UNIVERSIDADE FEDERAL DO RIO GRANDE – FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Vinícius Borges Martins

GorimWEB: um RPG para Gestão de Recursos Hídricos na plataforma Web

Rio Grande, 2022

UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

GorimWEB: um RPG para Gestão de Recursos Hídricos na plataforma Web

Vinícius Borges Martins

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande - FURG, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientadora: Profª. Drª. Diana Francisca Adamatti
Coorientadora: Profª. Drª. Raquel de Miranda Barbosa

Rio Grande, 2022

Ficha Catalográfica

M386g Martins, Vinícius Borges.
GorimWEB: um RPG para Gestão de Recursos Hídricos na plataforma Web / Vinícius Borges Martins. – 2022.
130 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Computação, Rio Grande/RS, 2022.
Orientadora: Dra. Diana Francisca Adamatti.
Coorientadora: Dra. Raquel de Miranda Barbosa.

1. Jogos de Papéis 2. Gerenciamento de Recursos Hídricos 3. Plataforma Web I. Adamatti, Diana Francisca II. Barbosa, Raquel de Miranda III. Título.

CDU 004:556.18

Catalogação na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

GorimWEB: um RPG para Gestão de Recursos Hídricos na plataforma Web

Vinícius Borges Martins

Banca examinadora:



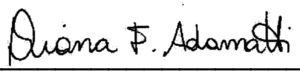
Prof. Dr. Cleo Zanella Billa



Prof. Dr. Marilton Sanchotene de Aguiar



Profa. Dra. Raquel de Miranda Barbosa
Coorientadora



Profa. Dra. Diana Francisca Adamatti
Orientadora

AGRADECIMENTOS

Agradeço a todos que me ajudaram no processo de estudos, desenvolvimento, teste e escrita desse texto, ainda que indiretamente.

Um obrigado extremamente especial à minha orientadora e à minha coorientadora que me aguentaram falando várias vezes que "eu sei que estou atrasado, eu só não consegui fazer" e fizeram muitas e muitas vezes discursos motivacionais pra me fazer chegar ao final do projeto. Por ficarem comigo durante as partidas como Mestre também. A companhia de vocês fez as coisas ficarem bem mais leves e o medinho de não dar certo *de novo* foi menor com vocês lá.

Agradeço também à minha família, visto que eu sumi por um bom tempo de almoços, jantas, etc, com um único motivo: precisava fazer as coisas do mestrado. Houve erros, acertos, reestruturação do código, mais erros, mais *bugfixes*, um pouco de magia negra (como diz o Dado) e uma pitada de sorte por cair nos fóruns com respostas que davam realmente certo. Enfim, é um obrigado e um pedido formal de desculpas também.

Aos meus colegas de laboratório e projeto que me ajudaram desde o início do processo, dando ideias sobre o que achavam que era importante nas interfaces e nos primórdios do Motor do Jogo. Agradeço especialmente à Giulia Tondin (para mim, Felicidade) que fez a fôrma das interfaces a partir de desenhos. Sério, eu não teria terminado a tempo (ok ok, não terminei exatamente à tempo, mas isso fica implícito) se não fosse por ti, Felicidade.

Agradeço também aos jogadores que se disponibilizaram para testar a interface e deram sugestões e *feedbacks* extremamente valiosas. Obrigado por se engajarem assim.

Por fim, agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e à Agência Nacional das Águas (ANA) pelo apoio financeiro via Edital nº 16/2017.

The trouble with having an open mind, of course, is that people will insist on coming along and trying to put things in it.
— TERRY PRATCHETT

RESUMO

MARTINS, Vinícius Borges. **GorimWEB: um RPG para Gestão de Recursos Hídricos na plataforma Web**. 2022. 127 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

Jogos computacionais têm sido cada vez mais utilizados para diversão, mas também para treinamentos de empresas. *Role-Playing Games* são um dos tipos de jogos que pode ser utilizado para esse fim. Este trabalho tem como principal objetivo a implementação e análise de um jogo do tipo RPG em versão *web*, denominado GorimWEB. Este trabalho se encontra inserido no projeto nº 16/2017 da Agência Nacional das Águas (ANA). Para implementação desta versão do jogo, foi realizado o levantamento dos seguintes requisitos funcionais: apresentar a jogabilidade encontrada no Jogo de Mesa, guardar e exportar dados (como logs e chat), dar suporte a quantidades variáveis de jogadores e dar suporte a sistemas multiagente; e, como não funcionais: desempenho, informações bem estruturadas na interface e disponibilidade via *web*, bem como foi definida uma arquitetura de *software* para o mesmo. De forma a validar o jogo proposto, duas hipóteses foram definidas: (I) Jogadores do Grupo 1 (que jogaram o jogo em sua forma física) terminam as jogadas mais rápido e realizam um comparativo entre interface web e física, visto que já tiveram uma interação prévia com o jogo; e (II) Jogadores do Grupo 2 (que nunca jogaram Gorim) tem mais questionamentos sobre o material de informações necessários aos seus papéis e jogo, visto que nunca jogaram Gorim; e os requisitos definidos foram avaliados por meio de questionários pré e pós jogo. Como resultados, tem-se a análise das respostas dos questionários e das mensagens trocadas durante as duas partidas, análise essa que resultou na validação das interfaces e das hipóteses. As ideias e considerações dos jogadores serão usadas para refinar ainda mais o GorimWEB, como trabalhos futuros.

Palavras-chave: Jogos de Papéis, Gerenciamento de Recursos Hídricos, Plataforma Web.

ABSTRACT

MARTINS, Vinícius Borges. **GorimWEB: an Hidric Resources Management RPG over the web**. 2022. 127 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

Computational games has been increasingly used for fun, but also as companies trainings. Role-Playing Games are a kind of game that should be used as training. This project has as main objective the implementation and the analysis for a web version of a RPG, which was called GorimWEB. The project described in this text is part of a bigger project, defined in the public notice number 16/2017 from National Water Agency (ANA – from Portuguese "Agência Nacional das Águas"). For the implementation functional and non-functional requirements were defined: support the game board jogability, store and export data (as logs and chat), support different quantity for players and support multiagent systems as functional, and performance, well structured info in the interface and web availability as non-functional, as well as define the software architecture of the game. The validation was made with two surveys, one before and one after the gameplays. Two hypotheses were created: (I) Players from Group 1 (that played the board game) end their actions faster and compare the web and board game interfaces hence the previous interaction with the game; and (II) Players from Group 2 (the ones who never played Gorim) have more questions regarding the info needed for the game and the roles in it. As results, we validated the interfaces and the hypotheses using the answers for the surveys and the messages exchanged via chat during the game. The ideas and considerations done by the players are being used to improve even more the GorimWEB, as future works.

Keywords: Role-Playing Games, Hidric Resources Management, Web Platform.

LISTA DE FIGURAS

Figura 1 Grupos que separam os trabalhos com RPG e SMA (FARIAS et al., 2019). 24

Figura 2 Fluxograma da metodologia de desenvolvimento do trabalho. 31
Figura 3 Diagrama em visão conceitual da arquitetura proposta. 33
Figura 4 Fichas preenchíveis pelos jogadores, onde: A – Empresário, B – Fiscal Ambiental, C – Prefeito, D – Vereador e E – Agricultor (este recebe seis fichas, um representando cada Parcela). 37
Figura 5 Aquivos Resumo recebido pelos jogadores ao final de cada rodada. . 38
Figura 6 Diagrama de Classes do Motor do Jogo. 39
Figura 7 Diagrama de Classes – Classe `Pessoa`. 40
Figura 8 Diagrama de Classes – Classes `Empresario`, `Produto`, `Maquina`, `Semente`, `Agrotoxico` e `Fertilizante`. 40
Figura 9 Diagrama de Classes – Classes `Agricultor` e `Parcela`. 41
Figura 10 Diagrama de Classes – Classe `FiscalAmbiental`. 41
Figura 11 Diagrama de Classes – Classes `Prefeito` e `AcaoAmbiental`. . . 42
Figura 12 Diagrama de Classes – Classe `Vereador`. 42
Figura 13 Alunos e professores participando de uma partida do Gorim. 44
Figura 14 Broches criados para a identificação dos personagens durante as partidas presenciais. 45
Figura 15 Arquitetura em visão conceitual do componente *backend* da Figura 3. 46
Figura 16 Diagrama de Componentes do *frontend*. 47
Figura 17 Interface da página inicial (`home`) 48
Figura 18 Diagrama de blocos do *módulo* `home` 48
Figura 19 Interface do Mestre (`master`). 49
Figura 20 Diagrama de blocos do *módulo* `master`. 49
Figura 21 Diagrama de blocos do *módulo* `world` 50
Figura 22 Interface da votação (`voting`). 51
Figura 23 Interface do chat. 51
Figura 24 Fluxo do recebimento de um evento do chat. 52
Figura 25 Definição do tipo `Person`. 53
Figura 26 Interface do Empresário (`businessman`). 53
Figura 27 Diagrama de blocos do *módulo* `businessman`. 54
Figura 28 Interface do Agricultor (`farmer`). 54
Figura 29 Diagrama de blocos do *módulo* `farmer`. 55
Figura 30 Formulário referente à Parcela 1 na interface do Agricultor. 55

Figura 31 Aba lateral mostrando (A) os orçamentos recebidos pelo Agricultores (B) as respostas aos orçamentos recebidos na interface do Empresário. 56
Figura 32 Interface do Fiscal Ambiental (`supervisor`). 57
Figura 33 Diagrama de blocos do *módulo* `supervisor`. 57
Figura 34 Interface dos formulários "Multar" e "Selo Verde" do Fiscal Ambiental. 58
Figura 35 Interface da janela de histórico do Empresário. 59
Figura 36 Interface da janela de histórico do Fiscal Ambiental. 59
Figura 37 Interface do Prefeito (`mayor`). 60
Figura 38 Diagrama de blocos do *módulo* `mayor`. 60
Figura 39 Interface dos formulários "Usar Ação Ambiental" e "Mudar Taxas de Imposto" do Prefeito. 61
Figura 40 Interface do Vereador (`alderman`). 62
Figura 41 Diagrama de blocos do *módulo* `alderman`. 62
Figura 42 Interface dos jogadores sem papel na segunda Etapa (`alternative-second-page-stage`). 63
Figura 43 Diagrama de blocos do *módulo* `alternative-second-page-stage`. 63
Figura 44 Interface do fim do jogo (`game-over`). 64
Figura 45 Diagrama de blocos do *módulo* `game-over`. 64

Figura 46 Dispersão das mensagens em função do tempo (em segundos) da partida do Grupo 1. 70
Figura 47 Dispersão das mensagens em função do tempo (em segundos) da partida do Grupo 2. 70

LISTA DE TABELAS

Tabela 1 Tabela comparativa entre os trabalhos relacionados com RPG. 29
Tabela 2 Tabela comparativa entre os trabalhos relacionados com arquitetura de *software*. 30

Tabela 3 Tipos de eventos do jogo possíveis 52
Tabela 4 Lista com os Game Status 52

Tabela 5 Questões do formulário pré-jogo 65
Tabela 6 Questões do formulário pós-jogo 66
Tabela 7 Questões 8 a 12 do Questionário Pós-Teste - Grupo 1 74
Tabela 8 Questões 8 a 12 do Questionário Pós-Teste - Grupo 2 74
Tabela 9 Questões 8 a 12 do Questionário Pós-Teste - Média Ponderada 75

LISTA DE ABREVIATURAS E SIGLAS

ANA	Agência Nacional das Águas
ComMod	*Companion Modelling*
CSV	*Comma-Separated Values*
GMABS	*Games and MultiAgent Based Systems*
GRH	Gerenciamento de Recursos Hídricos
GRN	Gerenciamento de Recursos Naturais
IA	Inteligência Artificial
MMORPG	*Massively Multiplayer Online Role-Playing Game*
NPC	*Non-Player Character*
ONG	Organização Não Governamental
RPG	*Role-Playing Game*
RSL	Revisão Sistemática da Literatura
SINGREH	Sistema Nacional de Gerenciamento de Recursos Hídricos
SMA	Sistema Multiagente
SPA	*Single-Page Application*

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Objetivos do Trabalho 14
1.2 Hipóteses definidas 14
1.3 Estrutura do Texto 15

2 EMBASAMENTO TEÓRICO E TECNOLÓGICO 16
2.1 Jogos de Papéis 16
2.2 Gerenciamento de Recursos Hídricos 17
2.3 Arquitetura de *Software* 18
2.4 Ferramentas Utilizadas 22
2.4.1 JAVA e Spring 22
2.4.2 Angular 22
2.5 Trabalhos Relacionados 23

3 GORIMWEB 31
3.1 Metodologia 31
3.1.1 Requisitos Funcionais e Não Funcionais 32
3.1.2 Arquitetura 33
3.2 Gorim 35
3.2.1 Estrutura do Jogo 35
3.2.2 Motor do Jogo 37
3.2.3 Testes com o Gorim 43
3.3 Implementação do GorimWEB 44

4 TESTES E VALIDAÇÃO DO GORIMWEB 65
4.1 Estrutura dos testes realizados 65
4.2 Grupo 1 – Jogadores que já jogaram 67
4.3 Grupo 2 – Jogadores que não jogaram 68
4.4 Análise comparativa entre grupos de teste 69
4.4.1 Análise das Hipóteses Propostas 69
4.4.2 Pré-Teste 71
4.4.3 Pós-Teste 73

5 CONCLUSÕES E TRABALHOS FUTUROS 78

REFERÊNCIAS 80

APÊNDICE A RESPOSTAS AOS QUESTIONÁRIOS - GRUPO 1 83
A.1 Jogador 1 (G1-J1) 84
A.1.1 Pré-Jogo 84
A.1.2 Pós-Jogo 84
A.2 Jogador 2 (G1-J2) 86
A.2.1 Pré-Jogo 86
A.2.2 Pós-Jogo 86
A.3 Jogador 3 (G1-J3) 88
A.3.1 Pré-Jogo 88
A.3.2 Pós-Jogo 88
A.4 Jogador 4 (G1-J4) 90
A.4.1 Pré-Jogo 90
A.4.2 Pós-Jogo 90
A.5 Jogador 5 (G1-J5) 91
A.5.1 Pré-Jogo 91
A.5.2 Pós-Jogo 92
A.6 Jogador 6 (G1-J6) 94
A.6.1 Pré-Jogo 94
A.6.2 Pós-Jogo 94

APÊNDICE B RESPOSTAS AOS QUESTIONÁRIOS - GRUPO 2 97
B.1 Jogador 1 (G2-J1) 98
B.1.1 Pré-Jogo 98
B.1.2 Pós-Jogo 98
B.2 Jogador 2 (G2-J2) 100
B.2.1 Pré-Jogo 100
B.2.2 Pós-Jogo 100
B.3 Jogador 3 (G2-J3) 102
B.3.1 Pré-Jogo 102
B.3.2 Pós-Jogo 102
B.4 Jogador 4 (G2-J4) 104
B.4.1 Pré-Jogo 104
B.4.2 Pós-Jogo 104
B.5 Jogador 5 (G2-J5) 106
B.5.1 Pré-Jogo 106
B.5.2 Pós-Jogo 107
B.6 Jogador 6 (G2-J6) 108
B.6.1 Pré-Jogo 108
B.6.2 Pós-Jogo 109

ANEXO A TABELAS DE VALORES DOS PERSONAGENS DO JOGO GORIM 111

1 INTRODUÇÃO

Jogos têm sido uma ferramenta de entretenimento e ensino. Com o aumento do uso de aparelhos eletrônicos (como *smartphones*, computadores e *tablets*) por pessoas de todas as idades, os jogos evoluíram para serem usados nesses dispositivos, inclusive de forma cooperativa.

Um dos tipos de jogos é o Jogo de Papéis (*Role-Playing Game* - RPG), onde os jogadores assumem as características dos personagens e agem dentro dos movimentos disponíveis para aquele personagem, tendo que atingir o objetivo do jogo. O RPG pode ser jogado de forma oral (onde um *Mestre* narra o ambiente e como ele está se desenrolando com base nas ações dos jogadores) ou por meio de um tabuleiro (onde os jogadores acompanham o ambiente de forma visual e se locomovem dentro de um mapa) (PEREIRA, 2004; ADAMATTI, 2007; BRAUNER et al., 2013). Assim como os outros tipos de jogos, o RPG também teve a transição para as plataformas computacionais, aumentando as possibilidades de interfaces dos jogos e inserção de instrumentos computacionais, como Inteligência Artificial (IA).

A área de Gerenciamento de Recursos Naturais estuda as formas de administrar águas, terras, plantas e animais buscando a qualidade destes para o futuro (FULLER et al., 2007). Dentro desta área, tem-se o Gerenciamento de Recursos Hídricos (GRH), que discute melhores práticas de distribuição e uso da água, envolvendo organizações e diferentes grupos sociais e é responsável por situações ambientais com consequências sociais que necessitam de ações preventivas como enchentes e secas. Para essas tomadas de decisões, são criados comitês de gerenciamento para regiões específicas (TUNDISI, 2006).

RPG tem sido uma ferramenta utilizada em diversos projetos (BARNAUD et al., 2007; ADAMATTI; SICHMAN; COELHO, 2007; CAMPO et al., 2009; PAGE et al., 2016; PAGE; PERROTTON, 2017) para a resolução dos problemas na área de GRH devido à sua interdisciplinaridade e facilidade quanto ao engajamento das pessoas para jogarem e para assimilarem o conhecimento do jogo, que pode estar explícita ou implicitamente sendo repassado.

O jogo chamado Gorim (LEITZKE et al., 2019) é um RPG criado com base no Comitê de Gerenciamento da Bacia Hidrográfica da Lagoa Mirim e do Canal São Gonçalo, que

tem como objetivo simular os papéis dos membros do Comitê com suas ações. O projeto[1] para criação do jogo (via Edital nº 16/2017 da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior/Agência Nacional das Águas [CAPES/ANA]) busca o desenvolvimento de um RPG a ser jogado pelos membros do Comitê, onde cada um jogará como outro membro, aprendendo, então, suas necessidades e consequências de suas ações, e, ao tomarem decisões em relação ao ambiente onde a bacia está inserida, tentarão buscar a alternativa que traga o maior número de vantagens para todos e, principalmente, para o meio ambiente.

O Gorim tem um formato onde os jogadores jogam (presencialmente) e definem suas ações em uma planilha (em papel), que é computada pelo *motor* do jogo, desenvolvido em Java. Este formato de jogo (com os dados sendo processado computacionalmente) foi o escolhido, visto que as ações de cada um dos jogadores estão atreladas a diversas funcionalidades e variáveis, o que demanda grande número de cálculos. Contudo, a operacionalização das ações escolhidas em um formato *web* é muito importante, pois permite que as jogadas sejam computadas mais rapidamente e também que a participação dos jogadores possa ocorrer em diferentes locais remotos. Dentro deste contexto, uma versão Web do jogo Gorim para uma versão Web é um passo importante para no escopo do projeto como um todo.

1.1 Objetivos do Trabalho

Este trabalho tem como objetivo geral a implementação e análise de um jogo do tipo RPG em versão *web*, denominado GorimWEB.

Para cumprir com este objetivo, foram traçados como objetivos específicos:

1. Levantamento de requisitos funcionais e não-funcionais;
2. Definição de uma arquitetura de *software* para a implementação do GorimWEB;
3. Validação do GorimWEB a partir dos requisitos definidos.

Visto que tanto para a versão em papel do Gorim, quanto para a versão web (GorimWEB) seria necessário um módulo computacional (vide Motor de Cálculos, na Seção 3.2.2), a implementação deste módulo fez parte do desenvolvimento deste trabalho.

1.2 Hipóteses definidas

A validação dos requisitos funcionais e não funcionais (prevista nos objetivos deste trabalho) foi realizada por dois grupos de jogadores:

- Grupo 1 (G1): jogadores que já jogaram o jogo Gorim;

[1] http://gprh.c3.furg.br/

- Grupo 2 (G2): jogadores que nunca jogaram o Gorim e irão jogar unicamente o GorimWEB.

A partir destes grupos, tem-se duas hipóteses formuladas:

- Hipótese 1 (H1) – Jogadores G1 terminam as jogadas **mais rápido** e realizam um **comparativo** entre interface web e física (jogo de mesa), visto que já tiveram uma interação prévia com o jogo.
- Hipótese 2 (H2) – Jogadores do G2 tem **mais questionamentos** sobre material de informações necessários aos seus papéis e jogo, visto que nunca jogaram o jogo.

1.3 Estrutura do Texto

O texto está dividido em capítulos onde, a partir do próximo, tem-se o Embasamento Teórico e Tecnológico (Cap. 2), onde se discute sobre as áreas, as ferramentas utilizadas para o desenvolvimento e os trabalho relacionados a este; o Capítulo 3 onde se explicita o GorimWEB, desde a metodologia utilizada para a criação do jogo, passando pela explicação da versão de mesa, até implementação do produto final deste trabalho; os Testes e Análise feitos sobre o jogo (Cap. 4), onde se explica a estrutura dos testes realizados, as características dos dois grupos de jogadores e os resultados dos testes e das hipóteses; e, por fim, as Conclusões e Trabalhos Futuros. Após, ainda há os Apêndices (com as respostas dos jogadores aos questionários) e o Anexo (onde se apresenta os cartões preenchíveis do Gorim de mesa).

2 EMBASAMENTO TEÓRICO E TECNOLÓGICO

Este capítulo apresenta a base teórica necessária para o desenvolvimento do trabalho. Serão apresentadas definições e características das áreas do trabalho: Jogos de Papéis, Gerenciamento de Recursos Hídricos e Arquitetura de *Software*, bem como as ferramentas utilizadas no desenvolvimento deste trabalho. Além disso, serão apresentados trabalhos relacionados que foram desenvolvidos dentro destas áreas.

2.1 Jogos de Papéis

Segundo o Dicionário Oxford, Jogo é definido por:

> Uma atividade que você faz para se divertir, muitas vezes tem regras e que você pode ganhar ou perder.

Existem vários tipos de jogos, como Jogos de Cartas, Jogos de Bola, Jogos de Peças ou Jogos de Papéis. Os Jogos de Papéis são jogos que estão presentes também dentro do escopo do teatro, visto que os jogadores interpretam as ações dos personagens dos jogos como sendo suas próprias para atingir os objetivos do jogo (BARRETEAU; PAGE; D'AQUINO, 2003). Porém, os Jogos de Papéis não têm uma abordagem competitiva, pois, se um jogador falha, significa que o grupo falhou em solucionar um problema e o objetivo geral do jogo pode ser perdido (BRAUNER et al., 2013).

O RPG pode ser jogado de três formas: impresso (por meio de um tabuleiro, ou fichas), de forma oral (onde o mestre narra o ambiente onde os personagens estão inseridos na história e os jogadores tomam as decisões girando um dado para saber se a ação pode ou não ser tomada) ou por um dispositivo computacional (onde insere os jogadores virtualmente em um ambiente).

Hoje, com a evolução da computação, tem-se os chamados MMORPGs (do inglês *Massively Multiplayer Online Role-Playing Games* – Jogos de Papéis Online para quantidades massivas de jogadores), onde os jogadores, de forma online, se inserem em um ambiente virtual dinâmico. Pode ser por meio do navegador ou por meio de aplicações próprias dos desenvolvedores.

Na área de interação humana, há um estudo que propõe quatro formas de interação quando depende de distância física e temporal (ELLIS; GIBBS; REIN, 1991):

- *Interação cara a cara.* Quando os indivíduos se encontram presentes no mesmo espaço físico ao mesmo tempo;
- *Interação distribuída síncrona.* Quando os indivíduos estão separados fisicamente, mas interagindo no mesmo momento;
- *Interação assíncrona.* Quando os indivíduos interagem em tempos diferentes no mesmo local;
- *Interação distribuída assíncrona.* Quando eles se encontram separados fisicamente e interagem em tempos diferentes.

Levando em consideração que o RPG está focado na interação humana (BRAUNER et al., 2013), a distribuição proposta por esse grupo pode também ser usada para os RPGs, podendo ocorrer em todas as quatro formas propostas por Ellis, Gibbs e Rein (1991).

Atualmente, os RPGs são utilizados, além de diversão fantasiosa, para treinamentos, pois é possível inserir os colaboradores em situações para serem testados (e treinados) tendo somente o aprendizado como consequência no mundo real. Visto que o RPG é um tipo de jogo, tais situações de treinamento acontecem de forma facilitada (PERROTTON et al., 2017; BARRETEAU; BOUSQUET; ATTONATY, 2001).

O RPG ainda permite ao jogador estudar formas de resolver problemas, visto que os personagens (e, por isso, os jogadores) passam por diversas situações (mesmo que fantasiosas). No mundo real, o jogador, após toda a carga emocional de ter enfrentado tais situações empenhando um papel, termina com a habilidade de resolver seus próprios problemas reais com maior rapidez e com mais alternativas (BANDINI; MANZONI; VIZZARI, 2002).

2.2 Gerenciamento de Recursos Hídricos

O aumento de poluição no mundo, causado pela crescente produção de lixo, gases tóxicos e produtos químicos utilizados no solo e despejados na água, vem cada vez mais se mostrando presente, trazendo, como consequências, a falta de qualidade na água e ar, morte de animais e falta da água (European Environment Agency, 2018).

A área de Gerenciamento de Recursos Naturais (GRN) é a área que estuda e combate o problema apresentado, buscando melhorar a qualidade dos recursos naturais disponíveis hoje para as próximas gerações. Essa área, portanto, gerencia terras, água, animais e plantas (FULLER et al., 2007).

A água, porém, é o recurso natural mais vital para os seres vivos e todo o ecossistema, além de ser um importante fator em questões sociais e econômicas (PONTE et al., 2016).

A GRN pode ser subdividida em outra área, chamada Gestão de Recursos Hídricos (GRH). Esta, por sua vez, tem como objetivo específico a água e como resolver os problemas relacionados a esse bem, como distribuição e tratamento. Ainda, é dos representantes dessa área a resolução de problemas quanto às situações de risco ambiental relacionado à água, como secas e enchentes (TUNDISI, 2006).

No Brasil, a Política Nacional de Recursos Hídricos, instituída pela Lei das Águas (Lei nº 9.433/97), definiu uma série de instrumentos para a tomada de decisão quanto à água no âmbito nacional, criando o Sistema Nacional de Gerenciamento de Recursos Hídricos (SINGREH)[1].

O SINGREH é composto por: Conselho Nacional de Recursos Hídricos, Secretaria de Recursos Hídricos e Qualidade Ambiental, Agência Nacional das Águas, Conselhos Estaduais de Recursos Hídricos, Órgãos Gestores de Recursos Hídricos Estaduais, Comitês de Bacia Hidrográfica e Agências de Água. O SINGREH prevê quatro principais objetivos[2]:

- Coordenar a gestão integrada das águas;
- Arbitrar administrativamente os conflitos relacionados aos recursos hídricos;
- Planejar, regular e controlar o uso, bem como a recuperação dos corpos d'água;
- Promover a cobrança pelo uso da água.

Uma das áreas mais estudadas quando se trata de água são as Bacias Hidrográficas. Estas são áreas que, devido ao relevo, as águas que saem de nascentes e da chuva descem até o fundo do vale e saem por um único ponto, normalmente caracterizado por ser um rio (AHMAD; SIMONOVIC, 2005). No estado do Rio Grande do Sul, uma das bacias hidrográficas é a Bacia Hidrográfica da Lagoa Mirim e do Canal São Gonçalo e, para gerenciá-la, foi montado um Comitê de Gerenciamento. Os jogos desenvolvidos neste trabalho são baseados neste comitê.

2.3 Arquitetura de *Software*

Em um projeto, ao se pensar na construção de um sistema, pensa-se na sua estrutura, tanto funcional, quanto organizacional.

Segundo a Microsoft (CONERY et al., 2009):

[1] "Política Nacional de Recursos Hídricos". Disponível em: ⟨https://www.ana.gov.br/gestao-da-agua/sistema-de-gerenciamento-de-recursos-hidricos⟩. Acessado: 25 de julho de 2020.

[2] " O que é o SINGREH". Disponível em: ⟨https://www.ana.gov.br/gestao-da-agua/sistema-de-gerenciamento-de-recursos-hidricos/o-que-e-o-singreh/o-que-e-singreh⟩. Acessado: 25 de julho de 2020.

> A arquitetura de aplicativos de *software* é o processo de definir uma solução estruturada que atenda a todos os requisitos técnicos e operacionais, enquanto otimiza atributos comuns de qualidade, como desempenho, segurança e capacidade de gerenciamento.

Assim, arquitetura de *software* é a estrutura organizacional de um sistema que o faça cumprir com os **requisitos funcionais e não funcionais** previstos para ele. Essa definição pode ser encontrada também em (SOMMERVILLE, 2011):

> Uma arquitetura de *software* é a descrição de como um sistema de *software* é organizado. Propriedades de um sistema como performance, segurança e disponibilidade são influenciados pela arquitetura utilizada.

Sommerville (2011) ainda divide os níveis de abstração em uma arquitetura de *software* em dois tipos:

- *Arquitetura em pequena escala.* Nesse nível de abstração, é considerada como sistemas individuais que são divididos em componentes;
- *Arquitetura em larga escala.* Que está preocupada com sistemas corporativos, que utilizam outros sistemas e componentes de outros programas para o funcionamento. Estes, normalmente são sistemas distribuídos, que fazem parte de mais de uma organização.

Bosch (2000) afirma que o projeto da arquitetura é importante, assim como a sua documentação, visto que a arquitetura final influencia diretamente nos requisitos não funcionais do sistema (como performance, robustez e manutenibilidade). Os requisitos funcionais, diz ainda, são definidos pelos componentes individuais, mas a forma como esses componentes se relacionam é que faz o sistema cumprir ou não os não funcionais.

Um modelo de arquitetura pode ser usado por dois motivos (SOMMERVILLE, 2011):

- *Para facilitar as discussões sobre a construção da arquitetura.* Uma visão de alto nível do sistema, que ajuda na comunicação com as partes interessadas (que não tem tanto conhecimento na área) e facilita o processo de atribuição de trabalho no desenvolvimento (por não apresentar tanto detalhe, mostrando somente os componentes chave);
- *Como forma de documentação.* Uma produção completa de baixo nível da arquitetura para documentação, mostrando a interface e as conexões de todos os componentes do sistema, facilitando o processo de atualização e manutenção.

Porém, a prática de produzir arquiteturas completas para a documentação do sistema não é uma prática recorrente, às vezes até devido ao alto preço para produção (BOSCH,

2000; SOMMERVILLE, 2011). Sommerville (2011) diz ainda que o método mais usado é o diagrama de blocos (que é uma representação de alto nível do sistema), caracterizado pela representação de uma caixa para cada componente e para seus subcomponentes (quando especificados), ligados por setas que mostram o fluxo de dados ou sinais de controle transmitidos entre esses componentes.

A concepção de uma arquitetura de sistema pode ser considerada como um conjunto de decisões a serem tomadas pelo arquiteto:

1. Existe alguma arquitetura genérica que possa ser usada como base para o sistema?

   Por mais que cada sistema tenha seu próprio objetivo e implementação, normalmente *softwares* do mesmo ramo têm a estrutura parecida (como *e-commerces*, por exemplo). Essas estruturas genéricas podem ajudar o arquiteto a validar os *designs* feitos, estudar os componentes que podem ser reutilizados, comparar aplicações do mesmo tipo e, com isso, diminuir o tempo de produção da própria arquitetura (SOMMERVILLE, 2011).

2. Como o sistema vai ser distribuído entre os processadores?

   Sistemas pequenos normalmente executam em computadores pessoais (um único processador), não havendo necessidade de projetar como funcionaria o programa sobre esse processador. Porém, em sistemas maiores (como os empresariais), a busca tem sido pela distribuição do sistema para vencer os requisitos não funcionais traçados (como performance) (SOMMERVILLE, 2011).

3. Quais padrões (estilos) de arquitetura podem ser usados?

   Padrões para projetos se tornaram uma prática muito utilizada em todos os níveis do projeto, por oferecer uma base sólida com uma estrutura básica e um padrão de implementação. Isto traz ao projeto velocidade na implementação e na manutenção, visto que, conhecendo o padrão, o desenvolvedor (independente de ter implementado ou não o componente) saberá onde procurar e como arrumar (GAMMA et al., 1995). Na área de arquitetura, há vários padrões (cliente-servidor, baseada em componente, orientado a domínio, em camadas, orientada a objetos, orientada a serviços, modelo-visão-controle, tubo e filtro, repositório, etc) que apresentam suas aplicações, vantagens e desvantagens (CONERY et al., 2009; SOMMERVILLE, 2011; GARLAN; SHAW, 1993).

4. Qual será a abordagem fundamental usada na estrutura do sistema?

   Após o levantamento dos requisitos do sistema, deve-se escolher um dos padrões de arquitetura para ser usado no projeto (pergunta anterior) (SOMMERVILLE, 2011).

5. Como os componentes estruturais do sistema serão divididos em subcomponentes?

Analisa-se como os componentes identificados na arquitetura serão divididos em subcomponentes para fazer o sistema cumprir com os requisitos funcionais, visto que estes são definidos como os subcomponentes se comportam (SOMMERVILLE, 2011).

6. Qual estratégia vai ser utilizada para controlar a operação dos componentes?

   O estilo (ou padrão) a ser escolhido para o projeto influencia na forma como os componentes controlam uns aos outros, logo, o arquiteto deverá escolher o padrão que faça o sistema agir como deveria (SOMMERVILLE, 2011; GARLAN; SHAW, 1993).

7. Qual organização de arquitetura é a melhor para entregar os requisitos não funcionais do *software*?

   Para tanto, os requisitos não funcionais do sistema devem ser estudados e decididos. Cada padrão tem vantagens e desvantagens, o que faz o sistema cumprir ou não com as decisões feitas. Por exemplo, um padrão em camadas separa o usuário dos dados protegidos (devido às camadas que o sistema foi estruturado), porém a performance do sistema pode ser atingida devido aos dados passarem por várias camadas até chegar ao usuário (SOMMERVILLE, 2011).

8. Como a arquitetura vai ser validada?

   A validação de uma arquitetura pode ser feita através da comparação dela com arquiteturas genéricas. Porém, a melhor forma de validá-la é em uso, verificando se o sistema cumpre com todos os requisitos decididos (funcionais ou não) (SOMMERVILLE, 2011).

9. Como a arquitetura deve ser documentada?

   Há cinco formas de apresentar a arquitetura de um programa (KRUCHTEN, 1995; HOFMEISTER; NORD; SONI, 2000):

   (a) *Visão Lógica.* É uma decomposição orientada a objetos. Mostra as abstrações chave do sistema como objetos ou classes.

   (b) *Visão de Processo.* Leva em consideração requisitos não-funcionais. Mostra a sincronização e concorrência dos componentes durante a execução do programa.

   (c) *Visão Física.* Usado quando o programa é executado de forma distribuída. Nessa visão são identificados os componentes onde o sistema age no *hardware* (como rede, processadores, processos e tarefas), como um mapa.

(d) *Visão de Desenvolvimento.* Foca na organização do *software* enquanto está em desenvolvimento. Mostra como o sistema é decomposto em componentes (de *software*), onde cada um é implementado por um desenvolvedor ou um pequeno time de desenvolvedores.

(e) *Visão Conceitual.* É uma visão abstrata dos componentes de alto nível, normalmente a visão mostrada aos *stakeholders* por não poluir o diagrama do sistema com muitos detalhes internos dos componentes.

2.4 Ferramentas Utilizadas

2.4.1 JAVA e Spring

Java é uma linguagem de programação que foi lançada em 1995 por Patrick Naughton, Mike Sheridan e James Gosling para ser uma linguagem rápida e que funcionasse em todos os dispositivos da época (BYOUS, 1998). Hoje, Java tem suporte para Orientação a objetos e, segundo o site da Computer Society (IEEE)[3], é uma das linguagens de programação mais utilizadas do mundo e uma das principais escolhas para se trabalhar no *backend* de aplicações.

Esta linguagem apresenta diversos *frameworks*, como Hibernate[4], Swing[5], ADF[6], DOJO Mobile[7] e Spring[8]. Esse último é um *framework* dividido em "projetos" para diversos tipos de aplicações (como o Spring Data – para acesso de dados em servidor; o Spring Android – para aplicações *mobile* que rodam no sistema operacional Android; e o Spring Security – para serviços de autorização e autenticação).

O chamado Spring Boot[9] é um projeto do Spring que facilita o início do desenvolvimento e a execução do sistema, devido às bibliotecas disponibilizadas. Ele ainda conta com um serviço de *RESTful Web Service*, para criação de APIs.

Esta linguagem e esse *framework* foram utilizados neste trabalho, visto que o Motor de Cálculos (ver seção 3.2.2) foi desenvolvido utilizando Java Orientado a objetos e, para ligar com a interface do usuário, está sendo utilizado Spring Boot (com o serviço *web* de REST, para o desenvolvimento da API).

2.4.2 Angular

Angular é uma plataforma de desenvolvimento e um *framework* para a criação de aplicativos de página única (SPAs – do inglês *Single-Page Applications*) que utiliza a

[3] Disponível em: ⟨https://www.computer.org/publications/tech-news/trends/programming-languages-you-should-learn-in-2020⟩

[4] ⟨http://hibernate.org⟩

[5] ⟨https://docs.oracle.com/javase/tutorial/uiswing/⟩

[6] ⟨https://www.oracle.com/database/technologies/developer-tools/adf/⟩

[7] ⟨http://dojotoolkit.org/⟩

[8] ⟨https://spring.io/⟩

[9] ⟨https://spring.io/projects/spring-boot⟩

linguagem TypeScript, além de HTML, CSS e JavaScript[10]. Além disso, o Angular trabalha com *componentes* utilizados para construir a página que o usuário irá ver. Estes componentes podem ser utilizados em qualquer página do sistema que importá-los.

SPAs são aplicações *web* onde todo conteúdo necessário (HTML, CSS e JavaScript) é transferido para o cliente de uma só vez. Enquanto o cliente interage com a página, o navegador gerencia o que aparece para o cliente, sem nunca atualizar a página. Caso seja necessário algum dado do servidor, a aplicação faz requisição via JSON, somente variando os dados requeridos pelo usuário (HOLMES, 2015).

Esta ferramenta utiliza o padrão de projeto chamado Modelo-Visão-Controle (MVC) que separa a apresentação (interface do usuário) do processamento da página e do sistema de dados. O *Modelo* é responsável pelo gerenciamento dos dados e pelas operações do sistema. A *Visão* é o componente que renderiza a interface do usuário. O *Controle* administra as interações do usuário com a página (cliques de mouse, por exemplo) e passa essa informação ao *modelo* (SOMMERVILLE, 2011).

Com esta tecnologia, a interface gráfica para os jogadores foi implementada (ver seção 3.3, pois é um *framework* sólido que funciona utilizando componentes, dando a possibilidade de reutilizar os componentes para montar as fichas dos personagens do jogo. Adicionalmente, o Angular consegue se alimentar de APIs, possibilitando o uso com o Spring Boot. Além disso, por ser uma SPA, os módulos necessários são carregados somente uma vez para o jogador, aumentando a performance da interface.

2.5 Trabalhos Relacionados

Nesta seção são apresentados os trabalhos que foram eleitos como relevantes para o desenvolvimento deste devido à natureza similar quanto às áreas que eles se encontram. Mesmo os Sistemas Multiagente (SMA) não fazendo parte das áreas envolvidas neste trabalho, indiretamente é importante a comunicação com esta tecnologia, para que, futuramente, seja implementado um módulo de inteligência artificial ao GorimWEB. Desta forma, trabalhos que fazem o uso de SMA foram incluídos aos trabalhos relacionados.

Em Farias et al. (2019) foi implementada uma Revisão Sistemática da Literatura (RSL), que fez o levantamento sistemático de trabalhos pertencentes às áreas de RPG, SMA e GRN. Do resultado da RSL, serão apresentados aqui os trabalhos que têm como produtos sistemas com RPGs de mesa utilizados junto aos SMAs (*Multi-Agent Systems* - MAS) computacionais (RPG + SMA) e sistemas com RPGs e SMAs trabalhando em conjunto de forma computacional (RPG ++ SMA). Os grupos de como os resultados foram analisados na RSL estão apresentados na Figura 1, sendo os grupos (c) e (d) relevantes ao trabalho aqui proposto.

O trabalho proposto por Gurung, Bousquet e Trébuil (2006) apresenta a modelagem

[10] ⟨https://angular.io/docs⟩

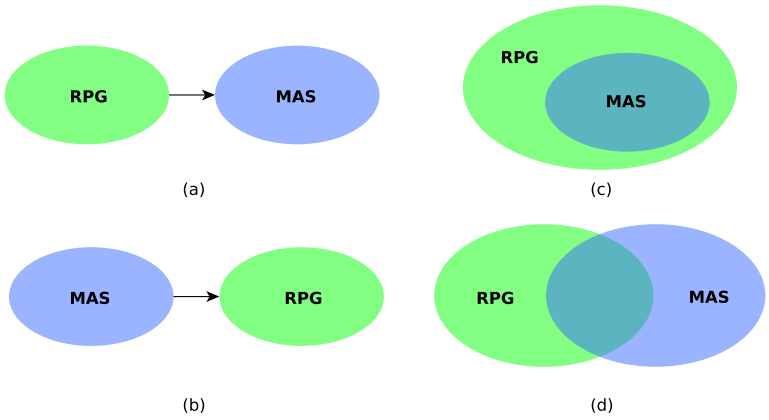


Figura 1: Grupos que separam os trabalhos com RPG e SMA (FARIAS et al., 2019).

e uso de um sistema para auxiliar na resolução de conflitos e na criação de um comitê responsável pela tomada de decisão e compartilhamento do uso da água na bacia hidrográfica em Lingmuteychu, no Butão. Há presente nessa Bacia sete vilas, as quais estão distribuídas na encosta de uma montanha com distâncias consideráveis uma da outra. As regras principais do abastecimento de água para plantações (antes do projeto) eram: a primeira vila (ou seja, a mais no alto) era a primeira a ser servida e as vilas mais altas têm direito a ter mais água. Tendo em vista o problema de comunicação entre as vilas e que a vila mais baixa na encosta tem acesso extremante limitado à água, foi proposto um RPG (inicialmente com as duas vilas mais altas e, após, com todas), utilizando a abordagem ComMod[11], onde os jogadores teriam que lidar com a situação que eles se encontravam de uma maneira mais reduzida, ou seja, teriam que fazer o plantio e o racionamento da água em um ambiente mais abstrato (uma reprodução enxuta da realidade). Foram feitos 3 modos de jogo: um com os tabuleiros que simulam as vilas com a distância física real entre eles, outro com os tabuleiros um ao lado do outro e, ainda, o terceiro com os tabuleiros perto e com os jogadores invertidos, ou seja, quem era de uma vila jogava no tabuleiro da outra vila. Além do RPG, foi utilizado um sistema de simulação baseado em agentes onde, após a inserção dos parâmetros encontrados no RPG (redes sociais, padrões de chuva e protocolos de troca) e a simulação de diferentes cenários (20 simulações de 36 cenários), foi constatado que a melhoria das vilas depende da definição dos protocolos de comunicação. Após as simulações com todas as vilas presentes, foi relatado que os jogadores tiveram interesse em relação ao segundo modo de jogo, ou seja, com os tabuleiros próximos, pois viram a necessidade e a importância da comunicação entre as vilas para o desenvolvimento de ambas. Além disso, foi criado o comitê da Bacia, com a presença de uma constituição que regra o compartilhamento de água entre as vilas e o tempo que cada

[11]Companion Modelling (BOUSQUET et al., 1999): modelo criado com o intuito de padronizar uma metodologia para a criação de sistemas/simulações a serem utilizadas como auxiliares nas tomadas de decisões de gestores. Gurung, Bousquet e Trébuil (2006)

vila tem a água em seu poder.

O trabalho de Barnaud et al. (2007), por sua vez, descreve o uso da metodologia ComMod para facilitar o processo de aprendizagem coletiva na comunidade Akha, em Mae Salaep (norte da Tailândia), em relação ao problema de erosão do solo. Este trabalho é a continuação do primeiro ciclo do ComMod apresentado em (TRÉBUIL et al., 2002), onde, ao final do ciclo, os participantes identificaram como solução promissora a expansão das plantas perenes (por exemplo, arroz), e pediram que o segundo ciclo fosse voltado para as consequências socioeconômicas na adoção desse tipo de semente. Após os pesquisadores voltarem com o novo modelo e os encontros terem sido feitos, teve-se como resultado a comparação de como os *stakeholders* diziam se comportar e como eles se comportavam realmente, possivelmente devido ao RPG simular de forma real as situações vividas pelos envolvidos. Além disso, os participantes relataram ainda que a forma apresentada da situação vivida por eles na época (utilizando ComMod) ajudou-os a pensar nas consequências antecipadamente e, ainda, que esse método ajudou-os a entender a situação e pontos de vista um do outro, produzindo uma comunicação quase democrática, o que eles não tinham na situação social e institucional. Percebeu-se, também, que no segundo ciclo do ComMod os participantes mostraram mais interesse e participação, devido ao fato de discutirem mais as estratégias e problemas entre eles que com os pesquisadores. Como trabalhos futuros, os pesquisadores relataram a intenção de criar um sistema onde outras vilas da região pudessem utilizar, espalhando mais o conhecimento e comunicação alcançados pela vila de Akha.

Adamatti, Sichman e Coelho (2009) descrevem em seu trabalho (de uma forma resumida em (ADAMATTI, 2009)) um projeto que faz o uso de uma ferramenta própria para fazer os jogadores determinarem a qualidade e quantidade de água no sistema de captação em um perímetro urbano. Foram feitos dois protótipos (JogoMan - Jogo dos Mananciais - e ViP-JogoMan - *Virtual Players* in JogoMan) utilizando o simulador Cormas[12], onde o primeiro protótipo era somente com jogadores reais e, para o segundo, viu-se a necessidade do preenchimento da quantidade de jogares quando não havia o suficiente para o início da simulação. Logo, para o ViP-JogoMan, foram definidos agentes com inteligência criada a partir da estratégia de jogadores reais. Para o projeto, foi utilizado a metodologia GMABS[13] desenvolvida pelos autores. Como resultados são apresentados que o primeiro modelo ajudou os jogadores a entender o domínio proposto. Além disso, eles aprovaram o modo de jogo (divertido) e, segundo os autores, por causa da possibilidade da interação entre os jogadores, o processo de negociação foi importante para a discussão de novas abordagens para a solução do problema na área. O ViP-JogoMan teve

[12] *COmmon-pool Resources and Multi-Agent Simulations*: framework de simulação baseado em agentes. ⟨http://cormas.cirad.fr/indexeng.htm⟩

[13] *Games and Multi-Agent-Based Simulation* modelo criado em (BARRETEAU; BOUSQUET; ATTONATY, 2001) e nomeada em (ADAMATTI; SICHMAN; COELHO, 2007). Criado para dar enfoque na descrição dinâmica das técnicas de integração entre RPG e MABS.

como resultado principal a forma como os jogadores interagiam sem saber que estavam jogando com agentes virtuais. Ambos os modelos ajudaram a validar o GMABS como uma boa metodologia no processo de negociação.

Em Campo et al. (2009) é apresentado um estudo de caso em três comunidades de Palawan, Filipinas, onde o objetivo foi criar um ambiente que pudesse facilitar a comunicação e negociação entre os diferentes *stakeholders* sobre seus interesses, consequências de estratégias e identificação de áreas de intervenção. Além da complexa rede de *stakeholders*, as vilas ainda eram regidas por regulações e políticas que não eram claras e, muitas vezes, geravam confusão e conflitos. A metodologia ComMod foi utilizada neste projeto. Como resultados, foi relatado como mais importante o fato de os participantes apreciarem em questão de valor e aplicabilidade as ferramentas apresentadas a eles. Viu-se também que o RPG ofereceu uma forma de todos os *stakeholders* participarem, fazendo-os imitarem suas ações, reações e decisões dentro de um ambiente igualitário, além de permitir o aumento suas experiências, testando suas estratégias num jogo realístico. Por fim, as simulações multiagente ajudaram a testar e desenvolver estratégias de gerenciamento.

O projeto descrito em Page et al. (2016) possui maior semelhança ao estado atual do projeto no qual este trabalho está inserido. Nele, é apresentada a ferramenta ReHab, que tem como objetivo o aprendizado experimental visando a produção e comunicação de conhecimento pela conservação e gestão dos recursos naturais. Foi utilizado o ComMod para o desenvolvimento do jogo de mesa que apresenta um cenário onde os jogadores tem que proteger os pássaros e captar a biomassa. O jogo foi reproduzido 45 vezes em 15 cidades diferentes, de forma não igualitária. Como resultados, os autores relataram que a comunicação melhora o rendimento dos jogadores, fazendo-os criarem estratégias racionais com seus entendimentos sobre as regras do jogo, mesmo que entendidas erroneamente. A semelhança achada neste projeto com esta dissertação é que o ReHab é um RPG de mesa, onde todas as movimentações dos jogadores são transferidas para serem computadas em um SMA, assim como o Gorim.

Em Page e Perrotton (2017) é apresentado o KILT, uma abordagem para utilizar-se no caso de explicitar o processo de modelagem com as partes envolvidas quando escolher o grau de complexidade do modelo de simulação multiagente. KILT é o acrônimo para *Keep It a Learning Tool*, e é comparada com outras duas abordagens chamadas KIDS (*Keep It Descriptive Stupid*) e KISS (*Keep It Simple, Stupid*), onde o projeto obtém um ambiente extremamente realístico e inteiramente abstrato, respectivamente. O KILT é definido por produzir um ambiente estilizado, ou seja, não tão realístico, mas, ainda, não tão abstrato, aumentando a utilidade do ambiente. Foi utilizada, então, a abordagem ComMod para a criação de um ambiente com um grupo de 11 moradores de Hwange National Park, Zimbábue, onde foram modeladas as interações entre as atividades agrícolas, práticas agropecuárias e vida silvestre. Em sua primeira versão, foi feito um tabuleiro onde não

foram inseridos fatores como perda das produções por clima e por elefantes (o que era comum na região). Apesar de terem achado o jogo prazeroso, todos os participantes o acharam não-realístico, fazendo o pedido de outra versão aumentando a realidade, mas deixando a jogabilidade. Foi, então, desenvolvida uma segunda versão mais completa do jogo, ainda em tabuleiro, porém, com a complexidade adicionada (tendo que mudar manualmente as cores do tabuleiro de acordo com o crescimento e perda das plantas, predação de gado por leões, níveis de água, etc), foi pedido uma versão computacional. A forma que acharam para a resolução deste problema foi a inserção de um projetor vertical no qual era mostrado o tabuleiro sobre a mesa, absorvendo a necessidade de mudanças manuais no tabuleiro. Como resultados, o jogo foi definido como útil, além de ter servido como uma oportunidade de pensar, aprender e abrir novas perspectivas. O jogo ainda serviu como validador da abordagem KILT, mostrando que um ambiente muito simples pode não mostrar relação com a realidade e um muito complexo pode distrair os jogadores do objetivo principal do jogo (que seria o de aprender a lidar com alguma situação).

Caltagirone et al. (2002) mostram em seu trabalho a proposta de uma arquitetura para um MMORPG. Foram definidos como requisitos: (i) segurança, (ii) manutenibilidade, (iii) escalabilidade, (iv) baixo tráfico de internet, (v) performance e (vi) balanceamento de carga. Foram propostas três arquiteturas: a geral do sistema (utilizando o padrão cliente servidor com quatro camadas – aplicação, chat, encriptador e rede), a do lado do servidor (em camadas e dividindo o banco de dados do Mundo entre vários servidores, para diminuir a carga de clientes em um único servidor) e a do lado do cliente (que utiliza um modo Modelo-Visão-Controle modificado para jogos – utilizando um componente de gráficos). Além disso, foram usados os padrões Broker e Blackboard para fazer o gerenciamento da distribuição dos servidores e das requisições dos usuários. Para melhorar a performance da aplicação, foi implementado um serviço de fila, onde os clientes (jogadores) que fazem a inscrição na fila (quando entram no jogo) são avisados pelo servidor se um outro usuário interferiu de alguma maneira no ambiente. A proposta desta abordagem para a melhora na performance, porém, se o módulo de gerenciamento do cliente que recebeu o aviso do servidor sobre a mudança no ambiente julgar que essa mudança não afeta na jogabilidade do personagem que está sendo gerenciado por ele (se está fora do alcance de visão, por exemplo), ele adia a importação desta mudança até julgar necessário.

Em Moskalev (2014) é apresentada uma arquitetura de software para um servidor de um jogo MMORPG. O jogo consistia em uma página com alguns cenários, onde o jogador poderia escolher um dos cenários e convidar os outros jogadores presentes nesse cenário para batalhar, e outra página como personagem e os itens que poderiam ser usados para equipá-lo. A batalha era feita por turnos, onde cada um dos jogadores escolhia uma (entre cinco) partes do corpo para atingir o seu adversário e escolhia onde (dentro das mesmas cinco opções) queria que o próprio personagem defendesse o ataque. Se o outro adversário escolhesse onde o jogador tinha escolhido para o personagem defender, ele não

sofria dano. O jogo ainda dava randomicamente chances de dar golpe crítico (o golpe vale o dobro de um golpe normal) ou de desvio automático, caso o jogador não defendesse o golpe do adversário. Para a definição da arquitetura do servidor foram levantados os seguintes requisitos: (i) mandar mensagens aos clientes de modo assíncrono; (ii) ser capaz de suportar algumas centenas de clientes simultaneamente; (iii) ser modular e extensível; (iv) registrar/logar jogadores; (iv) quebrar conexão caso o jogador fique um tempo sem jogar com a janela aberta; (v) não permitir o uso concorrente de uma conta em dois locais diferentes; (vi) dar a habilidade de ver características do próprio personagem e de seus equipamentos; (vii) ver o log da luta enquanto ela está sendo realizada; e, (viii) ver a lista de jogadores disponíveis em cada localidade. Foi escolhido o padrão em camadas para implementar a arquitetura, e esta foi implementada em JAVA utilizando os *frameworks Non-blocking I/O* (para protocolos de *socket* cliente-servidor), *Java Persistence Application Programming Interface* (JPA) (para fazer a persistência de dados no BD, pois diminui a complexidade quando grava e busca dados) e banco de dados em MySQL.

A Tabela 1 mostra o resumo dos trabalhos relacionados com RPG apresentados, destacando seus objetivos, abordagens e a integração entre RPG e SMA, e a Tabela 2 apresenta os trabalhos relacionados à arquitetura de *software*, comparando os objetivos e tipo de arquitetura dos trabalhos.

Tabela 1: Tabela comparativa entre os trabalhos relacionados com RPG.

Trabalho	Objetivo	Abordagem	Integração
(GURUNG; BOUSQUET; TRÉBUIL, 2006)	Criar um sistema para auxiliar na resolução de conflitos e na criação de um comitê responsável pela tomada de decisão e compartilhamento do uso da água na Bacia em questão.	ComMod	(não faz parte da RSL)
(BARNAUD et al., 2007)	Criação de um jogo/sistema que facilite o processo de aprendizagem coletiva em relação ao problema de erosão do solo.	ComMod	(não faz parte da RSL)
(ADAMATTI, 2009; ADAMATTI; SICHMAN; COELHO, 2009)	Implementação de ferramenta para conscientização do uso e qualidade da água.	GMABS	RPG ++ MAS
(CAMPO et al., 2009)	Desenvolvimento de ambiente que pudesse facilitar a comunicação e negociação entre os diferentes envolvidos sobre seus interesses, consequências de estratégias e identificação de áreas de intervenção.	ComMod	RPG + MAS
(PAGE et al., 2016)	Uso de ferramenta para o aprendizado experimental visando a produção e comunicação de conhecimento pela conservação e gestão dos recursos naturais.	ComMod	RPG ++ MAS
(PAGE; PERROTTON, 2017)	Apresentação de nova abordagem de modelagem de ambientes e criação de um sistema para a tomada de decisão sobre possíveis ocorrências nas plantações.	ComMod + KILT	RPG + MAS

Tabela 2: Tabela comparativa entre os trabalhos relacionados com arquitetura de *software*.

Trabalho	Objetivo	Tipo de arquitetura
(CALTAGIRONE et al., 2002)	Proposta de uma arquitetura para o servidor e cliente de um MMORPG.	MVC + Camadas
(MOSKALEV, 2014)	Aplicação e proposta de uma arquitetura voltada para o servidor de um MMORPG.	Camadas

3 GORIMWEB

Neste capítulo é apresentado todo o desenvolvimento do GorimWEB. Primeiramente é apresentada a metodologia adotada para o trabalho, seguido da apresentação do Gorim em formato de jogo de mesa. Após, o jogo em sua versão web é apresentado em detalhes.

3.1 Metodologia

A metodologia adotada neste trabalho é mostrado na Figura 2.

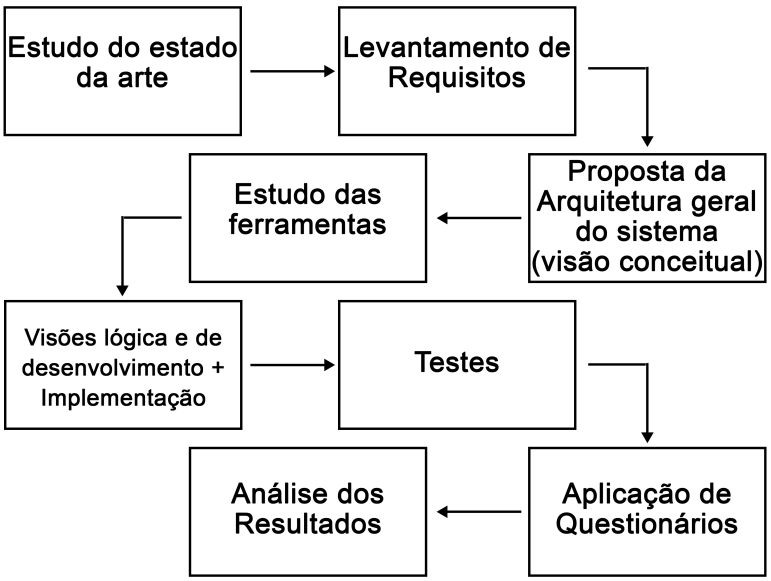


Figura 2: Fluxograma da metodologia de desenvolvimento do trabalho.

Inicialmente, foi feito o estudo das áreas que envolvem o trabalho: RPG, GRH e Arquitetura de Software, assim como trabalhos relacionados. Após, foi feito o levantamento de requisitos, levando em consideração o RPG de mesa chamado Gorim, e a proposta de arquitetura, utilizando a visão conceitual. Para a validação da arquitetura proposta, foram estudadas as ferramentas para a implementação do GorimWEB. Com a implementação, foram produzidos mais diagramas (nas visões conceitual, de desenvolvimento e lógica) para a documentação do projeto. Após terminada toda a implementação, foram realizadas

partidas com a interface criada. Nesse ponto, foram realizadas partidas com dois grupos de pessoas: os que já jogaram o Gorim na forma de jogo de mesa, e os que nunca jogaram. Para cada grupo, foram aplicados questionários, permitindo a validação das funcionalidades e estrutura do jogo (ou seja, dos requisitos definidos).

A seguir serão apresentados os requisitos levantados para a arquitetura do GorimWEB (Seção 3.1.1), a arquitetura (Seção 3.1.2) em si e, após, o jogo base: Gorim (Seção 3.2).

3.1.1 Requisitos Funcionais e Não Funcionais

Os requisitos levantados para o desenvolvimento de um projeto podem ser divididos em dois tipos: funcionais e não funcionais. Os funcionais se definem por serem as funcionalidades das aplicações e em como as aplicações devem se comportar em diferentes situações. Já os não funcionais são limitações encontradas nas funções ou serviços disponibilizados pelo sistema, como performance, disponibilidade ou segurança (SOMMERVILLE, 2011).

Dentro do escopo do jogo Gorim e pensando no desenvolvimento do jogo GorimWEB, foram identificados os seguintes **requisitos funcionais**:

1. Apresentar a jogabilidade encontrada no Jogo de Mesa, que pode ser dividido em:
   (a) Acessar informações relacionadas ao próprio personagem, como ações disponíveis, nome ou recursos;
   (b) Acessar as informações públicas de outros personagens, como nome ou preço de produtos;
   (c) Comunicação entre os jogadores.
2. Guardar e exportar dados (como logs e chat);
3. Dar suporte a quantidades variáveis de jogadores (mínimo = 6);
4. Dar suporte a Sistemas Multiagente.

E como **requisitos não funcionais**, identificaram-se:

1. Desempenho;
2. Informações bem estruturadas na interface;
3. Disponibilidade via *web*.

3.1.2 Arquitetura

Levando em consideração os requisitos levantados, a arquitetura proposta é apresentada na Figura 3, retratada em Visão Conceitual. Neste diagrama vê-se que o *backend* se comunica com os outros dois módulos por meio de uma API Rest, dando acesso ao banco de dados e ao processamento do jogo. Os jogadores, tanto virtuais quanto reais, enviam e recebem requisições JSON para submeter as ações no motor e para receber a atualização do ambiente. Além disso, tem acesso ao Chat, onde podem negociar e decidirem jogadas.

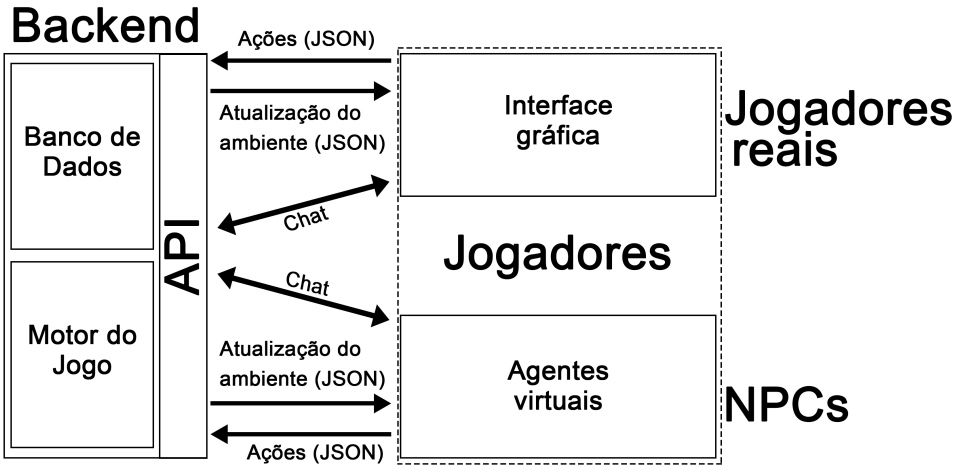


Figura 3: Diagrama em visão conceitual da arquitetura proposta.

Sommerville (2011) sugere a resposta de nove perguntas para fazer a modelagem de uma arquitetura de software (ver Seção 2.3). A seguir foram respondidas estas perguntas com base neste trabalho.

1. Existe alguma arquitetura genérica que possa ser usada como base para o sistema?

   A proposta de arquitetura será realizada para RPGs de mesa (que serão implementados computacionalmente) que utilizam cartões preenchíveis para a realização das ações. Não foi encontrada uma arquitetura genérica que encaixasse os requisitos funcionais e não-funcionais levantados para o sistema.

2. Como o sistema vai ser distribuído entre os processadores?

   Será um sistema *web* em pequena escala, logo será utilizado somente o processador do servidor para a execução e a página dos navegadores dos jogadores para carregar as páginas (modelo cliente servidor tradicional).

3. Quais padrões (estilos) de arquitetura podem ser usados?

   Nesse trabalho poderiam ser utilizados:

   - MVC
   - Baseado em camadas

- Baseado em componentes
- Cliente-servidor

4. Qual será a abordagem fundamental usada na estrutura do sistema?

   Foram utilizados os padrões MVC para a interface gráfica (*frontend*) do usuário e Baseado em Camadas para o *backend*. Para a comunicação entre os dois, foi utilizado o padrão cliente-servidor.

5. Como os componentes estruturais do sistema serão divididos em subcomponentes?

   No caso do componente "Jogadores reais" (mostrado na Figura 3), será dividido em componentes para página *web*, visto que cada jogador vai assumir um papel e cada um precisa ter informações do próprio papel. Além de utilizar o padrão MVC, o Angular é um *framework* que utiliza o sistema de *componentes*, montando a página com os *componentes* chamados. A quantidade variável de jogadores será feita por meio da inserção desta quantidade por parte do Mestre do jogo.

   No caso dos outros requisitos funcionais, estes serão resolvidos no componente *backend* (também visto na Figura 3), onde a possibilidade de uso de SMA será feita com a API (o módulo "NPC" da Figura 3 acessa as funções do jogo do mesmo modo que o módulo "Jogadores reais") e o requisito de exportar arquivos será feito com o componente do Motor de Cálculos (na Figura 3) em conjunto com a API.

6. Qual estratégia vai ser utilizada para controlar a operação dos componentes?

   A operação dos componentes vai ser feita via jogadores/Mestre, ou seja, os jogadores atuam de forma síncrona durante o jogo e avisam o Mestre que terminaram, assim como é realizado no Gorim. Para isso, os comandos partirão do componente "Jogadores reais" (Figura 3) e serão transmitidos ao componente *backend*, por meio da API por seus serviços, até ser processada pelo Motor do Jogo.

7. Qual organização de arquitetura é a melhor para entregar os requisitos não funcionais do *software*?

   Apesar da organização em Camadas ser menos sugerida quando se escolhe performance, foi necessária a aplicação de camadas no *backend* da aplicação para permitir a execução de mais de um jogo ao mesmo tempo.

8. Como a arquitetura vai ser validada?

   A arquitetura foi validada com o uso do GorimWEB, onde foi implementada a interface gráfica e aplicados questionários aos jogadores (pré e pós jogo) para validar os requisitos funcionais e não funcionais do sistema.

9. Como a arquitetura deve ser documentada?

   A arquitetura foi documentada inteiramente. Foram feitas as visões de processo, de desenvolvimento e conceitual dos componentes apresentados na arquitetura geral do sistema.

3.2 Gorim

Gorim é o jogo oriundo do projeto da ANA (edital nº 16/2017), que tem como objetivo a criação de uma ferramenta de apoio à decisão para o Comitê de Gerenciamento Local. Para tanto, foi desenvolvido um RPG de mesa, onde os papéis jogáveis são as profissões de alguns membros do Comitê (Empresários, Agricultores, Fiscais Ambientais, ONGs, Prefeitos e Vereadores).

O Gorim precisou de um motor de cálculos (chamado a partir daqui de "Motor do Jogo", devido à quantidade de cálculos (percentuais, vendas, produtividade, etc) e sua complexidade, o que demandaria um tempo excessivo para finalização das jogadas, se realizadas manualmente.

3.2.1 Estrutura do Jogo

O jogo começa ao serem sorteados os papéis entre os jogadores. Existem cinco classes jogáveis de personagens: Empresário, Agricultor, Fiscal Ambiental, Prefeito e Vereador. Estas estão distribuídas entre as duas cidades do jogo: Atlantis e Cidadela. Cada *rodada* do jogo é dividida em duas *etapas*, onde na primeira etapa jogam os Empresários e os Agricultores e na segunda etapa jogam os Fiscais, Prefeitos e Vereadores.

Os Empresários podem ser de quatro tipos: Semente, Fertilizante, Máquinas ou Agrotóxico. Estes são responsáveis por negociar o preço de seus produtos com os Agricultores. O Empresário de Semente tem a opção de vender sementes de Hortaliça, Arroz ou Soja; o de Fertilizante e o de Agrotóxico tem três tipos de seus produtos: Comum, Premium e Super Premium; e o Empresário de Máquinas pode alugar os Pacotes de Máquina 1, 2 ou 3 e, ainda, alugar um Pulverizador. Todas as vendas geram uma *poluição* específica do produto (para a simulação da fabricação deste) e todo o preço da venda retorna ao Empresário como *produtividade*, que será a quantidade a mais de "Dinheiros" (moeda do jogo) disponível para o Empresário na próxima rodada. O Empresário, ainda, pode vender os produtos por três preços: Baixo, Normal e Alto, podendo escolher valores para cada cliente que vende.

O Agricultor tem em seu poder seis *parcelas* de terra, onde ele escolhe plantar em todas ou em algumas na rodada. Em cada parcela é possível plantar um tipo de semente, um tipo de fertilizante e um tipo de agrotóxico ou um pacote de máquinas, ou seja, não se pode utilizar agrotóxico e máquina na mesma parcela, com exceção do Pulverizador, que quando colocado na parcela, diminui a poluição daquela parcela pela metade. Com cada

combinação, o Agricultor gera uma *produtividade* e uma *poluição* diferente. Cada parcela pode ainda ter o Selo Verde. Este selo é dado pelo Fiscal Ambiental da cidade quando o Agricultor não utiliza agrotóxicos na parcela em questão. Ter o Selo Verde significa 5% de desconto no imposto sobre a produtividade da parcela.

Essas duas classes, ao terminarem a primeira etapa da rodada, devem pagar os *Impostos* à sua Prefeitura. Para tanto, os jogadores são divididos no início do jogo entre as cidades, onde os Empresários de Semente e de Máquinas ficam em Atlantis e os Empresários de Fertilizante e Agrotóxico pertencem à Cidadela. Isso é feito para equilibrar os impostos recebidos por cada cidade. Além disso, os papéis da segunda etapa são jogados pelos mesmos jogadores da primeira etapa, sendo eleitos a cada duas rodadas.

Os Fiscais Ambientais são responsáveis por analisar as jogadas dos atores da primeira etapa pertencentes à sua cidade. Aos que exageraram na poluição, o Fiscal pode aplicar Multa Leve, Média, Pesada ou nenhuma, dependendo da situação e do acordo que pode ser feito com o outro jogador. Além disso, é encarregado de dar ou tirar o Selo Verde das parcelas dos Agricultores. Todo dinheiro recebido por multas é direcionado à Prefeitura local.

Os Prefeitos devem, também, analisar as jogadas da sua cidade, mas com o intuito de controlar a poluição mundial. Este ator pode utilizar três diferentes Ações Ambientais: Tratamento de Água, Lixo e Esgoto, onde cada um tem um preço e um impacto na poluição mundial. Além disso, o Prefeito tem o poder de aumentar ou diminuir o preço dos impostos, divididos em três classes, que dão taxas de preço diferente para faixas de produtividade. Estes devem ainda ouvir seus Vereadores quanto à ideias de gestão.

O Vereador de cada cidade deve ajudar seu Prefeito com a gestão da cidade, analisando os impactos das ações tomadas e do dinheiro investido, o que inclui o convencimento do Prefeito quanto às suas ideias. Além disso, devem escutar os moradores da cidade e repassar as informações relevantes ao Prefeito.

O jogo de mesa é jogado a partir de 6 (seis) pessoas, sendo que quatro dessas são Empresários e o restante Agricultores, divididos entre as duas cidades, ficando, pelo menos, quatro jogadores para cada uma. Logo, a cada duas rodadas, três jogadores em cada cidade são escolhidos para os cargos elegíveis (Fiscal, Prefeito e Vereador), jogando nas duas etapas da rodada, enquanto os outros jogadores (se o número de jogadores for maior que 6) jogam somente a primeira etapa e influenciam na segunda indagando seus representantes. Além disso, todos os papéis são permitidos transferir “Dinheiros” para todos os outros.

O jogo termina ao ser atingido 100% de Poluição Mundial, visto que, à medida que a Poluição vai subindo, a Produtividade dos jogadores diminui (para simular a infertilidade do solo e impureza da água e do ar para plantação) e as Ações Ambientais dos Prefeitos aumentam o preço (para simular que quanto mais poluição quer tirar, mais caro é).

Todas as ações jogáveis dos jogadores (não inclui negociação) são marcadas em uma

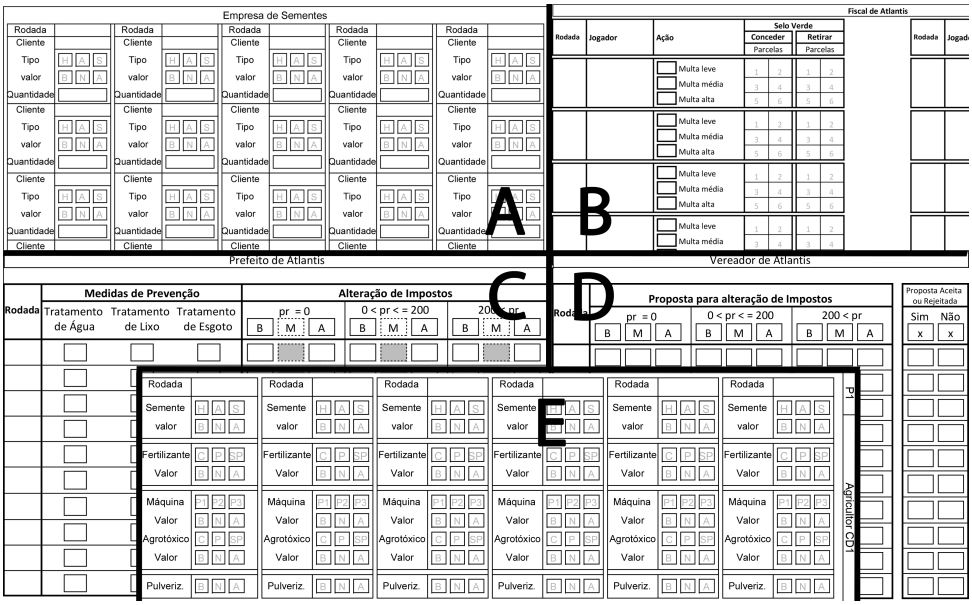


Figura 4: Fichas preenchíveis pelos jogadores, onde: A – Empresário, B – Fiscal Ambiental, C – Prefeito, D – Vereador e E – Agricultor (este recebe seis fichas, um representando cada Parcela).

ficha preenchível para ser inserida no Motor do Jogo no final de cada etapa, mostradas na Figura 4. Além disso, todos os jogadores recebem tabelas com valores relevantes para o papel, como preço de Produtos, custo das Ações Ambientais, poluição e produtividade causada por cada combinação de Produtos, etc (Anexo A).

3.2.2 Motor do Jogo

Devido à complexidade encontrada quanto à quantidade de cálculos a serem feitos no final de cada etapa do jogo e para a então futura implementação do GorimWEB, criou-se um motor de cálculos que tem como entrada as jogadas de cada papel e, como saída, os resumos para os papéis da próxima etapa, onde se encontram as informações relevantes às suas jogadas e atualizações do ambiente (como Poluição Mundial – ver Figura 5). Além dos resumos de cada personagem, o Motor ainda cria dois arquivos de *Log*, onde o primeiro guarda todas as jogadas em um aquivo do tipo CSV (*Comma-Separarted Values*) e no outro estão as mesmas informações, mas mais detalhada em forma de texto. Esse Motor foi feito utilizando Java Orientado a objetos e tem a estrutura ilustrada no Diagrama de Classes da Figura 6.

A Figura 6 mostra que todas as classes referentes aos personagens são filhas da Classe `Pessoa`. Essa classe possui os atributos e métodos básicos de todos os personagens: `nome`, `saldo`, `cidade` e `negociarCapital`, que seria a transferência de Dinheiros entre os papéis. A classe pode ser melhor observada na Figura 7.

A Classe `Empresario` (vide Figura 8) foi implementada como um agrupamento de `Produto`, além de ter em específico o `setor` de Empresário que pode assu-

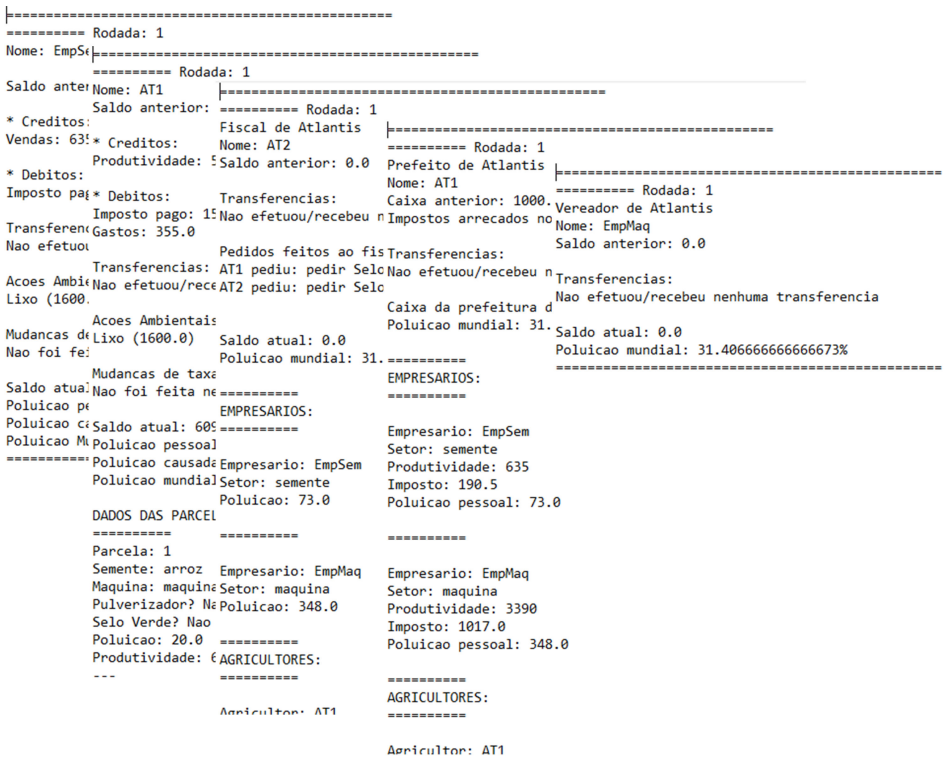


Figura 5: Aquivos Resumo recebido pelos jogadores ao final de cada rodada.

mir, `produtividade` e `poluicao` gerada e `imposto` e `multa` pagos na rodada atual. A Classe `Produto` (também na Figura 8) guarda as informações dos produtos que, na execução do programa, entram, cada um, como um Objeto (com `tipo`, `custo` e `poluicao`). Esta classe se liga à Classe `Agricultor` por causa da comunicação entre as classes quanto à venda/aluguel de produtos, e às Classes `FiscalAmbiental` e `Prefeito` devido ao pagamento de multas e impostos. Os métodos específicos são: `venderAlugar` (para vender o Produto ao Agricultor), `pagarMulta`, `pagarImposto` e `iniciaRodada` (que zera os atributos `poluicao`, `produtividade`, `imposto` e `multa` para a próxima rodada).

A Classe `Agricultor` contém uma agregação de `Parcela`, `qntdParcelas` (quantidade de Parcelas), `produtividade` e `poluicao` gerados na rodada, o `imposto` e a `multa` pagos e os `gastos` da rodada, ou seja, a quantidade de Dinheiros que foi gasto com Produtos. O método `plantar` percorre as Parcelas do Agricultor e calcula a produtividade e a poluição das combinações feitas. `setProdutoParcela` é usado quando uma compra é feita, sendo chamado pelo método `venderAlugar` da `Empresario`. O método `setSeloVerde` é chamado pela Classe `FiscalAmbiental` colocando ou tirando o Selo Verde das Parcelas. Os métodos `pagaMulta`, `pagaImposto` e `iniciaRodada` fazem o mesmo que para a `Empresario`, porém, tem-se o método `finalizarSaldo`, que, diferentemente da Classe `Empresario` (que tem o valor do produto diretamente no `saldo` quando faz a

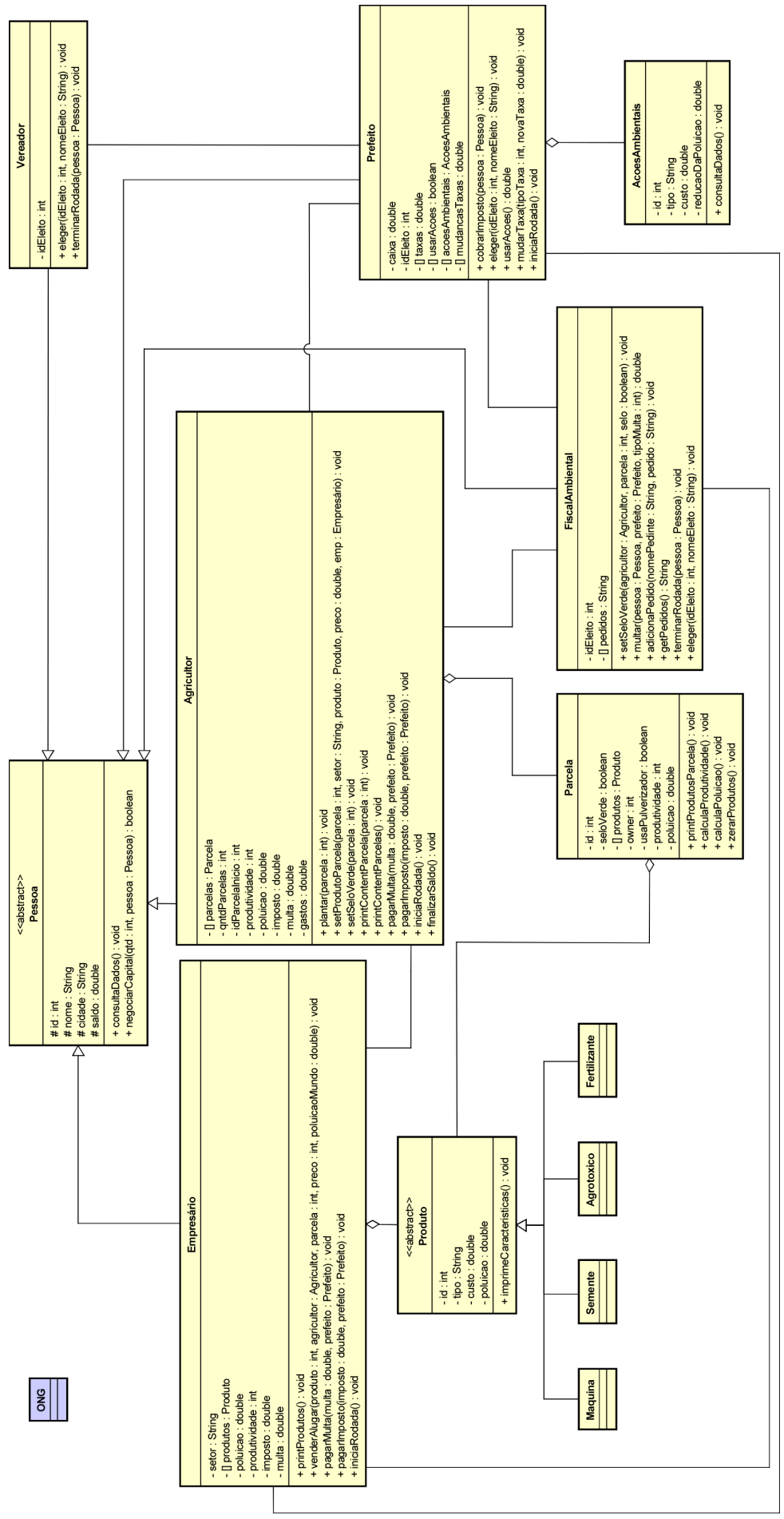


Figura 6: Diagrama de Classes do Motor do Jogo.

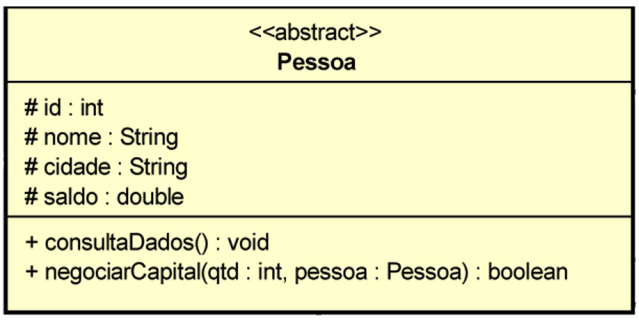


Figura 7: Diagrama de Classes – Classe `Pessoa`.

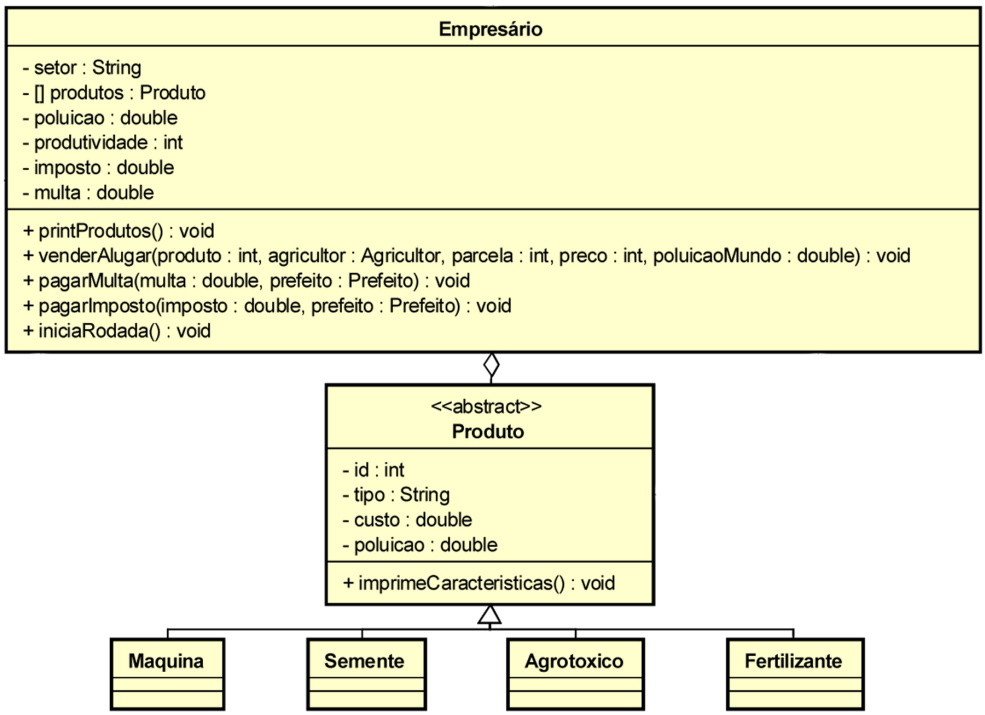


Figura 8: Diagrama de Classes – Classes `Empresario`, `Produto`, `Maquina`, `Semente`, `Agrotoxico` e `Fertilizante`.

venda devido ao método `negociarCapital`), ao terminar a rodada, o valor presente no atributo `produtividade` precisa ser somado ao do `saldo` do Agricultor. A relação presente entre essa classe, a `FiscalAmbiental` e a `Prefeito` existe para mostrar, assim como a `Empresario`, o pagamento de multas e impostos. A classe `Agricultor` também pode ser vista na Figura 9.

A Classe `Parcela` (também na Figura 9) tem, assim como a `Empresario` uma agregação de `Produto`, porém, no caso desse *array* só é possível a inserção de um objeto da Classe `Semente`, um da `Fertilizante` e um da `Máquina` ou do `Agrotóxico`, por conta das regras do jogo. No caso de ser usado uma Máquina ou Agrotóxico em conjunto com o Pulverizador, é ativada a *flag* `usaPulverizador`. Além disso, esta classe guarda o *id* do dono (`owner`), a `produtividade` e `poluicao` causada na Parcela du-

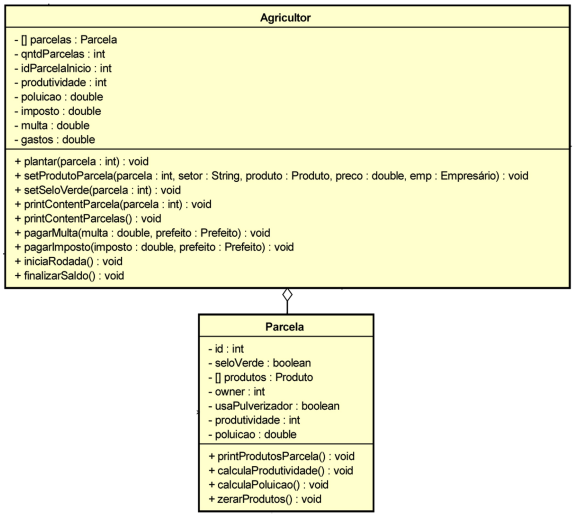


Figura 9: Diagrama de Classes – Classes `Agricultor` e `Parcela`.

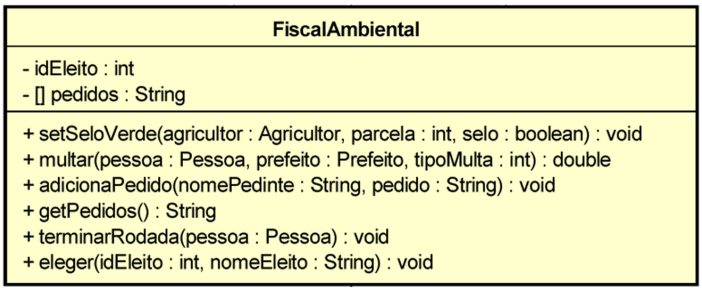


Figura 10: Diagrama de Classes – Classe `FiscalAmbiental`.

rante a rodada e tem a *flag* `seloVerde` que é editada pelo Fiscal Ambiental. O método `zerarProdutos` é chamado toda vez que o método `iniciaRodada` da Classe `Agricultor` é chamado, nela é retirado os produtos utilizados na última rodada (inclusive o Pulverizador). Os métodos `calculaProdutividade` e `calculaPoluicao` calculam a produtividade e poluição da Parcela quando o método `plantar` da `Agricultor` é chamado.

A `FiscalAmbiental` (ver Figura 10) tem um *array* de *pedidos*, que é quando o Agricultor pede Selo Verde para alguma Parcela (utilizando o método `adicionaPedido` da `FiscalAmbiental`). Além disso, tem o atributo que guarda o id do Eleito (`idEleito`) durante o mandato. Como métodos, apresenta o `setSeloVerde` (chamado para dar ou tirar o Selo Verde da Parcela do Agricultor), `multar` (para multar um Empresário ou um Agricultor), `adicionaPedido` (adiciona o pedido do Agricultor no *array* `pedidos`), `eleger` (troca os atributos `nome` e `idEleito` para os valores do novo eleito) e `terminaRodada` (limpa a lista de pedidos). Apresenta ligações com a Classe `Prefeito` para mostrar que a multa cobrada aos papéis da primeira etapa são transferidos ao `caixa`.

A Classe `Prefeito` (Figura 11) tem o atributo `caixa` que representa a quantidade

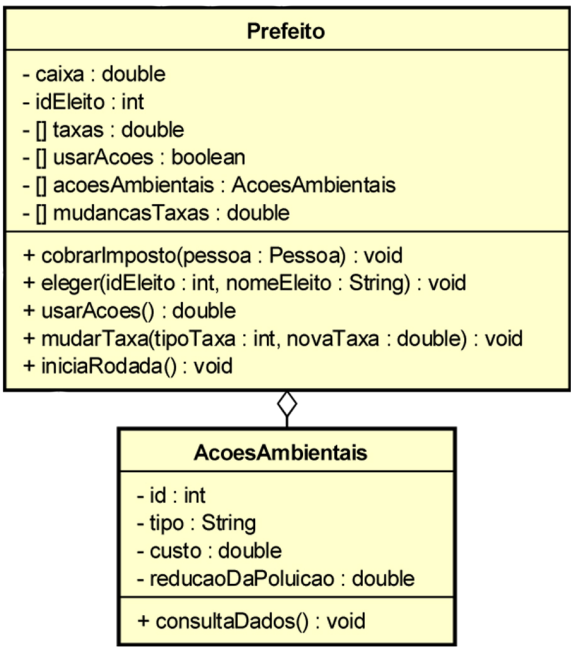


Figura 11: Diagrama de Classes – Classes `Prefeito` e `AcaoAmbiental`.

de Dinheiros disponível para o uso do Prefeito durante a segunda etapa. É caracterizado também por ser uma agregação de `AcoesAmbientais` e ter um *array* com os valores das taxas a serem cobrados de imposto. Por se tratar de um RPG, as ações são computadas no final da etapa, logo tem-se os atributos `usarAcoes` e `mudancasTaxas`, que guardam até o final da etapa as Ações Ambientais a serem implementadas e as taxas a serem mudadas, se houver alguma. Os métodos dessa classe são: `cobrarImposto` (que é chamado no início da segunda etapa do jogo para todos os papéis da primeira etapa), `eleger`, `usarAcoes` (aplica as Ações Ambientais escolhidas na rodada – presentes no *array* de mesmo nome – e retorna a porcentagem total a ser diminuída da poluição mundial) e `iniciaRodada` (que limpa os *arrays* `usarAcoes` e `mudancasTaxas`).

A Classe `AcoesAmbientais` (Figura 12) contém os atributos `tipo` (pode ser Água, Ar ou Esgoto), `custo` (quantidade de Dinheiros necessária para a implementação da Ação na cidade) e `reducaoDaPoluicao` (que informa a porcentagem a ser diminuída da poluição mundial).

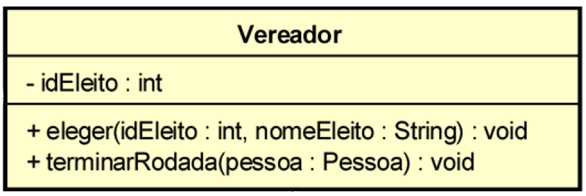


Figura 12: Diagrama de Classes – Classe `Vereador`.

Por último, `Vereador` (Figura 12) contém somente o atributo específico `idEleito` e os métodos `eleger` (que funciona da mesma forma que o das Classes `Prefeito` e `FiscalAmbiental`) e `terminarRodada` (que limpa algumas estruturas internas do objeto para iniciar a próxima etapa). Esta classe é pequena pois, durante o jogo, é um personagem de apoio ao Prefeito, logo, tem mais ações de interação entre jogadores que essencialmente ações computacionais.

Pode-se observar também a Classe `ONG`. Inicialmente, havia um personagem ONG para ajudar (ou atrapalhar) os Ficais Ambientais quanto a fiscalizar os Agricultores e Empresários. Porém, na prática, não haviam movimentos específicos (somente avisar os Fiscais ou Prefeitos sobre mau uso de terras e poluição em excesso), sendo pensado como um papel de apoio computacional na forma de um NPC após a implementação da interface gráfica do jogo (ainda não realizado nesta versão do jogo). Este NPC dará dicas aos jogadores de como plantar (no caso dos Agricultores), de quem fiscalizar (no caso dos Fiscais), etc. Atualmente, o Mestre do jogo realiza este papel.

Mesmo sem interface gráfica, o motor de cálculos precisava ser executado, logo criou-se a Classe `Mundo`, onde contém o *main* do código. Esta classe inicia todas as outras classes (menos a `ONG`) e um menu de terminal de comando foi montado para a inserção das jogadas escolhidas pelos jogadores. Essa classe gera, ainda, três tipos de arquivos de saída: os Arquivos de Resumo de cada jogador (com as atualizações do ambiente para jogarem a próxima rodada), os Arquivos de *Log* (um em formato CSV e um em formato TXT) e o Arquivo de *Save* (que guarda todas as inserções – números – digitados no menu para o caso de precisar refazer todas as jogadas performadas devido à quedas de luz, quebra no programa ou continuação da partida em outro momento).

Por se tratar de um RPG, tem-se a presença de um Mestre durante todo o jogo, informando o término de cada etapa, auxiliando nas dúvidas e distribuindo os resumos da etapa para os jogadores necessários, bem como o papel de ONG.

3.2.3 Testes com o Gorim

O Gorim teve diversas partidas executadas pelos membros do projeto (no qual foi modelado) em conjunto com alunos e professores que não fazem parte do projeto. Estas partidas tiveram como principal objetivo o teste e refinamento do jogo, assim como o refinamento do Motor do Jogo. A Figura 13 mostra os jogadores interagindo.

Para facilitar a comunicação entre os jogadores, foram criados broches, onde ficam identificados os papéis de cada jogador. Os que tem um papel de eleito na segunda etapa do jogo, utilizam dois broches (Figura 14).

Porém, com o estado pandêmico causado pelo novo coronavírus (COVID-19), as partidas presenciais não foram mais possíveis. Por a interface gráfica ser o produto final deste trabalho, as partidas passaram a ser realizadas utilizando os *softwares* Discord[1] (programa

[1] ⟨https://discord.com/inspiration⟩



Figura 13: Alunos e professores participando de uma partida do Gorim.

de chat e chamadas de voz/vídeo, tendo possibilidade de criar um servidor virtual com salas para bate-papos separados), Google Spreadsheets[2] (*software* online da Google para compartilhamento e edição simultânea de tabelas – utilizado para o preenchimento dos cartões de ações) e o Gmail[3] (utilizado para o envio dos Arquivos de Resumo dos personagens no final de cada etapa).

3.3 Implementação do GorimWEB

O desenvolvimento do GorimWEB iniciou pelo *backend*, visto que: (1) o Motor do Jogo estava finalizado e faz parte desta estrutura do projeto, e (2) precisaria da API pronta para testar e facilitar o desenvolvimento dos componentes a serem desenvolvidos nas janelas dos jogadores.

A Figura 15 mostra a arquitetura em camadas proposta para o *backend* da aplicação. Nela identificam-se os subcomponentes a serem utilizados nesse componente da arquitetura geral em visão conceitual. O subcomponente API recebe as requisições via JSON dos jogadores. Cada requisição é recebida em um formato específico, definido no subcomponente `modelos`. A API converte as requisições JSON nesses formatos de formulário e transfere para o serviço chamado `MundoService`. Ao se iniciar o jogo, o

[2]https://docs.google.com/spreadsheets/u/0/
[3]https://mail.google.com/



Figura 14: Broches criados para a identificação dos personagens durante as partidas presenciais.

`MundoService` cria os objetos a serem usados no jogo (diretamente com o Motor do Jogo). À medida que as jogadas dos jogadores chegam no `MundoService`, esse serviço chama o método necessário dentro de cada classe do Motor do Jogo. Este sempre salva todas as ações dentro dos logs (no subcomponente BD – *Banco de Dados*) e, ao final de cada etapa, cria os Arquivos Resumo, que ficam dentro do componente `data`. Cada um desses arquivos é salvo em JSON e esses JSONs são enviados aos jogadores ao serem requisitados. Além dos Arquivos Resumo, o componente `data` tem em arquivos características dos jogos criados (como quantidade de jogadores) separados por id do Mundo e o id do último Mundo criado.

O *backend* foi implementado utilizando o JAVA Spring Boot. O Motor do Jogo representado na Figura 15 é exatamente o mesmo utilizado nas partidas do Gorim.

Além disso, foi implementado o *web socket*. Essa funcionalidade permite o uso do chat pela interface, que tem todas as mensagens salvas no BD, e envios de sugestões, orçamentos e respostas a esses pelos jogadores. Na modalidade de mesa, os Empresários faziam os orçamentos e tinham a confirmação de preço por parte do Agricultor por meio da conversa entre os jogadores, e os Prefeitos tinham as sugestões dadas pelos Vereadores e, também davam a resposta e discutiam sobre a proposta verbalmente. Por esse

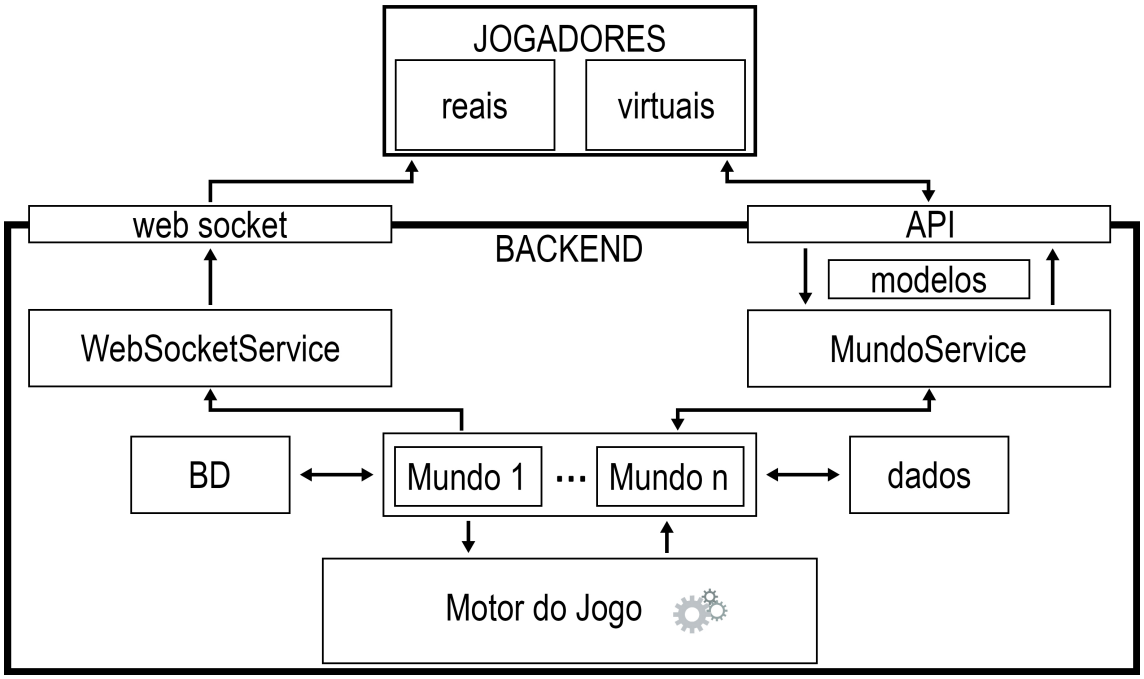


Figura 15: Arquitetura em visão conceitual do componente *backend* da Figura 3.

motivo que o *web socket* foi utilizado ao GorimWEB: mesmo que os movimentos reais dos jogadores só sejam computados ao final da etapa, essa troca de informação (como a confirmação do preço de venda) precisava ser feita durante a etapa. Desta forma, o GorimWEB "guarda" as informações destas negociações para futuras análises (utilizando, por exemplo, *machine learning*), o que não acontecia no Gorim, onde estas informações eram "perdidas" durante os jogos. O *web socket*, porém, faz somente o envio de dados, sendo o recebimento de orçamentos, mensagens do chat e sugestões feita via API Rest.

A implementação da interface gráfica, por ser feita com o uso do Angular, foi completamente dividida em *módulos*, sendo cada janela possível da interface um *módulo*, e cada *módulo* separado em: classe de processamento (a partir daqui referida como *componente*, *serviço*, *modelo* e interface (arquivos HTML e SCSS ou CSS) e *subcomponentes* (subcomponentes). Estes são subestruturas que foram separadas da classe principal do *módulo* por ter algum comportamento próprio ou, ainda, por terem chamadas diferentes ao *backend* e, cada um deles tem seu próprio *serviço*, interface, *componente* e, às vezes, *subcomponentes* e *modelos*.

Os *serviços* são utilizados para a aquisição e envio de dados do/para o *backend*, tendo como origem e destino desses dados um *componente*. Outro uso dos *serviços* é o de transferir dados de um *componente* para outro, deixando um canal aberto para essa troca interna, como quando um Fiscal Ambiental marca que multou um Empresário. Internamente, o *componente* `fine` envia essa "multa" para o *componente* principal do *módulo* `supervisor`, onde fica armazenado até o jogador cancelar essa multa ou o término da etapa, que, fazendo o uso do *serviço* principal do *módulo*, essa lista (e a lista de

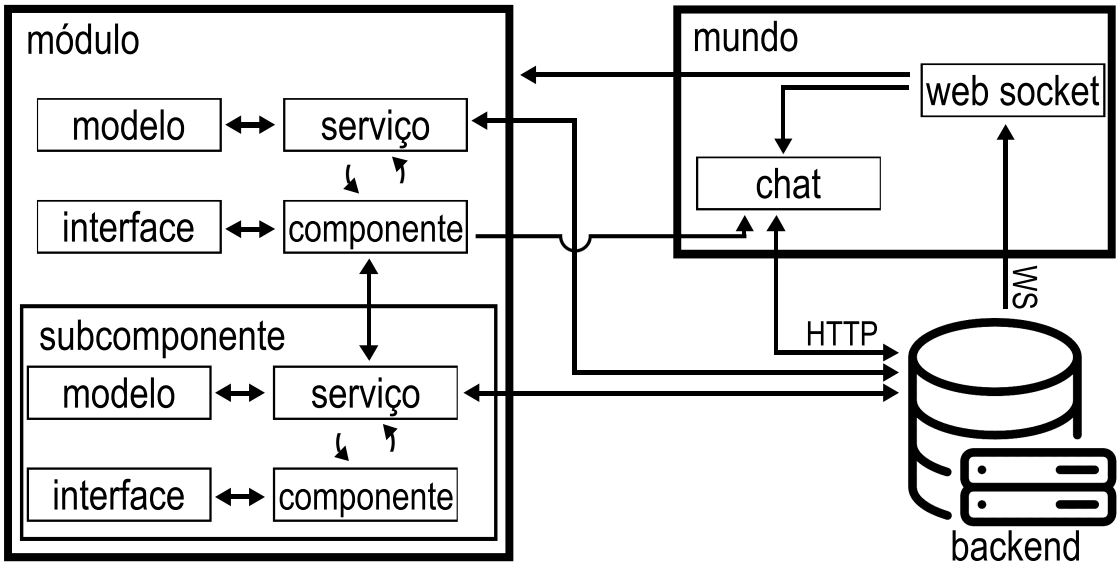


Figura 16: Diagrama de Componentes do *frontend*.

atribuição/desatribuição de selos verdes) é enviada ao *backend*.

A Figura 16 mostra o diagrama de componentes de como o *frontend* do projeto funciona. Na figura, o bloco *módulo* é genericamente qualquer um dos *módulos* que foram implementados, com exceção do `mundo`, que são *subcomponentes* comuns aos outros (será apresentado com detalhes mais à frente). Portanto, ao ser acessada uma página (do `Vereador`, por exemplo) a classe principal do *módulo* inicia os *subcomponentes* necessários (inclusive os que ficam alojados em `mundo`) para o funcionamento da página. Além dos *módulos* principais (como `businessman` – *empresário* – e `farmer` – *agricultor*), tem-se ainda os *módulos* auxiliares: `alternative-second-page` (janela alternativa para onde os jogadores que não jogam como um papel eleito vão na segunda etapa para que, ainda que não participem do jogo com ações, possam usar o chat e conversar com os outros jogadores, para tirar dúvidas com os gestores da cidade, por exemplo), `home` (janela inicial do jogo, onde pode-se entrar em um jogo como um jogador ou criar um novo jogo como Mestre), `game-over` (janela final da partida, onde mostra-se um resumo dos jogadores), `master` (janela onde o mestre controla o jogo, iniciando novas etapas ou finalizando o jogo), e `world`.

A interface responsável por acessar em uma partida é a `home`, apresentada na Figura 17. Nela, percebe-se a presença de dois formulários: (1) ”Novo Jogo”: onde a pessoa que será o Mestre da partida insere a quantidade de jogadores que participarão da partida; e, (2) ”Ingressar em um Jogo”: onde, após o jogador inserir o ID do jogo enviado pelo Mestre, aparecerá uma lista com os papéis do mundo para o jogador escolher o que lhe foi sorteado. Em nível de código (vide Figura 18), esse *módulo* é dividido em 2 componentes, onde um é responsável por criar um mundo (processando o formulário “Novo Jogo”) e o outro por encaminhar os jogadores para o mundo requerido (processamento do “Ingressar



Figura 17: Interface da página inicial (`home`)

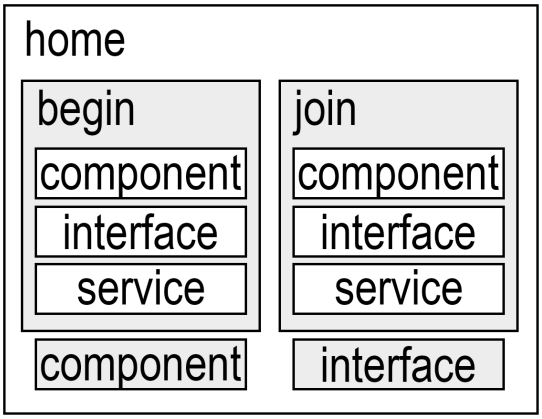


Figura 18: Diagrama de blocos do *módulo* `home`

em um Jogo").

Após a criação de um mundo, o Mestre é redirecionado para a janela apresentada na Figura 19, que contém as informações do Mundo, sendo em (1) ID do jogo, (2) etapa e rodada correntes, quantidade de jogadores da partida, poluição e produtividade atuais do mundo, (3) eleitos correntes e (4) *flags* que indicam a entrada dos jogadores nos papéis), (5) o botão para finalizar a etapa/rodada, (6) o botão para finalizar o jogo e, por fim, (7) o chat. Além do *componente* principal do mestre, existem os *subcomponentes* `rightside-bar`, representado por (4) na Figura 19, mostrando o estado dos jogadores (se entraram ou não na partida, assim como se o servidor não recebeu a jogada desse jogador) e o chat (localizado em (7) e importado do *módulo* `world`). O `rightside-bar` foi separado do *componente* principal por ter um comportamento próprio. Além de receber os eventos via *web socket*, ele ainda faz chamadas no início de cada etapa para saber o estado dos jogadores (se já entraram ou se não conseguiram entrar na etapa seguinte), recebendo os próximos eventos (de término da etapa ou de entrada na partida de algum

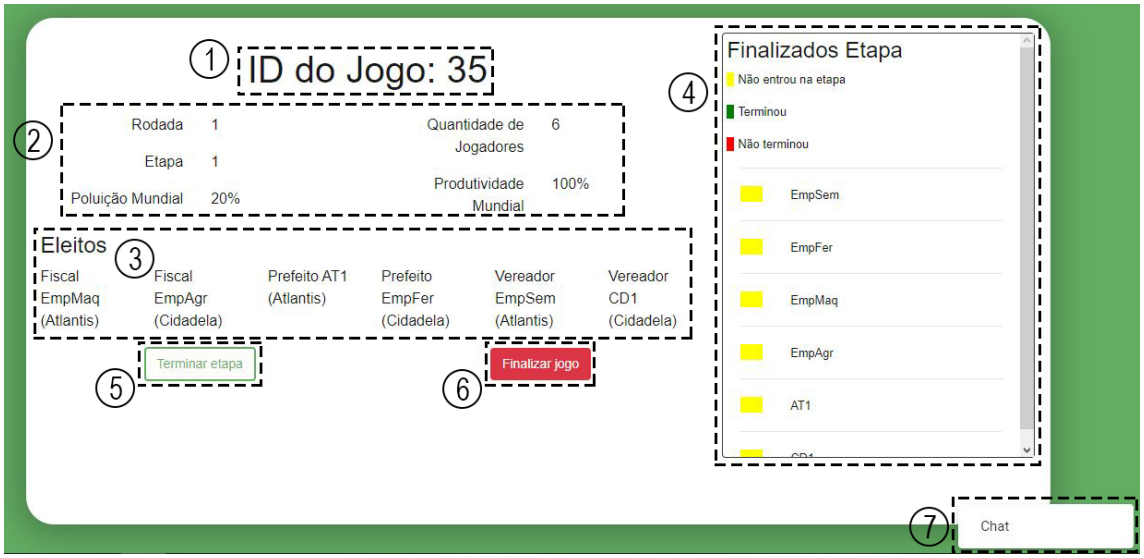


Figura 19: Interface do Mestre (`master`).

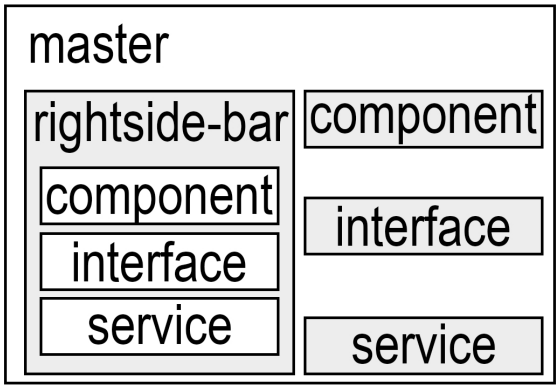


Figura 20: Diagrama de blocos do *módulo* `master`.

jogador que a interface demorou mais a ser carregada) pelo *web socket*. A disposição em código desse *módulo* pode ser vista na Figura 20.

O chat, além de outros *componentes*, *serviços*, *modelos* (estruturas de dados) e constantes comuns nas interfaces do jogo são pertencentes ao *módulo* `world`. A distribuição em visão de desenvolvimento pode ser vista no diagrama da Figura 21. São eles: `alert` (*componente*, *serviço* e interface que mostram uma mensagem ou aviso no topo da tela quando necessário, por exemplo, ao receber um orçamento de algum Empresário, esse alerta aparecerá no topo da janela do Agricultor avisando); `chat`, `confirming-modal` (para confirmar o término após apertar o botão, utilizado pelo mestre em ambos os botões da interface); `constants` (conjunto de constantes utilizados por todos os *componentes*); `header` (*componente* e interface utilizados pelas interfaces das cinco classes de personagens e pela janela alternativa da segunda etapa, mostrando dados do jogador, como nome, cidade e quantidade de dinheiros); `inline-alert` (*componente*, *serviço* e interface que mostra um alerta dentro de formulários, como va-

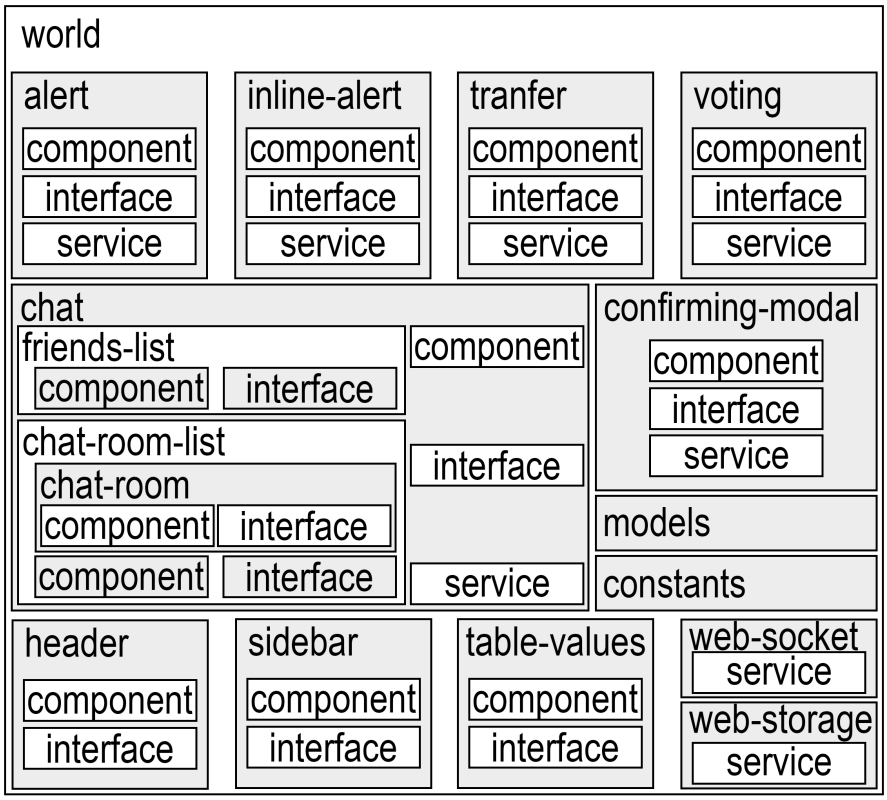


Figura 21: Diagrama de blocos do *módulo* `world`

lor inválido na caixa de digitação para entrar em um jogo, ao colocar o ID do jogo); `modelos`, `sidebar` (*componente* e *interface* responsáveis por abrir uma aba lateral na esquerda das interfaces dos personagens mostrando, por exemplo, os orçamentos enviados pelos empresários na interface do Agricultor); `table-values` (*componente* e *interface* que apresentam nas interfaces somente as tabelas necessárias àquela interface, como valores dos produtos, multas, imposto e poluição na interface do Empresário); `transfer` (*componente*, *serviço* e interface para o envio de Dinheiros para outros jogadores); `voting` (*componente serviço* e interface que aparecem a cada 2 rodadas nas interfaces do Empresário e do Agricultor para ser realizada a votação, onde, pelo *serviço*, faz-se a requisição da lista de "candidatos" e o envio do voto ao *backend* – sua interface é apresentada na Figura 22); `web-socket` e `web-storage` (*serviço* que guarda, recupera e apaga os dados salvos no *storage* do navegador, utilizado para guardar informações do personagem para a troca de janelas e para o caso de recarregar a página).

A interface do chat é mostrada na Figura 23, onde: (1) são mostrado os "contatos" do personagem (todos os outros personagens, menos ele mesmo, e o mestre), controlado pelo *componente* `friends-list`; em (2) as janelas de conversa com os outros jogadores, onde a estrutura é montada pelo *componente* `chat-room-list` e cada sala pelo `chat-room`. Percebe-se a presença de um ponto verde no cabeçalho da janela da conversa com o EmpFer e ao lado do nome do EmpFer na lista de amigos, o que indica que

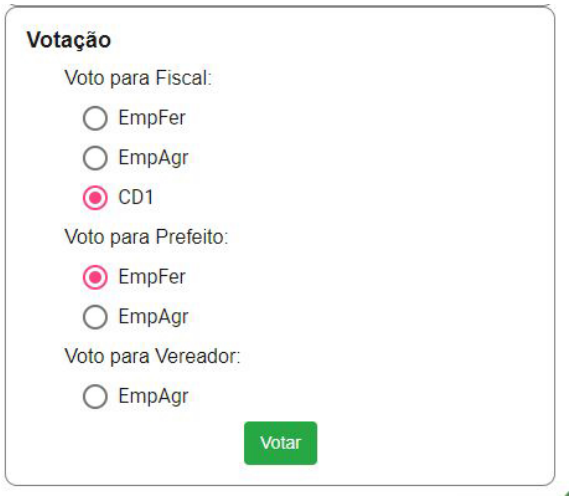


Figura 22: Interface da votação (`voting`).

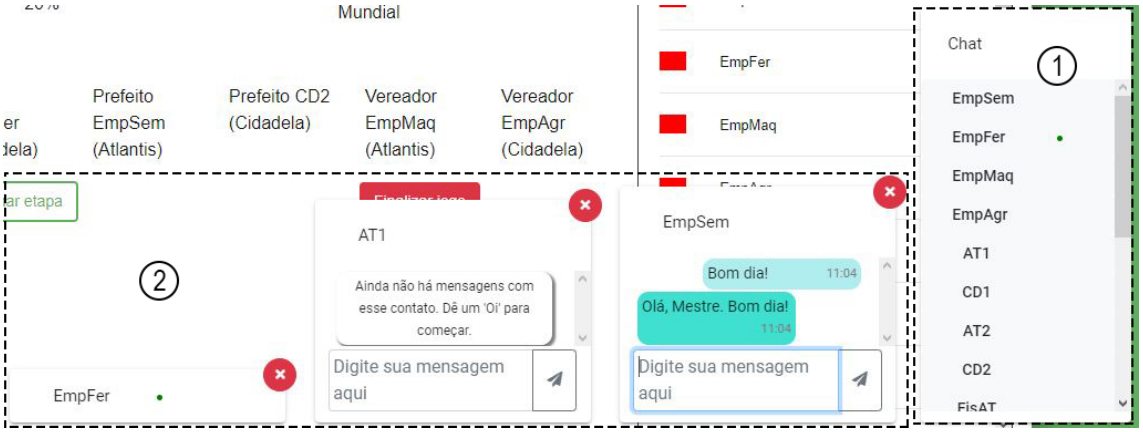


Figura 23: Interface do chat.

há uma mensagem não lida nesta conversa.

O *serviço* `web-socket` é responsável pela conexão com o servidor e recebimento dos eventos por ele enviados. Além disso, tem a capacidade da troca de conexão para quando o jogador tiver um papel na primeira etapa e outro na segunda etapa (Empresário e Prefeito, por exemplo, se tiver sido eleito) e, assim, receber os eventos enviados à ele. A Tabela 3 mostra os eventos possíveis do jogo. Percebe-se, ainda nessa tabela, que cada evento envia um tipo de mensagem sendo, algumas delas, os orçamentos (e respostas a estes) entre Empresários e Agricultores e as sugestões (e respostas) entre Vereadores e Prefeitos. Percebe-se, porém, que o evento `game status` pode ter diferentes valores, o que pode ser visto na Tabela 4. Além dos eventos do jogo, há os eventos do chat (notificações e mensagens) quando recebidas. Dependendo do evento recebido (do jogo ou do chat), esse evento é enviado à lugares diferentes na interface. Caso seja recebido um evento do jogo, o evento é enviado pelo *serviço* da interface corrente (do Vereador, por exemplo) para o *componente* responsável e, este, processa esse dado como precisa. Se for recebido um evento do chat, porém, esse evento é enviado diretamente aos *sub-*

Tabela 3: Tipos de eventos do jogo possíveis

Código	Descrição	Mensagem
0	*Game status*	Objeto com *status* (inteiro), rodada do evento (inteiro) e etapa do evento (inteiro)
1	Jogador começou etapa	Objeto com id do personagem (inteiro), nome do personagem (string) e etapa do evento (inteiro)
2	Jogador terminou jogada	Objeto com id do personagem (inteiro), nome do personagem (string) e etapa do evento (inteiro)
3	Orçamento	Objeto do tipo orçamento
4	Resposta ao orçamento	Objeto do tipo orçamento
5	Sugestão	Objeto do tipo sugestão
6	Resposta à sugestão	Objeto do tipo sugestão

Tabela 4: Lista com os Game Status

Código	Descrição
0	Início etapa
1	Todos os jogadores entraram na etapa
2	Todos os jogadores acabaram a etapa
3	Mestre acabou a etapa
4	Fim do jogo

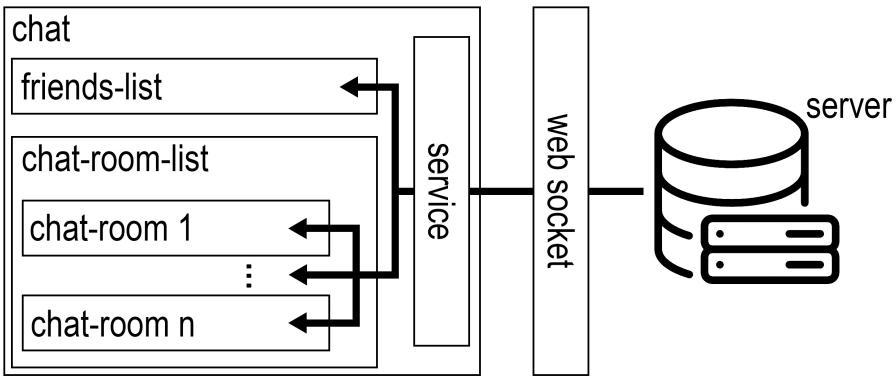


Figura 24: Fluxo do recebimento de um evento do chat.

componentes do `chat` pelo *serviço* próprio desse *componente*. A Figura 24 mostra como o recebimento de um evento do chat ocorre.

A `modelos` é uma pasta onde tem-se presente os modelos comuns a serem usados pelos *módulos*, como `person` e `game-notification`. Esses *modelos* são interfaces de estruturas de dados a serem usados como tipos de dados. A Figura 25 mostra o *modelo* `person` para exemplificação.

A interface do Empresário é apresentada na Figura 26, onde: (1) o *componente* `header`, importado do *módulo* `world`; (2) o botão para o acionamento da aba lateral (`sidebar` do `world`), oriundo do *subcomponente* `selling-board`; (3) botões

```
export interface Person {
    id: number;
    nome: string;
    cidade: string;
    saldo: number;
}
```

Figura 25: Definição do tipo `Person`.

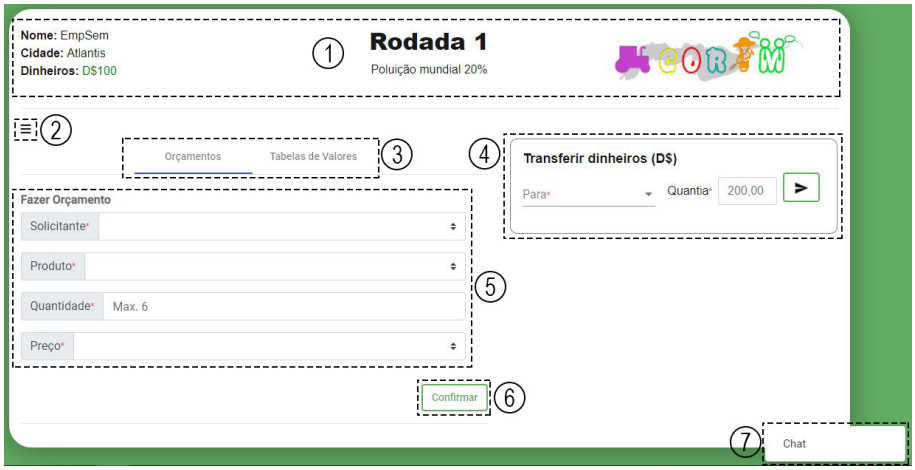


Figura 26: Interface do Empresário (`businessman`).

para a troca da visualização principal da página, trocando entre (5) "Fazer Orçamento" (fruto do *subcomponente* `order-product`) e "Tabelas de Valores" (que apresenta as tabelas necessárias ao jogador do papel, como multas e preços de produtos, vindo do `table-values` que é importado como *subcomponente* do `world`); (4) espaço para transferir Dinheiros para outros jogadores (oriundo do `transfer` do `world`); (5) campos para solicitar orçamento, indicando para que será feito, qual o prouto, a quantidade e preços desejados; (6) botão para o envio do orçamento (também faz parte do `order-product`); e (7) o chat. O botão representado por (6), ou seja, o de enviar um orçamento a um Agricultor, envia via API Rest o orçamento ao servidor que o envia ao Agricultor. Quando é retornado uma resposta a esse orçamento, o *web socket* envia esse evento (que contém a resposta) ao *subcomponente* `selling-board`, que lista essa resposta e chama o *serviço* `alert` (do *módulo* `world`) para avisar o jogador que há uma nova resposta. A estrutura do código desse *módulo* pode ser vista na Figura 27.

Ao entrar na janela do Agricultor, o jogador se depara com a interface mostrada na Figura 28, cuja estrutura em nível de código pode ser vista na Figura 29. Percebe-se em (1) a mesma estrutura da janela do Empresário, *componente* `header`; em (2) o botão também presente na janela do Empresário, que abre uma aba lateral onde são mostrados os orçamentos enviados ao Agricultor, fruto do *componente* `buying-board` e *sidebar*;

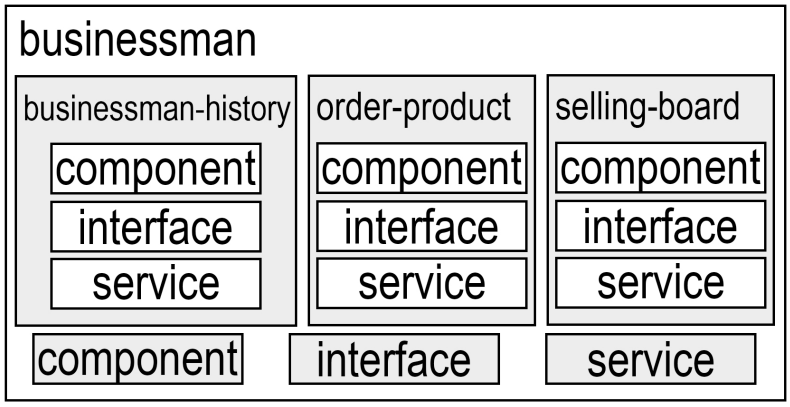


Figura 27: Diagrama de blocos do *módulo* `businessman`.

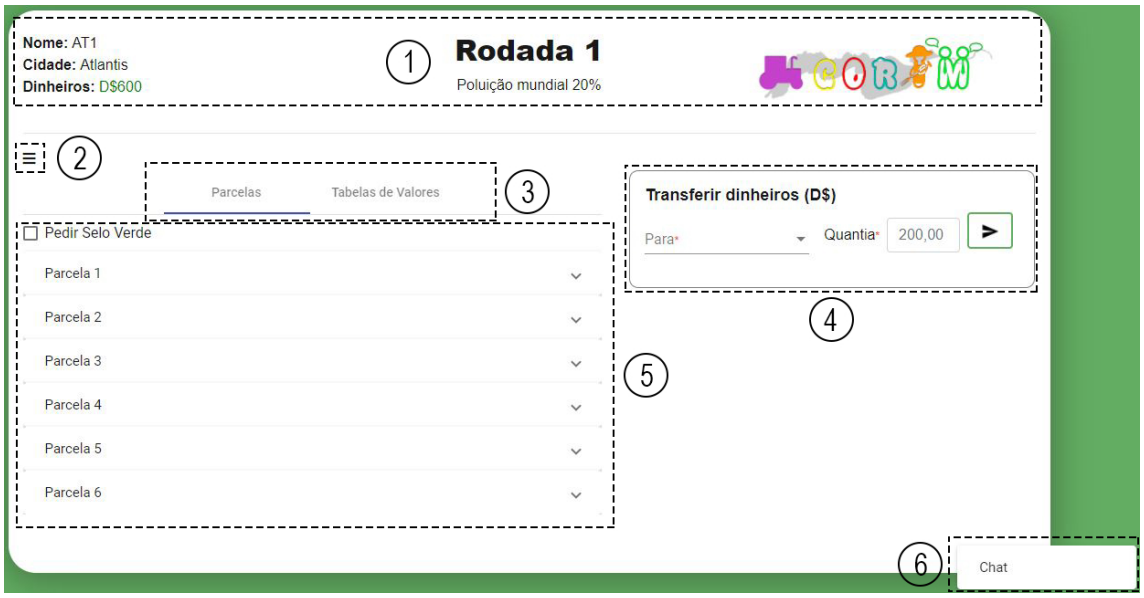


Figura 28: Interface do Agricultor (`farmer`).

em (3) há o menu para mostrar o formulário das parcelas (*componente* `parcel`, que está em (5)) ou as tabelas de valores (`table-values`); em (4) o formulário para envio de dinheiros (`transfer`); e, por fim, em (6) o chat, sendo esse, `header`, `sidebar`, `transfer` e `table-values` *componentes* importados de `world`).

O formulário para preenchimento das parcelas do Agricultor é um conjunto de várias estruturas que se expandem na interface aparecendo o formulário de cada parcela separado do formulário da outra parcela. Pode ser visto um desses formulários abertos na Figura 30. Na figura, pode-se ver que as opções estão inativas. Isto ocorre porque o Agricultor em questão não ter comprado ainda nenhum produto. Assim que é realizada uma compra, fica disponível acima de "Pedir Selo Verde" a quantidade e o produto comprado, podendo, assim, ser marcado no formulário em qualquer parcela. Além disso, os formulários de cada Parcela seguem as regras do Gorim: só é possível o plantio de um produto de cada tipo, sendo que não é possível a inserção de um Agrotóxico e um Pacote de Máquinas na

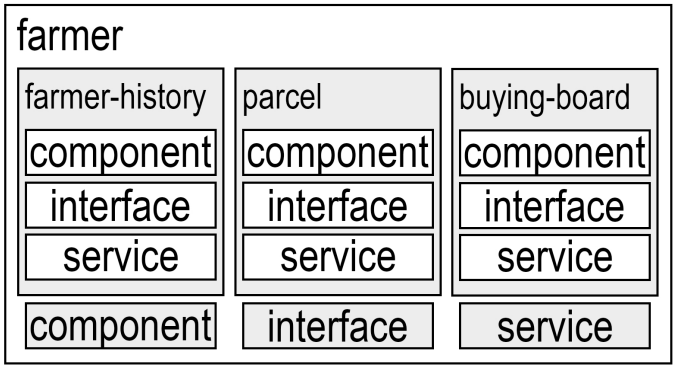


Figura 29: Diagrama de blocos do *módulo* `farmer`.

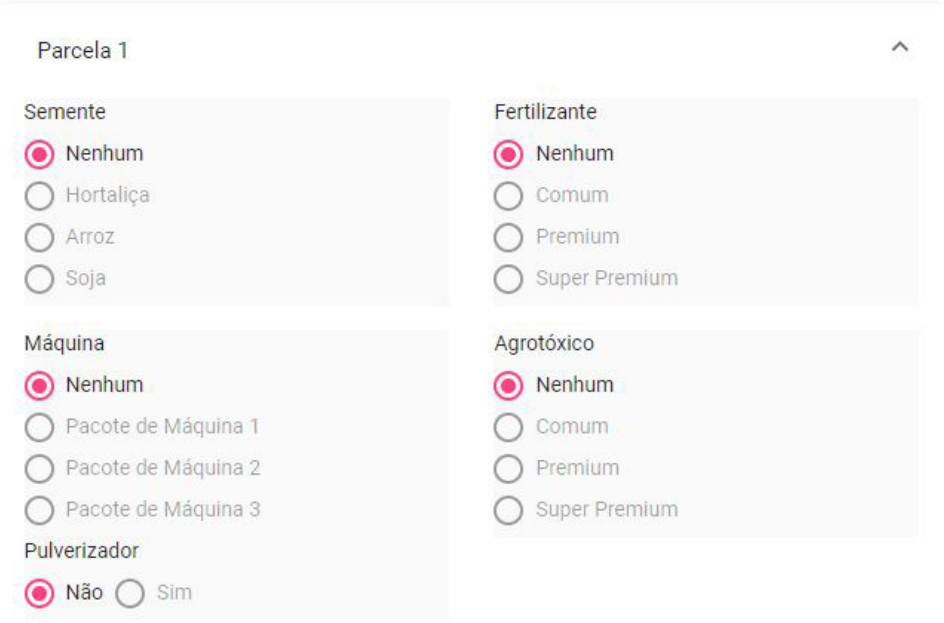


Figura 30: Formulário referente à Parcela 1 na interface do Agricultor.

mesma parcela, a não ser que a Máquina seja um Pulverizador. Ao marcar a opção em alguma parcela, portanto, o contador de produtos disponíveis para o plantio diminui.

Quando o Empresário envia um orçamento a algum Agricultor, ele utiliza o formulário do *componente* `order-product`, que envia o orçamento via REST. Esse orçamento aparece na interface do Agricultor (recebido via *web socket*) na aba lateral (que é uma junção dos *componentes* `sidebar` do `world` e do `buying-board` do `farmer`), fazendo parte de uma lista, com dois botões: um para negar e um para aceitar o orçamento, o que pode ser visto na Figura 31(A). Ao responder esse orçamento, essa resposta é enviada via REST ao servidor e encaminhada via *web socket* ao Empresário, onde é apresentada em uma aba lateral parecida com a do Agricultor (junção do `sidebar` do `world` com o `selling-board` do `businessman`). Porém, no caso do Empresário, a resposta não apresenta botões de resposta, mas bordas com cores diferentes diferenciando orçamentos aceitos dos negados: em verde os aceitos e em vermelhos os negados. A visão do Em-

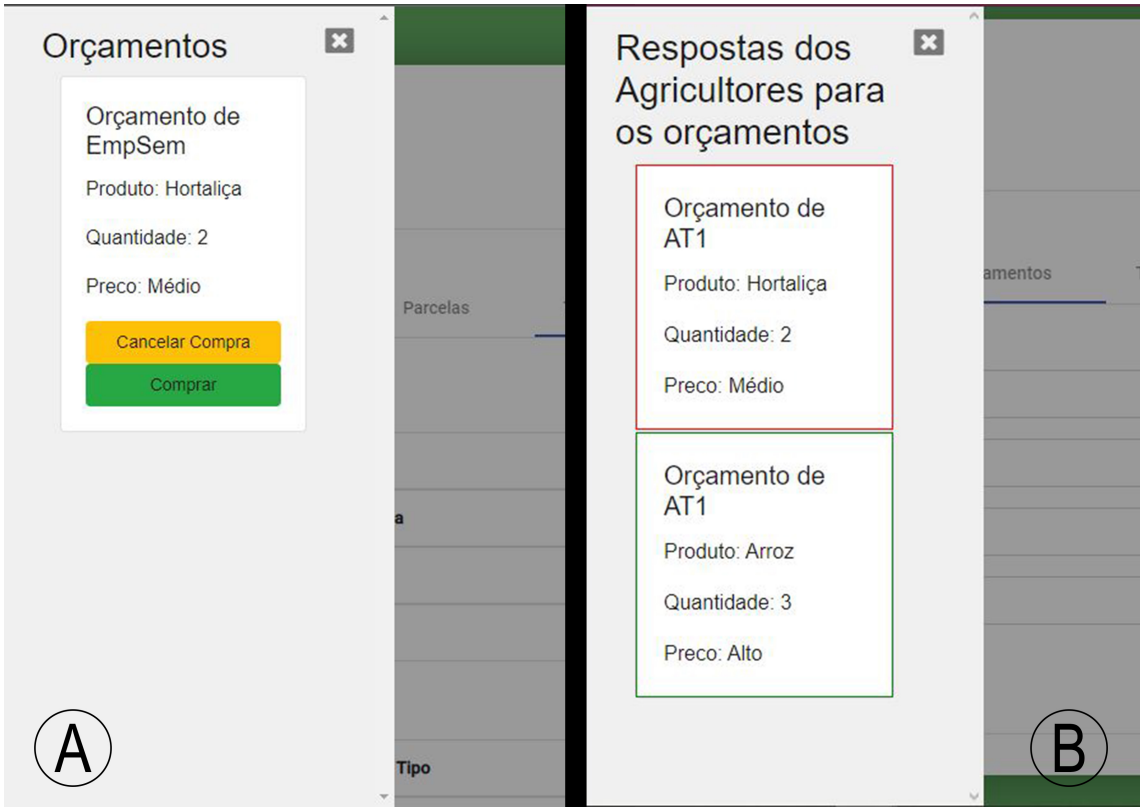


Figura 31: Aba lateral mostrando (A) os orçamentos recebidos pelo Agricultores (B) as respostas aos orçamentos recebidos na interface do Empresário.

presário é apresentada na Figura 31(B). O mesmo acontece no caso do envio-resposta de sugestões entre Vereadores e Prefeitos, com o Vereador enviando a Sugestão, chegando no Prefeito e, este, respondendo ao Vereador.

A interface desenvolvida para os Fiscais Ambientais pode ser vista na Figura 32 e, sua estrutura, no diagrama da Figura 33. Percebe-se que as estruturas identificadas por (1), (3), (6) e (7) se repetem das interfaces anteriormente apresentadas, sendo, consecutivamente: `header`, botões que trocam o que é apresentado no centro da janela para as tabelas de valores (`table-values`); formulário de transferência de dinheiros (`transfer`) e chat. Em (2) também é aberta a aba lateral (possibilitada por `sidebar` do *módulo* `world`) mas nesta interface são somente apresentados os Agricultores que pediram Selo Verde ao Fiscal Ambiental. Em (5), se "Ações" tiver selecionado em (3), são mostrados ao jogador dois formulários como as Parcelas do Agricultor (fechados inicialmente, mas quando abertos, apresentam os dados conforme Figura 34) para serem feitas as multas (*componente* `fine`) e atribuir/retirar selos verdes (`green-seal`). A novidade nessa interface é o botão identificado por (4) na Figura 32, que abre uma nova aba no navegador do jogador, onde ele vê o histórico do papel, ou seja, o `ArquivoResumo` que os jogadores do Gorim recebiam impressos, mostrando o resumo para o papel nas etapas/rodadas passadas.

Nos formulários do Fiscal (Figura 34) é possível aplicar multa nos produtores da ci-

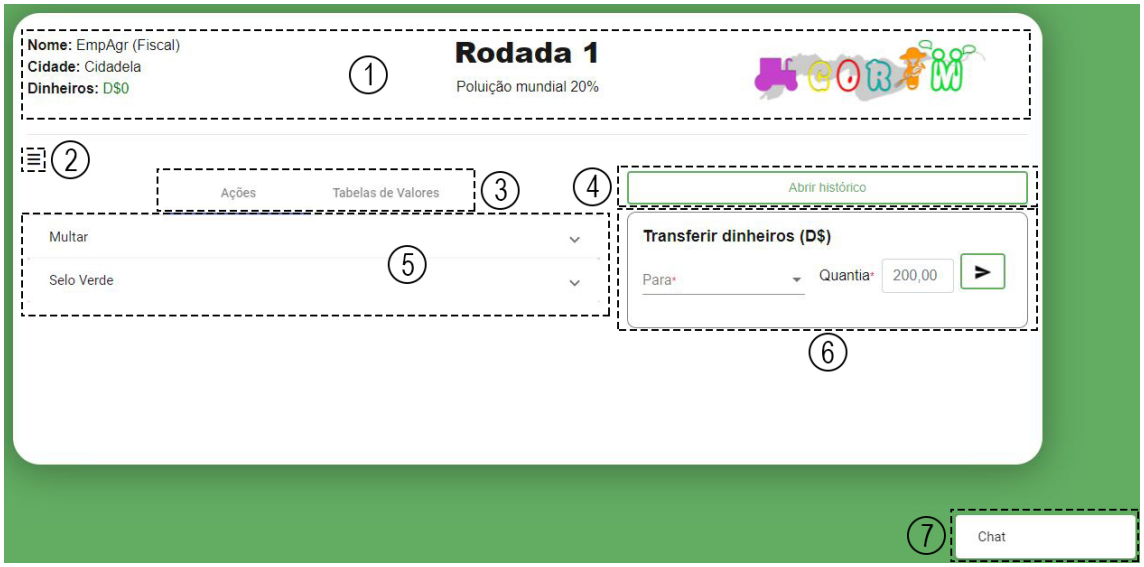


Figura 32: Interface do Fiscal Ambiental (`supervisor`).

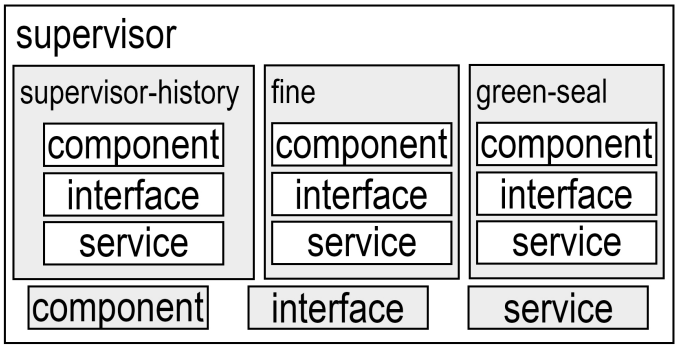


Figura 33: Diagrama de blocos do *módulo* `supervisor`.

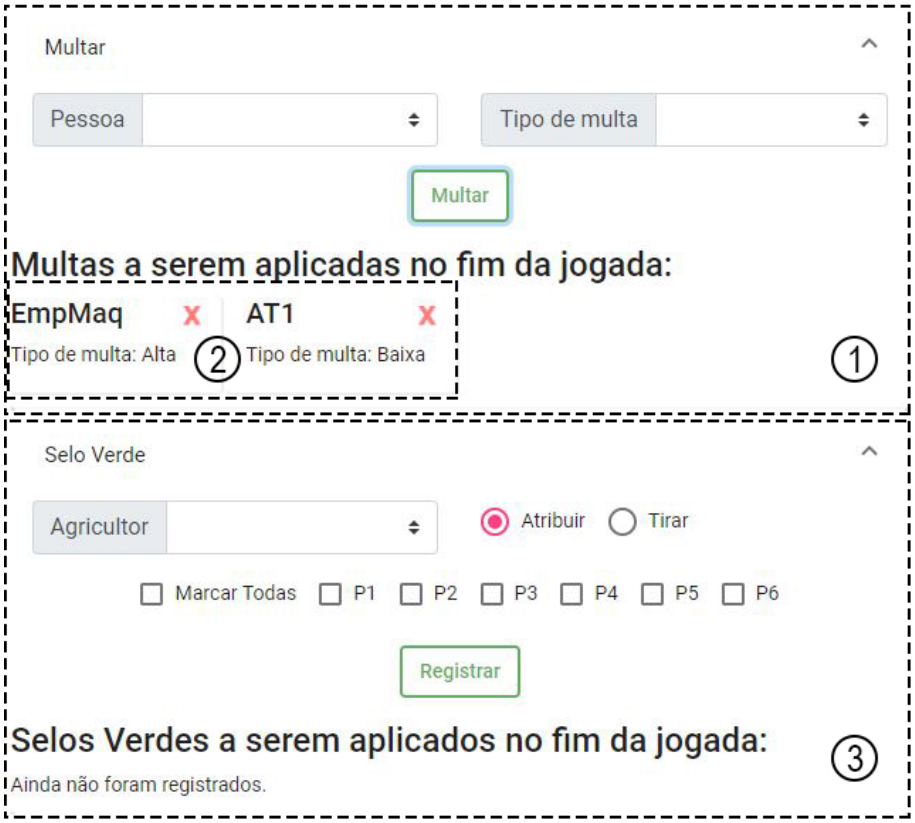


Figura 34: Interface dos formulários “Multar” e “Selo Verde” do Fiscal Ambiental.

dade (em (1)) e atribuir/tirar Selos Verdes das parcelas dos Agricultores (3). Além disso, há um sistema de cancelar a ação já selecionada, o que pode ser visto em (2), clicando-se nos “X”.

O botão para acesso ao histórico é apresentado nas interfaces a partir da segunda Etapa da primeira Rodada, ou seja, na primeira vez que os jogadores jogam como Empresários e Agricultores, esse botão não está na interface (como visto nas Figuras 26 e 28), apenas a partir da 2ª Rodada. O histórico dos papéis são produzidos por um JSON buscado pelos *componentes* `[papel]-history`, presentes no *módulo* de cada interface. No caso dos históricos do Empresário e do Agricultor, são separados por Rodada, enquanto os dos Fiscais Ambientais, Prefeitos e Vereadores são separados por Rodada e por Etapa, onde cada Etapa/Rodada na interface pode ser expandida ou recolhida. Na Figura 35 pode-se ver um exemplo de histórico de um Empresário e na Figura 36 um exemplo de um Fiscal.

O jogador do Prefeito tem como interface a janela apresentada na Figura 37. Os componentes em nível de código podem ser vistos na Figura 38. As estruturas sinalizadas por (1), (3), (4), (6) e (7) são as mesmas identificadas nas outras interfaces: `header`, botões que trocam ente formulários e `table-values`, botão para histórico (nesse caso `mayor-history`), `transfer` e chat, respectivamente. Em (2) é aberto a aba lateral com as sugestões enviadas pelo Vereador, com botões para aceitar ou negar, utilizando o *componente* `sidebar` do `world` e a lista de sugestões buscadas no/en-



Figura 35: Interface da janela de histórico do Empresário.



Figura 36: Interface da janela de histórico do Fiscal Ambiental.

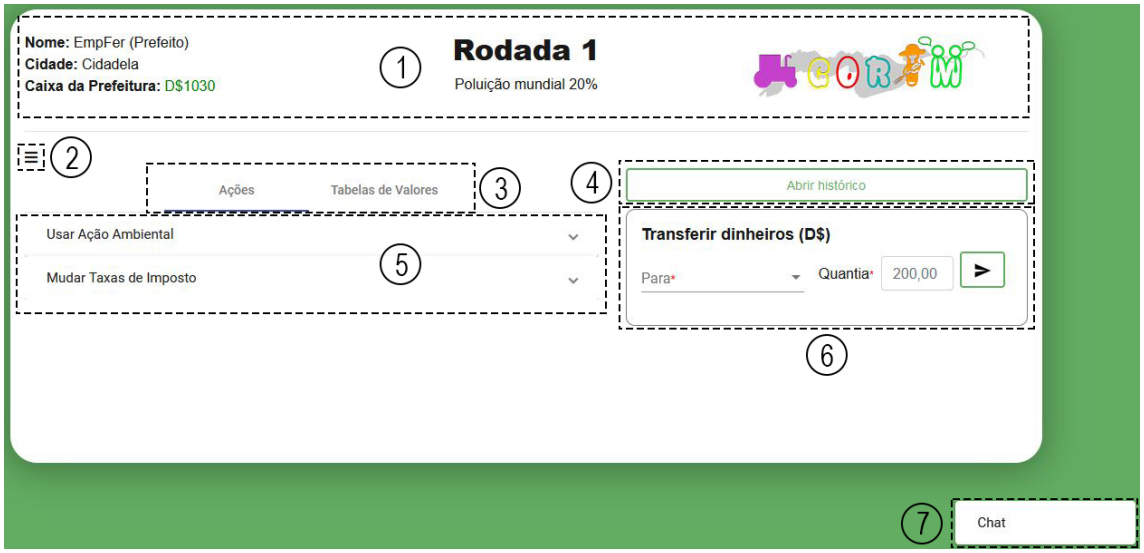


Figura 37: Interface do Prefeito (`mayor`).

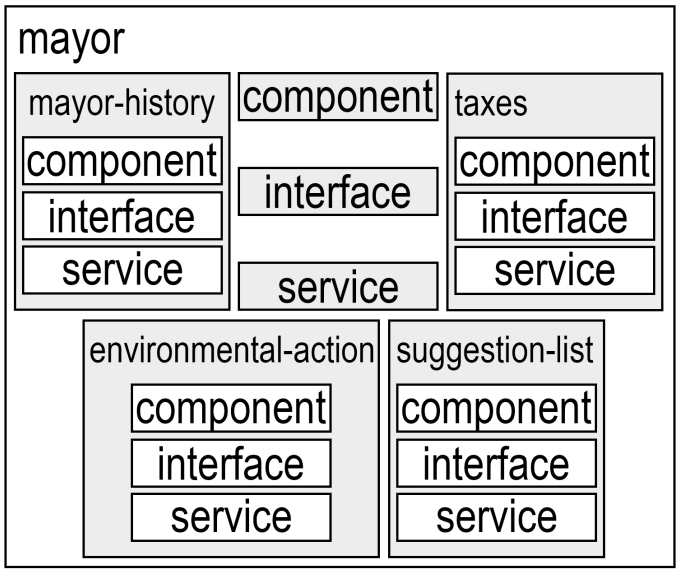


Figura 38: Diagrama de blocos do *módulo* `mayor`.

viadas ao servidor e apresentadas pelo `suggestion-list`. Os formulários identificados por (5) são expansíveis e apresentam o mesmo comportamento de selecionar/-cancelar que os formulários do Fiscal Ambiental. Em "Usar Ação Ambiental" o Prefeito pode selecionar as Ações Ambientais que deseja aplicar no mundo (*componente* `environmental-action`) e no "Mudar Taxa de Imposto" é possível a modificação nas Taxas de Impostos para pequenos, médio e grandes produtores (*componente* `taxes`). Estes formulários podem ser visualizados na Figura 39.

O Vereador tem como interface a janela apresentada pela Figura 40, processada pelo diagrama da Figura 41, que contém as mesmas estruturas das outras interfaces. O que altera é que em (2) é aberta a aba lateral com as respostas do Prefeitos quanto às sugestões feitas a eles (combinando o *componente* `sidebar` com `response-list`), (5) apre-

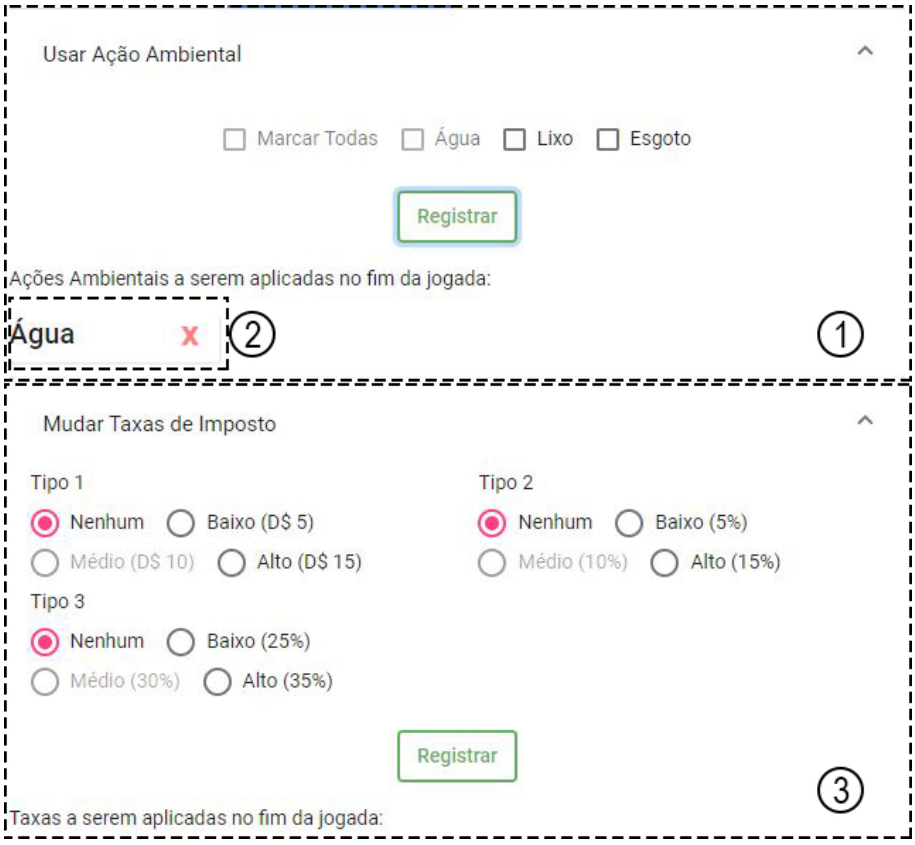


Figura 39: Interface dos formulários “Usar Ação Ambiental” e “Mudar Taxas de Imposto” do Prefeito.

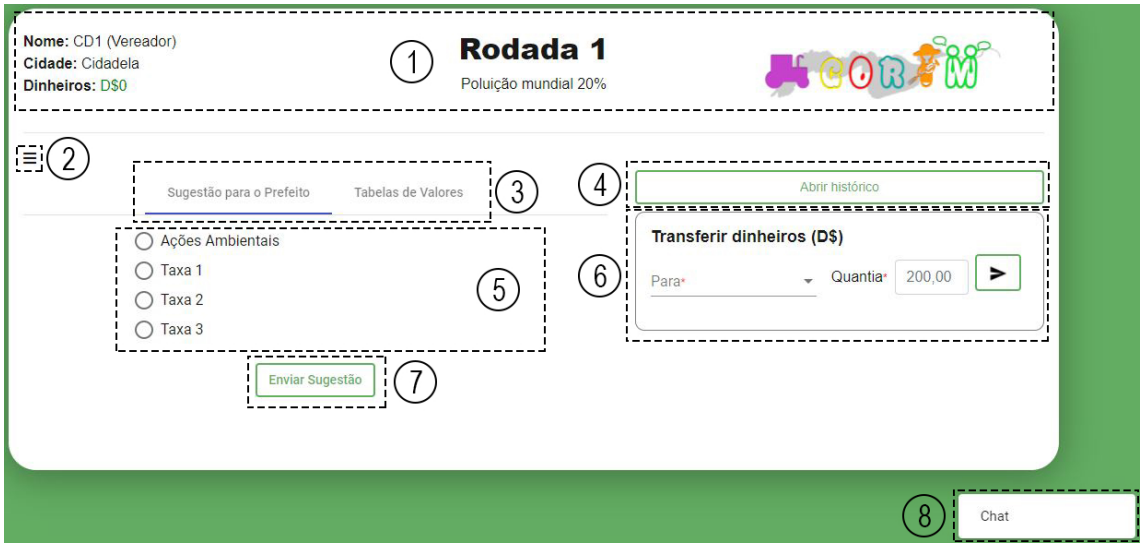


Figura 40: Interface do Vereador (`alderman`).

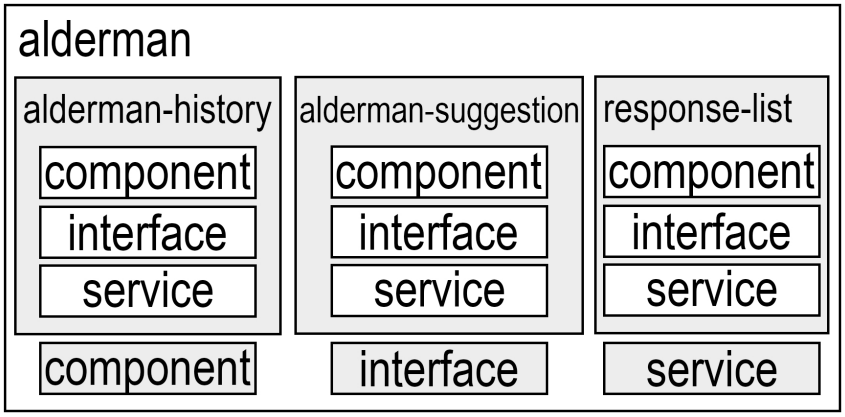


Figura 41: Diagrama de blocos do *módulo* `alderman`.

senta o formulário de envio de sugestões que, ao ser clicado em uma das opções, mostra na interface uma caixa de seleção que apresenta as opções dentro daquela que foi escolhido pelo Vereador (por exemplo clicando em “Ações Ambientais”, aparecerá uma caixa de seleção onde terão como opções “Água”, “Lixo” e “Esgoto”) e em (7) o botão de envio de sugestão.

Porém, quando o jogo tem mais de 6 participantes, alguns jogadores não jogam como um papel eleito, ou seja, ficam sem jogar na segunda Etapa das Rodadas. Para esses jogadores, é utilizada a interface apresentada na Figura 42, que é processada pelo *módulo* `alternative-second-page-stage`, ilustrado na Figura 43. Percebe-se que a janela é informativa, mostrando o nome e a cidade do personagem em (1) (utilizando o *componente* `header`), avisando que o jogador em questão não participa como papel eleito (2), mostrando quem são os eleitos do mandato corrente (3) e o chat (4), o que permite a troca de mensagens com os jogadores que estão jogando como um papel eleito para serem tiradas dúvidas quanto a imposto e multa, por exemplo, e com os outros jogadores que



Figura 42: Interface dos jogadores sem papel na segunda Etapa (`alternative-second-page-stage`).

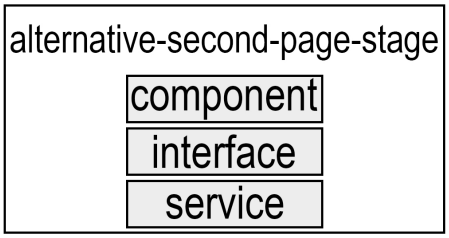


Figura 43: Diagrama de blocos do *módulo* `alternative-second-page-stage`.

não foram eleitos também, para serem traçadas possíveis estratégias de jogadas para as próximas Rodadas.

Por fim, há a interface do final do jogo (Figura 44), *módulo* `game-over` (Figura 45). Essa janela sinaliza o fim do jogo (que foi finalizado pelo Mestre) e todos os jogadores (o Mestre, incluído) são enviados a ela. Nela, vê-se um resumo geral dos papéis Empresário e Agricultor, mostrando a quantidade de Dinheiros que eles possuem. Além disso, mostra informações do Mundo (ID do Jogo, Etapa e Rodada que o jogo terminou e poluição final). Essas informações são utilizadas para serem discutidas com os jogadores no fim do jogo.

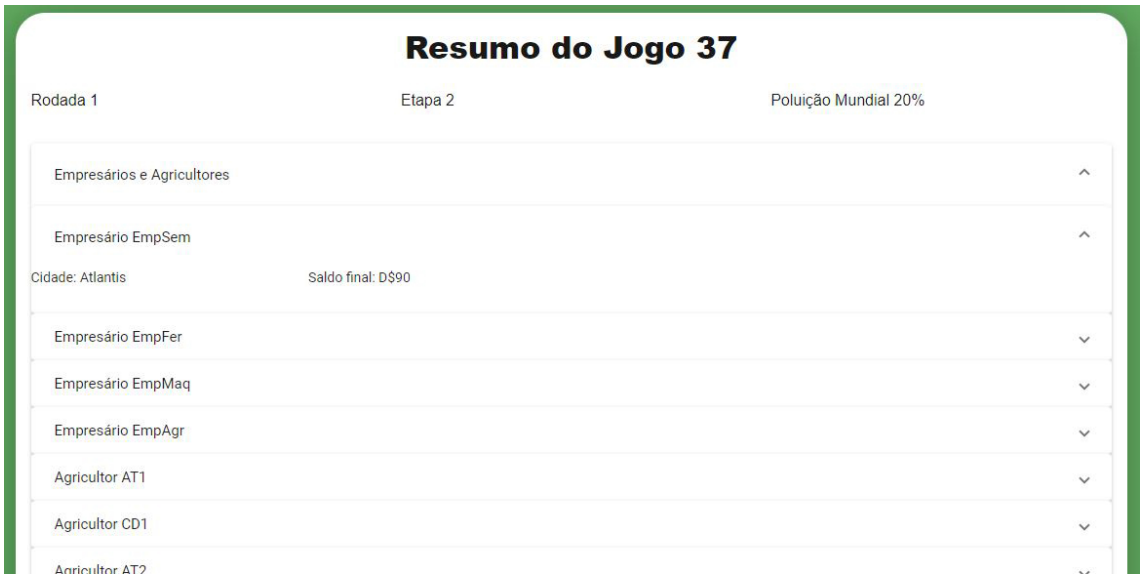


Figura 44: Interface do fim do jogo (`game-over`).

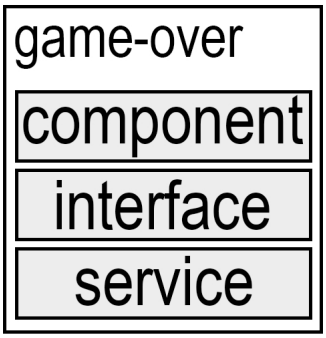


Figura 45: Diagrama de blocos do *módulo* `game-over`.