
Documentação

Projeto de Software

Professor: Rohit Gheyi

Equipe:

Nome: Henrique Galindo

Matricula: 119110732

Email: henrique.galindo@ccc.ufcg.edu.br

Github: HenriqueGalindo

Nome: Igor Serodio

Matricula: 119210338

Email: igor.barcelar@ccc.ufcg.edu.br

Github: IgorSerodio

Nome: João Pedro Silva de Melo

Matricula: 118210796

Email: joao.melo@ccc.ufcg.edu.br

Github: jpedrosml

Nome: Luiz Antonio

Matricula: 119110759

Email: luiz.cardozo@ccc.ufcg.edu.br

Github: LuizAntonio67

Nome: Ody Junior

Matricula: 118210174

Email: ody.mendes@ccc.ufcg.edu.br

Github: Odyjmm

Nome: Sanderson Junior

Matricula: 118111051

Email: sanderson.junior@ccc.ufcg.edu.br

Github: SandersonJunior

Sistema Novo

1. Introdução

1.1. Motivação

O problema	Nosso objetivo é desenvolver um jogo inspirado em alguns títulos famosos, mas que de certa forma mantenha sua originalidade. Dito isto, é obrigação de nossa equipe tentar afastar o máximo possível as semelhanças com os jogos que serviram de base para nosso projeto, elaborando ideias concisas e originais para atrair o público.
Quem afeta	Nos últimos anos os jogos independentes, também categorizados como indies, têm ganhado mais relevância. Isto aconteceu pelo fato de que as ideias normalmente são mais arriscadas e, conseqüentemente, criativas quando se comparadas àquelas propostas por jogos de grandes empresas. Além disso, mesmo o público que não é familiarizado com jogos eletrônicos tem se envolvido nesse cenário, graças a popularização dos smartphones e seus aplicativos. Os nossos stakeholders seriam o público em geral, uma vez que nossa ideia foca em atrair não somente aqueles que já jogam videogame ou que possuem uma idade específica.
O impacto disto	Iremos entreter o público em geral com a nossa proposta, que embora simples, terá seu próprio charme e visual. O que nosso jogo oferece pode ser intitulado como educativo, uma vez que estimulam o raciocínio e criatividade.
A solução	<ol style="list-style-type: none">1. Desenvolver o jogo utilizando Android Studio;2. Composto por diversos mini jogos que são escolhidos aleatoriamente, o jogo em si funciona através de partidas com uma dificuldade incremental;

1.2. Visão da Solução

Temos como objetivo desenvolver um jogo simples, mas que atraia o público de diversas idades, com seus visuais únicos e chamativos.

1.3. Visão Geral do Documento

Este documento está organizado da seguinte forma:

- Seção 1: apresenta a motivação e a ideia do jogo a ser desenvolvido;
- Seção 2: apresenta os requisitos organizacionais do projeto;
- Seção 3: apresenta o processo adotado durante todas as fases do projeto;
- Seção 4: apresenta os requisitos funcionais e não funcionais;
- Seção 5: apresenta o projeto arquitetural;
- Seção 6: apresenta a especificação formal do projeto;
- Seção 7: apresenta os links relativos a implementação do projeto;
- Seção 8: apresenta os testes do sistema novo;
- Seção 9: apresenta a introdução do sistema real;
- Seção 10: apresenta a análise da qualidade do sistema real;
- Seção 11: apresenta os testes do sistema real;
- Seção 12: apresenta a evolução do sistema real;
- Seção 13: apresenta uma breve conclusão da nossa jornada;

2. Planejamento

2.1. Introdução

No plano de projeto, apresentamos a descrição do jogo mobile 'Brainy Beans', que funciona de forma que as partidas são compostas por mini jogos escolhidos aleatoriamente, onde a dificuldade escala de um para o outro, dando pontos aos jogadores baseando-se no tempo em que cada jogo é finalizado. Além do propósito de entreter, o jogo também serve como um meio de estimular o raciocínio.

2.2. Gerência do Tempo

Naturalmente, o tempo foi gerenciado baseando-se no tempo disponível até cada entrega. Focar na entrega final seria um tanto desgastante, então existem diferentes cronogramas relativos às entregas, os quais nomeamos de 'fases'. Conforme avançamos no projeto, os cronogramas com seus respectivos detalhes serão colocados abaixo.

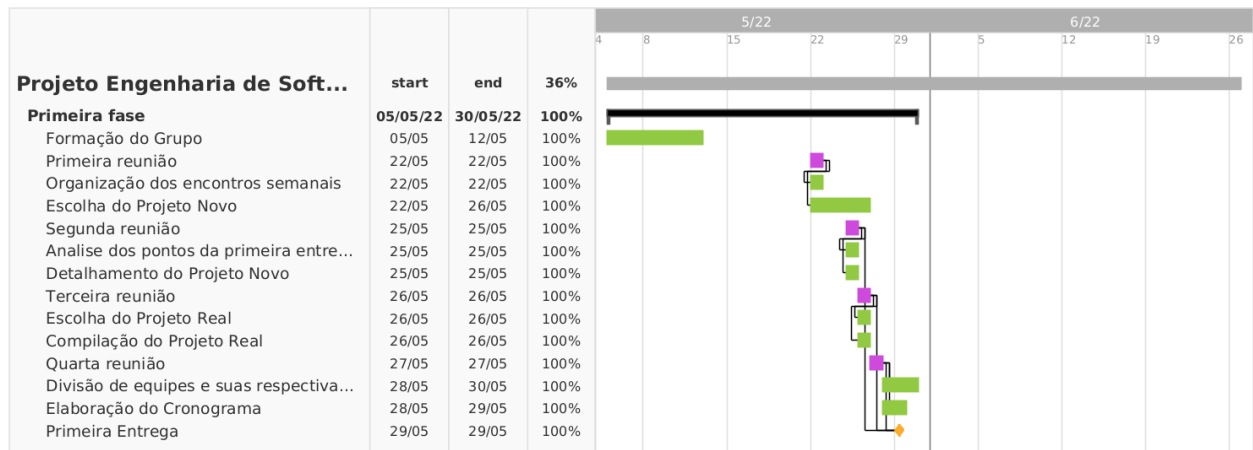


Figura 1 - Cronograma da primeira fase reajustado

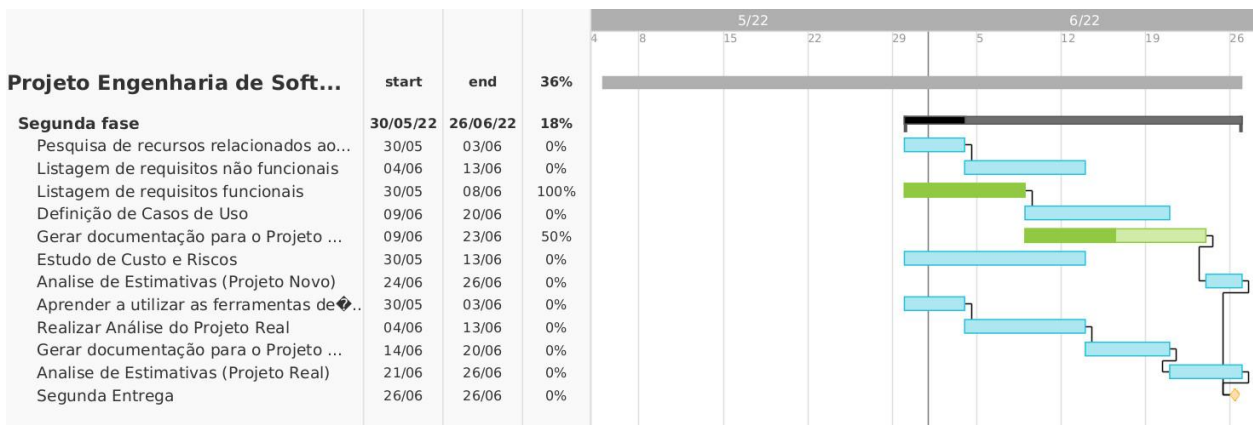


Figura 2 - Cronograma da segunda fase (inicial)

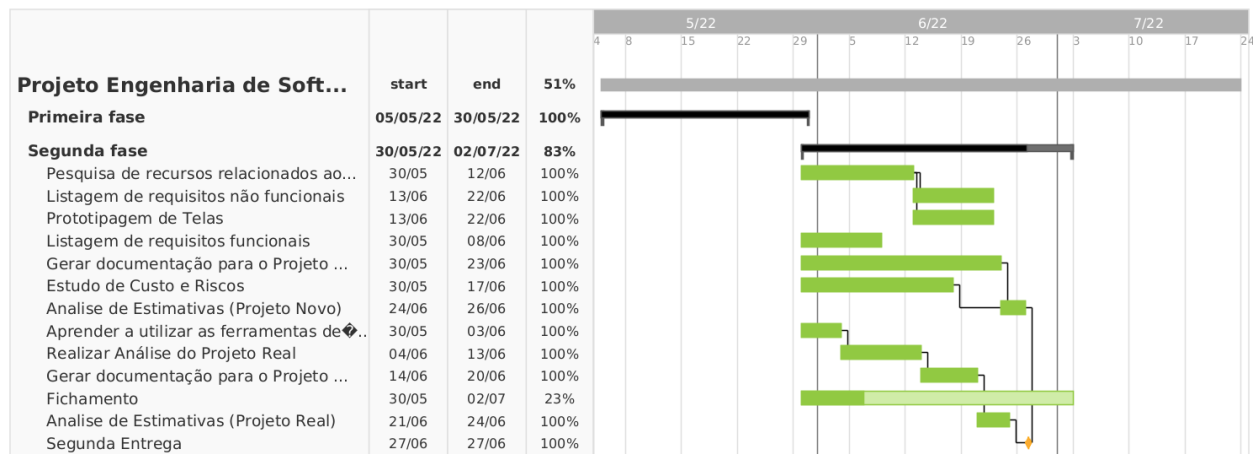


Figura 3 - Cronograma da segunda fase (final) com fichamentos em progresso

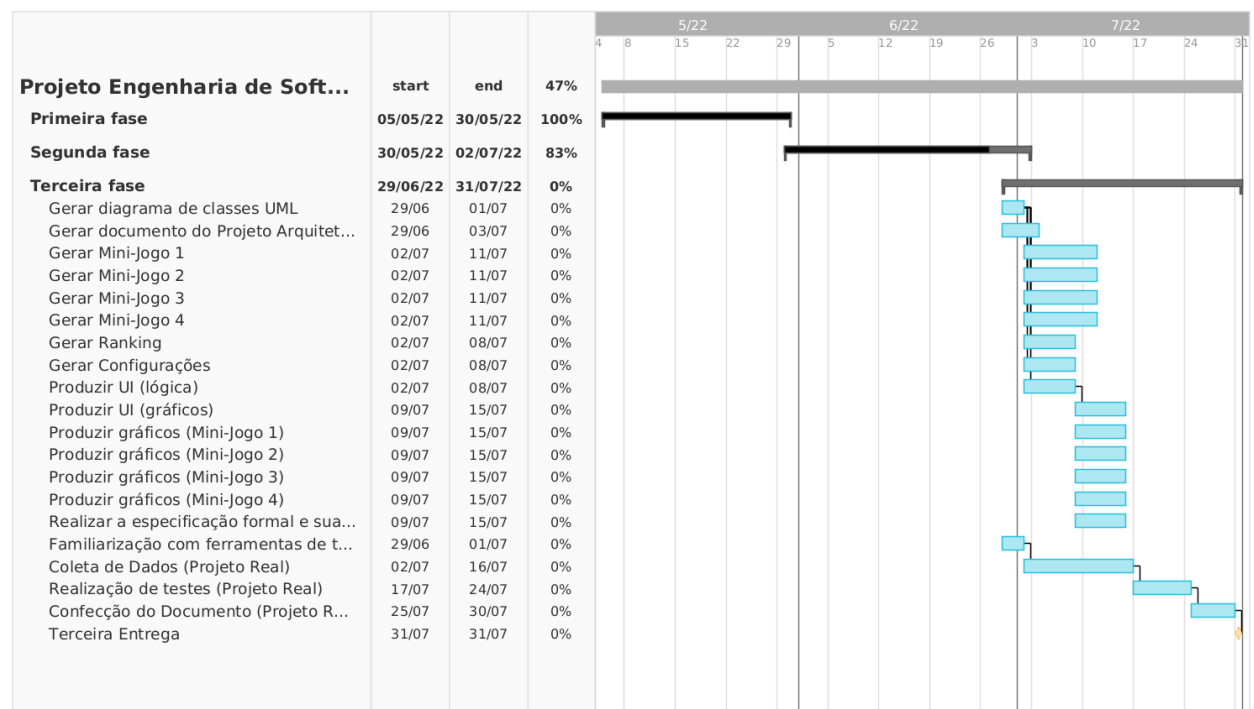


Figura 4 - Cronograma da terceira fase (inicial)

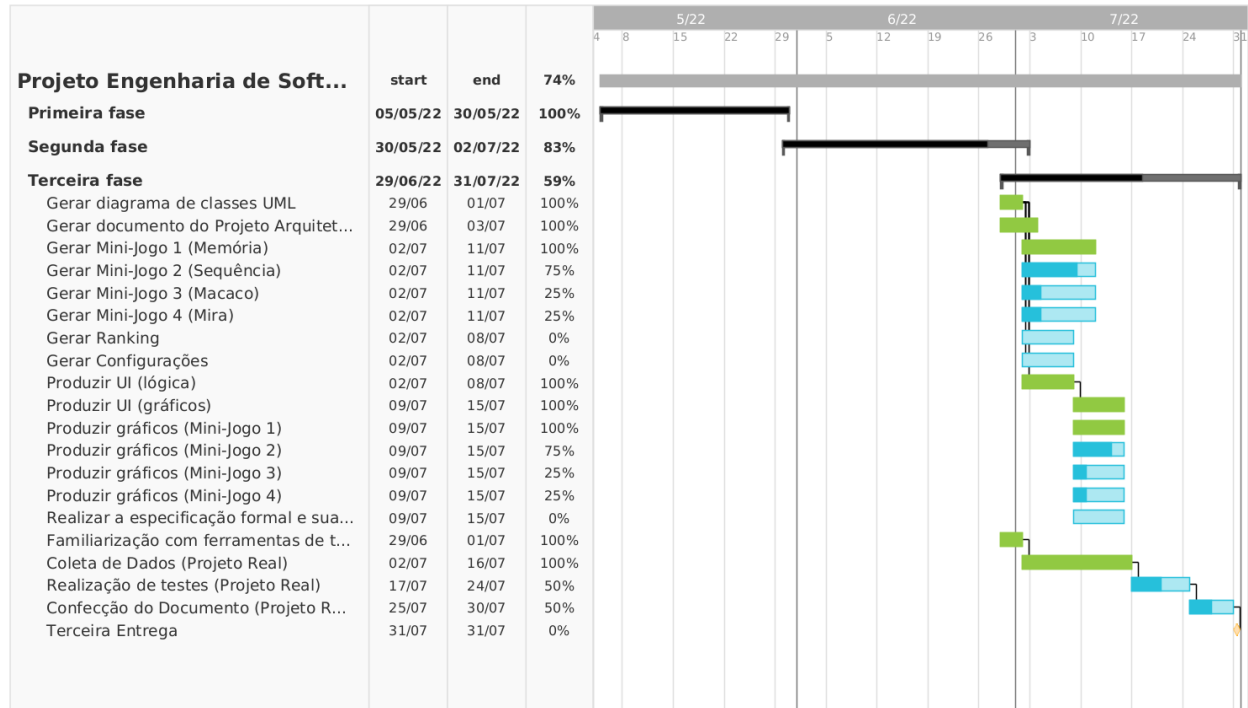


Figura 5 - Cronograma da terceira fase (final), com atividades incompletas.

2.2.1. Descrição das Estimativas

Planilha com a divisão das atividades do projeto (sempre atualizada):

[+ Divisão do projeto](#)

Cronogramas:

Cronograma da primeira fase (reajustado):

[CronogramaReajustado.pdf](#)

Cronograma inicial da segunda fase:

[CronogramaSegundaEntrega\(inicio\).pdf](#)


Cronograma final da segunda fase:

[Cronograma Segunda Entrega\(final\).pdf](#)

Cronograma inicial da terceira fase:

[Cronograma Terceira Entrega \(inicio\).pdf](#)

Cronograma final da terceira fase:

 CronogramaTerceira Entrega(final).pdf

Cronograma da última fase (em breve);

Foi-se necessário um ajuste no primeiro cronograma, pois o integrante responsável pelo cronograma ainda não sabia utilizar a ferramenta apropriadamente, deixando certos detalhes incompletos.

P.s: O aplicativo que usamos nos oferece uma versão gratuita que não nos permite separar as horas trabalhadas em cada atividade. Dito isto, optamos por deixar o cronograma apenas com as datas e as tarefas.

2.2.2. Análise das Estimativas

Task	Integrante	Dificuldade	Peso	Estado	Data Inicial	Data de Término	Data Final	Tempo
1. Introdução (Novo)	João Pedro	Baixa	1	Finalizado	02/06/2022	03/06/2022	23/06/2022	1 dia
2.2. Gerência do Tempo	João Pedro	Alta	3	Finalizado	02/06/2022	—	24/06/2022	—
2.3. Gerência de Custo	Rodrigo	Médio	2	Não realizou	02/06/2022	—	17/06/2022	—
2.3. Gerência de Custo	João Pedro	Médio	2	Finalizado	02/06/2022	23/06/2022	17/06/2022	21 dias
2.4. Gerência de Risco	Matheus Alves	Médio	2	Não realizou	02/06/2022	—	17/06/2022	—
2.4. Gerência de Risco	João Pedro	Médio	2	Finalizado	02/06/2022	23/06/2022	17/06/2022	21 dias
2.5. Recursos do Projeto	Todos	Baixo	1	Finalizado	02/06/2022	05/06/2022	23/06/2022	3 dias
2.6. Comunicação	João Pedro	Baixa	1	Finalizado	02/06/2022	03/06/2022	23/06/2022	1 dia
3. Processo	João Pedro	Baixa	1	Finalizado	02/06/2022	03/06/2022	23/06/2022	1 dia
4.1. Req. Não Funcionais	Henrique	Médio	2	Em andamento	13/06/2022	—	22/06/2022	—
4.2 Req. Funcionais	Luiz Antônio	Médio	2	Finalizado	02/06/2022	07/06/2022	08/06/2022	5 dias
4.2.1. Especific. de Req.	Luiz Antônio	Médio	2	Finalizado	02/06/2022	13/06/2022	24/06/2022	11 dias
4.3 Protótipo de Telas	Igor Serodio	Alta	3	Finalizado	13/06/2022	18/06/2022	22/06/2022	5 dias
9. Introdução (Real)	Ody / Sanderson	Baixa	1	Em Andamento	04/06/2022	—	13/06/2022	—
10. Qualidade	Ody / Sanderson	Alta	3	Em Andamento	14/06/2022	—	20/06/2022	—

Figura 6 - Divisão de atividades por membros (23/06/2022)

Task	Integrante	Dificuldade	Peso	Estado	Data Inicial	Data de Término	Data Final	Tempo
1. Introdução (Novo)	João Pedro	Baixa	1	Finalizado	02/06/2022	03/06/2022	23/06/2022	1 dia
2.2. Gerência do Tempo	João Pedro	Alta	3	Contínuo	02/06/2022	—	24/06/2022	—
2.3. Gerência de Custo	Rodrigo	Médio	2	Não realizou	02/06/2022	—	17/06/2022	—
2.3. Gerência de Custo	João Pedro	Médio	2	Finalizado	02/06/2022	23/06/2022	17/06/2022	21 dias
2.4. Gerência de Risco	Matheus Alves	Médio	2	Não realizou	02/06/2022	—	17/06/2022	—
2.4. Gerência de Risco	João Pedro	Médio	2	Finalizado	02/06/2022	23/06/2022	17/06/2022	21 dias
2.5. Recursos do Projeto	Todos	Baixo	1	Finalizado	02/06/2022	05/06/2022	23/06/2022	3 dias
2.6. Comunicação	João Pedro	Baixa	1	Finalizado	02/06/2022	03/06/2022	23/06/2022	1 dia
3. Processo	João Pedro	Baixa	1	Finalizado	02/06/2022	03/06/2022	23/06/2022	1 dia
4.1. Req. Não Funcionais	Henrique	Médio	2	Finalizado	13/06/2022	25/06/2022	22/06/2022	12 dias
4.2 Req. Funcionais	Luiz Antônio	Médio	2	Finalizado	02/06/2022	07/06/2022	08/06/2022	5 dias
4.2.1. Especific. de Req.	Luiz Antônio	Médio	2	Finalizado	02/06/2022	13/06/2022	24/06/2022	11 dias
4.3 Protótipo de Telas	Igor Serodio	Alta	3	Finalizado	13/06/2022	18/06/2022	22/06/2022	5 dias
9. Introdução (Real)	Ody / Sanderson	Baixa	1	Finalizado	04/06/2022	26/06/2022	13/06/2022	22 dias
10. Qualidade	Ody / Sanderson	Alta	3	Finalizado	14/06/2022	26/06/2022	20/06/2022	12 dias
	Legenda:	Em andamento	Finalizado	Finalizado depois do prazo		Contínuo	Não realizou	

Figura 7 - Divisão de atividades por membros - Reajustado e com legenda (26/06/2022)

Divisão do projeto *(Atualizada)*

Num balanço feito três dias antes da entrega, percebemos que a maioria das atividades distribuídas aos integrantes do grupo foram realizadas antes ou no prazo estimado (data de término). Três integrantes acabaram se ausentando do projeto neste período da segunda entrega, e como o prazo final se aproximava, preventivamente, fora decidido deixar as atividades que lhes foram atribuídas na responsabilidade de outra pessoa, o que de certa forma foi um contratempo.

A ideia inicial do jogo a ser desenvolvido consistia de toda sua descrição aqui mencionada neste documento *mais* a funcionalidade de ser uma aplicação *online/multiplayer*. Outras como *efeitos sonoros*, *tutoriais*, um número maior de *mini-jogos* e uma maior *robustez* na implementação viriam ser, também, incluídas. Porém, além da ausência dos integrantes citados no trecho anterior, o nosso grupo

passou de, inicialmente, *onze* pessoas, para apenas *nove*, e entre estas nove havia três ausentes durante a segunda fase inteira. Dito isto, optamos por reduzir o escopo do nosso projeto cortando tais funcionalidades que certamente nos colocariam em risco.

5. Projeto Arquitetural	João Pedro	Médio	2	Finalizado	29/06/2022	08/07/2022	04/07/2022	9 dias
7. Implement. Jogo da Memória	Igor Serodio	Alta	3	Finalizado	11/07/2022	21/07/2022	15/07/2022	10 dias
7. Implement. Jogo do Macaco	João Pedro	Alta	3	Em Andamento	11/07/2022	-	15/07/2022	-
7. Implement. Jogo da Sequência	Luiz Antônio	Alta	3	Em Andamento	11/07/2022	-	15/07/2022	-
7. Implement. Jogo da Mira	Henrique	Alta	3	Em Andamento	11/07/2022	-	15/07/2022	-
7. Implement. Ranking	Sanderson	Alta	3	Em Andamento	11/07/2022	-	15/07/2022	-
9. Introdução (Real)	Ody / Sanderson	Baixa	1	Finalizado	04/06/2022	26/06/2022	13/06/2022	22 dias
10. Qualidade	Ody / Sanderson	Alta	3	Finalizado	14/06/2022	26/06/2022	20/06/2022	12 dias
11. Coleta de Dados (Real)	Ody	Alta	3	Em Andamento	28/06/2022	-	17/07/2022	-
11. Testes (Real)	Ody	Alta	3	Em Andamento	17/07/2022	-	23/07/2022	-
12. Evolução (Real)	Ody	Alta	3	Em Andamento	17/07/2022	-	23/07/2022	-
	Legenda:	Em andamento	Finalizado	Finalizado depois do prazo	Contínuo	Não realizou		

Figura 8 - Divisão de atividades por membros (31/07/2022)

No dia 31 de Julho, ao fazermos uma última edição na planilha de atividades concluídas, percebemos que não conseguimos cumprir com boa parte das tarefas relacionadas ao projeto novo, como uma consequência da saída dos três integrantes (Matheus Alves, Mateus Ferreira, Rodrigo) que se garantiram de realizá-las mas trancaram a disciplina sem nenhum aviso (ambos continuando no grupo da equipe). Quanto às atividades do projeto real, o integrante responsável por elas tem lidado com problemas de internet em sua cidade, e até as 23h da data mencionada não entrou em contato, embora tenha assegurado de que estava quase pronto há alguns dias antes. Infelizmente o nosso grupo, que de início já era composto por poucos integrantes, foi prejudicado por saídas repentinas e sem nenhum tipo de contato prévio; adultos têm

ciência de suas responsabilidades, portanto outros adultos os dizendo a todo instante o que fazer chega a ser incômodo para ambos os lados.

2.3. Gerência do Custo

O jogo que a nossa pequena equipe está desenvolvendo é *indie* (independente), e portanto não houve nenhuma aquisição com *software* ou *hardware*, e tampouco qualquer contribuição financeira vinda de terceiros. A equipe chegou a um consenso geral que os custos relacionados a água, luz e derivados não tinham nenhuma necessidade de serem incluídos no gerenciamento de custo.

2.4. Gerência de Riscos

Classificação do Risco	Impacto e Descrição do Risco	Estratégia de Diminuição e/ou Plano de Contingência
Alto	Inexperiência em criação de jogos	O integrante que tem domínio ou já participou da criação de algum jogo pode auxiliar os demais membros que ainda têm dificuldade ou nenhuma experiência na parte de criação de jogos.
Alto	Inexperiência com as ferramentas de desenvolvimento	Aprender no meio tempo entre as entregas as ferramentas necessárias para o desenvolvimento do projeto novo, através de vídeo aulas e documentos on-line.
Alto	Dificuldade em fazer os mini-jogos	Discutir com os membros como será possível aplicar em linhas de código o que seria a ideia inicial dos mini-jogos.
Médio	Dificuldade de conciliar o projeto com as demais disciplinas do período	Planejar horários que coincidam com o tempo livre de cada um para a realização de reuniões e desenvolvimento conjunto caso necessário.

Médio	Dificuldade em dividir as tarefas para os membros do grupo	Separar um grupo responsável pelo projeto novo, cujos integrantes têm mais afinidade com a linguagem a ser utilizada, e um responsável pelo projeto real. Além disso, separar no máximo dois integrantes para ficarem responsáveis pela parte da documentação.
Médio	Dificuldade com Inglês	Auxiliar os que têm dificuldade na leitura do idioma, especialmente na parte da divisão de artigos para os fichamentos.

2.5. Recursos do Projeto

Recursos de Hardware:

Henrique Galindo	Processador: Intel Core i5 Memória: 8 GB Placa de vídeo: RX550 4 GB Armazenamento: 512 GB
Igor Serodio	Processador: Intel Core i7 Memória: 8 GB Placa de vídeo: Nvidia Geforce GTX 660 Armazenamento: 1 TB
João Pedro	Processador: Intel Core i5 Memória: 8 GB Placa de vídeo: — Armazenamento: 500 GB
Luiz Antonio	Processador: Intel Core i7 Memória: 12 GB Placa de vídeo: GeForce 840M Armazenamento: 1 TB

Ody Junior	Processador: Intel Core i7 Memória: 8 GB Placa de vídeo: GTX 1050 Armazenamento: 512 GB
Sanderson Junior	Processador: Intel Core i5 Memória: 8 GB Placa de vídeo: GeForce 820M Armazenamento: 1 TB

Recursos de Software:

- Android Studio;
 - JUnit – framework para testes de casos de uso, em Java
 - libGDX – framework multi-plataforma para desenvolvimento de jogos em Java
- Team Gantt (para a elaboração dos cronogramas);
- Google Docs e Sheets (para demais documentos);
- Trello (para distribuição de atividades);

Não houve nenhum custo na aquisição de ferramentas.

2.6. Comunicação

As atas das reuniões principais — semanais — encontram-se no seguinte repositório:

<https://github.com/Odyjmm/Projeto-ES-2022-Documentos/tree/main/Atas>

Reuniões complementares, i.e. *face to face*, discussões pelo WhatsApp, não foram incluídas. As reuniões realizadas semanalmente englobavam, também, o que se era brevemente discutido em reuniões extras.

3. Processo

Foi utilizado uma metodologia similar à SCRUM — pequenos ciclos de atividades chamados de *sprint*, realizados dentro de um certo período de tempo — mas dadas as circunstâncias de cada membro, adaptamos conforme era necessário.

- Reuniões semanais nas quartas-feiras pela plataforma Discord, com no máximo 1 hora de duração, normalmente no horário da noite após as 18h;
- Atividades separadas na plataforma de gerenciamento de projetos Trello, onde cada integrante era responsável por uma task
- Em um certo momento após a divisão de tarefas, os grupos responsáveis pelo *projeto novo* e pelo *projeto real* faziam suas próprias reuniões em horários/dias distintos;
- Nas reuniões semanais, onde todos os integrantes estavam presentes, era discutido o progresso de ambas as partes — *projeto novo* e *projeto real* — e outras atividades referentes à disciplina de Engenharia de Software, como o fichamento. Dúvidas e previsão de próximas atividades também complementavam tais reuniões;
- Durante a fase inicial do projeto, demonstrou-se necessário mais de uma reunião semanal; duas ou três nas duas primeiras semanas, em virtude da ausência de alguns integrantes e de assuntos ainda então não discutidos;
- Reuniões *face to face* também ocorreram durante o desenvolvimento do projeto. Sentia-se a necessidade de tais encontros, feitos antes ou após o horário da aula, pois era primordial avaliar como os integrantes do grupo estavam se sentindo no quesito psicológico e motivacional;

4. Requisitos

4.1. Requisitos Não-Funcionais

4.1.1. Requisitos de Processo

Ident.	Descrição
--------	-----------

RNF/PROC-01	O sistema deve ser implementado na linguagem JAVA.
RNF/PROC-02	O sistema deve ser desenvolvido utilizando o Android Studio.
RNF/PROC-03	O sistema deve ser construído utilizando um diagrama de classes.
RNF/PROC-04	Os mini jogos devem ser simples de entender.
RNF/PROC-05	A resolução de tela do sistema deve ser definida levando em consideração a resolução de tela padrão de dispositivos móveis atuais.

4.1.2. Requisitos de Produto

Ident.	Descrição
RNF/PROD-01	O sistema não deve ocupar uma quantidade de armazenamento elevada.
RNF/PROD-02	O sistema deve ser compatível com sistemas operacionais Android atuais.

4.1.2.1. Segurança

4.1.2.2. Performance

Ident.	Descrição
RNF/PER-01	O sistema deve ser executado numa taxa de frames por segundo aceitável.
RNF/PER-02	O sistema não deve ter um tempo de loading elevado entre os mini jogos ou ao iniciar o jogo a partir do menu.
RNF/PER-03	Não deve haver um delay considerável entre os inputs do usuário e a execução das ações no sistema.
RNF/PER-04	O sistema não deve consumir uma quantidade elevada da memória RAM do dispositivo durante sua execução.

4.1.3. Requisitos Externos

4.1.3.1. Requisitos Legais

Ident.	Descrição
RNF/LEG-01	O sistema deve utilizar apenas assets criados pela equipe ou que não possuam copyright.

4.1.3.2. Restrições Econômicas

4.2. Requisitos Funcionais

Cód.	Nome	Prioridade
RF-01	Cadastro de Usuário	Essencial
RF-02	Login de Usuário	Essencial
RF-03	Entrada no Jogo	Essencial
RF-04	Consulta ao Ranking	Importante
RF-05	Alteração das Configurações	Importante
RF-06	Menu	Importante
RF-07	Menu de Pausa	Opcional

4.2.1 Especificação dos Requisitos

RF-01	
Nome:	Cadastro de Usuário
Descrição:	O sistema deve permitir que o usuário se cadastre no sistema. Para efetuar esse cadastro será necessário apenas o seu apelido (<i>nickname</i>).
Atores:	Usuário do Sistema, Sistema
Prioridade:	Essencial
Requisitos Não Funcionais Associados:	—
Entradas e pré-condições:	

Saídas e pós-condições:	O usuário é cadastrado no sistema e faz <i>login</i> .
Fluxos de eventos	
Fluxo principal:	<ol style="list-style-type: none"> 1. O usuário informa para a realização do cadastro: <ul style="list-style-type: none"> • Um apelido escolhido pelo usuário; 2. O sistema informa que o cadastro foi realizado com sucesso

RF-02	
Nome:	Login de Usuário
Descrição:	O sistema deve permitir que o usuário efetue <i>login</i> no sistema. Para efetuar esse cadastro serão necessários apenas o seu apelido (<i>nickname</i>) e uma senha. Uma mensagem deve ser exibida caso o apelido já exista ou a senha não seja igual à confirmação.
Atores:	Usuário do Sistema, Sistema
Prioridade:	Essencial
Requisitos Não Funcionais Associados:	—
Entradas e pré-condições:	O usuário ter efetuado o cadastro no sistema.
Saídas e pós-condições:	Menu inicial do jogo.
Fluxos de eventos	
Fluxo principal:	<ol style="list-style-type: none"> 1. O usuário informa os dados necessários para a realização do login: <ul style="list-style-type: none"> • Um apelido escolhido pelo usuário; • Senha; 2. O sistema verifica se o apelido está cadastrado; 3. O sistema verifica se a senha está correta; 4. O sistema passa para o menu inicial do jogo e informa que o login foi realizado com sucesso.
Fluxo secundário 1:	No fluxo principal 2, se o apelido não estiver cadastrado, o sistema exibe uma mensagem informando o ocorrido e retorna ao passo 1.
Fluxo secundário 2:	No fluxo principal 3, se a senha informada não for a senha daquele apelido, o sistema exibe uma

	mensagem solicitando que a senha seja digitada novamente, voltando ao passo 1.
--	--

RF-03	
Nome:	Entrada no jogo
Descrição:	O sistema deve permitir que o usuário consiga entrar no jogo a partir do menu inicial, onde o jogador poderá jogar 5 mini jogos de forma aleatória e conseguir pontos em cada um deles.
Atores:	Usuário do Sistema, Sistema
Prioridade:	Essencial
Requisitos Não Funcionais Associados:	—
Entradas e pré-condições:	O usuário estar no menu inicial.
Saídas e pós-condições:	O usuário começa o jogo.
Fluxos de eventos	
Fluxo principal:	<ol style="list-style-type: none"> 1. O usuário aperta o botão de Jogar no menu inicial; 2. Ao selecionar tal opção, o usuário é levado a um mini jogo selecionado aleatoriamente pelo sistema.

RF-04	
Nome:	Consulta ao Ranking
Descrição:	O sistema deve permitir que o usuário consiga consultar o Ranking das suas 10 melhores pontuações no jogo, assim como as 10 melhores pontuações entre todos os usuários e seus respectivos <i>nicknames</i> .
Atores:	Usuário do Sistema, Sistema
Prioridade:	Essencial
Requisitos Não Funcionais Associados:	—
Entradas e pré-condições:	O usuário estar no menu inicial.
Saídas e pós-condições:	Exibição do Ranking.
Fluxos de eventos	

Fluxo principal:	<ol style="list-style-type: none"> 1. Durante a tela de Menu, são apresentadas ao usuário algumas opções, dentre elas a de 'Ranking'; 2. Ao selecionar tal opção, o usuário é levado a outra tela onde pode consultar os seus pontos. 3. Na tela do ranking pessoal haverá outra opção para ver o ranking geral de todos os usuários.
Fluxo secundário 1:	—
Fluxo secundário 2:	—

RF-05	
Nome:	Alteração das Configurações
Descrição:	O sistema deve permitir que o usuário consiga fazer pequenas alterações no jogo.
Atores:	Usuário do Sistema, Sistema
Prioridade:	Essencial
Requisitos Não Funcionais Associados:	—
Entradas e pré-condições:	O usuário estar no menu inicial ou menu de pausa.
Saídas e pós-condições:	Exibição das possíveis configurações.
Fluxos de eventos	
Fluxo principal:	<ol style="list-style-type: none"> 1. Nas telas do menu inicial e do menu de pausa, o usuário terá a opção de Alterar Configurações; 2. Ao selecionar tal opção, o usuário é levado a outra tela onde pode fazer pequenas alterações no jogo em si.
Fluxo secundário 1:	—
Fluxo secundário 2:	—

RF-06	
Nome:	Acessar o Menu

Descrição:	O sistema, logo assim que aberto, deve exibir um Menu com as opções de Jogar, Ver Ranking e Alterar Configurações.
Atores:	Sistema
Prioridade:	Importante
Requisitos Não Funcionais Associados:	—
Entradas e pré-condições:	O usuário fazer <i>login</i>
Saídas e pós-condições:	Exibição da lista de mini games.
Fluxos de eventos	
Fluxo principal:	<ol style="list-style-type: none"> 1. O jogo inicia, apresentando a tela de Menu; 2. Na tela de Menu, são exibidas as seguintes opções: <ul style="list-style-type: none"> • Jogar; • Ranking; • Tutorial; • Modo Treino; • Configurações; 3. Uma vez escolhida uma das opções anteriores, o sistema segue para a mesma.
Fluxo secundário 1:	—
Fluxo secundário 2:	—

RF-07	
Nome:	Menu de Pausa
Descrição:	O sistema deve ser pausado quando o usuário apertar o botão de pausa dentro do jogo.
Atores:	Usuário do Sistema, Sistema
Prioridade:	Opcional
Requisitos Não Funcionais Associados:	—
Entradas e pré-condições:	O jogo deve estar em andamento, isto é, dentro de algum mini-game.
Saídas e pós-condições:	O jogo retorna ao seu estado normal, de onde havia parado previamente.
Fluxos de eventos	

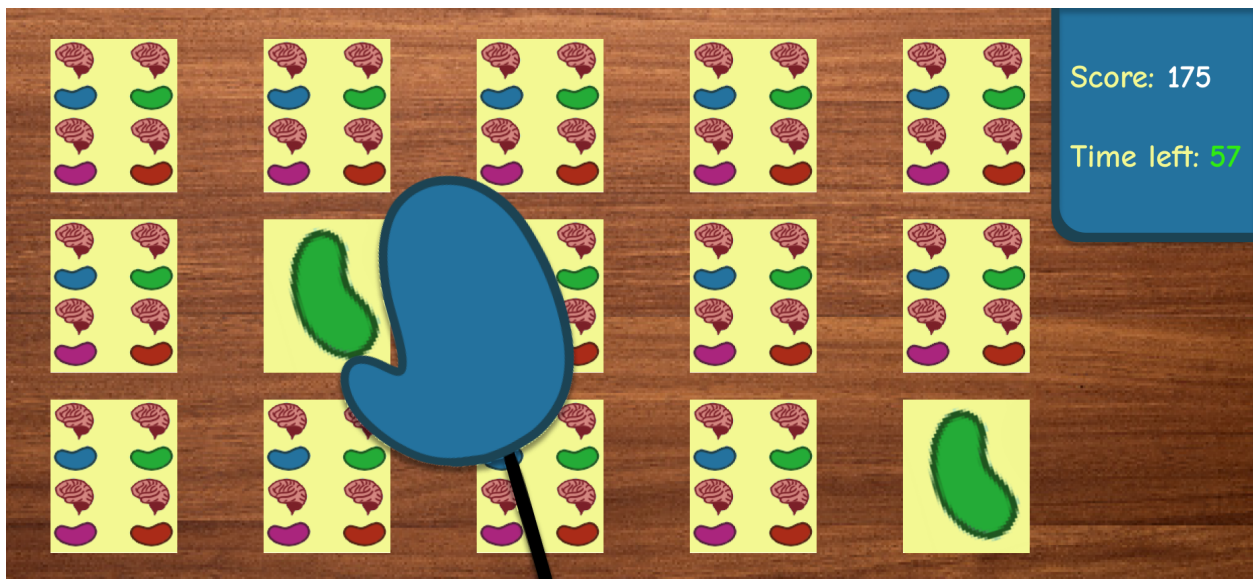
Fluxo principal:	<ol style="list-style-type: none"> 1. Durante o jogo, o jogador pode escolher pausar o jogo através de um botão específico; 2. Uma vez selecionado tal botão, o jogo é pausado, de forma que não interrompa o progresso feito pelo jogador até aquele momento; 3. A tela do Menu de Pausa é aberta, oferecendo algumas opções ao jogador, como a de sair do jogo (o que faria o jogador perder seu progresso) e alterar as configurações.
Fluxo secundário 1:	—
Fluxo secundário 2:	—

4.3. Prototipação de Telas

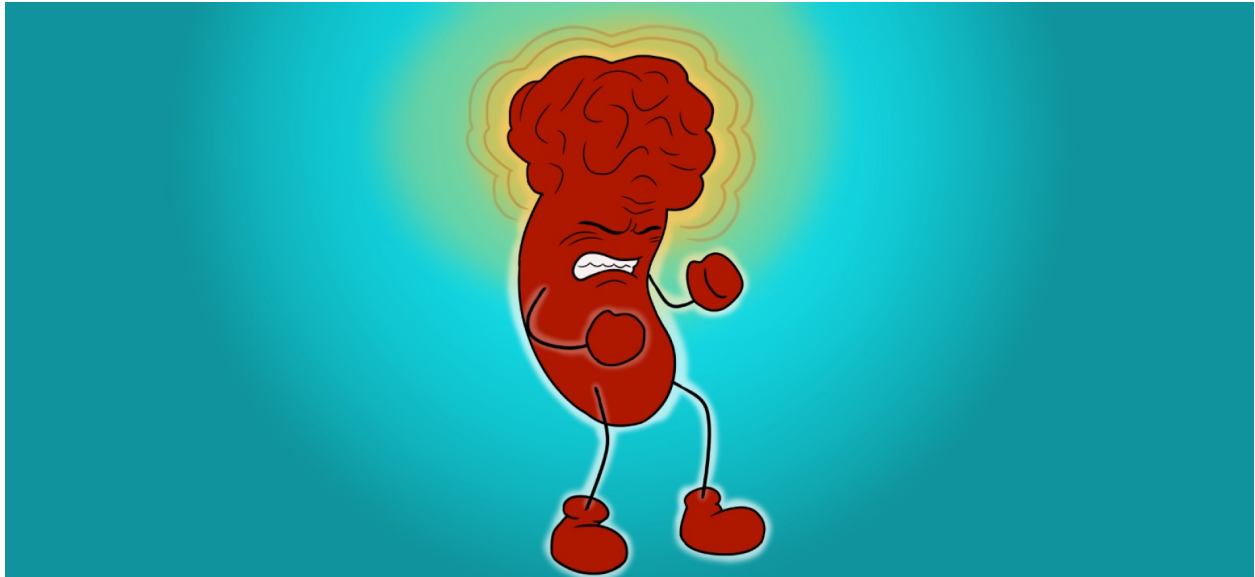
Tela 1 - Menu principal



Tela 2 - Mini-jogo: jogo da memória



Tela 3 - Transição entre mini-jogos (avanço de dificuldade)



5. Projeto Arquitetural

 Documento - Projeto Arquitetural

6. Especificação Formal

7. Implementação

O link para o repositório do projeto é:

<https://github.com/jpedrosml/Projeto-Engenharia-de-Software---2021.2>

8. Testes

Sistema Real

9. Introdução

O projeto real escolhido é uma adaptação da engine de xadrez CuckooChess 1.12, com suporte a maven e uma interface gráfica para representar o tabuleiro de xadrez, o programa é capaz de analisar uma determinada posição e sugerir o próximo melhor movimento, assim como oferece a possibilidade de desafiar a IA.

10. Qualidade

10.1 o3sMeasures (Metrics)

o3sMeasures é um plugin para Eclipse que fornece uma grande variedade de métricas sobre o projeto. De número de classes a Abstração, ele oferece uma excelente visão estrutural do projeto. Ele fornece uma quantidade pré-determinada de métricas, das quais abaixo estão listadas as mais relevantes que foram avaliadas:

- Number of Classes (NOCI)
- Lines of Code (LOC)
- Number of Methods (NOM)
- Number of Attributes (NOA)
- Cyclomatic Complexity (CC)
- Weight methods per Class (WPC)
- Depth of Inheritance Tree (DIT)
- Number of Children (NOCd)
- Coupling between Objects (CBO)
- Fan-Out (FO)
- Response for Class (ROC)
- Tight Class Cohesion (TCC)
- Loose Class Cohesion (LCC)
- Abstractness (Ab)
- Instability (In)
- Distance from Main Sequence (DMS)

Também é possível visualizar e gerar gráficos e tabelas de acordo com a informação obtida, sendo os mais relevantes acessíveis por este [link](#).

10.1.1 Listagem e análise das métricas geradas

Para simplificar o processo de obtenção de métricas, foi analisado apenas o módulo principal do programa, ignorando as classes de interface e de gerenciamento interno dos dados. Pode se observar as métricas geradas na tabela abaixo:

Métrica	Valor	Máximo	Média (Por Classe)
NOCI	57	7	-
LOC	4691	1041	82,289
NOM	411	35	7,211
NOA	288	46	5,053
CC	837	28	14,684
WPC	1201	29	21,07
DIT	40	3	0,702
NOCd	0	2	0,35
CBO	68	8	1,193
FO	38	1	0,667
ROC	717	71	12,579
TCC	2624	0,834	0,046
LCC	2,62	0,834	0,046
Ab	2,143	1	0
In	26,383	0,463	1
DMS	11,617	0,204	1

Como pode ser visto na tabela acima, o módulo principal do projeto real escolhido é composto por 57 classes e tem 4691 linhas de código. Com esses valores, temos a média de 82,3 linhas de código por classe, isso demonstra que as classes são pequenas e de fácil manutenção.

Pela métrica de Abstração (Ab), podemos interpretar que temos mais pacotes concretos que abstratos no projeto, por isso a média é tão baixa que tende a zero.

Pela métrica de Distância até a Sequência Principal (DMS), quando esse número deve ser próximo de zero, é indicado um bom design de pacotes, nesse caso temos uma média de 1 por classe para um valor somado de quase 12, indicando que a organização de pacotes poderia ser melhor.

Pela métrica Complexidade Ciclomática (CC) que indica o número de caminhos diferentes que o programa pode seguir durante a execução, ter um valor relativamente alto não é necessariamente ruim nesse caso, pois a engine deverá analisar vários caminhos diferentes para chegar a uma conclusão satisfatória.

10.2 Spotbugs

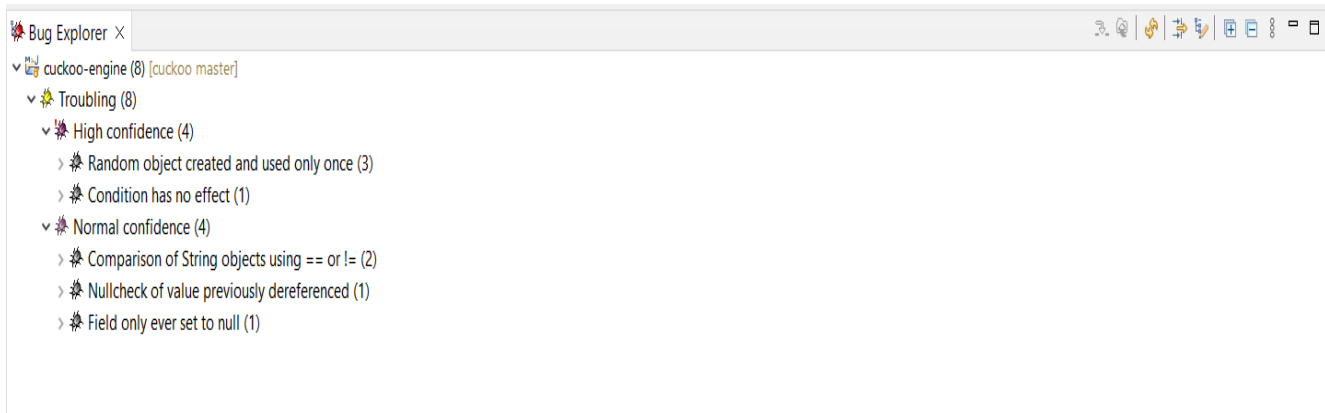
SpotBugs é uma ferramenta de detecção de bugs para Java que usa análise estática para procurar mais de 400 padrões de bug (as descrições dos bugs podem ser encontradas [aqui](#)), como null pointers, loops recursivos infinitos e mau uso das bibliotecas Java.

A ferramenta classifica os problemas encontrados de acordo com a sua gravidade em 4 categorias, são elas:

- Scariest: gravidade entre 1 e 4;
- Scary: gravidade entre 5 e 9;
- Troubling: gravidade entre 10 e 14;
- Of concern: gravidade entre 15 e 20.

Também agrupa os bugs encontrados de acordo com o seu grau de confiança, ou seja, a possibilidade de que esses bugs possam ser marcados como reais.

Executando o plugin no módulo cuckoo-engine do projeto, foram encontrados um total de 8 possíveis bugs. Os bugs foram classificados como indicados na figura abaixo:



10.2.1 Análise e possíveis soluções

Como a ferramenta não encontrou nenhum bug das categorias Scary ou Scariest, podemos concluir que o código foi bem revisado para eliminar possibilidades de falha catastrófica. Dos bugs encontrados, simples alterações são suficientes para satisfazer o algoritmo do Spotbugs, como alterar a criação de um dos campos para algo uma instância de determinado objeto ao invés de criá-lo como *null*, trocar a comparação de strings de “==” para o “.equals()” e observar objetos que são criados e usados somente uma vez.

10.3 Checkstyle

Checkstyle é uma ferramenta de desenvolvimento de código fonte aberto que ajuda programadores a escrever código Java de forma que este siga um padrão de codificação, melhorando assim a qualidade do código, a reusabilidade, clareza, entre outros fatores.

A ferramenta permite verificar, por exemplo:

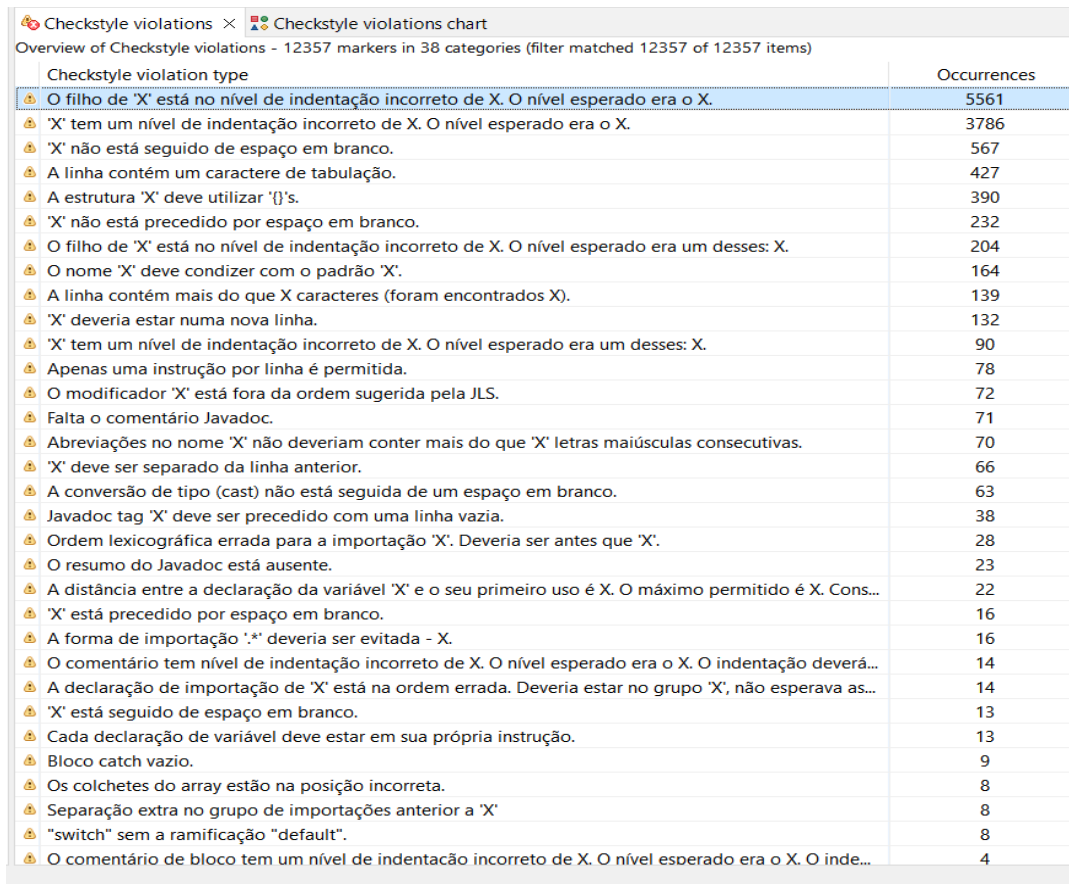
- Comentários Javadoc;
- Cabeçalhos obrigatórios;
- Convenções nos nomes dos atributos e métodos;
- Limite no número de parâmetros;
- A utilização dos pacotes importados nas classes;
- Os espaços entre caracteres;
- Boas práticas no desenvolvimento de classes;
- Código duplicado;

O checkstyle automatiza o processo de verificação do código Java para poupar os programadores dessa tarefa enfadonha, porém necessária. Isso o torna ideal para projetos que desejam impor um padrão de codificação.

10.3.1 Resultados da Análise

De acordo com as métricas obtidas pela ferramenta CheckStyle, o módulo pdfbox apresentou um total de 12.357 violações distribuídas em 38 categorias. Na figura abaixo, é possível ver o exemplo do relatório de violações fornecido pela

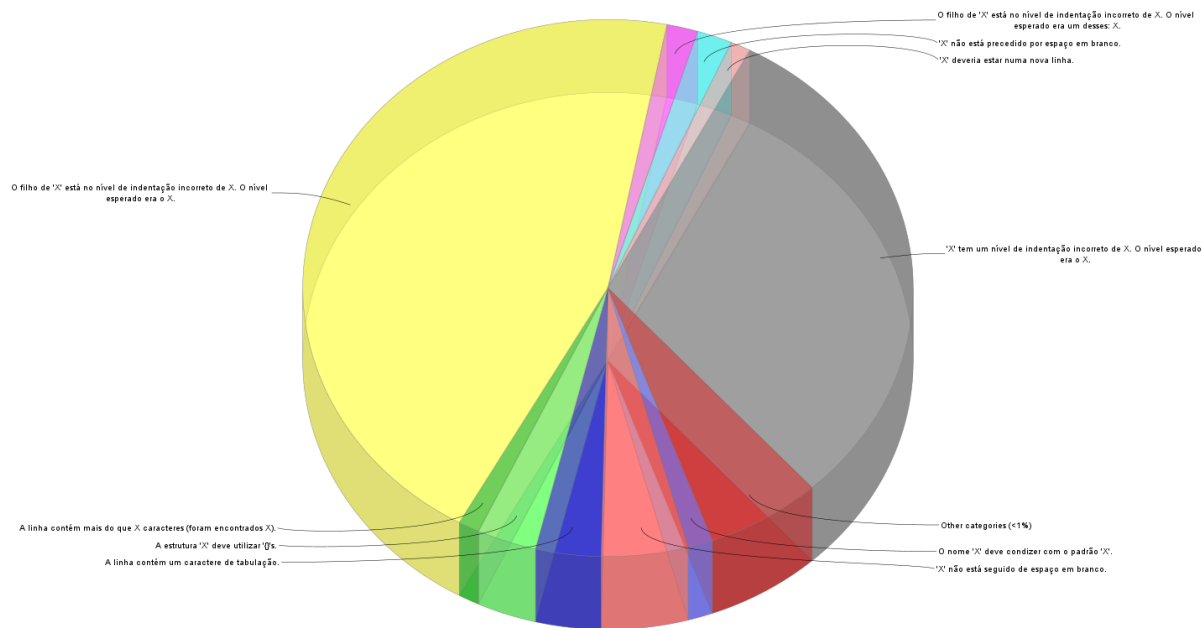
ferramenta. Este consiste em uma relação apresentando o tipo da violação encontrada e o número de ocorrências.



Checkstyle violation type	Occurrences
O filho de 'X' está no nível de indentação incorreto de X. O nível esperado era o X.	5561
'X' tem um nível de indentação incorreto de X. O nível esperado era o X.	3786
'X' não está seguido de espaço em branco.	567
A linha contém um caractere de tabulação.	427
A estrutura 'X' deve utilizar '{}'.s.	390
'X' não está precedido por espaço em branco.	232
O filho de 'X' está no nível de indentação incorreto de X. O nível esperado era um desses: X.	204
O nome 'X' deve condizer com o padrão 'X'.	164
A linha contém mais do que X caracteres (foram encontrados X).	139
'X' deveria estar numa nova linha.	132
'X' tem um nível de indentação incorreto de X. O nível esperado era um desses: X.	90
Apenas uma instrução por linha é permitida.	78
O modificador 'X' está fora da ordem sugerida pela JLS.	72
Falta o comentário Javadoc.	71
Abreviações no nome 'X' não deveriam conter mais do que 'X' letras maiúsculas consecutivas.	70
'X' deve ser separado da linha anterior.	66
A conversão de tipo (cast) não está seguida de um espaço em branco.	63
Javadoc tag 'X' deve ser precedido com uma linha vazia.	38
Ordem lexicográfica errada para a importação 'X'. Deveria ser antes que 'X'.	28
O resumo do Javadoc está ausente.	23
A distância entre a declaração da variável 'X' e o seu primeiro uso é X. O máximo permitido é X. Cons...	22
'X' está precedido por espaço em branco.	16
A forma de importação '*.*' deveria ser evitada - X.	16
O comentário tem nível de indentação incorreto de X. O nível esperado era o X. O indentação deverá...	14
A declaração de importação de 'X' está na ordem errada. Deveria estar no grupo 'X', não esperava as...	14
'X' está seguido de espaço em branco.	13
Cada declaração de variável deve estar em sua própria instrução.	13
Bloco catch vazio.	9
Os colchetes do array estão na posição incorreta.	8
Separação extra no grupo de importações anterior a 'X'	8
"switch" sem a ramificação "default".	8
O comentário de bloco tem um nível de indentação incorreto de X. O nível esperado era o X. O inde...	4

Figura - Exemplo das principais violações encontradas no programa

A ferramenta ainda oferece um gráfico de violações, como apresentado na próxima figura. O gráfico permite uma melhor visualização da predominância das violações encontradas. As mais recorrentes estão descritas na tabela abaixo.



Violação	Número de Ocorrências	Causa
O filho de “X” está no nível de indentação incorreto. O nível esperado era X.	5561	Essa violação refere-se ao nível de indentação utilizado no código que é diferente daquele estabelecido pelo padrão. Por exemplo, dentro de um método o nível de indentação utilizado é 8, mas o esperado é 4.
“X” tem um nível de indentação incorreto. O nível esperado era “X”.	3786	Essa violação refere-se ao nível de indentação utilizado no código que é diferente daquele estabelecido pelo padrão. Por exemplo, dentro de um método o nível de indentação utilizado é 4, mas o esperado é 2.
“X” não está seguindo de um espaço em branco.	567	De acordo com o padrão depois de comandos, como “if”, “catch” ou “for”, estes devem ser seguidos por um espaço em branco antes dos ‘()’, assim como sinais ‘-’, ‘+’, ‘,’ , entre outros, devem ter um espaço em branco antes e depois.

A linha contém um caractere de tabulação	427	De acordo com o padrão, a indentação não pode ser feita com caracteres de tabulação.
A estrutura "X" deve utilizar "{}'s.	390	De acordo com o padrão, o conteúdo de comandos, como "if", "catch" ou "for", deve sempre ser contido por chaves. No código, em várias instâncias, "if's" e "for's" estão ocupando apenas uma linha.

Os problemas de padrão do código apontados pelo Checkstyle são de fácil correção e não comprometem o funcionamento ou performance do módulo. É necessário, no entanto, ter atenção a estes, pois seguir um padrão ajuda a manter a clareza do código e facilita sua compreensão e manutenção.

10.4 PDM

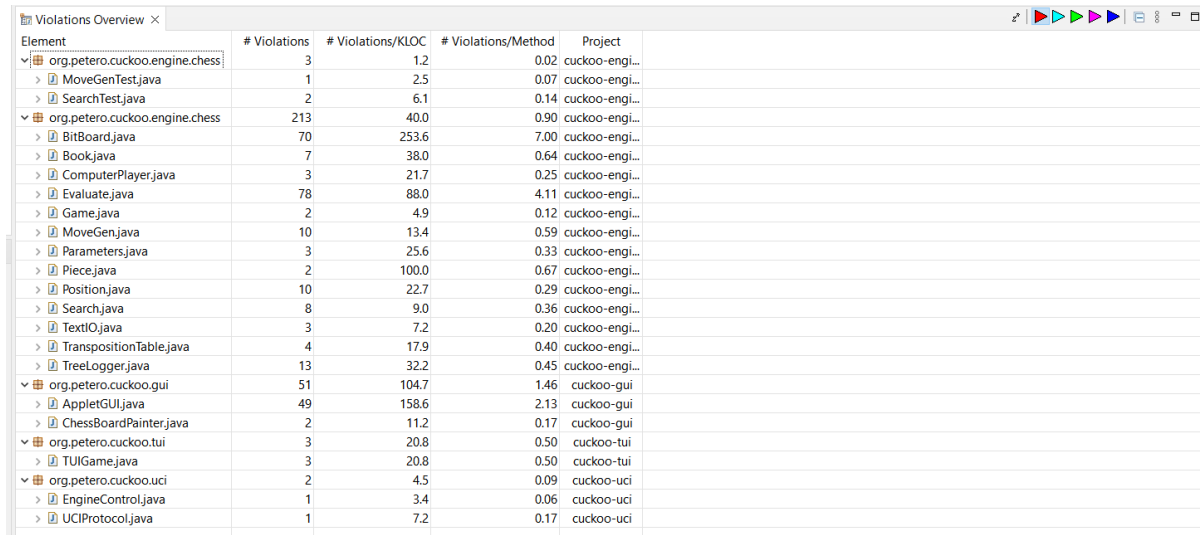
PMD, segundo sua própria documentação, é um analisador de código-fonte. Ele encontra falhas de programação comuns, como variáveis não utilizadas, blocos catch vazios, criação desnecessária de objetos e assim por diante. Suporta Java, JavaScript, Salesforce.com Apex e Visualforce, PLSQL, Apache Velocity, XML, XSL, além de outras seis linguagens de programação.

O PMD é capaz de detectar falhas no código-fonte, como possíveis bugs em blocos try, catch, finally e/ou switch vazios, variáveis locais, parâmetros e métodos privados não usados, declarações if, while vazias, expressões complicadas demais, instruções if desnecessárias, organização de código abaixo do ideal, classes com medidas de alta complexidade e código duplicado.

Quando o plug-in é rodado no código, a análise é possível através de cliques, escolhendo quais classes que devem ser mostradas e as regras que ela está violando. Também pode-se filtrar as regras, habilitando e desabilitando-as, além de obter descrições mais detalhadas sobre a própria regra e suas alternativas de correção. Essas regras se dividem em cinco grandes grupos numa escala crescente de prioridade, são eles warnings, urgent, important, critical e blocker.

10.4.1 Resultados da análise do PMD

Analisando os resultados obtidos com a ferramenta PMD, podemos notar alguns sinais de alerta no código do projeto CuckooChess: Ao total, o projeto apresentou 9.670 violações de regras. Dessas, 325 são consideradas Blocker Violations, 233 Critical Violations, 8.151 Urgent Violations, 514 Important Violations e 447 Warning Violations.



Element	# Violations	# Violations/KLOC	# Violations/Method	Project
org.petero.cuckoo.engine.chess	3	1.2	0.02	cuckoo-engi...
> MoveGenTest.java	1	2.5	0.07	cuckoo-engi...
> SearchTest.java	2	6.1	0.14	cuckoo-engi...
org.petero.cuckoo.engine.chess	213	40.0	0.90	cuckoo-engi...
> BitBoard.java	70	253.6	7.00	cuckoo-engi...
> Book.java	7	38.0	0.64	cuckoo-engi...
> ComputerPlayer.java	3	21.7	0.25	cuckoo-engi...
> Evaluate.java	78	88.0	4.11	cuckoo-engi...
> Game.java	2	4.9	0.12	cuckoo-engi...
> MoveGen.java	10	13.4	0.59	cuckoo-engi...
> Parameters.java	3	25.6	0.33	cuckoo-engi...
> Piece.java	2	100.0	0.67	cuckoo-engi...
> Position.java	10	22.7	0.29	cuckoo-engi...
> Search.java	8	9.0	0.36	cuckoo-engi...
> TextIO.java	3	7.2	0.20	cuckoo-engi...
> TranspositionTable.java	4	17.9	0.40	cuckoo-engi...
> TreeLogger.java	13	32.2	0.45	cuckoo-engi...
org.petero.cuckoo.gui	51	104.7	1.46	cuckoo-gui
> AppletGUI.java	49	158.6	2.13	cuckoo-gui
> ChessBoardPainter.java	2	11.2	0.17	cuckoo-gui
org.petero.cuckoo.tui	3	20.8	0.50	cuckoo-tui
> TUIGame.java	3	20.8	0.50	cuckoo-tui
org.petero.cuckoo.uci	2	4.5	0.09	cuckoo-uci
> EngineControl.java	1	3.4	0.06	cuckoo-uci
> UCIProtocol.java	1	7.2	0.17	cuckoo-uci

Figura: Funcionamento do PMD, filtrando os erros de maior prioridade (Blocker)

10.4.2 Interpretação dos resultados da análise do PMD

Uma cuidadosa análise dos dados gerados pelo PMD nos mostra que cada erro deve ser visto de maneira individual tendo em vista que o programa não avalia os possíveis erros em tempo de execução. Sendo assim, é evidente que, devido ao grande número de casos de erro, é possível, sim, refatorações de código que gerem melhorias na qualidade e desempenho do sistema.

11. Testes

12. Evolução

13. Conclusões