

UNIVERSIDADE PAULISTA

B22816-4 MARCIO FERNANDES CRUZ

B408FA-3 DANILO DE OLIVEIRA DOROTHEU

T618FA-6 RENATO GOULART RODRIGUES

B30DCD-7 JEFERSON PAWELS DE MACEDO

Desenvolvimento de um jogo com a utilização de interface gráfica

SÃO PAULO

2013

B22816-4 MARCIO FERNANDES CRUZ
B408FA-3 DANILO DE OLIVEIRA DOROTHEU
T618FA-6 RENATO GOULART RODRIGUES
B30DCD-7 JEFERSON PAWELS DE MACEDO

Desenvolvimento de um jogo com a utilização de interface gráfica

Dados obtidos através dos métodos e aplicações estudados em aulas e com pesquisas para fins de obtenção de nota para o curso de Ciência da Computação 3/4º Semestre (Noturno) da Universidade Paulista (UNIP) sendo entrega ao Professor/Orientador

Ciência da Computação 3º e 4º Semestre – Noturno

Professor José.

SÃO PAULO

2013

SUMÁRIO

OBJETIVO DO TRABALHO.....	6
RESUMO.....	7
ABSTRACT.....	8
INTRODUÇÃO.....	9
TEMA ESCOLHIDO.....	11
1. REGRAS E FUNCIONAMENTO DO JOGO.....	12
1.1. Comandos do Jogo.....	14
2. PLANO DE DESENVOLVIMENTO.....	16
2.1. Compra de Matéria Prima.....	17
2.2. Fabricação de Produto Acabado.....	17
2.3. Venda de Produto Acabado.....	18
2.4. Palestra de Sustentabilidade.....	18
2.5. Filtro Antipoluição.....	18
2.6. Produtos Recicláveis.....	18
2.7. Programas de Eco Eficiência.....	18
3. ESTRUTURA DO PROJETO.....	21
3.1. Divisão Tipos.....	21
3.2. Divisão Model.....	21
3.3. Divisão Controller.....	24
3.4. Divisão View.....	26
4. LINHAS DE CODIGO.....	30
5. PROGRAMA EM FUNCIONAMENTO.....	54
5.1. Dia Comercial.....	54
5.2. Dia Fora de Comercio.....	55
5.3. Tela GameOver.....	56
5.4. Final de Semana.....	57
5.5. Compar Matéria Prima.....	58
5.6. Fabricar Peças.....	59
5.7. Vender Peças.....	60
5.8. Nota Natureza.....	61
6. BIBLIOGRAFIA.....	62
7. FICHA APS.....	63

OBJETIVO DO TRABALHO

O trabalho apresentado tem o objetivo de ganhar nota dentro das diretrizes do curso da Universidade Paulista (UNIP) com pretensão futura de comércio eletrônico nas lojas virtuais de aplicativos para Smartphones e gadgets. Consiste em desenvolver um jogo utilizando a linguagem de programação Java com uso da interface gráfica.

O jogo escolhido foi um jogo com ideia de sustentabilidade ambiental com intuito de ensinar o jovem sobre os principais acontecimentos do mundo ambiental. O arco temporal do jogo é da Eco 92 até a Rio+10 mas, suas ideias são de aplicação atemporais. O jogador terá que fazer uma empresa sobreviver dentro do período das 2 conferências: Eco 92 e Rio +10.

O principal objetivo é levar o conhecimento sobre a preservação da natureza, não ver o capitalismo, representado como o papel da indústria, como um vilão, mas, sim, um agente que degrada a natureza, mas, como qualquer um de nós, deve retribuir a ela o que se tira dela.

RESUMO

O trabalho apresentado demonstra o desenvolvimento de um jogo educativo voltado ao meio ambiente. Traz conceitos básicos de compra de matéria-prima, produção e fabricação de produtos, bem como conceitos como investimentos em sustentabilidade e redução de custos operacionais graças à redução de gastos em função de práticas sustentáveis. Sendo assim, é voltado para um público juvenil, a partir dos 15 anos.

A indústria é uma empresa que muitas vezes agride a natureza pelo simples fato de ter que processar matéria-prima em produtos acabados. Apesar de ela ter sua importância para o país, por trazer emprego e desenvolvimento, não justifica o fato de prejudicar o ecossistema.

O jogo “Fábrica 21” resgata as ideias da educação ambiental trazidas a partir da Eco 92, onde foi concebido a Agenda 21. Onde se busca desenvolver um estudo constante em seu ambiente, seja ele uma escola, trabalho ou no caso, uma empresa, buscando buscar práticas para melhorar o trato com a Natureza.

O jogo traz um painel onde se observa uma nota que vai de 0 a 10 e, conforme aumenta esta nota, reduz os gastos operacionais da indústria, diminui o preço de custo do produto e, por fim, aumenta a lucratividade. O objetivo do jogo é de qualquer empresa, sobreviver. Tem que manter um saldo azul e, para isso, tem que saber comprar matéria-prima, fabricar, vender investir em sustentabilidade. “Fabrica 21” tem um arco temporal que vai do final da Eco 92 e vai até o início da RIO+10 em 2002. Se o jogador no papel do empresário conseguir levar a empresa no azul até lá, vence o jogo.

ABSTRACT

The work presented demonstrates the development of an educational game directed to the environment. Brings basics of buying raw materials, production and manufacturing, as well as concepts such as sustainability investments and reduce operating costs through reduced spending due to sustainable practices. Therefore, it is geared toward a young audience, after 15 years.

The Industry is a company that often harms nature simply by having to process raw materials into finished products. Although it has its importance for the country, by bringing jobs and development, does not justify the fact harming the ecosystem.

The game "Factory 21" rescues the ideas of environmental education brought from the Eco 92, which was designed to Agenda 21. Which attempts to develop a constant study in your environment, be it a school, work or in this case, a company seeking practices seek to improve the deal with nature.

The game brings a panel where it is observed that a score ranging from 0 to 10 and the note increases as reduces operating costs the industry reduces the cost price of the product and, ultimately, increases the profitability. The goal of any business is to survive. Have to maintain a balance blue and, therefore, have to know buying raw materials, manufacture, sell, and invest in sustainability. "Fabrica 21" has a time span from the end of the Eco 92 and runs until the start of the RIO +10 in 2002. If the player in the role of the entrepreneur can take the company up there in blue, wins the game.

INTRODUÇÃO

No ano de 1992, ocorreu no Rio de Janeiro a CNUMAD (Conferência das Nações Unidas sobre o Meio Ambiente). No término desta conferência internacional foram apresentados vários trabalhos, dentre eles, a Agenda 21 que, segundo a definição do próprio Ministério do Meio Ambiente é Instrumento de planejamento para a construção de sociedades sustentáveis, em diferentes bases geográficas, que concilia métodos de proteção ambiental, justiça social e eficiência econômica.

É um projeto que pode ser implantado em qualquer aspecto da sociedade, na escola, na comunidade, dentro do próprio ambiente familiar ou mesmo nas empresas. Ela tem um objetivo comum: conscientizar as pessoas com boas práticas para conservar e proteger o meio ambiente para que, num futuro tenhamos um mundo melhor.

O cenário deste trabalho é a aplicação da Agenda 21 dentro de uma fábrica. Foi desenvolvido um jogo em que, um jogador acompanha diariamente a vida de uma empresa. O jogo possui alguns comandos básicos, normalmente utilizados em qualquer indústria, que são: comprar matéria-prima, produzir produtos acabados e vender as mercadorias produzidas, para assim, alcançar o lucro. Este ciclo básico faz a empresa sobreviver.

No decorrer da vida de um fábrica, direta ou indiretamente, vai degradando a natureza. Por isso, foi introduzido uma variável do jogo, que é uma “nota da natureza”. Esta nota, fictícia, é atribuída à empresa pela própria Mãe Natureza que, conforme a empresa vai passando os dias, vai diminuindo.

O objetivo principal do jogador é fazer a empresa sobreviver mantendo um saldo em dinheiro positivo. Basicamente, para conseguir isso, esta deve comprar matéria-prima, fabricar e vender, assim, obtendo o lucro. Isto, conforme foi escrito parágrafo acima, vai degradando a natureza e, conforme esta nota vai ficando mais baixa, vai ficar mais difícil para o jogador fazer operações de produção de mercadorias, vendas, etc, pois, o custo da empresa vai aumentar em todos os aspectos e chegará uma hora em que a fábrica começará a diminuir seu lucro e, começar a ter prejuízo diário, até fechar.

Para a empresa sobreviver, ela deve executar boas práticas para a natureza, aplicando programas de sustentabilidade, implantação de programas de ecodesenvolvimento, investir em programas de reciclagem, etc. Vai haver ações na tela do jogo, para que se possa investir na natureza. Fazendo isso, a nota da natureza vai aumentando e, a empresa terá custos mais baixos de produção, o custo diário da empresa como iluminação, operacional, também é reduzido e, por consequência, a empresa volta a lucrar e a enriquecer.

No jogo Fábrica 21, o jogador, no papel de um industrial, vai aprender que a luta para fazer uma empresa sobreviver é diária e constante e, vai se conscientizar que investir na natureza é investir em seu próprio negócio. Fazer palestras de sustentabilidade, comprar filtros de fumaça como outras boas atitudes tem seu custo inicial, mas, que pode virar investimento no decorrer dos dias pois, acaba por reduzir os custos administrativos e operacionais.

O arco temporal do jogo é o final da Eco-92 e vai até o início da Rio+10, que foi uma conferência feita em Setembro/2002, na África do Sul. Se o jogador chegar o final de

10 anos de existência de fábrica, sem a falir, será considerado um vencedor.

TEMA ESCOLHIDO

De acordo com a normalização interna da Universidade Paulista (UNIP), estudos disciplinares APS (Atividade Práticas Supervisionadas). Desenvolvimento de um jogo com a utilização de interface gráfica.

1. REGRAS E FUNCIONAMENTO DO JOGO

O jogo é de caráter educativo e tem nível baixo de dificuldade, porém visa ser profundo no objetivo de ensinar a administrar uma empresa aplicando boas práticas de sustentabilidade. O jogo tem seu arco temporal entre o término da Eco 92 e o início da Rio+10, ocorrida em 2002. Se o jogador conduzir a empresa com o saldo positivo em conta corrente por 10 anos até a Rio+10, o jogador vence o jogo.

Para conseguir administrar a empresa, o jogador além de saber comprar matéria, prima, executar ordem de produção, vender, tem que aplicar boas práticas de sustentabilidade, a fim de ter obter uma boa nota da Natureza que, esta pode atribuir de 0 a 10 e, assim, reduzir seus custos de produção.

O jogador vai perceber que gastar dinheiro com o meio ambiente não é apenas um ato benevolente para a Natureza, mas, sim, investimento para seu próprio negócio. Práticas sustentáveis reduzem custos operacionais e o preço de fabricação do produto final vai ter um preço de custo menor.

São demonstrados em tempo real os custos de operação da empresa com ou sem investimento em meio ambiente e, o público do jogo Fábrica 21 desenvolverá uma consciência ambiental mais apurada graças a este software. Reduções de custos graças a boas práticas ambientais podem ser verificadas não só na ficção, mas sim em qualquer empresa na vida real e, isso, é um dos grandes objetivos da criação deste jogo.

Para facilitar o entendimento do jogo, buscamos um denominador comum e que é uma nota da natureza. Esta nota vai de 0 a 10. O jogo começa com uma nota base de 5 e, conforme vai se passando o tempo de operação, vai se reduzindo a nota. Conforme a empresa vai operando, direta ou indiretamente, vai degradando a Natureza e, esta, tende a diminuir a nota da empresa.

Esta nota, a cada dia se reduz em 3 pontos e, o jogador deve em algum momento tomar alguma medida para melhorar esta nota, senão, sua administração ficará difícil e sua empresa poderá ir a falência!

O custo base de operação da empresa é de R\$ 3.000,00 em dias úteis e, quanto maior a nota da natureza, menor será este custo. Quanto maior for a preocupação do administrador em manter esta nota próxima dos 10 mais fácil ficará sua administração.

A contagem de tempo real se dá em cada 1,5 segundos e, neste tempo, acrescenta-se 15 minutos em horário comercial, para não ficar entediante, fora deste horário, acrescenta-se o tempo em 1 hora. A cada passagem de dia, é demonstrado em painel o saldo do dia anterior e o acumulado da despesa e receita acumulado do dia.



O jogo possui uma animação de uma fábrica com uma chaminé em funcionamento que aparece em funcionamento em vários horários do dia, com a iluminação e, nos finais de semana e a noite, não funciona.

Para trazer um ar de realidade ao jogo, todas as operações de compra de matéria-prima, ordem de produção, venda de produtos acabados e mesmo investimento em programas de sustentabilidade, só podem ser feitos em dias de semana, em horário comercial.

1.1. Comandos do Jogo

Tabela 1 - Comandos do Jogo

	<p>Este é o comando utilizado para comprar matéria-prima. Cada clique no barril entra no estoque de matéria-prima da Fábrica 21 50 Kg de material ao preço de R\$ 4,00 por quilo, ou seja. A cada clique, debita-se da conta R\$ 200,00.</p>
	<p>Caso tenha 25 kg de matéria-prima no estoque, clicando aqui, fabrica-se 5 peças de produto acabado.</p> <p>O ato de produzir é agressor a natureza e por isso, o jogo diminui a nota da natureza nesta operação e, por isso, vemos um efeito de aceleração do tempo.</p>
	<p>Caso cliquemos neste botão, vendemos 5 peças ao preço de R\$ 100,00 a unidade, ou seja, acrescentamos R\$ 500,00 de receita ao nosso caixa.</p>
	<p>O jogo traz a noção que o investimento em educação aos colaboradores de boas práticas de sustentabilidade é de suma importância ao sucesso geral do negócio e, por isso, inserimos uma ação só com este fim. Através deste botão, investe-se R\$ 300,00 e aumenta 0,50 a nota da natureza.</p>
	<p>O jogo trouxe a ideia de uma indústria de natureza poluidora e, que, precisa investir em filtros antipoluidores constantemente. Através deste botão, compra-se estes filtros e aumenta-se esta nota. O preço deste aparelho é R\$ 600,00 e aumenta a nota em 1 ponto.</p>

	<p>Reciclagem é um dos temas de educação ambientais mais utilizados quando se trata em reaproveitar bens de consumo diminuindo a agressão à natureza. Quando uma empresa compra matérias de consumo, escritório reciclado contribui para o bem da natureza. O custo fictício no jogo Fabrica 21 é de R\$ 900,00 e aumenta a nota em 3 pontos.</p>
	<p>Programas de Eco Eficiência podem ser muito abrangentes e por isso é o mais caro, mas, sua nota é a maior. Custa R\$ 1.200,00 e aumenta a nota em 4 pontos.</p>

Se o jogador mantiver a nota do zero e continuar com o processo normal de compra de matéria-prima, fabricação e venda de produção, a fábrica continuará normalmente, mas, não com a lucratividade normal de uma fábrica sustentável. A ideia do jogo não é traçar uma jogabilidade de derrota ou vitória, mas, sim, um caminho em que a consciência ecológica prevaleça seja a grande vencedora e, a aplicabilidade da aclamada Agenda 21 concebida na ECO 92 seja esboçada neste software chamado Fábrica 21.

2. PLANO DE DESENVOLVIMENTO DO JOGO

A primeira questão a ser avaliada antes de ser traçado o caminho para o desenvolvimento do jogo foi traçar o cenário do aplicativo dentro do contexto de preservação ambiental que o trabalho estava sendo exigido.

Foi traçado um plano em criar um aplicativo educativo que traga ideias de sustentabilidade ambiental ao usuário. A ideia do jogo não é vencer ou perder, mas sim, fazer-se absorver os conceitos da importância de investir na Natureza para que esta possa trazer benefícios imediatos a todos.

A ideia de se escolher uma empresa, no caso, uma indústria, foi devido a sua função de transformação de matéria-prima para produtos acabados que, diretamente, acaba por agredir a Natureza.

O jogo representa uma indústria, mas, pode representar qualquer pessoa ou mesmo o sistema Capitalista em si e, o software foi concebido para que o jogador se sinta responsável pela preservação da Natureza. Isto não significa que devemos renegar o sistema atual (capitalismo) e mudar radicalmente a forma de vida, mas, sim, devemos retribuir a Natureza o que tomamos dela.

Dentro deste cenário foi criado um software em que o jogador pode executar funções comuns a qualquer empresa, que são: *comprar matéria-prima, fabricar produto acabado e vender produto de venda* e, outras funções de prática socioambiental que são: *palestra de sustentabilidade, comprar filtro para emissão de gases tóxicos, reciclagem, e programas e eco desenvolvimento*.

Depois de ter sido traçado as ações que o jogador pode fazer, chegou o momento de traçar a história do jogo e, neste cenário, acabamos por definir o marco em que o jogador pode vencer o jogo. O jogo tem sua história entre a Eco 92 e a RIO + 10. Duas conferências histórias do meio ambiente. A fábrica começa o jogo em 1992 com 10.000,00 de saldo e, a cada dia, com as ações de compra e venda e investimento em sustentabilidade, vai fazendo a empresa sobreviver e, o objetivo, é fazer a empresa chegar com saldo positivo em 2002, ou seja, na conferência RIO+10.

O jogo é educativo e tem nível baixo de dificuldade. O jogador só consegue dar *game over* se não fazer a empresa gerar receita, vendendo produtos de venda e também, não reduzindo seus custos operacionais e, para isso, tem que investir na natureza.

A Natureza reduz a nota da empresa em 4 pontos diários pois esta vai operando diariamente. Se a empresa começar a investir em ideias sustentáveis, a nota da natureza aumenta e, conseqüentemente a lucratividade aumenta. O jogo traz a ideia da Agenda 21, em que, em sua definição diz que é um projeto para que buscar o bem maior, que é a preservação da Natureza, deve estar em constante discussão e evolução.

Depois de traçado todo o plano teórico do jogo, chegou a hora de desenvolver o projeto de implementação. Foi definido a utilização do design pattern MVC para o desenvolvimento do jogo. Pode-se separar o desenvolvimento em duas linhas distintas, sendo: as regras de negócio e a camada de visualização. As ações principais que o jogador tem acesso estão todas encapsuladas na camada de negócio, chamada de Model e, estas, só são acessíveis na operação do jogo em horário comercial. Ou seja, o jogo possui uma linha de tempo que inicia dia 15/07/1992 às 08h e, vai se dando passos de 15 minutos, simulando a realidade. E, todas as ações de operação da empresa só podem ser feitas em dias da semana, em horário comercial. Fora deste horário, o tempo do sistema se acelera em 1 hora ao invés de 15 minutos, para que o jogador não se sinta entediado em jogar.

2.1. *Compra de Matéria Prima:*

Em horário comercial, o sistema está configurado para comprar 50 Kg de matéria-prima por vez, ao preço de R\$ 4,00 a unidade, ou seja, a cada “clique”, o saldo em conta reduz em R\$ 200,00.

2.2. *Fabricação de Produto Acabado:*

Caso houver estoque de matéria-prima no sistema, o botão que faz esta ação estará habilitado e, o seu clique fará produzir 5 peças de produtos acabados, ou seja, produtos de venda. Pra produzir 5 peças, utiliza-se 25 Kg de matéria-prima e, como as máquinas da indústria precisam operar, a Natureza reduz a nota da empresa.

2.3. *Venda de produto Acabado:*

Caso houver estoque de produto acabado, vende-se 5 peças por venda, ao preço de R\$100,00 por unidade. É o único caminho para trazer receita a empresa.

2.4. *Palestra de Sustentabilidade*

O jogo traz a ideia de conscientização como uma das principais ferramentas da preservação da Natureza e para isso, foi inserida esta ação em que o jogador compra uma palestra ao custo de R\$ 300,00 e, isso, faz aumentar a nota da empresa perante a natureza em 0.5 pontos.

2.5. *Filtro antipoluição*

O jogo colocou esta ação fictícia em que o usuário pode comprar filtros ao preço de R\$ 600,00 a unidade e, assim, aumentar 1 ponto sua nota da natureza.

2.6. *Produtos Recicláveis*

A empresa tem a opção de comprar produtos de consumo recicláveis e assim ter a oportunidade de colaborar com a preservação da Natureza e, assim, aumentar sua nota. O preço da compra de produtos recicláveis é de R\$ 900,00 e aumenta a nota em 3 pontos.

2.7. *Programas de Eco Eficiência*

Programas de eco eficiência podem ser muito genéricos como pesquisas de novas formas de geração de energia, mudanças de hábitos internos e formas de produção. Se o jogador investir neste projeto gasta R\$ 1.200,00 e tem a nota da natureza aumentada em 4 pontos.

Além de todas estas variáveis, o jogo tem um custo diário operacional de R\$ 3.000,00 que é diluído durante o dia. Se o usuário não mantiver sua empresa com boas práticas socioambientais, sua nota vai diminuir e, seu custo operacional vai estar próximo dos 3.000,00 ou mesmo, este valor se for o próprio 0 mas, terá um valor inverso, quase 0, se mantiver uma nota da natureza com valor alto, próximo do 10.

O preço do custo do produto de venda, apesar de menor grau, também é atingido pelo aumento da nota da Natureza, sendo assim, se o jogador não investir na natureza, terá dificuldade em lucrar.

Em paralelo ao desenvolvimento a camada de negócios (Model) foi desenvolvido a camada de controle (Control). Apesar do projeto foi pequeno, foi respeitado as definições do *design pattern* MVC e, para isso, foi criado uma classe para que a camada de visualização chama-se as ações encapsuladas nos métodos da camada de negócio.

Para teste dos métodos da camada de controle, foi desenvolvido uma camada de visualização e, esta, foi criada uma classe, em console, para chamada dos métodos e, simulação do jogo, tudo em modo texto.

Esta classe teste, foi descartada no fim do projeto, pois, foi anexada a camada de visualização GUI originalmente projetada e, que foi desenvolvida em paralelo.

Em paralelo ao desenvolvido da camada de negócios e controle, foi desenvolvido a camada de visualização. Esta camada foi desenvolvida em um ambiente de programação separada. Esta tela foi estada, sem nenhum evento, apenas com o desenho em tela e, posteriormente, anexado ao projeto principal.

Quando anexada ao projeto principal, criaram-se as ações e viu-se a necessidade de criar algumas classes, como segue abaixo a explicação.

Existe uma área no jogo, que apresenta a fábrica nos diferentes momentos do dia e, para isso, mantemos em disco várias imagens, representando estes estados. A camada de controle recebe da camada de modelo à informação do momento em que está no dia para devida atualização da informação da tela e, assim, apresenta a imagem correta.

Para isso, foi necessário criar uma classe chamada *PainelComImagem* que estende *Jpanel* que tem um método especial chamado *setImagem* que, recebe a imagem que queremos atribuir.

Baseado na classe `Timer` do `Swing`, criamos uma classe chamada *AgendadorTarefa* que, faz a chamada a cada 2 segundos do método da classe controle, chamando uma “nova jogada” para que, faça toda a movimentação necessária do saldo, redução da nota da empresa, etc.

A classe principal possui todos os componentes visuais e instancia a classe *AgendadorTarefa* e a classe de controle e, o usuário sem executar nenhuma ação, consegue perceber o jogo se movimentar em sua linha temporal, e, assim, o jogador, pode interagir com esse.

A fase final do desenvolvimento do jogo foi os testes do mesmo, onde foram feitos pequenos ajustes visuais e de usabilidade do mesmo. Após o término do projeto ficou claro que o padrão MVC foi de grande utilidade pois:

- Podem-se dividir fases do projeto em desenvolvimento em paralelo;
- Cada fase do projeto pode ser bem testada em paralelo;
- Na junção das linhas de desenvolvimento (regras de negócio, controle e visualização) não houve dificuldades técnicas;
- As regras de negócio ficaram muito bem encapsuladas na camada de modelo.

O Jogo pode ser baixado e executado no link abaixo:

www.marciofcruz.com/aps/trabalhoapsjogo.zip

3. ESTRUTURA DO PROJETO

3.1. Divisão Tipos:

PeriodoDia.java:

Classe enumerada que tem o conjunto do periodo dia de acordo com o horário do dia.

TipolInvestimentoNatureza.java:

Classe enumerada com os tipos de investimento de natureza suportados pelo jogo.

3.2. Divisão Model:

ParametrosCalculo.java

Esta classe de parametrização é auxiliar da classe de negócio do sistema.

Elementos privados:

- custoMateriaPrimaPorUnidade
- despesaBaseDia
- fatorProducaoProdutoAcabado
- precoUnitarioVendaPorUnidade
- reducaoBaseDia
- reducaoBaseHora
- reducaoBaseJogada
- getFatorMultiplicadorCusto(float):float
- getPorcentagemCustoBaseEmpresa(float):float

Elementos públicos:

- getCustoInvestimento(TipolInvestimentoNatureza):float
- getCustoMateriaPrimaCompra():float
- getCustoMateriaPrimaPorUnidade(float):float
- getDespesaBaseDia(float):float

- getDespesaBaseJogada(float):float
- getFatorMultiplicadorCusto(float):float
- getPrecoUnitarioVendaPorUnidade():float
- getReducaoBaseJogada():float

ModelEmpresa.java

Esta classe está centralizada todas as regras de negócio do jogo.

Elementos privados:

- dataAtual
- dataFimJogo
- despesaDia
- diasOperacao
- estoqueMateriaPrima
- estoqueProdutoAcabado
- fimDeSemana
- horarioComercial
- notaNatureza
- parametrosCalculo
- periodoDia
- receitaDia
- saldoDiaAnterior
- addNotaNatureza(float):void

- getDespesaBaseJogada():float
- getCustoOperacionalProducao(int):float

Elementos públicos:

- comprarMateriaPrima(int):void
- fabricarProdutoAcabado(int):void
- getCustoFabricaDia():float
- getDataAtual():Calendar
- getDespesaDia():float
- getDiasOperacao():int
- getEstoqueAtualMateriaPrima():Integer
- getEstoqueAtualProdutoAcabado():Integer
- getfimDeSemana():boolean
- getGameOver():boolean
- getHorarioComercial():boolean
- getNegociacaoemAberto():boolean
- getNotaNatureza():float
- getPeriodoDia():PeriodoDia
- getPrecoCustoProdutoAcabado():float
- getPrecoVenda():float
- getReceitaDia():float
- getSaldoAtual():float
- getSaldoDiaAnterior():float

- `iniciarJogo():void`
- `investirSustentabilidade(TipoInvestimentoSustentabilidade):void`
- `novaJogada():void`
- `podeMovimentar():String`
- `setDataAtual(Calendar):void`
- `venderProdutoAcabado(int):void`

3.3. Divisão Controller:

ControllerEmpresa.java

Esta classe é recebe as requisições da camada de visualização e as atende junto à camada de negócio.

Elementos privados:

- `camadaModel`

Elementos públicos:

Todos os métodos tem retorno *String*, pois pode retornar mensagem que são tratadas pela camada de visualização, sejam elas valores previamente formatados ou mesmo mensagens de erros devido a operações indevidas.

- `comprarMateriaPrima():String`
- `ecoEficiencia():String`
- `executarOrdemProducao():String`
- `filtroFumaca():String`
- `gameOver():String`

- getCustoFabricaDia():String
- getDataAtual():String
- getDespesaDia():String
- getDiasOperacao():String
- getEstoqueAtualMateriaPrima():String
- getEstoqueAtualProdutoAcabado():String
- getFimDeSemana():String
- getHorarioComercial():String
- getNegociacaoEmAberto():String
- getNotaNatureza():String
- getPeriodoDia():String
- getPrecoCustoProdutoAcabado():String
- getPrecoVenda():String
- getReceitaDia():String
- getSaldoAtual():String
- getSaldoAtualDiaAnterior():String
- palestraSustentabilidade():String
- reciclagem():String
- temEstoqueMateriaPrima():String
- temEstoqueProdutoAcabado():String
- venderProdutoAcabado():String

3.4. Divisão View:

PainelComImagem.java:

Esta classe é responsável pela exibição do painel da imagem e das informações da nota da natureza, saldo da empresa, data atual, etc.

Elementos privados:

- imagemFundo
- mensagemAviso
- mensagemFimJogo
- notaNatureza
- saldoAtual
- operacao
- estoqueMateriaPrima
- estoqueProdutoAcabado
- dataCompleta

Elementos públicos:

- getNotaNatureza():String
- getSaldoAtual():String
- paintComponent(Graphics):void
- setImagem(BufferedImage):void
- getDataCompleta():String
- getEstoqueMateriaPrima():String
- setDataCompleta(String)

- `getOperacao():String`
- `setEstoqueMateriaPrima(String):void`
- `setEstoqueProdutoAcabado(String):void`

AtualizadorGUI.java

Esta classe faz uma solicitação a camada controle e pega os valores atualizados da situação atual do jogo e, atualiza os métodos da classe *PainelComImagem()*.

Elementos privados:

- `controllerEmpresa`
- `imagemAmanhecer`
- `imagemInicioAnoitecer`
- `imagemManha08diaUtil`
- `imagemManha08fds`
- `imagemManha09diaUtil`
- `imagemManha09fds`
- `imagemManha10diaUtil`
- `imagemManha10fds`
- `imagemManha11diaUtil`
- `imagemManha11fds`
- `imagemManha12diaUtil`
- `imagemManha12fds`
- `imagemNoite`

- imagemTarde13diaUtil
- imagemTarde13fds
- imagemTarde14diaUtil
- imagemTarde14fds
- imagemTarde15diaUtil
- imagemTarde15fds
- imagemTarde16diaUtil
- imagemTarde16fds
- imagemTarde17diaUtil
- imagemTarde17fds
- imagemTarde18diaUtil
- imagemTarde18fds
- pathImagem
- viewGame
- carregarImagemFabrica(BufferedImage):void

Elementos públicos:

- getPathImagem():String

AgendadorTarefa.java

Esta classe utiliza os recursos do Timer da API Swing para que chame a atualização sequencial das jogadas.

Elementos privados:

- atualizarGUI
- controllerEmpresa
- timer
- novaJogada():void

ViewGame.java

Elementos privados:

- controllerEmpresa

Componentes visuais:

- arealmagem
- btnComprarMateriaPrima
- btnEcoEficiencia
- btnFiltroFumaca
- btnOrdemProducao
- btnPalestraSustentabilidade
- btnReciclagem
- btnVenderProdutoAcabado

Elementos públicos:

- actionPerformed(ActionEvent):void
- janela:void

4. LINHAS DE CODIGO DO JOGO

```
package controller;

import java.text.DecimalFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;

import tipos.PeriodoDia;
import tipos.TipoInvestimentoNatureza;

import model.ModelEmpresa;

public class ControllerEmpresa {

    private DecimalFormat df = new DecimalFormat("###,##0.00");
    SimpleDateFormat sdf = new SimpleDateFormat("E dd/MM/yyyy HH:mm");

    private ModelEmpresa camadaModel = new ModelEmpresa();

    public ControllerEmpresa() throws ParseException {
        camadaModel.iniciarJogo();
    }

    public String getFimDeSemana() {
        if (camadaModel.getfimDeSemana()) {
            return "Fim de Semana";
        } else {
            return "";
        }
    }

    public String palestraSustentabilidade() {
        try {

            camadaModel.investirSustentabilidade(TipoInvestimentoNatureza.palestraSustentabilidade);
            return "";
        } catch (RuntimeException e) {
            return e.getMessage();
        }
    }

    public String filtroFumaca(){
        try {

            camadaModel.investirSustentabilidade(TipoInvestimentoNatureza.filtroFumaca);
            return "";
        } catch (RuntimeException e) {
            return e.getMessage();
        }
    }

    public String reciclagem(){
        try {

            camadaModel.investirSustentabilidade(TipoInvestimentoNatureza.reciclagem);
            return "";
        } catch (RuntimeException e) {
```

```

        return e.getMessage();
    }
}

public String ecoEficiencia(){
    try {

        camadaModel.investirSustentabilidade(TipoInvestimentoNatureza.ecoEficiencia);
        return "";
    } catch (RuntimeException e) {
        return e.getMessage();
    }
}

public String comprarMateriaPrima() {
    try {
        camadaModel.comprarMateriaPrima(50);
        return "";
    } catch (RuntimeException e) {
        return e.getMessage();
    }
}

public String executarOrdemProducao() {
    try {
        camadaModel.fabricarProdutoAcabado(5);
        novaJogada();
        return "";
    } catch (RuntimeException e) {
        return e.getMessage();
    }
}

public String venderProdutoAcabado() {
    try {
        camadaModel.venderProdutoAcabado(5);
        return "";
    } catch (RuntimeException e) {
        return e.getMessage();
    }
}

public String getNotaNatureza() {
    float auxiliar = camadaModel.getNotaNatureza();

    if (auxiliar > 10) {
        auxiliar = 10;
    }

    return df.format(auxiliar);
}

public String getPrecoCustoProdutoAcabado() {
    return "R$ "+df.format(camadaModel.getPrecoCustoProdutoAcabado());
}

public String getPrecoVenda() {
    return "R$ "+df.format(camadaModel.getPrecoVenda());
}

```

```

public String getCustoFabricaDia() {
    return "R$ "+df.format(camadaModel.getCustoFabricaDia());
}

public boolean gameOver() {
    return camadaModel.getGameOver();
}

public String getEstoqueAtualMateriaPrima() {
    return camadaModel.getEstoqueAtualMateriaPrima().toString()+" Kg";
}

public String getEstoqueAtualProdutoAcabado() {
    return camadaModel.getEstoqueAtualProdutoAcabado().toString()+ " Pç";
}

public boolean temEstoqueMateriaPrima() {
    return camadaModel.getEstoqueAtualMateriaPrima() != 0;
}

public boolean temEstoqueProdutoAcabado() {
    return camadaModel.getEstoqueAtualProdutoAcabado() != 0;
}

public String getSaldoDiaAnterior() {
    return df.format(camadaModel.getSaldoDiaAnterior());
}

public String getDespesaDia() {
    return df.format(camadaModel.getDespesaDia());
}

public String getReceitaDia() {
    return df.format(camadaModel.getReceitaDia());
}

public String getSaldoAtual() {
    return df.format(camadaModel.getSaldoAtual());
}

public int getDiasOperacao() {
    return camadaModel.getDiasOperacao();
}

public String getDataAtual() {
    return sdf.format(camadaModel.getDataAtual().getTime());
}

public String novaJogada() {
    try {
        camadaModel.novaJogada();
        return "";
    } catch (RuntimeException e) {
        return e.getMessage();
    }
}

public PeriodoDia getPeriodoDia() {
    return camadaModel.getPeriodoDia();
}

```

```

        public String getHorarioComercial() {
            if (camadaModel.getfimDeSemana()) {
                return "Fim de Semana";
            } else {
                if (camadaModel.gethorarioComercial()) {
                    return "Horário Comercial";
                } else {
                    return "Fora do Horário Comercial";
                }
            }
        }

        public boolean getNegociacaoEmAberto() {
            return camadaModel.getNegociacaoemAberto();
        }
    }

package model;

import java.text.ParseException;
import java.util.Calendar;
import java.util.GregorianCalendar;

import tipos.PeriodoDia;
import tipos.TipoInvestimentoNatureza;

public class ModelEmpresa {

    ParametrosCalculo parametrosCalculo = new ParametrosCalculo();

    private boolean fimDeSemana;
    private boolean horarioComercial;

    private float notaNatureza = 5;

    private void addNotaNatureza(float valor) {
        float auxiliar = notaNatureza + valor;

        if (auxiliar > 11) {
            notaNatureza = 11;
        }
        else if (auxiliar < 0) {
            notaNatureza = 0;
        } else {
            notaNatureza = auxiliar;
        }
    }

    public float getNotaNatureza() {
        return notaNatureza;
    }

    private int diasOperacao; // Qtde de dias que a empresa esta funcionando

    private Calendar dataFimJogo = new GregorianCalendar();
    private Calendar dataAtual = new GregorianCalendar();

    private float saldoDiaAnterior = 0;
    private float receitaDia = 0;

```



```

private float despesaDia = 0;

private int estoqueMateriaPrima = 0;
private int estoqueProdutoAcabado = 0;

private PeriodoDia periodoDia;

public Integer getEstoqueAtualMateriaPrima() {
    return estoqueMateriaPrima;
}

public Integer getEstoqueAtualProdutoAcabado() {
    return estoqueProdutoAcabado;
}

private boolean podeMovimentar(String mensagemBloqueio) {
    boolean auxiliar = false;

    if (!getGameOver()) {
        if (!getNegociacaoemAberto()) {
            throw new RuntimeException(mensagemBloqueio);
        }

        auxiliar = true;
    }

    return auxiliar;
}

public void investirSustentabilidade(
    TipoInvestimentoNatureza tipoInvestimentoNatureza) {
    if (podeMovimentar("Fora do horário comercial!")) {
        int qtde = 1;

        if (getNotaNatureza() >= 10) {
            throw new RuntimeException(
                "Não é necessário investir em Sustentabilidade neste
momento!");
        }

        float custoOperacao = parametrosCalculo
            .getCustoInvestimento(tipoInvestimentoNatureza) * qtde;

        if ((getSaldoAtual() - custoOperacao) < 0) {
            throw new RuntimeException(
                "Não há saldo disponível para realizar esta
operação!");
        }

        float aumentoNota = custoOperacao / 600; // razão de aumento da nota

        // da Natureza em função

        // do investimento

        addNotaNatureza(aumentoNota);

        despesaDia = despesaDia + custoOperacao;
    }
}

```

```

    }

    public void comprarMateriaPrima(int qtde) {
        if (podeMovimentar("Fornecedor está fechado!")) {
            float custoOperacao = parametrosCalculo
                .getCustoMateriaPrimaCompra() * qtde;

            if ((getSaldoAtual() - custoOperacao) < 0) {
                throw new RuntimeException(
                    "Não há saldo disponível para realizar esta
operação!");
            }

            estoqueMateriaPrima = estoqueMateriaPrima + qtde;

            despesaDia = despesaDia + custoOperacao;
        }
    }

    public float getPrecoVenda() {
        return parametrosCalculo.getPrecoUnitarioVendaPorUnidade();
    }

    private float getCustoOperacionalProducao(int qtdeProdutoAcabado) {
        return parametrosCalculo.getCustoMateriaPrimaPorUnidade(notaNatureza)
            * qtdeProdutoAcabado;
    }

    public void fabricarProdutoAcabado(int qtdeProdutoAcabado) {
        if (podeMovimentar("Fabrica está fechada neste momento!")) {
            int qtdeNecessariaMateriaPrima = qtdeProdutoAcabado
                * parametrosCalculo.getFatorProducaoProdutoAcabado();

            float custoOperacionalProducao =
getCustoOperacionalProducao(qtdeProdutoAcabado);

            if ((getSaldoAtual() - custoOperacionalProducao) < 0) {
                throw new RuntimeException(
                    "Não há saldo suficiente para executar ordem de
produção!");
            }

            if (estoqueMateriaPrima - qtdeNecessariaMateriaPrima < 0) {
                throw new RuntimeException(
                    "Não há estoque suficiente de Materia Prima para
produção!");
            }

            // produzir as peças
            estoqueProdutoAcabado = estoqueProdutoAcabado + qtdeProdutoAcabado;
            estoqueMateriaPrima = estoqueMateriaPrima
                - qtdeNecessariaMateriaPrima;

            // calcular nova despesa
            despesaDia = despesaDia + custoOperacionalProducao;

```

```

        addNotaNatureza(-1 * parametrosCalculo.getReducaoBaseJogada());
    }
}

public void venderProdutoAcabado(int qtdeProdutoAcabadoVenda) {
    if (podeMovimentar("Fabrica está fechada neste momento!")) {
        if (estoqueProdutoAcabado - qtdeProdutoAcabadoVenda < 0) {
            throw new RuntimeException(
                "Não há estoque produto de produto acabado");
        }

        float auxiliarTotalVenda = qtdeProdutoAcabadoVenda
            * parametrosCalculo.getPrecoUnitarioVendaPorUnidade();

        receitaDia = receitaDia + auxiliarTotalVenda;
        estoqueProdutoAcabado = estoqueProdutoAcabado
            - qtdeProdutoAcabadoVenda;
    }
}

public boolean getNegociacaoemAberto() {
    if (fimDeSemana || periodoDia == PeriodoDia.amanhecer
        || periodoDia == PeriodoDia.inicioAnoitecer
        || periodoDia == PeriodoDia.noite) {
        return false;
    } else {
        return true;
    }
}

private float getDespesaBaseJogada() {
    float auxiliar = parametrosCalculo.getDespesaBaseJogada(notaNatureza);

    // de final de semana e fora do horário comercial, a empresa gasta 80%
    // menos
    // dinheiro
    if (!getNegociacaoemAberto()) {
        auxiliar = auxiliar * 0.8f; // a Despesa a noite é 80% menor
    }

    return auxiliar;
}

public float getSaldoDiaAnterior() {
    return saldoDiaAnterior;
}

public float getSaldoAtual() {
    return saldoDiaAnterior + receitaDia - despesaDia;
}

public float getDespesaDia() {
    return despesaDia;
}

public float getReceitaDia() {

```

```

        return receitaDia;
    }

    public boolean getfimDeSemana() {
        return fimDeSemana;
    }

    public boolean gethorarioComercial() {
        return horarioComercial;
    }

    public void iniciarJogo() throws ParseException {
        diasOperacao = 1;

        dataAtual.set(1992, 6, 15, 8, 0); // Fim da Eco 92

        setDataAtual(dataAtual);

        dataFimJogo.set(2002, 8, 26, 0, 0); // Inicio Cúpula Mundial sobre
Desenvolvimento Sustentável //

        saldoDiaAnterior = 10000f;
        receitaDia = 0;
        despesaDia = 0;
    }

    public Calendar getDataAtual() {
        return dataAtual;
    }

    public void novaJogada() {
        if (!getGameOver()) {
            Calendar anterior = (Calendar) dataAtual.clone();

            int somaMinuto = 15;
            int multiplicadorReceitaDespesa = 1;

            if (getfimDeSemana()) {
                multiplicadorReceitaDespesa = 8;
            } else if (!gethorarioComercial()) {
                multiplicadorReceitaDespesa = 4;
            }

            somaMinuto = somaMinuto * multiplicadorReceitaDespesa;

            dataAtual.add(Calendar.MINUTE, somaMinuto);

            if (dataAtual.get(Calendar.DAY_OF_YEAR) != anterior
                .get(Calendar.DAY_OF_YEAR)) {
                saldoDiaAnterior = getSaldoAtual();
                receitaDia = 0;
                despesaDia = 0;

                diasOperacao++;
            } else {

                float auxiliarDespesa = getDespesaBaseJogada()
                    * multiplicadorReceitaDespesa;
            }
        }
    }

```

```

        if ((getSaldoAtual() - auxiliarDespesa) < 0) {
            throw new RuntimeException("Você perdeu");
        }

        despesaDia = despesaDia + auxiliarDespesa;
    }

    // redução constante da nota da natureza em qualquer dia e horário
    addNotaNatureza(-1 * parametrosCalculo.getReducaoBaseJogada());

    setDataAtual(dataAtual);

} else {
    throw new RuntimeException("Você perdeu");
}
}

private void setDataAtual(Calendar gc) {
    dataAtual = gc;

    int hora = dataAtual.get(Calendar.HOUR_OF_DAY);
    int diaDaSemana = dataAtual.get(Calendar.DAY_OF_WEEK);

    // 1/3 - setando a variável de final de semana -
    fimDeSemana = (diaDaSemana == 1) || (diaDaSemana == 7);

    // 2/3 - identificando o período do dia (vai ser usado na cena, se o
    // programador da preferir
    if ((hora >= 0 && hora <= 6) || (hora >= 20)) {
        periodoDia = PeriodoDia.noite;
    } else if (hora >= 6 && hora <= 7) {
        periodoDia = PeriodoDia.amanhecer;
    } else if (hora >= 7 && hora <= 8) {
        periodoDia = PeriodoDia.manha08;
    } else if (hora >= 8 && hora <= 9) {
        periodoDia = PeriodoDia.manha09;
    } else if (hora >= 9 && hora <= 10) {
        periodoDia = PeriodoDia.manha10;
    } else if (hora >= 10 && hora <= 11) {
        periodoDia = PeriodoDia.manha11;
    } else if (hora >= 11 && hora <= 12) {
        periodoDia = PeriodoDia.manha12;
    } else if (hora >= 12 && hora <= 13) {
        periodoDia = PeriodoDia.tarde13;
    } else if (hora >= 13 && hora <= 14) {
        periodoDia = PeriodoDia.tarde14;
    } else if (hora >= 14 && hora <= 15) {
        periodoDia = PeriodoDia.tarde15;
    } else if (hora >= 15 && hora <= 16) {
        periodoDia = PeriodoDia.tarde16;
    } else if (hora >= 16 && hora <= 17) {
        periodoDia = PeriodoDia.tarde17;
    } else if (hora >= 17 && hora <= 18) {
        periodoDia = PeriodoDia.tarde18;
    } else {
        periodoDia = PeriodoDia.inicioAnoitecer;
    }

    // 3/3 - Identificar se é horário comercial
    horarioComercial = (hora >= 8 && hora < 18);
}

```

```

    }

    public PeriodoDia getPeriodoDia() {
        return periodoDia;
    }

    public boolean getFimDeSemana() {
        return fimDeSemana;
    }

    public boolean getGameOver() {
        float saldoFinal = getSaldoAtual() - getDespesaBaseJogada();

        return saldoFinal <= 0;
    }

    public int getDiasOperacao() {
        return diasOperacao;
    }

    public float getCustoFabricaDia() {
        return parametrosCalculo.getDespesaBaseDia(notaNatureza);
    }

    public float getPrecoCustoProdutoAcabado() {
        float precoCusto = parametrosCalculo.getCustoMateriaPrimaCompra()
            * parametrosCalculo.getFatorProducaoProdutoAcabado();

        return precoCusto + getCustoOperacionalProducao(1);
    }
}

package model;

import tipos.TipoInvestimentoNatureza;

public class ParametrosCalculo {

    private float despesaBaseDia = 3000;

    // custo base da materia prima
    private final float custoMateriaPrimaPorUnidade = 4;

    private final float precoUnitarioVendaPorUnidade = 100;

    private final int fatorProducaoProdutoAcabado = 5; // Produção: São

        // necessário 10

    // redução da nota da natureza
    private final float reducaoBaseDia = 3;
    private final float reducaoBaseHora = reducaoBaseDia / 24;
    private final float reducaoBaseJogada = reducaoBaseHora / 4;

    // valores da Sustentabilidade
    public float getCustoInvestimento(
        TipoInvestimentoNatureza tipoInvestimentoNatureza) {
        switch (tipoInvestimentoNatureza) {
            case palestraSustentabilidade:
                return 300;

```

```

        case filtroFumaca:
            return 600;
        case reciclagem:
            return 900;
        case ecoEficiencia:
            return 1200;
        default:
            return 0;
    }
}

public float getReducaoBaseJogada() {
    return reducaoBaseJogada;
}

public float getCustoMateriaPrimaPorUnidade(float notaNatureza) {
    float auxiliar1 = custoMateriaPrimaPorUnidade;
    float auxiliar2 = getFatorMultiplicadorCusto(notaNatureza) * auxiliar1;

    return auxiliar2;
}

public float getCustoMateriaPrimaCompra() {
    return custoMateriaPrimaPorUnidade;
}

public int getFatorProducaoProdutoAcabado() {
    return fatorProducaoProdutoAcabado;
}

private float getFatorMultiplicadorCusto(float notaNatureza) {
    /*
     * Nota de 0 até menor ou igual a 5: aumento do gasto Nota igual a 5:
     * não á fator de cálculo Nota maior que 5: redução do gasto
     */

    if (notaNatureza >= 10) {
        notaNatureza = 10;
    }

    float auxiliar = notaNatureza - 5;

    auxiliar = 1 - (auxiliar / 5);

    if (auxiliar == 0.00) {
        auxiliar = 0.1f;
    }

    return auxiliar;
}

private float getPorcentagemCustoBaseEmpresa(float notaNatureza) {
    if (notaNatureza >= 10) {
        return 0.01f;
    }
    else if (notaNatureza <= 0) {
        return 100;
    } else {
        /*

```

```

        * Exemplo de cálculo: Para Nota 10: Base Porcentagem é 0; Para nota
        * 5: Base Porcentagem é 50 Para nota 0: Base Porcentagem é 100
        */
        float parte1 = notaNatureza / 10;

        float parte2 = 100 - (parte1 * 100);

        return parte2;
    }
}

public float getPrecoUnitarioVendaPorUnidade() {
    return precoUnitarioVendaPorUnidade;
}

public float getDespesaBaseJogada(float notaNatureza) {
    // despesa base da Jogada
    float despesaBaseHora = despesaBaseDia / 24;

    float parte1 = despesaBaseHora / 4;

    float parte2 = (getPorcentagemCustoBaseEmpresa(notaNatureza) / 100)
        * parte1;

    return parte2;
}

public float getDespesaBaseDia(float notaNatureza) {
    return (getPorcentagemCustoBaseEmpresa(notaNatureza) / 100)
        * despesaBaseDia;
}

}
package tipos;

public enum PeriodoDia {
    amanhecer, manha08, manha09, manha10, manha11, manha12,
    tarde13, tarde14, tarde15, tarde16, tarde17, tarde18, inicioAnoitecer, noite;

    public static String getDescricao(PeriodoDia periodoDia) {
        switch (periodoDia) {
            case amanhecer:
                return "Amanhecer";
            case tarde13:
                return "Tarde 13";
            case tarde14:
                return "Tarde 14";
            case tarde15:
                return "Tarde 15";
            case tarde16:
                return "Tarde 16";
            case tarde17:
                return "Tarde 17";
            case tarde18:
                return "Tarde 18";
            case inicioAnoitecer:
                return "Inicio Anoitecer";
            case manha08:
                return "Manhã 08";

```



```

        case manha09:
            return "Manhã 09";
        case manha10:
            return "Manhã 10";
        case manha11:
            return "Manhã 11";
        case manha12:
            return "Manhã 12";
        case noite:
            return "Noite";
        default:
            return null;
    }
}

package tipos;

public enum TipoInvestimentoNatureza {
    palestraSustentabilidade, filtroFumaca, reciclagem, ecoEficiencia;
}

package view;

import java.awt.event.ActionListener;
import java.io.IOException;

import javax.swing.Timer;

import controller.ControllerEmpresa;

/**
 * Camada: View; Classe responsável para chamar o timer do sistema, fazer nova
 * jogada e fazer a chamada para atualização dos componentes
 */

public class AgendadorTarefa {

    private Timer timer;
    private ControllerEmpresa controllerEmpresa;
    private AtualizarGUI atualizarGUI;

    public AgendadorTarefa(ControllerEmpresa camadaController,
        AtualizarGUI atualizarGUI) {
        super();

        ActionListener action = new ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent e) {

                try {
                    novaJogada();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
        };

        this.atualizarGUI = atualizarGUI;
    }
}

```

```

        this.timer = new Timer(1500, action); // rodar a cada 3 segundos
        timer.start();

        this.controllerEmpresa = camadaController;
    }

    private void novaJogada() throws IOException {
        String retorno = controllerEmpresa.novaJogada();
        timer.stop();

        atualizarGUI.atualizar(retorno);

        if (retorno.isEmpty()) {
            timer.start();
        }
    }
}

package view;

import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;

import controller.ControllerEmpresa;

/**
 * Camada: View;
 * Classe responsável para atualizar os componentes GUI na tela;
 */

public class AtualizarGUI {

    private ViewGame viewGame;
    private ControllerEmpresa controllerEmpresa;
    private String pathImagem = System.getProperty("user.dir") + "\\imagens\\";

    private BufferedImage imagemAmanhecer;
    private BufferedImage imagemInicioAnoitecer;
    private BufferedImage imagemNoite;
    private BufferedImage imagemManha08diaUtil;
    private BufferedImage imagemManha08fds;
    private BufferedImage imagemManha09diaUtil;
    private BufferedImage imagemManha09fds;
    private BufferedImage imagemManha10diaUtil;
    private BufferedImage imagemManha10fds;
    private BufferedImage imagemManha11diaUtil;
    private BufferedImage imagemManha11fds;
    private BufferedImage imagemManha12diaUtil;
    private BufferedImage imagemManha12fds;
    private BufferedImage imagemTarde13diaUtil;
    private BufferedImage imagemTarde13fds;
    private BufferedImage imagemTarde14diaUtil;
    private BufferedImage imagemTarde14fds;
    private BufferedImage imagemTarde15diaUtil;
    private BufferedImage imagemTarde15fds;

```

```

private BufferedImage imagemTarde16diaUtil;
private BufferedImage imagemTarde16fds;
private BufferedImage imagemTarde17diaUtil;
private BufferedImage imagemTarde17fds;
private BufferedImage imagemTarde18diaUtil;
private BufferedImage imagemTarde18fds;

public String getPathImagem() {
    return pathImagem;
}

public AtualizarGUI(ViewGame viewGame, ControllerEmpresa controllerEmpresa) {
    super();

    try {
        imagemAmanhecer = ImageIO.read(new
File(pathImagem+"amanhecer.png"));
        imagemInicioAnoitecer = ImageIO.read(new
File(pathImagem+"inicioanoitecer.png"));
        imagemNoite = ImageIO.read(new File(pathImagem+"noite.png"));
        imagemManha08diaUtil = ImageIO.read(new
File(pathImagem+"manha08diautil.png"));
        imagemManha08fds = ImageIO.read(new
File(pathImagem+"manha08fds.png"));
        imagemManha09diaUtil = ImageIO.read(new
File(pathImagem+"manha09diautil.png"));
        imagemManha09fds = ImageIO.read(new
File(pathImagem+"manha09fds.png"));
        imagemManha10diaUtil = ImageIO.read(new
File(pathImagem+"manha10diautil.png"));
        imagemManha10fds = ImageIO.read(new
File(pathImagem+"manha10fds.png"));
        imagemManha11diaUtil = ImageIO.read(new
File(pathImagem+"manha11diautil.png"));
        imagemManha11fds = ImageIO.read(new
File(pathImagem+"manha11fds.png"));
        imagemManha12diaUtil = ImageIO.read(new
File(pathImagem+"manha12diautil.png"));
        imagemManha12fds = ImageIO.read(new
File(pathImagem+"manha12fds.png"));
        imagemTarde13diaUtil = ImageIO.read(new
File(pathImagem+"tarde13diautil.png"));
        imagemTarde13fds = ImageIO.read(new File(pathImagem+"tarde13fds.png"));
        imagemTarde14diaUtil = ImageIO.read(new
File(pathImagem+"tarde14diautil.png"));
        imagemTarde14fds = ImageIO.read(new File(pathImagem+"tarde14fds.png"));
        imagemTarde15diaUtil = ImageIO.read(new
File(pathImagem+"tarde15diautil.png"));
        imagemTarde15fds = ImageIO.read(new File(pathImagem+"tarde15fds.png"));
        imagemTarde16diaUtil = ImageIO.read(new
File(pathImagem+"tarde16diautil.png"));
        imagemTarde16fds = ImageIO.read(new File(pathImagem+"tarde16fds.png"));
        imagemTarde17diaUtil = ImageIO.read(new
File(pathImagem+"tarde17diautil.png"));
        imagemTarde17fds = ImageIO.read(new File(pathImagem+"tarde17fds.png"));
        imagemTarde18diaUtil = ImageIO.read(new
File(pathImagem+"tarde18diautil.png"));
        imagemTarde18fds = ImageIO.read(new File(pathImagem+"tarde18fds.png"));
    } catch (IOException e) {
        // TODO Auto-generated catch block

```

```

        e.printStackTrace();
    }

    this.viewGame = viewGame;
    this.controllerEmpresa = controllerEmpresa;

    carregarImagemFabrica(imagemManha08diaUtil);
}

private void carregarImagemFabrica(BufferedImage imagem) {
    viewGame.arealImagem.setImagem(imagem);
}

public void atualizar(String mensagem) throws IOException {
    boolean gameOver = controllerEmpresa.gameOver();

    if (gameOver) {
        viewGame.arealImagem.setMensagemFimJogo("Fim jogo: Não há saldo
suficiente para continuar a empresa em funcionamento!");
    }
    else {
        if (!mensagem.isEmpty()) {
            viewGame.arealImagem.setMensagemFimJogo(mensagem);
        }
    }

    viewGame.arealImagem.setNotaNatureza(controllerEmpresa.getNotaNatureza());
    viewGame.arealImagem.setSaldoAtual(controllerEmpresa.getSaldoAtual());

    viewGame.arealImagem.setEstoqueMateriaPrima(controllerEmpresa.getEstoqueAtualMateriaPrima());

    viewGame.arealImagem.setEstoqueProdutoAcabado(controllerEmpresa.getEstoqueAtualProdutoAcab
ado());

    viewGame.arealImagem.setDataCompleta(controllerEmpresa.getDataAtual());
    viewGame.arealImagem.setOperacao(controllerEmpresa.getHorarioComercial());

    // atualizar os labels
    boolean negociacaoEmAberto = controllerEmpresa.getNegociacaoEmAberto() && !
gameOver;

    viewGame.btnComprarMateriaPrima.setEnabled(negociacaoEmAberto);
    viewGame.btnOrdemProducao.setEnabled(negociacaoEmAberto &&
controllerEmpresa.temEstoqueMateriaPrima());
    viewGame.btnVenderProdutoAcabado.setEnabled(negociacaoEmAberto &&
controllerEmpresa.temEstoqueProdutoAcabado());
    viewGame.btnPalestraSustentabilidade.setEnabled(negociacaoEmAberto);
    viewGame.btnFiltroFumaca.setEnabled(negociacaoEmAberto);
    viewGame.btnReciclagem.setEnabled(negociacaoEmAberto);
    viewGame.btnEcoEficiencia.setEnabled(negociacaoEmAberto);

    // fazer um teste aqui para carregar uma imagem dinamicamente

    if (controllerEmpresa.getFimDeSemana().isEmpty()) {

        switch (controllerEmpresa.getPeriodoDia()) {
            case amanhecer:
                carregarImagemFabrica(imagemAmanhecer);
                break;

```

```

        case inicioAnoitecer:
            carregarImagemFabrica(imagemInicioAnoitecer);
            break;
        case manha08:
            carregarImagemFabrica(imagemManha08diaUtil);
            break;
        case manha09:
            carregarImagemFabrica(imagemManha09diaUtil);
            break;
        case manha10:
            carregarImagemFabrica(imagemManha10diaUtil);
            break;
        case manha11:
            carregarImagemFabrica(imagemManha11diaUtil);
            break;
        case manha12:
            carregarImagemFabrica(imagemManha12diaUtil);
            break;
        case noite:
            carregarImagemFabrica(imagemNoite);
            break;
        case tarde13:
            carregarImagemFabrica(imagemTarde13diaUtil);
            break;
        case tarde14:
            carregarImagemFabrica(imagemTarde14diaUtil);
            break;
        case tarde15:
            carregarImagemFabrica(imagemTarde15diaUtil);
            break;
        case tarde16:
            carregarImagemFabrica(imagemTarde16diaUtil);
            break;
        case tarde17:
            carregarImagemFabrica(imagemTarde17diaUtil);
            break;
        case tarde18:
            carregarImagemFabrica(imagemTarde18diaUtil);
            break;
        default:
            break;
    }
}
else {
    switch (controllerEmpresa.getPeriodoDia()) {
        case amanhecer:
            carregarImagemFabrica(imagemAmanhecer);
            break;
        case inicioAnoitecer:
            carregarImagemFabrica(imagemInicioAnoitecer);
            break;
        case manha08:
            carregarImagemFabrica(imagemManha08fds);
            break;
        case manha09:
            carregarImagemFabrica(imagemManha09fds);
            break;
        case manha10:
            carregarImagemFabrica(imagemManha10fds);
            break;
    }
}

```

```

        case manha11:
            carregarImagemFabrica(imagemManha11fds);
            break;
        case manha12:
            carregarImagemFabrica(imagemManha12fds);
            break;
        case noite:
            carregarImagemFabrica(imagemNoite);
            break;
        case tarde13:
            carregarImagemFabrica(imagemTarde13fds);
            break;
        case tarde14:
            carregarImagemFabrica(imagemTarde14fds);
            break;
        case tarde15:
            carregarImagemFabrica(imagemTarde15fds);
            break;
        case tarde16:
            carregarImagemFabrica(imagemTarde16fds);
            break;
        case tarde17:
            carregarImagemFabrica(imagemTarde17fds);
            break;
        case tarde18:
            carregarImagemFabrica(imagemTarde18fds);
            break;
        default:
            break;
    }
}

}

}

package view;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import javax.swing.JPanel;

public class PaineComImagem extends JPanel {

    private static final long serialVersionUID = 1L;

    private String notaNatureza = "";
    private String saldoAtual = "";
    private String estoqueMateriaPrima = "";
    private String estoqueProdutoAcabado = "";
    private String dataCompleta = "";
    private String operacao = "";

    public String getNotaNatureza() {
        return notaNatureza;
    }
}

```

```

public void setNotaNatureza(String notaNatureza) {
    this.notaNatureza = notaNatureza;
}

private BufferedImage imagemFundo = null;
private String mensagemAviso = "";
private String mensagemFimJogo = "";

public void setMensagemFimJogo(String mensagemFimJogo) {
    this.mensagemFimJogo = mensagemFimJogo;
}

public void setMensagemAviso(String mensagemAviso) {
    this.mensagemAviso = mensagemAviso;
}

protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D) g.create();

    g2d.drawImage(imagemFundo, 0, 0, this.getWidth(), this.getHeight(),
        null);

    if (!mensagemAviso.isEmpty()) {
        g2d.setColor(Color.BLUE);
        g2d.drawString("Aviso: " + mensagemAviso, 110, 50);

        mensagemAviso = "";
    } else if (!mensagemFimJogo.isEmpty()) {
        g2d.setColor(Color.BLUE);
        g2d.drawString(mensagemFimJogo, 110, 50);
    }

    g2d.setColor(Color.BLUE);
    g2d.drawString("Natureza: " + getNotaNatureza(), 05, 360);

    g2d.setColor(Color.BLUE);
    g2d.drawString("Saldo R$: " + getSaldoAtual(), 05, 380);

    g2d.setColor(Color.BLUE);
    g2d.drawString("Matéria Prima: " + getEstoqueMateriaPrima(), 150, 360);

    g2d.setColor(Color.BLUE);
    g2d.drawString("Produto Acabado: " + getEstoqueProdutoAcabado(), 150, 380);

    g2d.setColor(Color.BLUE);
    g2d.drawString(getDataCompleta(), 300, 360);

    g2d.setColor(Color.BLUE);
    g2d.drawString(getOperacao(), 300, 380);

    g2d.dispose();
}

public void setImagem(BufferedImage bufferedImage) {
    if (bufferedImage == null) {
        throw new NullPointerException("Imagem inválida!");
    }

    this.imagemFundo = bufferedImage;
}

```

```

        repaint();
    }

    /**
     * @return the saldoAtual
     */
    public String getSaldoAtual() {
        return saldoAtual;
    }

    /**
     * @param saldoAtual the saldoAtual to set
     */
    public void setSaldoAtual(String saldoAtual) {
        this.saldoAtual = saldoAtual;
    }

    /**
     * @return the estoqueMateriaPrima
     */
    public String getEstoqueMateriaPrima() {
        return estoqueMateriaPrima;
    }

    /**
     * @param estoqueMateriaPrima the estoqueMateriaPrima to set
     */
    public void setEstoqueMateriaPrima(String estoqueMateriaPrima) {
        this.estoqueMateriaPrima = estoqueMateriaPrima;
    }

    /**
     * @return the estoqueProdutoAcabado
     */
    public String getEstoqueProdutoAcabado() {
        return estoqueProdutoAcabado;
    }

    /**
     * @param estoqueProdutoAcabado the estoqueProdutoAcabado to set
     */
    public void setEstoqueProdutoAcabado(String estoqueProdutoAcabado) {
        this.estoqueProdutoAcabado = estoqueProdutoAcabado;
    }

    /**
     * @return the dataCompleta
     */
    public String getDataCompleta() {
        return dataCompleta;
    }

    /**
     * @param dataCompleta the dataCompleta to set
     */
    public void setDataCompleta(String dataCompleta) {
        this.dataCompleta = dataCompleta;
    }
}

```



```

    /**
     * @return the operacao
     */
    public String getOperacao() {
        return operacao;
    }

    /**
     * @param operacao the operacao to set
     */
    public void setOperacao(String operacao) {
        this.operacao = operacao;
    }
}

package view;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import controller.ControllerEmpresa;

import java.text.ParseException;

public class ViewGame implements ActionListener {

    public JButton btnComprarMateriaPrima;
    public JButton btnOrdemProducao;
    public JButton btnVenderProdutoAcabado;
    public JButton btnPalestraSustentabilidade;
    public JButton btnFiltroFumaca;
    public JButton btnReciclagem;
    public JButton btnEcoEficiencia;

    public PainelComImagem arealmagem;

    private ControllerEmpresa controllerEmpresa;
    private AtualizarGUI atualizarGUI;

    public ViewGame() throws ParseException {
        arealmagem = new PainelComImagem();
        arealmagem.setBounds(10, 10, 600, 390);
        arealmagem.setBackground(Color.WHITE);

        controllerEmpresa = new ControllerEmpresa();

        atualizarGUI = new AtualizarGUI(this, controllerEmpresa);

        new AgendadorTarefa(controllerEmpresa, atualizarGUI);
    }

    public static void main(String[] args) {
        ViewGame b;
        try {
            b = new ViewGame();

```

```

        b.janela();
    } catch (ParseException e) {
        e.printStackTrace();
    }
}

public void janela() {
    JFrame frmJanela = new JFrame("Fabrica 21");
    frmJanela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frmJanela.setResizable(false);
    frmJanela.setLayout(null);

    /*
     * Container Principal que contém todos os Componentes
     */
    JPanel jpnContainer = new JPanel();
    jpnContainer.setBounds(0, 0, 800, 500);
    jpnContainer.setLayout(null);

    JPanel areaBotoes = new JPanel();
    areaBotoes.setBounds(10, 410, 600, 102);
    areaBotoes.setBackground(Color.WHITE);
    areaBotoes.setLayout(null);

    int left;
    int width;

    left = 3;
    width = 83;

    btnComprarMateriaPrima = new JButton("");
    btnComprarMateriaPrima.setIcon(new javax.swing.ImageIcon(atualizarGUI
        .getPathImagem() + "comprarmateriaprima.png"));
    btnComprarMateriaPrima.setBounds(left, 10, width, width);
    btnComprarMateriaPrima
        .setToolTipText("Clique aqui para comprar 50 Kg Matéria Prima para
produção");
    btnComprarMateriaPrima.setEnabled(false);
    btnComprarMateriaPrima.addActionListener(this);

    left += width + 2;
    btnOrdemProducao = new JButton("");
    btnOrdemProducao.setIcon(new javax.swing.ImageIcon(atualizarGUI
        .getPathImagem() + "ordemproducao.png"));
    btnOrdemProducao.setBounds(left, 10, width, width);
    btnOrdemProducao
        .setToolTipText("Clique aqui para fabricar 5 peças de produto
acabado!");
    btnOrdemProducao.setEnabled(false);
    btnOrdemProducao.addActionListener(this);

    left += width + 2;
    btnVenderProdutoAcabado = new JButton("");
    btnVenderProdutoAcabado.setIcon(new javax.swing.ImageIcon(atualizarGUI
        .getPathImagem() + "vender.png"));
    btnVenderProdutoAcabado.setBounds(left, 10, width, width);
    btnVenderProdutoAcabado
        .setToolTipText("Clique aqui para vender 5 peças de produto
acabado!");
    btnVenderProdutoAcabado.setEnabled(false);

```

```

btnVenderProdutoAcabado.addActionListener(this);

left += width + 2;
btnPalestraSustentabilidade = new JButton("");
btnPalestraSustentabilidade.setIcon(new javax.swing.ImageIcon(atualizarGUI
    .getPathImagem() + "palestrasustentabilidade.png"));
btnPalestraSustentabilidade.setBounds(left, 10, width, width);
btnPalestraSustentabilidade
    .setToolTipText("Investir em Palestra de sustentabilidade!");
btnPalestraSustentabilidade.setEnabled(false);
btnPalestraSustentabilidade.addActionListener(this);
btnPalestraSustentabilidade.setBackground(Color.GREEN);

left += width + 2;
btnFiltroFumaca = new JButton("");
btnFiltroFumaca.setIcon(new javax.swing.ImageIcon(atualizarGUI
    .getPathImagem() + "filtro.png"));
btnFiltroFumaca.setBounds(left, 10, width, width);
btnFiltroFumaca
    .setToolTipText("Investir em Filtros para reduzir emissão de gases
tóxicos!");

btnFiltroFumaca.setEnabled(false);
btnFiltroFumaca.addActionListener(this);
btnFiltroFumaca.setBackground(Color.GREEN);

left += width + 2;
btnReciclagem = new JButton("");
btnReciclagem.setIcon(new javax.swing.ImageIcon(atualizarGUI
    .getPathImagem() + "reciclagem.png"));
btnReciclagem.setBounds(left, 10, width, width);
btnReciclagem.setToolTipText("Investir em programas de Reciclagem!");
btnReciclagem.setEnabled(false);
btnReciclagem.addActionListener(this);
btnReciclagem.setBackground(Color.GREEN);

left += width + 2;
btnEcoEficiencia = new JButton("");
btnEcoEficiencia.setIcon(new javax.swing.ImageIcon(atualizarGUI
    .getPathImagem() + "ecodesenvolvimento.png"));
btnEcoEficiencia.setBounds(left, 10, width, width);
btnEcoEficiencia
    .setToolTipText("Investir em programas de Ecodesenvolvimento!");
btnEcoEficiencia.setEnabled(false);
btnEcoEficiencia.addActionListener(this);
btnEcoEficiencia.setBackground(Color.GREEN);

// == ADICIONAR COMPONENTES =====
frmJanela.getContentPane().add(arealImagem);
frmJanela.getContentPane().add(areaBotoes);

areaBotoes.add(btnComprarMateriaPrima);
areaBotoes.add(btnOrdemProducao);
areaBotoes.add(btnVenderProdutoAcabado);
areaBotoes.add(btnPalestraSustentabilidade);
areaBotoes.add(btnFiltroFumaca);
areaBotoes.add(btnReciclagem);
areaBotoes.add(btnEcoEficiencia);

frmJanela.getContentPane().add(jpnContainer);

```

```

// =====

frmJanela.setSize(630, 550);
frmJanela.setLocationRelativeTo(null);
frmJanela.setVisible(true);

}

@Override
public void actionPerformed(ActionEvent e) {
    String retorno = "";

    if (e.getSource() == btnComprarMateriaPrima) {
        retorno = controllerEmpresa.comprarMateriaPrima();
    } else if (e.getSource() == btnOrdemProducao) {
        retorno = controllerEmpresa.executarOrdemProducao();
    } else if (e.getSource() == btnVenderProdutoAcabado) {
        retorno = controllerEmpresa.venderProdutoAcabado();
    } else if (e.getSource() == btnPalestraSustentabilidade) {
        retorno = controllerEmpresa.palestraSustentabilidade();
    } else if (e.getSource() == btnFiltroFumaca) {
        retorno = controllerEmpresa.filtroFumaca();
    } else if (e.getSource() == btnReciclagem) {
        retorno = controllerEmpresa.reciclagem();
    } else if (e.getSource() == btnEcoEficiencia) {
        retorno = controllerEmpresa.ecoEficiencia();
    }

    if (!retorno.isEmpty()) {
        arealmagem.setMensagemAviso(retorno);
    }

    try {
        atualizarGUI.atualizar("");
    } catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}
}

```

5. PROGRAMA EM FUNCIONAMENTO

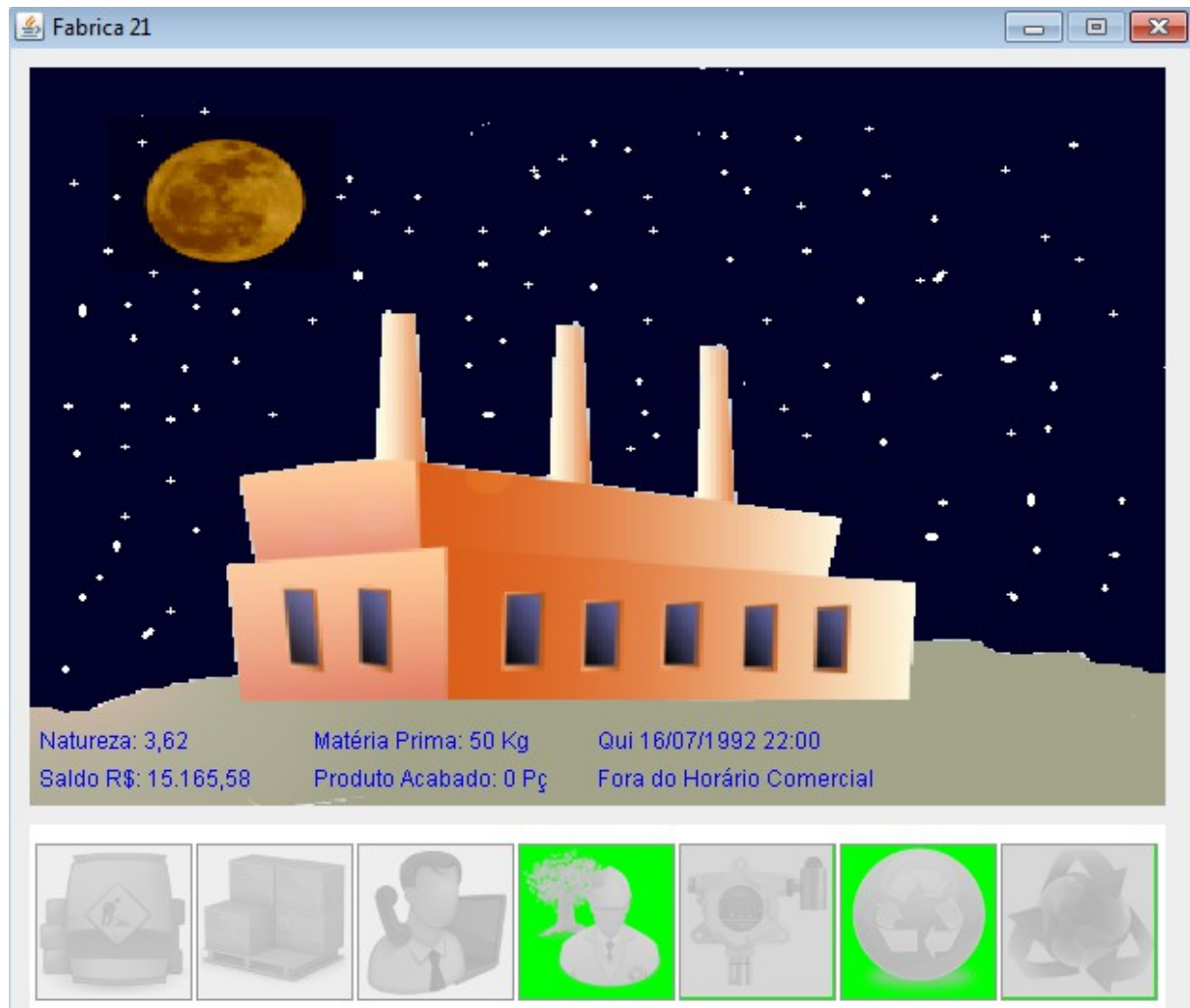
5.1. Dia Comercial

Cara do Jogo.



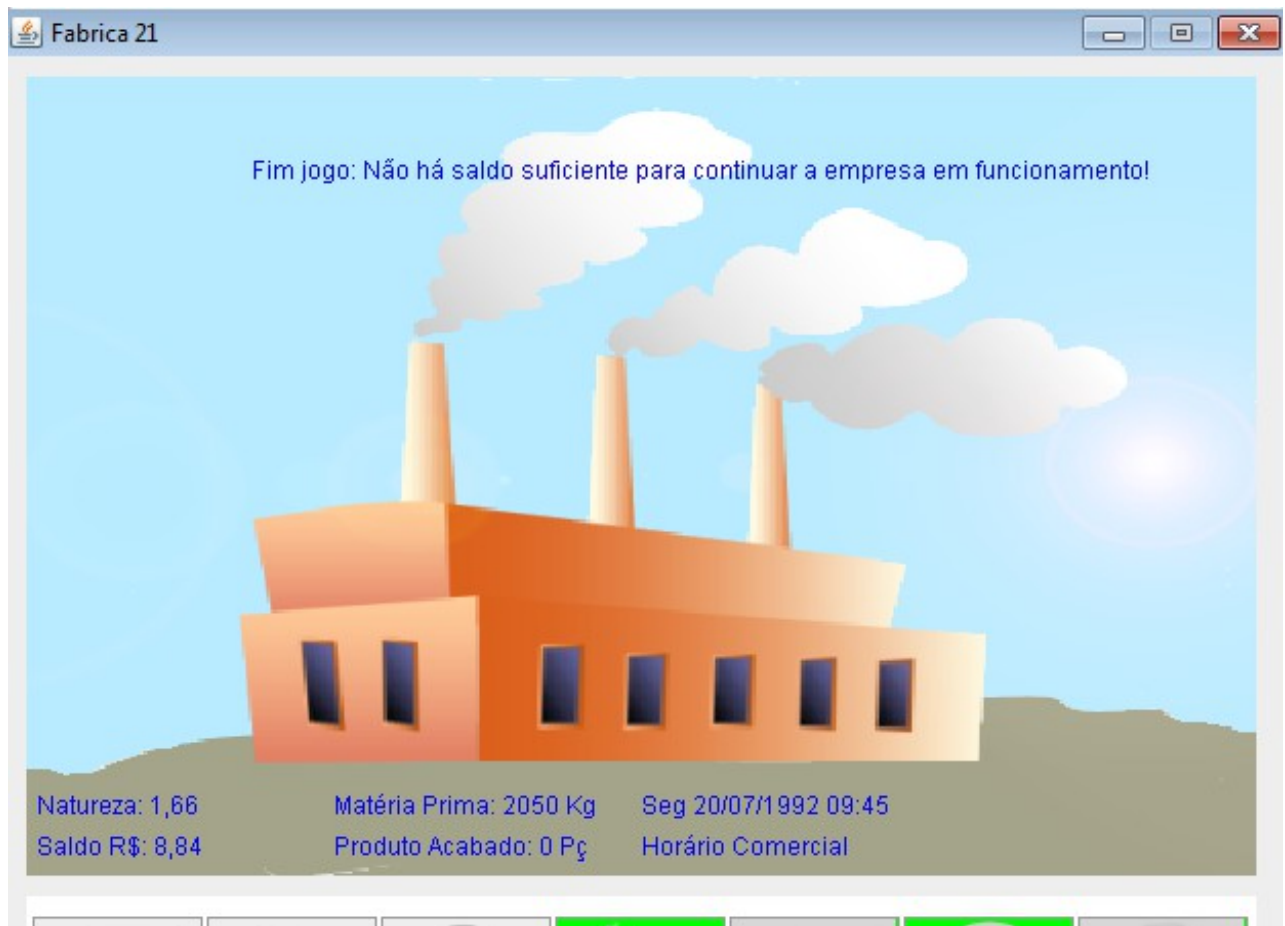
5.2. Noite Fora de Comercial

Durante o período noturno o jogo fica fora de serviço não possibilitando o jogo ser jogado.



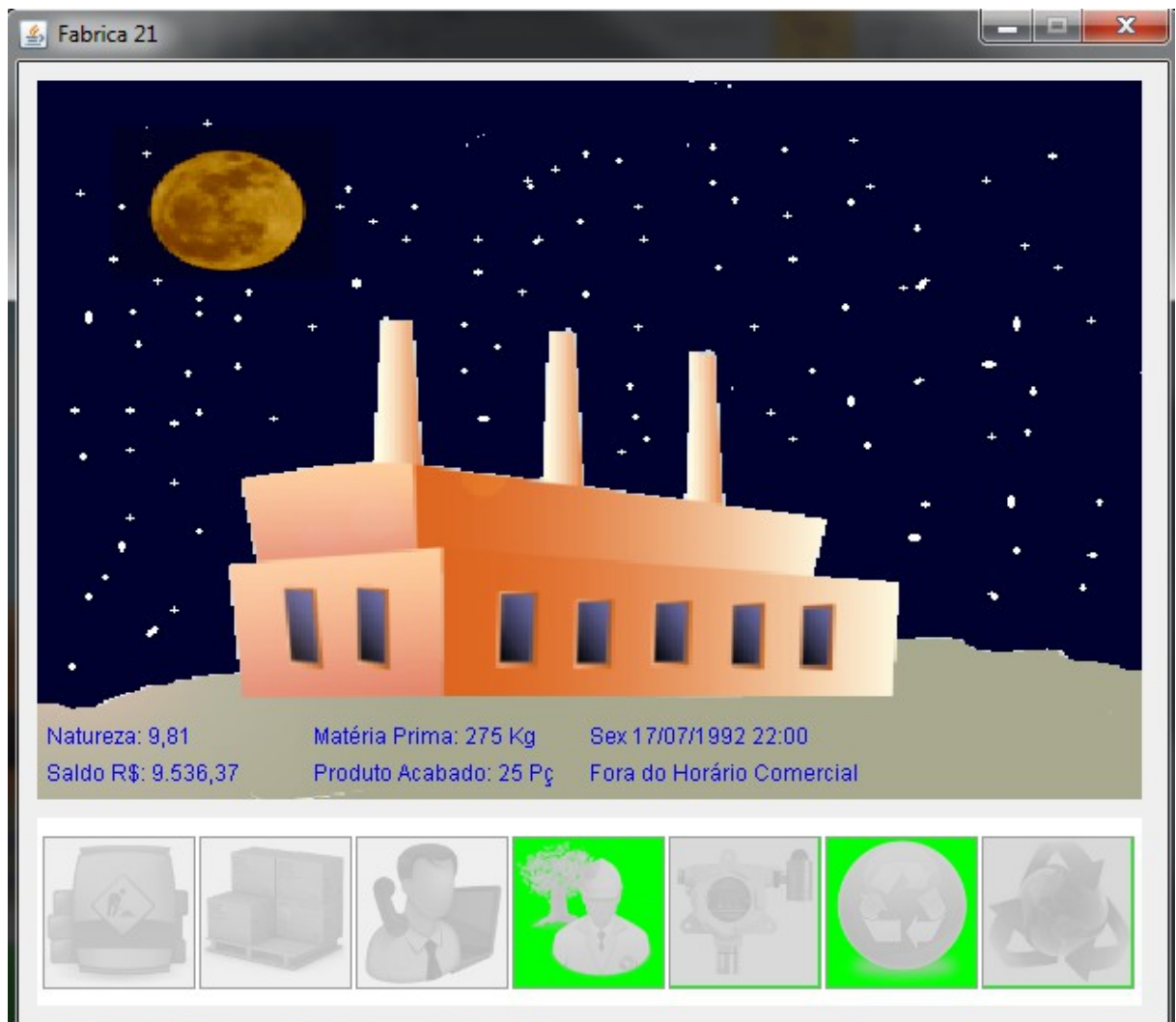
5.3. Tela de Game Over

Tela do Final do Jogo aonde o jogador tem que reiniciar o jogo desde de o inicio.



5.4. Final de Semana

Durante o final de semana a empresa fica fora de serviço aonde o jogador fica sem poder jogar tendo que esperar chegar até o início da semana.



5.5. Comprar Materia Prima

Para interagir com o jogo como comprar o matéria prima você precisa interagir com os botões.



5.6. Fabricar Peças

Para fabricar peças basta clicar no botão de fabricar peças ao lado do botão de comprar matéria prima.



5.7. Vender Peças

Para vender produtos basta somente clicar no botão de vender



5.8. Nota da Natureza

Para ajudar a natureza você deve clicar nos últimos ícones para comprar coisas para ajudar a natureza



6. BIBLIOGRAFIA

<http://www.pontov.com.br/site/>

Livro - Desenvolvimento de Jogos Eletrônicos - Teoria e Prática

Autor: Antonio cordova de Berthem, Guilherme Lage Bertschinger, Alexandre Souza Perucia

Trail: 2D Graphcs: <http://docs.oracle.com/javase/tutorial/2d/>

Livro: Programação de games com Java - HARBOUR, J S

MVC – Padrões de Projeto: O Modelo MVC: http://www.macoratti.net/vbn_mvc.htm

Livro: USE A CABECA - JAVA 2 - BERT BATES, KATHY SIERRA

Livro: Deitel; Java - Como Programar - 8.a Edição - Deitel

7. Ficha APS

Nome: Danilo de Oliveira Dorotheu
RA: B408FA-3

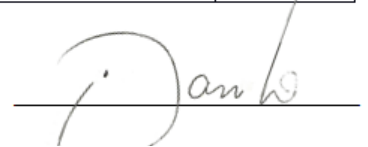
Curso: Ciência da Computação (Noturno)

Campus: Marquê de São Vicente

Semestre: 4º Semestre

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, Iniciação Científica, trabalhos Individuais e em grupo, práticas de ensino e outras).

DATA	Atividade	Total de Horas
02/09/2013	Reunião com o grupo para definição do tema do jogo	2:00
03/09/2013	Reunião com o grupo para definição das da tela e das funcionalidades do jogo	2:00
10/09/2013	Digitalização da imagem escolhida	2:00
08/09/2013	Desenvolvimento da classe <u>ParametrosCalculo</u> (camada <u>Model</u>)	3:00
14/09/2013	Desenvolvimento da classe <u>ModelEmpresa</u> (camada <u>Model</u>)	4:00
15/09/2013	Desenvolvimento da classe <u>ControllerEmpresa</u> (camada <u>Controller</u>)	3:00
15/09/2013	Desenvolvimento da classe de testes <u>ViewTesteConsole</u> (camada <u>View</u>)	2:00
21/09/2013	Revisão do código do jogo e remoção dos primeiros bugs	4:00
21/09/2013	Desenvolvimento da classe de tipo auxiliar <u>TipoInvestimentoNatureza</u> (pacote tipo)	2:00
21/09/2013	Pesquisa sobre o período em que o jogo se passa	2:00
22/09/2013	Desenvolvimento da classe <u>PainelComImagem</u> (camada <u>View</u>)	4:00
05/10/2013	Pesquisa de ícones para o jogo	1:30
05/10/2013	Obtenção dos ícones gratuitos para os botões do programa no site www.iconfinder.com	1:00
07/10/2013	Edição e adequação dos ícones ao jogo	4:00
14/10/2013	Ajustes na <u>Jogabilidade</u>	8:00
13/10/2013	Testes finais no funcionamento do jogo	3:00
14/10/2013	Fechamento do jogo	3:00
17/10/2013	Finalização da parte teórica	2:00
	Total de Horas	52:30



Nome: Marcio Fernandes Cruz
RA: B22816-4

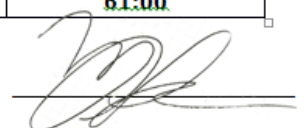
Curso: Ciência da Computação (Noturno)

Campus: Marquê de São Vicente

Semestre: 3º Semestre

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, Iniciação Científica, trabalhos Individuais e em grupo, práticas de ensino e outras).

DATA	Atividade	Total de Horas
02/09/2013	Reunião com o grupo para definição do tema do jogo	2:00
03/09/2013	Reunião com o grupo para definição das da tela e das funcionalidades do jogo	2:00
14/09/2013	Definição do projeto MVC e divisão das responsabilidades do desenvolvimento das classes	4:00
08/09/2013	Desenvolvimento da classe <u>ParametrosCalculo</u> (camada <u>Model</u>)	2:00
14/09/2013	Desenvolvimento da classe <u>ModelEmpresa</u> (camada <u>Model</u>)	4:00
15/09/2013	Desenvolvimento da classe <u>ControllerEmpresa</u> (camada <u>Controller</u>)	3:00
15/09/2013	Desenvolvimento da classe de testes <u>ViewTesteConsole</u> (camada <u>View</u>)	2:00
21/09/2013	Desenvolvimento da classe de tipo auxiliar <u>PeriodoDia</u> (pacote tipo)	2:00
21/09/2013	Desenvolvimento da classe de tipo auxiliar <u>TipoInvestimentoNatureza</u> (pacote tipo)	2:00
21/09/2013	Desenvolvimento da classe <u>AgendadorTarefa</u> (camada <u>View</u>)	3:00
22/09/2013	Desenvolvimento da classe <u>PainelComImagem</u> (camada <u>View</u>)	4:00
22/09/2013	Desenvolvimento da classe <u>AtualizarGUI</u> (camada <u>GUI</u>)	3:00
05/10/2013	Criação das imagens <u>png</u> da fábrica utilizando o software GIMP	3:00
05/10/2013	Obtenção dos <u>ícones</u> gratuitos para os botões do <u>rograma</u> no site www.iconfinder.com	1:00
06/10/2013	Teste geral de funcionamento do sistema e ajustes de <u>jogabilidade</u>	3:00
12/10/2013	Ajuste na <u>jogabilidade</u> do jogo Fabrica 2 1	8:00
13/10/2013	Testes finais no funcionamento do jogo	6:00
14/10/2013	Fechamento do jogo	3:00
15/10/2013	Digitação da parte teórica	4:00
	Total de Horas	61:00



Nome: Renato Goulart Rodrigues
RA: T618FA-4

Curso: Ciência da Computação (Noturno)

Campus: Marquê de São Vicente

Semestre: 4º Semestre

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, Iniciação Científica, trabalhos Individuais e em grupo, práticas de ensino e outras).

DATA	Atividade	Total de Horas
02/09/2013	Reunião com o grupo para definição do tema do jogo	2:00
03/09/2013	Reunião com o grupo para definição das da tela e das funcionalidades do jogo	2:00
08/09/2013	Croqui das imagens do jogo	4:00
08/09/2013	Desenvolvimento da classe <u>ParametrosCalculo</u> (camada <u>Model</u>)	3:00
14/09/2013	Desenvolvimento da classe <u>ModelEmpresa</u> (camada <u>Model</u>)	4:00
15/09/2013	Desenvolvimento da classe <u>ControllerEmpresa</u> (camada <u>Controller</u>)	3:00
15/09/2013	Desenvolvimento da classe de testes <u>ViewTesteConsole</u> (camada <u>View</u>)	2:00
21/09/2013	Digitação da parte teórica (Introdução, objetivo do trabalho, resumo)	3:00
21/09/2013	Desenvolvimento da classe de tipo auxiliar <u>TipoInvestimentoNatureza</u> (pacote tipo)	2:00
21/09/2013	Pesquisa sobre o período em que o jogo se passa	2:00
22/09/2013	Desenvolvimento da classe <u>PainelComImagem</u> (camada <u>View</u>)	4:00
24/09/2013	Digitação do código do programa	3:00
05/10/2013	Início de Digitação Parte teórica (manual do Jogo)	2:50
05/10/2013	Obtenção dos <u>ícones</u> gratuitos para os botões do programa no site www.iconfinder.com	1:00
06/10/2013	Teste geral de funcionamento do sistema e ajustes de <u>jogabilidade</u>	3:00
12/10/2013	Pesquisa para <u>implementar</u> bugs no jogo	8:00
13/10/2013	Testes finais no funcionamento do jogo	6:00
14/10/2013	Fechamento do jogo	3:00
17/10/2013	Finalização da parte teórica	5:00
	Total de Horas	62:30

Renato Goulart Rodrigues