VisEdu-Química: Visualizador de Material Educacional – Módulo de Química

Aluno(a): Arthur Henrique Eggert
Orientador: Dalton Solano dos Reis

FURB
UNIVERSIDADE DE BLUMENAU

Roteiro

- Introdução
- Objetivos
- Fundamentação Teórica
- Desenvolvimento
- Resultados e Discussões
- Conclusões
- Extensões
- Demonstração

Introdução

- Complexidade da Química
- Dificuldade em associar assunto com realidade
- Dificuldade do professor
- Uso da informática
- Navegadores, HTML5 e WebGL

Objetivos

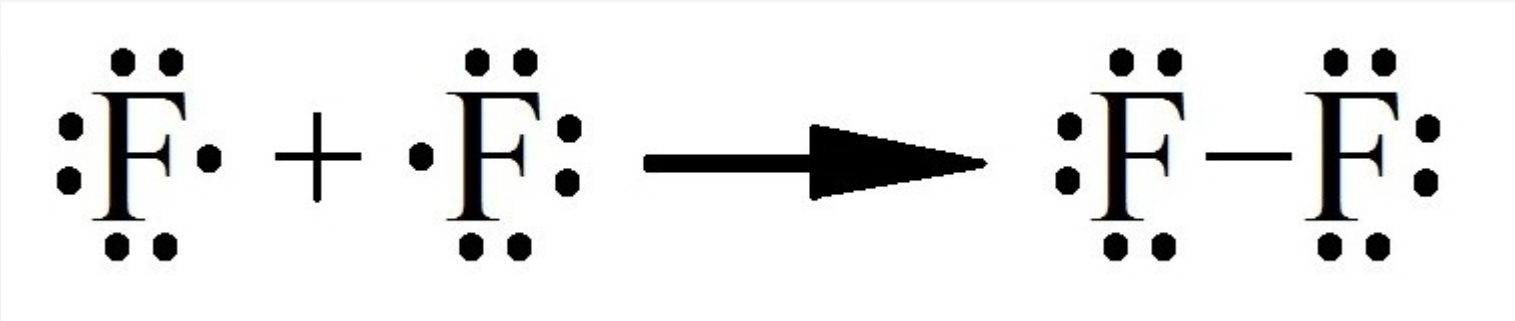
- Ferramenta Web para auxilio do ensino da química(Fórmulas Moleculares)
- Criar um ambiente 2D para a modelagem de moléculas em suas fórmulas estruturais completas
- Validar a molécula criada no ambiente 2D
- Criar uma visualização 3D da molécula

Fundamentação Teórica

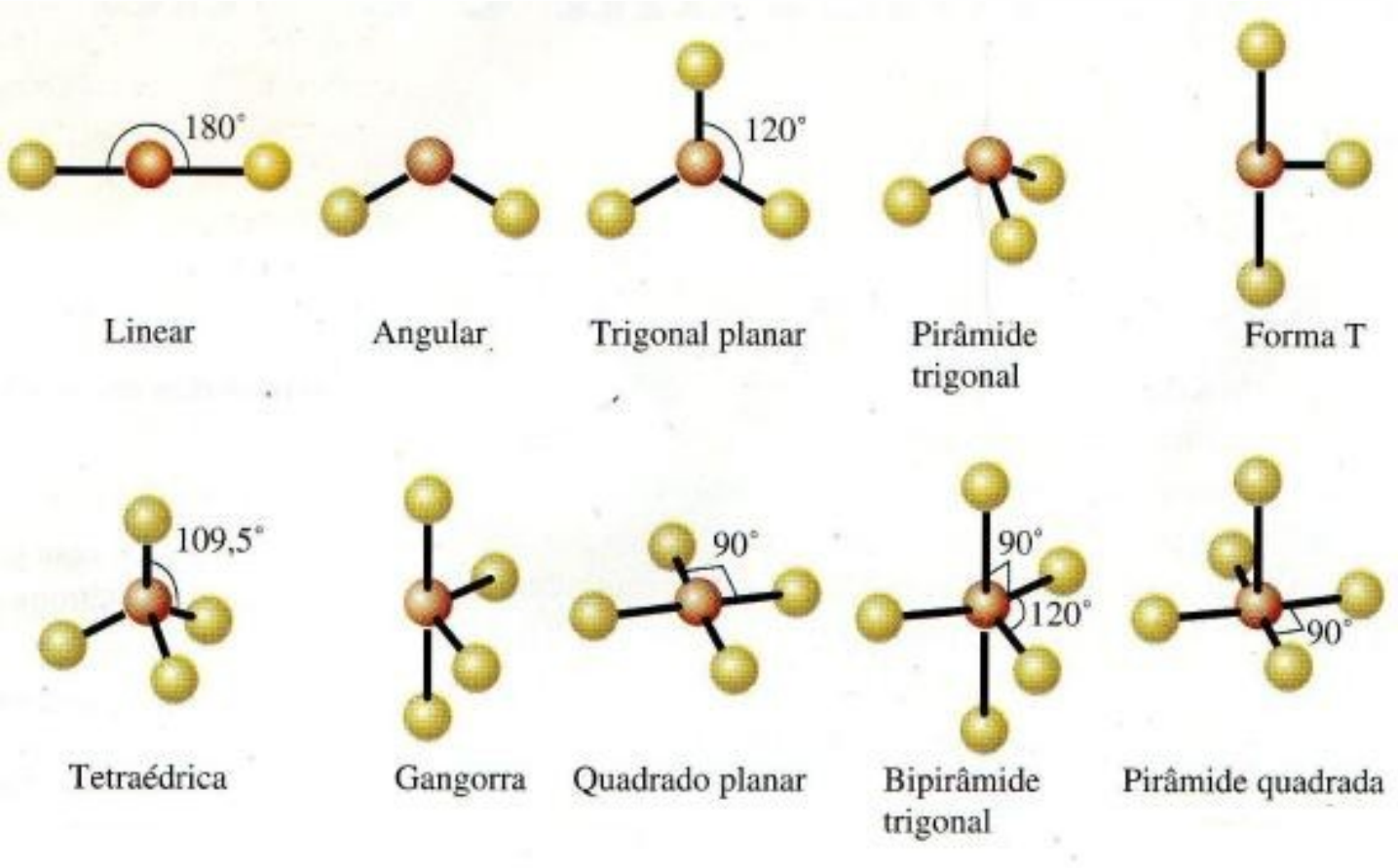
- Informática para otimizar processo de aprendizagem
- Professor deve saber explorar a informática para o surgimento de novas práticas pedagógicas
- Química é um processo experimental, a informática pode auxiliar neste ponto

Fundamentação Teórica

- Átomos, Elementos e Moléculas
- Organização dos Elementos
- Fórmulas
- Ligações
- Estrutura de Lewis
- Geometria Molecular



Fundamentação Teórica

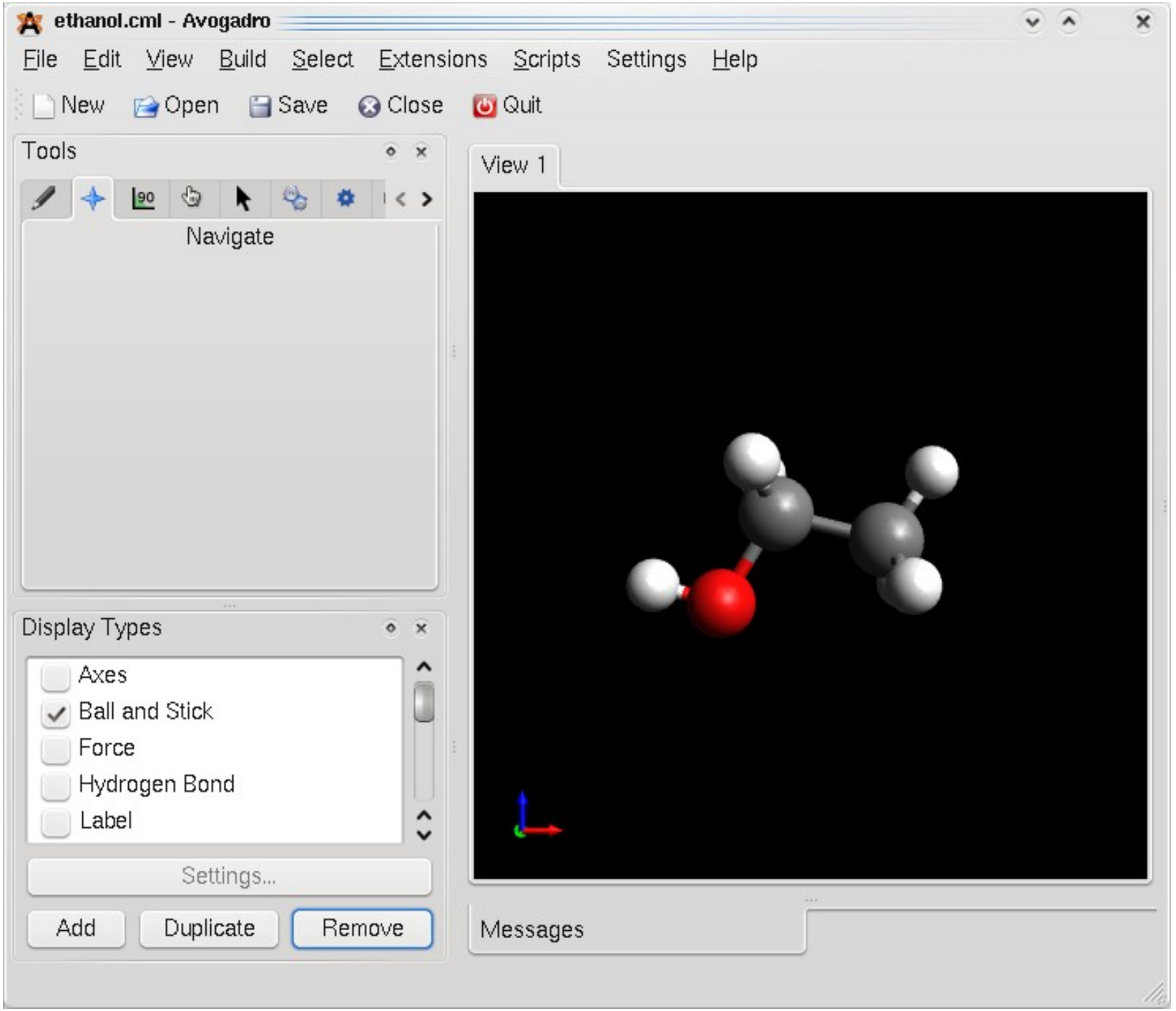


Fundamentação Teórica

- Ambiente 3D
  - Three.js
  - Engine (Montibeler, 2014)
- Ambiente 2D
  - Fabric.js

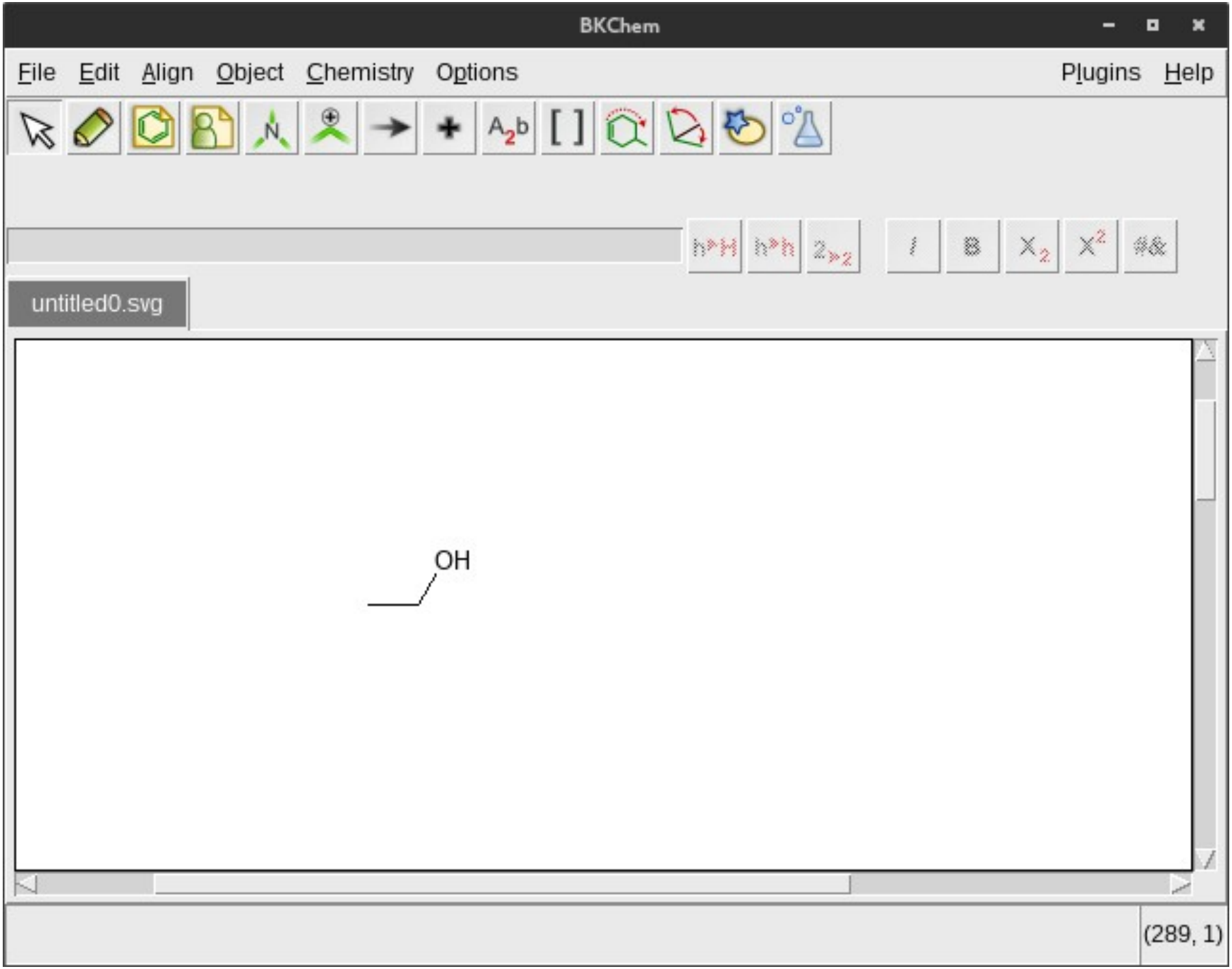
Trabalhos Correlatos

- Avogadro
  - Edição 3D
  - Visualização 3D
  - Bases Públicas de Moléculas
  - Tipo de Apresentação


ethanol.cml - Avogadro
File Edit View Build Select Extensions Scripts Settings Help
New Open Save Close Quit
Tools
Navigate
Display Types
Axes
Ball and Stick
Force
Hydrogen Bond
Label
Settings...
Add Duplicate Remove
View 1
Messages

Trabalhos Correlatos

- BKChem
  - Edição 2D
  - Validação da estrutura
  - Bases Públicas de Moléculas


BKChem
File
Edit
Align
Object
Chemistry
Options
Plugins
Help
untitled0.svg
OH
(289, 1)

Requisitos Funcionais

- Permitir a digitação da Fórmula Química
- Criação da Fórmula Estrutural com base na Fórmula Química
- Validar Fórmula Estrutural
- Exibir resultado em ambiente 3D
- Interação de câmera no ambiente 3D

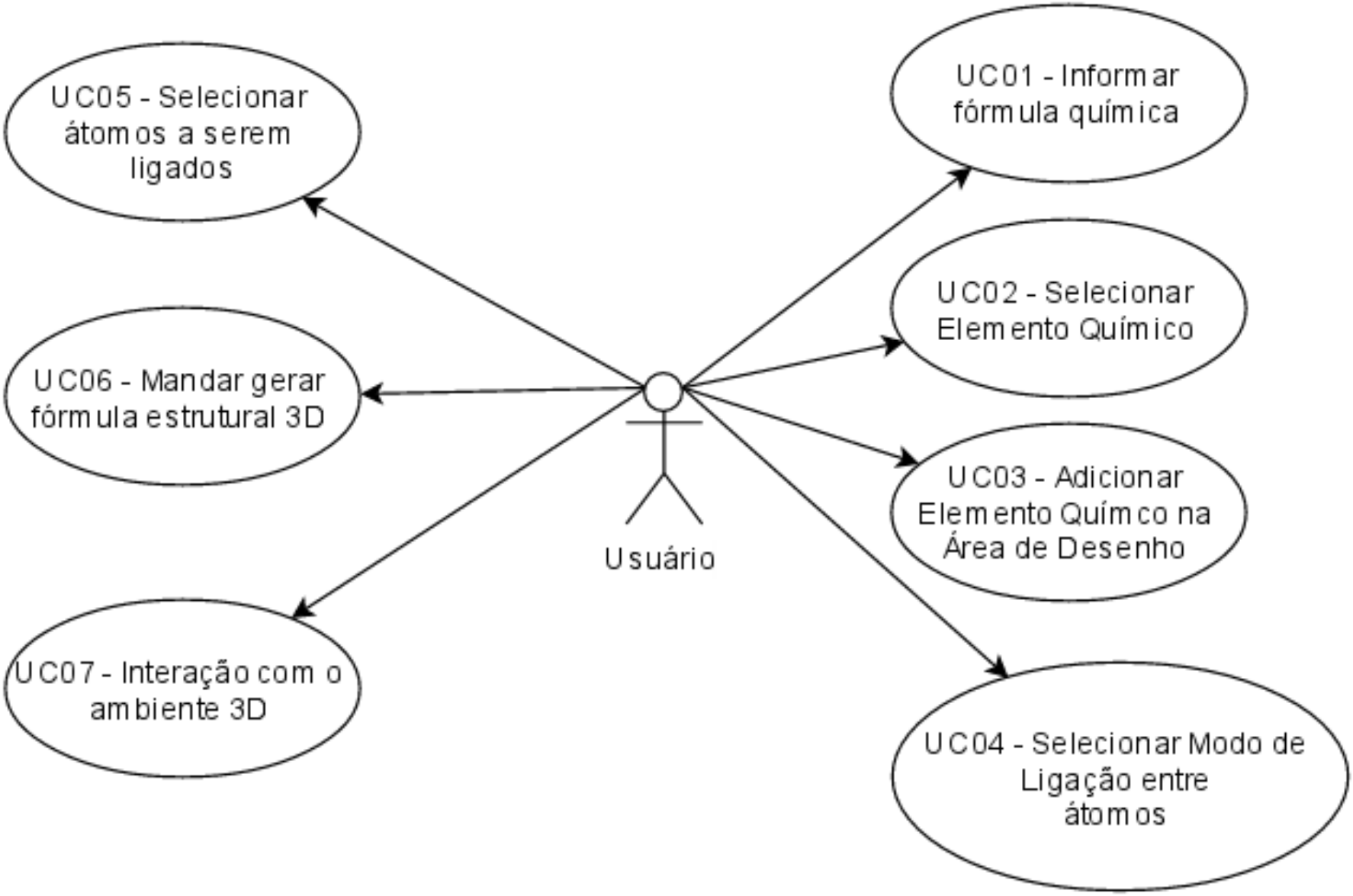
Requisitos Não Funcionais

- Javascript e HTML5
- WebGL + Three.js
- Intellij Idea como IDE

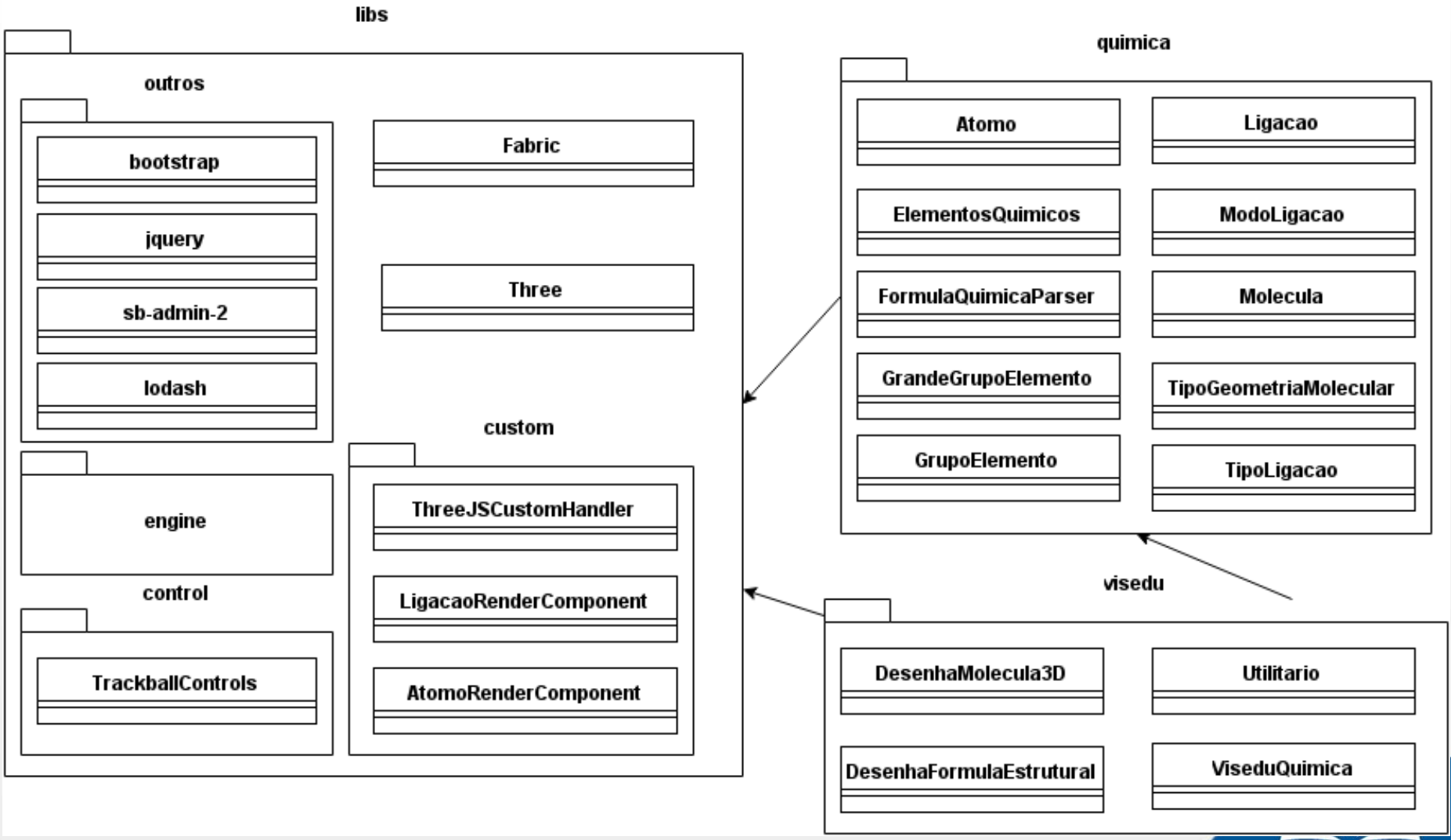
Especificação

- Quatro itens principais
  - Área de Entrada
  - Lista de Elementos
  - Área de Desenho
  - Área de Visualização

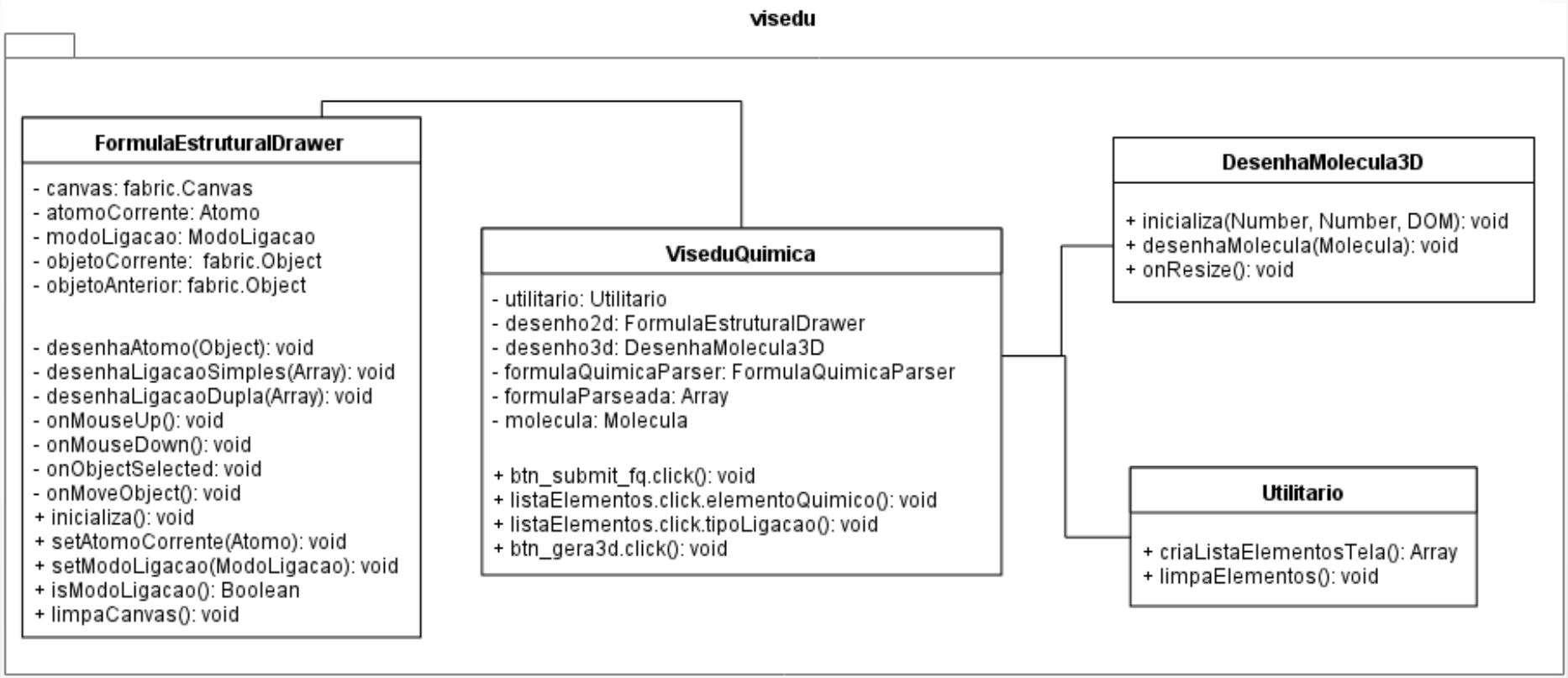
Especificação - Casos de Uso



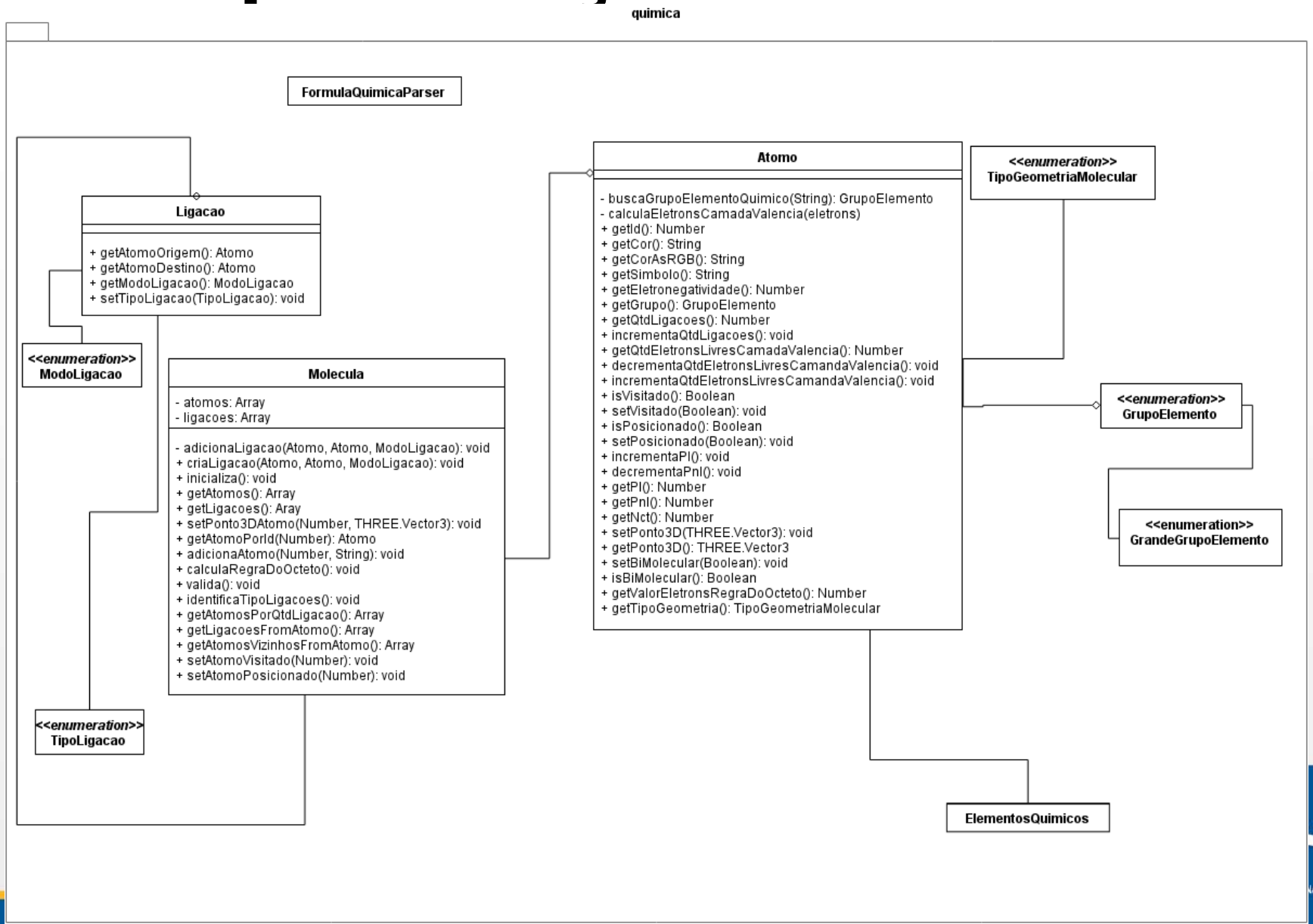
Especificação – Pacotes

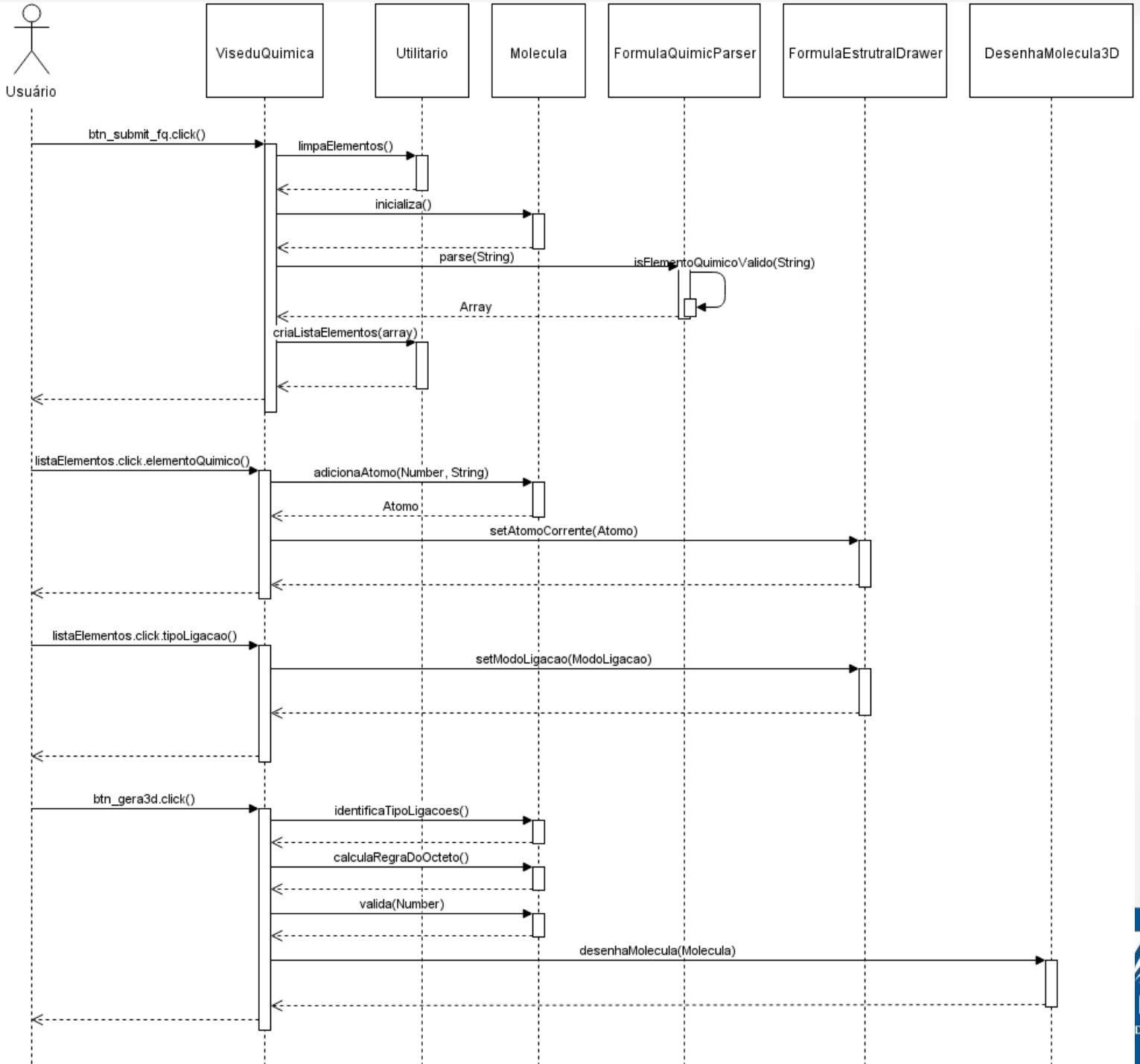


Especificação - Visedu



Especificação - Quimica




Usuário
ViseduQuimica
Utilitario
Molecula
FormulaQuimicParser
FormulaEstrutralDrawer
DesenhaMolecula3D
btn_submit_fq.click()
limpaElementos()
inicializa()
parse(String)
isElementoQuimicoValido(String)
Array
criaListaElementos(array)
listaElementos.click.elementoQuimico()
adicionaAtomo(Number, String)
Atomo
setAtomoCorrente(Atomo)
listaElementos.click.tipoLigacao()
setModoLigacao(ModoLigacao)
btn_gera3d.click()
identificaTipoLigacoes()
calculaRegraDoOcteto()
valida(Number)
desenhaMolecula(Molecula)

Implementação

```
var isElementoQuimicoValido = function (simbolo) {
        return _.includes(ElementosQuimicos.SIGLAS, simbolo);
    };

self.parse = function (formula) {
    var formulaSplited;
    var elementos = [];
    if (formula) {
        formulaSplited = formula.split(/(?=[A-Z])/);
        formulaSplited.forEach(function (elemento) {
            var elementoSplited, elementoQuimico, qtdElemento;
            elementoSplited = elemento.split(/(?=[0-9])/);
            elementoQuimico = _.get(elementoSplited, 0);
            qtdElemento = _.get(elementoSplited, 1, 1);
            for (var i = 2; i < elementoSplited.length; i++) {
                qtdElemento += _.get(elementoSplited, 2, 1);
            }
            if (isNaN(qtdElemento)) {
                throw new EvalError('Elemento Químico ' + elementoQuimico + ' com quantidade inválida (' + qtdElemento + ')');
            } else {
                if (isElementoQuimicoValido(elementoQuimico) === true) {
                    for (i = 0; i < parseInt(qtdElemento); i++) {
                        elementos.push({sigla: elementoQuimico});
                    }
                } else {
                   throw new EvalError('Elemento Químico ' + elementoQuimico + ' não encontrado!');
                }
            }
        });
    } else {
        throw new EvalError('Valor não pode ser nula/branco')
    }
    return elementos;
};
```

Implementação

```
var onMouseUp = function () {
   if (atomoCorrente) {
      var atomoId = atomoCorrente.getId();
      atomoCorrente = null;
      onMouseUpCallback(atomoId);
   }
};

var onMouseDown = function (evento) {
   if (self.isModoLigacao()) {
      if (objetoAnterior !== null && objetoCorrente != null) {
         if (_.isEqual(modoLigacao, ModoLigacao.SIMPLES)) {
            desenhaLigacaoSimples([objetoAnterior.left, objetoAnterior.top, objetoCorrente.left, objetoCorrente.top]);
         } else {
            desenhaLigacaoDupla([objetoAnterior.left, objetoAnterior.top, objetoCorrente.left, objetoCorrente.top])
         }
         onMouseDownCallback(modoLigacao, objetoCorrente.atomoId, objetoAnterior.atomoId);
         canvas.deactivateAll().renderAll();
         objetoAnterior = null;
         objetoCorrente = null;
         modoLigacao = null;
      }
   } else {
      if (atomoCorrente) {
         desenhaAtomo(canvas.getPointer(evento.e));
      }
   }
   canvas.renderAll();
};
```

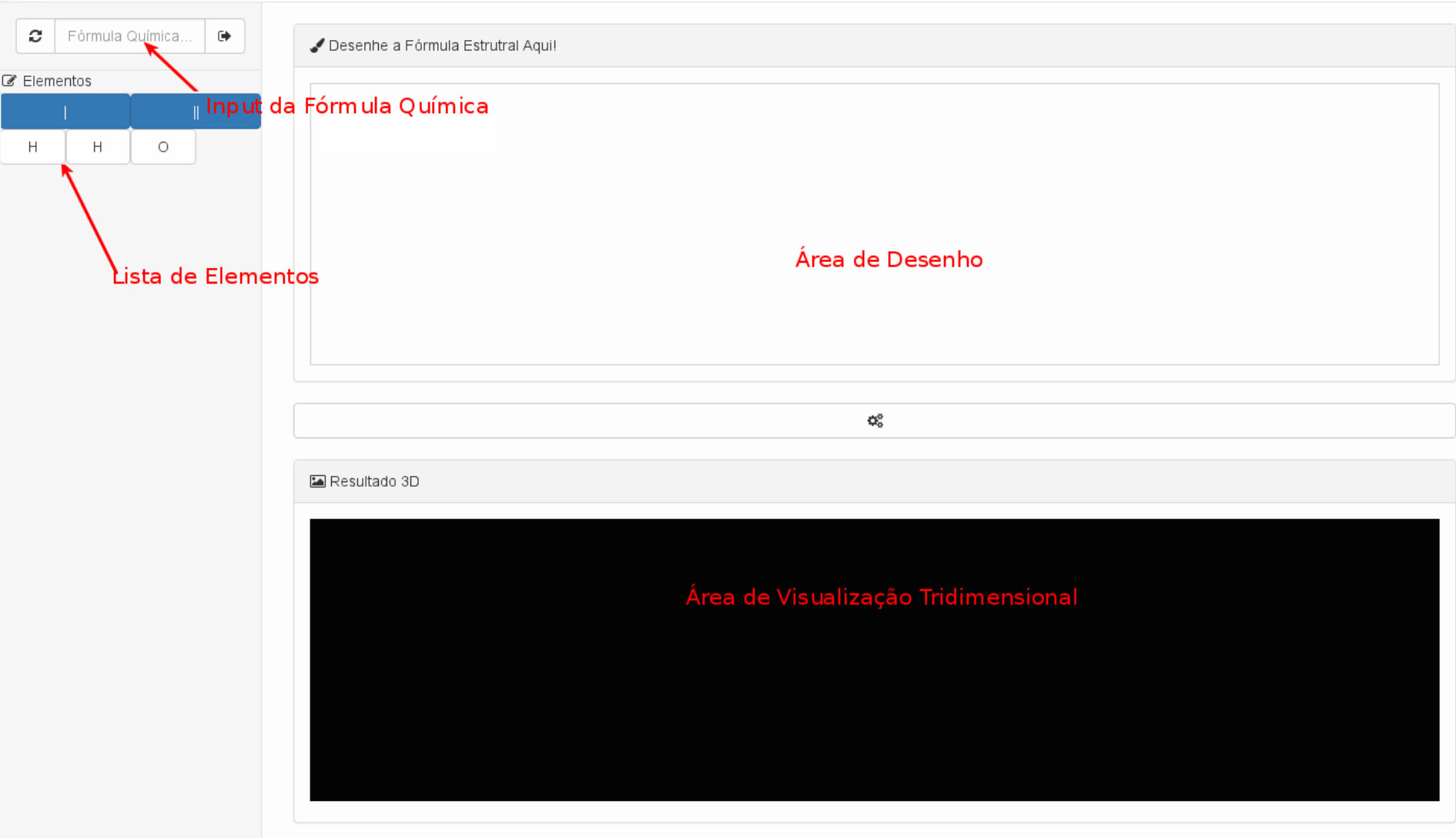
Implementação

```
self.valida = function (qtdElementosFormula) {
   if (qtdElementosFormula != atomos.length) {
         throw new Error('Nem todos os elementos da formula fora utilizados![Qtd Elementos: ' + qtdElementosFormula +
         '] [Qtd Atomos: ' + atomos.length + ']');
   }

   atomos.forEach(function (atomo) {
      ...
      var achou = false;
      ligacoes.forEach(function (ligacao) {
         ...
         if (ligacao.getTipoLigacao() === TipoLigacao.IONICA && ligacao.getModoLigacao() !== ModoLigacao.SIMPLES) {
            throw new Error('Ligações Ionicas são sempre simples');
         }
         if (ligacao.getTipoLigacao() === TipoLigacao.METALICA) {
            throw new Error('Ligações Metálicas não são tratadas pelo aplicativo');
         }
      });

      if (validaRegraOcteto) {
         if (atomo.getQtdEletronsLivresCamadaValencia() !== 0) {
            if (atomo.getValorEletronsRegraDoOcteto() !== atomo.getQtdEletronsLivresCamadaValencia()) {
               throw new Error('Molécula não esta obedecendo a regra do octeto ' + '[ Atomo => ' + atomo.getSimbolo() +
               ' Valor Regra do Octeto => ' + atomo.getValorEletronsRegraDoOcteto() + ' Eletrons Livres => ' +
               atomo.getQtdEletronsLivresCamadaValencia() + ' ]')
            }
         }
      }
      if (!achou) {
         throw new Error('Atomo ' + atomo.getSimbolo() + ' não é usado em nenhuma ligação![1]');
      }
      if (atomo.getQtdLigacoes() === 0) {
         throw new Error('Atomo ' + atomo.getSimbolo() + ' não é usado em nenhuma ligação![2]');
      }
   });
};
```

Implementação

```
if (tipoGeometria.eixos === "xyz") {
  x2 = x + 4 * Math.cos(tipoGeometria.angulos[countLigacoes][1]) * Math.sin(tipoGeometria.angulos[countLigacoes][0]);
  y2 = y + 4 * Math.sin(tipoGeometria.angulos[countLigacoes][1]);
  z2 = z + 4 * Math.cos(tipoGeometria.angulos[countLigacoes][1]) * Math.cos(tipoGeometria.angulos[countLigacoes][0])
} else {
  x2 = x + Math.cos(tipoGeometria.angulos[countLigacoes]) * 4;
  y2 = y + Math.sin(tipoGeometria.angulos[countLigacoes]) * 4;
  z2 = z;
}
```

Operacionalidade



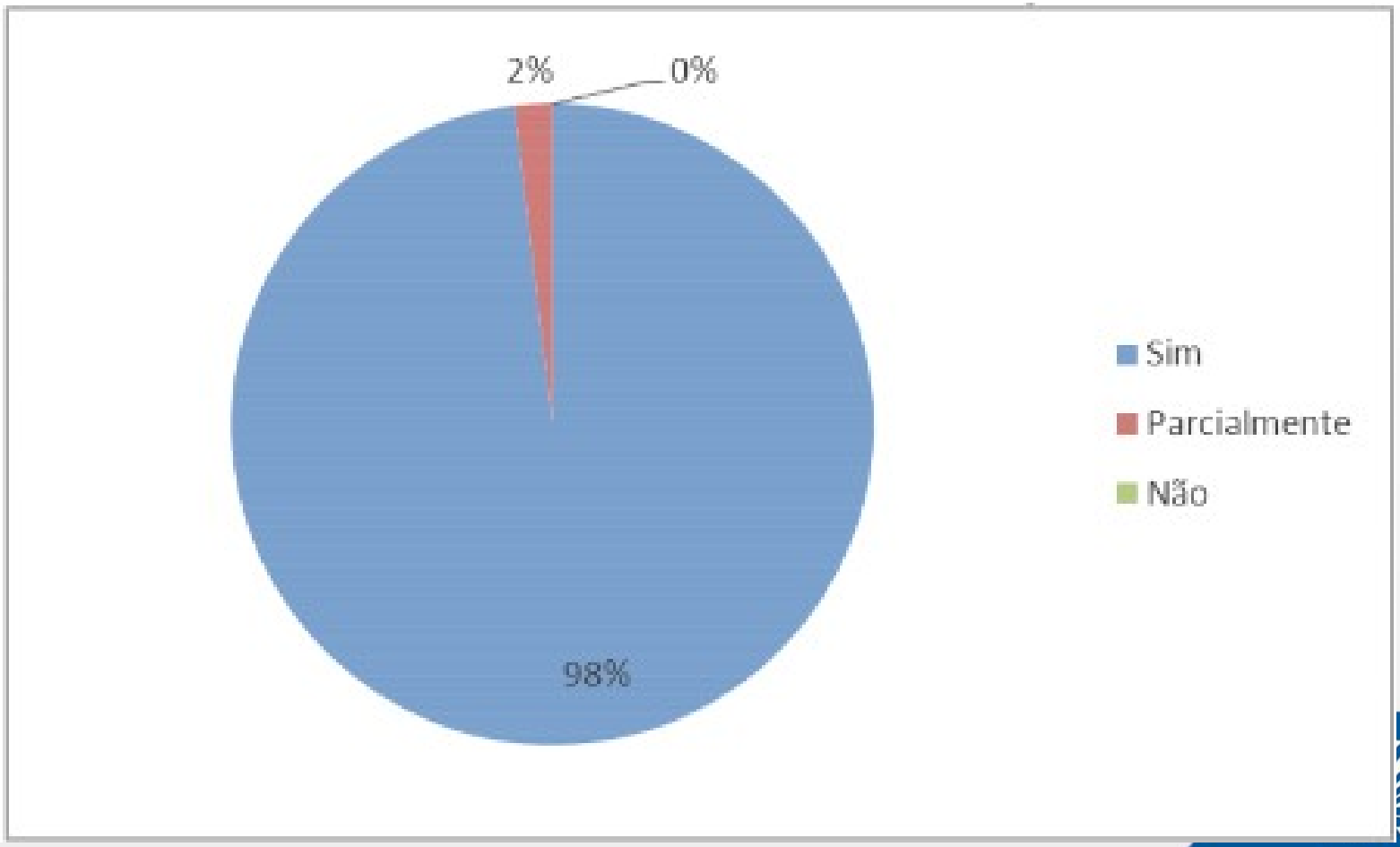
Resultados e Discussões

Questões aplicadas aos estudantes (53 alunos)

Nº	Questão	Tipo de Resposta
1	Você considera que ferramentas interativas podem auxiliar no processo de educação?	Múltipla Escolha.
2	No cenário atual de ensino, é viável para as instituições de ensino disponibilizarem *notebooks* para grupos de alunos?	
3	Você considera a aplicação apresentada como algo que poderia auxiliar a fixação do conteúdo?	
4	Como você classifica a usabilidade da aplicação?	
5	Com base nas respostas anteriores, quais foram os "pontos positivos" encontrados na aplicação?	Descritiva.
6	Com base nas respostas anteriores, quais foram os "pontos negativos" encontrados na aplicação?	
7	Com base nas respostas anteriores, quais seriam suas sugestões para melhorar o aplicativo?	

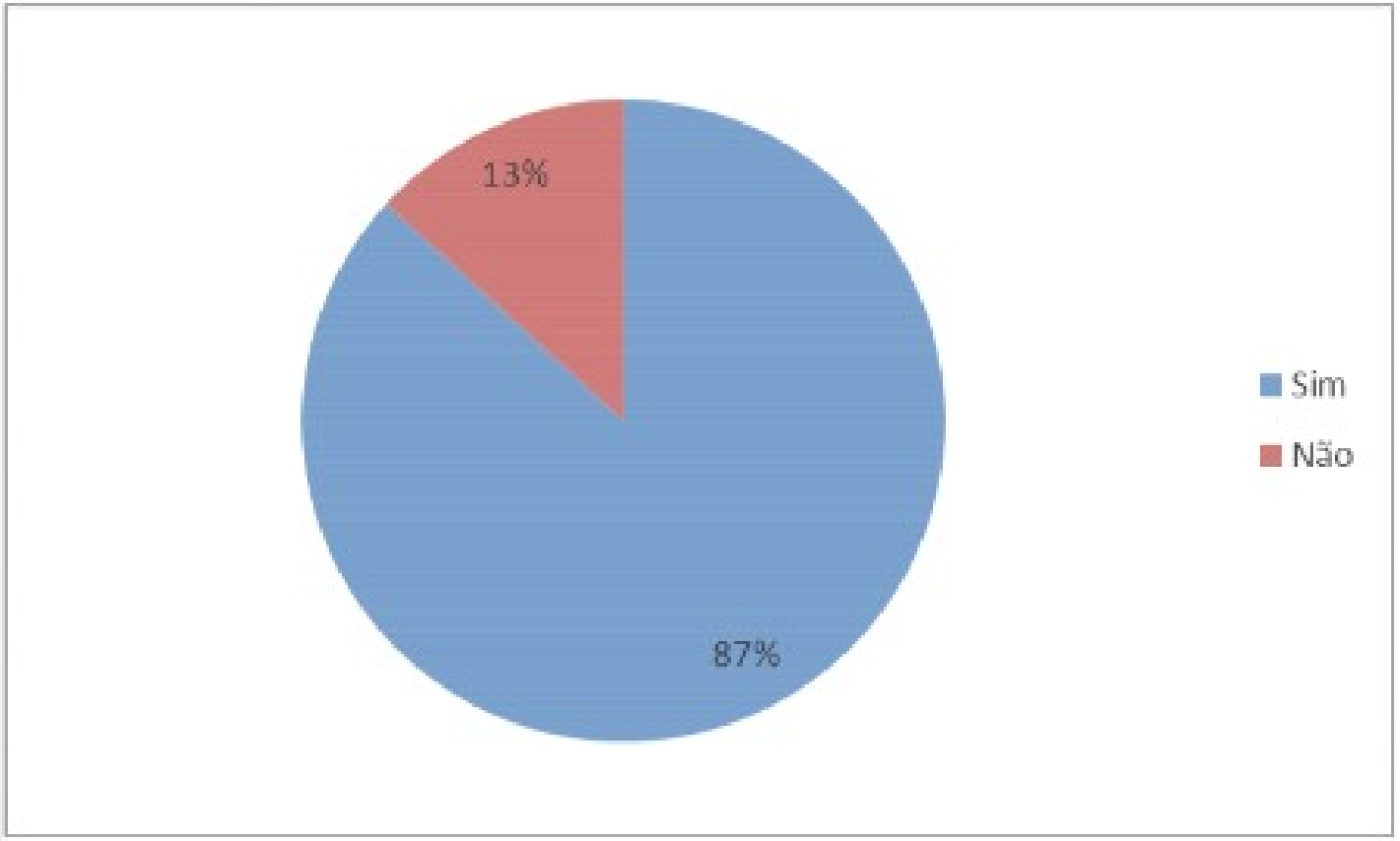
Resultados e Discussões

Você considera que ferramentas interativas podem auxiliar no processo de educação?



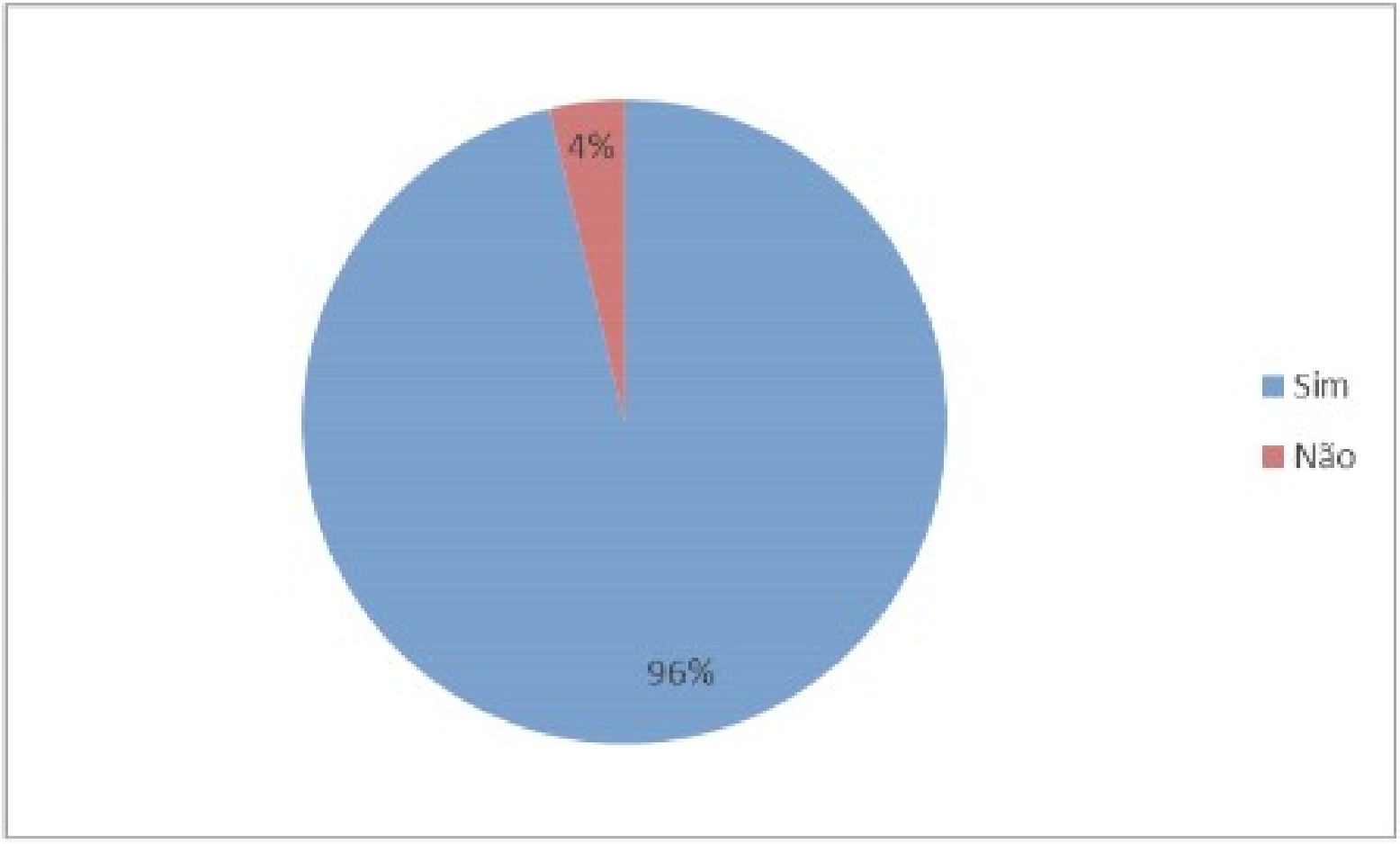
Resultados e Discussões

No cenário atual de ensino, é viável para as instituições de ensino disponibilizarem notebooks para grupos de alunos?



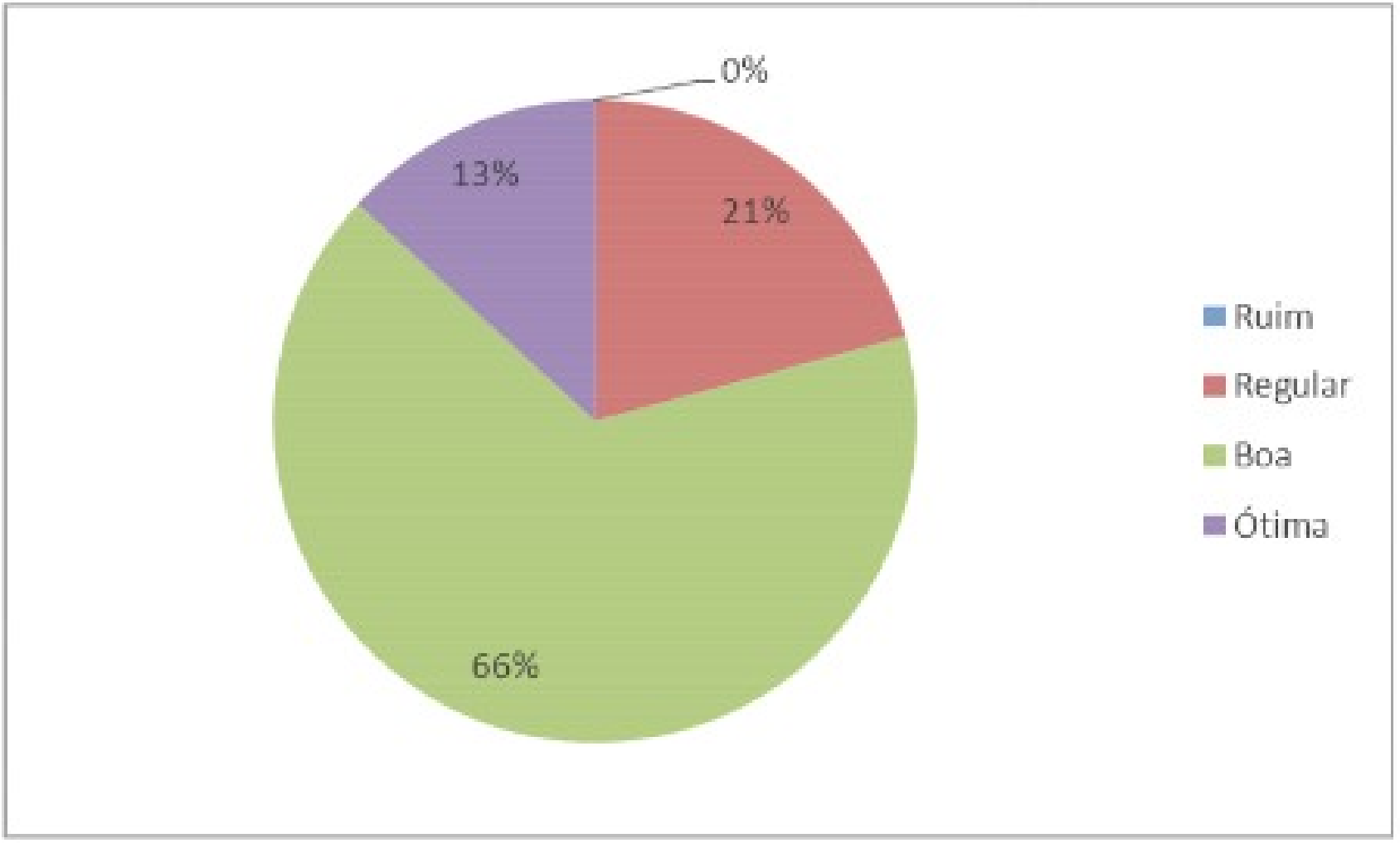
Resultados e Discussões

Você considera a aplicação apresentada como algo que poderia auxiliar a fixação do conteúdo?



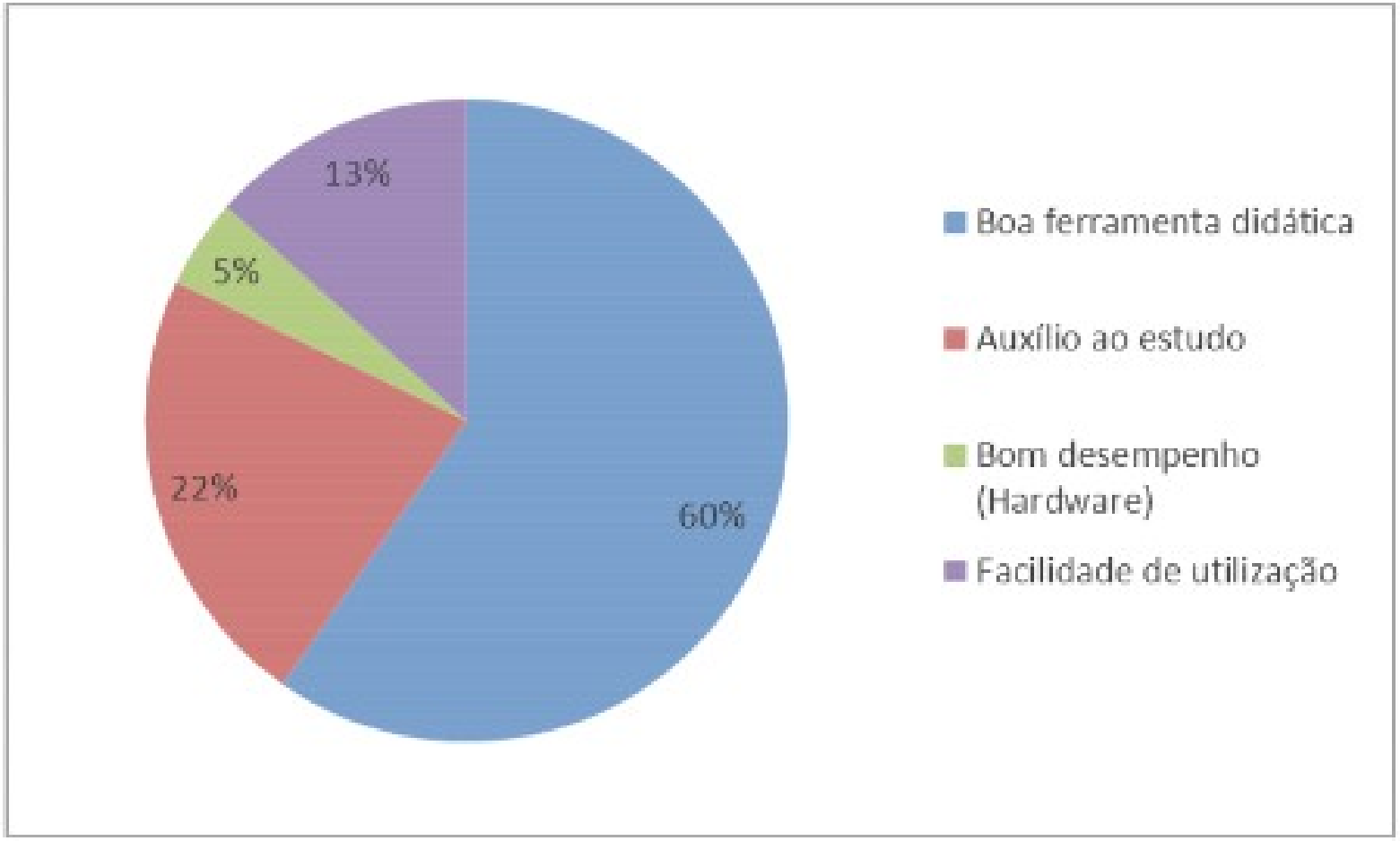
Resultados e Discussões

Como você classifica a usabilidade da aplicação?



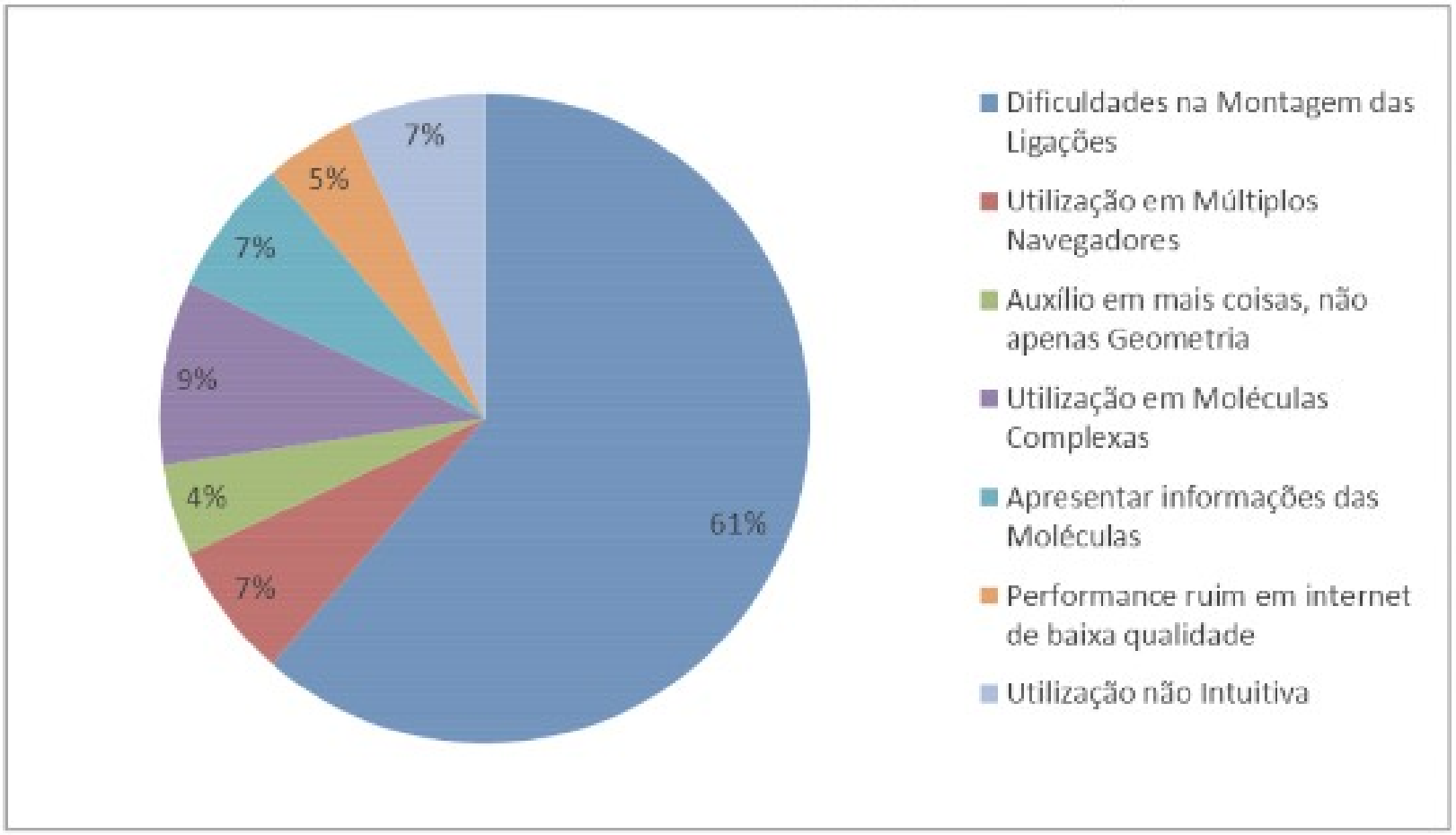
Resultados e Discussões

Com base nas respostas anteriores, quais foram os “pontos positivos” encontrados na aplicação?



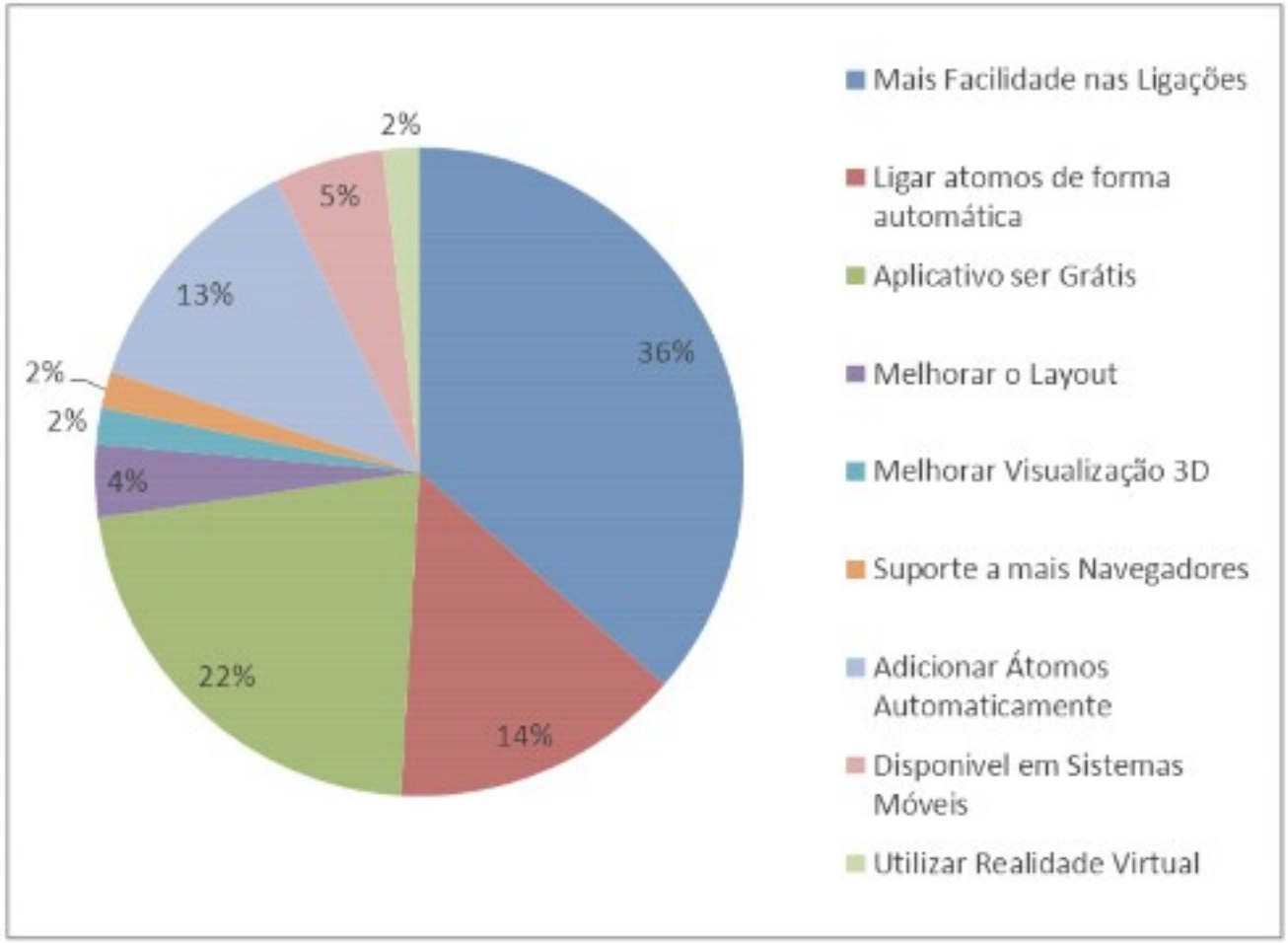
Resultados e Discussões

Com base nas respostas anteriores, quais foram os “pontos negativos” encontrados na aplicação?



Resultados e Discussões

Com base nas respostas anteriores, quais seriam suas sugestões para melhorar o aplicativo?



Resultados e Discussões

Testes de performance 3D no Google Chrome

Qtde de Objetos	FPS	Memória (MB)	Observação
40	60	202	Sistema funcionou normalmente, as taxas de FPS e memória estavam constantes durante a utilização.
80	60	237	
160	60	307	
320	60	432	
640	42~60	732	Sistema funcionou normalmente, porem as taxas de FPS variaram constantemente entre os eventos de zoom e rotação.
1.280	12~42	1300	Sistema não estava funcionando corretamente, o Google Chrome travou por aproximadamente 30 segundos até que conseguiu renderizar todos os objetos, a utilização dos eventos de zoom e rotação ficaram prejudicadas.

Resultados e Discussões

Testes de performance 3D no Firefox

Qtde de Objetos	FPS	Memória (MB)	Observação
40	50	500	Sistema funcionou normalmente, as taxas de FPS e memória estavam constantes durante a utilização.
80	35	600	Sistema funcionou normalmente, porém as .taxas de FPS estava muito baixas mesmo sem nenhuma interação com o ambiente.
160	30	650	
320	28	700	
640	18	1.500	Sistema não funcionou normalmente, e as .taxas de FPS estava muito baixas mesmo sem nenhuma interação com o ambiente, o inverso pode ser dito da memória estava com um consumo excessivo
1.280	2,1	2.000	Sistema não estava funcionando corretamente, o Firefox travou por aproximadamente 60 segundos até que conseguiu renderizar todos os objetos, a utilização dos eventos de zoom e rotação ficaram prejudicadas.

Resultados e Discussões

Testes de performance 2D no Google Chrome

Qtde de Objetos	FPS	Memória (MB)	Observação
40	60	73	Sistema funcionou normalmente, as taxas de FPS e memória estavam constantes durante a utilização.
80	60	230	
160	60	250	
320	55	260	
640	50	265	
1.280	40	270	

Testes de performance 2D no Firefox

Qtde de Objetos	FPS	Memória (MB)	Observação
40	41	600	Sistema funcionou normalmente, as taxas de FPS e memória estavam constantes durante a utilização.
80	13	1200	Sistema funcionou normalmente, porém as .taxas de FPS estava muito baixas mesmo sem nenhuma interação com o ambiente.
160	10	1300	
320	13	1300	
640	-	-	Sistema travou e não terminou de carregar as peças
1.280	-	-	Sistema travou e não terminou de carregar as peças

Resultados e Discussões

Comparação entre os trabalhos correlatos e o proposto

Caraterística	BKChem	Avogadro	Proposto
ambiente de edição 2D	x		x
ambiente de edição 3D		x	
visualização 3D		x	x
validação da estrutura química	x		x
exportar resultado	x	x	
ambiente web			x
leitura de estruturas InChI	x		

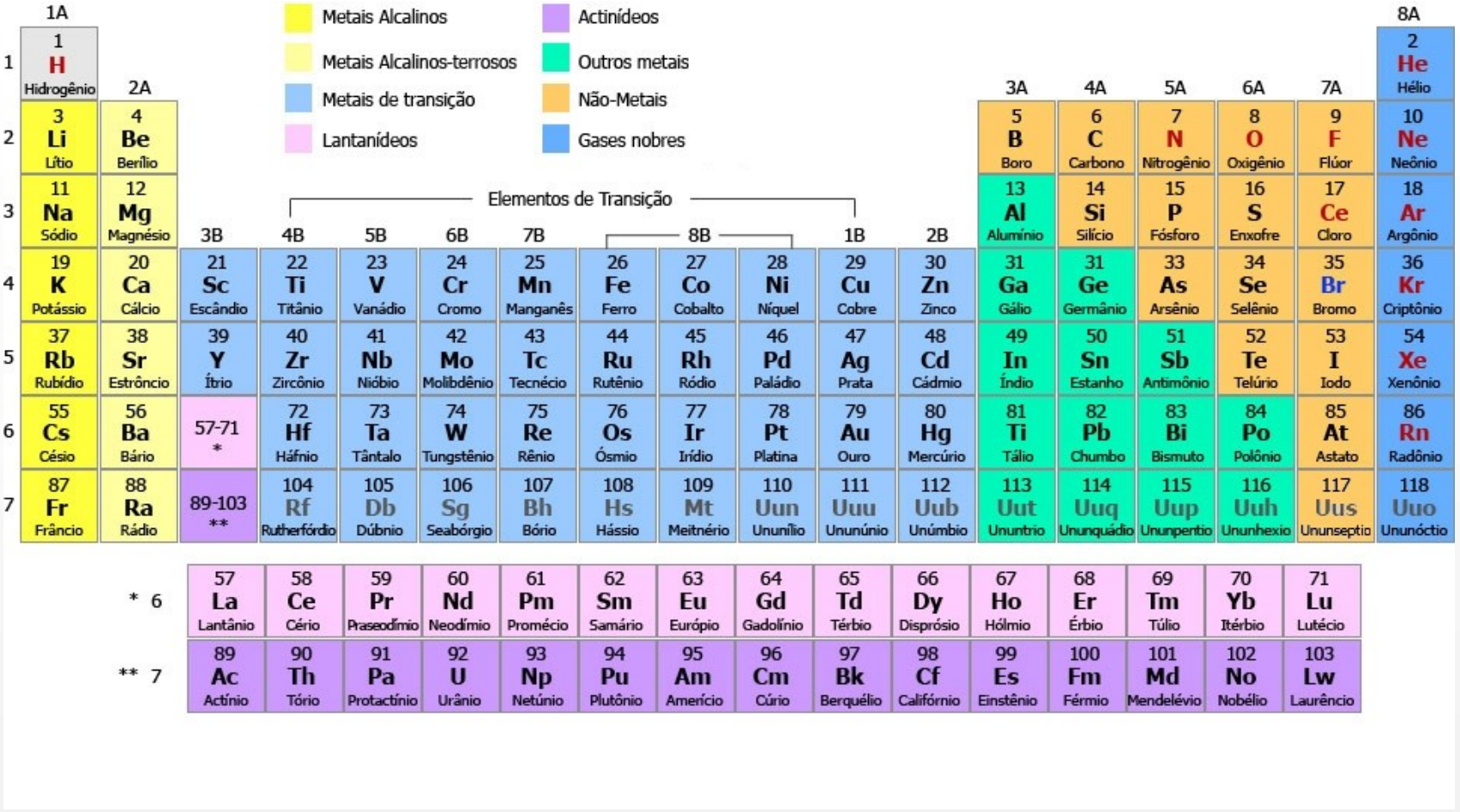
Conclusões

- Utilização da Engine (Montibeler, 2014) e do Fabric.js se mostraram satisfatórias
- Desempenho satisfatório para o que foi proposto
- Ajustes são necessários
- Boa aceitação nos testes de usabilidade

Sugestões

- Melhorar a forma com que as ligações são feitas
- Adicionar mais informações sobre a molécula no ambiente tridimensional
- Adicionar automaticamente os átomos da fórmula no ambiente bidimensional
- Melhorar a visualização do ambiente tridimensional
- Utilizar iluminação na visualização tridimensional
- Permitir a utilização de mais navegadores
- Permitir o desenho de moléculas complexas

Demonstração



Grupos	Tipo Ligação	Condição	Exemplo de Molécula
Metais Alcalinos ou Metais Alcalinos Terrosos e Hidrogênio	Iônica	Metais com eletronegatividade menor ou igual a 1,0	LiH, NaH, KH, CaH_2, SrH_2, BaH_2
	Covalente	Metais com eletronegatividade maior que 1,0	BeH_2, MgH_2
Não Metais ou Semi-metais e Hidrogênio	Covalente	Nenhuma	HF, HCl, HBr, HI, H_2O, H_2S, H_2Se, PH_3, SI_4, CH_4
Não Metais ou Semi-metais e Não Metais ou Semi-metais	Covalente	Nenhuma	F_2, Cl_2, CO, CO_2, P_4, Cl_2O
Qualquer Metal e Qualquer Metal	Metálica	Nenhuma	$Fe_{(n)}$, $Al_{(n)}$
Qualquer Metal e Não Metais ou Semi-metais	Iônica	Diferença de eletronegatividade maior ou igual a 1,7	NaCl, KCl, AlF_3, K_2O
	Covalente	Diferença de eletronegatividade menor que 1,7	$AlCl_3$, $HgCl_2$

Pares	Pares Ligantes	Pares não Ligantes	Geometria	Ângulo das Ligações	Exemplo
2	2	0	Linear	180º	CO_2
3	2	1	Angular	120º	SO_2
4	2	2	Angular	110º	H_2O
3	3	0	Trigonal Plana	120º	BF_3
4	4	0	Tetraédrica	109,5º	CH_4
4	3	1	Piramidal	110º	NH_3
5	5	0	Bipiramidal Trigonal	90º, 120º	PCl_5
5	4	1	Gangorra	180º, 120º	SF_4
5	3	2	Forma de T	90º, 180º	ClF_3
5	2	3	Linear	180º	XeF_2
6	6	0	Octaédrica		
6	5	1	Piramidal Quadrada	90º	BrF_5
6	4	2	Quadrada Planar	90º	XeF_4
7	7	0	Bipiramidal Pentagonal	90º, 72º	IF_7

Grupos	Grandes Grupos
Hidrogênio	Hidrogênio
Metais Alcalinos	Metais
Metais Alcalinos Terrosos	
Metais de Transição	
Metais de Transição Interna	
Outros Metais	
Não Metais	Ametais
Gases Nobres	Gases Nobres

Implementação

```
    self.calculaRegraDoOcteto = function () {

        ligacoes.forEach(function (ligacao) {

            var origem = self.getAtomoPorId(ligacao.getAtomoOrigem());
            var destino = self.getAtomoPorId(ligacao.getAtomoDestino());

            if (ligacao.getTipoLigacao() === TipoLigacao.COVALENTE) {
                origem.incrementaQtdEletronsLivresCamandaValencia();
                destino.incrementaQtdEletronsLivresCamandaValencia();
                if (ligacao.getModoLigacao() === ModoLigacao.DUPLA) {
                    origem.incrementaQtdEletronsLivresCamandaValencia();
                    destino.incrementaQtdEletronsLivresCamandaValencia();
                }
            } else {
                if (ligacao.getTipoLigacao() === TipoLigacao.IONICA &&
ligacao.getModoLigacao() !== ModoLigacao.SIMPLES) {
                    throw new Error('Ligações Ionicas são sempre
simples');
                } else {
                    if (origem.getEletronegatividade() >
destino.getEletronegatividade()) {

origem.incrementaQtdEletronsLivresCamandaValencia();

destino.decrementaQtdEletronsLivresCamandaValencia();
                    } else {

origem.decrementaQtdEletronsLivresCamandaValencia();

destino.incrementaQtdEletronsLivresCamandaValencia();
                    }
                }
            }
        });
    };
```

Implementação

```
    var isElementoQuimicoValido = function (simbolo) {
        return _.includes(ElementosQuimicos.SIGLAS, simbolo);
    };
    self.parse = function (formula) {
        var formulaSplited;
        var elementos = [];
        if (formula) {
            formulaSplited = formula.split(/(?=[A-Z])/);
            formulaSplited.forEach(function (elemento) {
                var elementoSplited, elementoQuimico, qtdElemento;
                elementoSplited = elemento.split(/(?=[0-9])/);
                elementoQuimico = _.get(elementoSplited, 0);
                qtdElemento = _.get(elementoSplited, 1, 1);
                for (var i = 2; i < elementoSplited.length; i++) {
                    qtdElemento += _.get(elementoSplited, 2, 1);
                }
                if (isNaN(qtdElemento)) {
                    throw new EvalError('Elemento Químico ' +
elementoQuimico + ' com quantidade inválida ' +
                        '(' + qtdElemento + ')');
                } else {
                    if (isElementoQuimicoValido(elementoQuimico) === true)
{
                        for (i = 0; i < parseInt(qtdElemento); i++) {
                            elementos.push({sigla: elementoQuimico});
                        }
                    } else {
                        throw new EvalError('Elemento Químico ' +
elementoQuimico + ' não encontrado!');
                    }
                }
            });
        } else {
            throw new EvalError('Valor não pode ser nula/branco')
        }
        return elementos;
    };
```

Implementação

```
    var onMouseUp = function () {
        if (atomoCorrente) {
            var atomoId = atomoCorrente.getId();
            atomoCorrente = null;
            onMouseUpCallback(atomoId);
        }
    };

    var onMouseDown = function (evento) {
        if (self.isModoLigacao()) {
            if (objetoAnterior !== null && objetoCorrente != null) {
                if (_.isEqual(modoLigacao, ModoLigacao.SIMPLES)) {
                    desenhaLigacaoSimples([objetoAnterior.left,
objetoAnterior.top, objetoCorrente.left, objetoCorrente.top]);
                } else {
                    desenhaLigacaoDupla([objetoAnterior.left,
objetoAnterior.top, objetoCorrente.left, objetoCorrente.top])
                }
                onMouseDownCallback(modoLigacao, objetoCorrente.atomoId,
objetoAnterior.atomoId);
                objetoAnterior = null;
                objetoCorrente = null;
                modoLigacao = null;
            }
        } else {
            if (atomoCorrente) {
                desenhaAtomo(canvas.getPointer(evento.e));
            }
        }
        canvas.renderAll();
    };
```

Implementação

```
    self.valida = function (qtdElementosFormula) {
        if (qtdElementosFormula != atomos.length) {
            throw new Error('Nem todos os elementos da formula fora
utilizados![Qtd Elementos: ' + qtdElementosFormula +
                '] [Qtd Atomos: ' + atomos.length + ']');
        }

        atomos.forEach(function (atomo) {

            if (atomos.length === 2) {
                atomo.setBiMolecular(true);
            }

            var achou = false;
            ligacoes.forEach(function (ligacao) {
                if (ligacao.getAtomoOrigem() === atomo.getId() ||
ligacao.getAtomoDestino() === atomo.getId()) {
                    achou = true;
                }
                if (ligacao.getTipoLigacao() === TipoLigacao.IONICA &&
ligacao.getModoLigacao() !== ModoLigacao.SIMPLES) {
                    throw new Error('Ligações Ionicas são sempre
simples');
                }
                if (ligacao.getTipoLigacao() === TipoLigacao.METALICA) {
                    throw new Error('Ligações Metálicas não são tratadas
pelo aplicativo');
                }
            });
            if (atomo.getQtdEletronsLivresCamadaValencia() !== 0) {
                if (atomo.getValorEletronsRegraDoOcteto() !==
atomo.getQtdEletronsLivresCamadaValencia()) {
                    throw new Error('Molécula não esta obedecendo a regra
do octeto ' + '[ Atomo => ' + atomo.getSimbolo() +
                        ' Valor Regra do Octeto => ' +
atomo.getValorEletronsRegraDoOcteto() + ' Eletrons Livres => ' +
                        atomo.getQtdEletronsLivresCamadaValencia() + ' ]')
                }
            }
            if (!achou) {
                throw new Error('Atomo ' + atomo.getSimbolo() + ' não é
usado em nenhuma ligação![1]');
            }
            if (atomo.getQtdLigacoes() === 0) {
                throw new Error('Atomo ' + atomo.getSimbolo() + ' não é
usado em nenhuma ligação![2]');
            }
        });
    };
```

Implementação

```
if (tipoGeometria.eixos === "xyz") {
    x2 = x + 4 * Math.cos(tipoGeometria.angulos[countLigacoes][1]) *
Math.sin(tipoGeometria.angulos[countLigacoes][0]);
    y2 = y + 4 * Math.sin(tipoGeometria.angulos[countLigacoes][1]);
    z2 = z + 4 * Math.cos(tipoGeometria.angulos[countLigacoes][1]) *
Math.cos(tipoGeometria.angulos[countLigacoes][0])
} else {
    x2 = x + Math.cos(tipoGeometria.angulos[countLigacoes]) * 4;
    y2 = y + Math.sin(tipoGeometria.angulos[countLigacoes]) * 4;
    z2 = z;
}
```