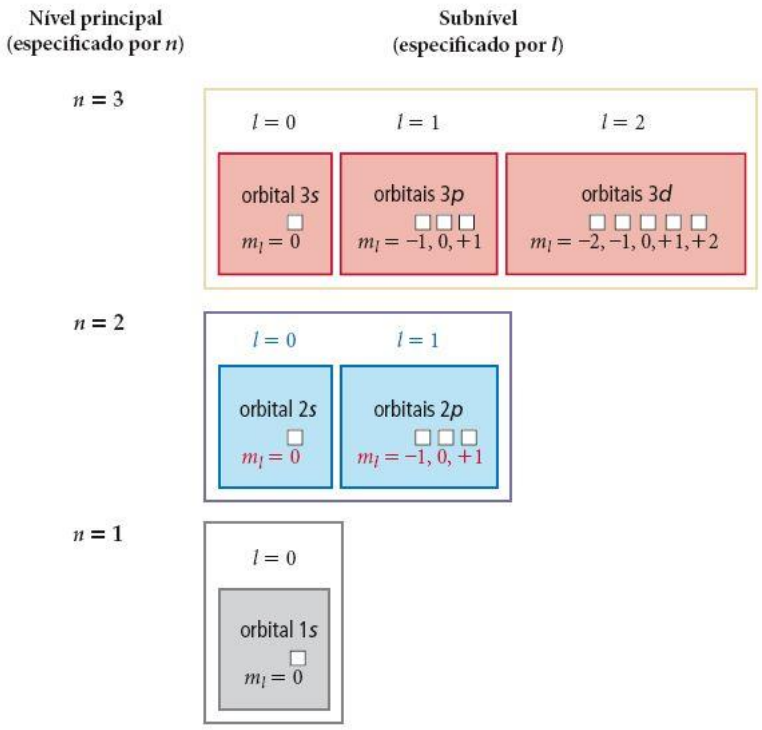
O número quântico magnético é um número inteiro que determina a orientação do orbital. Pode assumir os valores inteiros, inclusive o zero, no intervalo de -l até +l. Ele especifica o número de estados energéticos (Figura 19) para cada subcamada.(CALLISTER; RETHWISCH, 2007; TRO, 2017)

O número quântico de spin pode assumir dois valores ($+\frac{1}{2}$ e $-\frac{1}{2}$). Está associado ao momento de rotação do elétron, que pode estar orientado para cima ou para baixo. Ele é importante quando consideramos como os elétrons ocupam os orbitais. (CALLISTER; RETHWISCH, 2007)

A ideia de um elétron “girando” é metafórica. Uma maneira melhor de expressar a ideia é dizer que o elétron tem momento angular intrínseco. (TRO, 2017)

Um orbital atômico é determinado por uma combinação específica dos três primeiros números quânticos (*n, l* e *ml*). Por exemplo, o orbital n = 1, l = 0 e ml = 0 é conhecido como o orbital 1s. O 1 em 1s é o valor de n e o s especifica que l = 0. Há apenas um orbital 1s em um átomo, seu valor de ml é zero. (TRO, 2017)

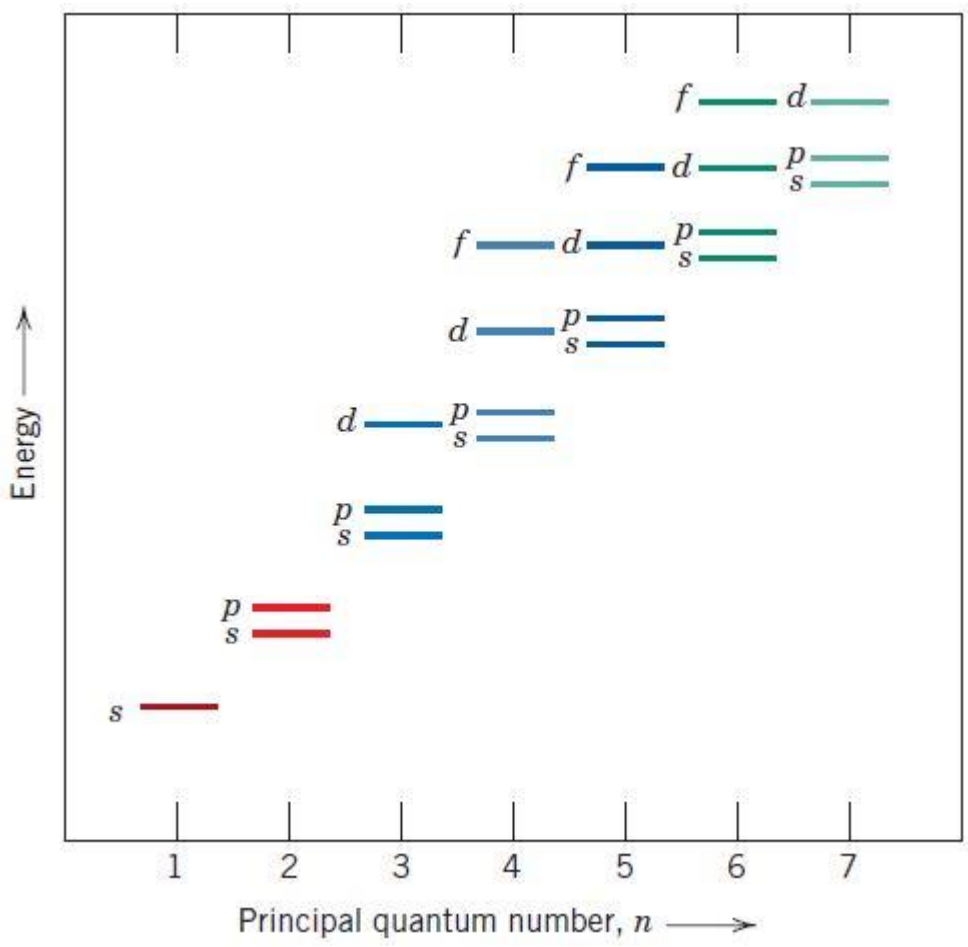
Figura 20 Orbitais nos três primeiros níveis principais



Os pequenos quadrados representam os orbitais. Fonte: (TRO, 2017)

Para uma subcamada s, existe um único estado energético, enquanto existem três, cinco e sete estados de energia para as subcamadas p, d e f, respectivamente. Quanto menor o número quântico principal, menor o nível energético. A energia do estado do estado 1s é menor do que a energia do estado 2s, conforme pode ser visto no diagrama da Figura 21. Observe também que a energia de uma subcamada aumenta com o valor de l, a energia do estado 3d é maior do que a do estado 3p. (CALLISTER; RETHWISCH, 2007; VOLLHARDT; SCHORE, 2013)

Figura 21 Representação Esquemática das Energias para Camadas e Subcamadas

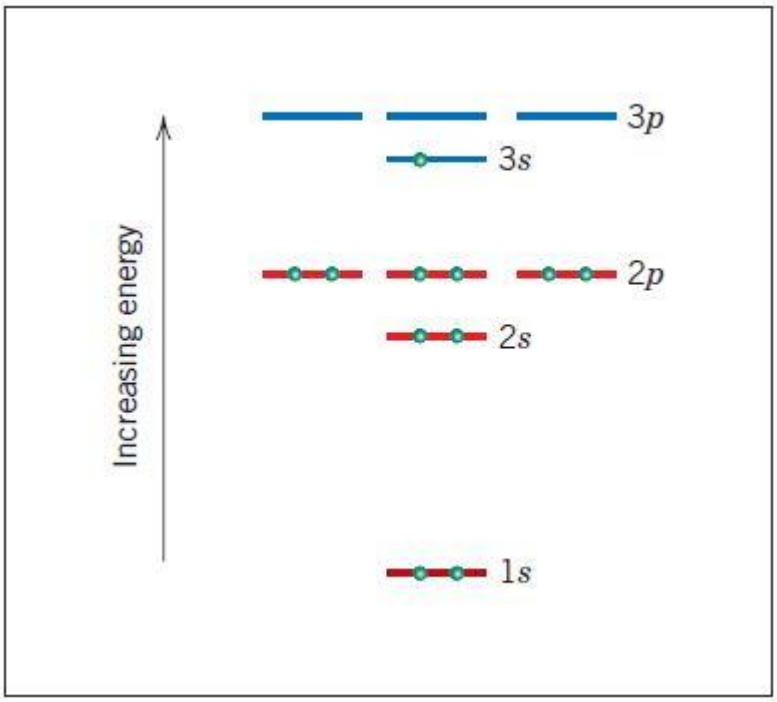


Fonte: (RALLS; COURTNEY; WULFF, 1976 Apud CALLISTER; RETHWISCH, 2007)

A configuração eletrônica representa a maneira como esses estados são ocupados por elétrons. O princípio da exclusão de Pauli estipula que cada estado pode comportar um número máximo de dois elétrons, os quais devem possuir valores opostos de spin. Dessa forma, as subcamadas s, p, d e f podem acomodar um total de 2,

6, 10 e 14 elétrons, cada uma, respectivamente. Para a maioria dos átomos, os elétrons preenchem os estados energéticos mais baixos, dois elétrons (com spins opostos) por estado. Por exemplo, a Figura 22 representa esquematicamente a configuração eletrônica para um átomo de sódio (Na). (CALLISTER; RETHWISCH, 2007)

Figura 22 Configuração Eletrônica para um átomo de sódio (Na)



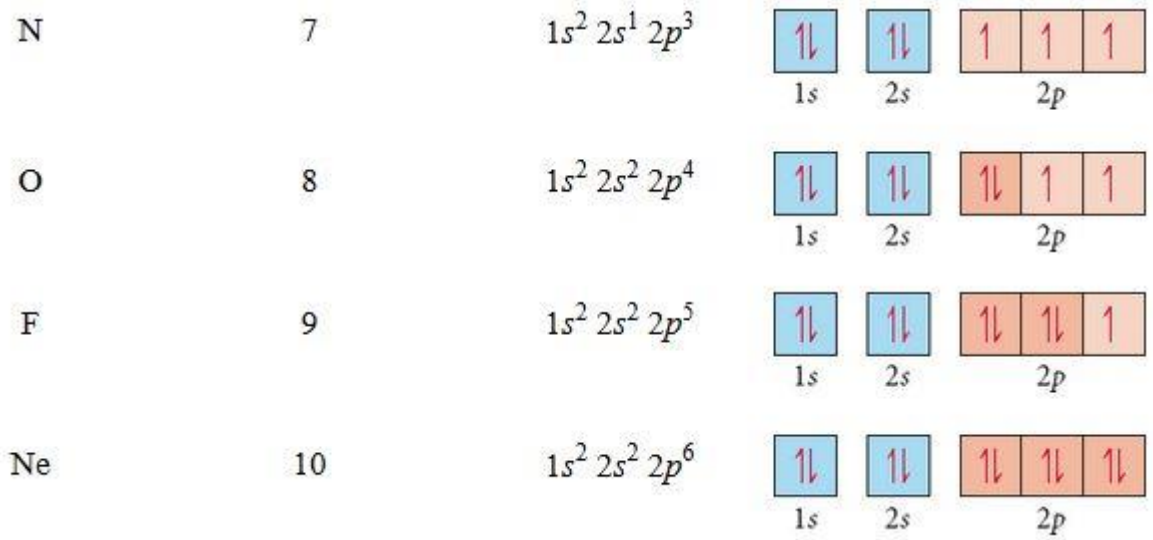
Fonte: (CALLISTER; RETHWISCH, 2007)

A notação convencional indica o número de elétrons em cada subcamada por um índice sobrescrito após a designação da camada e subcamada. Por exemplo, para o átomo de sódio a configuração eletrônica esperada, segundo a notação convencional, é $1_s^2 2_s^2 2_p^6 3_s^1$. Os elétrons que ocupam a camada mais externa são chamados de elétrons de valência, eles são importantes porque participam de ligações que formam moléculas. Alguns elementos, chamados de gases inertes (ou nobres), possuem uma configuração eletrônica estável, isso significa que os estados da camada mais externa estão completamente preenchidos. Uma maneira de outros elementos, que não possuem a camada de valência completamente preenchida, adquirirem estabilidade é através do compartilhamento de elétrons com outros átomos. (CALLISTER; RETHWISCH, 2007)

Resumidamente, em outras palavras, os elétrons ocupam orbitais de modo a minimizar a energia do átomo; por isso, orbitais com menor energia são preenchidos primeiro, na seguinte ordem: 1s 2s 2p 3s 3p 4s 3d 4p 5s 4d 5p 6s. Cada orbital não pode conter mais do que dois elétrons. Os spins são opostos quando dois elétrons ocupam o mesmo orbital. Quando orbitais de energia igual estão disponíveis, os elétrons ocupam primeiramente esses orbitais individualmente e com spins paralelos, ao invés de em pares. Isso é definido pela regra de *Hund,* que afirma que os elétrons só começam a se emparelhar depois que os orbitais de mesma energia estiverem cheios pela metade. (TRO, 2017; VOLLHARDT; SCHORE, 2013)

A Figura 23 mostra a configuração eletrônica e diagrama de orbitais de quatro elementos, com número atômico entre 7 e 10. Observe como os orbitais p são preenchidos com elétrons individuais antes dos elétrons se emparelharem

Figura 23 Exemplos de Configurações Eletrônicas



As setas representam os spins. Fonte: (TRO, 2017)

2.2.3 – Diagrama de Pauling

O padrão de preenchimento de orbitais, descrito anteriormente, é conhecido como o princípio de *aufbau* (construção, em alemão). Com ele, podemos sistematicamente construir as configurações eletrônicas para os elementos. (TRO, 2017)

Linus Pauling incluiu um diagrama de distribuição eletrônica em seu livro *The Nature of the Chemical Bond,* em 1939, como estratégica didática, um recurso para distanciar a interpretação do fenômeno dos formalismos matemáticos. Desde a publicação desse trabalho, houve várias tentativas de apresentação de esquemas que indicassem a configuração eletrônica dos átomos. Muitas modificações foram realizadas no diagrama original como, por exemplo, a inversão da ordem das camadas, mas no Brasil, o modelo ainda é atribuído ao químico Linus Pauling, em muitos livros na área do ensino médio e vestibulares. A Figura 24 contém uma versão do diagrama, que é usado como auxílio mnemônico, para descrição da ordem correta de preenchimento dos subníveis por elétrons em um átomo. (BIANCO; MELONI, 2019)

Figura 24 O princípio de *aufbau*, sendo aplicado no Diagrama de Pauling

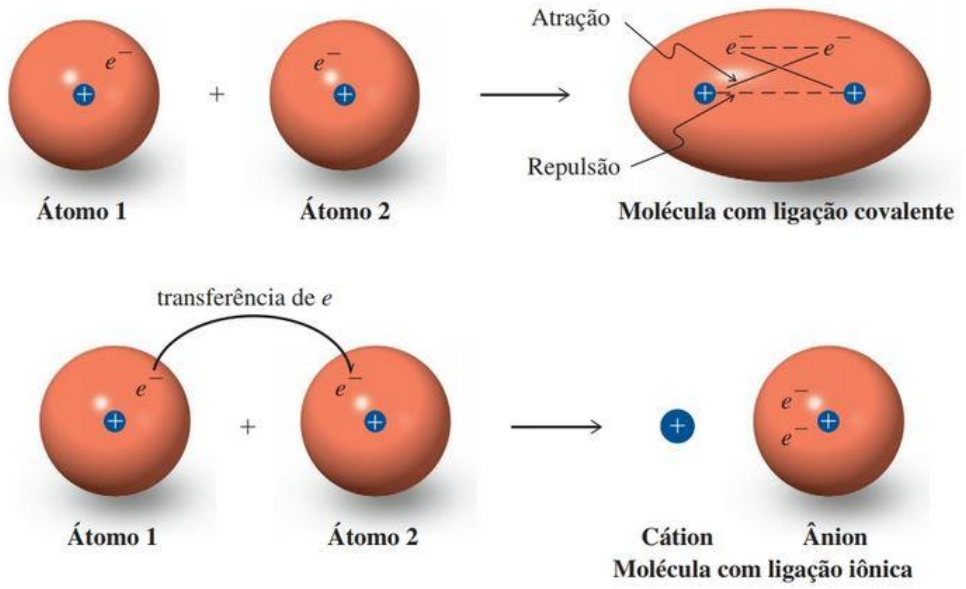
As setas indicam a ordem de preenchimento das subcamadas. Fonte: (BIANCO; MELONI, 2019)

2.2.4 – Ligações Covalentes

Moléculas surgem da ligação entre átomos, que interagem entre si quando a isso for energeticamente favorável, por exemplo, calor pode ser liberado quando a

ligação se forma. Durante a formação, a liberação de energia baseia-se na Lei de Coulomb[2]: Os elétrons e prótons se atraem (cargas opostas) e os elétrons se afastam no espaço (carga iguais se repelem). Quando dois átomos estão próximos, o núcleo positivo de um atraí os elétrons do outro e vice-versa, ou seja, os núcleos são mantidos juntos pelos elétrons localizados entre eles. Existe um modo de ligação alternativo, onde há transferência completa de um elétron de um átomo para outro. O resultado é a formação de dois íons, um ânion (carga negativa) e um cátion (carga positiva). A Figura 25 ilustra, de forma simplificada, a diferença entres os dois tipos de ligação. (VOLLHARDT; SCHORE, 2013)

Figura 25 Diferença entre Ligação Covalente e Iônica



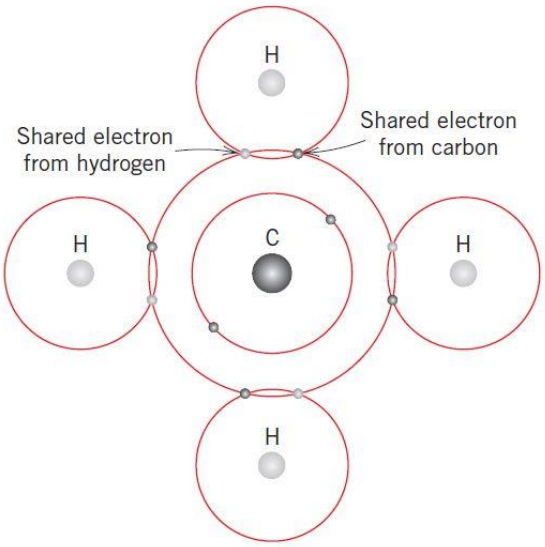
Na ligação covalente há compartilhamento de elétrons, enquanto na ligação iônica ocorre transferência. Fonte: (VOLLHARDT; SCHORE, 2013)

A tabela periódica baseia-se na regra do octeto: Os átomos tendem a formar moléculas de modo a atingir a configuração de um gás nobre. Octeto, nesse contexto, se refere aos oito elétrons, que são encontrados na camada de valência, nos gases

[2] Cargas opostas se atraem com força inversamente proporcional ao quadrado da distância entre os centros das cargas.

nobres, elementos estáveis que, por isso, apresentam reatividade química baixa. O número de ligações covalentes possível para um determinado átomo é obtido pelo número de elétrons de valência. Para N' elétrons de valência, um átomo pode se ligar covalentemente com, no máximo, 8 – N'. Por exemplo, o carbono tem quatro átomos na camada de valência e, portanto, pode compartilhar 8 – 4, ou quatro, elétrons. Figura 26 mostra uma representação esquemática desse exemplo. (CALLISTER; RETHWISCH, 2007; VOLLHARDT; SCHORE, 2013)

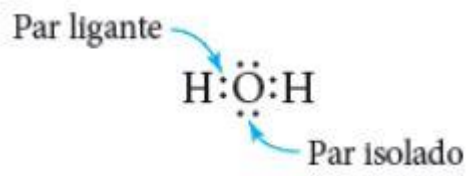
Figura 26 Representação Esquemática da Ligação Covalente de uma Molécula CH_4



O Hidrogênio cede um elétron, assim como o Carbono. Fonte: (CALLISTER; RETHWISCH, 2007)

Um par de elétrons compartilhado é chamado de par ligante, enquanto um par que esteja associado com apenas um átomo é chamado de par isolado, ou elétrons não ligantes, pois não estão envolvidos na ligação. Elétrons compartilhado são contados nos octetos (ou duetos) de ambos os átomos. Outro exemplo é ilustrado na Figura 27, observe que o átomo de oxigênio tem seis elétrons em sua camada de valência (precisa de dois para ficar com a camada completa) e o átomo de hidrogênio tem um elétron na camada de valência (precisa de um para ficar com a camada completa). (TRO, 2017)

Figura 27 Exemplo de Pares de Elétrons não Compartilhados



Oxigênio ficou com dois pares de elétrons isolados (não ligantes). Fonte: (TRO, 2017)

2.2.5 – Geometria Molecular

O modelo utilizado, nesse trabalho, para determinar as formas das moléculas, foi o da teoria da *repulsão dos pares de elétrons da camada de valência* (RPECV). Essa teoria se baseia na ideia de que grupos[3] de elétrons se repelem uns aos outros. Sabemos que os grupos também são atraídos pelo núcleo, mas a teoria RPECV se concentra nas forças de repulsão. De acordo com essa teoria, a geometria preferida de uma molécula é aquela onde os grupos de elétrons tem separação máxima possível e, consequentemente, a energia mínima. Moléculas que tem apenas um átomo interior (central) dependem dos seguintes fatores para determinar sua geometria molecular: (TRO, 2017)

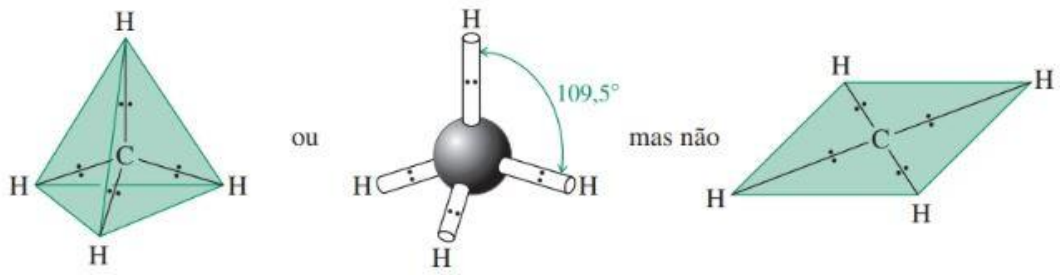
- O número de grupos de elétrons em torno do átomo central;
- Quantos desses grupos são elétrons ligantes (compartilhados);
- Quantos são pares de elétrons isolados (não compartilhados);

Por exemplo, na Figura 26, o desenho da molécula de CH_4 (Metano) poderia levar o leitor a acreditar que os ângulos são de 90º. A representação frequentemente é plana porque é mais fácil de desenhar, não se deve confundir este tipo de desenho com a geometria real da molécula (tetraédrica, no caso do CH_4). A Figura 28 mostra a geometria molecular do CH_4, resultado da aplicação do princípio da RPECV. A

[3] São considerados grupos de elétrons: um par isolado, uma ligação simples, uma ligação dupla, uma ligação tripla ou um elétron único.

orientação das quatro valências para os vértices de um tetraedro regular leva à menor energia de repulsão possível. (VOLLHARDT; SCHORE, 2013)

Figura 28 Geometria Molecular do CH_4 Segundo a Teoria RPECV

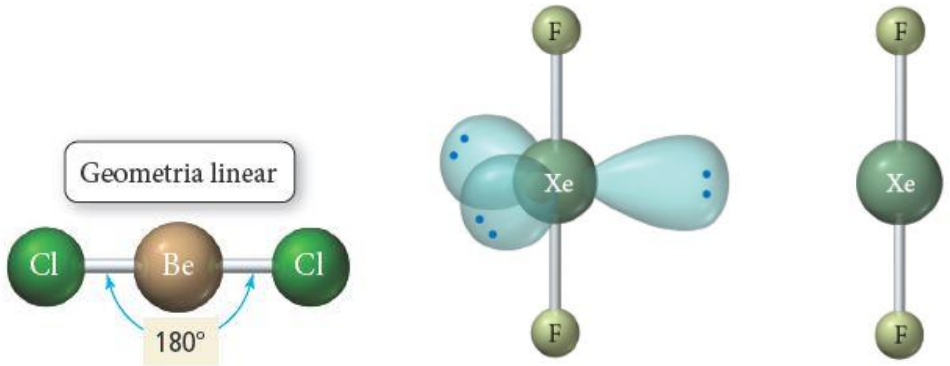


Fonte: (VOLLHARDT; SCHORE, 2013)

2.2.5.1 – Geometria Linear

Estruturas com dois grupos de elétrons (duas ligações) em torno do átomo central produzem uma geometria linear. De acordo com a teoria RPECV, a geometria de moléculas de $BeCl_2$ (Figura 29) ou CO_2, por exemplo, assumem uma separação máxima com um ângulo de 180º. Observações experimentais confirmam que ambas as moléculas de $BeCl_2$ e CO_2 são realmente lineares. Quando existem cinco grupos de elétrons em torno do átomo central e três desses grupos são pares isolados, eles ocupam todas as três posições equatoriais, gerando também uma geometria linear, como ocorre no XeF_2 (Figura 29). (TRO, 2017)

Figura 29 Exemplos de Geometrias Lineares

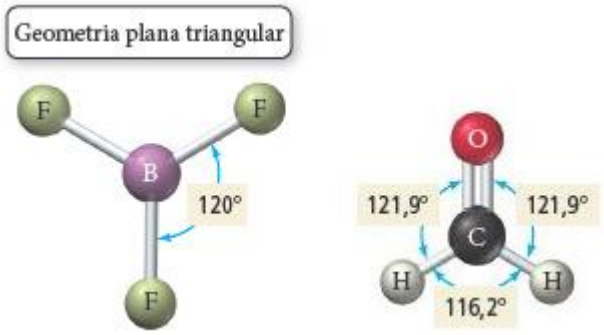


Fonte: Adaptado de (TRO, 2017)

2.2.5.2 – Geometria Plana Triangular

Três grupos de elétrons em torno do átomo central maximizam sua separação assumindo ângulos de 120º em um plano, que forma uma geometria denominada plana triangular. A Figura 30 mostra dois exemplos (BF_3 e HCO) dessa geometria, observações experimentais mostram que os ângulos de ligação do HCO são diferentes dos idealizados, isso ocorre porque ligações duplas contém maior densidade eletrônica do que ligações simples e, por isso, exercem repulsões um pouco maiores sobre as ligações simples. Geralmente, cada tipo de grupo de elétron exerce uma repulsão ligeiramente diferente, os ângulos de ligação refletem essas diferenças. (TRO, 2017)

Figura 30 Exemplos de Geometrias Planas Triangulares



Fonte: (TRO, 2017)

2.2.5.3 – Geometria Tetraédrica

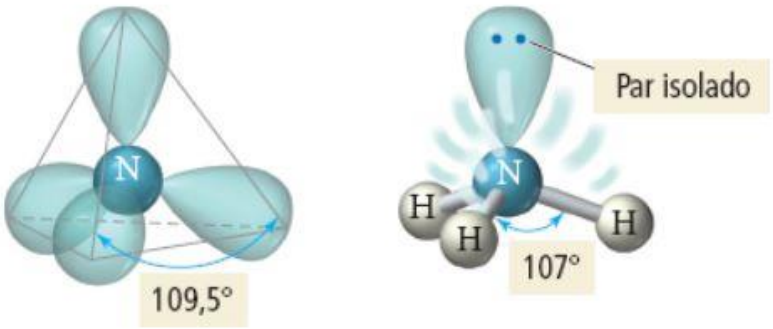
Para quatro grupo de elétrons, o tetraedro é a forma tridimensional que permite a separação máxima entre os grupos, formando ângulos de 109,5º, conforme pôde ser visto anteriormente na Figura 28.

2.2.5.3.1 – Geometria Piramidal Triangular

Quando um dos grupos de elétrons, em um arranjo eletrônico tetraédrico, é um par de elétrons não compartilhado, o arranjo geométrico dos átomos é chamado de

piramidal triangular, conforme pode ser visualizado na molécula de NH_3 (Figura 31). O ângulo das ligações é ligeiramente menor porque um par isolado é atraído somente por um núcleo e exerce uma força repulsiva maior sobre os elétrons vizinhos, comprimindo os outros ângulos. (TRO, 2017)

Figura 31 Exemplo de Geometria Piramidal Triangular

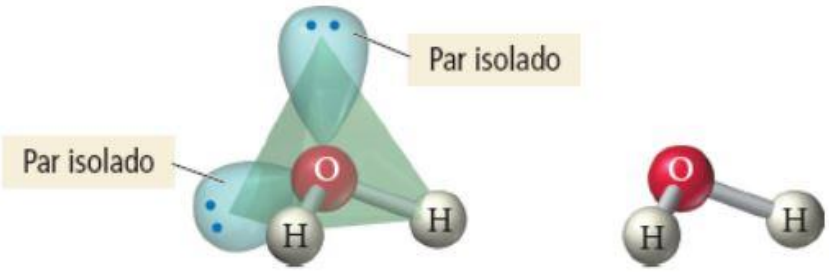


Par de elétrons isolado ocupa mais espaço que um par ligante. Fonte: (TRO, 2017)

2.2.5.3.2 – Geometria Angular

Se dois dos quatro grupos de elétrons forem pares isolados e dois forem pares ligantes a geometria eletrônica também é tetraédrica, mas a geometria molecular é angular. Um exemplo dessa configuração é a molécula de água (H_2O), seus ângulos de ligações são ainda menores (104,5º) do que a molécula de NH_3 (107º) devido à força de repulsão exercida por mais um par de elétrons não compartilhado. (TRO, 2017)

Figura 32 Exemplo de Geometria Angular

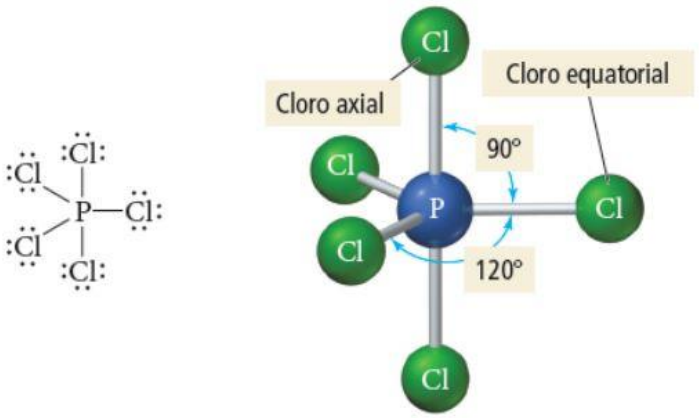


Dois pares isolados geram uma força repulsiva maior do que um par isolado. Fonte: (TRO, 2017)

2.2.5.4 – Geometria Bipiramidal Triangular

Uma estrutura com cinco grupos de elétrons assume uma geometria bipiramidal triangular, três grupos ficam em um único plano (como na geometria plana triangular), enquanto dois grupos ficam acima e abaixo desse plano. Um exemplo de molécula, com essa geometria, é exibido na Figura 33 (PCl_5). (TRO, 2017)

Figura 33 Exemplo de Geometria Bipiramidal Triangular

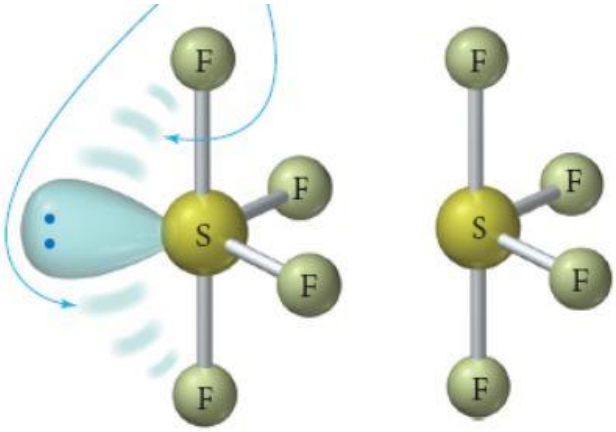


Fonte: (TRO, 2017)

2.2.5.4.1 – Geometria Gangorra

A molécula SF_4, por exemplo, tem um átomo de enxofre cercado de quatro ligações e um par de elétron não compartilhado, ou seja, cinco grupos de elétrons. Nesse caso o par isolado ocupa a posição que minimiza a interação com pares ligantes, isso é, a posição equatorial, que tem apenas duas interações de 90º. O nome da geometria molecular resultante é gangorra ou tetraedro irregular. A Figura 34 mostra uma representação tridimensional da molécula SF_4. Observe que, se o par isolado ocupasse uma posição axial, teria três interações de 90º. (TRO, 2017)

Figura 34 Exemplo de Geometria Gangorra

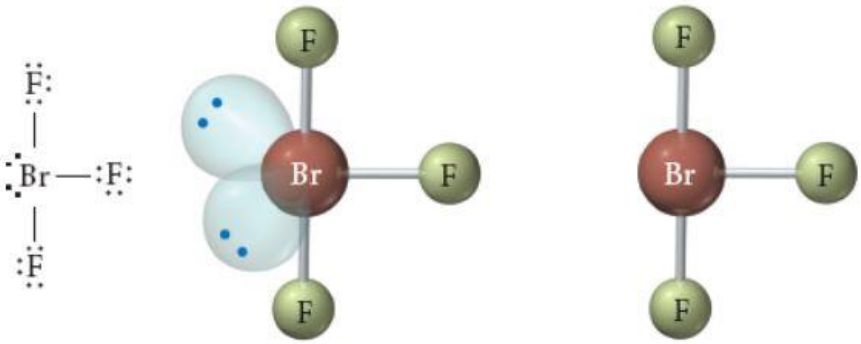


Fonte: Adaptado de (TRO, 2017)

2.2.5.4.2 – Geometria Forma de T

Se dois dos cinco grupos de elétrons forem pares de elétrons não compartilhados, eles ocupam duas das três posições equatoriais, conforme ilustrado na Figura 35. Nessas posições uma repulsão de 90º entre pares isolados é evitada. O nome da geometria é forma de T, um exemplo de molécula com essa geometria é a molécula de BrF_3. (TRO, 2017)

Figura 35 Exemplo de Geometria Forma de T

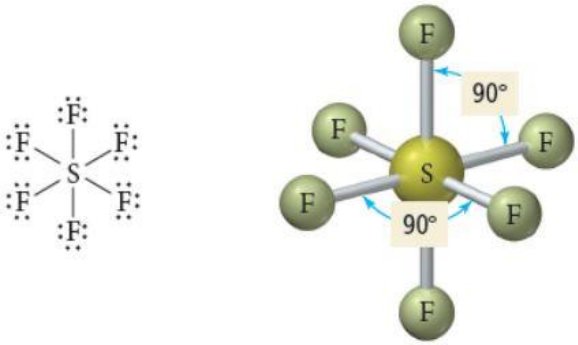


Fonte: (TRO, 2017)

2.2.5.5 – Geometria Octaédrica

A molécula SF_6 (Figura 36), por exemplo, tem seis grupos de elétrons ao redor do átomo central, todos assumindo ângulos de 90º. Quatro desses grupos estão em um único plano, o quinto e sexto grupo estão acima e abaixo do plano, respectivamente. O fato de existirem oito lados em sua forma geométrica é o que faz com que essa geometria seja chamada de octaédrica. (TRO, 2017)

Figura 36 Exemplo de Geometria Octaédrica

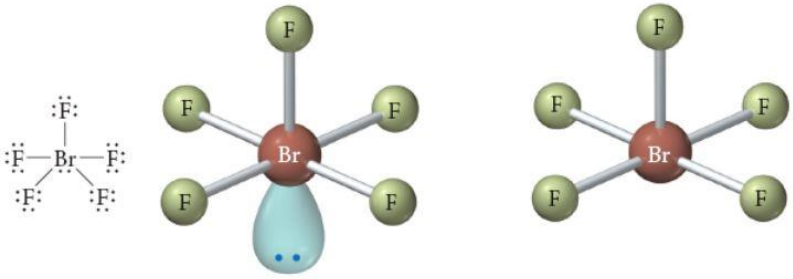


Fonte: (TRO, 2017)

2.2.5.5.1 – Geometria Piramidal Quadrada

Como todos os ângulos em uma forma octaédrica são iguais, o par de elétrons não compartilhado pode estar em qualquer uma das posições. Como exemplo, considere a estrutura do BrF_5, exibida na Figura 37, sua geometria é denominada piramidal quadrada. (TRO, 2017)

Figura 37 Exemplo de Geometria Piramidal Quadrada

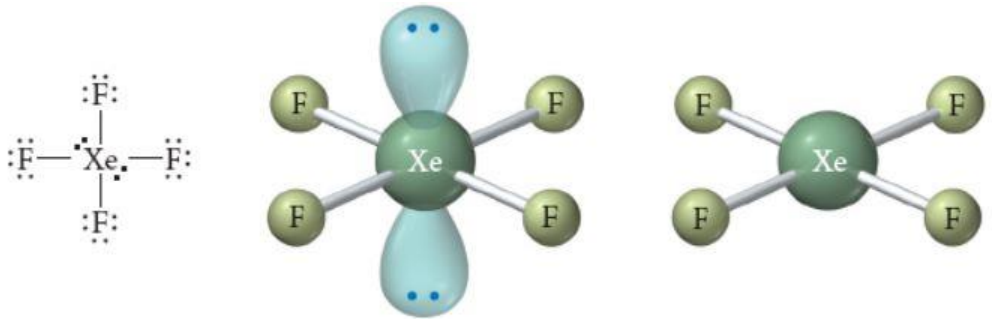


Fonte: (TRO, 2017)

2.2.5.5.2 – Geometria Quadrada Plana

Se houverem dois pares isolados em uma estrutura com seis grupos de elétrons, eles assumem posições transversais umas às outras, de forma a minimizar repulsões do tipo par isolado – par isolado, por isso o nome da geometria é plana quadrada. Um exemplo é o XeF_4, ilustrado na Figura 38. (TRO, 2017)

Figura 38 Exemplo de Geometria Plana Quadrada



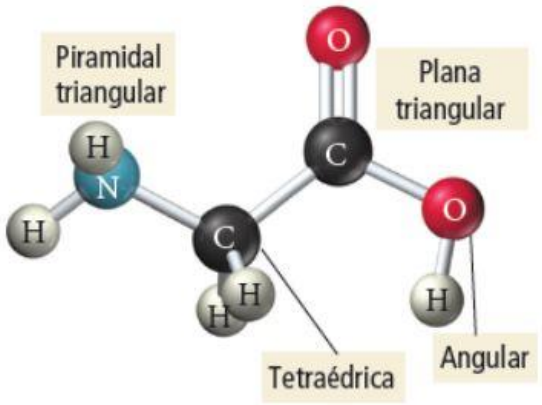
Fonte: (TRO, 2017)

2.2.6 – Previsão de Formas Moleculares Maiores

Os princípios anteriores podem ser aplicados em cada átomo interior de uma molécula maior, para prever sua forma. Por exemplo, a glicina (Figura 39) tem quatro átomos centrais: um átomo de nitrogênio com um par isolado, um átomo de oxigênio com dois pares isolados e dois átomos de carbono (ambos sem pares de elétrons não compartilhados). Para determinar a forma da glicina, especificamos primeiro a geometria em torno de cada átomo interior: (TRO, 2017)

- Nitrogênio: Geometria Piramidal Triangular;
- Carbono (à esquerda): Geometria Tetraédrica;
- Carbono (à direita): Geometria Plana Triangular;
- Oxigênio: Geometria Angular.

Figura 39 Forma de Molécula com Outros Átomos Interiores



A geometria de cada átomo interior pode determinar a forma da molécula inteira. Fonte: (TRO, 2017)

2.2.7 – Hibridização

Os orbitais em uma molécula não são sempre os mesmos que os orbitais em um átomo. Para explicar as geometrias moleculares em termos de orbitais é preciso compreender a hibridização de orbitais, uma técnica quantomecânica. A mistura de orbitais atômicos do mesmo átomo forma orbitais híbridos, ou seja, um orbital híbrido é produto da mistura de dois ou mais orbitais atômicos comuns. Os orbitais híbridos minimizam a energia da molécula ao maximizar a sobreposição de orbitais em uma ligação. Átomos interiores (centrais) têm uma tendência maior à hibridização, pois formam a maior parte das ligações, consequentemente, átomos terminais que formam menos ligações, têm uma tendência menor de hibridizar. As seguintes afirmações são válidas sobre a hibridização: (TRO, 2017; VOLLHARDT; SCHORE, 2013)

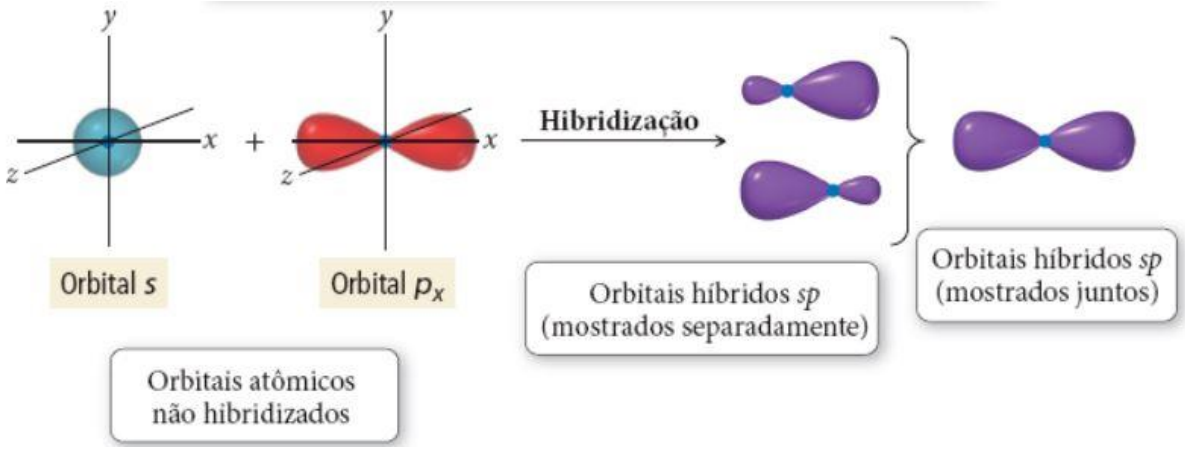
- O número total de orbitais é conservado;
- O tipo de hibridização ocorrida é a que produz a menor energia global;
- Combinações de orbitais comuns determinam as formas dos híbridos;

2.2.7.1 – Hibridização *sp*

Quando uma função de onda 2s e uma 2p se misturam, dois novos orbitais híbridos são formados, os orbitais têm 50% de caráter s e 50% de caráter p e são

chamados de orbitais sp (Figura 40). Os lobos maiores, chamados de lobos frontais, formam um ângulo de 180º entre si, os lobos menores têm sinais opostos aos frontais. Os outros dois orbitais p não sobrem alterações. (VOLLHARDT; SCHORE, 2013)

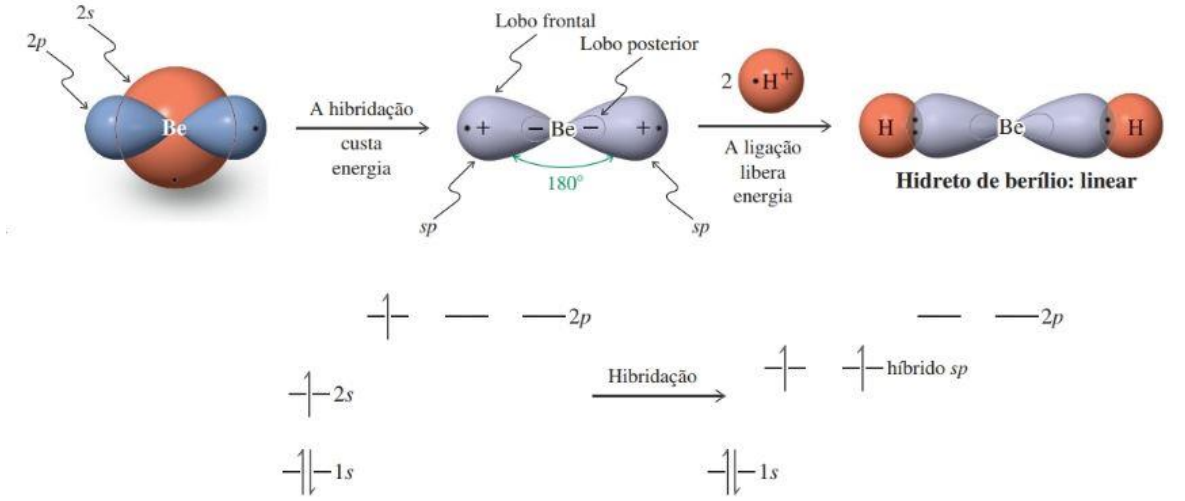
Figura 40 Orbitais Híbridos *sp*



Os outros dois orbitais p não aparecem na imagem. Fonte: (TRO, 2017)

Um exemplo de molécula com essa hibridização é o hidreto de berílio (Figura 41), BeH_2. A teoria de repulsão dos elétrons de valência prediz que o composto deve ser linear. Experimentos realizados confirmam essa predição mostrando que as ligações têm o mesmo comprimento. (VOLLHARDT; SCHORE, 2013)

Figura 41 Estrutura Linear dos Orbitais Híbridos sp



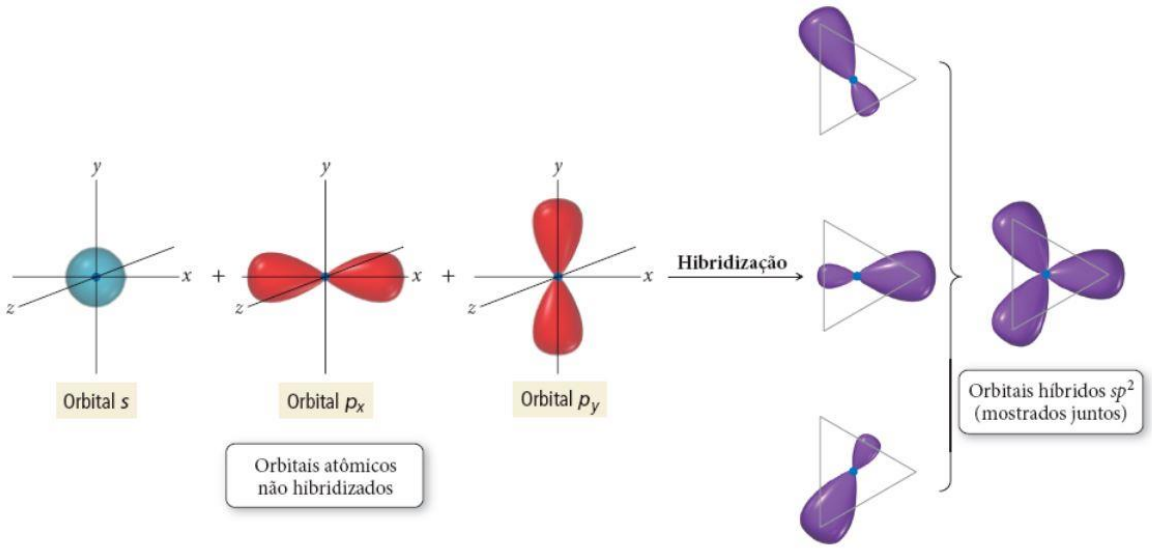
Orbitais não envolvidos foram omitidos. Fonte: (VOLLHARDT; SCHORE, 2013)

Observe na Figura 41, as alterações de energia resultantes da hibridização, os orbitais híbridos *sp* têm energia intermediária entre o orbital 2s e o 2p. As energias dos orbitais 1s e 2p não utilizados permanecem inalteradas. Repare também que a hibridização não muda o número de orbitais disponíveis para ligações. (TRO, 2017; VOLLHARDT; SCHORE, 2013)

2.2.7.2 – Hibridização *sp²*

Orbitais híbridos sp^2 (Figura 42) dão origem a estruturas trigonais. A mistura de um orbital *s* com dois orbitais *p* leva a três novos orbitais híbridos, chamados de orbitais sp^2, indicando que eles têm 67% de caráter p e 33% de caráter s. O terceiro orbital 2p permanece inalterado, portanto, o número de orbitais total permanece igual (quatro). (VOLLHARDT; SCHORE, 2013)

Figura 42 Os Orbitais Híbridos sp^2

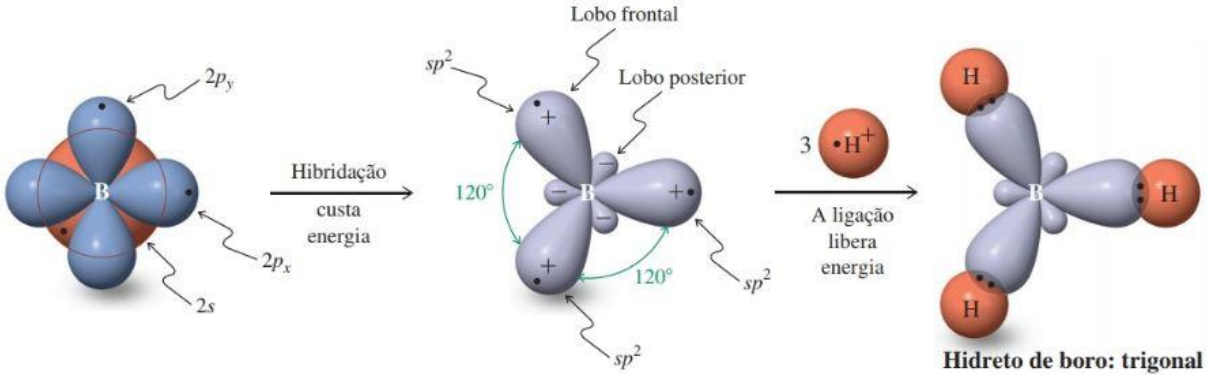


Fonte: (TRO, 2017)

A Figura 43 mostra um exemplo dessa hibridização. O borano (BH_3) apresenta uma geometria trigonal plana. Cada um dos lobos frontais dos três orbitais híbridos superpõe um orbital 1s do hidrogênio. A hibridização, portanto, minimiza a repulsão dos elétrons e leva a ligações mais fortes (devido a melhor superposição). O orbital

2p remanescente é perpendicular ao plano formado, ele fica vazio e permanece inalterado. (VOLLHARDT; SCHORE, 2013)

Figura 43 Estrutura Trigonal Planar da Hibridização sp²



Orbital p não envolvido foi omitido. Fonte: (VOLLHARDT; SCHORE, 2013)

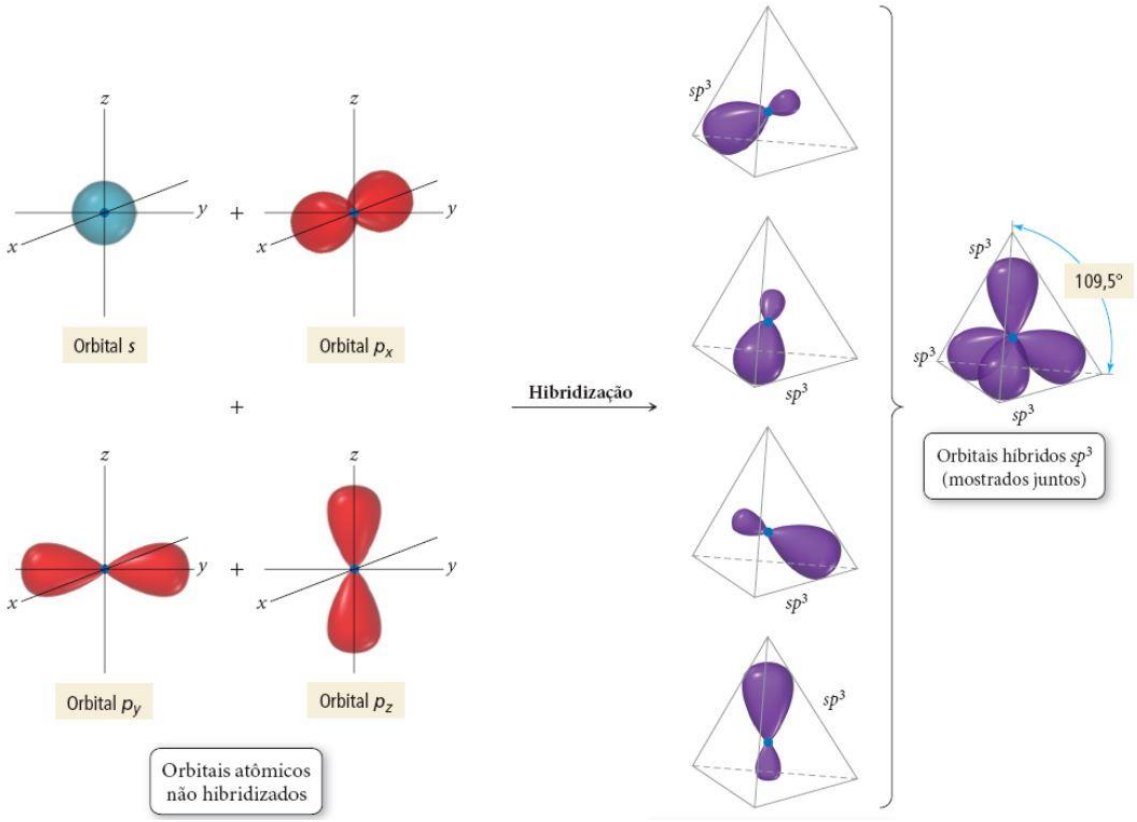
2.2.7.3 – Hibridização *sp³*

Orbitais híbridos sp^3 (Figura 44) dão origem a estruturas tetraédricas (ângulos de 109,5º), nessa hibridização, um orbital *s* é hibridizado com três orbitais *p,* formando quatro novos orbitais, cada um com ¾ (75%) de caráter *p* e ¼ (25%) de caráter *s*. (VOLLHARDT; SCHORE, 2013)

Um elemento que frequentemente se hibridiza dessa forma é o carbono, cujas ligações são as mais importantes para a química orgânica e é responsável pela diversidade da química orgânica. Por exemplo, podemos explicar a geometria tetraédrica do metano (Figura 45), CH_4, ocupando os orbitais híbridos sp^3 do carbono com os quatro elétrons de valência. Os elétrons ocupam os orbitais individualmente, com spins paralelos, segundo a regra de *Hund,* dessa forma, o carbono pode formar quatro ligações com os átomos de hidrogênio. (TRO, 2017; VOLLHARDT; SCHORE, 2013)

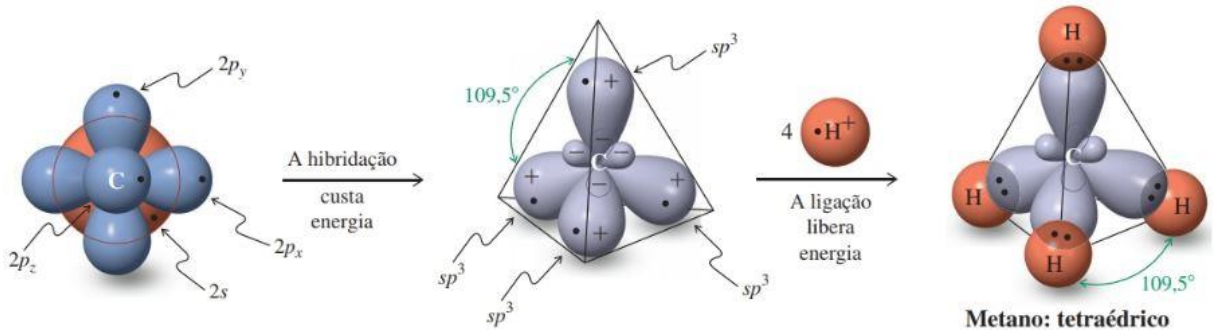
Os orbitais híbridos também podem acomodar pares de elétrons não compartilhados. Por exemplo, a amônia (NH_3) tem três de seus orbitais híbridos envolvidos em ligações com três átomos de hidrogênio, mas o quarto orbital híbrido contém um par isolado. A presença desse par diminui a tendência à hibridização nos orbitais do nitrogênio, pois existe um número menor de ligações formadas. (TRO, 2017)

Figura 44 Os Orbitais Híbridos sp³



Fonte: (TRO, 2017)

Figura 45 Estrutura Tetraédrica da Hibridização sp³



Fonte: (VOLLHARDT; SCHORE, 2013)

CAPÍTULO 3

MATERIAIS E MÉTODOS

Nesse capítulo as ferramentas e métodos propostos para o desenvolvimento desse trabalho serão apresentados em detalhes. Assim como descrição dos procedimentos utilizados para desenvolvimento dos aplicativos propostos nesse trabalho. Começando pelo modelo de processo de engenharia de software e seguindo para análise, design e implementação dos programas. Para isso é importante compreender como é arquitetura de um projeto no motor de jogo *Unity,* assim como conhecer seus principais componentes. Cada componente que for apresentado será relacionado com algum aspecto de algum dos aplicativos desenvolvidos.

Após a etapa de desenvolvimento e distribuição dos aplicativos, um questionário de avaliação dos aplicativos foi preenchido pelos usuários. A descrição do questionário, sua forma de distribuição e de análise são apresentados.

3.1 – *Unity*

A ferramenta mais utilizada foi o motor de jogo *Unity.* Antes de seguir para a análise, *design* e desenvolvimento dos aplicativos (no capítulo seguinte) é preciso compreender como é a arquitetura de um software nesse motor de jogo e como funcionam seus principais componentes.

Existem outros motores de jogo disponíveis gratuitamente no mercado, *Unity* foi escolhido devido a experiência do autor com a ferramenta. Adquirida durante o desenvolvimento de um provador virtual de roupas, durante seu projeto de mestrado. (ARAÚJO, 2015)

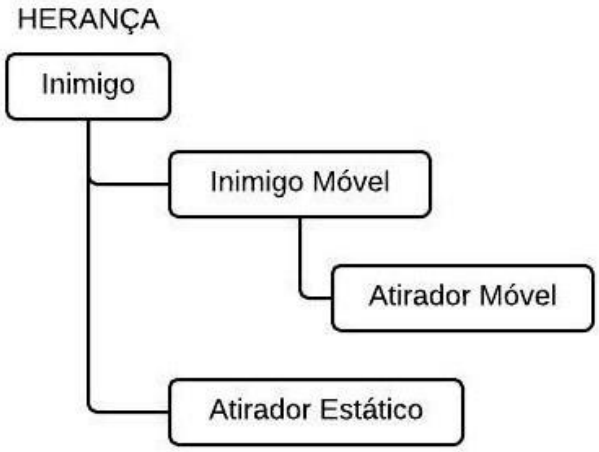
Unity foi construído com desenvolvimento de jogo como objetivo, mas criar jogos não é tão diferente de criar outros *softwares* como aplicações *desktop* ou *web.* A principal diferença entre jogos e outros programas está na quantidade de interações. Jogos são muito mais interativos que sites, por exemplo, e por isso envolvem tipos de

código bem diferentes. Mesmo assim, as habilidades necessárias e processos envolvidos em ambos são semelhantes. Alguém familiarizado com escrita de código de programação terá que apenas traduzir esse conhecimento para o campo da programação de jogos. (CARPENTER, 2015)

3.1.1 – Introdução ao Sistema de Componentes

Uma das vantagens do *Unity* está na modularização que seu sistema de componentes fornece durante a criação de objetos de jogo. No sistema baseado em componentes, é possível combinar “pacotes” de funcionalidades. Objetos são construídos como uma coleção de componentes, ao invés de uma hierarquia estrita de classes. É uma abordagem diferente (geralmente mais flexível) de fazer programação orientada a objetos, onde objetos de jogo são construídos usando composição ao invés de herança. CARPENTER (2015) faz uma comparação (Figura 46 e Figura 47) entre usar composição e herança para desenvolver um sistema de reaproveitamento de código de comportamento.

Figura 46 Sistema de divisão de comportamento usando herança



Fonte: Adaptado de (CARPENTER, 2015)

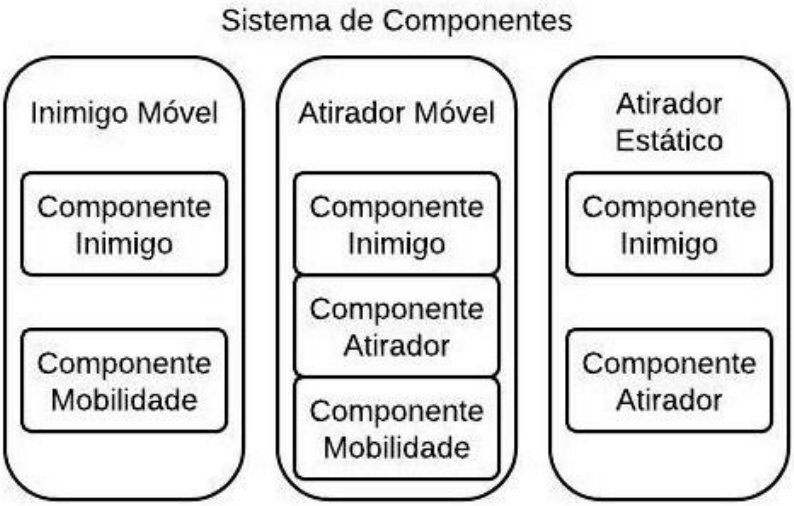
Observe, na Figura 46, que foi preciso criar duas classes distintas para os diferentes comportamentos entre os inimigos que são estáticos e os que são móveis. Na imagem é possível notar que houve duplicação do comportamento “Atirador”, toda

alteração de comportamento e criação de um novo tipo de inimigo precisará de uma grande quantidade de refatoração, devido a esse tipo de duplicação. (CARPENTER, 2015)

Refatorar (*refactor*, em inglês) é o processo de mudar um sistema de *software* de um modo que não altere o comportamento externo do código, mas melhore a estrutura interna. (FOWLER et al., 1999)

A Figura 47 mostra o mesmo exemplo da Figura 46 implementado com o sistema de componentes, para que as duas abordagens sendo aplicadas na resolução de um mesmo problema possam ser comparadas. Observe que, com a abordagem de composição podemos escrever um único componente atirador, que pode ser reutilizado em qualquer lugar quando necessário. Nesse caso, tanto em inimigos móveis quanto estáticos. (CARPENTER, 2015)

Figura 47 Sistema de Divisão de Comportamentos usando Composição



Fonte: Adaptado de (CARPENTER, 2015)

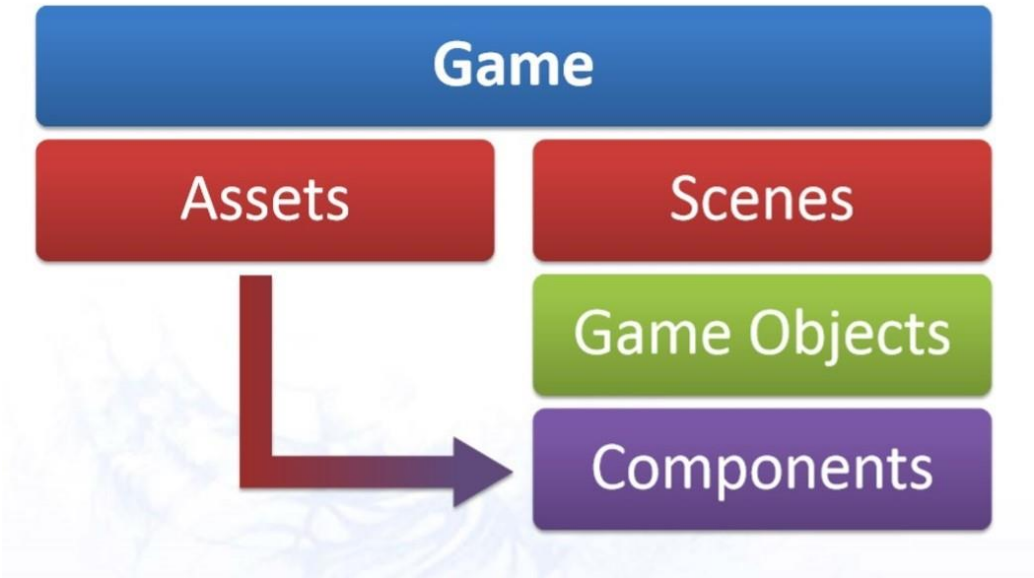
3.1.2 – Arquitetura de um Projeto no *Unity*

Um projeto no *Unity* é dividido em partes chamadas de cenas. Cenas podem ser usadas para criar um menu principal, níveis individuais e qualquer outra coisa. A Figura 48 ilustra a arquitetura de um software (*Game*) no *Unity*. Nela é possível

observar que um software é composto de muitas cenas (*Scenes*) e arquivos (*Assets).* A cena, por sua vez, contém objetos de jogo (G*ame Objects)* e um objeto de jogo tem muitos componentes, os quais estão relacionados algumas vezes com os *assets*. (UNITY TECHNOLOGIES, 2017)

O aplicativo “Configuração Eletrônica”, por exemplo, é dividido em três cenas. Uma para o programa, uma para o menu e outra para a leitura da parte teórica.

Figura 48 Arquitetura de um Projeto no *Unity*



Fonte: (GOUVEIA, 2013)

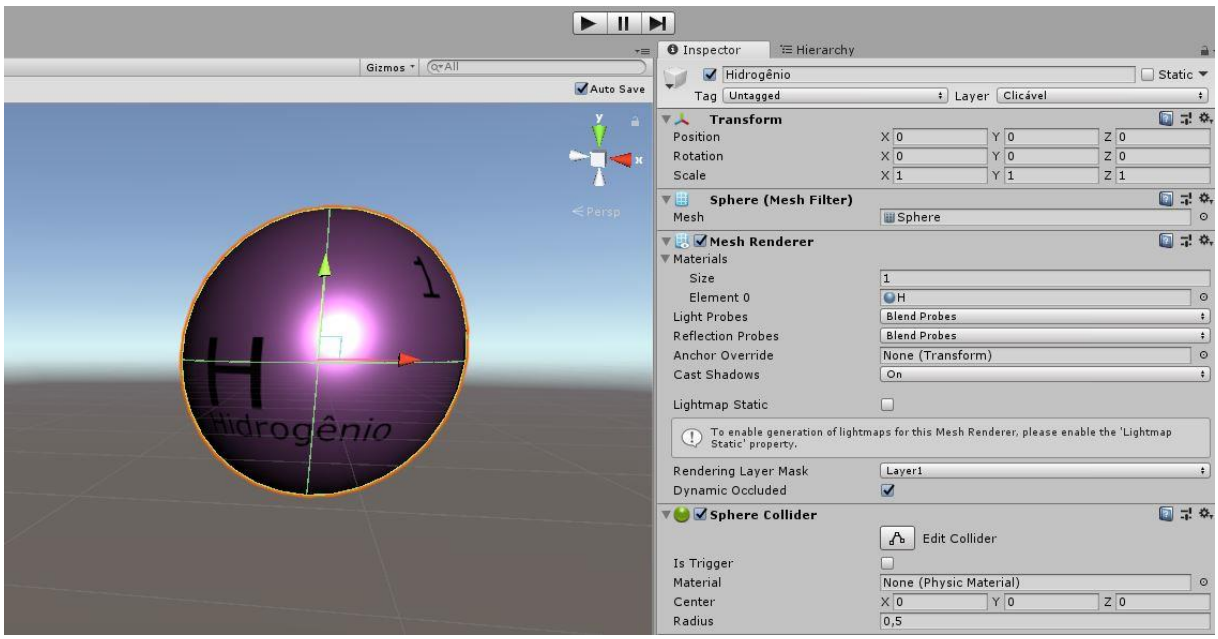
3.1.3 – Objetos de Jogo (*Game Objects*)

O termo objeto de jogo não deve ser confundido com objeto, no sentido de instância de uma classe em orientação a objeto. *Game Object* é o conceito mais importante no Editor do *Unity*. Tudo o que existir em uma cena é um objeto de jogo. No entanto, um *Game Object* não pode fazer nada por conta própria. Em outras palavras, ele não tem comportamento, você precisa adicionar componentes a ele antes que ele possa se tornar um botão, um ambiente ou um efeito especial, por exemplo. (TECHNOLOGIES, 2018)

Dependendo do objeto que deseja criar, você precisará adicionar diferentes combinações de componentes. Por exemplo, um objeto de luz é criado anexando um componente *Light* a um *Game Object*, enquanto um som pode ser criado ao anexar um componente "*Audio Source"* a esse ou outro *Game Object*. Cada componente pode ter uma série de propriedades configuráveis. (UNITY MANUAL, 2017)

Uma esfera, no *Unity*, tem os componentes exibidos na Figura 49. Dois componentes: o *Mesh Filter* e o *Mesh Renderer*, para desenhar a superfície da esfera e um componente *Sphere Collider* para representar o volume sólido do objeto em termos de física. O *Mesh Filter* é o componente responsável por dizer qual malha (*mesh)* será desenhada (renderizada). O *Mesh Renderer* é responsável por dizer como esse objeto será renderizado (texturas, materiais, shaders) e o *Sphere Collider* diz qual é o formato desse objeto para o motor de física.

Figura 49 Componentes de uma Esfera



Fonte: Próprio Autor

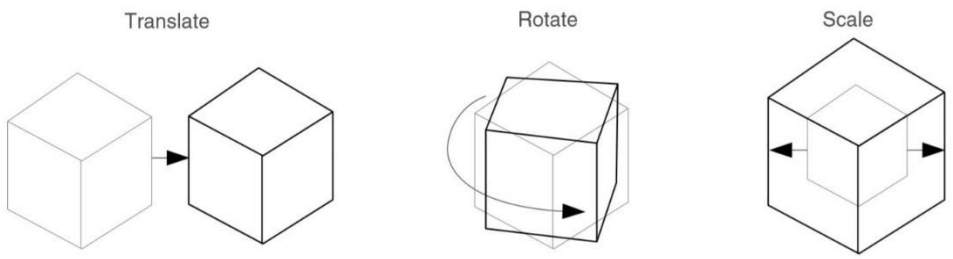
3.1.4 – O Componente *Transform*

É impossível criar um *Game Object* sem um componente *Transform*. Este componente define a posição, a direção e a tamanho do objeto de jogo no mundo do jogo

e na janela de visão da cena. Toda transformada geométrica aplicada faz uso desse componente. A Figura 50 ilustra três transformadas: Translação (*Translate),* Rotação (*Rotate*) e Escala (*Scale).* As linhas mais claras exibem o estado dos objetos antes da transformação. (CARPENTER, 2015)

Esse componente está ligado a um conceito chamado “parentalidade”, que é uma parte importante do trabalho com os *Game Objects*. O aplicativo Geometria Molecular, por exemplo, permite que os usuários rotacionem as geometrias. A parentalidade simplifica o processo de rotação de todos os elementos simultaneamente.

Figura 50 Transformadas Geométricas



Fonte: (CARPENTER, 2015)

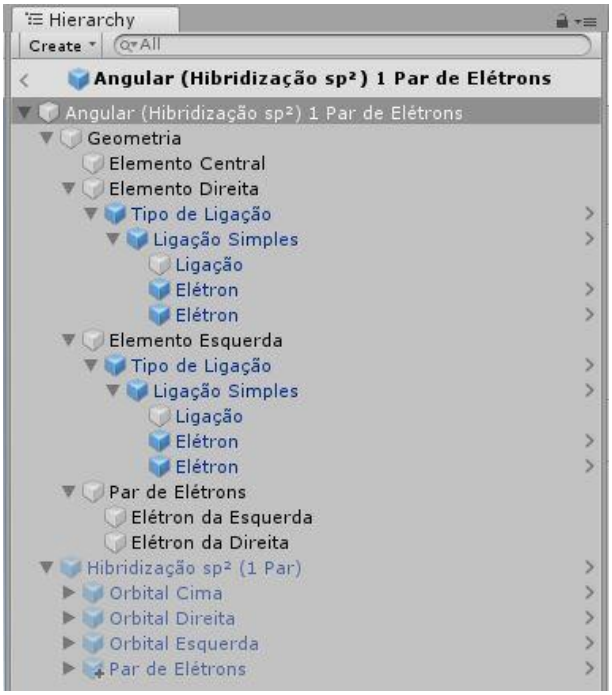
3.1.5 – Parentalidade

Quando um *Game Object* é pai de outro objeto de jogo, o *Game Object* filho moverá, girará e escalará exatamente como o pai fizer, ou seja, o componente *Transform* do filho é influenciado pelo mesmo componente no objeto pai. Qualquer objeto pode ter múltiplos filhos, mas apenas um pai. Esses níveis múltiplos de relações pai-filho formam uma hierarquia de *Transform*. O objeto no topo de uma hierarquia (ou seja, o único objeto na hierarquia que não possui um pai) é conhecido como raiz. Desativar um objeto pai, também desativa todos os objetos abaixo, na hierarquia. Algumas vezes, os objetos são organizados hierarquicamente apenas para se manter a organização em uma cena cheia de objetos de jogo. Em outras situações, a parentalidade é interessante para *design* do código de programação. É possível enviar mensagens para cima, e para baixo, na hierarquia, através de *scripts*. (CREIGHTON, 2010)

É importante tomar cuidado para não confundir esse conceito com o conceito de herança em orientação a objetos, pois não estão relacionados de forma alguma.

Todas as geometrias no aplicativo de Geometria Molecular, são organizadas de forma hierárquica. A Figura 51 exibe um exemplo, nela o objeto de jogo “Angular” é pai dos *Game Objects* “Geometria” e “Hibridização sp²”. Ambos são filhos de “Angular” e ambos, por sua vez, também possuem filhos. Os filhos do objeto “Geometria” são “Elemento Central”, “Elemento Direita” e “Elemento Esquerda”. Alguns desses também possuem filhos. Dessa forma, podemos organizar objetos de jogo em estruturas mais complexas.

Figura 51 Exemplo de Parentalidade



Hierarquia de um único objeto de jogo que representa uma geometria. Fonte: Próprio Autor

3.1.6 – Câmera (Tipos de Projeções)

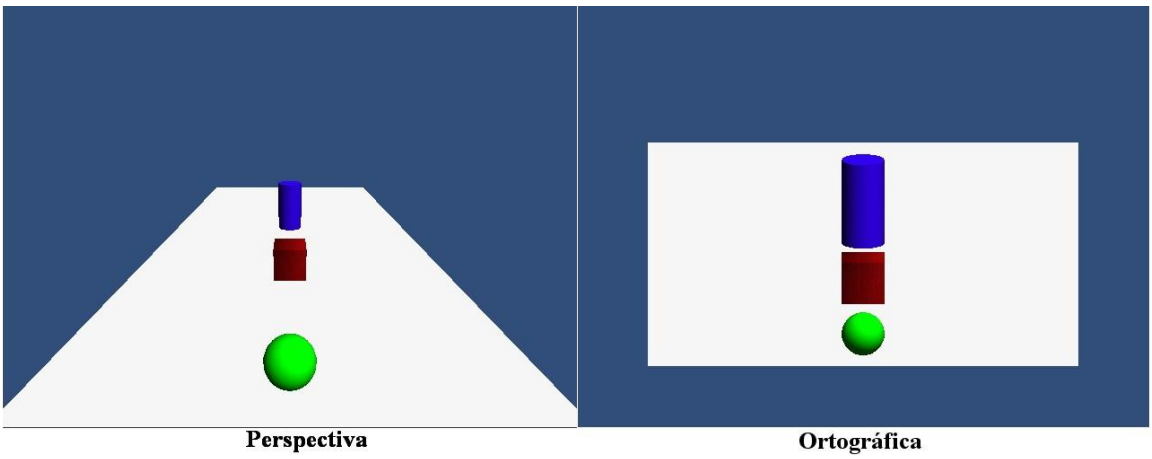
O componente câmera tem uma propriedade chamada projeção, que pode ser configurada com dois valores: Perspectiva ou Ortográfica.

Na projeção em perspectiva o tamanho dos objetos é alterado de acordo com a distância deles em relação a câmera. Em outras palavras, nesse tipo de projeção os

objetos serão desenhados com tamanhos maiores conforme se aproximam da câmera e tamanhos menores conforme se afastam da câmera. Com o tipo de projeção selecionado no modo ortográfico, o tamanho dos objetos não será afetado de acordo com distância deles em relação a câmera. É comum usar esse tipo de projeção em jogos 2D. (SMITH; QUEIROZ, 2015)

A Figura 52 ilustra uma cena sendo exibida de uma mesma posição, tendo apenas o atributo projeção da câmera alterado. Observe que os objetos foram posicionados um atrás do outro. A projeção em perspectiva permite que tenhamos a noção da distância entre os objetos, por isso é geralmente utilizada em jogos tridimensionais. (CREIGHTON, 2010)

Figura 52 Comparação entre os tipos de projeção



Fonte: (ARAÚJO, 2015)

As câmeras, nas cenas dos aplicativos desenvolvidos nesse trabalho, foram configuradas para projetarem ortograficamente.

3.1.7 – O Arquivo de *Script*

Os *scripts no Unity* são executados por uma versão modificada de uma biblioteca chamada Mono, uma implementação de código aberto do *framework .Net* da *Microsoft.* (PASSOS; JR, 2009)

É possível adicionar funcionalidades ao Editor do *Unity* através de código de programação, usando a API de *Scripting* do Unity. Um *script* é, de forma resumida, um algoritmo, ou seja, uma série de instruções para o objeto de jogo seguir. (CREIGHTON, 2010)

A palavra *script* pode gerar confusão, porque é comumente encontrada quando o assunto é código *JavaScript* sendo executado no seu navegador. Essa palavra costuma se referir a pequenos programas utilitários independentes. *Scripts* no contexto de *Unity,* no entanto*,* estão mais próximos de classes da Orientação a Objetos. *Scripts* anexados ao *Game Objects* na cena são instâncias (objetos de uma classe). (CARPENTER, 2015)

Quando você cria um *script* e o anexa a um *Game Object*, o *script* aparece na janela de inspeção do *Game Object*, assim como qualquer outro componente. Isso ocorre porque os *scripts* se tornam componentes quando você os salva em seu projeto. Em termos técnicos, qualquer *script* que você desenvolve será compilado como um tipo de componente. O Editor do *Unity* trata seu *script* como um componente interno. Você define as propriedades do *script* que devem ser expostas no Inspetor (as variáveis públicas), e o Editor executa qualquer funcionalidade que você escreveu. Um *script* se conecta com o funcionamento interno do motor de jogo *Unity* ao implementar qualquer classe que herde da classe **MonoBehaviour**. Ao anexar um componente de *script* a um *Game Object*, uma instância do objeto definido pelo *script* é criada. (UNITY MANUAL, 2017)

A Figura 53 mostra a anatomia padrão de uma classe C# que se conecta ao Editor do *Unity*. As funcionalidades básicas são implementadas sobrescrevendo duas funções herdadas da classe **MonoBehaviour**. A função Update é o lugar para colocar o código que irá lidar com a atualização em cada *frame* para o *Game Object* ao qual o script foi anexado. Isso pode incluir movimentos, ativação de ações e resposta à entrada (*Input*) do usuário. Basicamente tudo o que precisa ser tratado ao longo do tempo durante o jogo. Qualquer código escrito nessa função será executado toda vez que um quadro (*frame*) for desenhado. (UNITY TECHNOLOGIES, 2017)

Para habilitar a função *Update* para fazer seu trabalho, muitas vezes é útil inicializar variáveis, configurar e ler preferências e criar conexões com outros *Game*

Objects antes de qualquer ação do jogo começar. A função *Start* será chamada pelo *Unity* antes do início do jogo (ou seja, antes que a função *Update* seja chamada pela primeira vez) e é um local ideal para fazer qualquer inicialização. As cenas são compostas de vários *Game Objects* e cada um deles pode conter vários componentes de *script.* Isso significa que existirão diversas funções *Start*, uma para cada um desses *scripts,* sendo executada quando a cena for carregada. (SMITH; QUEIROZ, 2015)

Figura 53 Script Padrão de uma Classe Filha de *MonoBehaviour*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NewBehaviourScript : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Fonte: Próprio Autor

Por padrão o *Unity* adiciona algumas bibliotecas como *UnityEngine* e *System.Collections*. É preciso adicionar bibliotecas se for necessário acessar classes ou funções que não estejam nessas duas bibliotecas. Por exemplo, se precisar usar as classes de interface de usuário no seu *script*, precisará adicionar a biblioteca de *User Interface* - UnityEngine.UI. (NORTON, 2013)

A inicialização de um objeto que herda de **MonoBehaviour** não deve feita usando uma função de construtor, como esperado em orientação a objetos. Isso

ocorre porque a construção de objetos, nesse caso, é tratada pelo Editor do *Unity* e não ocorre no início da jogabilidade como você poderia esperar. Se você tentar definir um construtor para um componente de script, ele irá interferir com o funcionamento normal do *Unity* e pode causar grandes problemas no projeto. (UNITY, 2018)

É possível ativar e desativar objetos de jogo e componentes. Isso pode ser feito usando sua propriedade *activeSelf* através de um *script* ou com a caixa de seleção na janela de inspeção. Quando um objeto principal é desativado, todos os seus objetos filhos ficam inativos, ou seja, toda a hierarquia do pai é desativada. A função *Start* de um *Script* só é executada se o componente, e o objeto de jogo do script, estiverem ativados. (BLOW, 2004)

3.1.8 – Prefabs

Muitas vezes é conveniente criar um *Game Object* na cena, adicionando componentes e definindo suas propriedades com valores apropriados. Isso pode criar problemas, no entanto, quando você tem um objeto que é reutilizado na cena várias vezes. Simplesmente copiar o objeto produzirá duplicatas, mas todas serão editáveis independentemente, como se fossem diferentes. Geralmente, você quer que todas as instâncias de um objeto específico tenham as mesmas propriedades. Então, quando você edita um desses objetos na cena, você preferiria não ter que fazer a mesma edição repetidamente em todas as cópias.

O *Unity* possui um tipo de recurso chamado *prefab,* que permite guardar um objeto de jogo completo com seus componentes e propriedades configuradas. O *prefab* atua como um modelo a partir do qual você pode criar instâncias daquele objeto na cena. Todas as edições feitas em um arquivo *prefab* são imediatamente refletidas em todas as instâncias produzidas a partir dele, mas você também pode substituir componentes e configurações para cada instância individualmente. (UNITY MANUAL, 2017)

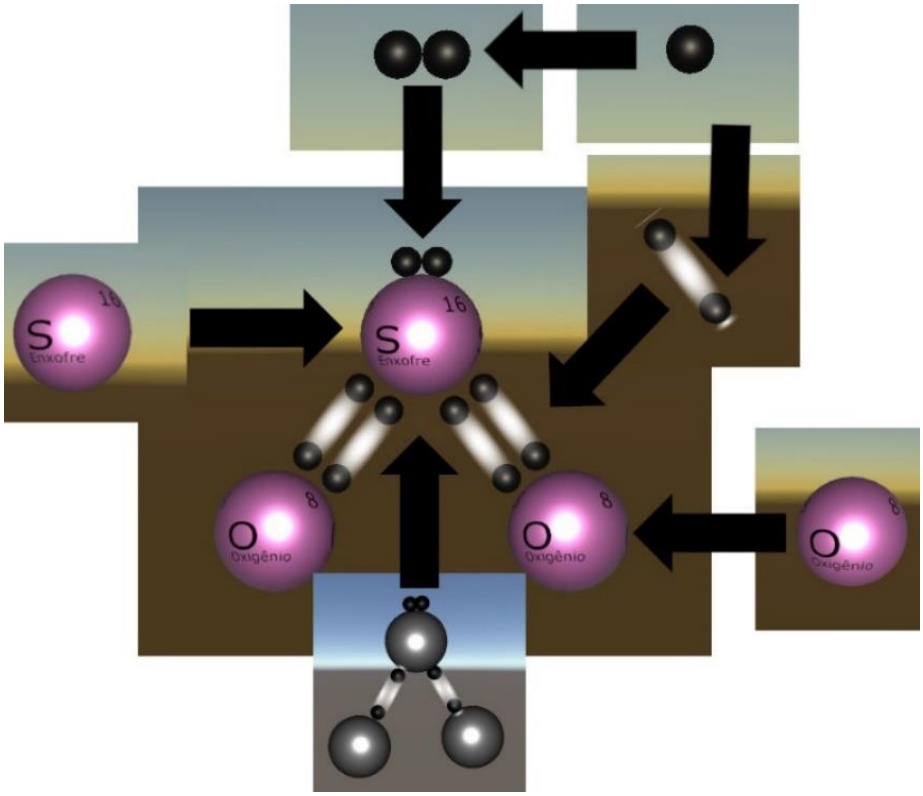
Um *prefab* é um tipo de *asset* criado a partir de um objeto de jogo, ele pode ser utilizado muitas vezes em uma mesma cena e em várias cenas ao mesmo tempo no decorrer do seu projeto. Quando você adiciona um *prefab* em uma cena, você cria

uma instância dele, mas ela continua ligada ao objeto *prefab* original. Essa ligação faz com que alterações no *prefab* principal afetem todas as instâncias em todas as cenas. (BLOW, 2004)

Os *prefabs* são úteis quando você deseja instanciar *Game Objects* complicados em tempo de execução, a partir de código de programação. Em jogos, por exemplo, é comum a utilização de *prefabs* em objetos interativos como itens, personagens ou veículos. Geralmente cria-se um conjunto complexo de objetos de jogo na cena como, por exemplo, o exterior de um carro junto com a suspensão, rodas e corpo para colisão. Quando finalizado, o objeto da cena é armazenado no projeto como um *prefab* a partir desse conjunto e remove-se os objetos originais da cena. A partir de agora, sempre que esse tipo de objeto for necessário, em qualquer cena, utiliza-se a instância do *prefab*.(BLOW, 2004)

No aplicativo de Geometria Molecular, por exemplo, todas as geometrias são *prefabs* que foram construídos a partir de *prefabs* menos complexos. Alterar qualquer *prefab* atualiza todas as geometrias que o utilizam. Todo objeto de jogo, em ambos os projetos desse trabalho, que se repetem, em uma ou mais cenas, são armazenadas como *prefab* para facilitar o processo de manutenção e atualização dos aplicativos. A Figura 54 mostra um exemplo de *prefab* usado em um dos projetos propostos nesse trabalho. Muitos outros foram criados seguindo a mesma ideia.

Figura 54 Exemplo de um Prefab de Geometria



Todas as imagens menores são *prefabs* que existem individualmente. Fonte: Próprio Autor

Em outras palavras, *prefabs* são uma abordagem flexiva para definir visualmente objetos interativos. São objetos finalizados, com os componentes já adicionados e configurados, que não existem em nenhuma cena em específico, mas apenas como um arquivo que pode ser copiado em qualquer cena. Esse processo de copiar pode ser feito manualmente, pelo editor, arrastando o *prefab* da janela de projeto de volta para a janela de hierarquia, para garantir que o objeto seja o mesmo em todas as cenas, ou através de código de programação usando a função *Instantiate* (instanciar, em inglês). (CARPENTER, 2015)

O termo para se referir a uma das cópias de um *prefab* é instância. Deve-se fazer uma analogia com a palavra instância da terminologia de orientação a objetos, um objeto criado a partir de uma classe. *Prefab* é o objeto de jogo que existe fora de qualquer cena, instância se refere a uma cópia desse objeto de jogo que foi colocado em alguma cena. Instanciar é a ato de criar uma instância. (CARPENTER, 2015)

3.1.9 – Materiais, Texturas e *Shaders*

O *rendering* no *Unity* envolve Materiais (*Materials*), *Shaders* e Texturas (*Textures*). Portanto existe uma estreita relação entre eles e isso faz com que seja difícil falar de cada um isoladamente.

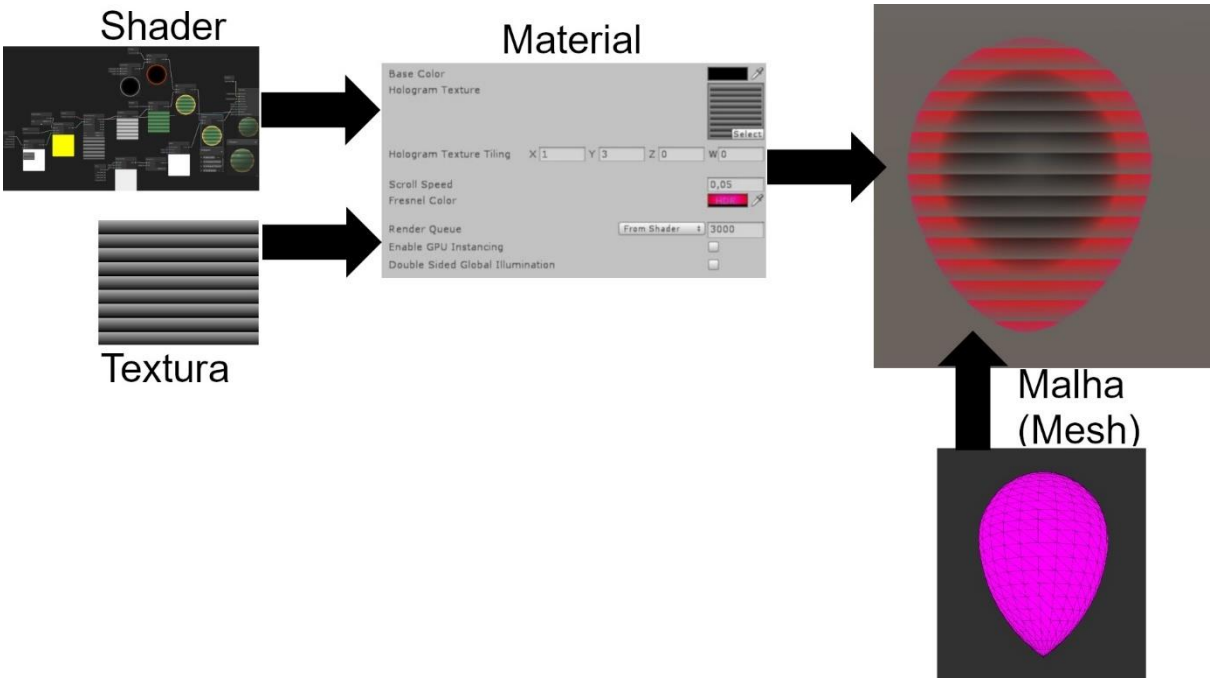
Um material é uma descrição completa das propriedades visuais de uma malha. Essa descrição incluí especificações de quais texturas estão mapeadas para sua superfície e propriedades de alto nível como quais programas de *shaders* usar ao renderizar a malha. O material também pode guardar informações de entradas para os parâmetros dos *shaders* e informações que controlam as funcionalidades da placa de vídeo. (GREGORY, 2009)

Os materiais são definições de como uma superfície deve ser renderizada, incluindo referências a texturas usadas, informações de revestimento, tons de cor e muito mais. As opções disponíveis para um material dependem do *shader* que o material está usando. (UNITY TECHNOLOGIES, 2017)

Shaders são scripts que contêm os algoritmos matemáticos para calcular a cor de cada pixel que será renderizado, com base na entrada de iluminação e na configuração do material. Um *shader* descreve instruções sobre como desenhar uma superfície, incluindo se deve ou não usar alguma textura. O computador usa essas instruções para calcular cada pixel quando renderiza as imagens. O *shader* mais comum pega a cor de um material e escurece de acordo com a luz da cena, mas *shaders* podem ser utilizados para os mais diversos tipos de efeitos visuais. (CARPENTER, 2015)

Geralmente a malha geométrica de um objeto de jogo fornece apenas uma aproximação grosseira de sua forma. Os detalhes mais finos são proporcionados pelas texturas. Uma textura é apenas uma imagem bitmap padrão que é aplicada sobre a superfície da malha. Um material pode conter referências a texturas, de forma que o *shader* do material possa usar as texturas ao calcular a cor da superfície de um objeto de jogo. Além da cor básica (albedo) da superfície de um *Game Object*, as texturas podem representar muitos outros aspectos da superfície de um material, como a sua refletividade ou rugosidade. (TECHNOLOGIES, 2018)

Figura 55 Relação entre Material, Textura e *Shader*



Fonte: Próprio Autor

A Figura 55 relaciona um material, uma textura e um *shader* que foram utilizados no componente *Mesh Renderer,* do objeto usado para representar um orbital, no aplicativo de Geometria Molecular. Observe que o material funciona como um container para as propriedades visuais do objeto. A especificação de um *shader* define quais são as propriedades que este necessita para realizar a exibição do objeto, enquanto o material define os valores para essas propriedades.

Passos (2009) explica que um material define quais texturas serão usadas para a renderização, assim como quais cores serão utilizadas e os valores de outros *assets* quaisquer. Um material especifica um *shader* para utilização e o *shader* escolhido determina quais opções estão disponíveis no material. Um *shader* especifica uma ou mais variáveis para a textura usar e o Inspetor de materiais do *Unity* permite que você atribua seus próprios *assets* de textura a essas variáveis.

Um *shader* define quais propriedades ou *assets* são usados quando uma malha é renderizada na tela. *Unity* usa uma linguagem chamada ShaderLab para criar o código dos shaders. Seu estilo é similar ao da linguagem FX do Direct3D. O *Unity* já vem com mais de 30 *shaders* gratuitos. (BLOW, 2004)

Shaders definem o método de renderização de um objeto, incluindo muitas variações que dependem da placa de vídeo do computador. Por isso *Shaders* podem conter *SubShaders,* nesse caso, o motor de jogo *Unity* percorre todos os *SubShaders* e utiliza o primeiro que seja completamente suportado pelo hardware em uso. (PASSOS; JR, 2009)

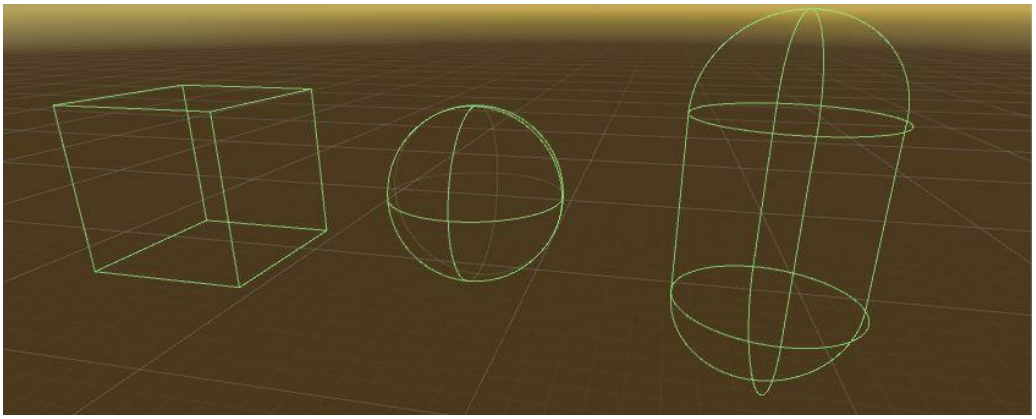
3.1.10 *Colliders* e Triggers (Colisões e Gatilhos)

A menos que o projeto esteja lidando com colisões no próprio código, os objetos da sua cena precisam ter um componente responsável por detectar colisões. A forma desses componentes não precisa corresponder a forma do objeto ao qual foram adicionados. Isso significa, por exemplo, que uma pessoa poderia ser representada como uma cápsula no sistema de colisões de um jogo. (BLOW, 2004)

Os componentes do tipo *Collider* definem a forma de um objeto para o motor de física resolver colisões. Um *Collider* é invisível para o usuário e não precisa ser exatamente da mesma forma que a malha do objeto. Na verdade, uma aproximação grosseira é muitas vezes mais eficiente e indistinguível para o usuário. (UNITY TECHNOLOGIES, 2017)

Os *Colliders* mais simples e menos intensivos em processamento são os chamados c*olliders* primitivos. Em 3D, eles são o cubo, a esfera e a cápsula, ilustrados na Figura 56, respectivamente. Em 2D, eles são o círculo e o quadrado. É possível construir *Colliders* compostos ao adicionar diversos *Colliders* primitivos a um mesmo objeto. O motor de física utilizará *colliders* para saber quando dois objetos estão se tocando, atravessando ou sobrepondo. (CREIGHTON, 2010)

Figura 56 *Colliders* 3D Primitivos



Fonte: Próprio Autor

Adicione *colliders* a um objeto sem um componente *Rigidbody* para criar pisos, paredes e outros elementos imóveis em uma cena. Estes são referidos como *colliders* estáticos. Em geral, você não deve reposicionar *colliders* estáticos mudando a posição do *Transform* deles, pois isso afetará fortemente o desempenho do motor de física. *Colliders* em objetos que possuem um *Rigidbody* são conhecidos como *colliders* dinâmicos. Os *colliders* estáticos podem interagir com *colliders* dinâmicos, mas como eles não têm um *Rigidbody*, eles não se moverão em resposta a colisões. (UNITY MANUAL, 2017)

Ambos os aplicativos usam o motor de física durante a interação do usuário com os objetos selecionáveis em cena. Isso será explicado em detalhes no capítulo seguinte.

3.1.11 *COROUTINES (CO-ROTINAS)*

Co-rotinas são um jeito específico do *Unity* lidar com tarefas que executam de forma incremental ao longo do tempo, ao contrário da maioria das funções que fazem o programa esperar até elas finalizarem sua execução. Quando você chama uma função, ela é executada até a conclusão antes de retornar. Isso efetivamente significa que qualquer ação que ocorra em uma função deve ocorrer dentro de uma única atualização de quadro. Uma chamada de função não pode ser usada para conter uma animação procedural ou uma sequência de eventos ao longo do tempo. Como

exemplo, considere a tarefa de reduzir gradualmente o valor do canal alfa (opacidade) de um objeto até ficar completamente invisível, ilustrado na Figura 57. (CARPENTER, 2015)

Figura 57 Código C# para Alterar Opacidade Gradualmente

```
void Fade() {
    for (float f = 1f; f >= 0; f -= 0.1f) {
        Color c = renderer.material.color;
        c.a = f;
        renderer.material.color = c;
    }
}
```

Fonte: (UNITY MANUAL, 2017)

O código da função *Fade,* na imagem acima, não terá o efeito esperado. Para que o *fading* (desaparecimento gradual) seja visível, o alfa deve ser reduzido em uma sequência de quadros (frames), ao longo do tempo, para mostrar os valores intermediários que estão sendo *renderizados*. No entanto, a função será executada em sua totalidade dentro de uma única atualização de quadro. Os valores intermediários nunca serão vistos e o objeto desaparecerá instantaneamente. É possível lidar com situações como esta, adicionando código à função *Update* que executa o fading em frame-a-frame. No entanto, muitas vezes é mais conveniente usar uma co-rotina para esse tipo de tarefa. (UNITY MANUAL, 2017)

Uma co-rotina é similar a uma função, mas tem a capacidade de pausar sua execução e retornar o controle para o *Unity.* Depois ela é capaz de continuar de onde ela deixou no frame anterior. É uma técnica avançada, onde a execução de um método pode ser interrompida e resumida em um determinado ponto. O comando *yield* (suspender, em inglês) interrompe temporariamente a execução do código no método, permitindo que *Unity* possa executar código de outros objetos de jogo enquanto o motor de física continua processando as simulações de física. (SMITH; QUEIROZ, 2015)

É essencialmente uma função declarada com um tipo de retorno do tipo IEnumerator e com a declaração de retorno *yield* incluída em algum lugar do corpo do método. A linha de retorno *yield* é o ponto em que a execução irá pausar e retomar no quadro a seguir. Para colocar uma co-rotina em execução, é preciso também usar a função *StartCoroutine*. Por padrão, uma co-rotina é retomada no quadro depois que ela é suspendida, mas também é possível introduzir uma demora por tempo usando *WaitForSeconds*, como na Figura 58. (TECHNOLOGIES, 2018)

Figura 58 Exemplo de Implementação de uma Coroutine

```
IEnumerator Fade() {
    for (float f = 1f; f >= 0; f -= 0.1f) {
        Color c = renderer.material.color;
        c.a = f;
        renderer.material.color = c;
        yield return new WaitForSeconds(.1f);
    }
}
```

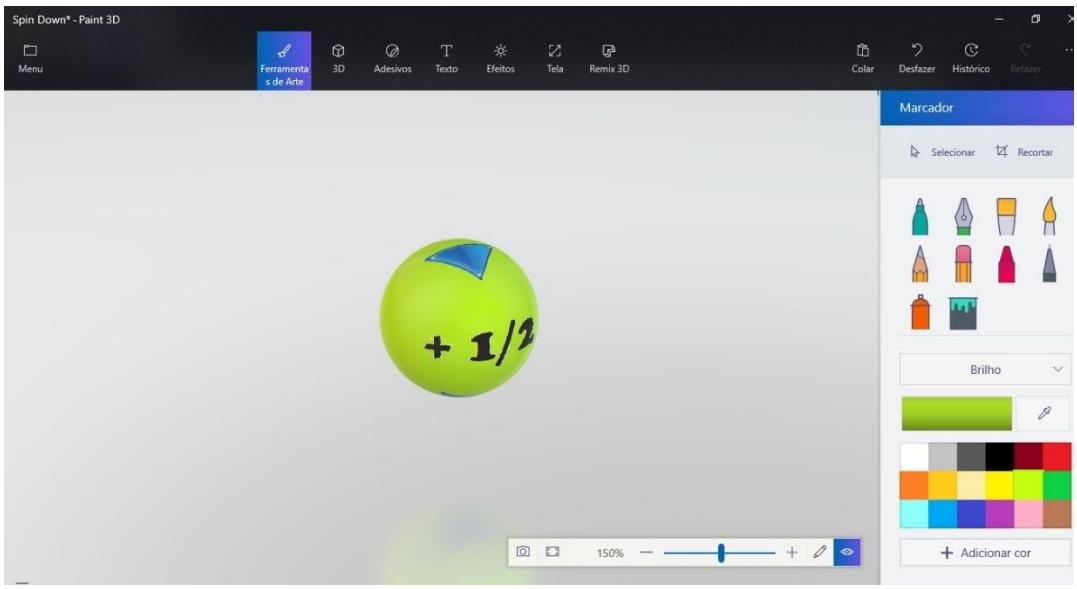
Fonte: (UNITY MANUAL, 2017)

Co-rotinas foram utilizadas nos aplicativos sempre que uma mensagem de texto precisa aparecer na tela por um determinado intervalo de tempo, ou quando um objeto precisa mudar de cor e retornar a cor original automaticamente após alguns segundos. Por exemplo, no aplicativo Configuração Eletrônica, sempre que o usuário faz uma escolha errada, o objeto onde ele clicou fica vermelho, por alguns segundos, antes de retornar a cor original. Uma mensagem de texto na tela o informa o motivo da escolha estar errada, ela some depois de alguns segundos por estar implementada em uma co-rotina. Mesmo ao fazer uma escolha correta, por exemplo quando o usuário seleciona o próximo subnível no diagrama de Pauling, uma co-rotina é iniciada para esconder o diagrama após um segundo e mostrar uma mensagem informando ao usuário que ele fez a escolha correta.

3.2 – Paint 3D

O *Paint 3D* é uma ferramenta gratuita da Microsoft para modelagem 3D. Ela é simples e consequentemente limitada, a curva de aprendizado é bem inferior em relação a outras ferramentas de modelagem 3D disponíveis no mercado. Como os objetos utilizados nos aplicativos possuem formas simples, essa ferramenta apresentou o melhor custo benefício, em relação ao tempo de desenvolvimento, para as soluções envolvidas no projeto. A Figura 59 mostra o resultado da construção de um objeto que foi utilizado para representar um tipo de spin de um elétron. Esse modelo foi exportado, posteriormente, para o motor de jogo *Unity*. (MICROSOFT, 2015)

Figura 59 Modelagem no *Paint 3D*

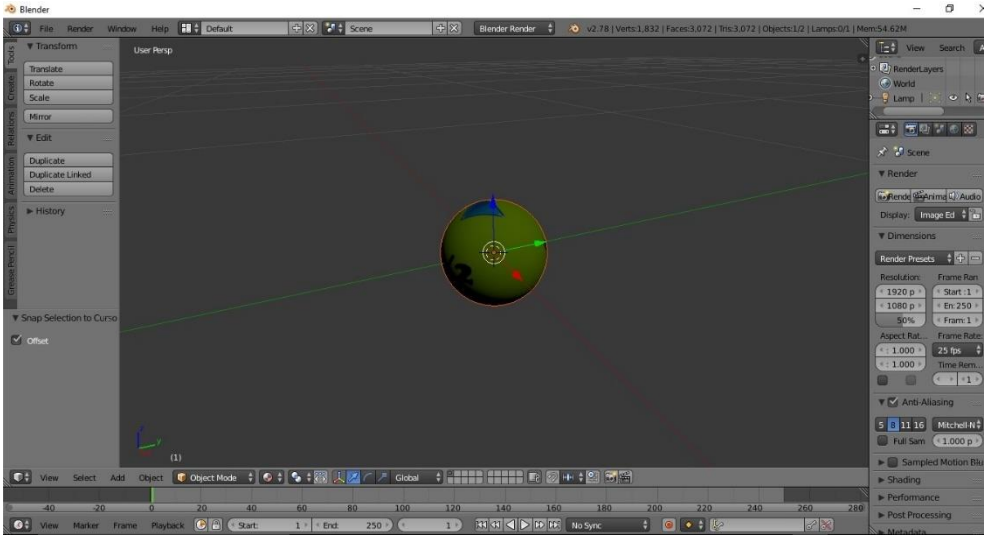


Fonte: Próprio Autor

3.3 – Blender

Blender é um suíte de criação de conteúdo 3D gratuita e de código aberto. Ela fornece ferramentas para trabalhar com modelagem, animação, simulação, *rendering*, edição de vídeo, criação de jogos, entre outras. É multiplataforma, o que garante que ela execute igualmente em computadores Linux, Windows e Mac. Sua interface usa a biblioteca OpenGL. (BLENDER, 2019)

Figura 60 Ajustes para Exportação no Blender



Fonte: Próprio Autor

Nesse trabalho, Blender foi utilizado para fazer pequenas alterações antes de importar os modelos no *Unity*, ela não precisaria ser utilizada se o *Paint 3D* não fosse tão limitado. Os modelos, algumas vezes, precisavam ter sua rotação alterada para seguir as convenções utilizadas no *Unity*, exibidas na Tabela 1, ou suas posições alteradas para que os modelos fossem deslocados para o centro do ambiente virtual (x, y e z com valor 0).

Tabela 1 Convenções em Direções no *Unity*

Direção *(Unity)*	**Vetor (x, y, z)**
Frente	Vetor (0, 0, 1)
Trás	Vetor (0, 0, -1)
Direita	Vetor (1, 0, 0)
Esquerda	Vetor (-1, 0, 0)
Cima	Vetor (0, 1, 0)
Baixo	Vetor (0, -1, 0)

Fonte: (TECHNOLOGIES UNITY, 2018)

3.4 – Git

O Git é uma ferramenta de controle de versão, um sistema que registra mudanças realizadas em um conjunto de arquivos ao longo do tempo, de forma que seja possível recuperar versões específicas facilmente. Esse tipo de recurso ajuda a encontrar modificações que possam ter causado problemas, quando um bug foi introduzido e restaurar arquivos perdidos. (DOUGLAS, 2014)

Essa ferramenta serve como apoio aos processos de engenharia de software em geral. Git é uma ferramenta de controle de versão distribuído, isso significa que existe um repositório local em uma máquina cliente, interagindo com um repositório externo em um servidor. Repositório é o local onde o Git armazena uma série de *snapshots* (captura de um arquivo em um determinado instante, como uma foto) e referências para essas capturas. (CHACON; STRAUB, 2014)

Ambos os projetos de software foram desenvolvidos com ferramentas de controle de versão desde o início, o servidor utilizado foi o *Github*. O uso dessa ferramenta ajuda na administração de atualizações desses projetos ao longo do tempo.

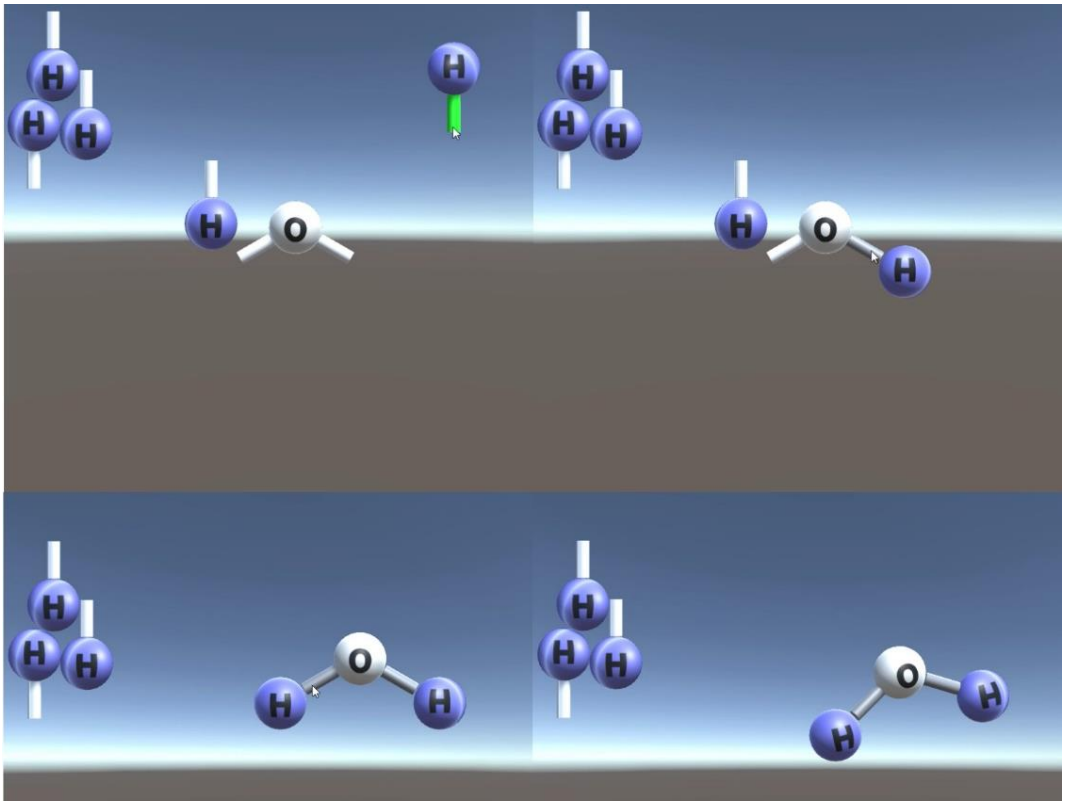
3.5 – Modelo de Processo

O modelo de processo utilizado inicialmente foi o da prototipação, um modelo tradicional evolucionário de engenharia de software. Esse modelo foi escolhido porque durante a fase de análise de requisitos foram definidos objetivos gerais, sem detalhes de funções e recursos. Pressman (2016) sugere a elaboração de um protótipo como primeira etapa quando o cliente não tem ideia com relação aos detalhes e o produto cresce e muda, ou seja, não há uma linha reta bem definida a ser seguida no início do projeto.

Os primeiros protótipos dos aplicativos foram descartados, tendo servido apenas para que os envolvidos compreendessem melhor o que está para ser construído e esquematizassem quais área precisariam de uma definição mais ampla.

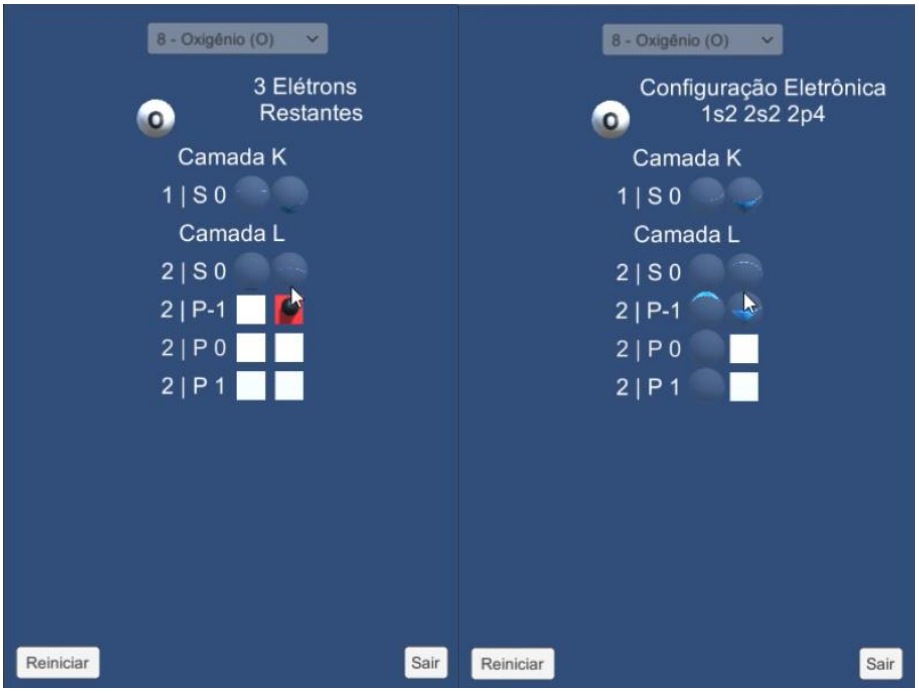
A Figura 61 mostra quatro imagens do primeiro protótipo do aplicativo de Geometria Molecular, e a Figura 62 duas imagens do aplicativo de configuração eletrônica. Além de requisitos que foram mal interpretados, gerando funcionalidade incorretas, o primeiro *feedback* recebido foi uma sugestão de melhoria na interação do usuário com o software. Os primeiros protótipos foram desenvolvidos com interação via mouse e uma interface voltada para toque na tela (*TouchScreen*) requeria diversos ajustes nos posicionamentos de todos os elementos.

Figura 61 Protótipo Descartado do Aplicativo de Geometria Molecular



Fonte: Próprio Autor

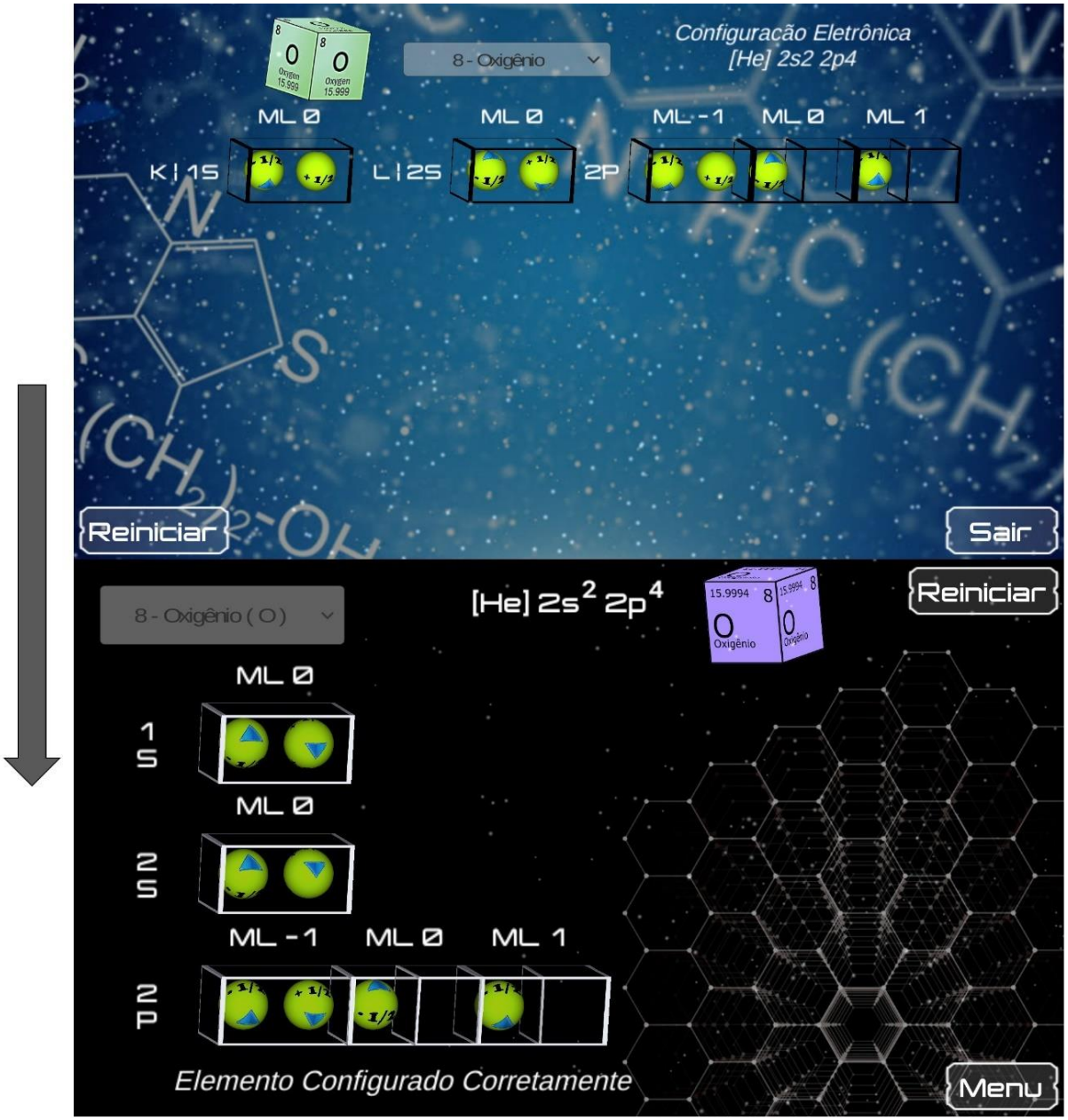
Figura 62 Protótipo Descartado do Aplicativo de Configuração Eletrônica



Fonte: Próprio Autor

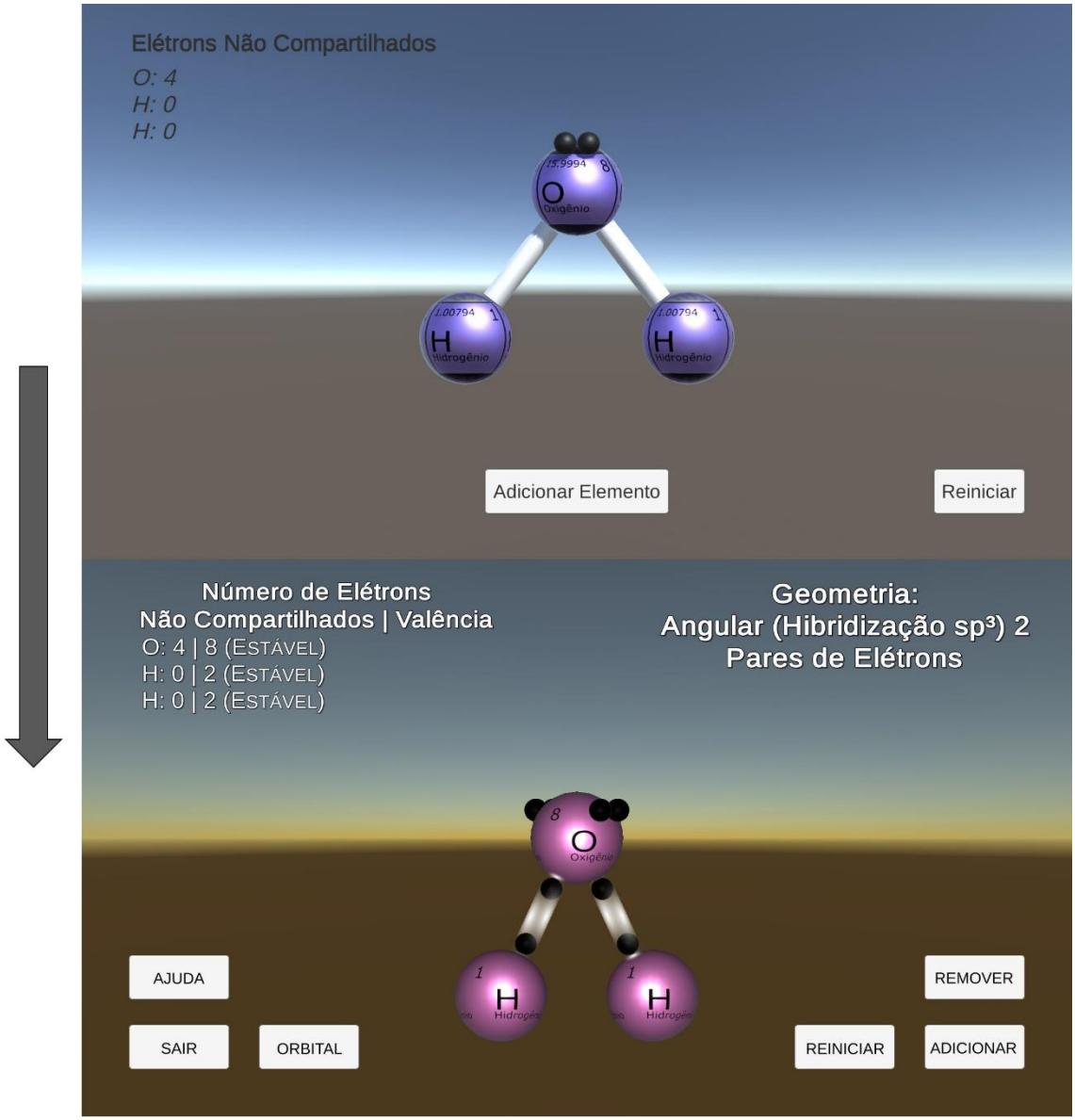
Esses protótipos foram descartados, mas os próximos foram elaborados com o objetivo de evoluírem até o produto final. A evolução ocorreu por incrementos, seguindo o modelo de processo incremental. Em outras palavras, versões cada vez mais completas do software foram produzidas a cada iteração. A Figura 63 e Figura 64 mostram parte da evolução dos protótipos seguintes. A parte que não aparece nas figuras é referente as novas funcionalidades que não são exibidas nas imagens, como a visualização dos orbitais no aplicativo de Geometria Molecular e o diagrama de Pauling no aplicativo de Configuração Eletrônica.

Figura 63 Evolução do Segundo Protótipo de Configuração Eletrônica



A seta aponta a direção da evolução. Fonte: Próprio Autor

Figura 64 Evolução do Segundo Protótipo de Geometria Molecular



A seta aponta a direção da evolução. Fonte: Próprio Autor

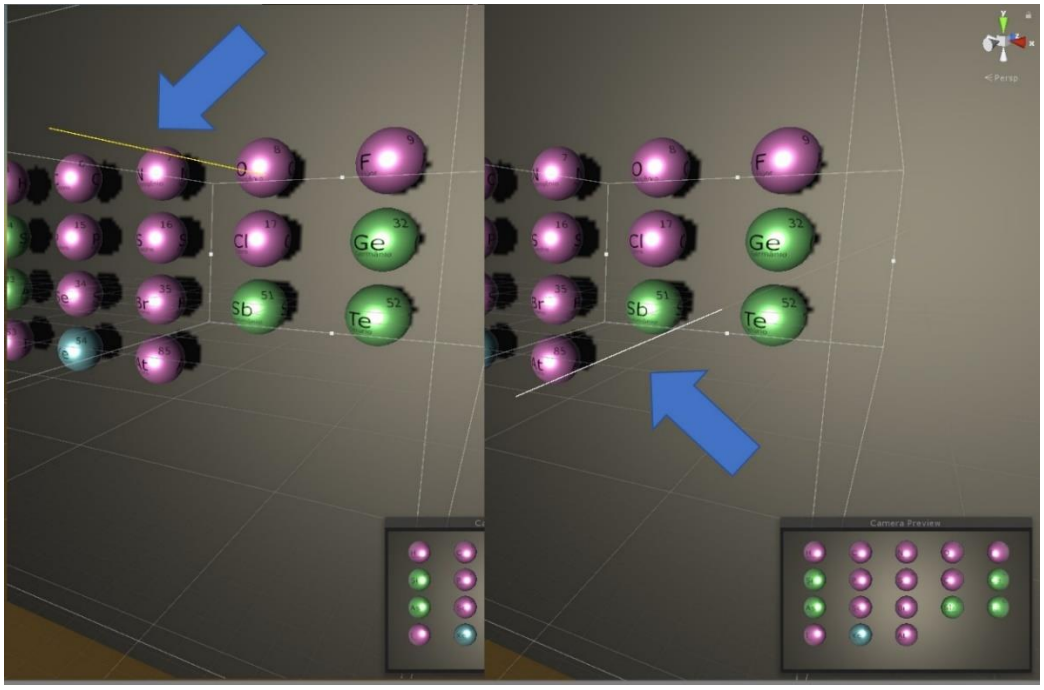
3.6 – Aspectos Comuns aos Aplicativos

Antes de falar do desenvolvimento de cada aplicativo individualmente, esse subcapítulo apresenta escolhas de *design* de software que foram implementadas em ambos os aplicativos desse trabalho.

O usuário interage com os aplicativos através de toques. Cada toque na tela projeta uma linha *Raycast,* que retorna todos componentes de *colliders* que interceptar. A função *Raycast* tem como origem um ponto no ambiente tridimensional, obtido através da conversão de um ponto na tela sensível ao toque, onde o usuário tenha tocado. É preciso converter as coordenadas em duas dimensões para um ponto no espaço tridimensional. O resultado é um ponto contido no plano Y, que intercepta a câmera na cena. A partir desse ponto, é projetado um *Raycast,* em formato de uma reta perpendicular ao plano Y. Sua direção é “para frente”, que por convenção no *Unity,* significa em direção ao eixo Z positivo.

A Figura 65 mostra dois exemplos de projeções de *Raycast,* no aplicativo de Geometria Molecular. Uma projeção intercepta um *collider* (parte esquerda na imagem), enquanto a outra não (parte direita na imagem). Essas projeções são invisíveis para os usuários, funcionando apenas como mecanismo para verificação durante o desenvolvimento.

Figura 65 Projeção do *Raycast*



As setas apontam para os *Raycasts*. Fonte: Próprio Autor

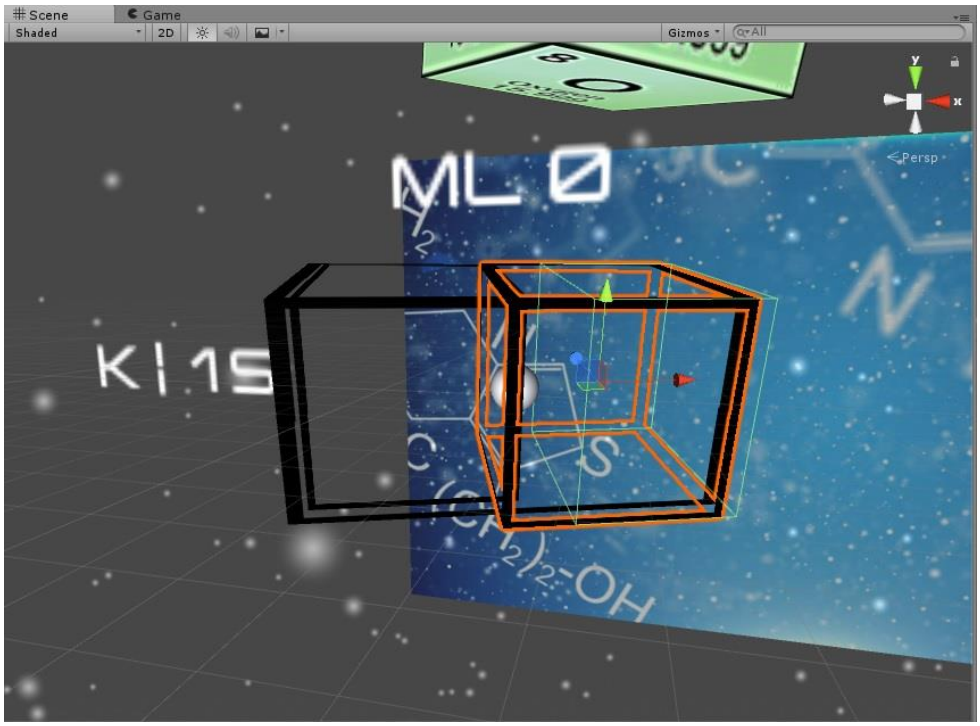
O *Raycast* também recebe como parâmetro um valor para a distância da projeção a partir do ponto de origem e uma camada (*layer*). A distância usada foi “infinito”,

através da função de classe *Mathf.Infinity*, mas um valor de 20 unidades também seria suficiente, uma vez que essa é a distância máxima de visualização de profundidade, configurada na câmera na cena.

Por fim, criou-se uma camada para os objetos interceptáveis pelo *Raycast.* Essa camada é passada para a função como uma máscara de bits. Isso é importante para separar os objetos os quais o usuário pode interagir na cena.

Antes da implementação com uma linha, o motor de física era utilizado para detectar as interações do usuário usando o *collider* primitivo de uma esfera. O usuário clicava na tela e as coordenadas serviam para posicionar uma esfera na cena. Eventos eram gerados quando o *collider* da esfera ficava sobreposto com o *collider* de algum objeto interativo. A Figura 66, mostra a esfera, apenas para fins de testes, ela é invisível para o usuário. Observe que o *collider* do cubo não corresponde ao formatado visível do objeto, dessa forma o usuário não conseguia clicar em dois cubos ao mesmo tempo.

Figura 66 Interação Através de Eventos de Gatilho de *Colliders*



Um toque entre os dois cubos não gera uma colisão. Fonte: Próprio Autor

As coordenadas x, y dos toques na tela foram convertidas para coordenadas tridimensionais x_{3D}, y_{3D} na cena, através das seguintes fórmulas;

$$x_{3D} = x_{min} + x_{total} * \frac{T_x}{l}$$

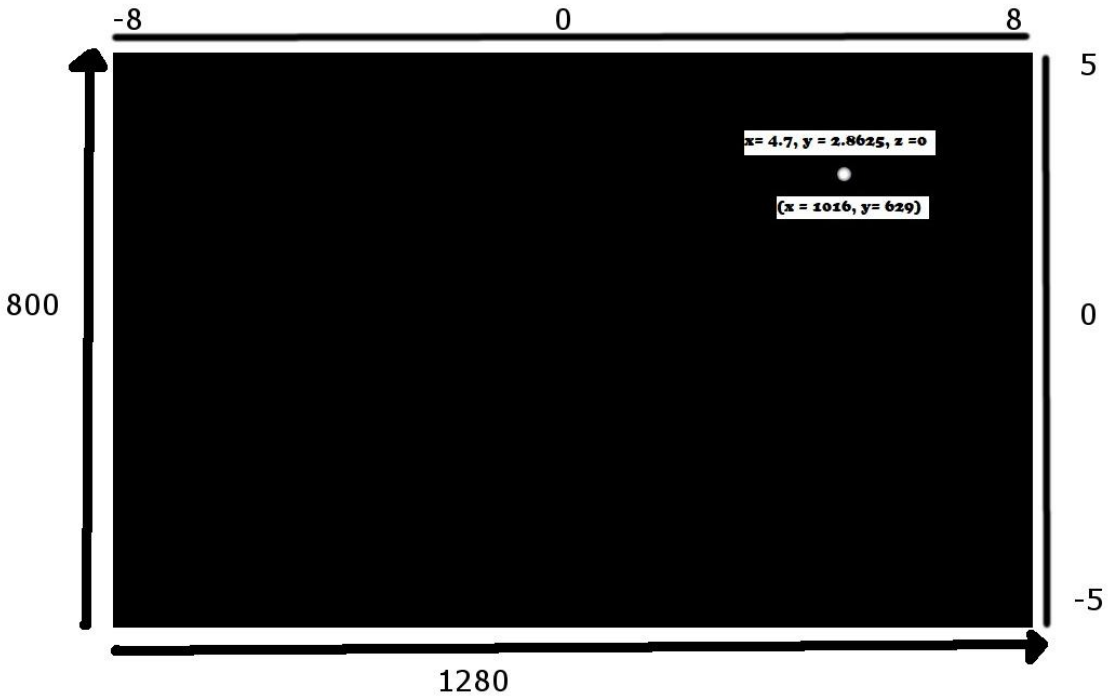
$$y_{3D} = y_{min} + y_{total} * \frac{T_y}{a}$$

Onde, dentro espaço visível da câmera na cena, em unidade do *Unity*:

- x_{min} e y_{min} equivalem aos menores valores que x e y podem ter;
- x_{total} e y_{total} ao tamanho total, da largura e altura respectivamente;
- T_x e T_y são as coordenadas x e y, de um toque detectado na tela;
- l e a são os valores da resolução (largura x altura) da tela;

A Figura 67 mostra um exemplo de conversão: Os valores na esquerda e abaixo da imagem são as coordenadas da tela (resolução). Os valores acima e na direita da imagem são as coordenadas da cena.

Figura 67 Conversão de Coordenadas de um Toque na Tela



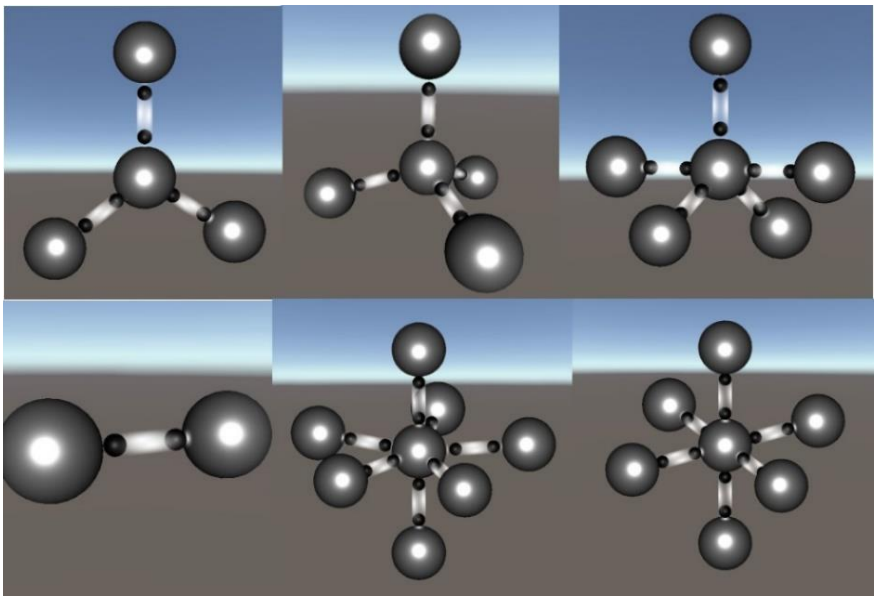
Exemplo de conversão para uma tela com resolução 1280x800 pixels. Fonte: Próprio Autor

Houve muitas reclamações com relação a imprecisão nos toques, o que levou a diversos ajustes finos nos *colliders* de cada objeto e no tamanho da esfera, entre outros elementos. Problemas que foram resolvidos ao se usar a abordagem da projeção de uma reta, perpendicular ao plano Y, que corta a câmera na cena.

3.7 – Desenvolvimento do Aplicativo Geometria Molecular

O núcleo principal desse aplicativo consiste em *prefabs* que representam as geometrias moleculares. Cada *prefab* tem um componente que o permite receber, e posicionar corretamente, uma lista de átomos. O posicionamento funciona por substituição, de cada átomo recebido, em uma das esferas do *prefab*. Cada esfera é um objeto de jogo com uma *tag*, que o identifica como objeto marcador de posição (*placeholder*, em inglês). A classe Geometria tem um comportamento que permite que, durante sua inicialização, ela percorra sua hierarquia, procurando por *placeholders* e armazenando as posições de seus componentes do tipo *Transform*. As geometrias são organizadas de acordo com a quantidade de átomos. A Figura 68 exibe alguns *prefabs* que são usados como geometria no aplicativo.

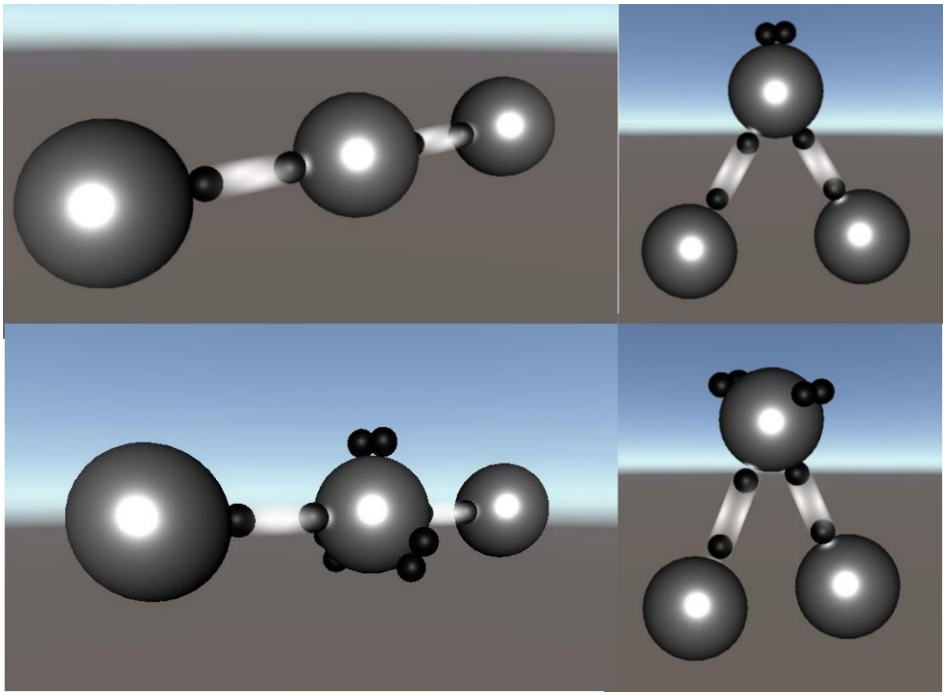
Figura 68 *Prefabs* de algumas geometrias



As esferas funcionam como marcadores de posições. Fonte: Próprio Autor

As geometrias moleculares não são descritas apenas em função da quantidade átomos ao redor do elemento central. Também é preciso levar em consideração o número de pares de elétrons não compartilhados. Por isso, cada geometria tem variações de acordo com esse fator. A Figura 69 ilustra as possibilidades de variações da geometria de três átomos, que é diferente se o átomo central tem um, dois ou três pares de elétrons não compartilhados. Cada geometria foi preparada para possuir diferentes versões de acordo com essa variável.

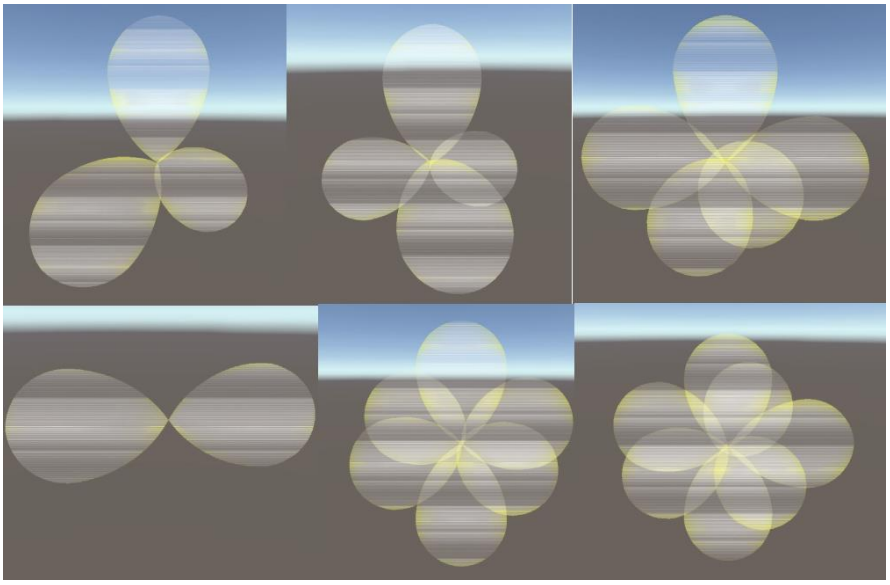
Figura 69 Variações na geometria de 3 átomos



Fonte: Próprio Autor

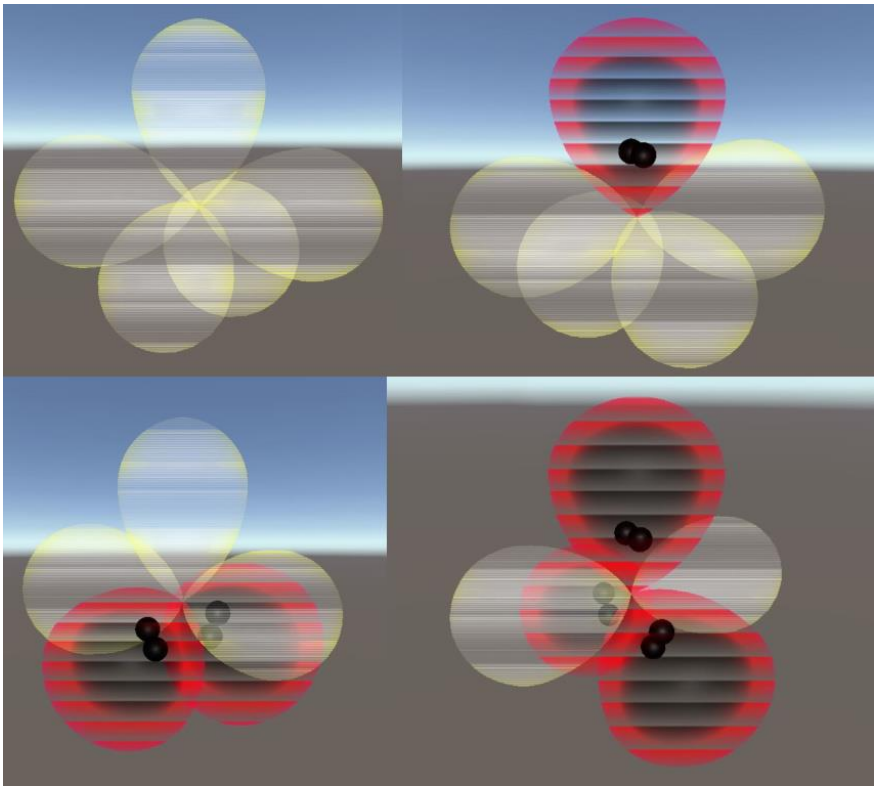
Cada geometria está relacionada a *prefabs* que representam a hibridização dos orbitais do átomo central. A Figura 70 mostra os orbitais que estão relacionados com as geometrias na Figura 68. Da mesma forma que as geometrias, esses objetos precisam responder de forma diferente de acordo com o número de elétrons não compartilhados no elemento central. A Figura 71 exibe como uma mesma hibridização é desenhada na tela de forma diferente, de acordo com os pares não compartilhados. Cada orbital híbrido foi implementado para permitir mudanças de acordo com esse fator.

Figura 70 *Prefabs* de alguns orbitais híbridos



Fonte: Próprio Autor

Figura 71 Variações na representação visual de uma mesma hibridização



O objeto de jogo precisa ser desenhado na tela de forma diferente, de acordo com a quantidade de pares de elétrons não compartilhados no elemento central. Fonte: Próprio Autor

As classes do aplicativo foram segmentadas de forma a respeitar o primeiro princípio de *design* orientado a objetos: O *Single Responsability Principle* (SRP), esse princípio sugere que cada classe em uma aplicação tenha uma, e apenas uma, responsabilidade. A Tabela 2 mostra a única responsabilidade das classes mais importantes do aplicativo Geometria Molecular. (MARTIN, 2009)

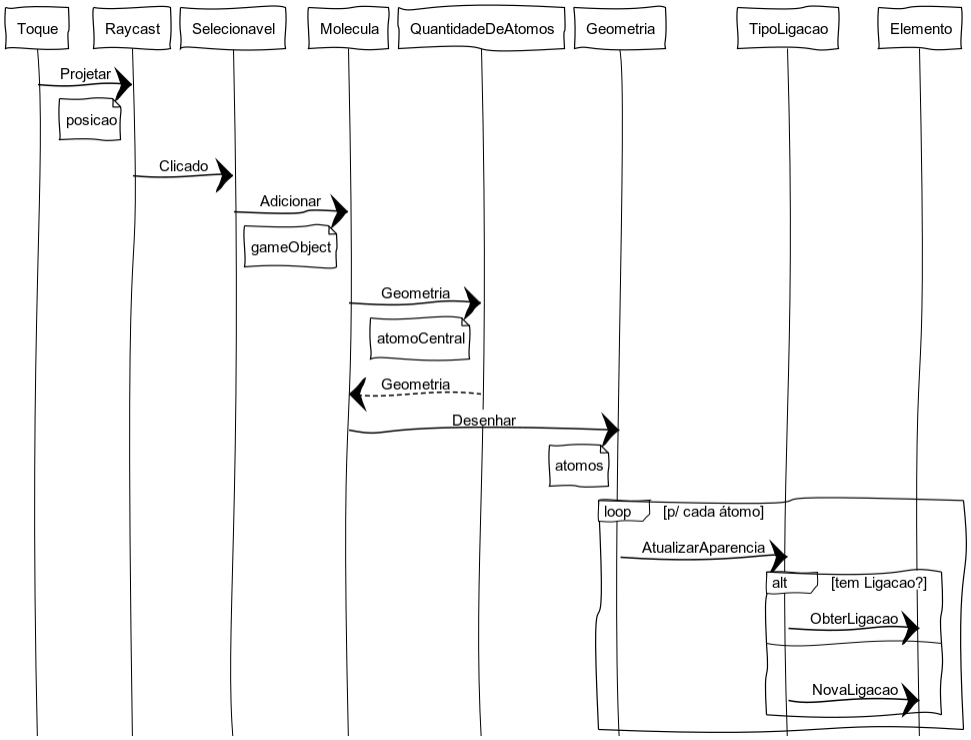
Tabela 2 Responsabilidade das Classes em Geometria Molecular

Classe	Responsabilidade
Molecula	Administrar a lista de átomos selecionados
Geometria	Desenhar a geometria atual na cena
TipoLigacao	Desenhar o tipo de ligação entre dois átomos
Elemento	Armazenar informações sobre o elemento
QuantidadeAtomos	Decidir qual deve ser a geometria atual
Selecionavel	Permitir que um objeto possa ser selecionado
Raycast	Permitir que o usuário possa interagir com objetos
Toque	Administrar comandos de entrada (*Input*) do *Touch Screen*
Mouse	Administrar comandos de entrada (*Input*) do mouse
Menu	Gerenciar todos os botões da Interface de Usuário (UI)
Escolher	Desenhar os objetos do menu de escolha de átomos
Mensagem	Permitir que mensagens de alerta sejam exibidas
Rotacionar	Rotacionar os objetos de jogo (*Game Object*) na cena

Fonte: Próprio Autor

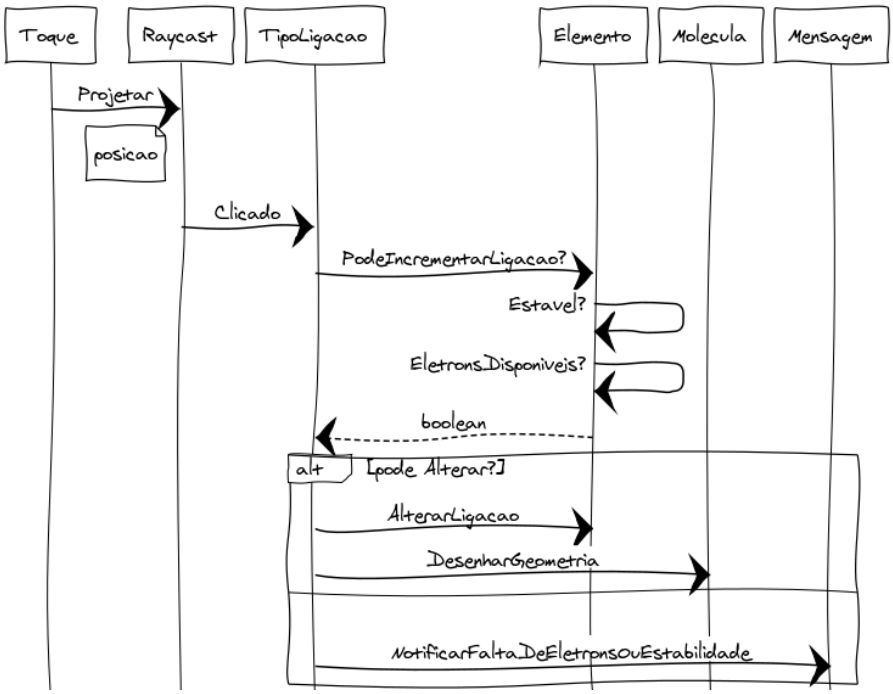
A Figura 72 mostra um diagrama de sequência resumido do cenário onde um usuário adiciona um átomo. O diagrama permite visualizar a troca de mensagens e ajudar a enxergar o acoplamento que existe entre as classes. Os parâmetros enviados foram organizados em notas, abaixo das mensagens, ao invés de estarem entre parêntesis como sugere a UML, para evitar que a figura fique muito larga. A Figura 73 exibe outro diagrama de sequência de interação do usuário, dessa vez clicando em um tipo de ligação, para alterá-la entre os tipos: simples, dupla e tripla. Para finalizar, a Figura 74 apresenta o diagrama de classes simplificado do sistema, com as associações, atributos e métodos mais importante dos principais componentes do sistema.

Figura 72 Diagrama de Sequência de Adição de Átomo



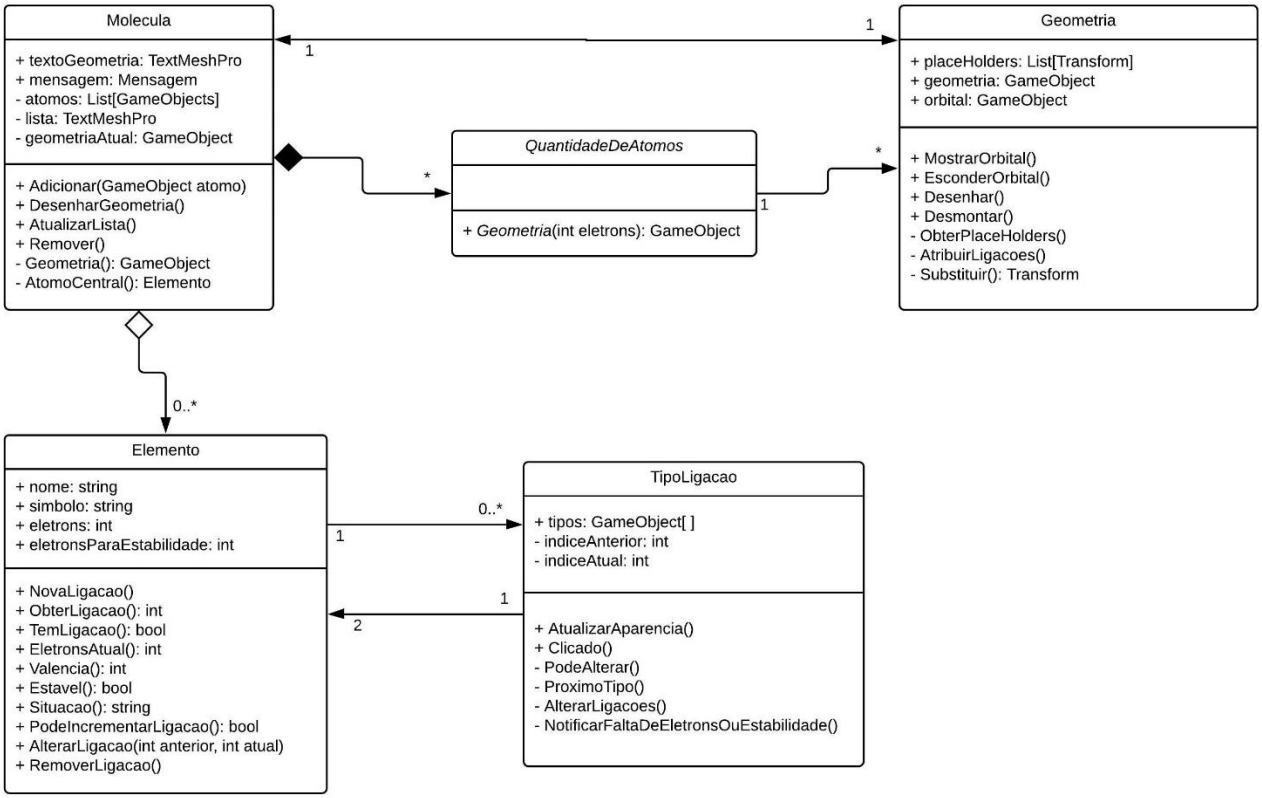
As notas abaixo das mensagens são os parâmetros enviados. Fonte: Próprio Autor

Figura 73 Diagrama de Sequência de Alteração de Ligação



A mensagem DesenharGeometria envolve a mesma sequência da Figura 72. Fonte: Próprio Autor

Figura 74 Diagrama de Classes do Aplicativo Geometria Molecular



Principais classes e seus atributos, métodos e associações. Fonte: Próprio Autor

Cada correção de *bug*, ou implementação de *feature*, envolvia um processo de testes usando as moléculas da Tabela 3. Todas as moléculas da tabela eram construídas sequencialmente usando as ligações descritas na terceira coluna, com a finalidade de se confirmar que a geometria apresentada pelo software estava de acordo com a descrita na coluna do meio. A intenção desses testes era confirmar que a alteração no sistema não fez com que nenhum componente anterior parasse de funcionar corretamente, por causa de algum efeito colateral não previsto. Em outras palavras, as moléculas da tabela serviram para fazer testes de regressão no sistema, esses testes servem para certificar que não existem erros de regressão no sistema a cada nova versão. Isso ajuda a evitar que novas versões apareçam com erros que já foram corrigidos em versões anteriores. A tabela contém moléculas que testam todas geometrias disponíveis no sistema, algumas resultam em uma mesma geometria, mas usam diferentes tipos de ligações (simples, dupla ou tripla). Por exemplo, a molécula

HCN, que é linear e contém uma ligação tripla, ou a molécula CO2, que também é linear, mas contém ligações duplas.

Tabela 3 Lista de Elementos Usados em Testes de Cada Nova Versão

Molécula	Geometria	Ligações
CO2	Linear	Duas Ligações Duplas
CH2O	Trigonal plana	Duas Ligações Simples (H) e Uma Dupla (O)
SO2	Angular com 1 Par de Elétrons Não compartilhado	Duas Ligações Duplas
CH4	Tetraédrica	Apenas Ligações Simples
NH3	Trigonal piramidal	Apenas Ligações Simples
H2O	Angular com 2 Pares de Elétrons Não Compartilhados	Apenas Ligações Simples
PCl5	Bipiramidal trigonal	Apenas Ligações Simples
ClF3	Forma de T	Apenas Ligações Simples
SF4	Gangorra	Apenas Ligações Simples
SF6	Octaédrica	Apenas Ligações Simples
IF5	Bipiramidal pentagonal	Apenas Ligações Simples
XeF2	Linear com 3 Pares de Elétrons Não compartilhados	Apenas Ligações Simples
XeF4	Quadrada plana	Apenas Ligações Simples
PF3	Trigonal piramidal	Apenas Ligações Simples
OF2	Angular com 2 Pares de Elétrons Não compartilhados	Apenas Ligações Simples
SbH3	Trigonal piramidal	Apenas Ligações Simples
BrF5	Piramidal quadrada	Apenas Ligações Simples
HCN	Linear	Uma Ligação Simples (H) e Uma Ligação Tripla (N)

Fonte: Adaptado de (DEWITT, 2012b) e (DEWITT, 2012a)

3.8 – Desenvolvimento do Aplicativo Configuração Eletrônica

O aplicativo Configuração Eletrônica contém três cenas. Uma cena funciona apenas como um menu de opções, que permite ao usuário navegar entre as cenas ou sair do aplicativo. Ele também contém referências e créditos aos criadores dos arquivos de áudio e imagens que foram usados na construção desse *software*. Uma segunda cena funciona apenas como referencial teórico, caso o usuário deseje estudar distribuição eletrônica antes de usar o programa.

A cena principal do aplicativo consiste em três partes: Seleção de um elemento para configuração, distribuição dos elétrons desse elemento em camadas e o menu de escolha da camada seguinte. Antes de explicar cada uma delas, é importante conhecer as principais classes do sistema e suas responsabilidades, conforme informa a Tabela 4.

Tabela 4 Responsabilidade das Classes em Configuração Eletrônica

Classe	Responsabilidade
Distribuição	Armazenar as respostas do usuário (configuração eletrônica)
ElementoSelecionado	Atualizar o sistema com informações sobre o elemento atual
Exceção	Guardar configurações eletrônicas que são exceções à regra
Elétron	Responder qual é o destino e spin correto do elétron atual
Elemento	Armazenar informações sobre cada elemento
Camadas	Atualizar o sistema conforme as camadas são preenchidas
Subnível	Armazenar informações sobre cada subnível
Espaço	Fornecer uma interface de interação com subnível
SeletorDeCamadas	Fornecer uma interface de interação com as camadas

Fonte: Próprio Autor

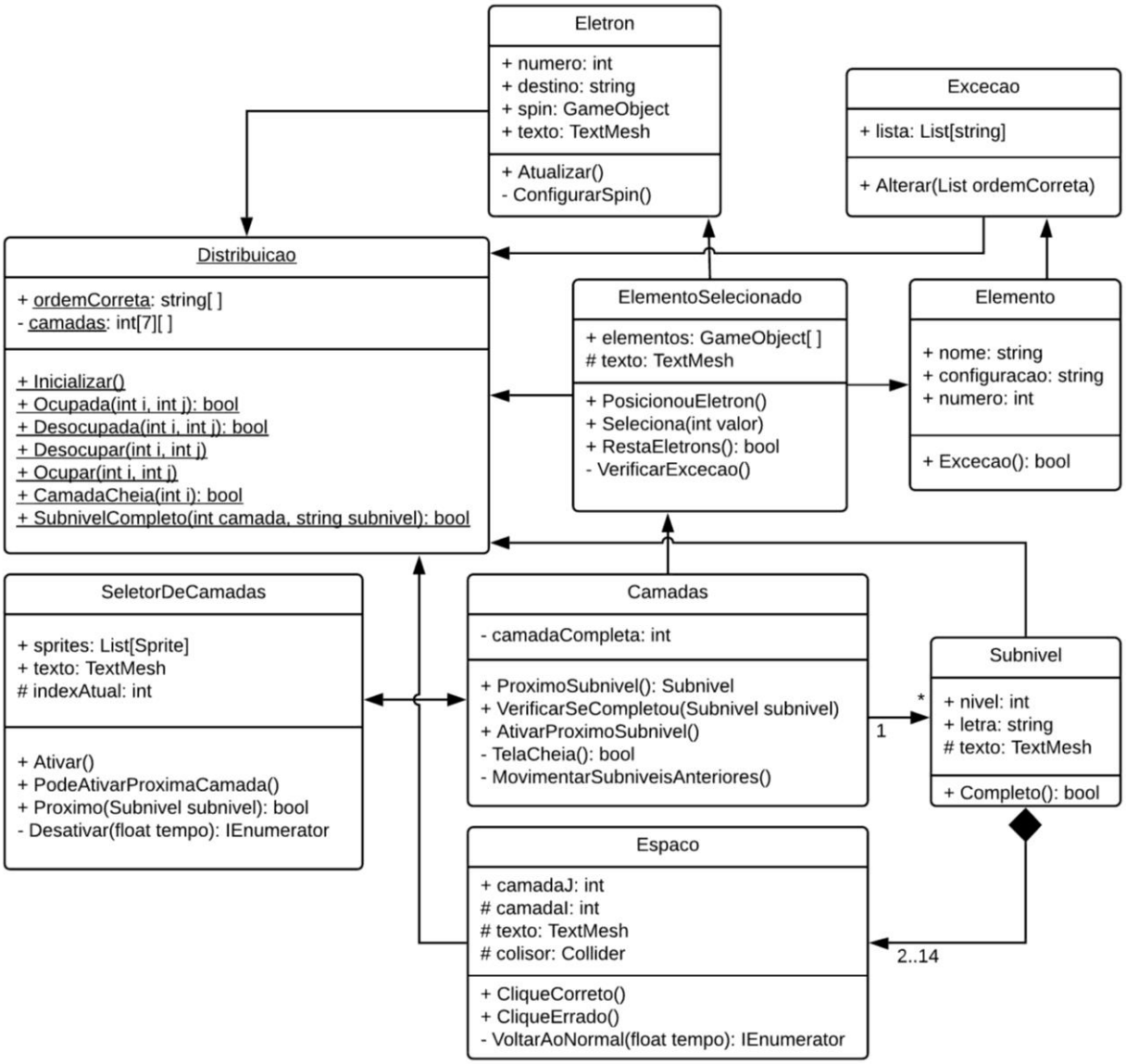
A maior parte do sistema interage com os métodos de classe de Distribuição, em outras palavras, não são criadas instâncias dessa classe. Caso fossem, seria preciso aplicar o padrão de projeto *Singleton*, o qual garante que exista uma, e apenas uma, instância dessa classe no sistema ao mesmo tempo e fornece um ponto de acesso global a ela. (GAMMA et al., 2000)

A classe Distribuição guarda as respostas do usuário em uma estrutura de dados do tipo matriz de inteiros. Existe uma linha na matriz para cada camada, a qual é preenchida com zeros e uns, representando uma camada não preenchida e preenchida, respectivamente. Essa classe também contém uma lista ordenada com *strings* que representam a ordem de distribuição padrão.

A Figura 75 mostra o diagrama de classes do sistema. Nela é possível observar quais partes do sistema interagem diretamente com a classe Distribuição: Elétron, Exceção, Elemento Selecionado, Subnível e Espaço.

A classe Elétron interage com Distribuição para saber qual é a posição correta do o elétron sendo distribuído atualmente. A classe Exceção contém listas de configurações diferentes para cada elemento que contém uma distribuição eletrônica que foge do padrão. Ela interage com Distribuição para atualizar a lista que contém a ordem correta das respostas que o usuário precisa fornecer. Subnível pergunta para Distribuição se está completo, sempre que o usuário posiciona um elétron em um espaço, o qual notifica Distribuição de seu preenchimento.

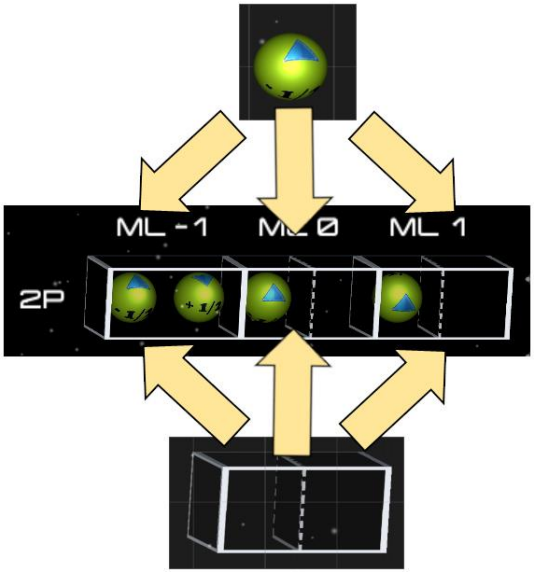
Figura 75 Diagrama de Classes do Aplicativo Configuração Eletrônica



As associações sem multiplicidade são associações 1 para 1. Fonte: Próprio Autor

No diagrama acima, é importante observar que uma camada contém vários subníveis e um subnível é composto por uma quantidade de espaços, que são preenchidos por elétrons quando o usuário faz um clique no local correto. Todos os subníveis são *prefabs* construídos a partir de *prefabs* menores que são repetidos, conforme exibido na Figura 76.

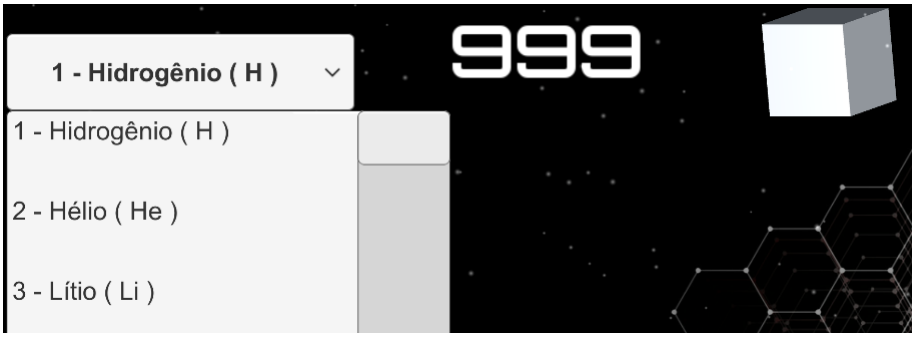
Figura 76 Exemplo de um *Prefab* de Subnível



Subnível é criado a partir de *prefabs* menores. Fonte: Próprio Autor

A primeira parte do sistema (Figura 77) é composta por três objetos principais: Um menu de seleção do tipo *Dropdown,* que contém 108 elementos ordenados por número atômico. Um texto central que exibe o número de elétrons que falta distribuir, esse valor é atualizado quando o usuário seleciona um elemento e mostra a configuração eletrônica em formato de texto, quando o usuário finaliza a distribuição. O último elemento funciona apenas como marcador de posição para mostrar um cubo com uma imagem do elemento selecionado, extraída de uma tabela periódica.

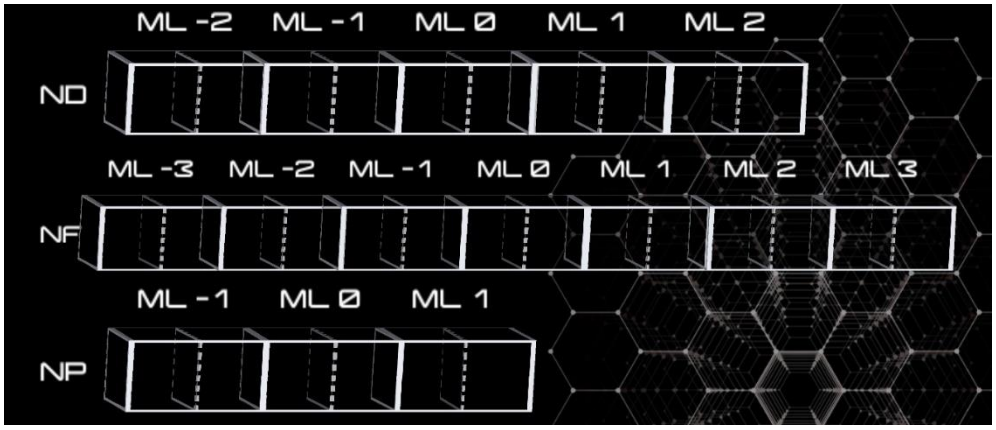
Figura 77 Objetos Relacionados à Seleção de Elementos



Fonte: Próprio Autor

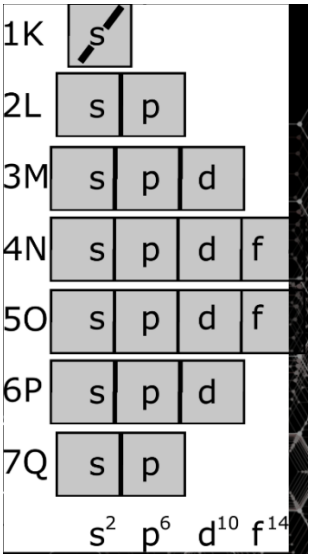
As outras duas partes do sistema se relacionam constantemente. O sistema de preenchimento de camadas (Figura 78) é interrompido sempre que o usuário completa a distribuição de um subnível. Ele aguarda a escolha do subnível correto no menu de escolhas do subnível seguinte (Figura 79), o qual devolve o controle ao sistema de distribuição quando o usuário faz uma seleção correta. O seletor de camada pergunta ao sistema de camadas qual é o próximo subnível, porque ele contém todas os subníveis ordenados por nível energético e mantém o registro de qual camada e subnível o usuário está preenchendo no momento.

Figura 78 Parte do Sistema Relacionada ao Preenchimento de Camadas



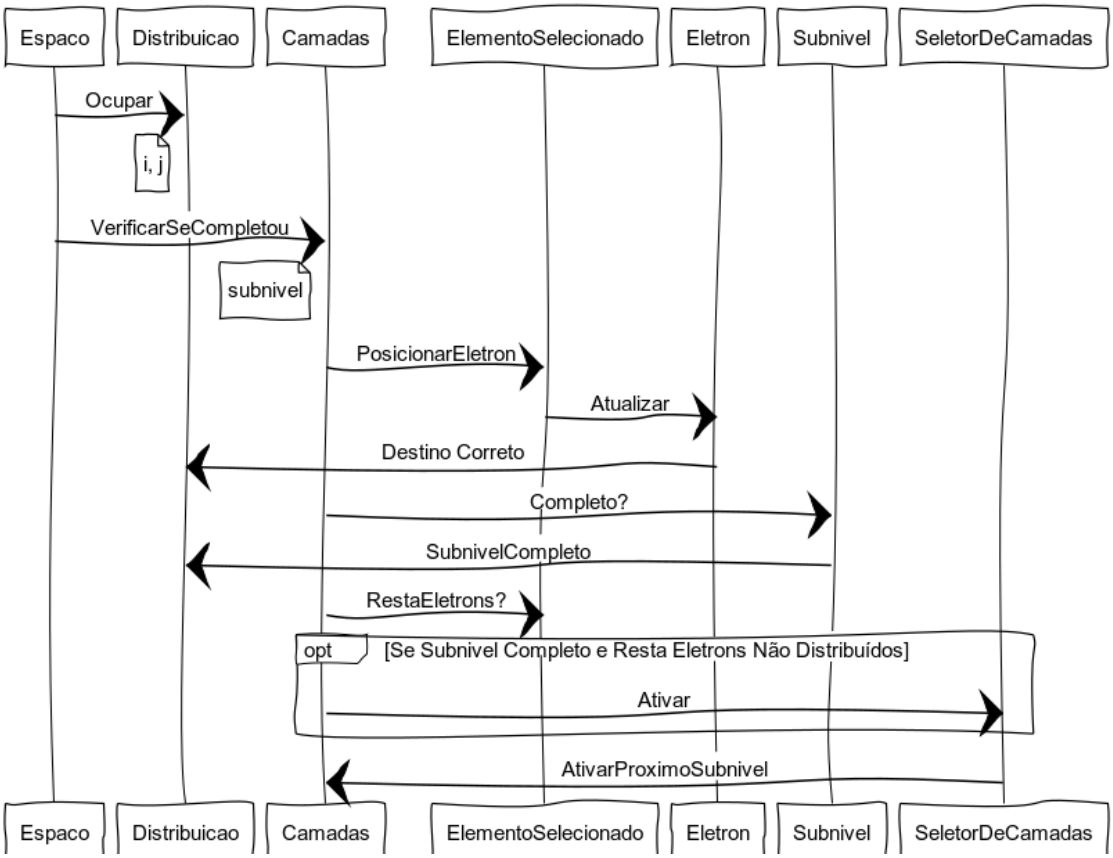
Fonte: Próprio Autor

Figura 79 Menu de Escolha da Camada e Subnível Seguinte



Fonte: Adaptador de (FIDI; LT, 2006)

Figura 80 Diagrama de Sequência de Preenchimento de Subnível



As notas abaixo das mensagens são os parâmetros enviados. Fonte: Próprio Autor

A principal interação do usuário com o sistema é através de cliques nos espaços vazios dos subníveis. A Figura 80 mostra um diagrama de sequência que ilustra quais mensagens são trocadas entre as duas partes do sistema mencionadas anteriormente: O sistema de distribuição de camadas e o seletor de próxima camada.

O diagrama mostra apenas o cenário onde o usuário faz um clique no espaço correto, cada clique correto posiciona, nas coordenadas do cubo que representa o espaço, uma esfera que corresponde à um spin.

Um clique errado coloca em execução uma co-rotina que altera a cor da textura, do espaço clicado, para vermelho e escreve a mensagem de texto “Sequência Energética Incorreta” na tela. A co-rotina é suspendida por 1 segundo e ao retornar altera a cor da textura para branco e remove a mensagem de texto.

Quando o usuário finaliza a distribuição dos elétrons, o local que exibe o número de elétrons restantes altera para mostrar a configuração eletrônica em formato de texto para o usuário. Espera-se que os elementos gráficos forneçam recursos visuais que possam auxiliar o estudante, a assimilar mais facilmente a configuração eletrônica em formato de texto, que é apresentada nos livros e em sala de aula.

3.9 – Formulários de Avaliação dos Aplicativos

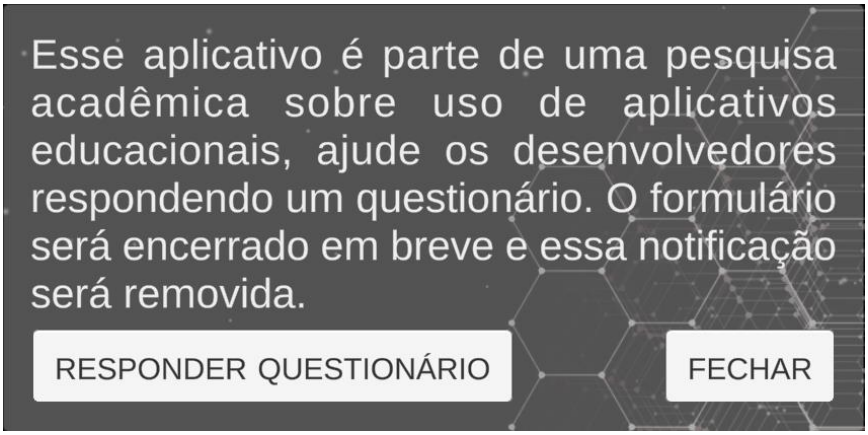
Após o desenvolvimento e publicação dos aplicativos, houve uma pesquisa em busca de referências para elaboração de questionários que pudessem validar a eficácia dos *softwares.* Para isso foram escolhidos os trabalhos de SILVA (2015) e JÚNIOR et al. (2012). SILVA (2015) elaborou uma metodologia para avaliação de aplicativos educacionais de matemática que poderia ser adaptada para os aplicativos de química desenvolvidos nesse trabalho. Portanto, o formulário desenvolvido por ela foi usado como referência. JÚNIOR et al. (2012) avaliaram um aplicativo de química através de questionários de avaliação e seu procedimento de análise dos dados dos formulários será usado nesse trabalho.

Foram elaborados dois formulários de avaliação, um para cada *software*. Os formulários possuem 17 perguntas, 15 delas foram adaptadas do trabalho de SILVA

(2015). Ambos os questionários podem ser encontrados no apêndice. A primeira pergunta tem como objetivo descobrir se os usuários do aplicativo eram, em sua maioria, alunos graduandos ou professores de química. Já a segunda pergunta tinha como objetivo apontar como a divulgação do aplicativo aconteceu. Todas as perguntas seguintes são afirmações que fazem parte da avaliação quantitativa, que será usada para validar os aplicativos. Os usuários deveriam manifestar a concordância com cada afirmação, assinalando um número em uma escala de 1-5, sendo que o número um (1) significa “discordância plena” e o número cinco (5) “concordância plena”. Foi calculado uma média desses valores, para cada questão, que será usada como métrica durante a discussão dos resultados, no final da pesquisa.

Durante o período de 18 a 22 de julho/2019, foram implementadas mensagens (Figura 81) em cada uma das versões dos aplicativos, na tela inicial junto com um link para o formulário, convidando os usuários para preencherem o questionário de avaliação para esse projeto de pesquisa. Essas mensagens serão removidas dos aplicativos após a finalização desse trabalho. Os links para os formulários não foram publicados em grupos de redes sociais, um dos objetivos dos questionários era obter uma amostra dos usuários dos aplicativos e publicar os links em um grupo de professores de química, por exemplo, convidando-os para avaliar as ferramentas desenvolvidas, poderia distorcer os resultados.

Figura 81 Link para Formulário de Avaliação



Mensagem na tela inicial dos aplicativos durante o período de avaliação. Fonte: Próprio Autor

CAPÍTULO 4

RESULTADOS

Nesse capítulo são apresentados e discutidos os resultados obtidos dos questionários de avaliação. As respostas dos questionários de avaliação foram coletadas em 03 de setembro de 2019, o período de coleta durou aproximadamente 47 dias e resultou em um total de 115 submissões: 72 referentes ao aplicativo Geometria Molecular e 43 ao Configuração Eletrônica. Era de se esperar que o Geometria Molecular recebesse mais avaliações, uma vez que teve 1.590 *downloads* a mais que o Configuração Eletrônica, em um período menor.

Esse capítulo também apresenta algumas imagens dos aplicativos em suas versões finais e alguns relatórios das lojas virtuais onde os aplicativos foram publicados.

4.1 – Resultados do Questionário de Geometria Molecular

Uma avaliação voluntária do aplicativo Geometria Molecular foi realizada por setenta e dois (72) usuários que clicaram no link apresentado na tela inicial do aplicativo e responderam ao formulário de avaliação (disponível no Apêndice) criado no *Google Forms.* A primeira pergunta (Figura 82) do formulário tinha como objetivo descobrir se os usuários do aplicativo eram apenas alunos, graduandos ou graduados em química, e professores de química. Vinte e cinco (25) usuários (36,1%) responderam que não são graduandos, graduados ou professores de química (63,9%), o que nos sugere que há interesse e uso do aplicativo em outras áreas como, por exemplo, engenharia e ciência dos materiais. Seria útil adicionar um campo extra para o usuário preencher com essa informação, em um próximo formulário.

A segunda pergunta (Figura 83) tinha como objetivo descobrir a forma como o aplicativo está sendo encontrado. Pode-se observar que os nomes dos aplicativos realmente facilitam as buscas, uma vez que a maior parte (38,9%) dos usuários (28) encontrou o aplicativo pesquisando diretamente nas lojas de aplicativos. Como os

aplicativos foram divulgados em grupos no Facebook, esperava-se que a maior parte das respostas seria a de Redes Sociais, quando na realidade ela representou 8,3%, a menor quantidade (apenas 6 usuários). As respostas dessa pergunta também revelaram que a segunda maior parte (27,8%) dos usuários (20) tomou conhecimento do aplicativo através de indicação de um colega, professor ou aluno, o que sugere que os usuários estão recomendando o uso do aplicativo.

Figura 82 Primeira Pergunta do Formulário de Geometria Molecular

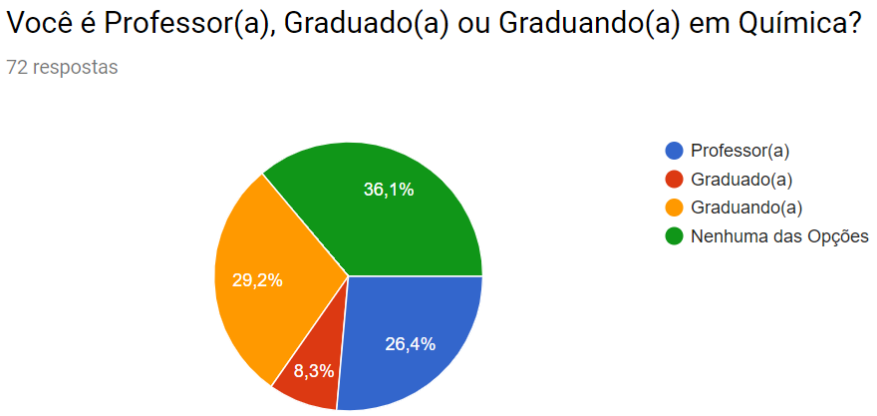
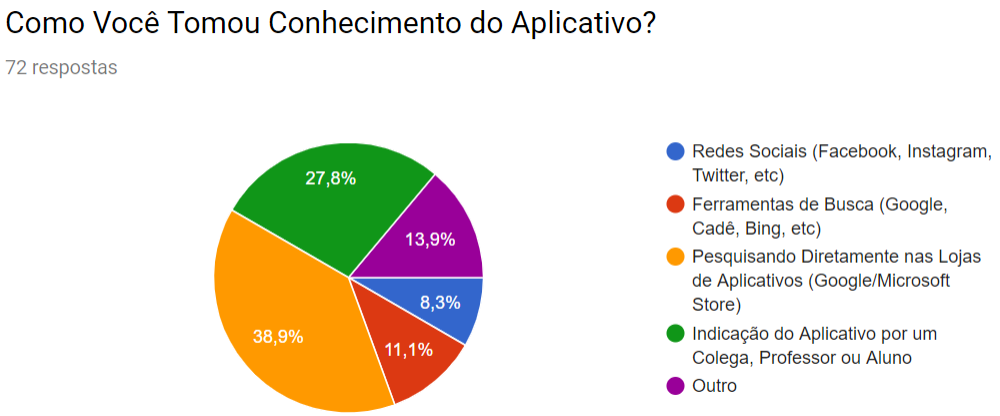


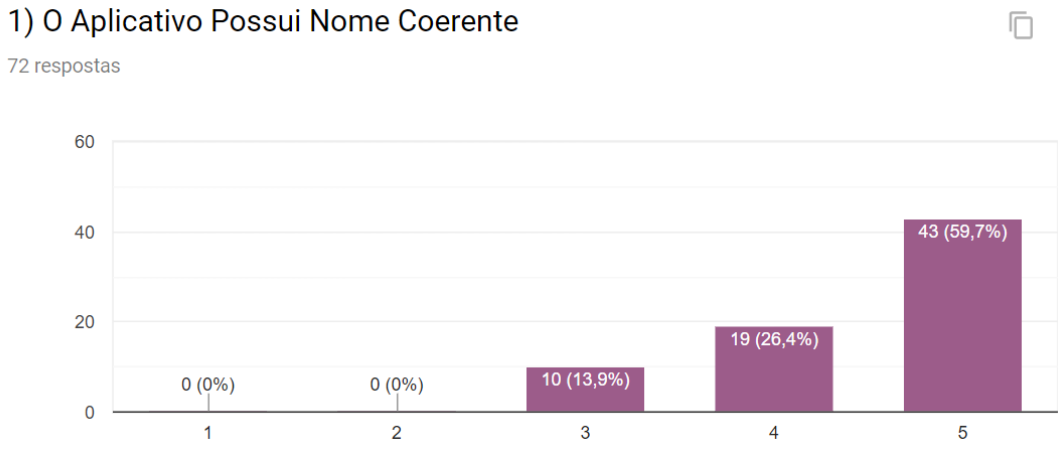
Figura 83 Segunda Pergunta do Formulário de Geometria Molecular



Todas as próximas perguntas são afirmações que fazem parte da avaliação quantitativa, que será usada para validar os aplicativos. Os usuários manifestaram sua concordância com cada afirmação, marcando um número em uma escala de 1 a 5, sendo que o número um (1) significa “discordância completa” e o número cinco (5) “concordância completa”. Ao final, a média desses valores foi calculada.

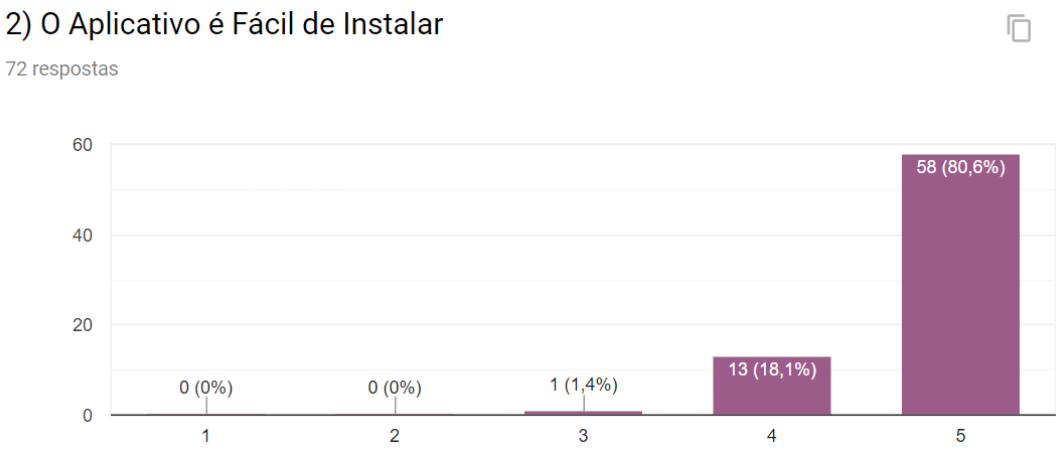
A terceira pergunta (Figura 84) tinha como objetivo avaliar o nome escolhido para o aplicativo. A média de respostas foi 4,45 e não houve formulário algum discordando. Portanto, pôde-se considerar que nome do aplicativo é coerente.

Figura 84 Terceira Pergunta do Formulário de Geometria Molecular



A quarta pergunta (Figura 85) tinha como objetivo avaliar a facilidade de instalação do aplicativo. Acreditava-se que o aplicativo deveria estar disponível nas lojas virtuais, da *Google* e *Microsoft,* justamente pela praticidade que proporcionam para instalação de aplicativos. A média de concordância foi de 4,79 e não houve formulário algum discordando. Por isso, concluímos que o aplicativo é fácil de instalar.

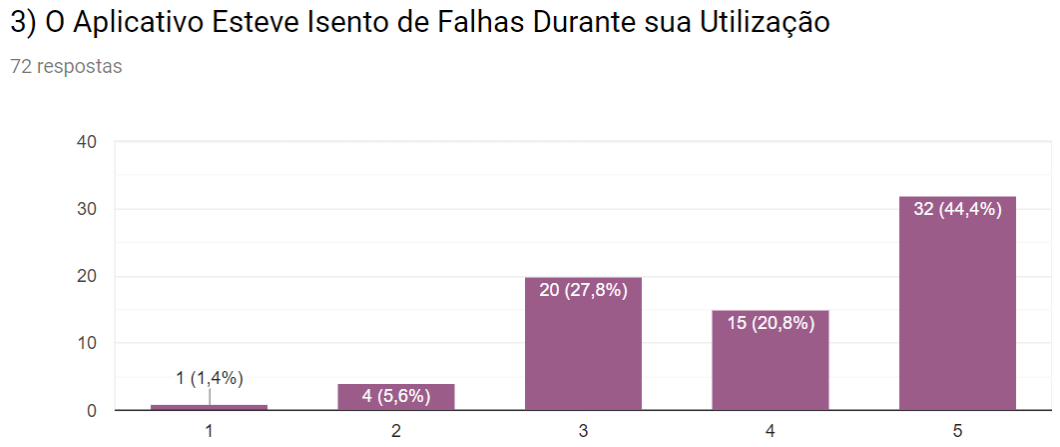
Figura 85 Quarta Pergunta do Formulário de Geometria Molecular



A quinta pergunta (Figura 86) tinha como objetivo validar que o aplicativo está funcionando em *hardwares* diferentes daqueles usados durante seu desenvolvimento.

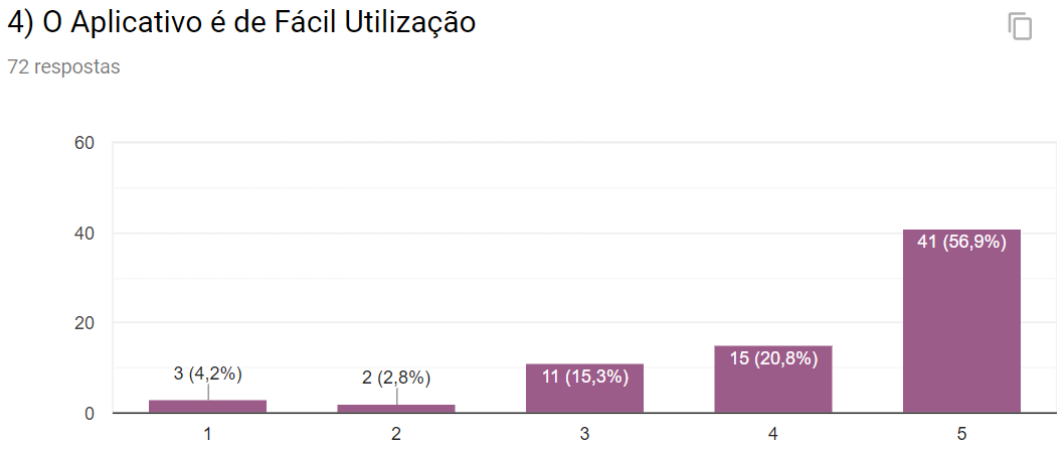
A média de respostas foi 4,00 e apesar da concordância, acredita-se que esse é um ponto que representa uma oportunidade de melhoria. Foi feita uma pesquisa por relatórios de falhas na *Google Play Store* durante o período da pesquisa (18/07/19 – 03/09/19) e nenhuma falha foi encontrada, o que aponta que as falhas relatadas pelos usuários não são técnicas como, por exemplo, o aplicativo travar.

Figura 86 Quinta Pergunta do Formulário de Geometria Molecular



A sexta pergunta (Figura 87) tinha como objetivo avaliar a facilidade de uso do aplicativo. A média de respostas foi 4,23 e apesar da concordância, sugere-se que seja implementado um tutorial de utilização, na tela inicial, quando o aplicativo for inicializado pela primeira vez, para facilitar ainda mais seu uso.

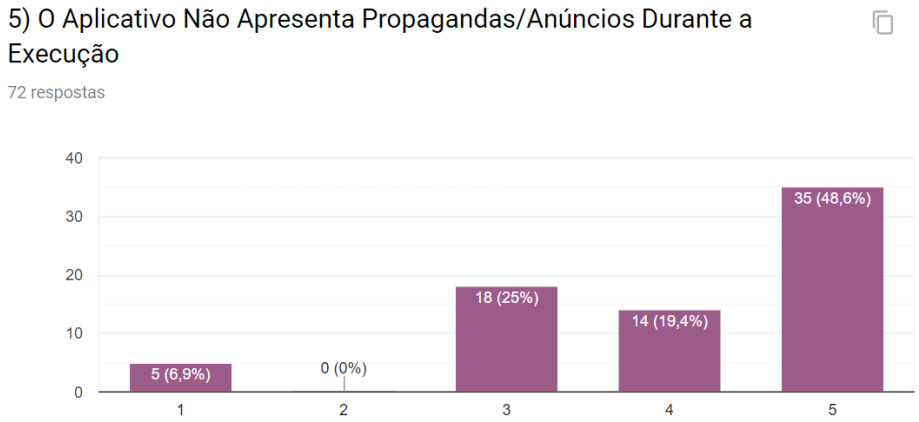
Figura 87 Sexta Pergunta do Formulário de Geometria Molecular



A sétima pergunta (Figura 88) foi adicionada porque poderia ser usada como filtro, caso houvesse interesse. O aplicativo não tem propagandas, podemos supor

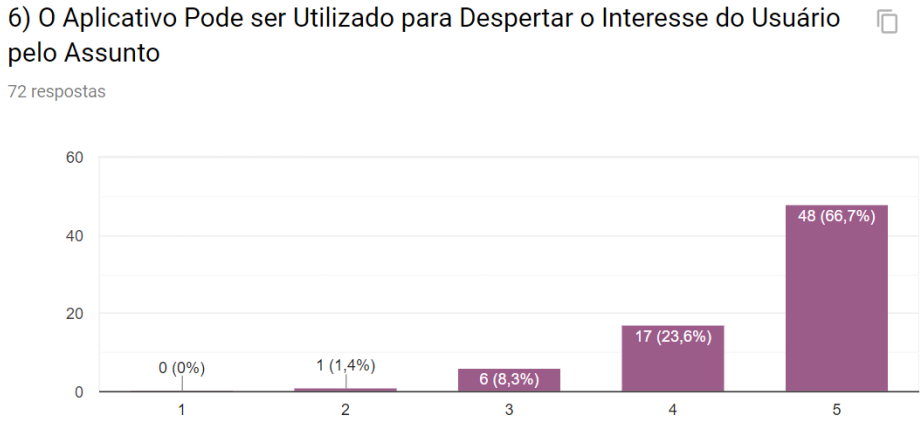
que os usuários que discordaram dessa afirmativa podem não ter respondido o formulário de avaliação com seriedade. Os usuários que marcaram a opção 3 (não concordaram, nem discordaram) podem ter ficado na dúvida, o que sugere que essa pergunta seria beneficiada de uma opção N.A. (Não Aplicável) para remover dos cálculos os usuários que não souberam o que responder. Concordância média: 4,01.

Figura 88 Sétima Pergunta do Formulário de Geometria Molecular



A oitava pergunta (Figura 89) tinha como objetivo avaliar o uso do aplicativo para despertar o interesse dos estudantes. A média de respostas foi 4,55 e maioria das respostas concordou completamente com a afirmativa. Considera-se, então, que o aplicativo pode ser usado para despertar o interesse do usuário pelo assunto.

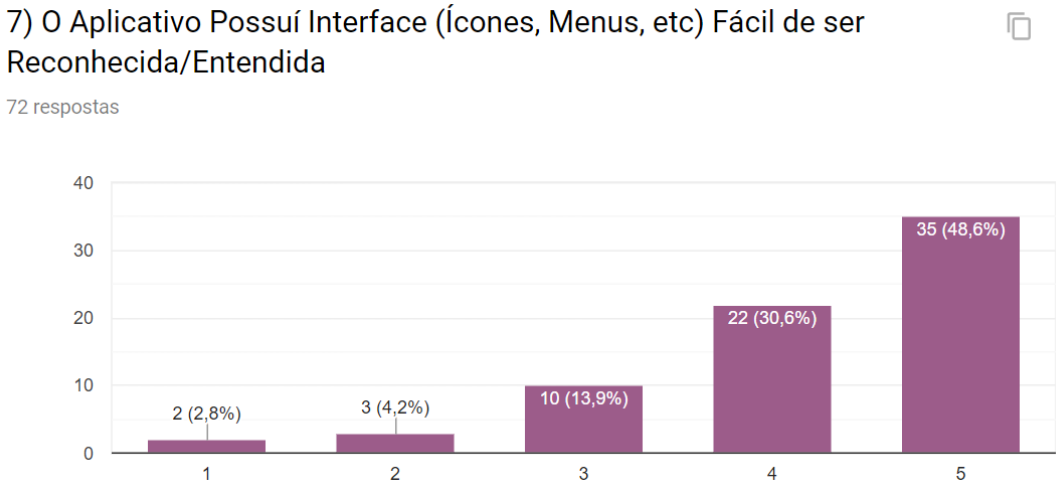
Figura 89 Oitava Pergunta do Formulário de Geometria Molecular



A nona pergunta (Figura 90) tinha como objetivo avaliar a interface do aplicativo. O aplicativo possui poucas funcionalidades e botões atualmente, acredita-se que

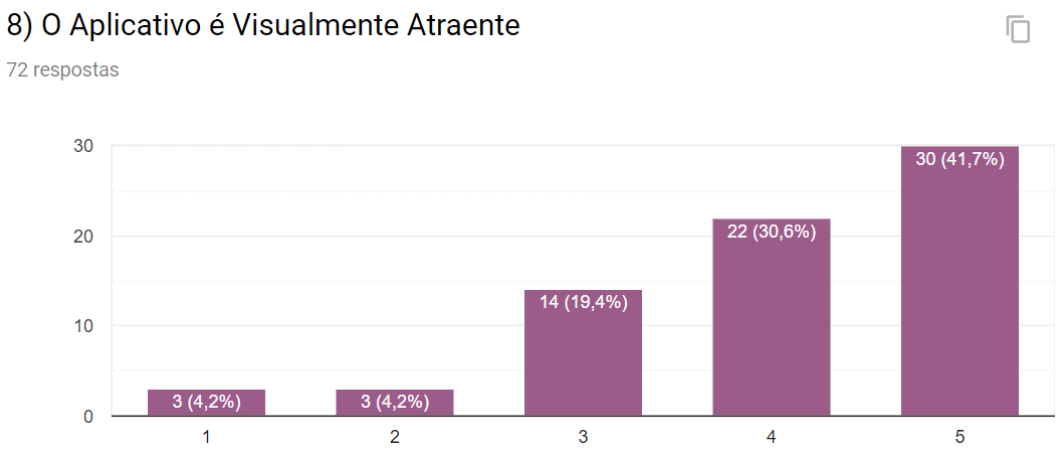
não há muito espaço para alterações nesta área. A média de respostas foi 4,17 e 79,2% dos usuários concordaram com a afirmativa. Portanto, pôde-se determinar que o aplicativo tem uma interface fácil ser entendida.

Figura 90 Nona Pergunta do Formulário de Geometria Molecular



A décima pergunta (Figura 91) tinha como objetivo avaliar visualmente o aplicativo. Houve certo investimento de tempo para tornar os gráficos dos orbitais visualmente atraentes, porque acredita-se que isso atrairia mais a atenção dos usuários. A média de respostas foi 4,01 e 72,3% dos usuários concordaram com a afirmativa. Apesar da concordância, existe oportunidade de melhorias na aparência dos botões.

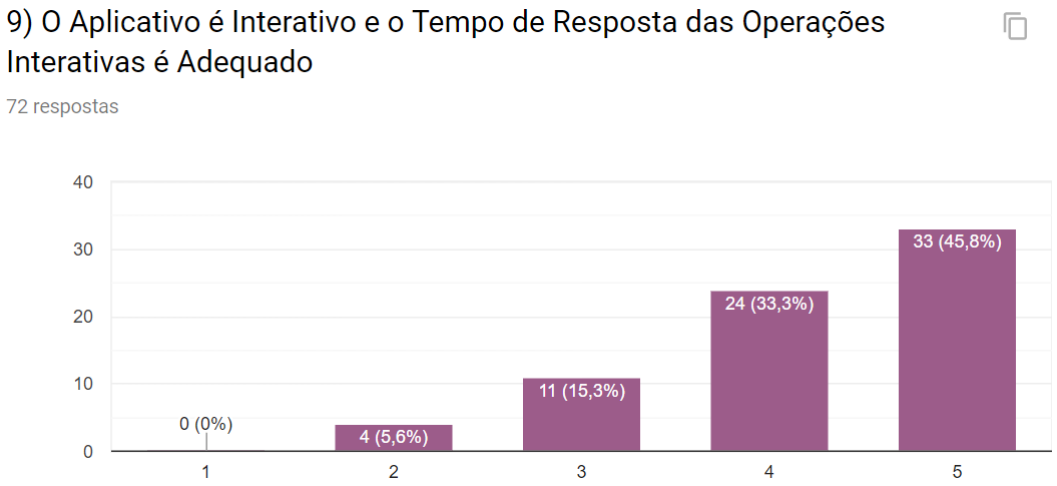
Figura 91 Décima Pergunta do Formulário de Geometria Molecular



A décima primeira pergunta (Figura 92) tinha como objetivo avaliar o desempenho do aplicativo. O aplicativo foi desenvolvido usando-se um celular Moto G5 para

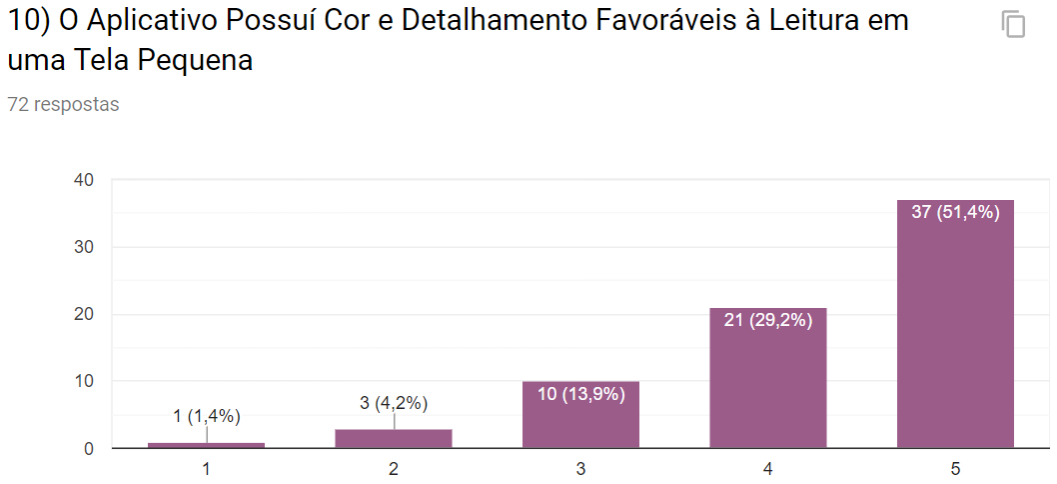
testes, em um *hardware* inferior poderia haver queda na taxa de quadros e tempo de resposta. Como a média de respostas foi 4,18, acredita-se que os usuários estão satisfeitos com o tempo de resposta do aplicativo no geral.

Figura 92 Décima Primeira Pergunta do Formulário de Geometria Molecular



A décima segunda (Figura 93) tinha como objetivo avaliar o estilo do texto. Durante o desenvolvimento, houve uma a tendência em aumentar o tamanho dos botões e da fonte dos textos, com a intenção de facilitar o uso em telas pequenas. A média de respostas foi 4,24 e 80,6% dos usuários concordaram com a afirmativa.

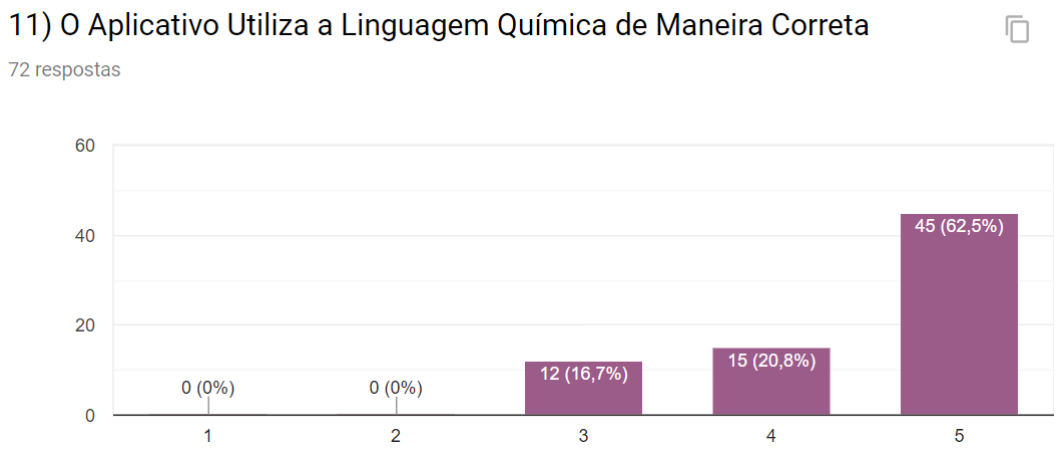
Figura 93 Décima Segunda Pergunta do Formulário de Geometria Molecular



A décima terceira pergunta (Figura 94) tinha como objetivo avaliar a precisão na linguagem, referente ao conteúdo de química. A média de respostas foi 4,45 e

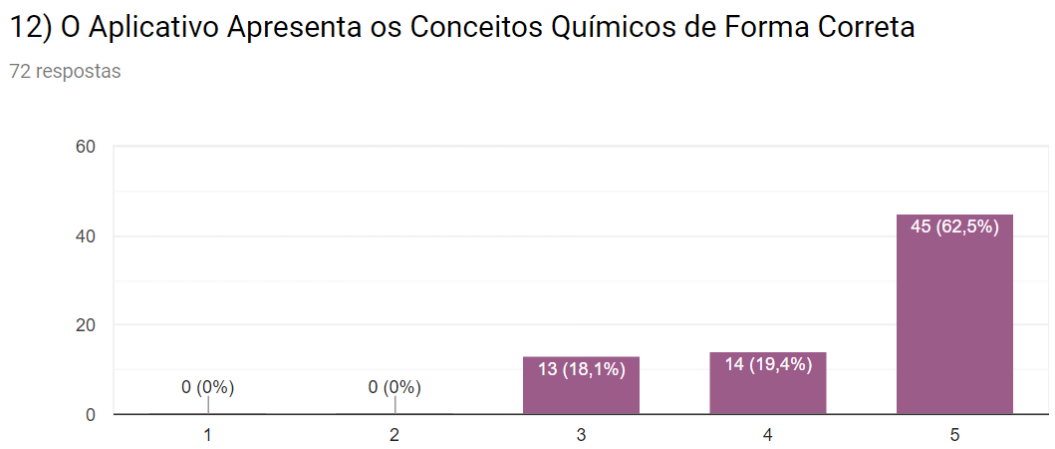
83,3% dos usuários concordaram com a afirmativa. Não houve formulário algum discordando, portanto, pôde-se considerar que linguagem química está correta.

Figura 94 Décima Terceira Pergunta do Formulário de Geometria Molecular



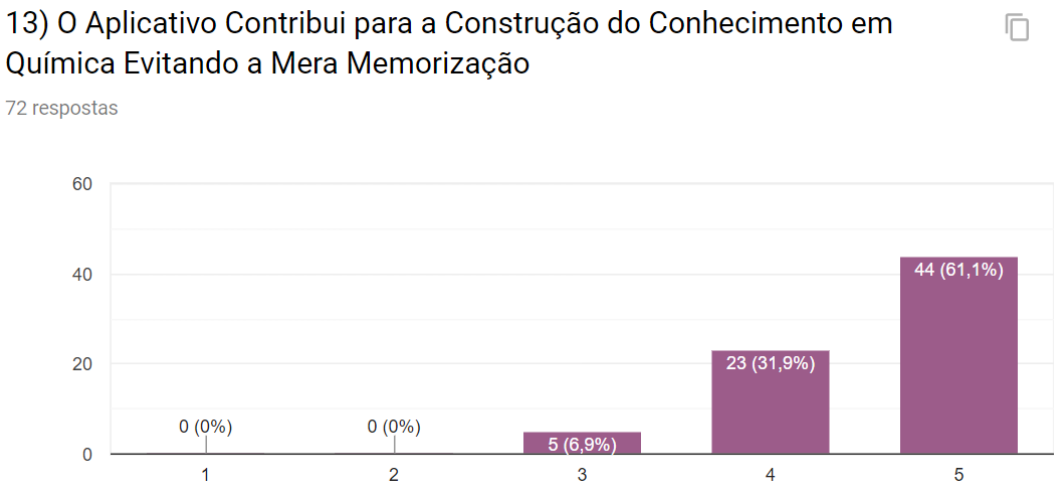
A décima quarta pergunta (Figura 95) tinha como objetivo avaliar a exatidão nos conceitos de química apresentados. A média de respostas foi 4,44 e 81,9% dos usuários concordaram com a afirmativa. Não houve formulário algum discordando, por isso, considera-se que os conceitos químicos apresentados estão corretos.

Figura 95 Décima Quarta Pergunta do Formulário de Geometria Molecular



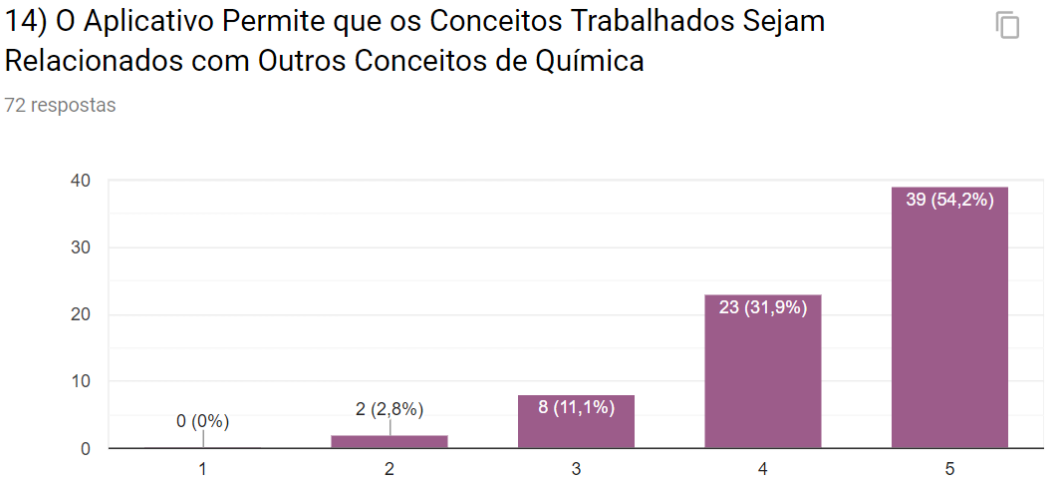
O objetivo da décima quinta pergunta (Figura 96) era avaliar se o aplicativo poderia ser usado como um incentivo a experimentação pelo aluno, e produzir um aprendizado através da exploração, ao invés de memorização. Não houve formulário algum discordando e essa foi uma das respostas com maior concordância, com uma média de 4,54 onde 93% dos usuários concordaram com a afirmativa.

Figura 96 Décima Quinta Pergunta do Formulário de Geometria Molecular



O objetivo da décima sexta pergunta (Figura 97) era avaliar se conceitos trabalhados no aplicativo poderiam ser relacionados com outros conceitos de química. Esperava-se uma concordância alta, porque os conceitos de geometria molecular e configuração eletrônica podem ser relacionados, por exemplo. Apesar de algumas discordâncias, a média foi de 4,37 onde 86,1% concordaram com a afirmativa.

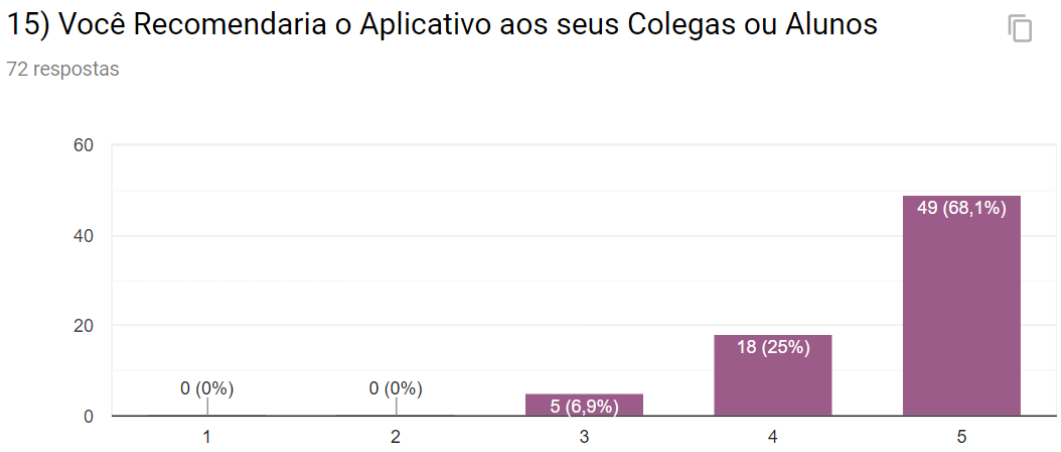
Figura 97 Décima Sexta Pergunta do Formulário de Geometria Molecular



A última pergunta do formulário (Figura 98) tinha como objetivo validar se os usuários recomendariam o aplicativo. Nenhum usuário afirmou que não recomendaria, essa foi uma das respostas com maior concordância, com uma média de 4,61 onde 93,1% afirmaram que recomendariam o aplicativo, uma informação que pode ser

relacionada com a segunda pergunta do formulário, que apontava que 27,8% dos usuários souberam do aplicativo através de recomendação de colegas.

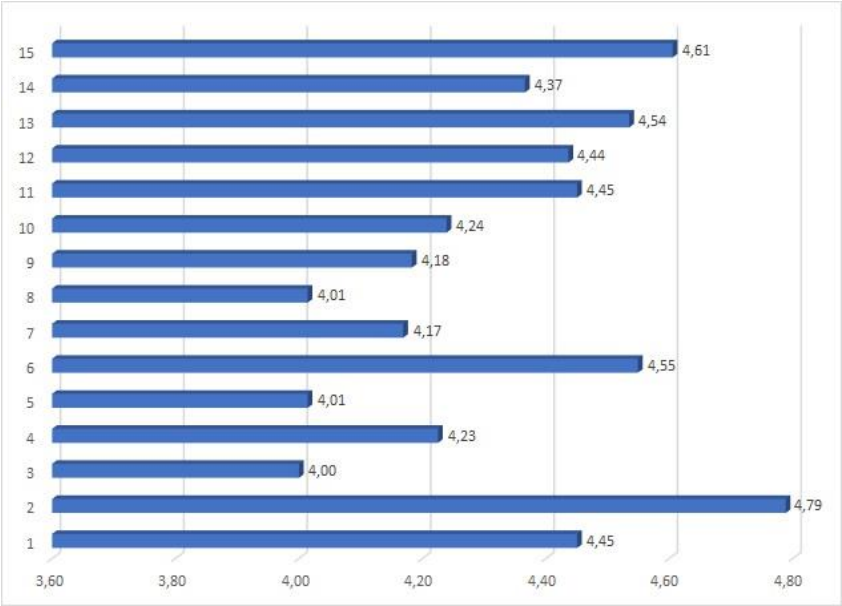
Figura 98 Décima Sétima Pergunta do Formulário de Geometria Molecular



A Figura 99 mostra um gráfico com a média das respostas de cada pergunta. Observou-se que todas as afirmativas ficaram com uma média igual ou superior a 4, o que equivale a opção de “Concordância” segundo a legenda do formulário.

A média total do aplicativo é 4,34, o que se permite afirmar principalmente que o software contribui para a construção do conhecimento em química, apresenta conceitos químicos de forma correta e pode ser usado para despertar o interesse do usuário pelo assunto. Além de validar aspectos técnicos como facilidade de uso e instalação, assim como tempo de resposta, isenção de falhas, interface fácil de usar e gráficos visualmente atraentes. De forma geral, pode-se dizer que o aplicativo foi bem aceito e é recomendado por seus usuários.

Figura 99 Média das Respostas dos Formulários de Geometria Molecular



Foram Registradas um Total de 72 Respostas (Média Total: 4,34). Fonte: Próprio Autor

4.2 – Resultados do Questionário de Configuração Eletrônica

Uma avaliação voluntária, do aplicativo Configuração Eletrônica, foi realizada por quarenta e três (43) usuários, que clicaram no link apresentado na tela inicial do aplicativo e responderam ao formulário de avaliação (disponível no Apêndice) criado no *Google Forms.* A primeira pergunta (Figura 100) do formulário tinha como objetivo descobrir se os usuários do aplicativo eram apenas alunos, graduandos ou graduados em química, e professores de química. Treze (13) usuários (30,2%) responderam que não são graduandos, graduados ou professores de química (69,8%), o que nos sugere que há interesse e uso do aplicativo em outras áreas como, por exemplo, engenharia e ciência dos materiais. Seria útil adicionar um campo extra para o usuário preencher com essa informação, em um próximo formulário.

A segunda pergunta (Figura 101) tinha como objetivo descobrir a forma como o aplicativo está sendo encontrado. Pode-se observar que os nomes dos aplicativos realmente facilitam as buscas, uma vez que a maior parte (41,9%) dos usuários (18) encontrou o aplicativo pesquisando diretamente nas lojas de aplicativos. Como os aplicativos foram divulgados em grupos no Facebook, esperava-se que a maior parte

das respostas seria a de Redes Sociais, quando na realidade ela representou, 7%, a menor quantidade (apenas 5 usuários). As respostas dessa pergunta também revelaram que a segunda maior parte (25,6%) dos usuários (11) tomou conhecimento do aplicativo através de indicação de um colega, professor ou aluno, o que sugere que os usuários estão recomendando o uso do aplicativo.

Figura 100 Primeira Pergunta do Formulário de Configuração Eletrônica

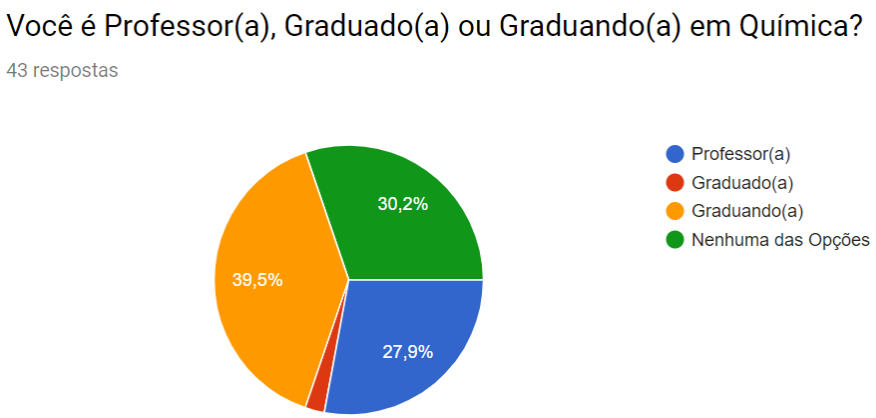
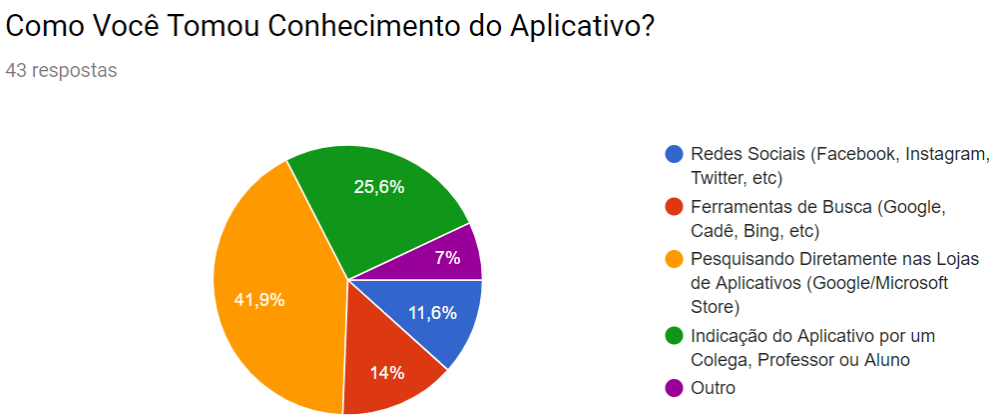


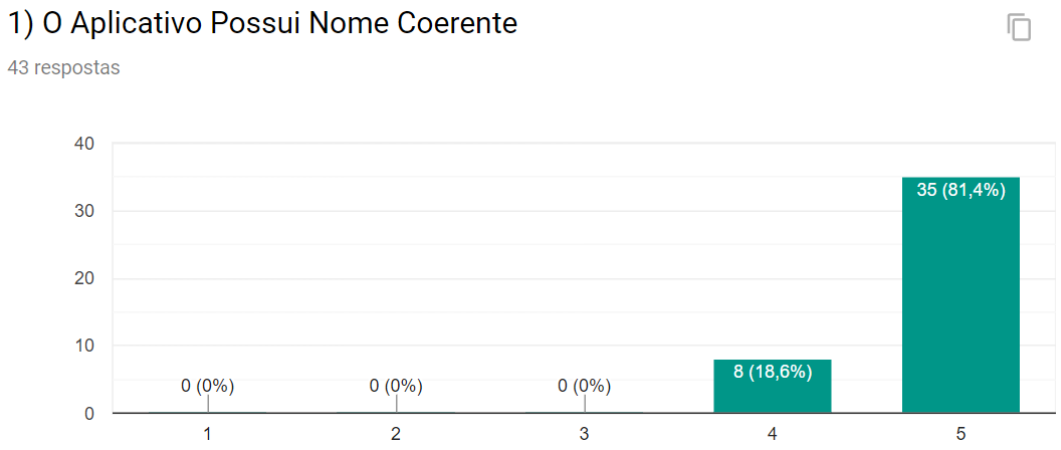
Figura 101 Segunda Pergunta do Formulário de Configuração Eletrônica



Todas as próximas perguntas são afirmações que fazem parte da avaliação quantitativa, que será usada para validar os aplicativos. Os usuários manifestaram sua concordância com cada afirmação, marcando um número em uma escala de 1 a 5, sendo que o número um (1) significa “discordância completa” e o número cinco (5) “concordância completa”. Ao final, a média desses valores foi calculada.

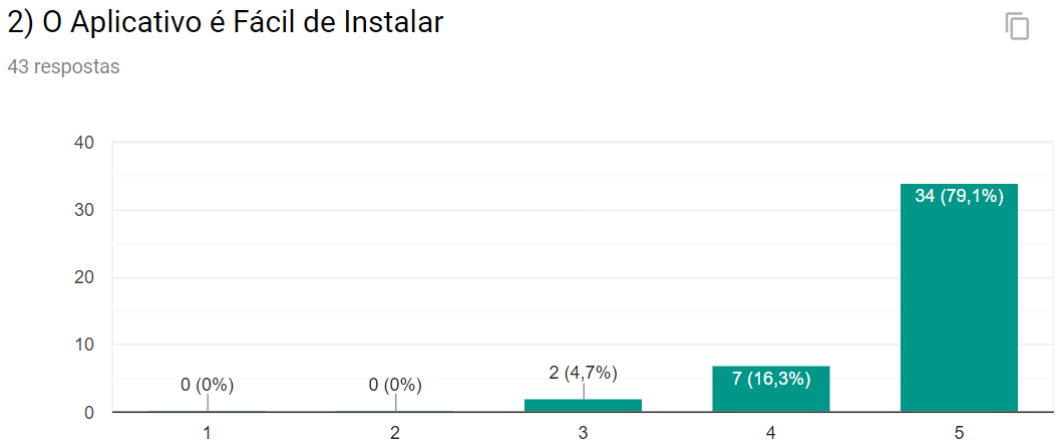
A terceira pergunta (Figura 102) tinha como objetivo avaliar o nome escolhido para o aplicativo. A média de respostas foi 4,81 e todos os usuários concordaram. Portanto, pôde-se considerar que nome do aplicativo é coerente.

Figura 102 Terceira Pergunta do Formulário de Configuração Eletrônica



A quarta pergunta (Figura 103) tinha como objetivo avaliar a facilidade de instalação do aplicativo. Acreditava-se que o aplicativo deveria estar disponível nas lojas virtuais, da *Google* e *Microsoft*, justamente pela praticidade que proporcionam para instalação de aplicativos. A média de concordância foi de 4,74 e 95,4% dos usuários concordaram com a afirmativa. Nenhum usuário discordou da afirmativa, por isso, concluímos que o aplicativo é fácil de instalar.

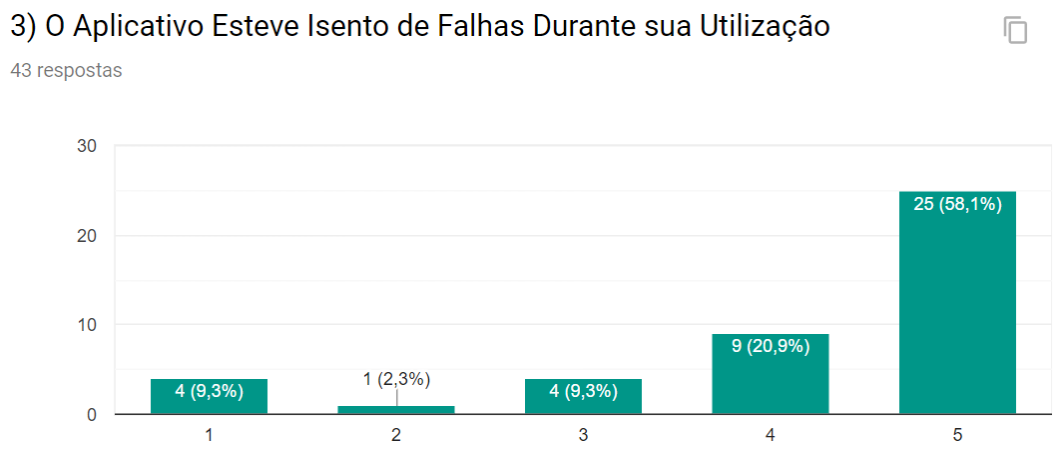
Figura 103 Quarta Pergunta do Formulário de Configuração Eletrônica



A quinta pergunta (Figura 104) tinha como objetivo validar que o aplicativo está funcionando em hardwares diferentes daqueles usados durante seu desenvolvimento.

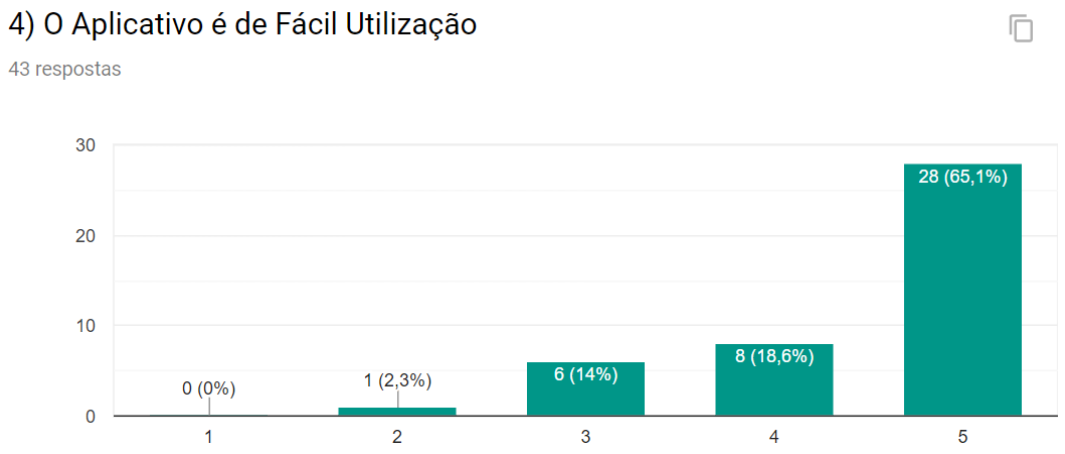
A média de respostas foi 4,16 e apesar da concordância, acredita-se que esse é um ponto que representa uma oportunidade de melhoria. Foi feita uma pesquisa por relatórios de falhas na *Google Play Store* durante o período da pesquisa (18/07/19 – 03/09/19) e foram encontradas 4 falhas, o que aponta que as falhas relatadas pelos usuários são, em sua maioria, técnicas como, por exemplo, o aplicativo travar ou não executar.

Figura 104 Quinta Pergunta do Formulário de Configuração Eletrônica



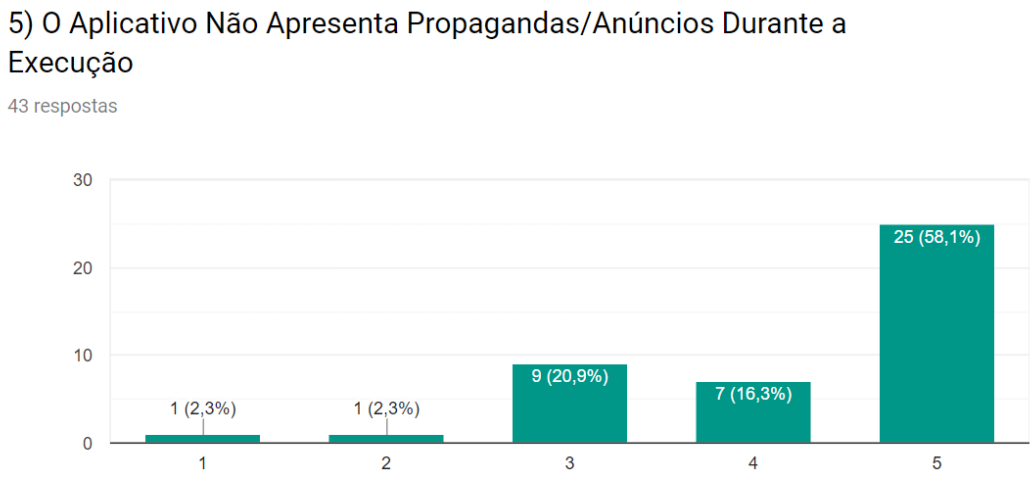
A sexta pergunta (Figura 105) tinha como objetivo avaliar a facilidade de uso do aplicativo. A média de respostas foi 4,47 e apesar da concordância, sugere-se que seja implementado um tutorial de utilização, na tela inicial, quando o aplicativo for inicializado pela primeira vez, para facilitar ainda mais seu uso.

Figura 105 Sexta Pergunta do Formulário de Configuração Eletrônica



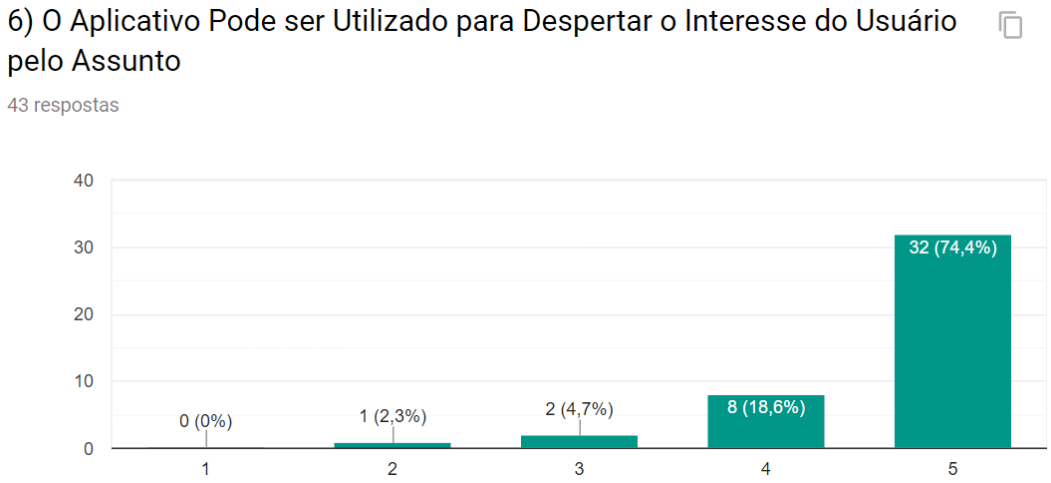
A sétima pergunta (Figura 106) foi adicionada porque poderia ser usada como filtro, caso houvesse interesse. O aplicativo não tem propagandas, podemos supor que os usuários que discordaram dessa afirmativa podem não ter respondido o formulário de avaliação com seriedade. Os usuários que marcaram a opção 3 (não concordaram, nem discordaram) podem ter ficado na dúvida, o que sugere que essa pergunta seria beneficiada de uma opção N.A. (Não Aplicável) para remover dos cálculos os usuários que não souberam o que responder. Uma possibilidade é a de que os usuários interpretaram a logo da UENF ou CAPES como algum tipo de propaganda. Concordância média: 4,26.

Figura 106 Sétima Pergunta do Formulário de Configuração Eletrônica



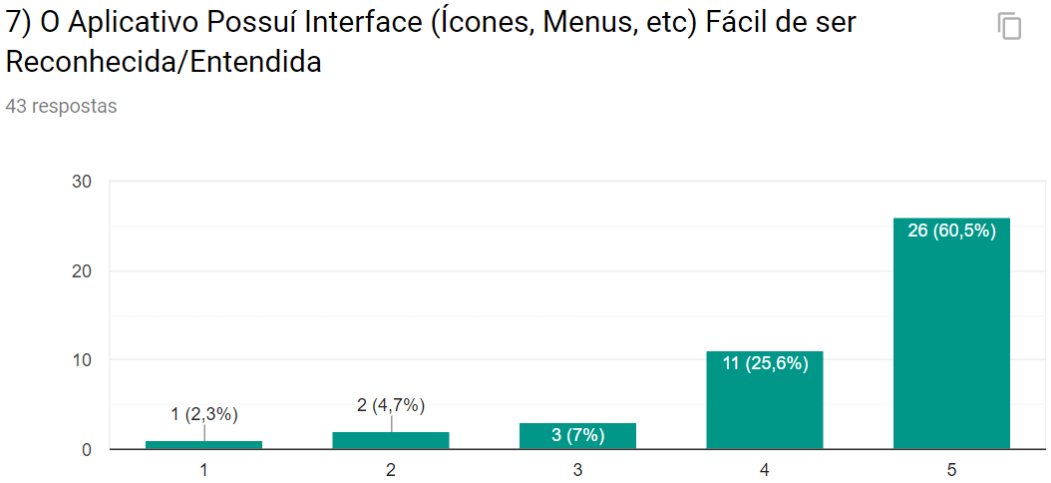
A oitava pergunta (Figura 107) tinha como objetivo avaliar o uso do aplicativo para despertar o interesse dos estudantes. A média de respostas foi 4,65 e 93% dos usuários concordaram com a afirmativa. Considera-se, então, que o aplicativo pode ser usado para despertar o interesse do usuário pelo assunto.

Figura 107 Oitava Pergunta do Formulário de Configuração Eletrônica



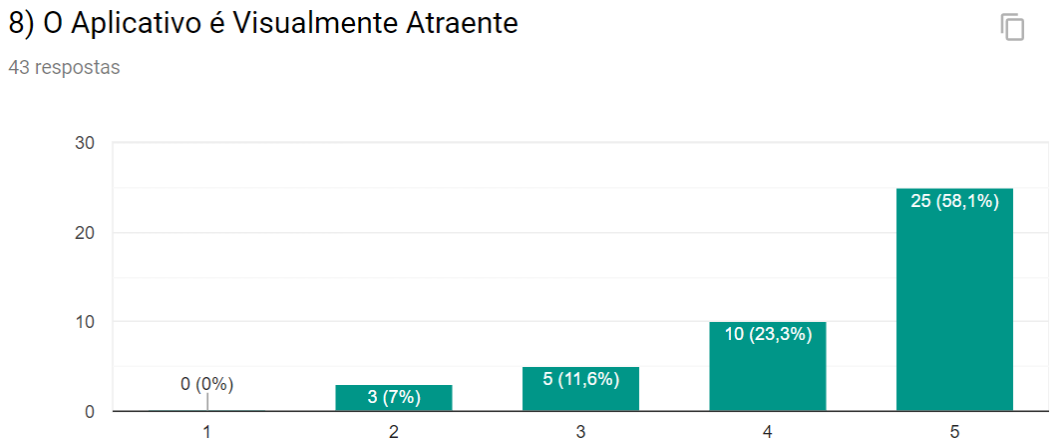
A nona pergunta (Figura 108) tinha como objetivo avaliar a interface do aplicativo. O aplicativo possui poucas funcionalidades e é bem simples, acredita-se que não há muito espaço para alterações nesta área. A média de respostas foi 4,37 e 86,1% dos usuários concordaram com a afirmativa. Portanto, pôde-se determinar que o aplicativo tem uma interface fácil ser entendida.

Figura 108 Nona Pergunta do Formulário de Configuração Eletrônica



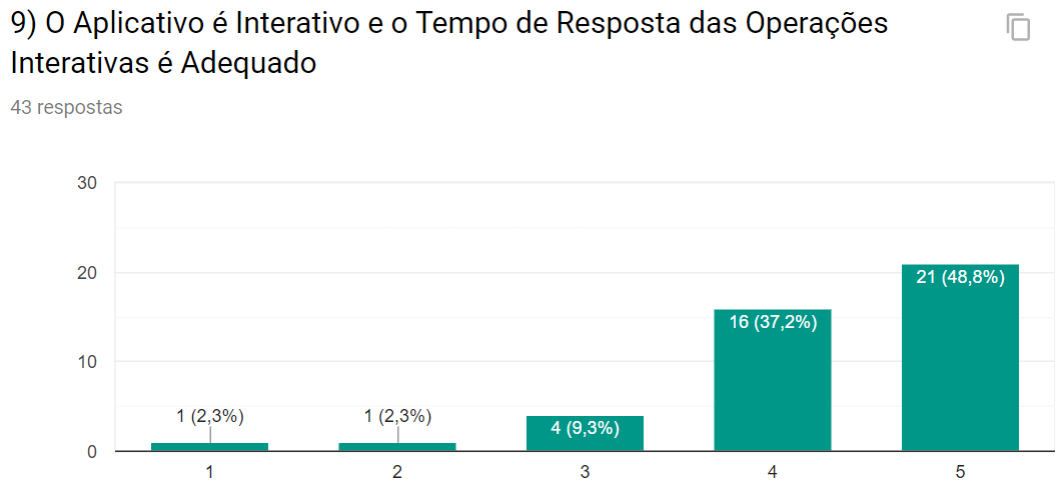
A décima pergunta (Figura 109) tinha como objetivo avaliar visualmente o aplicativo. Houve certo investimento de tempo para tornar os gráficos visualmente atraentes, porque acredita-se que isso atrairia mais a atenção dos usuários. A média de respostas foi 4,33 e 81,4% dos usuários concordaram com a afirmativa.

Figura 109 Décima Pergunta do Formulário de Configuração Eletrônica



A décima primeira pergunta (Figura 110) tinha como objetivo avaliar o desempenho do aplicativo. O aplicativo foi desenvolvido usando-se um celular Moto G5 para testes, em um hardware inferior poderia haver queda na taxa de quadros e tempo de resposta. Como a média de respostas foi 4,28 e 86% concordaram com a afirmativa, acredita-se que os usuários estão satisfeitos com o tempo de resposta do aplicativo no geral.

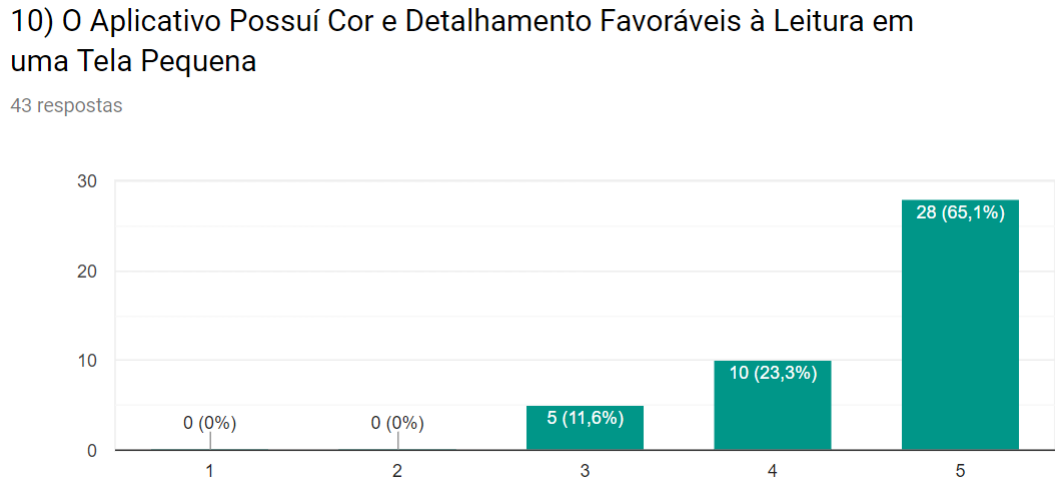
Figura 110 Décima Primeira Pergunta do Formulário de Configuração Eletrônica



A décima segunda (Figura 111) tinha como objetivo avaliar o estilo do texto. Durante o desenvolvimento, houve uma a tendência em aumentar o tamanho dos botões e da fonte dos textos, com a intenção de facilitar o uso em telas pequenas. A média de respostas foi 4,53 e 88,4% dos usuários concordaram com a afirmativa.

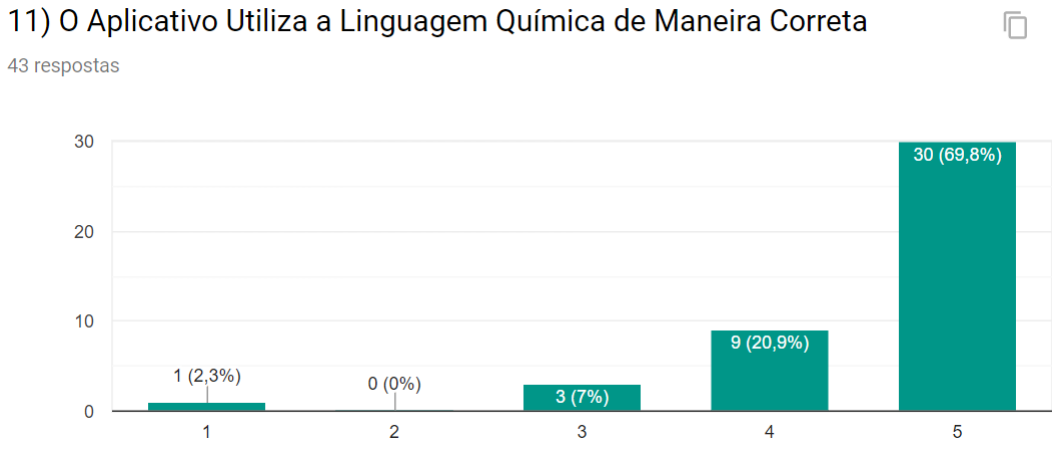
Nenhum usuário discordou, portanto, pôde-se concluir que nenhuma alteração precisa ser realizada nesse aspecto do aplicativo.

Figura 111 Décima Segunda Pergunta do Formulário de Configuração Eletrônica



A décima terceira pergunta (Figura 112) tinha como objetivo avaliar a precisão na linguagem, referente ao conteúdo de química. A média de respostas foi 4,56 e 90,7% dos usuários concordaram com a afirmativa. A única discordância é de um usuário que respondeu ser um(a) professor(a) na primeira pergunta, o que sugere que talvez seja preciso corrigir algum termo químico em algum lugar do *software*.

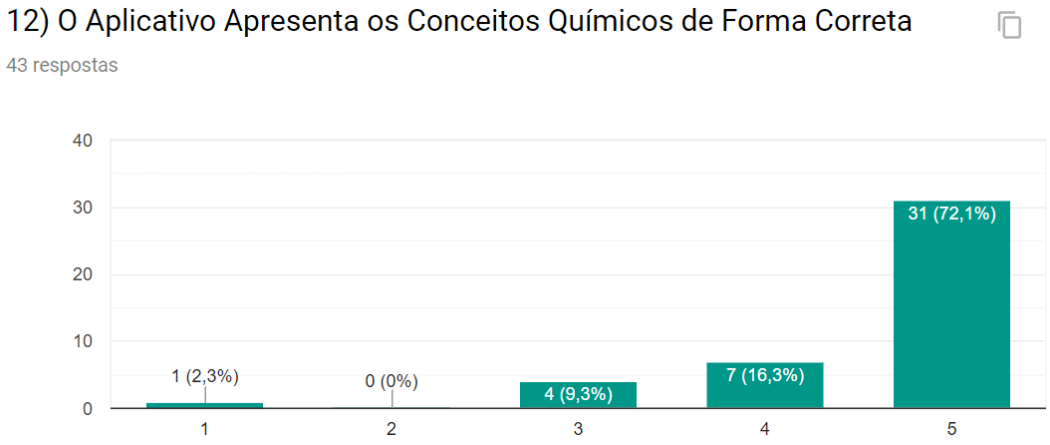
Figura 112 Décima Terceira Pergunta do Formulário de Configuração Eletrônica



A décima quarta pergunta (Figura 113) tinha como objetivo avaliar a exatidão nos conceitos de química apresentados. A média de respostas foi 4,56 e 88,4% dos usuários concordaram com a afirmativa. A única discordância é de um usuário que

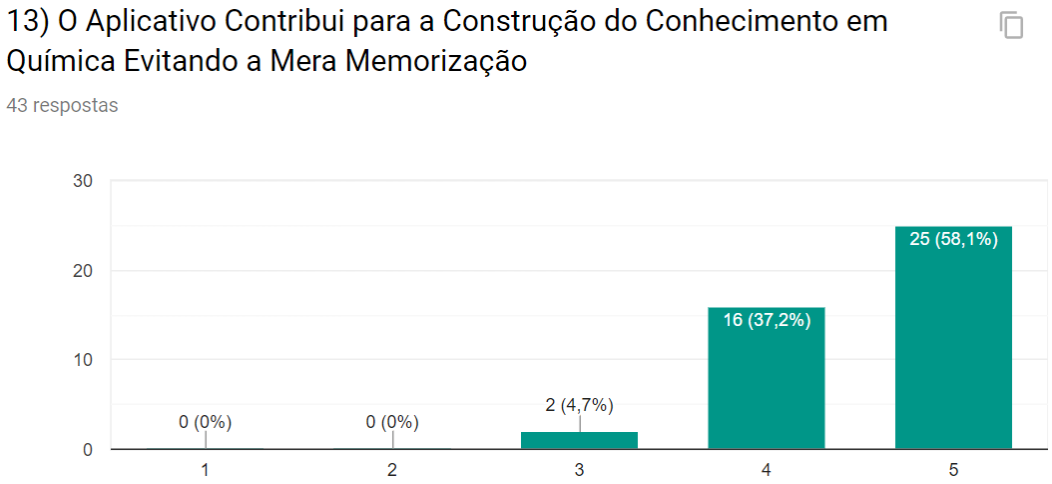
respondeu não ser graduando, graduado ou professor de química, na primeira pergunta.

Figura 113 Décima Quarta Pergunta do Formulário de Configuração Eletrônica



O objetivo da décima quinta pergunta (Figura 114) era avaliar se o aplicativo poderia ser usado como um incentivo a experimentação pelo aluno, e produzir um aprendizado através da exploração, ao invés de memorização. Não houve formulário algum discordando e essa foi uma das respostas com maior concordância, com uma média de 4,53 onde 95,3% dos usuários concordaram com a afirmativa.

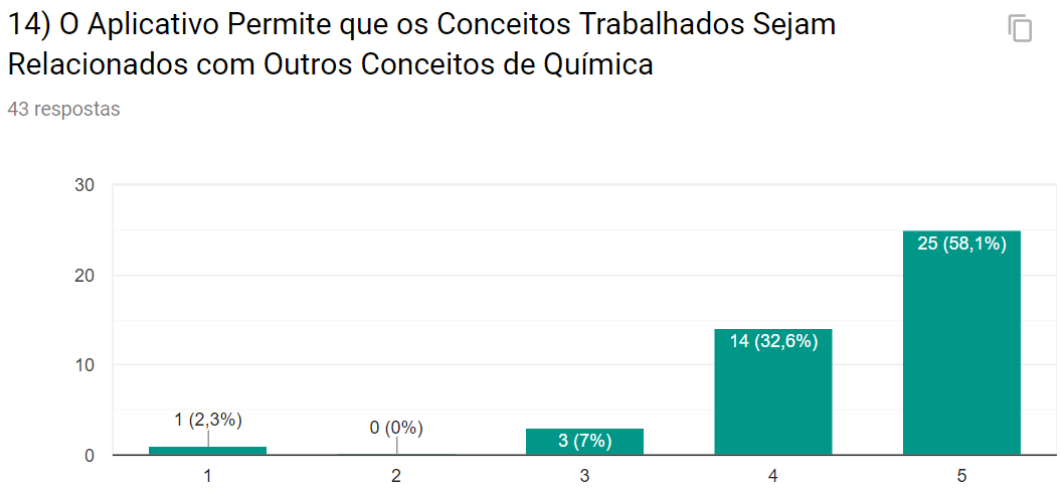
Figura 114 Décima Quinta Pergunta do Formulário de Configuração Eletrônica



O objetivo da décima sexta pergunta (Figura 115) era avaliar se conceitos trabalhados no aplicativo poderiam ser relacionados com outros conceitos de química. Esperava-se uma concordância alta, porque os conceitos de configuração eletrônica

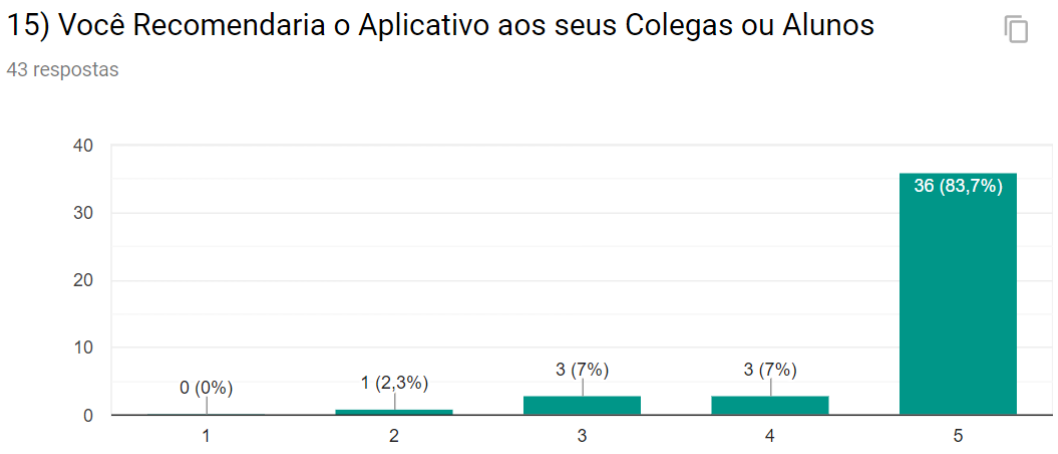
podem ser relacionados com hibridização, por exemplo. A média foi de 4,44 onde 90,7% concordaram com a afirmativa. A única discordância é de um usuário que respondeu não ser graduando, graduado ou professor de química, na primeira pergunta.

Figura 115 Décima Sexta Pergunta do Formulário de Configuração Eletrônica



A última pergunta do formulário (Figura 116) tinha como objetivo validar se os usuários recomendariam o aplicativo. Essa foi uma das respostas com maior concordância, com uma média de 4,72 onde 90,7% afirmaram que recomendariam o aplicativo, uma informação que pode ser relacionada com a segunda pergunta do formulário, que apontava que 25,6% dos usuários souberam do aplicativo através de recomendação de colegas.

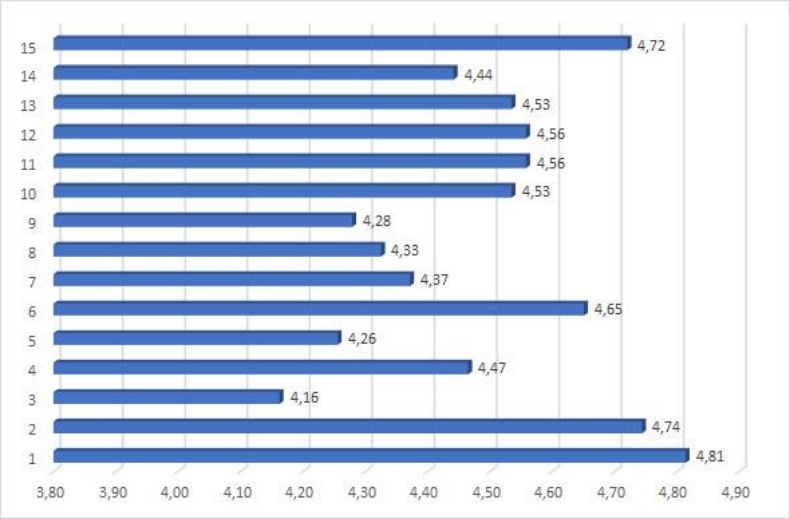
Figura 116 Décima Sétima Pergunta do Formulário de Configuração Eletrônica



A Figura 117 mostra um gráfico com a média das respostas de cada pergunta. Observou-se que todas as afirmativas ficaram com uma média igual ou superior a 4, o que equivale a opção de “Concordância”, segundo a legenda do formulário.

A média total do aplicativo é 4,49, o que se permite afirmar principalmente que o software contribui para a construção do conhecimento em química, apresenta conceitos químicos de forma correta e pode ser usado para despertar o interesse pelo assunto. Além de validar quesitos técnicos como facilidade de uso e instalação, assim como isenção de falhas. De forma geral, pode-se dizer que o aplicativo foi bem aceito e é recomendado por seus usuários.

Figura 117 Média das Respostas dos Formulários



Foram Registradas um Total de 43 Respostas (Média Total: 4,49). Fonte: Próprio Autor

4.3 – *Google Play*: Relatórios dos Aplicativos na Loja

Segundo os relatórios da *Google Play Store*, os aplicativos desenvolvidos foram instalados um total de 5.370 vezes e receberam 54 análises, desde o dia de lançamento até a data 29/08/2019, data na qual registrou-se aproximadamente 1.136 dispositivos com pelo menos um dos softwares instalados. A Figura 118 exibe gráficos de total de instalação de ambos os aplicativos. O aplicativo “Geometria Molecular” foi instalado um total de 1.111 vezes no mês de maio de 2019 (ponto mais alto do gráfico), nesse mesmo período, o aplicativo “Configuração Eletrônica” foi instalado 301 vezes.

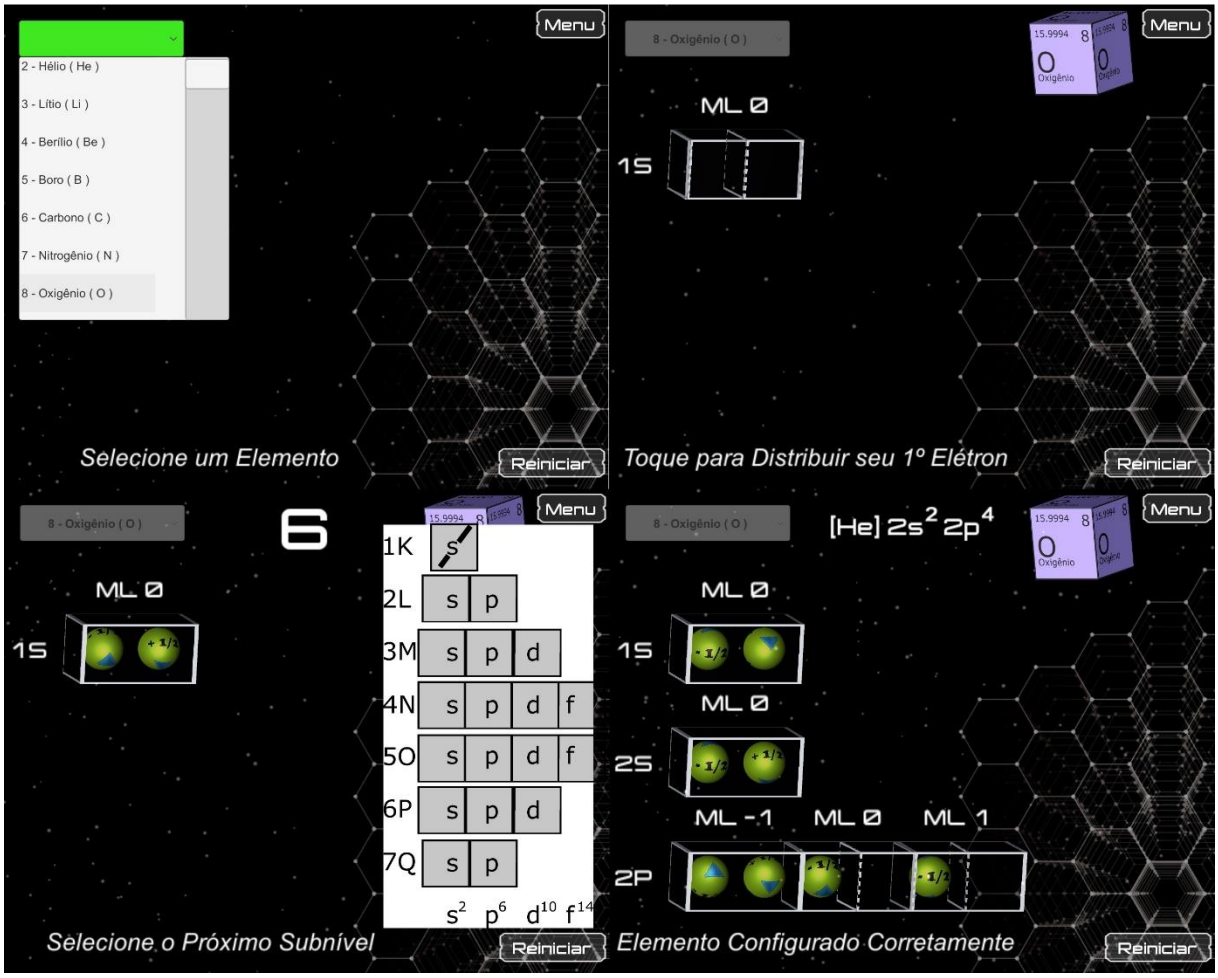
A quantidade de aquisições dos *softwares* na loja da *Microsoft* não foi expressiva: 156 (Geometria Molecular) e 76 (Configuração Eletrônica).

Figura 118 Total de Instalações dos Aplicativos na Google Play Store

Fonte: Próprio Autor

O aplicativo Configuração Eletrônica foi publicado na *Google Play Store* na data 01/04/2018, para o público de 6 países que falam português. Antes da versão oficial, o aplicativo foi disponibilizado apenas para um grupo de testadores internos (versão *alpha*), nesse caso os usuários precisavam ser convidados. Em seguida foi disponibilizado a versão *beta,* onde o aplicativo fica marcado como instável na loja.

Figura 119 Imagens da Cena Principal no Programa Configuração Eletrônica

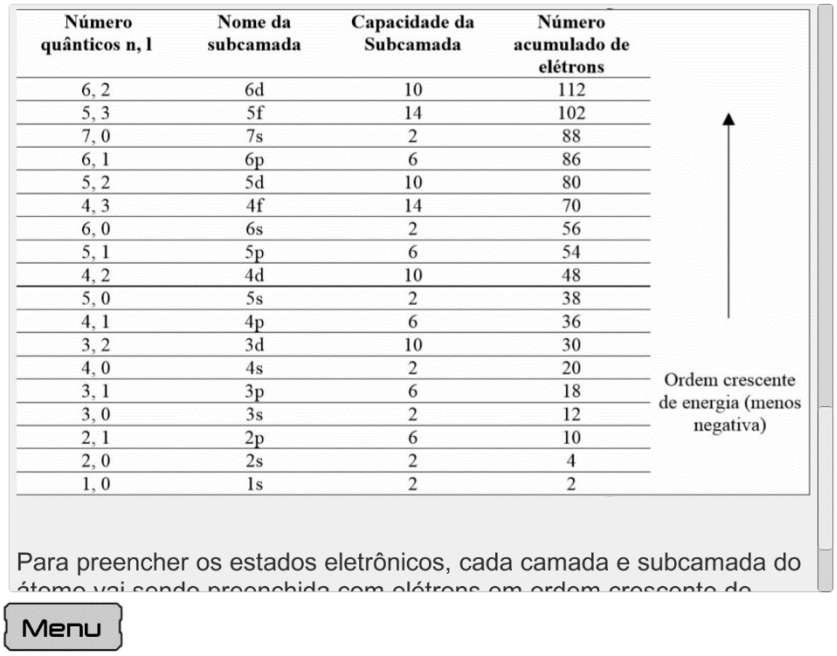


A ordem das imagens é da esquerda para direita, de cima para baixo. Fonte: Próprio Autor

A Figura 119 mostra uma sequência de imagens de utilização do software. Na cena principal, o usuário seleciona um elemento, clica nos espaços vazios para preenchê-los com elétrons até que o subnível esteja completo. A cada subnível finalizado, o usuário precisa selecionar o próximo subnível no Diagrama de *Pauling*. Esse processo se repete até que não reste elétrons para distribuir (imagem direita inferior).

Figura 120 Cena do Menu no Programa Configuração Eletrônica

Figura 121 Cena da Teoria no Programa Configuração Eletrônica



Número quânticos n, l	Nome da subcamada	Capacidade da Subcamada	Número acumulado de elétrons
6, 2	6d	10	112
5, 3	5f	14	102
7, 0	7s	2	88
6, 1	6p	6	86
5, 2	5d	10	80
4, 3	4f	14	70
6, 0	6s	2	56
5, 1	5p	6	54
4, 2	4d	10	48
5, 0	5s	2	38
4, 1	4p	6	36
3, 2	3d	10	30
4, 0	4s	2	20
3, 1	3p	6	18
3, 0	3s	2	12
2, 1	2p	6	10
2, 0	2s	2	4
1, 0	1s	2	2

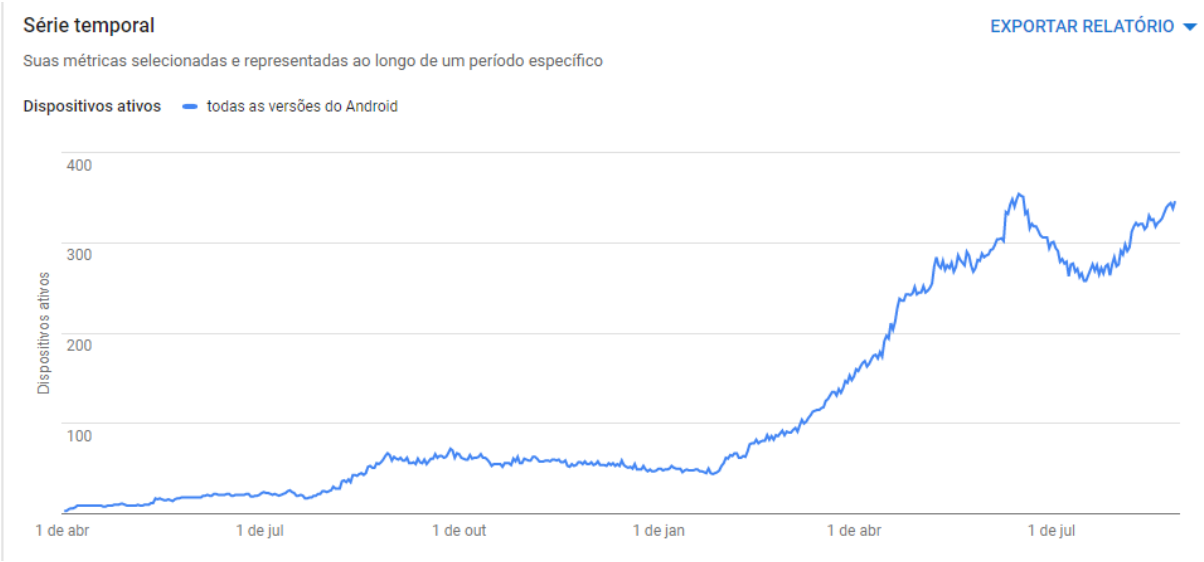
Fonte: Próprio Autor

A Figura 120 mostra duas imagens do menu, um local que permite que o usuário saia do programa, navegue entre as cenas ou veja os créditos. A Figura 121 exibe uma tela da cena de leitura da teoria.

O uso da loja da *Google* facilitou o processo de instalação e controle de versões. Cada atualização no *software* era automaticamente instalada nos dispositivos

que continham uma versão desatualizada do aplicativo. Além disso, a loja fornece *feedback* dos usuários, mensagens de erros aos desenvolvedores e estatísticas. A Figura 122 mostra um gráfico de instalações, em dispositivos ativos[4], do aplicativo ao longo do tempo até a data 29/08/2019. O aplicativo teve 1.890 instalações até essa data, segundo a loja da Google, e está instalado em 390 dispositivos. Houve um período de divulgação do aplicativo em comunidades e grupos de professores de química na rede social do site *Facebook*, que pode ter provocado alguns aumentos no gráfico.

Figura 122 Estatísticas de Instalações do Programa Configuração Eletrônica



Aplicativo teve um total de 1.890 instalações segundo a *Google.* Fonte: Próprio Autor

A Figura 123 mostra a presença do aplicativo na loja virtual da *Google*. Na loja é possível conferir que o aplicativo teve 32 avaliações e tem uma nota média de 4,8 / 5,0. O aplicativo pode ser encontrado usando os termos Configuração Eletrônica no campo de busca da loja, ou através do link[5]

[4] Número de dispositivos Android com o app instalado que estiveram on-line pelo menos uma vez nos últimos 30 dias.

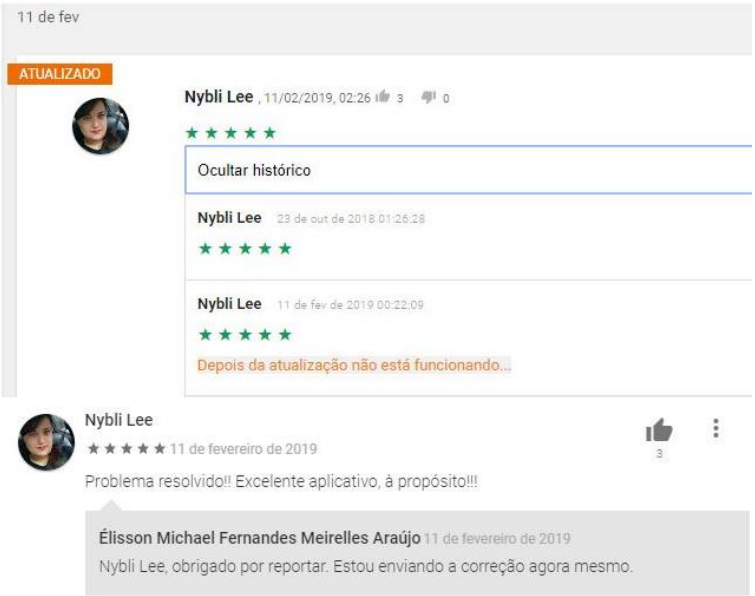
[5] https://play.google.com/store/apps/details?id=com.ElissonMichael.ConfiguracaoEletronica

Figura 123 Aplicativo Configuração Eletrônica na *Google Play Store*

Fonte: Próprio Autor

A Figura 124 mostra exemplo de *feedback* recebido de um usuário, alertando de um problema na mesma data em que a última versão do aplicativo foi lançada.

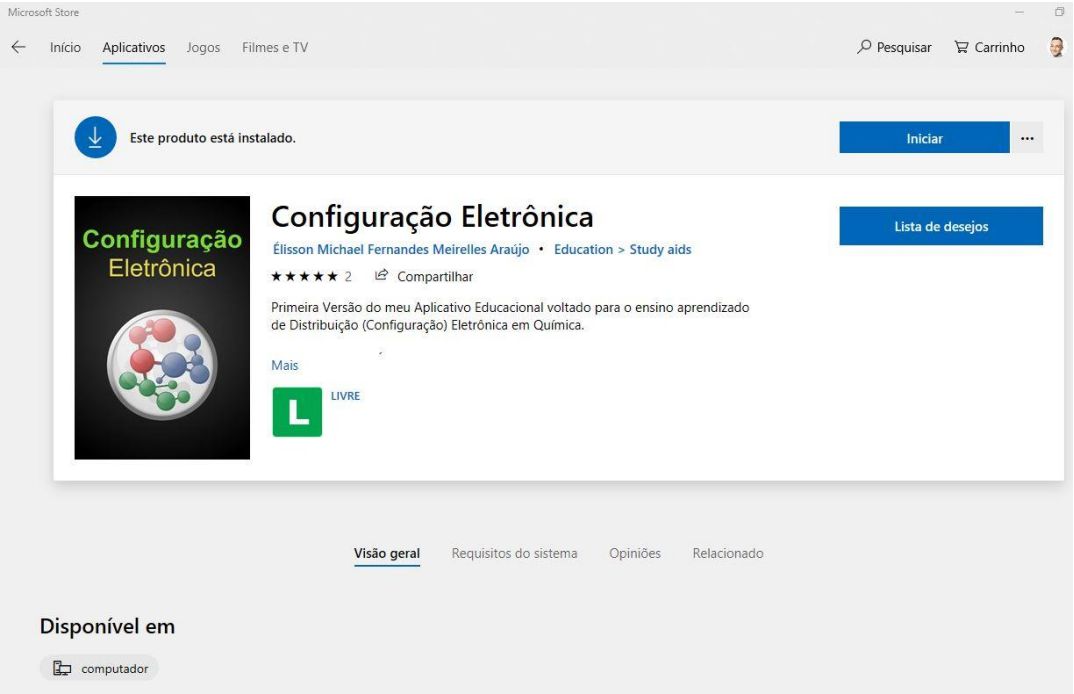
Figura 124 Exemplo de Feedback Recebido Através da Loja da Google



Uma pergunta recorrente era se existia uma versão do aplicativo para o sistema operacional da Apple. Como o aplicativo é gratuito, é inviável assumir o custo de 99 dólares por ano, para que o aplicativo fique disponível para iOS. Como alternativa para usuários que não possuem o sistema Android, o aplicativo também foi disponibilizado na loja da Microsoft, para computadores desktop com o sistema operacional Windows 10. O custo para publicar o aplicativo foi de 25 dólares e 50 reais, na *Google Play Store* e *Microsoft Store,* respectivamente. Não há um custo mensal ou anual envolvido. (MACKENZIE, 2012)

A Figura 125 mostra a presença do aplicativo na loja da Microsoft, o *software* pode ser encontrado pelo campo de busca, usando-se os termos Configuração Eletrônica ou através do link[6].

Figura 125 Aplicativo Configuração Eletrônica na Microsoft Store



Fonte: Próprio Autor

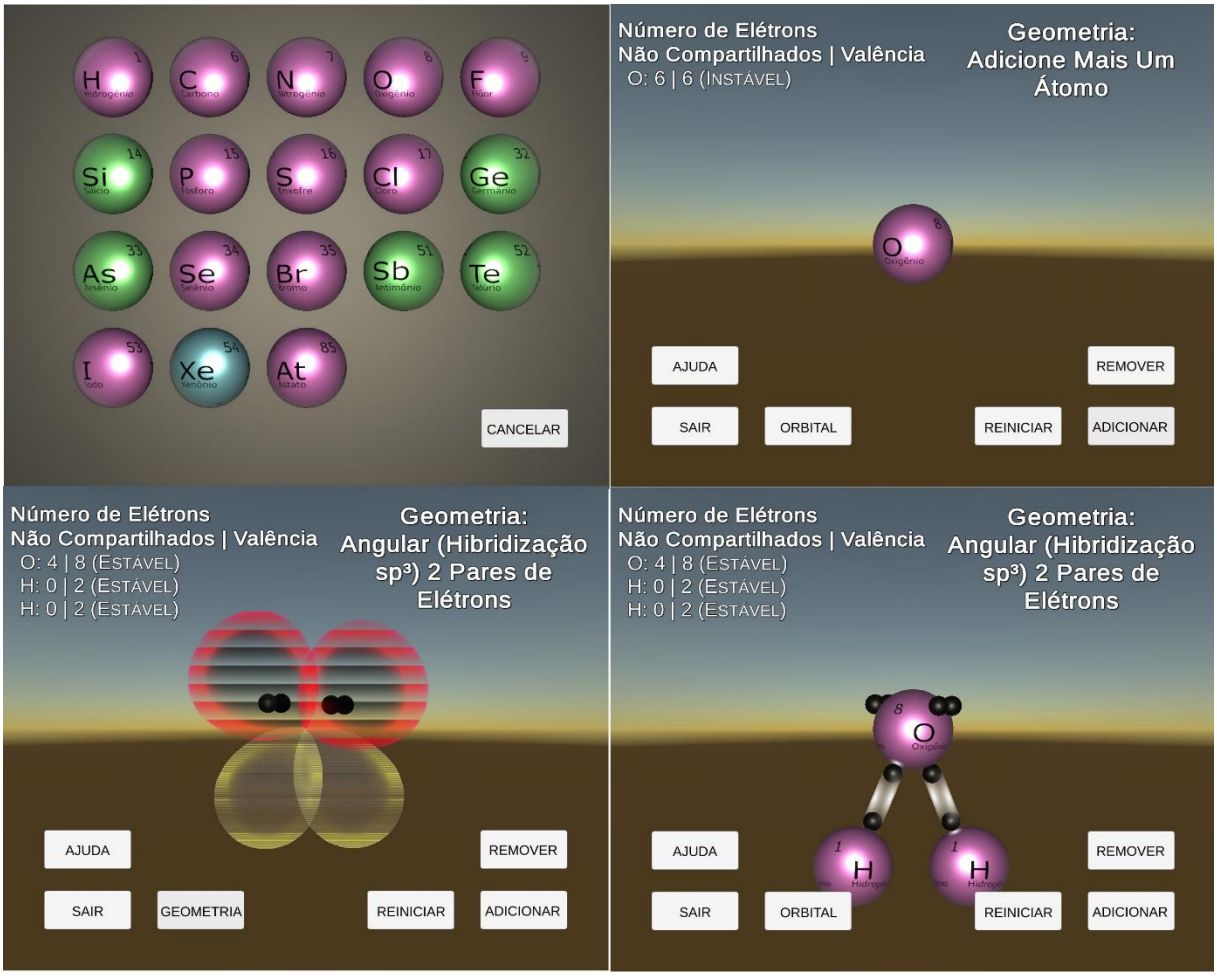
Disponibilizar o aplicativo na loja da Microsoft foi uma forma barata de tentar fornecer uma segunda opção de plataforma para usuários que não possuem

[6] https://www.microsoft.com/pt-br/p/configuracao-eletronica/9nwcqsxqwjd7

dispositivos com o sistema operacional Android instalado e, portanto, não podiam executar a versão *mobile* do aplicativo Geometria Molecular.

A Figura 126 mostra uma sequência de imagens do resultado da utilização do software para construção da molécula H2O. O usuário clica no botão adicionar, que leva a primeira imagem, esquerda superior na figura, o menu de seleção de elementos. Conforme o processo de adicionar elementos se repete, o usuário pode acompanhar as alterações provocadas na geometria da molécula atual. A qualquer momento ele pode remover o último elemento adicionado ou rotacionar a molécula em relação ao eixo X ou Y, arrastando o dedo na horizontal ou vertical, respectivamente. Ele também pode alternar o modo de visualização clicando no botão Orbital, o que resulta na imagem exibida na parte esquerda inferior da Figura 126.

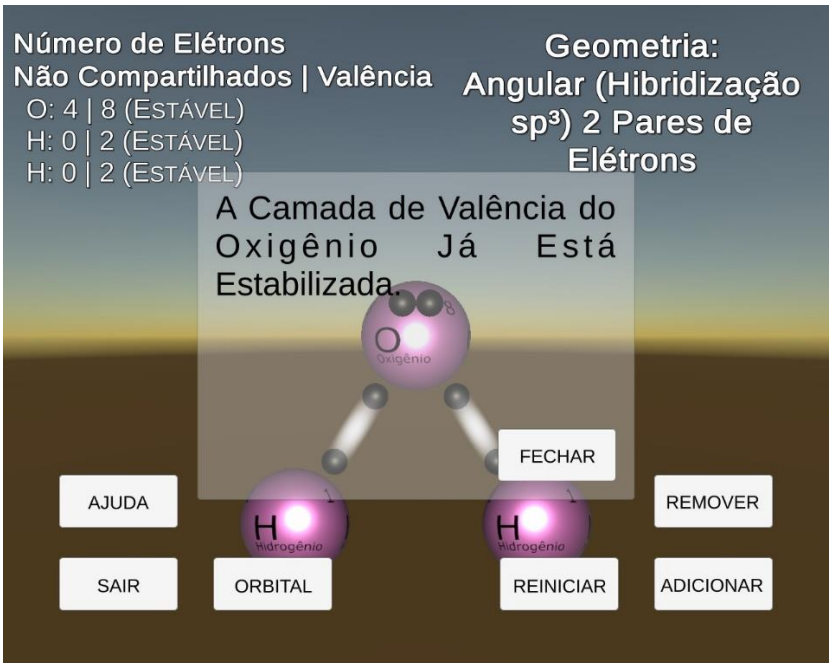
Figura 126 Resultados do Aplicativo Geometria Molecular



Fonte: Próprio Autor

O usuário sempre acompanha a quantidade de elétrons disponíveis na camada de valência de cada átomo, assim como o nome da geometria e o tipo de hibridização da molécula que está construindo no momento. O usuário não consegue adicionar um átomo se o elemento central não possuir elétrons para compartilhar ou se o elemento já estiver com sua camada de valência estabilizada, conforme mostra a Figura 127. O sistema também foi projetado para lidar com exceções à regra do octeto.

Figura 127 Mensagem de Camada Estabilizada

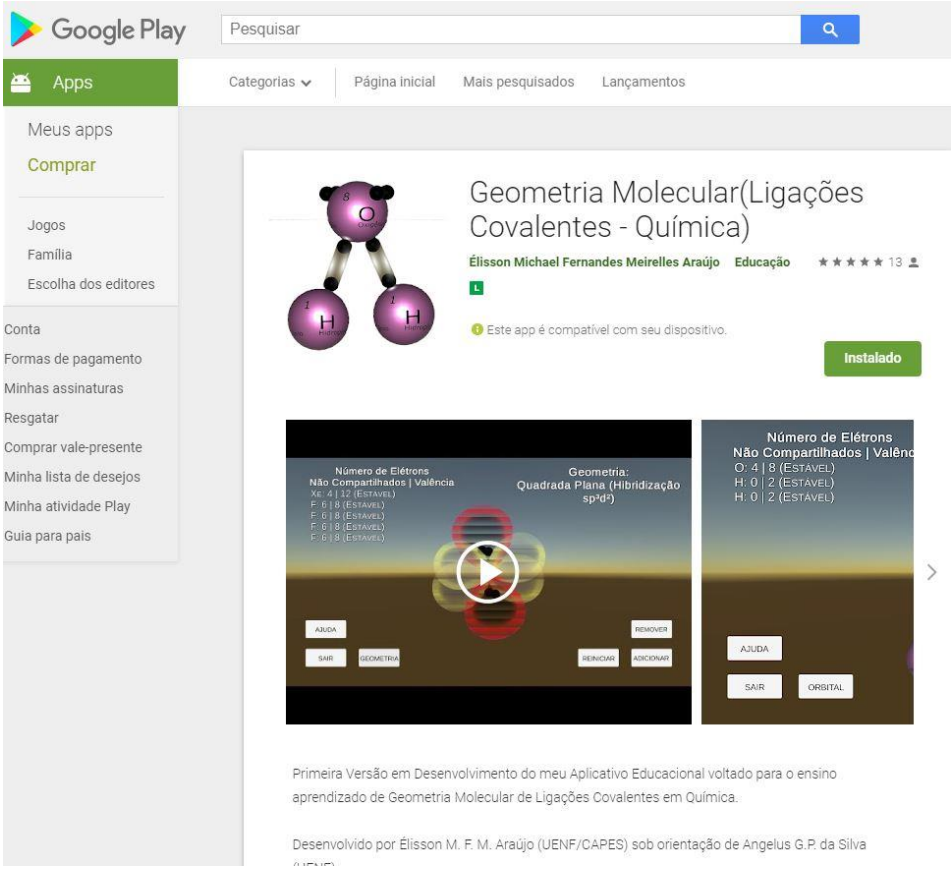


Exemplo de validação que restringe adição de átomo. Fonte: Próprio Autor

O aplicativo Geometria Molecular foi publicado na *Google Play Store* na data 28/12/2018, apenas em sua versão beta, aberta para qualquer usuário com uma conta da *Google*. Nessa versão os usuários não podem avaliar o aplicativo, isso evita que um aplicativo, ainda instável, receba nota baixas ou avaliações negativas. Ainda assim os usuários podem enviar feedback em forma de texto, visível apenas para o desenvolvedor. A Figura 128 mostra a presença do aplicativo na loja da *Google,* ele pode ser encontrado ao se usar os termos Geometria Molecular no campo de busca ou

através do link[7]. Na loja também é possível conferir que o aplicativo teve 22 avaliações e tem uma nota média de 4,4 / 5,0.

Figura 128 Aplicativo Geometria Molecular na Google Play Store



Fonte: Próprio Autor

A versão de produção foi lançada em 22 de janeiro de 2019, período onde houve início da divulgação do aplicativo em grupos de universidade em redes sociais como o *Facebook, LinkedIn, Twitter, Youtube* e *Google Plus* e em aplicativos de mensagens como o *WhatsApp* e *Facebook Messenger*. A Figura 129 mostra um gráfico de instalações, em dispositivos ativos, do aplicativo ao longo do tempo, até a data 29/08/2019. A loja da *Google* registrou um total de 3.480 instalações do aplicativo até essa data e a quantidade de dispositivos ativos é de 790.

[7] https://play.google.com/store/apps/details?id=com.ElissonMichael.LigacoesCovalentes

Figura 129 Estatísticas de Instalações do Programa Geometria Molecular

Aplicativo teve um total de 3.480 instalações segundo a *Google*. Fonte: Próprio Autor

A Figura 130 mostra exemplos de opiniões que foram colhidas através da ferramenta de análise de aplicativos da *Google Play Store.*

Figura 130 Exemplo de Feedback Recebido Através da Loja da Google

Fonte: Próprio Autor

A Figura 131 exibe a quantidade de comentários, curtidas e compartilhamentos das duas postagens que tiveram maior engajamento nas redes sociais em geral. Muito feedback foi recebido e essas postagens foram as que mais influenciaram o gráfico da Figura 129, provocando um aumento na quantidade de downloads do aplicativo.

Figura 131 Publicações que Renderam Maior Engajamento

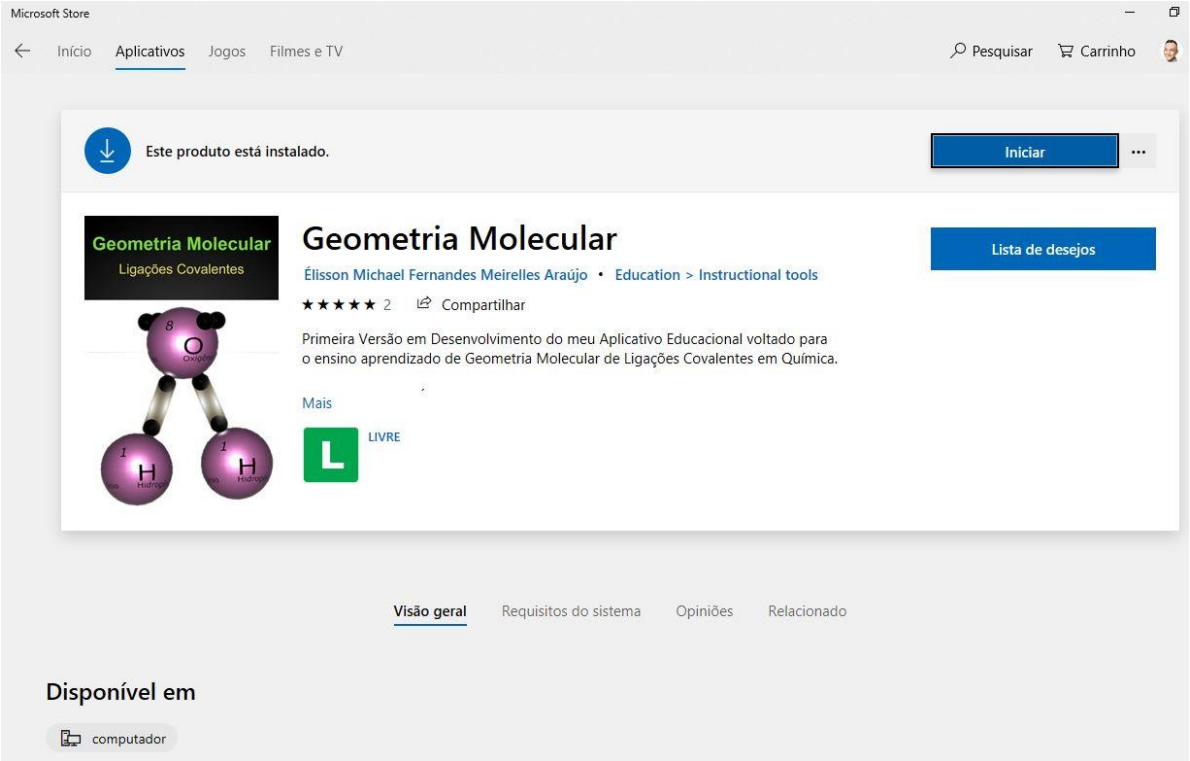
Fonte: Próprio Autor

Assim como ocorrido no aplicativo de Configuração Eletrônica, alguns usuários não possuíam dispositivos com sistema operacional *Android* e perguntavam se existia uma versão para iOS. Publicar na loja da Apple exigia o pagamento de uma anuidade no valor de 99 dólares, um motivo que inibe a publicação de aplicativos gratuitos para *iPhone* e *iPad*. (MACKENZIE, 2012)

Por isso, a segunda opção de sistema operacional escolhida foi a *Microsoft*, além de ser mais barato e não exigir anuidade, foi uma escolha que permitiu que o programa fosse utilizado em desktops, além de fornecer uma alternativa para usuários sem dispositivos com *Android.* Uma oportunidade de resolver dois problemas que surgiram durante a fase de divulgação. A Figura 132 mostra a presença do aplicativo na

Microsoft Store, que pode ser encontrado usando os termos Geometria Molecular no campo de busca ou pelo link[8].

Figura 132 Aplicativo Geometria Molecular na Microsoft Store



Fonte: Próprio Autor

[8] https://www.microsoft.com/pt-br/p/geometria-molecular/9pk3gthv5m0j

CAPÍTULO 5

CONCLUSÃO

Nesse trabalho, foram desenvolvidos dois aplicativos denominados Configuração Eletrônica e Geometria Molecular. Tecnicamente foram criados quatro softwares, porque ambos os aplicativos móveis foram usados como base para as versões desktop que foram desenvolvidas posteriormente. As versões desktop e mobile de ambos os programas existem individualmente, ou seja, estão isoladas uma das outras, de forma que, alterações no aplicativo móvel não interferem no programa desktop, mas por estarem em um mesmo repositório de controle de versão, podem compartilhar uma mesma base de código, quando isso oferecer alguma vantagem.

A ferramenta utilizada para desenvolvimento dos aplicativos (o motor de jogo *Unity*) é multiplataforma, o que facilita a conversão para outras plataformas como o sistema operacional da Apple (iOS) ou aplicações para sistema web, no futuro, se houver interesse.

Ambos os aplicativos foram idealizados como ferramentas de apoio ao aprendizado, servindo da forma como os usuários decidirem: professores em sala de aula, alunos em seus estudos extraclasse ou de qualquer outra forma. No aplicativo Configuração Eletrônica, o usuário seleciona um elemento e precisa distribuir os elétrons nas camadas, de acordo com a ordem energética. O usuário precisa selecionar as próximas camadas em um diagrama de Pauling, sempre que terminar de preencher os elétrons em um subnível. Já no aplicativo Geometria Molecular, o usuário precisa adicionar elementos em um átomo central e fazer ligações covalentes dessa forma, sempre monitorando a quantidade de elétrons disponíveis na camada de valência. A cada átomo adicionando, ou removido, o usuário pode observar e rotacionar a estrutura atual da molécula, assim como visualizar os orbitais híbridos, em três dimensões, do elemento central.

Os aplicativos foram divulgados em grupos de química, em geral, em redes sociais e em um blog de química, que recebe em média 200 visualizações diariamente. Como os aplicativos foram desenvolvidos fazendo uso de ferramentas de

controle de versão, seguindo modelos tradicionais de engenharia de software, princípios de *design* orientado a objetos e boas práticas de programação, acredita-se que ele tem uma base sólida que permite que novas funcionalidades e manutenção possa ser realizada a longo prazo. Evitando-se cair no problema tradicional de desenvolvimento de um software que se torna impossível de manter ao longo do tempo, pois se torna cada vez mais difícil de ser trabalhado conforme sua base de código cresce.

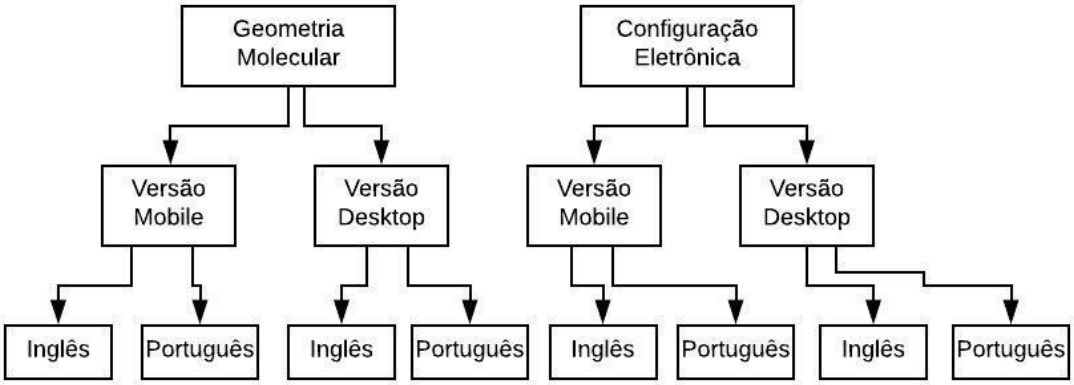
Após a publicação dos aplicativos foram elaborados dois formulários de avaliação, um para cada *software*, contendo 15 afirmações. Os usuários acessaram o formulário através de um link disponibilizado temporariamente na tela inicial dos aplicativos. O usuário manifestou a concordância com cada afirmação, assinalando um número em uma escala de 1-5, sendo que o número um (1) significa “discordância plena” e o número cinco (5) “concordância plena”. Todas as médias desses valores ficaram acima de 4, para ambos os aplicativos, em um total de 115 respostas, permitindo-se afirmar que os softwares foram bem aceitos. A média total de concordância com as afirmativas do aplicativo Geometria Molecular foi 4,34 e o do Configuração Eletrônica foi 4,49. Portanto, concluiu-se que os aplicativos:

1. Possuem nome coerente;
2. São fáceis de instalar;
3. Estiveram isentos de falhas durante sua utilização;
4. São fáceis de serem utilizados;
5. Não apresentam propagandas durante seu uso;
6. Podem ser usados para despertar o interesse pelo assunto;
7. Possuem interface fácil de ser entendida;
8. São visualmente atraentes;
9. Possuem um tempo de resposta adequado nas operações interativas;
10. Têm cores e detalhes favoráveis à leitura em uma tela pequena;
11. Utilizam a linguagem química de maneira correta;
12. Apresentam os conceitos químicos de forma correta;
13. Contribuem para construção do conhecimento em química;
14. Permitem que os conceitos sejam relacionados com outros conceitos;
15. São recomendados por seus usuários.

5.1 – Trabalhos Futuros

Como trabalhos futuros acredita-se que os aplicativos possam ser aprimorados de acordo com as necessidades e *feedbacks* apontados pelo uso das ferramentas. A lista abaixo contém sugestões dos autores e alguns professores de química que entraram em contato com algumas análises. As sugestões abaixo estão separadas por aplicativos, mas existe uma melhoria que é válida para os dois: Traduzir e lançar versões em inglês de ambos os aplicativos, para se alcançar um público de usuários maior. É preciso tomar cuidado para não ficar em uma situação complexa, onde é preciso gerenciar oito aplicativos diferentes, conforme visualizado na Figura 133, ao se trabalhar nessa melhoria.

Figura 133 Cenário onde Traduções são Versões Independentes



Fonte: Próprio Autor

Sugestões de trabalhos futuros específicos:

- **Configuração Eletrônica**
  - Adicionar a opção de distribuir apenas a camada de valência. A intenção é agilizar o processo da configuração eletrônica de elementos com número atômico alto.
  - Melhorar os aspectos visuais da parte de seleção de camada. Esse requisito não estava nos planos iniciais, foi implementado

rapidamente e nunca deixou de ter uma aparência de protótipo que não está de acordo com a qualidade do restante do aplicativo.

  - Mostrar os orbitais sendo preenchidos. A intenção é relacionar melhor o conteúdo dos dois aplicativos. A configuração poderia mostrar objetos tridimensionais contendo os orbitais, que mudariam de cor a cada elétron distribuído.
  - Adicionar a regra de *Hund* na cena de teoria e atualizar a referência.

- **Geometria Molecular**
  - Adicionar uma lista com moléculas conhecidas, de modo que, o usuário possa alternar rapidamente entre elas sem precisar adicionar e remover átomo por átomo;
  - Adicionar Orbitais de Ligações Covalentes do tipo Sigma e Pi;
  - Adicionar mais alguns elementos, como Boro e Berílio;
  - Implementar exceções à Regra do Octeto que possam estar faltando;
  - Possibilitar a alteração do átomo central, de modo a permitir a construção de moléculas mais complexas.

CAPÍTULO 6

REFERÊNCIAS BIBLIOGRÁFICAS

©STATCOUNTER. **Desktop vs Mobile vs Tablet vs Console Market Share Brazil | StatCounter Global Stats**. Disponível em: <http://gs.statcounter.com/platform-market-share/all/brazil/#monthly-201702-201902-bar>. Acesso em: 24 mar. 2019a.

_____. **Desktop vs Mobile vs Tablet vs Console Market Share Worldwide | StatCounter Global Stats**. Disponível em: <http://gs.statcounter.com/platform-market-share#monthly-201702-201902-bar>. Acesso em: 24 mar. 2019b.

_____. **FAQ | StatCounter Global Stats**. Disponível em: <http://gs.statcounter.com/faq#methodology>. Acesso em: 24 mar. 2019.

_____. **Mobile Operating System Market Share Brazil | StatCounter Global Stats**. Disponível em: <http://gs.statcounter.com/os-market-share/mobile/brazil/#monthly-201702-201902-bar>. Acesso em: 24 mar. 2019c.

_____. **Mobile Operating System Market Share Worldwide | StatCounter Global Stats**. Disponível em: <http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201702-201902-bar>. Acesso em: 24 mar. 2019d.

©STATISTA. **Number of smartphone users in the U.S. 2010-2022 | Statista**. Disponível em: <https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/>. Acesso em: 24 mar. 2019.

101 EDU, Inc. **Chem101 – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=bonds.one_o_one.bonds_android>. Acesso em: 18 mar. 2019.

57DERAJAT. **Periodic Table – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=litude.radian.kimia2>. Acesso em: 18 mar. 2019.

ALLARDICE, Simon. **Exploring Object Oriented Analysis Design and Development**. Disponível em: <https://www.linkedin.com/learning/programming-foundations-object-oriented-design/why-we-use-object-orientation>. Acesso em: 28 abr. 2018a.

_____. **What is a class?** Disponível em: <https://www.linkedin.com/learning/programming-foundations-object-oriented-design/what-is-a-class>. Acesso em: 30 abr. 2018b.

_____. **What is abstraction?** Disponível em: <https://www.linkedin.com/learning/programming-foundations-object-oriented-design/what-is-abstraction>. Acesso em: 28 abr. 2018c.

_____. **What is an object?** Disponível em: <https://www.linkedin.com/learning/programming-foundations-object-oriented-design/what-is-an-object>. Acesso em: 29 abr. 2018d.

_____. **What is encapsulation?** Disponível em: <https://www.linkedin.com/learning/programming-foundations-object-oriented-design/what-is-encapsulation>. Acesso em: 28 abr. 2018e.

_____. **What is inheritance?** Disponível em: <https://www.linkedin.com/learning/programming-foundations-object-oriented-design/what-is-inheritance>. Acesso em: 27 jul. 2018f.

_____. **What is polymorphism?** Disponível em: <https://www.linkedin.com/learning/programming-foundations-object-oriented-design/what-is-polymorphism>. Acesso em: 28 maio 2018g.

_____. **Why we use object-orientation.** Disponível em: <https://www.linkedin.com/learning/programming-foundations-object-oriented-design/why-we-use-object-orientation>. Acesso em: 28 abr. 2018h.

ARAÚJO, Élisson. Desenvolvimento de um Sistema de Medições Livre de Marcadores Utilizando Sensores de Profundidade. **UENF**, p. 55, 2015.

BIANCO, André A G; MELONI, Reginaldo A. O Conhecimento Escolar

ConCeitos CientífiCos em Destaque. v. 41, p. 148–155, 2019. Disponível em: <http://dx.doi.org/10.21577/0104-8899.20160156>.

BLENDER, Foundation. **About — blender.org**. Disponível em: <https://www.blender.org/about/>. Acesso em: 1 mar. 2019.

BLOW, Jonathan. **Game Development**. [S.l: s.n.], 2004. v. 1.

BROOKSHEAR, J. Glenn; SMITH, David T.; BRYLOW, Dennis. Programação orientada a objetos. In: BOOKMAN (Org.). . **Ciência da Comput. Uma Visão Abrangente**. 11. ed. [S.l.]: Bookman, 2013. p. 248–255.

BROOKSHEAR, J. Glenn; SMITH, David T.; DENNIS BRYLOW. Paradigmas de Programação. In: BOOKMAN (Org.). . **Ciência da Comput. Uma Visão Abrangente**. 11. ed. [S.l.]: Bookman, 2013. p. 213–217.

CALLISTER, William; RETHWISCH, David. **Ciência e Engenharia de Materiais**. 7. ed. [S.l.]: John Wiley & Sons Ltd., 2007.

CARPENTER, M. **Unity in action**. [S.l: s.n.], 2015. v. 76. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/6995941>.

CHACON, Scott; STRAUB, Ben. **Git - Livro**. Disponível em: <https://git-scm.com/book/pt-br/v2>. Acesso em: 21 out. 2018.

COSTA, A K P et al. Utilização De Jogos Didáticos Para O Ensino De Química : Up and Down Chemical. **IX Congresso de Iniciação Científica do IFRN**, p. 1446–1454, 2013. Disponível em: <http://www2.ifrn.edu.br/ocs/index.php/congic/ix/paper/viewFile/807/344>. Acesso em: 18 maio 2017.

CREIGHTON, Ryan Henson. **Unity 3D Game Development by Example**. [S.l: s.n.], 2010.

CUZIAC, Mihai. **Naked Chem 2 - Microsoft Store**. Disponível em: <https://www.microsoft.com/pt-br/p/naked-chem-2/9nblggh4wspc?activetab=pivot%3Aoverviewtab#>. Acesso em: 18 mar. 2019.

D SOLUTIONS. **Molecular 3D – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=com.hkdilan.molecularstructure>. Acesso em: 18 mar. 2019.

DALUZ, S. **Tabela Periódica dos Elementos - Modern PTE – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=br.com.daluz.android.apps.modernpte>. Acesso em: 18 mar. 2019.

DEMIRBILEK, Muhammet. Investigating Attitudes of Adult Educators towards Educational Mobile Media and Games in Eight European Countries. **Journal of Information Technology Education: Research**, v. 9, p. 235–247, 2010. Disponível em: <https://www.informingscience.org/Publications/1327>.

DEWITT, Tyler. **VSEPR Theory: Octahedral Family - YouTube**. Disponível em: <https://www.youtube.com/watch?v=Qcy-TjJ10xk>. Acesso em: 12 mar. 2019a.

_____. **VSEPR Theory: Trigonal Bipyramidal Family - YouTube**. Disponível em: <https://www.youtube.com/watch?v=cdo6FtSU_k8&t=1s>. Acesso em: 12 mar. 2019b.

DIAS, Raphael. **Game Engine: o que é, para que serve e como escolher a sua**. Disponível em: <https://producaodejogos.com/game-engine/>. Acesso em: 25 jul. 2018.

DOUGLAS, Eric. **Pro-Git**. [S.l: s.n.], 2014.

FIDI, Patrícia; LT, Paul. **Electron Orbitals**. Disponível em: <https://pt.wikipedia.org/wiki/Ficheiro:Electron_orbitals.svg>. Acesso em: 17 fev. 2019.

FOURELEM. **Ball & Stick - Microsoft Store**. Disponível em: <https://www.microsoft.com/pt-br/p/ball-stick/9nblggh4s694?activetab=pivot:overviewtab>. Acesso em: 18 mar. 2019.

FOWLER, Martin et al. **Refactoring - Improving the Design of Existing Code**. Massachusetts: Addison Wesley, 1999. Disponível em:

<https://martinfowler.com/books/refactoring.html>. Acesso em: 27 jul. 2018.

_____. **UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos**. 3. ed. Massachusetts: Bookman, 2005.

GALEMBECK, E. **Moléculas – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=lte.ib.unicamp.br.moleculas>. Acesso em: 18 mar. 2019.

GAMMA, Erich et al. **Padrões de Projeto Soluções reutilizáveis de software orientado a objetos**. São Paulo: Bookman, 2000.

GOUVEIA, David. **Game Development with Unity**. Disponível em: <https://pt.slideshare.net/davidluzgouveia/game-development-with-unity?from_action=save>. Acesso em: 25 jul. 2018.

GREGORY, Jason. **Game Engine Architecture**. Boca Raton: Taylor and Francis Group, LLC, 2009.

GUY, Geek. **Electron Configuration – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=lexcrd.electron>. Acesso em: 18 mar. 2019.

HEBERT, Nicholas. **Chemistry: Geometries - Microsoft Store**. Disponível em: <https://www.microsoft.com/pt-br/p/chemistry-geometries/9nblggh0j1vb?activetab=pivot:overviewtab#>. Acesso em: 18 mar. 2019.

JOHNSON, B. **Electron Orbitals – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=com.gputreats.orbitalexplorer>. Acesso em: 18 mar. 2019.

JÚNIOR, José; BARBOSA, Francisco; JUNIOR, Antonio. **POLARÍMETRO VIRTUAL: DESENVOLVIMENTO, UTILIZAÇÃO E AVALIAÇÃO DE UM SOFTWARE EDUCACIONAL**. **Quim. Nova**. [S.l: s.n.], 2012. Disponível em: <http://www.labvirtq.fe.usp.br/indice.asp,>.

KASINATHAN, Vinothini; MUSTAPHA, Aida; SAMSUDIN, Noor Azah. Keeping Up with Healthcare Trends: IcHeart as a Medication Management Application. **IOP**

Conference Series: Materials Science and Engineering, v. 160, p. 012109, 2016. Disponível em: <http://stacks.iop.org/1757-899X/160/i=1/a=012109?key=crossref.8f6518d602ef7fa91e0502596024a2ae>.

LABORATÓRIO DIDÁTICO DE QUÍMICA - LADQUIM. **Construa Seu Átomo! – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=br.com.danieljunior.bohrapp.db>. Acesso em: 18 mar. 2019.

LAKE, S. **Molecules App Store**. Disponível em: <https://itunes.apple.com/us/app/molecules/id284943090?mt=8&utm_campaign=elearningindustry.com&utm_source=%2F24-free-chemistry-ipad-apps-for-students&utm_medium=link>. Acesso em: 18 mar. 2019.

LDMD - UFSJ. **LDMD Química: Aprendendo Geometria Molecular – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=air.LdmdGeometriaMolecular>. Acesso em: 18 mar. 2019.

LINIETSKY, Juan; MANZUR, Ariel. **Introduction — Godot Engine latest documentation**. Disponível em: <http://docs.godotengine.org/en/3.0/about/introduction.html>. Acesso em: 25 jul. 2018.

MACKENZIE, Tim. **App store fees, percentagTes, and payouts: What developers need to know - TechRepublic**. Disponível em: <https://www.techrepublic.com/blog/software-engineer/app-store-fees-percentages-and-payouts-what-developers-need-to-know/>. Acesso em: 26 fev. 2019.

MARTIN, Robert. **Clean Code: A Handbook of Agile Software Craftsmanship**. [S.l.]: Prentice Hall, 2009. Disponível em: <www.it-ebooks.info>. Acesso em: 1 fev. 2019.

METZ, Sandi. **Practical Object-Oriented Design in Ruby : An Agile Primer**. Indiana: Pearson Education, Inc., 2013.

MICROSOFT, Store. **Get Paint 3D - Microsoft Store**. Disponível em:

<https://www.microsoft.com/en-us/p/paint-3d/9nblggh5fv99?activetab=pivot:overviewtab>. Acesso em: 1 mar. 2019.

MILETTO, Evandro Manara; BERTAGNOLLI, Silvia de Castro. **Desenvolvimento de Software II Introdução ao Desenvolvimento Web com HTML, CSS, Javascript e PHP**. [S.l.]: Bookman, 2014.

MOBI, Z. **Shapes of Molecules – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=zanojmobiapps.shapesofmolecules>. Acesso em: 18 mar. 2019.

MOLINARI, Willian. Ruby on Rails. **Casadocodigo**, 2012.

NETO, J F B; DA FONSECA, F S. Jogos educativos em dispositivos móveis como auxílio ao ensino da matemática. **Renote**, n. Elias 2011, p. 1–10, 2013. Disponível em: <http://www.seer.ufrgs.br/renote/article/download/41623/26403>.

NOGUEIRA, Filipe B et al. Quizmico : software educativo para o ensino de química . Introdução O Quizmico pode ser jogado em dois. n. Ic, p. 33, 2008. Disponível em: <http://sec.sbq.org.br/cdrom/33ra/resumos/T0074-1.pdf>. Acesso em: 15 maio 2017.

NORTON, Terry. **Learning C# by Developing Games with Unity 3D**. [S.l: s.n.], 2013. Disponível em: <https://www.packtpub.com/learning-csharp-by-developing-games-with-unity-3d/book%5Cnhttp://bookzz.org/book/2294472/744f5b>.

NOVELLINO, B. **Electron – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=electro.n.zero>. Acesso em: 18 mar. 2019.

PASSOS, Eb; JR, Jr Da Silva. Tutorial: Desenvolvimento de jogos com unity 3d. **… Entertainment. Rio de …**, p. 1–30, 2009. Disponível em: <http://sbgames.org/papers/sbgames09/computing/tutorialComputing2.pdf>.

PORTUGAL, Sílvia. Mensagens veiculadas nos conteúdos dos Videojogos : o caso dos The Sims 2 e os estereótipos de género. p. 120, 2009.

PRESSMAN, Roger S.; MAXIM, Bruno R. **Engenharia de Software: uma**

abordagem profissional. 8. ed. Porto Alegre: Bookman, 2016.

RALLS, K. M.; COURTNEY, T. H.; WULFF, J. **Introduction to Materials Science and Engineering**. [S.l: s.n.], 1976.

SCIENCEHIGH, G. **Electronic Structure 4.0 – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=com.sciencehighgames.electronstructure.free>. Acesso em: 18 mar. 2019.

SHITSUKA, Dorlivete Moreira et al. Aprendizagem ativa de programação em turmas de engenharia: uma pesquisa-ação. **Research, Society and Development**, v. 8, n. 3, p. 1–18, 2019.

SILVA, MONIELLE GOMES DA. **METODOLOGIA PARA AVALIAÇÃO DE APLICATIVOS EDUCACIONAIS DE MATEMÁTICA PARA O ENSINO MÉDIO**. 2015. 58 f. Instituto Federal de Educação, Ciência e Tecnologia Fluminense, 2015.

SMITH, Matt; QUEIROZ, Chico. **Unity 5.x Cookbook**. [S.l.]: Packt Publishing Ltd., 2015.

SOUSA, C. **InterÁtomo – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=app.interAtomo>. Acesso em: 18 mar. 2019.

SOUZA, Lucas. **Ruby Aprenda a Programar na Linguagem Mais Divertida**. Softcover ed. São Paulo: Casa do Código, 2013.

SPONHOLTZ PRODUCTIONS. **Hybridization Theory - YouTube**. Disponível em: <https://www.youtube.com/watch?v=J9oNPj_GAws&t=607s>. Acesso em: 23 jan. 2019.

STEIN, Stephanie Jedoz et al. Proposta para a utilização de um aplicativo no ensino de química para alunos do ensino médio integrado ao técnico em química Proposal for the use of an application in chemistry teaching for high school students integrated with chemistry technician. **Scientia Naturalis**, v. 1, n. 4, p. 165–173, 2019. Disponível em: <http://revistas.ufac.br/revista/index.php/SciNat>.

SUPPLY, Carolina B. **Building Atoms, Ions, and Isotopes HD Lite on the App Store**. Disponível em: <https://itunes.apple.com/us/app/building-atoms-ions-isotopes/id437001161?mt=8&utm_campaign=elearningindustry.com&utm_source=%2F24-free-chemistry-ipad-apps-for-students&utm_medium=link>. Acesso em: 18 mar. 2019.

TECHNOLOGIES, Unity. **Unity - Manual: Unity User Manual (2018.2)**. Disponível em: <https://docs.unity3d.com/Manual/index.html>. Acesso em: 28 jul. 2018.

TECHNOLOGIES UNITY. **Unity - Scripting API: Vector3**. Disponível em: <https://docs.unity3d.com/ScriptReference/Vector3.html>. Acesso em: 4 mar. 2019.

TEPLUKHIN, A. **Molecular Constructor – Apps no Google Play**. Disponível em: <https://play.google.com/store/apps/details?id=com.alextepl.molconstr>. Acesso em: 18 mar. 2019.

TRO, Nivaldo J. **Química Uma Abordagem Molecular: Volume 1**. 3. ed. Rio de Janeiro: LTC — Livros Técnicos e Científicos Editora Ltda, 2017.

UNITY. **Unity - Manual: GameObject**. Disponível em: <https://docs.unity3d.com/Manual/class-GameObject.html>. Acesso em: 21 fev. 2018.

UNITY MANUAL. **Unity User Manual (5.6)**. Disponível em: <https://docs.unity3d.com/Manual/>. Acesso em: 30 jul. 2018.

UNITY TECHNOLOGIES. **Unity User Manual (2017.3)**. Disponível em: <https://docs.unity3d.com/Manual/index.html>. Acesso em: 21 fev. 2018.

VOLLHARDT, Peter; SCHORE, Neil E. **Química Orgânica: Estrutura e Função**. 6. ed. [S.l.]: Bookman, 2013.

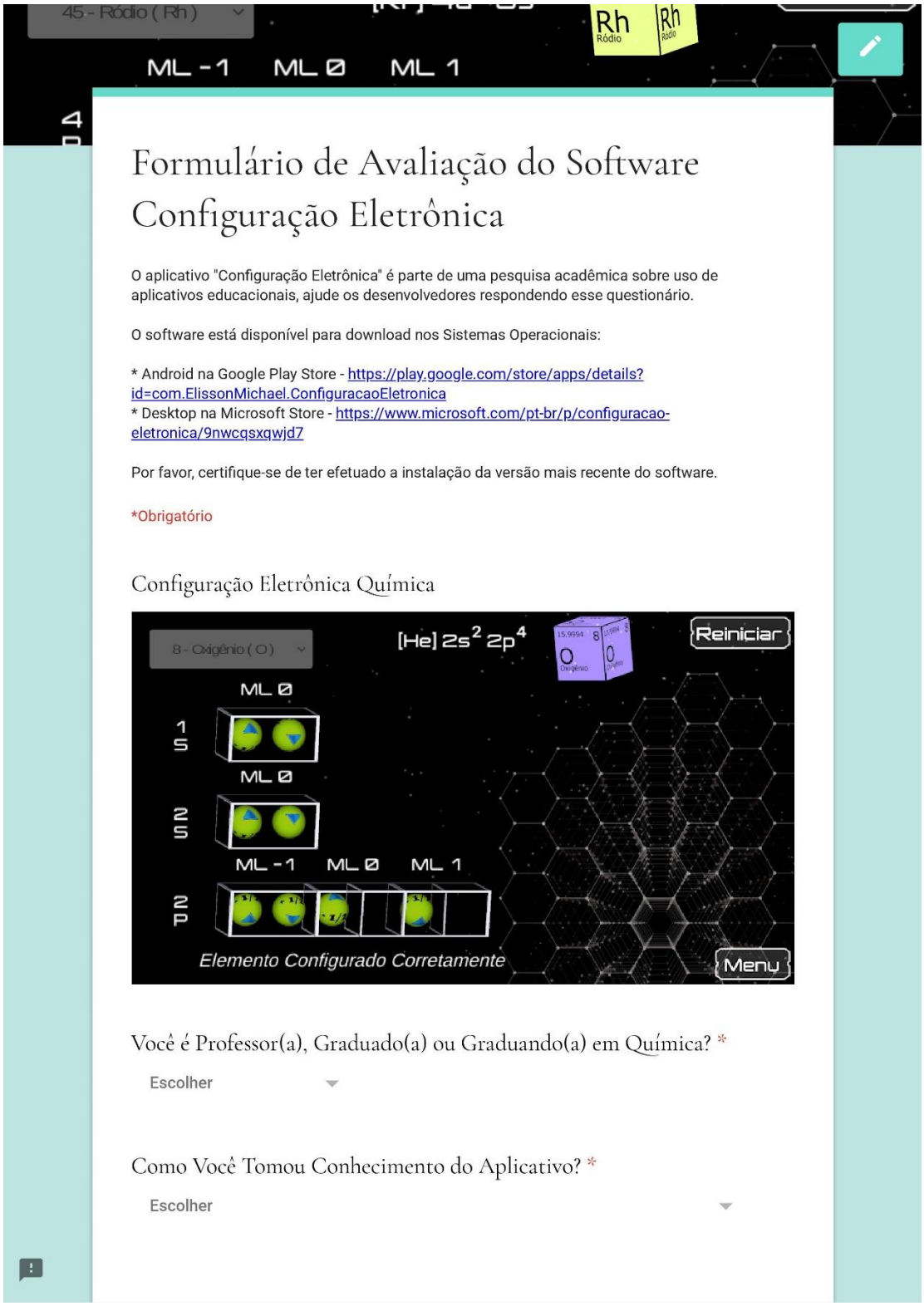
WARD, Jeff. **What is a Game Engine? - GameCareerGuide**. Disponível em: <http://www.gamecareerguide.com/features/529/what_is_a_game_.php>. Acesso em: 25 jul. 2018.

YUKOD SOFTWARE. **Electron Config Lite – Apps no Google Play**. Disponível em:

<https://play.google.com/store/apps/details?id=yukod.science.electronconfiglite>. Acesso em: 18 mar. 2019.

APÊNDICE

Apêndice 1 – Formulário de Avaliação do *App* Configuração Eletrônica



Formulário de Avaliação do Software Configuração Eletrônica

O aplicativo "Configuração Eletrônica" é parte de uma pesquisa acadêmica sobre uso de aplicativos educacionais, ajude os desenvolvedores respondendo esse questionário.

O software está disponível para download nos Sistemas Operacionais:

* Android na Google Play Store - https://play.google.com/store/apps/details?id=com.ElissonMichael.ConfiguracaoEletronica
* Desktop na Microsoft Store - https://www.microsoft.com/pt-br/p/configuracao-eletronica/9nwcqsxqwjd7

Por favor, certifique-se de ter efetuado a instalação da versão mais recente do software.

*Obrigatório

Configuração Eletrônica Química

Você é Professor(a), Graduado(a) ou Graduando(a) em Química? *

Escolher

Como Você Tomou Conhecimento do Aplicativo? *

Escolher

Legenda

Responda as questões a seguir conforme indicação da legenda:

1 - Discordo Completamente,
2 - Discordo,
3 - Não Concordo Nem Discordo,
4 - Concordo,
5 - Concordo Completamente

Afirmativas sobre o Aplicativo.

1) O Aplicativo Possui Nome Coerente *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

2) O Aplicativo é Fácil de Instalar *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

3) O Aplicativo Esteve Isento de Falhas Durante sua Utilização *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

4) O Aplicativo é de Fácil Utilização *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

5) O Aplicativo Não Apresenta Propagandas/Anúncios Durante a Execução *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

6) O Aplicativo Pode ser Utilizado para Despertar o Interesse do Usuário pelo Assunto *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

7) O Aplicativo Possuí Interface (Ícones, Menus, etc) Fácil de ser Reconhecida/Entendida *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

8) O Aplicativo é Visualmente Atraente *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

9) O Aplicativo é Interativo e o Tempo de Resposta das Operações Interativas é Adequado *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

10) O Aplicativo Possuí Cor e Detalhamento Favoráveis à Leitura em uma Tela Pequena *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

11) O Aplicativo Utiliza a Linguagem Química de Maneira Correta *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

12) O Aplicativo Apresenta os Conceitos Químicos de Forma Correta *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

13) O Aplicativo Contribui para a Construção do Conhecimento em Química Evitando a Mera Memorização *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

14) O Aplicativo Permite que os Conceitos Trabalhados Sejam Relacionados com Outros Conceitos de Química *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

15) Você Recomendaria o Aplicativo aos seus Colegas ou Alunos *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

Página 1 de 1

ENVIAR

Apêndice 2 – Formulário de Avaliação do *App* Geometria Molecular

Formulário de Avaliação do Software Geometria Molecular

O aplicativo "Geometria Molecular" é parte de uma pesquisa acadêmica sobre uso de aplicativos educacionais, ajude os desenvolvedores respondendo esse questionário.

O software está disponível para download nos Sistemas Operacionais:

* Android na Google Play Store - https://play.google.com/store/apps/details?id=com.ElissonMichael.LigacoesCovalentes
* Desktop na Microsoft Store - https://www.microsoft.com/pt-br/p/geometria-molecular/9pk3gthv5m0j

Por favor, certifique-se de ter efetuado a instalação da versão mais recente do software.

*Obrigatório

Geometria Molecular(Ligações Covalentes - Química)

Você é Professor(a), Graduado(a) ou Graduando(a) em Química? *

Escolher

Como Você Tomou Conhecimento do Aplicativo? *

Escolher

Legenda

Responda as questões a seguir conforme indicação da legenda:

1 - Discordo Completamente,
2 - Discordo,
3 - Não Concordo Nem Discordo,
4 - Concordo,
5 - Concordo Completamente

Afirmativas sobre o Aplicativo.

1) O Aplicativo Possui Nome Coerente *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

2) O Aplicativo é Fácil de Instalar *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

3) O Aplicativo Esteve Isento de Falhas Durante sua Utilização *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

4) O Aplicativo é de Fácil Utilização *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

5) O Aplicativo Não Apresenta Propagandas/Anúncios Durante a Execução *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

6) O Aplicativo Pode ser Utilizado para Despertar o Interesse do Usuário pelo Assunto *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

7) O Aplicativo Possuí Interface (Ícones, Menus, etc) Fácil de ser Reconhecida/Entendida *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

8) O Aplicativo é Visualmente Atraente *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

9) O Aplicativo é Interativo e o Tempo de Resposta das Operações Interativas é Adequado *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

10) O Aplicativo Possuí Cor e Detalhamento Favoráveis à Leitura em uma Tela Pequena *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

11) O Aplicativo Utiliza a Linguagem Química de Maneira Correta *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

12) O Aplicativo Apresenta os Conceitos Químicos de Forma Correta *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

13) O Aplicativo Contribui para a Construção do Conhecimento em Química Evitando a Mera Memorização *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

14) O Aplicativo Permite que os Conceitos Trabalhados Sejam Relacionados com Outros Conceitos de Química *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

15) Você Recomendaria o Aplicativo aos seus Colegas ou Alunos *

	1	2	3	4	5	
Discordo	○	○	○	○	○	Concordo

Página 1 de 1

ENVIAR