# Documento de Requisitos do Projeto QA+: Um Jogo Para o Ensino de Teste e Qualidade de Software

Documento de Requisitos do Projeto, apresentado à UTFPR como requisito parcial para a disciplina de Engenharia de Requisitos do PPGCA.

**Curitiba**
**2022**

**Vladimir Belinski**

# Documento de Requisitos do Projeto QA+: Um Jogo Para o Ensino de Teste e Qualidade de Software

Documento de Requisitos do Projeto, apresentado à UTFPR como requisito parcial para a disciplina de Engenharia de Requisitos do PPGCA.

**Orientador:**
Nome do orientador: Prof. Dr. Laudelino Cordeiro Bastos
E-mail do orientador: bastos@dainf.ct.utfpr.edu.br

**Curitiba**
**2022**

# Histórico de Modificações

| Data | Versão | Descrição | Autor |
|---|---|---|---|
| 05/05/2022 | 1.0 | Disponibilização da primeira versão completa do projeto | Vladimir Belinski |

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS E QUADROS

# LISTA DE ABREVIATURAS E SIGLAS

**ACM:** *Association for Computing Machinery* – Associação para Maquinaria da Computação (literal)

**ECF:** *Environmental Complexity Factor* – Fator de Complexidade do Ambiente

**IEEE:** *Institute of Electrical and Electronics Engineers* – Instituto de Engenheiros Eletricistas e Eletrônicos

**ISTQB:** *International Software Testing Qualifications Board*

**LGPD:** Lei Geral de Proteção de Dados

**REQ:** Requisito

**RF:** Requisito Funcional

**RNF:** Requisito Não Funcional

**SBC:** Sociedade Brasileira de Computação

**TCF:** *Technical Complexity Factor* – Fator de Complexidade Técnica

**UC:** *Use Case* – Caso de Uso

**UAW:** *Unadjusted Actor Weight* – Peso não ajustado dos atores

**UML:** *Unified Modeling Language* – Linguagem de Modelagem Unificada

**UUCP:** *Unadjusted Use Case Point* – Pontos de caso de uso não ajustados

**UUCW:** *Unadjusted Use Case Weight* – Peso não ajustado dos casos de uso

# 1 ESTADO DA ARTE

A qualidade é um aspecto vital no contexto de desenvolvimento de software (PRESSMAN, 2015; SOMMERVILLE, 2011), sendo a atividade de teste fundamental para sua garantia (VALLE; ROCHA; MALDONADO, 2017; DINIZ; DAZZI, 2011). Devido à importância do teste de software este assunto é abordado em todos os currículos de referência para cursos de Sistemas e Tecnologia de Informação, Ciência da Computação e Engenharias de Computação e de Software propostos pela Sociedade Brasileira de Computação (SBC) e pela *Association for Computing Machinery* (ACM) juntamente a outras associações (SILVA; BERNARDI; MÜLLER, 2011).

Contudo, apesar da presença desta temática nos currículos de referência, diversos estudos apontam para uma série de fatores que acabam fazendo com que alunos de graduação costumeiramente apresentem lacunas de conhecimento sobre o assunto, bem como desenvolvam hábitos de teste inadequados, o que se mostra prejudicial tanto às suas carreiras profissionais quanto às áreas de qualidade e desenvolvimento de software em si (SHERIF *et al.*, 2020). Dentre as causas comumente associadas a tais observações destacam-se: a falta de motivação dos alunos, a percepção destes de que a atividade de teste é tediosa e trivial (ou muito difícil, dependendo do enfoque dado ao assunto), a abordagem tardia e/ou superficial do tema nos cursos de graduação, a existência de diferenças entre o que é ensinado academicamente e o que é necessário na indústria, a falta de atividades práticas, a complexidade dos materiais e, considerando o emprego de abordagens tradicionais de ensino-aprendizagem, a dificuldade tanto de ensinar de modo eficaz quanto de despertar o interesse dos alunos acerca do tema (SHERIF *et al.*, 2020; VALLE; ROCHA; MALDONADO, 2017; CLEGG; ROJAS; FRASER, 2017; DINIZ; DAZZI, 2011; ELBAUM *et al.*, 2007).

Tendo conhecimento desta problemática, nos últimos anos alguns estudos têm sido realizados com o objetivo de apoiar o ensino de teste de

software por meio de abordagens alternativas aos métodos e técnicas educativas tradicionalmente empregadas. Seguindo o que já vinha sendo investigado e aplicado em outras áreas de conhecimento (FARIAS *et al.*, 2012) estes estudos sugerem a utilização de jogos sérios e gamificação, conceitos que se relacionam à ideia do emprego de jogos e seus elementos visando a criação de um contexto envolvente e motivador de ensino-aprendizagem em relação a um domínio de conteúdo específico (SOSKA; MOTTOK; WOLFF, 2016).

Sabendo que o emprego de jogos e gamificação na educação pode ser eficiente, aprimorando, por exemplo, a motivação e a experiência de aprendizagem dos alunos (SHERIF *et al.*, 2020; ROJAS; FRASER, 2016; DINIZ; DAZZI, 2011), no presente trabalho será apresentado o documento de requisitos da proposta de um jogo sério nomeado QA+, o qual busca contribuir com o rol de jogos voltados ao ensino de teste e qualidade de software, de modo com que os problemas apresentados no início desta seção possam ser sanados ou, ao menos, reduzidos. Como diferencial em relação aos demais jogos já existentes pode-se destacar o fato da proposta compreender um ambiente com dinâmicas novas/próprias que possibilitam ao aluno tanto aprender quanto ensinar sobre os assuntos, disponibilizando materiais pré-definidos, mas ao mesmo tempo permitindo a criação e compartilhamento ilimitado de conteúdo pelo próprio jogador. Ainda, cabe destacar que o conjunto de recursos foi planejado de modo a apoiar também atividades desempenhadas por quem ensina, o que será exposto no decorrer do trabalho por meio de seus requisitos.

Nas subseções a seguir serão expostos conceitos básicos que podem auxiliar no entendimento da proposta (subseção 1.1), bem como discorrido brevemente acerca dos trabalhos relacionados (subseção 1.2).

## *1.1 Conceitos Básicos*

### 1.1.1 Jogos sérios, gamificação e aprendizagem baseada em jogos

De acordo com Silva (2012), jogos sérios consistem em jogos com finalidade educacional explícita e detalhadamente pensada, não possuindo como intenção principal a utilização para diversão. Contudo, isto não significa que a diversão não deva ocorrer, mas sim que ela deve ser utilizada para motivar o jogador a atingir os resultados de aprendizagem planejados (SILVA, 2012; SOSKA; MOTTOK; WOLFF, 2016).

Por sua vez, segundo Nascimento (2019) a gamificação consiste na utilização de elementos de jogos em um dado ambiente – um contexto não-jogo, segundo Jesus (2019) – objetivando motivar e aumentar o envolvimento das pessoas.

Por fim, Silva (2012) define a aprendizagem baseada em jogos como uma abordagem que emprega o jogo como ferramenta para o engajamento dos alunos em relação ao aprendizado, sendo que seu processo geralmente ocorre por meio de: (a) uma entrada, representada por um sistema que une conteúdo instrucional e elementos de jogos; (b) um processamento, caracterizado pela utilização deste sistema, o que desencadeia reações e comportamentos nos jogadores, bem como *feedbacks* do sistema; e (c) uma saída, correspondente ao alcance dos objetivos educacionais e atendimento dos resultados de aprendizagem, dados a partir do engajamento do jogador.

### 1.1.2 Elementos de jogos

Os elementos de jogos correspondem a pequenas partes que unidas (mas não apenas) acabam moldando um jogo, um jogo sério ou um sistema gamificado (WERBACH; HUNTER, 2012).

Para Werbach e Hunter (2012) os elementos mais importantes podem ser divididos em três categorias, sendo elas (em nível decrescente de abstração): (a) dinâmicas — aspectos abstratos do que se pretende no jogo (e.g.: emoções, narrativa, progressão, etc); (b) mecânicas — conduzem as ações dos usuários de modo a engajá-los (e.g.: desafios, recompensas,

*feedback*, etc); e (c) componentes — elementos concretos e visíveis aos usuários (e.g.: avatar, nível, ponto, quadro de classificação, etc).

Por fim, de modo a facilitar o entendimento dos conceitos acima, vale apresentar o seguinte exemplo: “recompensar (mecânica) um aluno atribuindo pontos (componente) provê a ele um senso de sua progressão (dinâmica)” (JESUS, 2019, p. 26). Por meio do exemplo percebe-se que os pontos são visíveis aos jogadores, enquanto o ato de recompensar e sua progressão são apenas perceptíveis. Contudo, os três se relacionam e são importantes para a construção do jogo.

## *1.2 Trabalhos relacionados*

Primeiramente, cabe destacar que o levantamento bibliográfico inicial deste trabalho foi realizado na segunda quinzena de outubro de 2021 por meio de pesquisas nas bases IEEE, ACM, *ResearchGate* e *Google Scholar*, tendo sido utilizada a seguinte *string* de busca: *("software testing" AND (gamification OR game) AND (education OR teaching OR teach OR learning OR learn)).* Ainda, ressalta-se que foi igualmente empregada a técnica conhecida como *snowballing*.

Através do levantamento bibliográfico efetuado puderam ser encontradas na literatura algumas abordagens e jogos voltados ao ensino de teste de software, sendo que estes variam em relação a aspectos como formato, temas abordados, entre outros.

Nos trabalhos de Soska, Mottok e Wolff (2016; 2017) e Beppe *et al*. (2018), por exemplo, são apresentados dois jogos físicos de cartas. No jogo criado por Beppe *et al*. (2018), nomeado *GreaTest Card Game*, os participantes desempenham o papel de analistas de teste, devendo indicar quais tipos de teste são mais adequados aos cenários apresentados nas cartas. Por sua vez, no jogo SOFTTY, descrito por Soska, Mottok e Wolff (2016; 2017), as cartas propõem a resolução de atividades e questões baseadas em conteúdos do nível *foundation* (básico) de certificações do *International Software Testing Qualifications Board* (ISTQB).

Jogos que igualmente utilizam o corpo de conhecimento do ISTQB e/ou que se relacionam a suas certificações são os criados por: Barbosa, Neves e Dias-Neto (2016), que apresentam o *JoVeTest*, um jogo da velha digital onde o usuário possui o direito de marcar seu símbolo caso responda corretamente uma pergunta de múltipla escolha baseada em questões do ISTQB; Ribeiro e Paiva (2015), que introduzem o *iLearnTest*, um jogo para o ensino de teste que explora as matérias a serem ensinadas por meio de diversas atividades (e.g: jogo da forca, pegar conceitos corretos caindo da tela, etc) e que objetiva treinar os utilizadores para a obtenção da certificação base do ISTQB; e Silva *et al*. (2016), que em seu jogo nomeado *gTest Learning* apresentam diversos questionários ao jogador, focando em testes de software com base na certificação *foundation* do ISTQB.

Como exemplos de abordagens e jogos que focam em temáticas específicas de teste pode-se mencionar os apresentados por: Costa e Oliveira (2020), que visam o ensino de testes exploratórios por meio de um *framework* gamificado no estilo caça ao tesouro; Buffardi e Valdivia (2018), que apresentam o *Bug Hide-and-Seek*, um jogo para a exposição dos alunos a testes unitários (envolve a inserção de *bugs* em programas e a criação de suítes de testes unitários); Silva (2010), que introduz o *U-TEST*, um jogo de simulação focado em teste de unidade e em técnicas de caixa-preta; Ribeiro *et al*. (2017), que visam apoiar o ensino de teste funcional por meio das três missões que compõem o jogo *BlackBox*; Diniz e Dazzi (2011), que por meio do “Jogo das 7 falhas” igualmente buscam ensinar testes de caixa-preta (nele o jogador precisa identificar 7 falhas em uma determinada funcionalidade); Sherif *et al*. (2020), que apresentam o *CoverBot*, um jogo que objetiva apoiar o ensino de conceitos de cobertura de testes (nele a sobrevivência do jogador depende de sua habilidade em executar todas as linhas de códigos fornecidos com a menor quantidade de entradas possível); Farias *et al*. (2012), que no jogo *iTestLearning* focam na atividade de planejamento de testes (nele o jogador deve realizar o planejamento de teste de software a partir da especificação de um projeto); e Rojas e Fraser (2016), que apresentam o *Code Defenders*, um jogo no qual os utilizadores

competem sobre um trecho de código, gerando testes e introduzindo defeitos por meio da utilização dos princípios do teste de mutação.

Por sua vez, jogos que abordam uma quantidade maior de conceitos são os apresentados por: Valle (2017), que introduz o jogo *Testing Game*, composto por três níveis e vinte e três fases que contemplam as técnicas de teste funcional e estrutural, bem como o teste baseado em defeitos (especificamente o teste de mutação); e Macêdo (2014), que no jogo “As aventuras de Jack Test” apresenta 4 fases (uma para cada um dos seguintes temas: teste unitário, de integração, validação e sistema) e 4 desafios (sobre garantia de qualidade de software, métodos de teste de software, processo de teste de software e depuração).

Alguns jogos simulam contextos empresariais, como os expostos em: Silva, Bernardi e Müller (2011) e Silva (2012), que introduzem o “Jogo da Equipe de Teste de Software (JETS)”, o qual aborda o tópico estratégia de teste e visa simular as interações de uma equipe de teste (em cada fase o jogador assume o papel de um membro da equipe); e Oliveira, Afonso e Costa (2016), que no jogo *TestEG* imergem o jogador em um ambiente empresarial simulado (nele precisam ser respondidas perguntas de múltipla escolha acerca do tema).

Igualmente relacionados ao tema são os trabalhos de: Nascimento (2019), que propôs uma abordagem gamificada para o ensino de testes (principalmente teste funcional); Elbaum *et al.* (2007), que apresentam o *Bug Hunt*, um tutorial web que visa estimular os estudantes a aprenderem sobre estratégias de teste de software (o aluno passa por cinco aulas onde precisa utilizar estratégias de teste específicas para encontrar erros nos programas que lhe são apresentados; aborda conceitos de teste, técnicas de caixa-branca e caixa-preta e testes automatizados); Bell, Sheth e Kaiser (2011), que buscam ensinar a atividade de teste de software por meio da metodologia *Secret Ninja*, na qual os jogadores participam de *quests* (individuais ou em grupo), mas sem ter conhecimento do objetivo específico de aprendizagem; e Jesus (2019) que apresenta o *Bug Hunter*, uma

abordagem gamificada para o ensino de teste composta por *quizes*, desafios e outros elementos de gamificação.

Por fim, é importante destacar que a listagem apresentada nesta seção não é exaustiva, ainda existindo outros jogos neste contexto, tais como *Mutation*, *Rings*, *EMVille* (relacionado à análise e classificação de mutantes equivalentes), *Code Hunt* e *Bug Catcher* (relacionados à criação de casos de teste), trabalhos mencionados em Nascimento (2019) e Sherif *et al*. (2020), bem como abordagens adicionais, a exemplo das descritas no estudo de caracterização realizado por Jesus *et al*. (2018).

# 2 LEVANTAMENTO DE REQUISITOS

Conforme mencionado no Capítulo 1, este trabalho consiste no documento de requisitos da proposta de um jogo sério nomeado QA+, o qual busca contribuir com o rol de jogos voltados ao ensino de teste e qualidade de software.

Neste capítulo serão apresentados os resultados da fase de levantamento/elicitação de requisitos da proposta de jogo mencionada.

Inicialmente, são apresentados abaixo 10 requisitos relacionados à proposta:

- REQ01 – O jogo deve permitir que um usuário crie uma conta nele por meio do informe de nome completo (real), nome de usuário, e-mail e senha;
- REQ02 – O jogo deve permitir que um usuário logado mantenha sua conta (possa visualizar e alterar seus dados de cadastro e avatar, bem como excluir sua conta);
- REQ03 – O jogo deve apresentar uma página de login que permita a um usuário previamente cadastrado acessar suas funcionalidades restritas por meio do informe de e-mail e senha;
- REQ04 – O jogo deve apresentar a usuários logados, em sua tela principal, um quadro de líderes no estilo ranking, no qual devem ser expostas em ordem decrescente de pontos a classificação, o avatar, o nome e o total de pontos: do usuário classificado na 1ª posição, do usuário logado, do usuário que antecede o usuário logado no ranking (se houver) e do usuário que sucede o usuário logado no ranking (se houver);
- REQ05 – O jogo deve apresentar uma turma padrão através da qual deve ser possível a qualquer usuário logado acessar unidades de ensino-aprendizagem constituídas por: (a) uma lição (composta por um título, conteúdos em formato multimídia e uma lista de referências); e (b) um conjunto de atividades com correção automática (nomeados

desafios), cuja resolução correta acarreta na atribuição de pontos ao usuário;

- REQ06 – O jogo deve permitir que um usuário logado mantenha novas turmas: possa criar turmas, visualizar e alterar suas unidades de ensino-aprendizagem (podendo inclusive personalizar o conteúdo das lições e desafios) e gerenciar seus inscritos;
- REQ07 – O jogo deve permitir a importação de unidades de ensino-aprendizagem da turma padrão para as turmas criadas pelos usuários;
- REQ08 – O jogo deve permitir que um usuário logado ingresse em turmas criadas por outros usuários;
- REQ09 – O jogo deve permitir que o usuário criador de uma turma visualize estatísticas de desempenho dos usuários inscritos nela: seus totais de pontos na turma criada e na turma padrão do jogo;
- REQ10 – O jogo deve apresentar uma funcionalidade nomeada "Arena", através da qual deve ser possível a usuários logados competir entre si em duelos de cinco minutos envolvendo a resolução de desafios, sendo que ao usuário vencedor devem ser atribuídos pontos de acordo com a quantidade de desafios respondidos corretamente.

Após a definição dos requisitos acima o autor da proposta solicitou para três colegas de trabalho (todos analistas de tecnologia de informação e com formações acadêmicas na área) para que os classificassem de acordo com uma Escala de Likert com os seguintes níveis de concordância e respectivos valores: concordo totalmente (+2), concordo parcialmente (+1), incerto (0), discordo parcialmente (-1) e discordo totalmente (-2).

No Quadro 1 podem ser verificadas as classificações realizadas por cada respondente/avaliador (representadas por seus valores), bem como os somatórios dos valores referentes às avaliações de cada um dos requisitos.

Quadro 1 – Classificações dos requisitos pelos avaliadores

| Requisito | Classificações Avaliador 1 | Classificações Avaliador 2 | Classificações Avaliador 3 | Somatório das Classificações |
|---|---|---|---|---|
| REQ01 | 2 | 2 | 2 | **6** |
| REQ02 | 2 | 2 | 2 | **6** |
| REQ03 | 2 | 2 | 2 | **6** |
| REQ04 | 2 | 1 | 2 | **5** |
| REQ05 | 2 | 2 | 2 | **6** |
| REQ06 | 2 | 2 | 2 | **6** |
| REQ07 | 1 | 1 | 0 | **2** |
| REQ08 | 2 | 2 | 2 | **6** |
| REQ09 | 1 | 1 | 1 | **3** |
| REQ10 | 1 | 2 | 1 | **4** |

Fonte: elaborado pelo autor (2022)

Analisando o Quadro 1 é possível evidenciar que não existem variações extremas acerca da percepção de importância de cada requisito pelos avaliadores (para cada requisito a variação máxima é de um ponto entre as avaliações individuais). Além disso, por meio dos valores da coluna “Somatório das Classificações” é igualmente perceptível quais requisitos são entendidos como mais essenciais/importantes pelos avaliadores de modo geral: REQ1, REQ2, REQ3, REQ5, REQ6 e REQ8 tiveram as classificações mais favoráveis, seguidos de REQ4, REQ10, REQ9 e REQ7, nesta ordem.

# 3 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

No capítulo anterior foram apresentados 10 requisitos relacionados à proposta deste trabalho, mas sem considerar sua caracterização/definição de tipo. Por sua vez, neste capítulo serão detalhados os requisitos relacionados à proposta (tanto os já expostos, bem como novos), mas considerando os seguintes tipos/categorização:

- Requisitos Funcionais (RF): definem as funcionalidades do sistema (o que ele deve fazer e como deve ser seu comportamento em determinadas situações); descrevem as funções necessárias para atender os objetivos do sistema;
- Requisitos Não Funcionais (RNF): definem as propriedades e restrições do sistema; estabelecem condições de como os requisitos funcionais deverão ser implementados;
- Requisitos/Restrições de Design/Projeto (RD): impõem limitações sobre o projeto do sistema ou sobre os processos empregados em sua construção.

No Quadro 2 podem ser visualizados os requisitos funcionais da proposta deste trabalho. Para cada requisito são expostos seu código, sua descrição, visibilidade, prioridade e RNFs associados (cujo detalhamento será apresentado no Quadro 3).

Quadro 2 – Requisitos Funcionais

| Código | Descrição | Visibilidade | Prioridade | RNFs Associados |
|---|---|---|---|---|
| RF01 | O jogo deve permitir que um usuário crie uma conta nele por meio do informe de nome completo (real), nome de usuário, e-mail e senha. | Evidente | Essencial | RNF01<br>RNF02<br>RNF04<br>RNF05<br>RNF06<br>RNF07<br>RNF08 |

| | | | | |
|---|---|---|---|---|
| RF02 | O jogo deve permitir que um usuário logado mantenha sua conta (possa visualizar e alterar seus dados de cadastro e avatar, bem como excluir sua conta). | Evidente | Essencial | RNF01<br>RNF02<br>RNF03<br>RNF04<br>RNF05<br>RNF06<br>RNF07<br>RNF08 |
| RF03 | O jogo deve apresentar uma página de login que permita a um usuário previamente cadastrado acessar suas funcionalidades restritas por meio do informe de e-mail e senha. | Evidente | Essencial | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |
| RF04 | O jogo deve apresentar a usuários logados, em sua tela principal, um quadro de líderes no estilo ranking, no qual devem ser expostas em ordem decrescente de pontos a classificação, o avatar, o nome e o total de pontos: do usuário classificado na 1ª posição, do usuário logado, do usuário que antecede o usuário logado no ranking (se houver) e do usuário que sucede o usuário logado no ranking (se houver). | Evidente | Importante | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08<br>RNF09 |
| RF05 | O jogo deve apresentar uma turma padrão através da qual deve ser possível a qualquer usuário logado acessar unidades de ensino-aprendizagem constituídas por: (a) uma lição (composta por um título, conteúdos em formato multimídia e uma lista de referências); e (b) um conjunto de atividades com correção automática (nomeados desafios), cuja resolução correta acarreta na atribuição de pontos ao usuário. | Evidente | Essencial | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08<br>RNF10 |
| RF06 | O jogo deve permitir que um usuário logado mantenha novas turmas: possa criar turmas, visualizar e alterar suas unidades de ensino-aprendizagem (podendo inclusive personalizar o conteúdo das lições e desafios) e gerenciar seus inscritos. | Evidente | Essencial | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |
| RF07 | O jogo deve permitir a importação de unidades de ensino-aprendizagem da turma padrão para as turmas criadas pelos | Evidente | Desejável | RNF01<br>RNF02<br>RNF03<br>RNF06 |

| | | | | |
|---|---|---|---|---|
| | usuários. | | | RNF07<br>RNF08 |
| RF08 | O jogo deve permitir que um usuário logado ingresse em turmas criadas por outros usuários. | Evidente | Essencial | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |
| RF09 | O jogo deve permitir que o usuário criador de uma turma visualize estatísticas de desempenho dos usuários inscritos nela: seus totais de pontos na turma criada e na turma padrão do jogo. | Evidente | Desejável | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |
| RF10 | O jogo deve apresentar uma funcionalidade nomeada "Arena", através da qual deve ser possível a usuários logados competir entre si em duelos de cinco minutos envolvendo a resolução de desafios, sendo que ao usuário vencedor devem ser atribuídos pontos de acordo com a quantidade de desafios respondidos corretamente. | Evidente | Importante | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |

Fonte: elaborado pelo autor (2022)

Por sua vez, no Quadro 3 podem ser visualizados os requisitos não funcionais do jogo, com seus códigos, descrições, categorias e prioridades. Cabe destacar que para a caracterização dos RNFs foi utilizada a classificação apresentada por Sommerville e Sawyer (1997).

Quadro 3 – Requisitos Não Funcionais

| **Código** | **Descrição** | **Categoria** | **Prioridade** |
|---|---|---|---|
| RNF01 | O jogo deve estar disponível para acesso online 24 horas por dia e 7 dias por semana. | Disponibilidade | Importante |
| RNF02 | O jogo deve ser acessível minimamente a partir dos navegadores Google Chrome e Mozilla Firefox. | Portabilidade | Importante |
| RNF03 | O acesso às funcionalidades do jogo (exceto cadastro de usuário e tela de login) deve ser restrito a usuários autenticados. | Segurança | Essencial |

| | | | |
|---|---|---|---|
| RNF04 | O jogo deve armazenar as senhas de usuários criptografadas com o algoritmo/função *hash* MD5. | Segurança | Essencial |
| RNF05 | O jogo deve atender o que se encontra disposto na Lei nº 13.709/2018 (Lei Geral de Proteção de Dados - LGPD). | Legal | Essencial |
| RNF06 | O jogo deve ser de fácil utilização, permitindo que novos usuários o operem com o mínimo de esforço e sem a necessidade de treinamento. | Usabilidade | Essencial |
| RNF07 | O jogo deve ter um design minimalista (simples). | Usabilidade | Essencial |
| RNF08 | O jogo deve ser disponibilizado aos usuários nos idiomas “Português (Brasil)” e “Inglês (EUA)”. | Usabilidade | Desejável |
| RNF09 | O tempo de resposta para a apresentação do quadro de líderes não pode ser superior a 5 segundos. | Desempenho | Importante |
| RNF10 | O tempo de carregamento de uma lição não pode ser superior a 15 segundos. | Desempenho | Importante |

Fonte: elaborado pelo autor (2022)

Conforme pode ser visualizado por meio da coluna “RNFs Associados” do Quadro 2, os RNFs do Quadro 3 estão associados a RFs. Os RNFs de códigos RNF01, RNF02, RNF06, RNF07 e RNF08 estão associados a todos os RFs, pois abordam questões gerais de disponibilidade, portabilidade e usabilidade, sendo aplicáveis a todas as funcionalidades do jogo. Por sua vez, o RNF03 está associado a todos os RFs exceto o RF01, pois ele trata de uma questão de controle de acesso presente em todas as funcionalidades do jogo, exceto na descrita no RF01 (o RF02 foi considerado, pois a validação do RNF03 se relaciona à sua funcionalidade a partir de uma ação do usuário). Já os RNFs de códigos RNF04 e RNF05 foram associados aos RFs de códigos RF01 e RF02, pois tratam de questões legais e de segurança envolvendo informações relacionadas às contas e dados sensíveis dos usuários. Por fim, os RNFs de códigos RNF09 e RNF10 foram associados aos RFs de códigos RF04 e RF05,

respectivamente, pois se relacionam especificamente ao desempenho de funcionalidades descritas nestes requisitos funcionais.

Finalmente, no Quadro 4 podem ser visualizados os requisitos de design (projeto) do jogo (seus códigos, descrições e prioridades).

Quadro 4 – Requisitos de Design (Projeto)

| **Código** | **Descrição** | **Prioridade** |
|---|---|---|
| RD01 | O front-end do jogo deve ser implementado utilizando JavaScript, HTML e CSS. | Importante |
| RD02 | O back-end do jogo deve ser implementado utilizando Python. | Desejável |
| RD03 | O jogo deve utilizar o modelo relacional de banco de dados. | Essencial |

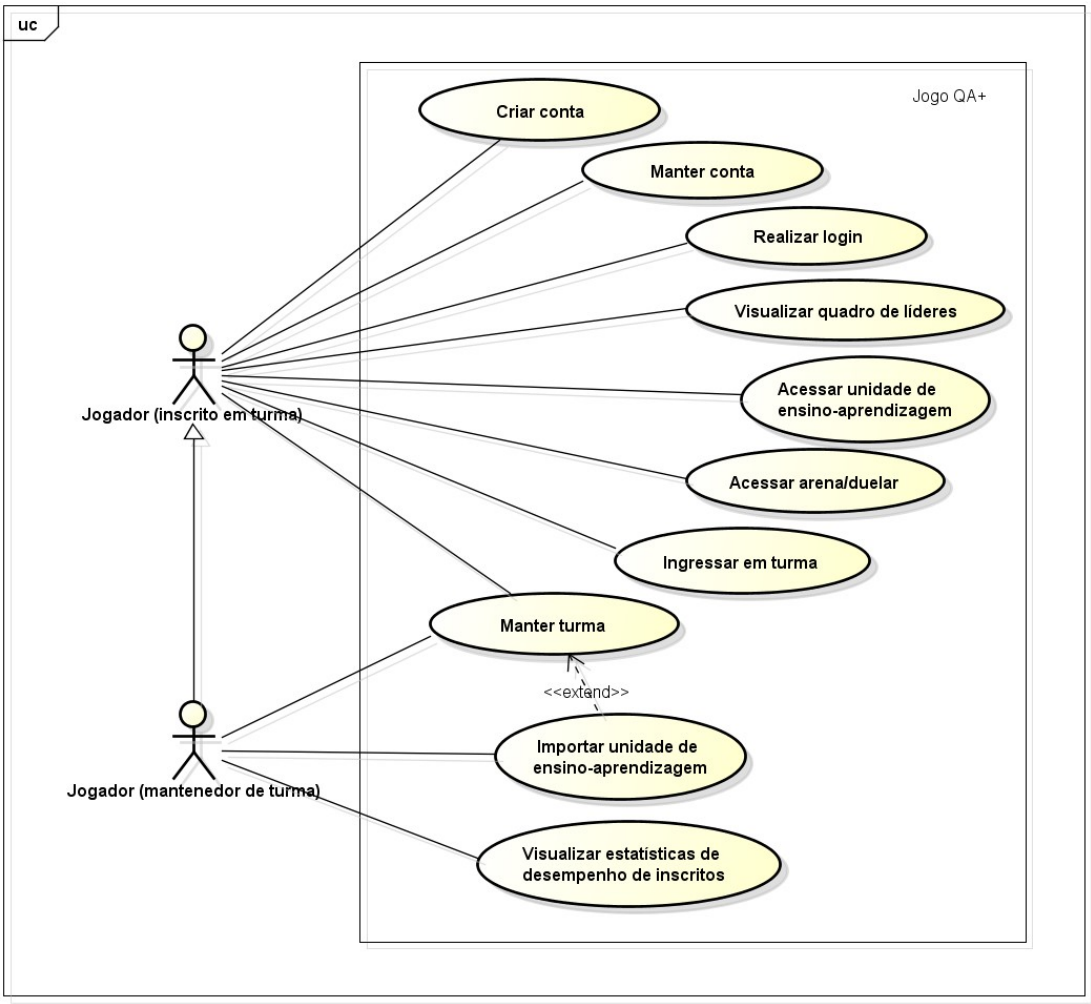
Fonte: elaborado pelo autor (2022)

# 4 DIAGRAMA DE CASOS DE USO

Considerando os requisitos funcionais descritos no Capítulo 3, neste capítulo será trabalhado com o conceito de caso de uso.

Inicialmente, na Figura 1 pode ser visualizado o diagrama de casos de uso do jogo proposto, o qual corresponde a um diagrama da UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada) que pode ser utilizado para a representação dos requisitos, descrevendo o que o sistema deverá fazer.

Figura 1 – Diagrama de casos de uso




Fonte: elaborado pelo autor (2022)

No Quadro 5 podem ser verificados os casos de uso do jogo proposto. Por sua vez, no Quadro 6 pode ser visualizada a relação entre os requisitos funcionais do jogo e seus casos de uso.

Quadro 5 – Casos de Uso

| Código | Descrição | Código | Descrição |
|---|---|---|---|
| UC01 | Criar conta | UC06 | Manter turma |
| UC02 | Manter conta | UC07 | Importar unidade de ensino-aprendizagem |
| UC03 | Realizar login | UC08 | Ingressar em turma |
| UC04 | Visualizar quadro de líderes | UC09 | Visualizar estatísticas de desempenho de inscritos |
| UC05 | Acessar unidade de ensino-aprendizagem | UC10 | Acessar arena/duelar |

Fonte: elaborado pelo autor (2022)

Quadro 6 – Relação entre Requisitos Funcionais e Casos de Uso

| RF / UC | RF01 | RF02 | RF03 | RF04 | RF05 | RF06 | RF07 | RF08 | RF09 | RF10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **UC01** | X | | | | | | | | | |
| **UC02** | | X | | | | | | | | |
| **UC03** | | | X | | | | | | | |
| **UC04** | | | | X | | | | | | |
| **UC05** | | | | | X | | | | | |
| **UC06** | | | | | | X | | | | |
| **UC07** | | | | | | | X | | | |
| **UC08** | | | | | | | | X | | |
| **UC09** | | | | | | | | | X | |
| **UC10** | | | | | | | | | | X |

Fonte: elaborado pelo autor (2022)

Tendo conhecimento dos casos de uso criados, nos Quadros 7 a 16 serão apresentadas suas especificações completas.

Quadro 7 – UC01: Criar conta

| **Nome** | UC01: Criar conta |
|---|---|
| **Atores** | Ator principal: jogador (inscrito em turma) |
| **Descrição** | Caso de uso executado quando o jogador necessita criar uma conta no jogo. |
| **Pré-condições** | - Ter acessado o site do jogo. |
| **Pós-condições** | - Conta criada. |
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador clica em “Criar Conta”. | |
| | 2 – Exibe o formulário de criação de conta com os campos “Nome completo”, “Nome de usuário”, “E-mail” e “Senha” limpos. |
| 3 – Jogador preenche o formulário. | |
| 4 – Jogador clica em “Criar”. | |
| | 5 – Verifica as informações conforme a Regra de Negócio RN01. Caso a regra não seja totalmente atendida vai para a ação 5a do “Fluxo de Exceção”. |
| | 6 – Registra as informações no banco de dados. |
| | 7 – Apresenta a mensagem “Conta criada com sucesso”. |
| **Regras de Negócio** | |
| [RN01] Os campos “Nome completo”, “Nome de usuário”, “E-mail” e “Senha” são obrigatórios. | |
| **Fluxo Alternativo 1** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador clica em “Cancelar”, fecha a página do jogo ou clica na opção “Voltar” do navegador. | |
| | 2 – Desconsidera as informações digitadas e encerra o caso de uso. |
| **Fluxo Alternativo 2** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador recarrega a página do jogo. | |

| | |
|---|---|
| | 2 – Desconsidera as informações digitadas e vai para a ação 2 do “Fluxo Básico”. |
| **Fluxo de Exceção** | |
| **Ações dos atores** | **Ações do sistema** |
| | 5a – Caso a RN01 não seja atendida apresenta a mensagem “Existem campos obrigatórios não preenchidos”, a cor da borda dos campos não preenchidos é alterada para vermelho e volta para a ação 3 do “Fluxo Básico”. |

Fonte: elaborado pelo autor (2022)

Quadro 8 – UC02: Manter conta

| **Nome** | UC02: Manter conta |
|---|---|
| **Atores** | Ator principal: jogador (inscrito em turma) |
| **Descrição** | Caso de uso executado quando o jogador necessita manter sua conta. |
| **Pré-condições** | - Estar logado no jogo. |
| **Pós-condições** | - Conta mantida (visualizada, alterada ou excluída). |
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador clica em “Perfil/Conta”. | |
| | 2 – Exibe as informações da conta do jogador ( “Nome completo”, “Nome de usuário”, “E-mail”, “Senha” e avatar) e botão “Excluir conta”. |
| 3 – Jogador realiza a manutenção da conta. | |
| | 4 - Verifica o tipo de solicitação de manutenção. Se for uma exclusão de conta vai para a ação 1 do “Fluxo Alternativo 1”. Senão, segue o fluxo básico. |
| | 5 - Verifica as informações conforme a Regra de Negócio RN01. Caso a regra não seja totalmente atendida vai para a ação 5a do “Fluxo de Exceção”. |
| | 6 – Atualiza as informações no banco de dados. |
| | 7 – Apresenta a mensagem “Conta mantida com sucesso”. |
| **Regras de Negócio** | |
| [RN01] Os campos “Nome completo”, “Nome de usuário”, “E-mail” e “Senha” são obrigatórios. | |

| **Fluxo Alternativo 1** | |
|---|---|
| **Ações dos atores** | **Ações do sistema** |
| | 1 – Apresenta a mensagem “Deseja realmente excluir sua conta? Esta ação é irreversível” e as opções “Sim” e “Não”. |
| 2 – Jogador escolhe uma das opções. | |
| | 3 – Se a opção escolhida for “Não”, desconsidera as alterações realizadas e vai para a ação 2 do “Fluxo Básico”. Se a opção escolhida for “Sim” remove os dados do jogador do banco de dados e o redireciona para a página de login do jogo. |
| **Fluxo Alternativo 2** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador clica em “Cancelar”, fecha a página do jogo ou clica na opção “Voltar” do navegador. | |
| | 2 – Desconsidera as manutenções realizadas e encerra o caso de uso. |
| **Fluxo Alternativo 3** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador recarrega a página do jogo. | |
| | 2 – Desconsidera as manutenções realizadas e vai para a ação 2 do “Fluxo Básico”. |
| **Fluxo de Exceção** | |
| **Ações dos atores** | **Ações do sistema** |
| | 5a – Caso a RN01 não seja atendida apresenta a mensagem “Existem campos obrigatórios não preenchidos”, a cor da borda dos campos não preenchidos é alterada para vermelho e volta para a ação 3 do “Fluxo Básico”. |

Fonte: elaborado pelo autor (2022)

Quadro 9 – UC03: Realizar login

| | |
|---|---|
| **Nome** | UC03: Realizar login |
| **Atores** | Ator principal: jogador (inscrito em turma) |
| **Descrição** | Caso de uso executado quando o jogador necessita realizar login no jogo. |
| **Pré-condições** | - Ter acessado o site do jogo. |
| **Pós-condições** | - Jogador logado no jogo/ redirecionado para sua tela principal. |

| **Fluxo Básico** | |
|---|---|
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador clica em “Entrar”. | |
| | 2 – Exibe o formulário de login com os campos “E-mail” e “Senha” limpos. |
| 3 – Jogador preenche o formulário. | |
| 4 – Jogador clica em “Entrar”. | |
| | 5 – Verifica as informações inseridas pelo jogador no formulário conforme as Regras de Negócio RN02, RN03 e RN04. Caso RN02 não seja atendida vai para a ação 5a do “Fluxo de Exceção”. Caso RN03 ou RN04 não sejam atendidas vai para a ação 5b do “Fluxo de Exceção”. |
| | 6 – Realiza o login. |
| | 7 – Apresenta a página/tela principal do jogo. |
| **Regras de Negócio** | |
| [RN02] Os campos “E-mail” e “Senha” são obrigatórios.<br>[RN03] O e-mail informado para login deve existir no banco de dados.<br>[RN04] A senha informada para login deve ser igual à senha existente no banco de dados para o jogador associado ao e-mail informado. | |
| **Fluxo Alternativo 1** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador fecha a página do jogo ou clica na opção “Voltar” do navegador. | |
| | 2 – Desconsidera as informações inseridas e encerra o caso de uso. |
| **Fluxo Alternativo 2** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador recarrega a página do jogo. | |
| | 2 – Desconsidera as informações inseridas e vai para a ação 2 do “Fluxo Básico”. |
| **Fluxo de Exceção** | |
| **Ações dos atores** | **Ações do sistema** |
| | 5a – Caso a RN02 não seja atendida apresenta a mensagem “Existem campos obrigatórios não preenchidos”, a cor da borda dos campos não preenchidos é alterada para vermelho e volta para a ação 3 do “Fluxo Básico”. |

| | |
|---|---|
| | 5b – Caso a RN03 ou a RN04 não sejam atendidas apresenta a mensagem “Alguma informação está incorreta. Tente novamente” e volta para a ação 3 do “Fluxo Básico”. |

Fonte: elaborado pelo autor (2022)

Quadro 10 – UC04: Visualizar quadro de líderes

| **Nome** | UC04: Visualizar quadro de líderes |
|---|---|
| **Atores** | Ator principal: jogador (inscrito em turma) |
| **Descrição** | Caso de uso executado quando o jogador acessa a tela principal do jogo. |
| **Pré-condições** | - Estar logado no jogo. |
| **Pós-condições** | - Quadro de líderes renderizado. |
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador clica em “Lições” (link para a página principal do jogo) ou acessa esta página em decorrência da ação 7 do “Fluxo Básico” do UC03. | |
| | 2 – Apresenta na parte superior da tela o quadro de líderes conforme as Regras de Negócio RN05, RN06 e RN07. |
| **Regras de Negócio** | |
| [RN05] Os registros do quadro de líderes devem ser exibidos em ordem decrescente de pontos.<br>[RN06] Devem ser apresentadas no quadro de líderes as seguintes informações de um jogador: sua classificação, avatar, nome e total de pontos.<br>[RN07] Devem ser apresentados no quadro de líderes os seguintes jogadores: jogador que ocupa a 1ª posição no ranking, jogador logado, jogador que antecede o jogador logado no ranking (se houver) e jogador que sucede o jogador logado no ranking (se houver). | |

Fonte: elaborado pelo autor (2022)

Quadro 11 – UC05: Acessar unidade de ensino-aprendizagem

| **Nome** | UC05: Acessar unidade de ensino-aprendizagem |
|---|---|
| **Atores** | Ator principal: jogador (inscrito em turma) |
| **Descrição** | Caso de uso executado quando o jogador necessita acessar uma unidade de ensino-aprendizagem. |
| **Pré-condições** | - Estar logado no jogo. |

| **Pós-condições** | - Unidade de ensino-aprendizagem com situação atualizada de acordo com o desempenho do jogador. |
|---|---|
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador clica no ícone da unidade de ensino-aprendizagem. | |
| | 2 – Exibe a lição relacionada à unidade (título, conteúdos em formato multimídia e lista de referências) e um botão nomeado “Desafios”. |
| 3 – Jogador clica em “Desafios”. | |
| | 4 – Zera o saldo temporário de pontos do jogador para a unidade de ensino-aprendizagem em questão. |
| | 5 – Exibe uma atividade/desafio relacionado ao conteúdo da lição. |
| 6 – Jogador seleciona resposta e clica em “Verificar”. | |
| | 7 – Verifica a corretude da resposta do jogador e atualiza seu saldo temporário de pontos para a unidade com base na Regra de Negócio RN08. |
| | 8 – Exibe feedback acerca da resposta (se o jogador respondeu o desafio corretamente ou não e qual é a resposta correta), bem como um botão “Continuar”. |
| 9 – Jogador clica em “Continuar”. | |
| | 10 – Se ainda existirem atividades não respondidas para a unidade vai para a ação 5. Senão, vai para a ação 11. |
| | 11 – Verifica se a unidade pode ser considerada concluída com base na Regra de Negócio RN09. Caso a unidade não possa ser considerada concluída vai para a ação 1 do “Fluxo Alternativo 1”. |
| | 12 – Atualiza as estatísticas de desempenho do jogador no banco de dados, somando a pontuação ganha na unidade (saldo temporário calculado com base na RN08) ao seu saldo global de pontos. |
| | 13 – Atualiza a situação da unidade de ensino-aprendizagem para “Concluída”. |
| | 14 – Exibe a mensagem “Unidade concluída com sucesso!”, a pontuação ganha pelo jogador na unidade e um botão “Continuar”. |

| | |
|---|---|
| 15 – Jogador clica em “Continuar”. | |
| | 16 – Redireciona o jogador para a página inicial do jogo. |
| **Regras de Negócio** | |
| [RN08] O saldo temporário de pontos de um jogador para cada unidade de ensino-aprendizagem é inicialmente igual à zero. A cada desafio respondido corretamente são acrescidos dois pontos ao saldo temporário do jogador. Desafios respondidos incorretamente não alteram o saldo de pontos do jogador.<br>[RN09] Uma unidade somente pode ter sua situação alterada para “Concluída” caso o jogador tenha acertado no mínimo 70% dos seus desafios. | |
| **Fluxo Alternativo 1** | |
| **Ações dos atores** | **Ações do sistema** |
| | 1 – Exibe a mensagem “Não foi desta vez! Você não atingiu a pontuação mínima de 70% de acertos para concluir a unidade. Quem sabe na próxima vez?” e um botão “Continuar”. |
| 2 – Jogador clica em “Continuar”. | |
| | 3 – Redireciona o jogador para a página inicial do jogo. |
| **Fluxo Alternativo 2** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador clica no ícone de fechar da unidade, fecha a página do jogo ou clica na opção “Voltar” do navegador. | |
| | 2 – Desconsidera o progresso do jogador na unidade e encerra o caso de uso. Ainda, se a ação tiver sido diferente de fechar a página do jogo, redireciona o jogador para a página principal do jogo. |
| **Fluxo Alternativo 3** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador recarrega a página do jogo. | |
| | 2 – Desconsidera o progresso do jogador na unidade e o redireciona para a página principal do jogo. |

Fonte: elaborado pelo autor (2022)

Quadro 12 – UC06: Manter turma

| | |
|---|---|
| **Nome** | UC06: Manter turma |

| **Atores** | Ator principal: jogador (inscrito em turma), jogador (mantenedor de turma) |
|---|---|
| **Descrição** | Caso de uso executado quando o jogador necessita manter uma turma própria. |
| **Pré-condições** | - Estar logado no jogo. |
| **Pós-condições** | - Turma mantida (criada, visualizada ou alterada). |
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador clica em “Turmas”. | |
| | 2 – Exibe um botão “Criar Turma” e uma listagem das turmas existentes. Para cada turma existente apresenta uma imagem, o nome da turma e o nome de usuário de seu jogador mantenedor. Ao lado de cada turma mantida pelo jogador logado é apresentado um botão nomeado “Gerenciar”. Ao lado de cada turma não mantida pelo jogador logado é apresentado o botão “Ingressar”. |
| 3 – Jogador clica em “Criar Turma” ou em “Gerenciar“. Se clicar em “Gerenciar”, vai para a ação 1 do “Fluxo Alternativo 3”. Senão, continua neste fluxo. | |
| | 4 – Exibe o formulário de criação de turma com um campo do tipo texto para informe de “Nome da turma”, uma opção para upload de imagem para identificação da turma, recursos para a montagem das unidades de ensino-aprendizagem e uma opção nomeada “Importar Unidades QA+”. |
| 5 – Jogador preenche o formulário. Caso clique na opção “Importar Unidades QA+”, vai para a ação 2 do “Fluxo Básico” do UC07. | |
| 6 – Jogador clica em “Salvar”. | |
| | 7 – Verifica as informações conforme a Regra de Negócio RN10. Caso a regra não seja totalmente atendida vai para a ação 7a do “Fluxo de Exceção”. |
| | 8 – Registra as informações no banco de dados. |
| | 9 – Apresenta a mensagem “Turma criada com sucesso”. |
| **Regras de Negócio** | |
| [RN10] O campo “Nome da turma” é obrigatório. | |

| **Fluxo Alternativo 1** | |
|---|---|
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador clica em “Cancelar”, fecha a página do jogo ou clica na opção “Voltar” do navegador. | |
| | 2 – Desconsidera as manutenções realizadas e encerra o caso de uso. |
| **Fluxo Alternativo 2** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador recarrega a página do jogo. | |
| | 2 – Desconsidera as manutenções realizadas e vai para a ação 2 do “Fluxo Básico”. |
| **Fluxo Alternativo 3** | |
| **Ações dos atores** | **Ações do sistema** |
| | 1 – Exibe as informações da turma. |
| 2 – Jogador realiza a manutenção das informações da turma e clica em “Salvar”. | |
| | 3 – Verifica as informações conforme a Regra de Negócio RN10. Caso a regra não seja totalmente atendida vai para a ação 7b do “Fluxo de Exceção”. |
| | 4 – Atualiza as informações da turma no banco de dados. |
| | 5 – Apresenta a mensagem “Turma mantida com sucesso”. |
| **Fluxo de Exceção** | |
| **Ações dos atores** | **Ações do sistema** |
| | 7a – Caso a RN10 não seja atendida apresenta a mensagem “Existem campos obrigatórios não preenchidos”, a cor da borda dos campos não preenchidos é alterada para vermelho e volta para a ação 5 do “Fluxo Básico”. |
| | 7b – Caso a RN10 não seja atendida apresenta a mensagem “Existem campos obrigatórios não preenchidos”, a cor da borda dos campos não preenchidos é alterada para vermelho e volta para a ação 2 do “Fluxo Alternativo 3”. |

Fonte: elaborado pelo autor (2022)

Quadro 13 – UC07: Importar unidade de ensino-aprendizagem

| **Nome** | UC07: Importar unidade de ensino-aprendizagem |
|---|---|
| **Atores** | Ator principal: jogador (inscrito em turma), jogador (mantenedor de turma) |
| **Descrição** | Caso de uso executado quando o jogar necessita importar uma unidade de ensino-aprendizagem para uma turma sendo criada ou já criada por ele. |
| **Pré-condições** | - Estar logado no jogo.<br>- Estar criando uma nova turma OU estar alterando uma turma própria já criada. |
| **Pós-condições** | - Unidade de ensino-aprendizagem importada para a turma sendo mantida. |
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador clica em “Importar Unidades QA+”. | |
| | 2 – Apresenta a estrutura de unidades de ensino-aprendizagem da turma padrão do jogo e caixas para seleção das unidades a serem importadas. |
| 3 – Jogador seleciona as unidades a serem importadas e clica em “Importar”. | |
| | 4 – Importa as unidades selecionadas para a estrutura da turma em questão. |
| | 5 – Apresenta a mensagem “Importação realizada com sucesso!” e o botão “OK”. |
| 6 – Jogador clica em “OK”. | |
| | 7 – Apresenta tela de manutenção da turma. |
| **Fluxo Alternativo 1** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador fecha a página do jogo ou clica na opção “Voltar” do navegador. | |
| | 2 – Desconsidera as manipulações realizadas pelo jogador e encerra o caso de uso. |
| **Fluxo Alternativo 2** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador recarrega a página do jogo. | |

| | |
|---|---|
| | 2 – Desconsidera as manipulações realizadas pelo jogador e vai para a ação 2 do “Fluxo Básico”. |

Fonte: elaborado pelo autor (2022)

Quadro 14 – UC08: Ingressar em turma

| **Nome** | UC08: Ingressar em turma |
|---|---|
| **Atores** | Ator principal: jogador (inscrito em turma) |
| **Descrição** | Caso de uso executado quando o jogador necessita ingressar em uma turma criada por outro jogador. |
| **Pré-condições** | - Estar logado no jogo.<br>- Existir uma turma criada por outro jogador. |
| **Pós-condições** | - Jogador inscrito na turma. |
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador clica em “Turmas”. | |
| | 2 – Exibe um botão “Criar Turma” e uma listagem das turmas existentes. Para cada turma existente apresenta uma imagem, o nome da turma e o nome de usuário de seu jogador mantenedor. Ao lado de cada turma mantida pelo jogador logado é apresentado um botão nomeado “Gerenciar”. Ao lado de cada turma não mantida pelo jogador logado é apresentado o botão “Ingressar”. |
| 3 – Jogador clica em “Ingressar”. | |
| | 4 – Registra o jogador (no banco de dados) como ingressante na turma em questão. |
| | 5 – Exibe a página principal da turma ingressada pelo jogador. |

Fonte: elaborado pelo autor (2022)

Quadro 15 – UC09: Visualizar estatísticas de desempenho de inscritos

| **Nome** | UC09: Visualizar estatísticas de desempenho de inscritos |
|---|---|
| **Atores** | Ator principal: jogador (mantenedor de turma) |
| **Descrição** | Caso de uso executado quando o jogador (mantenedor de turma) necessita visualizar estatísticas de desempenho dos jogadores inscritos em uma turma criada por ele. |

| | |
|---|---|
| **Pré-condições** | - Estar logado no jogo.<br>- Ser mantenedor de uma turma.<br>- Ter acessado a tela de gerenciamento de uma turma. |
| **Pós-condições** | - Estatísticas de desempenho de inscritos renderizadas. |
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador mantenedor de turma clica em “Visualizar estatísticas de inscritos”. | |
| | 2 – Exibe uma grid com os nomes completos e nomes de usuário de cada jogador inscrito na turma, bem como seus totais de pontos na turma em questão, na turma padrão do jogo e total geral de pontos. |

Fonte: elaborado pelo autor (2022)

Quadro 16 – UC10: Acessar arena/duelar

| | |
|---|---|
| **Nome** | UC10: Acessar arena/duelar |
| **Atores** | Ator principal: jogador (inscrito em turma) |
| **Descrição** | Caso de uso executado quando o jogador necessita acessar a funcionalidade “Arena” /duelar. |
| **Pré-condições** | - Estar logado no jogo. |
| **Pós-condições** | - Estatísticas dos jogadores participantes do duelo atualizadas. |
| **Fluxo Básico** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – Jogador acessa a funcionalidade “Arena”. | |
| | 2 - Exibe informações acerca da funcionalidade (regras de um duelo) e o botão “Começar Duelo”. |
| 3 – Jogador clica em “Começar Duelo”. | |
| | 4 – Apresenta a mensagem “Buscando por outro jogador para iniciar o duelo...”. |
| | 5 – Busca por outro jogador aguardando o início de um duelo. Ao encontrar, vai para a ação 6. |
| | 6 – Define o saldo temporário de pontos dos jogadores no duelo como 5 e define o tempo do duelo com base na Regra de Negócio RN11. |

| | |
|---|---|
| | 7 – Exibe um desafio para cada jogador e começa a decrementar o tempo definido pela RN11. |
| 8 – Jogador seleciona resposta e clica em “Verificar”. | |
| | 9 – Verifica a corretude da resposta do jogador e atualiza seu saldo temporário de pontos para a unidade com base na Regra de Negócio RN12. |
| | 10 – Exibe feedback acerca da resposta (se o jogador respondeu o desafio corretamente ou não e qual é a resposta correta), bem como um botão “Continuar”. |
| 11 – Jogador clica em “Continuar”. | |
| | 12 – Se o tempo definido na RN11 ainda não foi atingido vai para a ação 7. Senão, vai para a ação 13. |
| | 13 – Define o(s) vencedor(es) do duelo com base na RN13. |
| | 14 – Atualiza as estatísticas de desempenho do(s) jogador(es) vencedor(es) no banco de dados, somando sua pontuação final (saldo temporário) no duelo ao seu saldo global de pontos. |
| | 15 – Exibe a mensagem “Duelo finalizado!”, o(s) vencedor(es), a pontuação ganha e um botão “Continuar”. |
| 16 – Jogador clica em “Continuar”. | |
| | 17 – Redireciona o jogador para a funcionalidade “Arena”. |
| **Regras de Negócio** | |
| [RN11] O tempo de cada duelo é de 5 minutos.<br>[RN12] Em duelos, cada desafio respondido corretamente pelo jogador acarreta no incremento de seu saldo temporário de pontos em 1 ponto. Cada desafio respondido incorretamente pelo jogador acarreta no decremento de seu saldo temporário de pontos em 1 ponto.<br>[RN13] O vencedor de um duelo é o que, ao final do tempo definido na RN11, apresentar maior saldo temporário de pontos. Em caso de empate ambos os jogadores são considerados vencedores. | |
| **Fluxo Alternativo 1** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador fecha a página do duelo, do jogo ou clica na opção “Voltar” do navegador. | |

| | |
|---|---|
| | 2 – Desconsidera as manipulações realizadas pelo jogador, define o jogador adversário como vencedor e encerra o caso de uso. |
| **Fluxo Alternativo 2** | |
| **Ações dos atores** | **Ações do sistema** |
| 1 – A qualquer momento o jogador recarrega a página do jogo. | |
| | 2 – Desconsidera as manipulações realizadas pelo jogador, define o jogador adversário como vencedor e vai para a ação 2 do “Fluxo Básico”. |

Fonte: elaborado pelo autor (2022)

# 5 RASTREAMENTO DE REQUISITOS

Considerando as definições de requisitos e casos de uso expostas nos capítulos anteriores, neste capítulo será apresentado o rastreamento dos requisitos do projeto, que, segundo Gotel e Frinkelstein (1994), se refere à habilidade de descrever e acompanhar um requisito desde sua origem (bem como desenvolvimento e especificação) até sua distribuição e utilização. Complementarmente, Weringa (1995) define duas técnicas de rastreamento:

- Rastreamento para trás: relaciona-se ao rastreamento de um requisito em direção à sua fonte (seja esta uma pessoa, instituição, equipamento, sistema, etc); e
- Rastreamento para frente: relaciona-se ao rastreamento de um requisito em direção aos componentes de design ou à implementação.

Sabendo disso, nos Quadros 17 e 18 podem ser visualizadas as matrizes de rastreamento para trás relacionando os RFs e RNFs do projeto com as partes envolvidas/atores do sistema, respectivamente.

Quadro 17 – Matriz de rastreamento para trás (RFs x atores)

| RF / Ator | RF01 | RF02 | RF03 | RF04 | RF05 | RF06 | RF07 | RF08 | RF09 | RF10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Jogador (inscrito em turma)** | X | X | X | X | X | X | | X | | X |
| **Jogador (mantenedor de turma)** | | | | | | X | X | | X | |

Fonte: elaborado pelo autor (2022)

Quadro 18 – Matriz de rastreamento para trás (RNFs x atores)

| RNF / Ator | RNF01 | RNF02 | RNF03 | RNF04 | RNF05 | RNF06 | RNF07 | RNF08 | RNF09 | RNF10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Jogador (inscrito em turma)** | X | X | X | X | X | X | X | X | X | X |
| **Jogador (mantenedor de turma)** | X | X | X | | | X | X | X | | |

Fonte: elaborado pelo autor (2022)

Por sua vez, nos Quadros 19 e 20 podem ser encontradas as matrizes de rastreamento para frente relacionando os RFs e RNFs do projeto com os casos de uso do sistema, respectivamente.

Quadro 19 – Matriz de rastreamento para frente (RFs x UCs)

| RF / UC | RF01 | RF02 | RF03 | RF04 | RF05 | RF06 | RF07 | RF08 | RF09 | RF10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **UC01** | X | | | | | | | | | |
| **UC02** | | X | | | | | | | | |
| **UC03** | | | X | | | | | | | |
| **UC04** | | | | X | | | | | | |
| **UC05** | | | | | X | | | | | |
| **UC06** | | | | | | X | | | | |
| **UC07** | | | | | | | X | | | |
| **UC08** | | | | | | | | X | | |
| **UC09** | | | | | | | | | X | |
| **UC10** | | | | | | | | | | X |

Fonte: elaborado pelo autor (2022)

Quadro 20 – Matriz de rastreamento para frente (RNFs x UCs)

| RNF / UC | RNF01 | RNF02 | RNF03 | RNF04 | RNF05 | RNF06 | RNF07 | RNF08 | RNF09 | RNF10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **UC01** | X | X | | X | X | X | X | X | | |
| **UC02** | X | X | X | X | X | X | X | X | | |
| **UC03** | X | X | X | | | X | X | X | | |
| **UC04** | X | X | X | | | X | X | X | X | |
| **UC05** | X | X | X | | | X | X | X | | X |
| **UC06** | X | X | X | | | X | X | X | | |
| **UC07** | X | X | X | | | X | X | X | | |
| **UC08** | X | X | X | | | X | X | X | | |
| **UC09** | X | X | X | | | X | X | X | | |
| **UC10** | X | X | X | | | X | X | X | | |

Fonte: elaborado pelo autor (2022)

# 6 VERIFICAÇÃO E VALIDAÇÃO DE REQUISITOS

Neste capítulo serão abordadas duas fases muito importantes da Engenharia de Requisitos, conhecidas como Verificação e Validação. Em síntese, a verificação busca averiguar se o produto está sendo construído corretamente, enquanto a validação busca averiguar se o produto correto está sendo construído, ou seja, se ele realmente representa as necessidades/desejos de seus clientes e usuários.

A verificação de requisitos é baseada nos problemas mais comuns que podem surgir durante o levantamento de requisitos, geralmente relacionados a nove aspectos, sendo eles: validade, consistência, completeza, realismo, verificação, ambiguidade, rastreamento, redundância e conformidade.

Nos Quadros 21 e 22 podem ser visualizadas matrizes de verificação dos RFs e RNFs do projeto, respectivamente. Para sua construção analisou-se cada um dos RFs e RNFs apresentados no Capítulo 3 em busca de problemas relacionados aos aspectos mencionados no parágrafo anterior. A existência do símbolo “X” em uma célula da matriz representa que para o RF ou RNF em questão foi identificado um problema (detalhados a seguir).

Quadro 21 – Matriz de verificação dos RFs

| RF / Problema | RF 01 | RF 02 | RF 03 | RF 04 | RF 05 | RF 06 | RF 07 | RF 08 | RF 09 | RF 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Validade** | | | | | | | | | | |
| **Consistência** | | | | | | | | | | |
| **Completeza** | | | | | | | | | | |
| **Realismo** | | | | | | | | | | |
| **Verificação** | | | | | | | | | | |
| **Ambiguidade** | | | | X | | X | | | | |
| **Rastreamento** | | | | | | | | | | |
| **Redundância** | | | | | | | | | | |
| **Conformidade** | | | | | | | | | | |

Fonte: elaborado pelo autor (2022)

Quadro 22 – Matriz de verificação dos RNFs

| RNF / Problema | RNF 01 | RNF 02 | RNF 03 | RNF 04 | RNF 05 | RNF 06 | RNF 07 | RNF 08 | RNF 09 | RNF 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Validade** | | | | | | | | | | |
| **Consistência** | | | | | | | | | | |
| **Completeza** | | | X | | | | | | | |
| **Realismo** | | | | | | | | | | |
| **Verificação** | | | | | | | | | | |
| **Ambiguidade** | | | | | | | | | | |
| **Rastreamento** | | | | | | | | | | |
| **Redundância** | | | | | | | | | | |
| **Conformidade** | | | | | | | | | | |

Fonte: elaborado pelo autor (2022)

Inicialmente, cabe mencionar que para a análise relacionada ao Quadro 21 foram despendidos 23 minutos. Por sua vez, a análise relacionada ao Quadro 22 levou 15 minutos.

Conforme pode ser evidenciado no Quadro 21, em relação aos RFs foram identificadas duas situações de ambiguidade:

- O RF04 menciona que no quadro de líderes deve ser apresentado um “nome”, mas não define qual: o nome completo da pessoa (real) ou o nome de usuário no jogo, ambos existentes no cadastro da conta. Para eliminar esta ambiguidade a descrição do RF04 será alterada para: “O jogo deve apresentar a usuários logados, em sua tela principal, um quadro de líderes no estilo ranking, no qual devem ser expostas em ordem decrescente de pontos a classificação, o avatar, o nome de usuário e o total de pontos [...]”;
- Ao mencionar "conteúdo das lições" o RF06 pode levar à interpretação de que somente os "conteúdos em formato multimídia" da lição são personalizáveis (termo empregado no RF05). Todavia, todas as informações da lição (título, conteúdos em formato multimídia e lista de referências) devem ser passíveis de personalização. Assim, para

eliminar esta ambiguidade a descrição do RF06 será alterada para: “O jogo deve permitir que um usuário logado mantenha novas turmas: possa criar turmas, visualizar e alterar suas unidades de ensino-aprendizagem (podendo inclusive personalizar suas lições e desafios) e gerenciar seus inscritos”.

Já em relação aos RNFs, por meio do Quadro 22 é possível evidenciar que o RNF03 possui um problema de completeza:

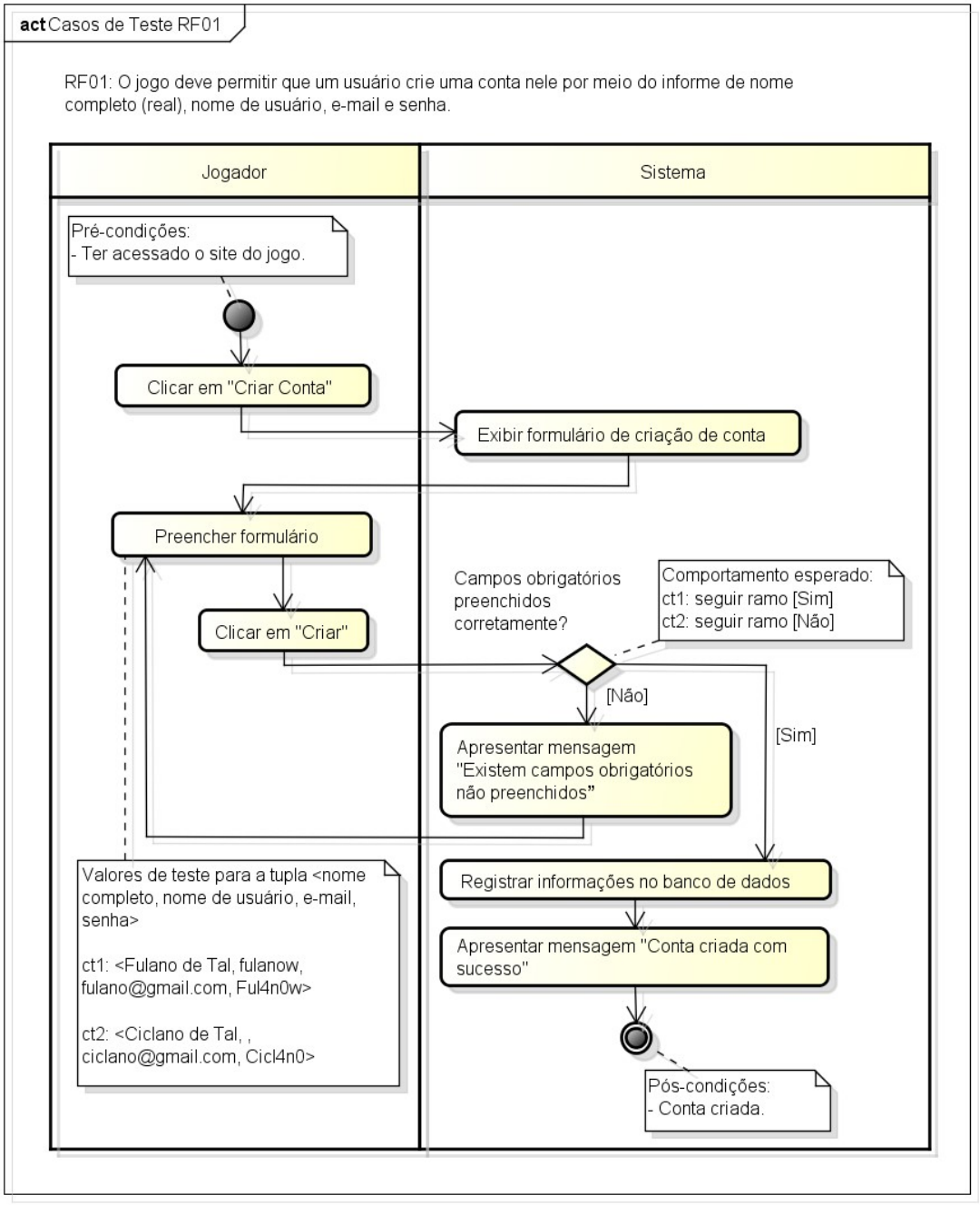
- O RNF03 menciona que o acesso a algumas funcionalidades do jogo deve ser restrito a usuários autenticados, mas não destaca quais são os elementos de autenticação. Assim, de modo sanar este problema a descrição do RNF03 será alterada para: “O acesso às funcionalidades do jogo (exceto cadastro de usuário e tela de login) deve ser restrito a usuários autenticados por meio do informe de e-mail e senha”.

Em relação aos demais aspectos não foram identificados problemas, uma vez que:

- Todos os RFs e RNFs são válidos e todos os usuários são atendidos, não existindo, então, problemas de validade;
- Os requisitos do projeto não são conflitantes, sendo consistentes;
- Considerando questões tecnológicas e recursos, os requisitos podem ser implementados. Logo, não possuem problemas de realismo;
- Os requisitos estão escritos de modo com que podem ser verificados inclusive por clientes. Desta forma, são facilmente verificáveis;
- Os requisitos são facilmente rastreáveis para frente e para trás, sendo suas matrizes de rastreamento apresentadas nos Quadros 17 a 20;
- Um mesmo serviço/funcionalidade não é solicitado em múltiplos requisitos, não havendo redundância; e
- Os requisitos estão escritos de acordo com as normas adotadas (apresentam um padrão de escrita e as informações necessárias) não havendo problemas de conformidade.

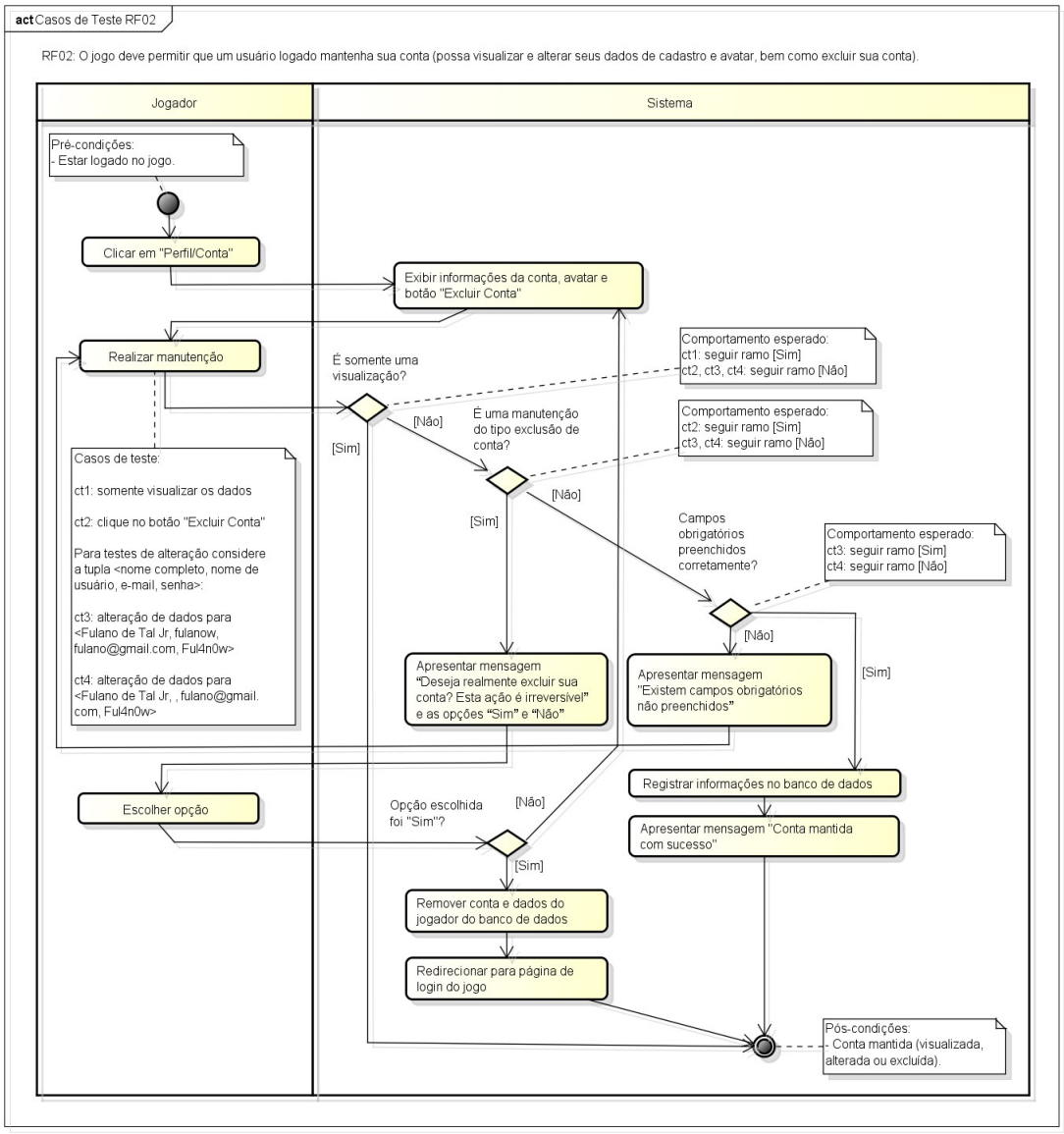
Por sua vez, uma das técnicas comumente utilizadas para a validação de requisitos consiste na geração de casos de teste. Sabendo disso, nas Figuras 2 a 11 serão apresentadas especificações de casos de teste para cada um dos 10 requisitos funcionais do projeto. Cabe destacar que a representação dos casos de teste será realizada graficamente por meio de diagramas de atividade da UML.

Figura 2 – Casos de Teste RF01



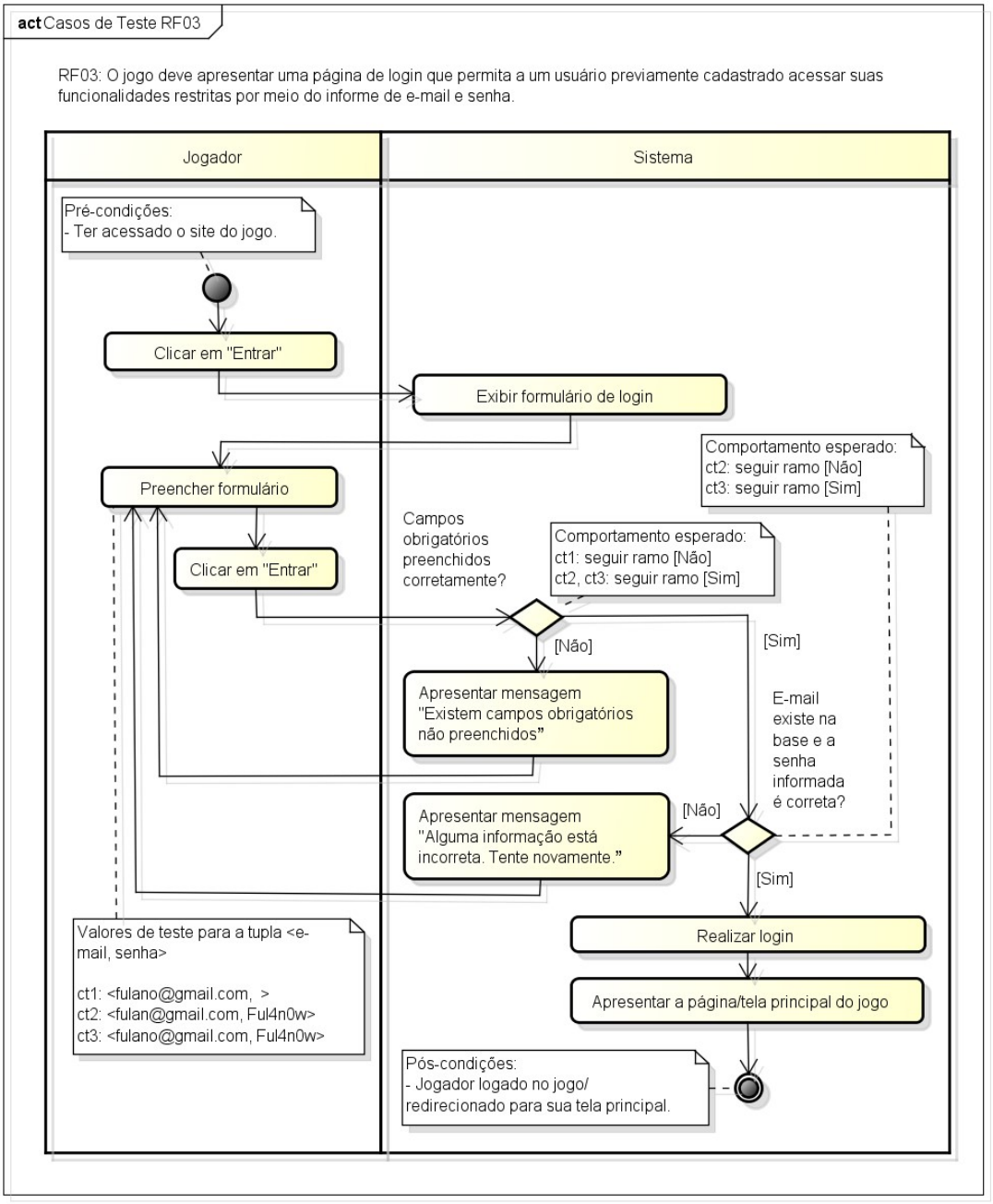

Fonte: elaborado pelo autor (2022)

Figura 3 – Casos de Teste RF02



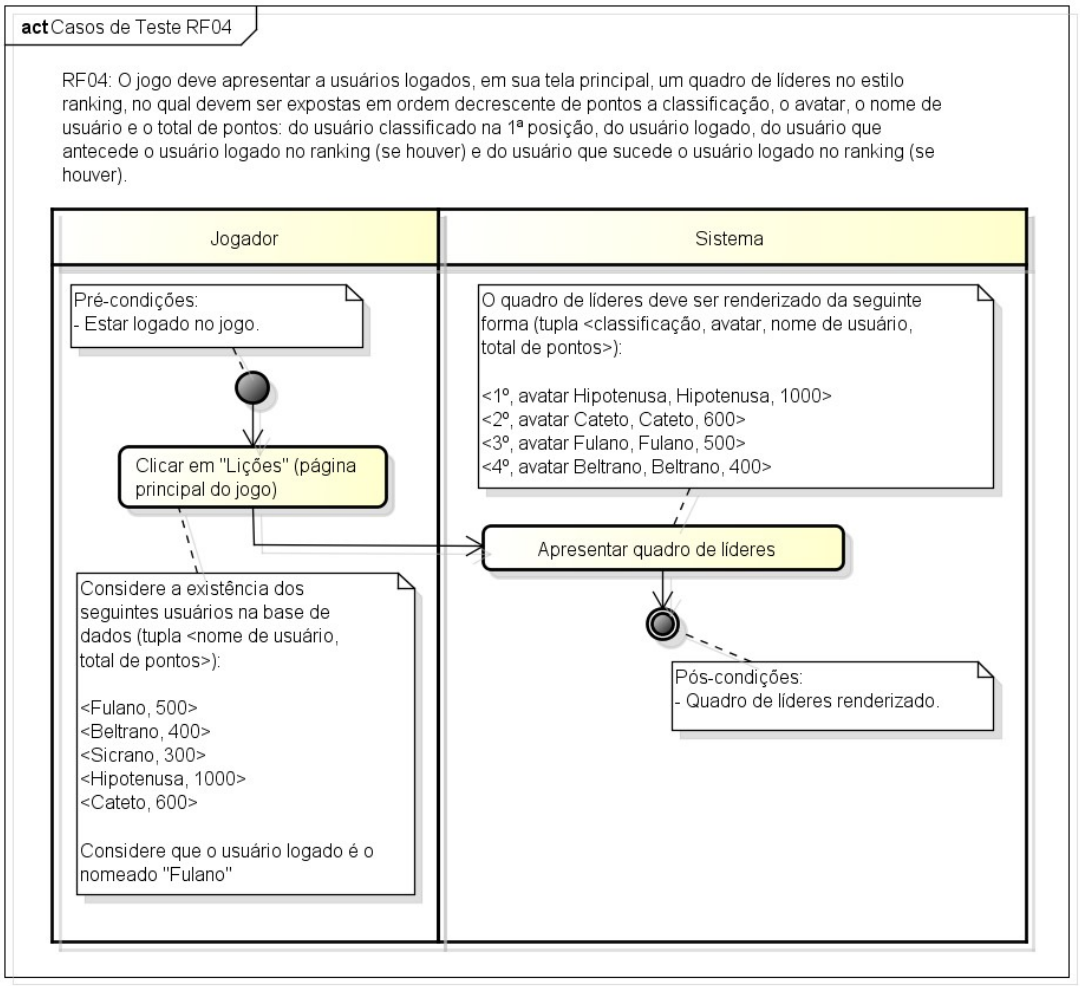

Fonte: elaborado pelo autor (2022)

Figura 4 – Casos de Teste RF03



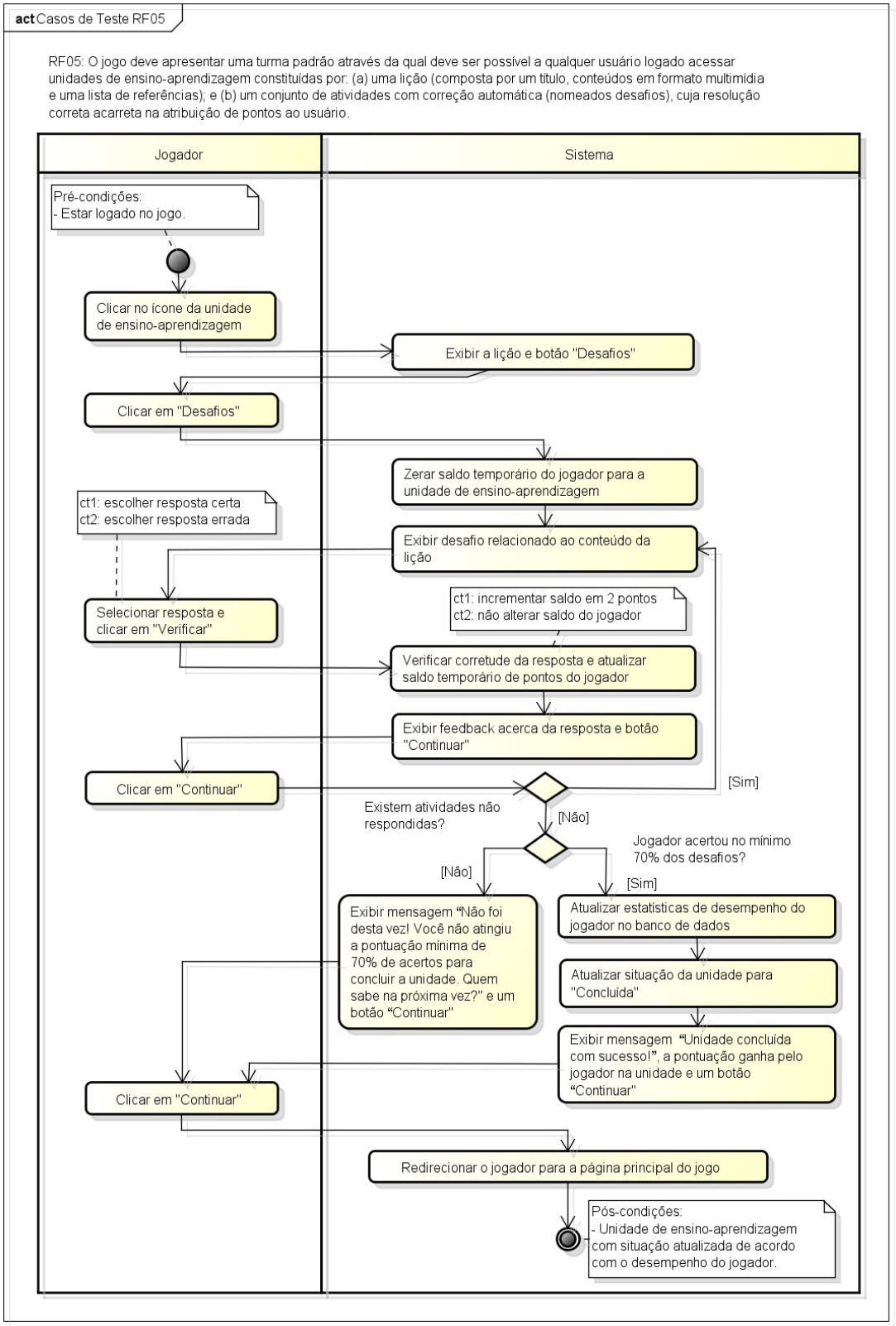

Fonte: elaborado pelo autor (2022)

Figura 5 – Casos de Teste RF04



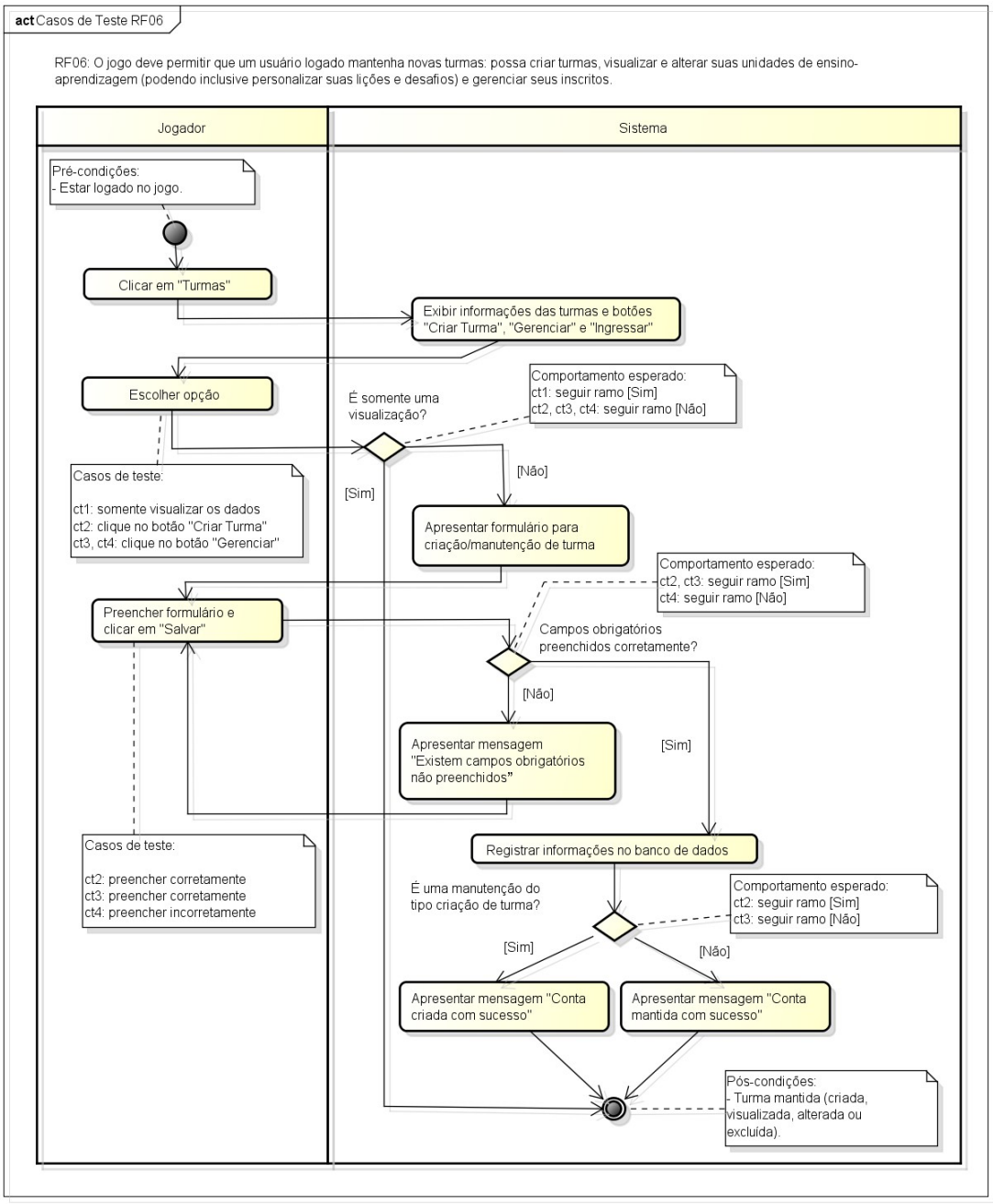

Fonte: elaborado pelo autor (2022)

Figura 6 – Casos de Teste RF05



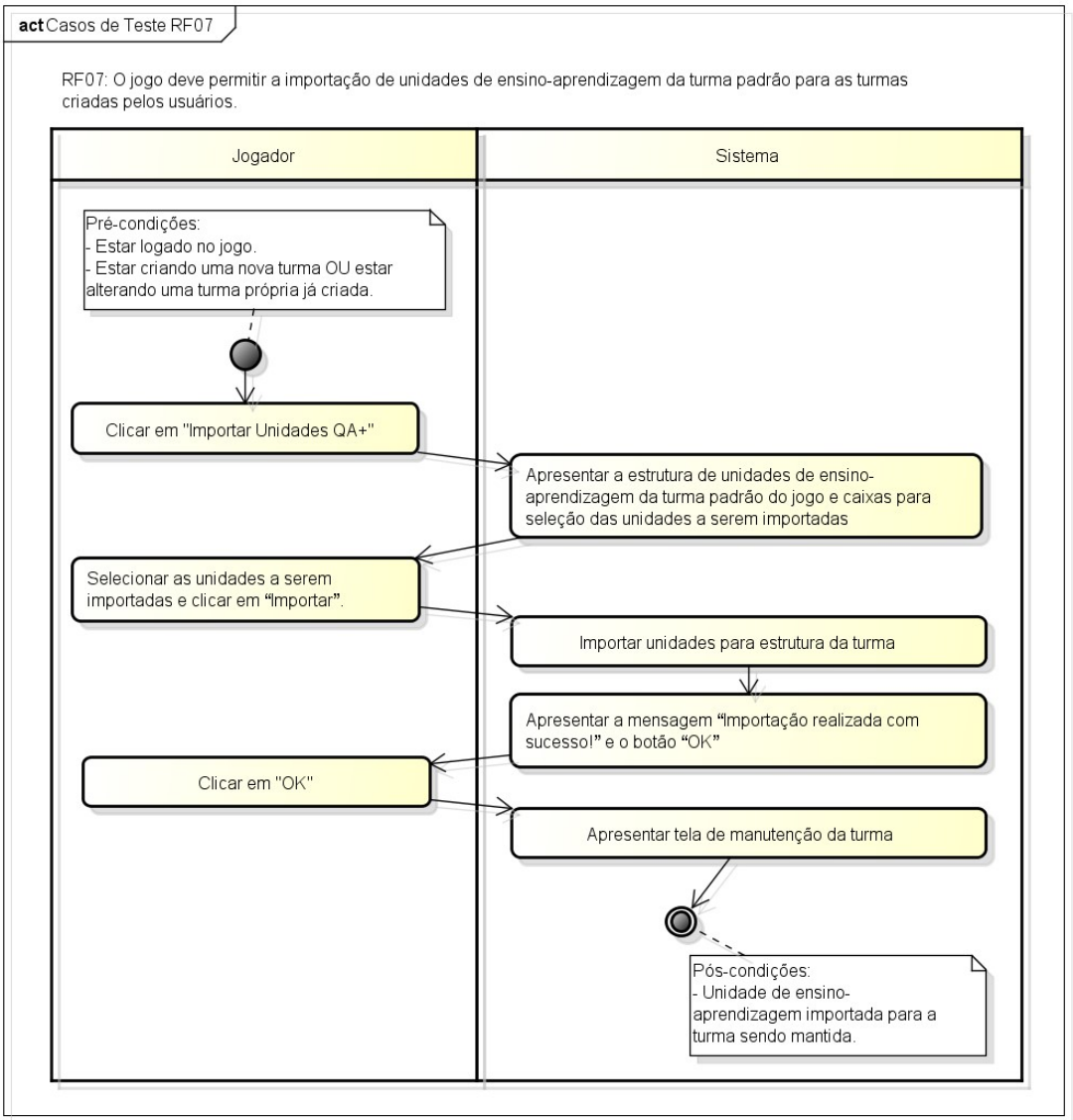

Fonte: elaborado pelo autor (2022)

Figura 7 – Casos de Teste RF06



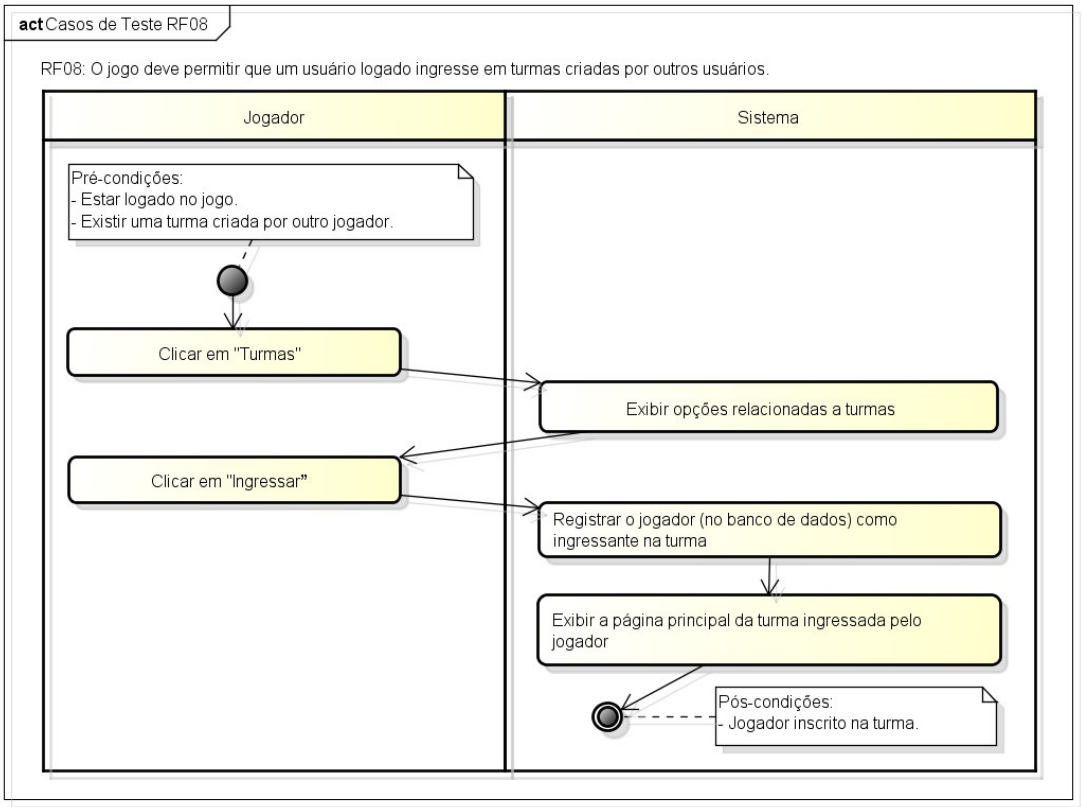

Fonte: elaborado pelo autor (2022)

Figura 8 – Casos de Teste RF07



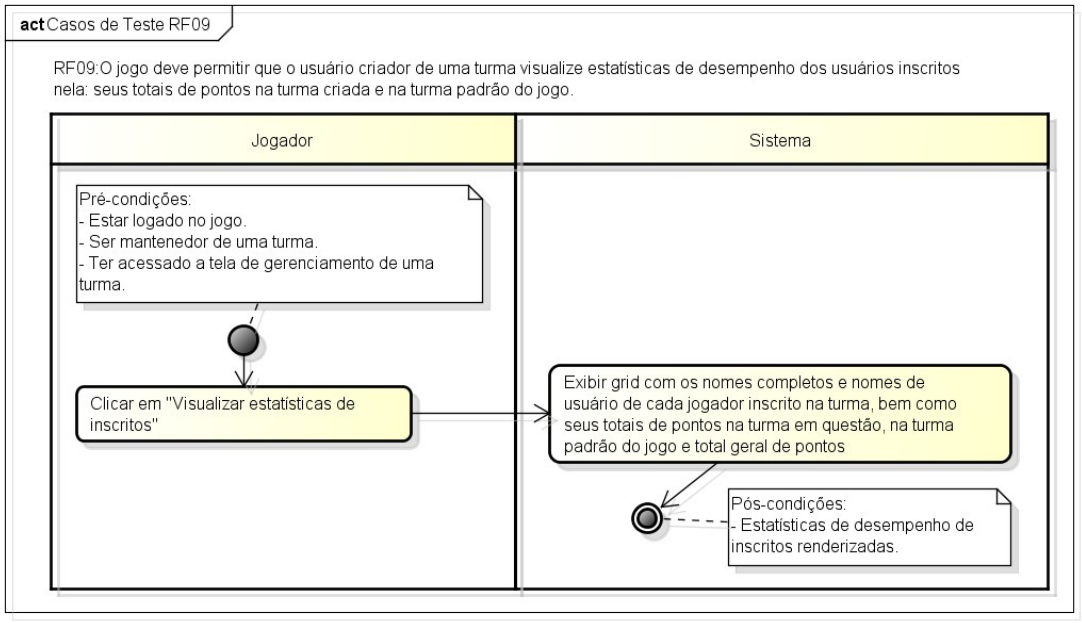

Fonte: elaborado pelo autor (2022)

Figura 9 – Casos de Teste RF08



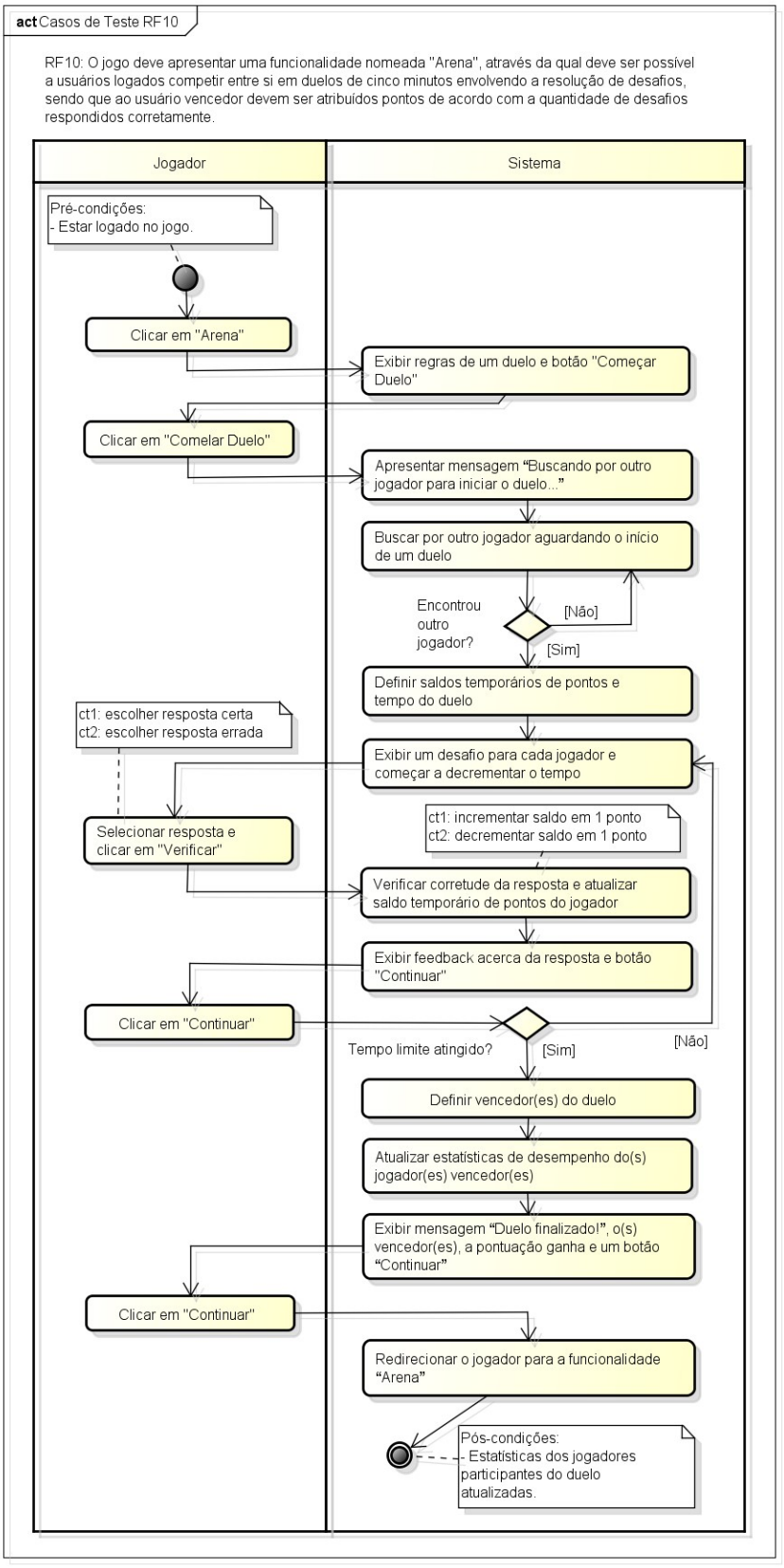

Fonte: elaborado pelo autor (2022)

Figura 10 – Casos de Teste RF09




Fonte: elaborado pelo autor (2022)

Figura 11 – Casos de Teste RF10




Fonte: elaborado pelo autor (2022)

# 7 GERÊNCIA DE REQUISITOS COM RASTREAMENTO DE DEPENDÊNCIAS

Neste capítulo será apresentada a gerência de requisitos do projeto através do rastreamento de dependências, o qual auxilia na análise do impacto/custo da mudança de um requisito, uma vez que relaciona requisitos entre si e entre outras características do sistema.

Sabendo disso, no Quadro 23 pode ser visualizada a matriz de verificação/rastreamento de dependências do projeto, na qual os RFs e RNFs apresentados nos capítulos anteriores são relacionados.

Quadro 23 – Matriz de verificação de dependências

| RF / RNF | RF01 | RF02 | RF03 | RF04 | RF05 | RF06 | RF07 | RF08 | RF09 | RF10 | Custo |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RNF01 | X | X | X | X | X | X | X | X | X | X | **10** |
| RNF02 | X | X | X | X | X | X | X | X | X | X | **10** |
| RNF03 | | X | X | X | X | X | X | X | X | X | **9** |
| RNF04 | X | X | | | | | | | | | **2** |
| RNF05 | X | X | | | | | | | | | **2** |
| RNF06 | X | X | X | X | X | X | X | X | X | X | **10** |
| RNF07 | X | X | X | X | X | X | X | X | X | X | **10** |
| RNF08 | X | X | X | X | X | X | X | X | X | X | **10** |
| RNF09 | | | | X | | | | | | | **1** |
| RNF10 | | | | | X | | | | | | **1** |
| Custo | **7** | **8** | **6** | **7** | **7** | **6** | **6** | **6** | **6** | **6** | |

Fonte: elaborado pelo autor (2022)

Como pode ser observado no Quadro 23, o custo de mudança de um requisito é diretamente proporcional à quantidade de dependências que apresenta.

Ainda, através do Quadro 23 é igualmente possível evidenciar que para o projeto em questão:

- Os custos totais de alteração de RFs e RNFs são iguais a 65 pontos;
- Todos os RFs e RNFs possuem custo de alteração, sendo que estes variam de 1 a 10 pontos cada;
- Os custos de alteração dos RFs são próximos, uma vez que se encontram na faixa de 6 a 8 pontos. Ainda, estes variam menos do que os custos de alteração dos RNFs, que se encontram na faixa de 1 a 10 pontos;
- Os RNFs com códigos RNF01, RNF02, RNF06, RNF07 e RNF08 são os que apresentam maior custo/impacto de alteração. Isto ocorre, pois eles se relacionam a questões a serem observadas em todos os RFs;
- Os RNFs com códigos RNF09 e RNF10 são os que apresentam menor custo/impacto de alteração, uma vez que se associam unicamente a um RF cada; e
- O RF com código RF02 é o que possui maior custo de alteração, enquanto os RFs com códigos RF03, RF06, RF07, RF08, RF09, RF10 são os que apresentam menor custo.

# 8 GERENCIAMENTO DE REQUISITOS COM USE CASE POINTS

Neste capítulo será apresentado o gerenciamento de requisitos do projeto a partir da utilização da técnica conhecida como *Use Case Points*, a qual é composta por algumas etapas (descritas a seguir).

Inicialmente, é necessário classificar os atores envolvidos em cada caso de uso (com base em uma tabela padrão composta pelos tipos de atores simples, médio e complexo) e, feito isso, calcular o peso total dos atores do sistema (*Unadjusted Actor Weight* – UAW), o qual é dado pela soma dos produtos do número de atores de cada tipo por seu respectivo peso.

Conforme pode ser evidenciado na Figura 1, somente 2 atores estão envolvidos com os casos de uso do projeto. Ainda, como ambos correspondem a usuários interagindo com o sistema através de uma interface gráfica, pode-se dizer que estes atores são do tipo “Ator complexo”. Sabendo disso, no Quadro 24 pode ser evidenciado o cálculo de UAW.

Quadro 24 – Definição de UAW

| **Tipo de ator** | **Peso** | **Nº de atores** | **Resultado** |
|---|---|---|---|
| Ator simples | 1 | 0 | 0 |
| Ator médio | 2 | 0 | 0 |
| Ator complexo | 3 | 2 | 6 |
| **Total de UAW** | | | **6** |

Fonte: elaborado pelo autor (2022)

Em sequência, é calculado o peso não ajustado dos casos de uso (*Unadjusted Use Case Weight* - UUCW), o qual é dado pela soma dos produtos do número de casos de uso de cada tipo por seu respectivo peso.

Por meio das especificações completas dos casos de uso do projeto (apresentadas nos Quadros 7 a 16) é possível evidenciar que:

- 7 casos de uso (UC01, UC02, UC03, UC05, UC06, UC07 e UC10) possuem mais de 7 transações (incluindo os fluxos alternativos). Logo, estes casos de uso são considerados complexos;
- 1 caso de uso (UC08) possui de 4 a 7 transações e, apesar de não serem apresentados diagramas de classe e/ou modelagem de banco de dados neste trabalho, a princípio deve ser possível efetuar o caso de uso acessando de 5 a 10 objetos (instâncias de uma classe) ou entidades (tabelas). Assim, este caso de uso pode ser considerado de complexidade média; e
- 2 casos de uso (UC04 e UC09) possuem até 3 transações e são passíveis de serem efetuados acessando menos de 5 objetos ou entidades. Desta forma, podem ser considerados como sendo simples.

Tendo conhecimento das informações acima, é apresentado o Quadro 25 o cálculo de UUCW.

Quadro 25 – Definição de UUCW

| **Complexidade** | **Peso** | **Nº de casos de uso** | **Resultado** |
|---|---|---|---|
| Caso de uso simples | 5 | 2 | 10 |
| Caso de uso médio | 10 | 1 | 10 |
| Caso de uso complexo | 15 | 7 | 105 |
| **Total de UUCW** | | | **125** |

Fonte: elaborado pelo autor (2022)

Conhecendo os valores de UAW e de UUCW é possível calcular os pontos de caso de uso não ajustados (*Unadjusted Use Case Point* – UUCP), dados pela soma de UAW e UUCW. Assim, para o projeto em questão têm-se: UUCP = UAW + UUCW = 6 + 125 = 131 pontos.

Após o cálculo de UUCP é necessário calcular o valor do Fator de Complexidade Técnica (*Technical Complexity Factor* – TCF), dado a partir da fórmula TCF = 0,6 + (0,01 * TFactor). Cabe destacar que TFactor é obtido pelo somatório dos níveis de influência atribuídos a cada um de 13 fatores técnicos multiplicados por seus pesos correspondentes. No Quadro 26 é apresentado o valor de TFactor para este projeto.

Quadro 26 – Definição do TFactor

| **Fator** | **Fatores que contribuem para a complexidade** | **Peso** | **Valor** | **Total** |
|---|---|---|---|---|
| F1 | Sistemas distribuídos | 2 | 0 | 0 |
| F2 | Tempo de resposta | 1 | 5 | 5 |
| F3 | Eficiência para o usuário final (on-line) | 1 | 5 | 5 |
| F4 | Processamento interno complexo | 1 | 3 | 3 |
| F5 | Código reusável | 1 | 5 | 5 |
| F6 | Facilidade de instalação | 0,5 | 0 | 0 |
| F7 | Facilidade de uso (facilidade operacional) | 0,5 | 5 | 2,5 |
| F8 | Portabilidade | 2 | 3 | 6 |
| F9 | Facilidade de mudança | 1 | 5 | 5 |
| F10 | Concorrência (acesso simultâneo à aplicação) | 1 | 5 | 5 |
| F11 | Recursos de segurança | 1 | 3 | 3 |
| F12 | Fornece acesso direto para terceiros | 1 | 5 | 5 |
| F13 | Requer treinamento especial para o usuário | 1 | 0 | 0 |
| **TFactor** | | | | **44,5** |

Fonte: elaborado pelo autor (2022)

Analisando o Quadro 26 percebe-se que TFactor é igual a 44,5. Logo, TCF = 0,6 + (0,01 * TFactor) = 0,6 + (0,01 * 44,5) = 1,045.

Ainda, cabe justificar o valor atribuído para cada fator do Quadro 26:

- F1, F6 e F13 receberam o valor 0, pois não se aplicam ao projeto, uma vez que este não envolve sistemas distribuídos, não precisa ser instalado e não requer treinamento especial para o usuário (algo inclusive reforçado no RNF06);
- F2, F3, F7 e F12 receberam o valor 5, pois o jogo será acessado diretamente por terceiros (será disponibilizado na internet, sendo que qualquer pessoa poderá criar uma conta nele) e há uma grande preocupação em relação à sua eficiência, tempos de resposta e usabilidade, tanto que estas questões são abordadas em RNFs como os de código RNF06, RNF07, RNF09 e RNF10;
- F10 recebeu o valor 5, pois como o jogo será disponibilizado na internet ele deve suportar diversos acessos simultâneos sem apresentar problemas;
- F5 e F9 igualmente receberam o valor 5, pois o jogo prevê a possibilidade dos usuários criarem suas próprias turmas e utilizarem recursos da turma padrão (como unidades de ensino-aprendizagem). Logo, devido a esta flexibilidade, o código do jogo deve ser reusável e de fácil manutenção, pois um mesmo recurso poderá ser utilizado em contextos variados**;**
- F4 recebeu o valor 3, pois acredita-se que o processamento interno apresentará uma complexidade média. F8 igualmente recebeu o valor 3, pois a portabilidade é uma questão importante para o jogo (mencionada no RNF02), mas a princípio será focado somente em 2 navegadores; e
- F11 recebeu 3 pontos, pois questões envolvendo segurança e dados são abordadas em RNFs como os de códigos RNF03, RNF04 e RNF05, mas os tratamentos não chegam a ser complexos e os dados armazenados pelo jogo não implicam em altos riscos (não são registrados e/ou manipulados números de documentos pessoais, dados bancários, telefone, localização, etc).

O próximo cálculo a ser realizado é o do Fator de Complexidade do Ambiente (*Environmental Complexity Factor* – ECF), dado pela fórmula ECF = 1,4 + (-0,03 * Efactor). Cabe destacar que EFactor é obtido pelo somatório dos níveis de influência atribuídos a cada um de 8 fatores ambientais multiplicados por seus pesos correspondentes. No Quadro 27 é apresentado o valor de EFactor para este projeto.

Quadro 27 – Definição do EFactor

| **Fator** | **Fatores que contribuem para a complexidade** | **Peso** | **Valor** | **Total** |
|---|---|---|---|---|
| F1 | Familiaridade da equipe com o processo formal de desenvolvimento adotado | 1,5 | 3 | 4,5 |
| F2 | Colaboradores de meio período | -1 | 5 | -5 |
| F3 | Capacidade do líder do projeto em análise de requisitos e modelagem | 0,5 | 5 | 2,5 |
| F4 | Experiência da equipe em desenvolvimento de aplicações do gênero em questão | 0,5 | 4 | 2 |
| F5 | Experiência em Orientação a Objetos | 1 | 2 | 2 |
| F6 | Motivação da equipe | 1 | 5 | 5 |
| F7 | Dificuldades com a linguagem de programação | -1 | 4 | -4 |
| F8 | Requisitos estáveis | 2 | 2 | 4 |
| **EFactor** | | | | **11** |

Fonte: elaborado pelo autor (2022)

Analisando o Quadro 26 percebe-se que EFactor é igual a 11. Logo, ECF = 1,4 + (-0,03 * Efactor) = 1,4 + (-0,03 * 11) = 1,07. Ainda, cabe justificar o valor atribuído para cada fator do Quadro 27:

- A F1 foi atribuído o valor 3, pois é interessante que haja familiaridade da equipe com o processo, mas a relevância desta questão é média;
- O valor de F2 é 5, pois o colaborador do projeto teria disponibilidade de meio período ou menos para se dedicar a ele;

- F3 possui valor 5, pois a análise dos requisitos e modelagem é fundamental para o sucesso do projeto (para que seu valor seja maximizado);
- F4 possui valor 4, pois tal experiência é interessante (mas não obrigatória) para que o jogo atenda sua finalidade educativa e ao mesmo tempo proporcione diversão aos usuários;
- F5 recebeu o valor 2, pois o jogo não precisa ser desenvolvido necessariamente utilizando orientação a objetos;
- A F6 foi atribuído o valor 5, pois a motivação da equipe é fundamental para que o projeto possa ser bem executado e finalizado;
- F7 possui o valor 4, pois as linguagens a serem empregadas no projeto já foram utilizadas no passado pela equipe, mas em projetos simples. Assim, caberia um tempo para a realização de estudos, de modo a relembrar alguns detalhes e se aprofundar nas tecnologias; e
- A F8 foi atribuído o valor 2, pois pretende-se utilizar métodos ágeis para o desenvolvimento do projeto. Logo, entende-se como normal a evolução dos requisitos ao longo do tempo.

Tendo conhecimento de UUCP, TCF e ECF é possível então calcular o valor total do sistema em *Use Case Points* (UCP) por meio da fórmula UCP = UUCP * TCF * ECF. Sabendo disso, para o projeto em questão temos: UCP = UUCP * TCF * ECF = 131 * 1,045 * 1,07 = 146,48 Pontos de Caso de Uso.

Por fim, cabe ressaltar que para padrões atuais de desenvolvimento (com ambientes integrados) sugere-se estimar o tempo necessário para o desenvolvimento de um projeto como uma média de 10 horas de trabalho por ponto de caso de uso (UCP). Sabendo disso, tem-se que o tempo estimado para o projeto em questão é de 1464,8 horas de trabalho (146,48 UCP * 10h).

# 9 PRIORIDADE DE REQUISITOS COM USE CASE POINTS

Neste capítulo será abordada a temática de priorização de requisitos e realizada a redução do custo do projeto em pelo menos 30% (considerando os valores obtidos no capítulo 8).

Inicialmente, tendo ciência da necessidade de redução do custo do projeto, realizou-se uma repriorização dos requisitos funcionais (resultado exposto no Quadro 28).

Comparando os RFs dos Quadros 2 e 28 é perceptível a realização das seguintes alterações: os RFs com códigos RF06 e RF08 tiveram suas prioridades alteradas de essencial para desejável. Estas mudanças ocorreram, pois o grupo de RFs com códigos entre RF06 a RF09 faz parte de um mesmo contexto e pode ser considerado como adicional à estrutura básica do jogo. Caso fossem mantidos unicamente os requisitos priorizados originalmente como essenciais, o propósito educativo do projeto seria atendido, mas elementos básicos de jogos precisariam ser removidos, o que descaracterizaria o projeto neste sentido.

Quadro 28 – Requisitos Funcionais repriorizados

| Código | Descrição | Visibilidade | Prioridade | RNFs Associados |
|---|---|---|---|---|
| RF01 | O jogo deve permitir que um usuário crie uma conta nele por meio do informe de nome completo (real), nome de usuário, e-mail e senha. | Evidente | Essencial | RNF01<br>RNF02<br>RNF04<br>RNF05<br>RNF06<br>RNF07<br>RNF08 |
| RF02 | O jogo deve permitir que um usuário logado mantenha sua conta (possa visualizar e alterar seus dados de cadastro e avatar, bem como excluir sua conta). | Evidente | Essencial | RNF01<br>RNF02<br>RNF03<br>RNF04<br>RNF05<br>RNF06<br>RNF07<br>RNF08 |
| RF03 | O jogo deve apresentar uma página de login que permita a um usuário previamente cadastrado acessar suas funcionalidades restritas por | Evidente | Essencial | RNF01<br>RNF02<br>RNF03<br>RNF06 |

| | | | | |
|---|---|---|---|---|
| | meio do informe de e-mail e senha. | | | RNF07<br>RNF08 |
| RF04 | O jogo deve apresentar a usuários logados, em sua tela principal, um quadro de líderes no estilo ranking, no qual devem ser expostas em ordem decrescente de pontos a classificação, o avatar, o nome de usuário e o total de pontos: do usuário classificado na 1ª posição, do usuário logado, do usuário que antecede o usuário logado no ranking (se houver) e do usuário que sucede o usuário logado no ranking (se houver). | Evidente | Importante | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08<br>RNF09 |
| RF05 | O jogo deve apresentar uma turma padrão através da qual deve ser possível a qualquer usuário logado acessar unidades de ensino-aprendizagem constituídas por: (a) uma lição (composta por um título, conteúdos em formato multimídia e uma lista de referências); e (b) um conjunto de atividades com correção automática (nomeados desafios), cuja resolução correta acarreta na atribuição de pontos ao usuário. | Evidente | Essencial | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08<br>RNF10 |
| RF06 | O jogo deve permitir que um usuário logado mantenha novas turmas: possa criar turmas, visualizar e alterar suas unidades de ensino-aprendizagem (podendo inclusive personalizar suas lições e desafios), gerenciar seus inscritos e excluir as turmas por ele criadas. | Evidente | Desejável | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |
| RF07 | O jogo deve permitir a importação de unidades de ensino-aprendizagem da turma padrão para as turmas criadas pelos usuários. | Evidente | Desejável | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |
| RF08 | O jogo deve permitir que um usuário logado ingresse em turmas criadas por outros usuários. | Evidente | Desejável | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |
| RF09 | O jogo deve permitir que o usuário criador de uma turma visualize estatísticas de desempenho dos usuários inscritos nela: seus totais de pontos na turma criada e na | Evidente | Desejável | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07 |

| | | | | |
|---|---|---|---|---|
| | turma padrão do jogo. | | | RNF08 |
| RF10 | O jogo deve apresentar uma funcionalidade nomeada "Arena", através da qual deve ser possível a usuários logados competir entre si em duelos de cinco minutos envolvendo a resolução de desafios, sendo que ao usuário vencedor devem ser atribuídos pontos de acordo com a quantidade de desafios respondidos corretamente. | Evidente | Importante | RNF01<br>RNF02<br>RNF03<br>RNF06<br>RNF07<br>RNF08 |

Fonte: elaborado pelo autor (2022)

De modo a atender a solicitação de redução dos custos do projeto em pelo menos 30% foram removidos os casos de uso UC06, UC07, UC08 e UC09, priorizados como desejáveis no Quadro 28 (ou seja, os casos de uso com menor prioridade). Cabe destacar que por meio desta redução o ator “Jogador (mantenedor de turma)” igualmente foi removido, uma vez que nenhum caso de uso associado a ele foi mantido.

Considerando as modificações acima e o procedimento de cálculo de UCP apresentado no capítulo 8, tem-se após as alterações em questão novos valores para UAW (Quadro 29), UUCW (Quadro 30) e, consequentemente, UUCP e UCP.

Quadro 29 – Definição de UAW após redução de custo

| **Tipo de ator** | **Peso** | **Nº de atores** | **Resultado** |
|---|---|---|---|
| Ator simples | 1 | 0 | 0 |
| Ator médio | 2 | 0 | 0 |
| Ator complexo | 3 | 1 | 3 |
| **Total de UAW** | | | **3** |

Fonte: elaborado pelo autor (2022)

Quadro 30 – Definição de UUCW após redução de custo

| **Complexidade** | **Peso** | **Nº de casos de uso** | **Resultado** |
|---|---|---|---|
| Caso de uso simples | 5 | 1 | 5 |

| | | | |
|---|---|---|---|
| Caso de uso médio | 10 | 0 | 0 |
| Caso de uso complexo | 15 | 5 | 75 |
| **Total de UUCW** | | | **80** |

Fonte: elaborado pelo autor (2022)

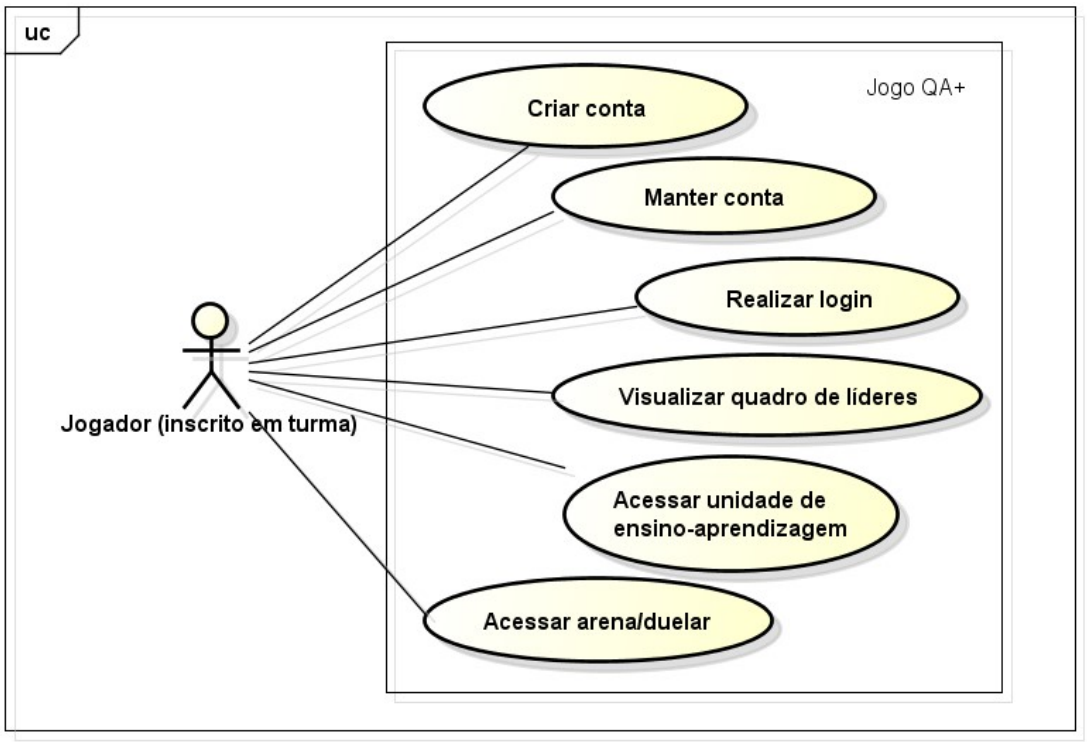
Considerando os valores dos Quadros 29 e 30 tem-se:

- UUCP = UAW + UUCW = 3 + 80 = 83;
- UCP = UUCP * TCF * ECF = 83 * 1,045 * 1,07 = 92,81;
- Tempo estimado = UCP * 10 = 928,1 horas de trabalho.

Como originalmente o tempo estimado de trabalho era de 1464,8 horas percebe-se que houve uma redução de 36,64% do custo do projeto.

Por fim, na Figura 12 pode ser visualizado o diagrama de casos de uso resultante da redução de custos apresentada.

Figura 12 – Diagrama de casos de uso após redução de custo




Fonte: elaborado pelo autor (2022)

# 10 CONCLUSÕES

A Engenharia de Requisitos consiste em uma importante área da Engenharia de Software, uma vez que objetiva o desenvolvimento de ferramentas metodológicas para orientar a descoberta, desenvolvimento, rastreamento, análise, teste, comunicação e gestão de requisitos de sistemas computacionais, de modo com que um sistema possa ser definido em diferentes níveis de abstração, bem como atenda adequadamente as necessidades de seus usuários e clientes.

Neste trabalho buscou-se consolidar o aprendizado do autor acerca de Engenharia de Requisitos através da construção do documento de requisitos de um projeto nomeado QA+, o qual corresponde a um jogo voltado ao ensino de teste e qualidade de software. Para a criação deste documento foram realizadas análises e criados artefatos típicos de atividades que compõem a gerência de requisitos.

Inicialmente foi apresentado o estado da arte referente ao tema do projeto, de modo a contextualizá-lo. Em seguida, foi discorrido acerca do levantamento dos requisitos do jogo proposto, expostos seus requisitos funcionais, não funcionais e de design, além dos casos de uso associados (representados em um diagrama, especificados e mapeados em relação aos requisitos). Igualmente foram apresentados no trabalho artefatos relacionados ao rastreamento, verificação, validação e gerência dos requisitos (este último por meio de uma matriz de rastreamento de dependências). Finalmente, foi detalhado o cálculo de pontos de caso de uso do projeto e abordada a temática de priorização de requisitos.

A partir da realização do trabalho foi possível reforçar o entendimento do autor acerca da importância de uma boa gerência de requisitos em projetos de sistemas computacionais, bem como exercitar a análise e criação de artefatos comumente utilizados em atividades da área de Engenharia de Requisitos.

Por fim, com base no exposto, conclui-se que o trabalho atingiu seus objetivos propostos.

# 11 REFERÊNCIAS BIBLIOGRÁFICAS


BARBOSA, Ana Karoline T.; NEVES, Larissa L. E.; DIAS-NETO, Arilo C. JoVeTest - Jogo da Velha para Auxiliar no Ensino e Estudo de Teste de Software. *In:* FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE (FEES), 9, 2016, Maringá. **Anais [...]** 2016.

BELL, Jonathan; SHETH, Swapneel; KAISER, Gail. Secret Ninja Testing with HALO Software Engineering. In: INTERNATIONAL WORKSHOP ON SOCIAL SOFTWARE ENGINEERING (SSE'11), 4, 2011, Szeged. **Proceedings** [...] New York: ACM, 2011, p. 43–47.

BEPPE, Thiago A. *et al*. GreaTest: A Card Game to Motivate the Software Testing Learning. *In*: BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING (SBES'18), 32, 2018, São Carlos. **Proceedings** [...] New York: ACM, 2018, p. 298–307.

BUFFARDI, Kevin; VALDIVIA, Pedro. Bug Hide-and-Seek: An Educational Game for Investigating Verification Accuracy in Software Tests. *In:* FRONTIERS IN EDUCATION CONFERENCE (FIE), 2018. **Proceedings** [...] IEEE, 2018, p. 1–8.

CLEGG, Benjamin S.; ROJAS, José Miguel; FRASER, Gordon. Teaching Software Testing Concepts Using a Mutation Testing Game. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING EDUCATION AND TRAINING TRACK (ICSE-SEET), 39, 2017. **Proceedings** [...] IEEE/ACM, 2017, p. 33–36.

COSTA, Igor Ernesto Ferreira; OLIVEIRA, Sandro Ronaldo Bezerra. The use of gamification to support the teaching-learning of software exploratory testing: an experience report based on the application of a framework. *In:* FRONTIERS IN EDUCATION CONFERENCE (FIE), 2020. **Proceedings** [...] IEEE, 2020, p. 1–9.

DINIZ, Lucio L.; DAZZI, Rudimar L. S. Jogo para o Apoio ao Ensino do Teste de Caixa-Preta. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA

EDUCAÇÃO (SBIE), 22 – WORKSHOP DE INFORMÁTICA NA ESCOLA (WIE), 17, 2011, Aracaju. **Anais** [...] Aracaju, 2011. p. 426–435.

ELBAUM, Sebastian et al. Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable. *In*: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE'07), 29, 2007. **Proceedings** [...] IEEE, 2007, p. 688–697.

FARIAS, Virgínia et al. iTest Learning: Um Jogo para o Ensino do Planejamento de Testes de Software. *In:* FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE (FEES), 5, 2012, Natal. **Anais [...]** 2012, p. 1–8.

GOTEL, O; FINKELSTEIN, A. An Analysis of the Requirements Traceability Problem. *In*: INTERNATIONAL CONFERENCE ON REQUIREMENTS ENGINEERING (ICRE'94), 1, 1994, Colorado. **Proceedings** [...] IEEE, 1994, p. 94–101.

JESUS, Gabriela Martins de et al. Gamification in Software Testing: A Characterization Study. *In:* BRAZILIAN SYMPOSIUM ON SYSTEMATIC AND AUTOMATED SOFTWARE TESTING (SAST'18), 3, 2018, São Carlos. **Proceedings** [...] New York: ACM, 2018, p. 39–48.

JESUS, Gabriela Martins de. **Explorando Gamificação no Ensino de Teste de Software**. 2019. Dissertação (Mestrado em Ciência da Computação) – Curso de Pós Graduação em Ciência da Computação, Universidade Federal de São Carlos, São Carlos, 2019.

MACÊDO, Kelly Santos. **As aventuras de Jack Test:** jogo educacional para o apoio ao ensino de teste de software. 2014. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Curso de Bacharelado em Ciência da Computação, Universidade Estadual do Sudoeste da Bahia, Vitória da Conquista, 2014.

NASCIMENTO, Eduardo Henrique Rocha do. **Aplicando Gamificação no Ensino de Teste de Software**. 2019. Dissertação (Mestrado em Sistemas e

Computação) – Curso de Pós Graduação em Sistemas e Computação, Universidade Federal do Rio Grande do Norte, Natal, 2019.

OLIVEIRA, Bruno; AFONSO, Paulo; COSTA, Heitor. TestEG — A computational game for teaching of software testing. *In:* INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY (SCCC), 35, 2016. **Proceedings** [...] IEEE, 2016, p. 1–10.

PRESSMAN, Roger S. **Software engineering**: a practioner's approach. 8th edition. New York: McGraw-Hill Education, 2015. 972 p. ISBN 0078022126.

RIBEIRO, Tânia P. B.; PAIVA, Ana C. R. iLearnTest: Educational game for learning software testing. *In:* IBERIAN CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGIES (CISTI), 10, 2015. **Proceedings** [...] IEEE, 2015, p. 1–6.

RIBEIRO, Nayara *et al*. Avaliando a Viabilidade do BlackBox em Sala de Aula: Um Jogo Sério para Ensino de Teste Funcional de Software. *In:* SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 28, 2017. **Anais** [...] 2017, p. 817–826.

ROJAS, José Miguel; FRASER, Gordon. Code Defenders: A Mutation Testing Game. *In*: CONFERENCE ON SOFTWARE TESTING, VERIFICATION AND VALIDATION WORKSHOPS (ICSTW), 9, 2016. **Proceedings** [...] IEEE, 2016, p. 162–167.

SHERIF, Eman et al. Gamification to Aid the Learning of Test Coverage Concepts. *In*: CONFERENCE ON SOFTWARE ENGINEERING EDUCATION AND TRAINING (CSEE&T), 32, 2020. **Proceedings** [...] IEEE, 2020, p. 1–5.

SILVA, Antonio Carlos. **Jogo educacional para apoiar o ensino de técnicas para elaboração de testes de unidade**. 2010. Dissertação (Mestrado em Computação Aplicada) – Curso de Mestrado Acadêmico em Computação Aplicada, Universidade do Vale do Itajaí, São José, 2010.

SILVA, Tarcila Gesteira da; BERNARDI, Giliane; MÜLLER, Felipe. Abordagem de Apoio ao Ensino e Aprendizagem de Teste de Software Baseada em Jogos Sérios e Mundos Virtuais. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 22 – WORKSHOP DE INFORMÁTICA NA ESCOLA (WIE), 17, 2011, Aracaju. **Anais** [...] Aracaju, 2011. p. 538–541.

SILVA, Tarcila Gesteira da. **Jogos sérios em mundos virtuais:** uma abordagem para o ensino-aprendizagem de teste de software. 2012. Dissertação (Mestrado em Computação) – Curso de Pós Graduação em Informática, Universidade Federal de Santa Maria, Santa Maria, 2012.

SILVA, Cleibson Gomes *et al*. gTest Learning: Um Jogo para Ensino Básico de Teste de Software. *In:* CONGRESSO REGIONAL SOBRE TECNOLOGIAS NA EDUCAÇÃO (Ctrl+E), 2016, Natal. **Anais** [...] Natal, 2016, p. 450–460.

SOMMERVILLE, Ian; SAWYER, Pete. **Requirements Engineering:** A Good Practice Guide. John Wiley & Sons, 1997. 404 p.

SOMMERVILLE, Ian. **Software Engineering**. 9th edition. Massachusetts: Addison-Wesley, 2011. 790 p. ISBN 9780137035151.

SOSKA, Alexander; MOTTOK, Jürgen; WOLFF, Christian. Pattern oriented card game development: SOFTTY — A card game for academic learning of software testing. *In*: GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 2017. **Proceedings** [...] IEEE, 2017, p. 1166–1173.

SOSKA, Alexander; MOTTOK, Jürgen; WOLFF, Christian. An experimental card game for software testing: Development, design and evaluation of a physical card game to deepen the knowledge of students in academic software testing education. *In*: GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON), 2016. **Proceedings** [...] IEEE, 2016, p. 576–584.

VALLE, Pedro Henrique Dias; ROCHA, Rafaela Vilela; MALDONADO, José Carlos. Testing Game: An Educational Game to Support Software Testing Education. *In*: BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING

(SBES'17), 31, 2017, Fortaleza. **Proceedings** [...] New York: ACM, 2017, p. 289–298.

WERBACH, Kevin; HUNTER, Dan. **For the Win:** How Game Thinking Can Revolutionize Your Business. 1 ed. Pennsylvania: Wharton Digital Press, 2012.

WIERINGA, R. J. **An introduction to requirements traceability**. Technical Report IR-389, Faculty of Mathematics and Computer Science, University of Vruje, Amsterdam, Septembet 1995.