

UM JOGO SÉRIO PARA PROMOÇÃO DA EDUCAÇÃO NO ENSINO BÁSICO

A SERIOUS GAME FOR THE PROMOTION OF EDUCATION IN PRIMARY EDUCATION

Jéssica Rosália Fernandes¹
Reudismam Rolim de Sousa²
Veronica Maria Lima Silva³

RESUMO:

A sala de aula tem se tornado cada vez mais dinâmica, a fim de acompanhar os avanços tecnológicos e enriquecer o ensino-aprendizagem e uma das formas de se alcançar tal feito é por meio de jogos sérios, em que se utiliza um jogo computacional voltado para fins educacionais. Dada tal pressuposição, o objetivo deste artigo é propor a arquitetura de *software* para um jogo sério no formato de um *quiz*, voltado principalmente para a educação básica. Com a colaboração de uma escola da região do Alto Oeste Potiguar, no estado do Rio Grande do Norte, situado no Brasil, chegou-se a conclusão da necessidade de criação de uma aplicação que combine determinados aspectos e funcionalidades que outras aplicações não possuem, como os vários modos de jogos (*solo*, *duo*, mas principalmente competição com líderes) e a autonomia dos professores no gerenciamento de questões e competições. Ao se utilizar de uma ferramenta deste tipo, ganha-se com a facilitação da aprendizagem, sem deixar de lado a diversão, garantindo que o ensino fique pautado no propósito principal, que é a aprendizagem.

PALAVRAS-CHAVE: jogo sério; ensino-aprendizagem; arquitetura de *software*.

ABSTRACT:

The classroom has become increasingly dynamic, to keep up with technological advances and enrich teaching-learning, and one of the ways to achieve this is through serious games, in which a computer game is used for educational purposes. Given this assumption, the objective of this article is to propose a software architecture for a serious game in the format of a quiz, aimed mainly at basic education. With the collaboration of a school in the Alto Oeste Potiguar region, in the state of Rio Grande do Norte, located in Brazil, it was concluded that there was a need to create an application that combines certain aspects and functionalities that are missing in other applications, such as the various game modes (*solo*, *duo*, but mainly competition with leaders) and the autonomy of teachers in managing questions and competitions. When using a tool of this type, you benefit from facilitating learning, without leaving aside the fun, ensuring that teaching is guided by the main purpose, which is learning.

KEYWORDS: serious game; teaching-learning; software architecture.

Disponível em: <https://recima21.com.br/index.php/recima21/article/view/5758>

¹ Jéssica Rosália Fernandes – Graduanda em Tecnologia da Informação – UFERSA – Pau dos Ferros.

² Reudismam Rolim de Sousa – Doutor em Computação – UFCG – Campina Grande.

³ Veronica Maria Lima Silva – Doutora em Engenharia Elétrica – UFCG – Campina Grande.

1 INTRODUÇÃO

Os jogos sérios têm se tornado cada vez mais frequentes no meio educacional, uma vez que eles podem trazer um impacto positivo no estímulo à aprendizagem dos alunos. Neste contexto, Mitamura, Suzuki e Oohori (2012) destacam a existência de um movimento ativo na busca de enriquecer os ambientes de aprendizagem através de jogos sérios, uma vez que os jogos são eficazes quando se trata de oportunizar e alavancar contextos lúdicos e ficcionais na forma de narrativas, sons e imagens, e com isso se tem a prerrogativa da geração e relação com o conhecimento (Domínguez *et al.*, 2013). Além disso, Furió *et al.* (2013 apud Busarello, 2016) sugerem que o ato de jogar, além de ser prazeroso, é capaz de estimular o desenvolvimento cognitivo, melhorando a atenção e a memória.

Uma vez que os jovens, adultos e, até crianças, costumam acessar jogos e tecnologias, de maneira geral, o uso de jogos sérios pode trazer maior dinamicidade às aulas e levar a refletir sobre as mudanças causadas pela tecnologia na forma como as pessoas trabalham, se relacionam e estudam. Por isso, é importante considerar que adaptar o ensino-aprendizagem a uma maneira que o indivíduo se sente motivado e está mais habituado pode ser fundamental na facilitação da aprendizagem. Bruner (1969 *apud* Moreira, 1999) defende que, uma vez que se usa as abordagens de ensino adequadas, qualquer assunto é passível de ser ensinado a uma criança, independentemente do seu estágio de desenvolvimento.

Diante deste contexto, a fim de transpor barreiras no processo ensino-aprendizagem, o presente trabalho vem propor a arquitetura de *software* para um *quiz* educacional, que poderá ser utilizado para reter conhecimento acerca de diversos temas enquanto se diverte jogando. A ideia, que surgiu de uma necessidade de uma escola localizada na região do Alto Oeste Potiguar, no estado do Rio Grande do Norte, Brasil, pretende fomentar e ajudar na mudança de postura no ensino, proporcionando uma aprendizagem ativa, dinâmica, atrativa e efetiva, em que o aluno é exposto ao conteúdo em um ambiente lúdico, controlado e de baixo risco, potencializando assim o engajamento e a exploração, de forma lúdica.

Para a aplicação em pauta, tendo em vista o envolvimento da escola da rede municipal de ensino, estudou-se os principais problemas enfrentados em sala de aula ao tentar usar jogos sérios e o porquê que outras ferramentas semelhantes não supriam as principais necessidades. A conclusão obtida destacava a necessidade de dar autonomia aos professores para gerirem questões e competições, mas, principalmente, de viabilizar o modo de competição (dois grupos disputando com todos os envolvidos assistindo em tempo real, porém com um número limitado de representantes respondendo). Atualmente, com base nas informações coletadas sobre os principais problemas enfrentados em sala de aula — obtidas por meio de entrevistas com educadores e alunos durante a fase de entendimento da demanda e início da elicitação de requisitos — verificou-se que, para realizar uma competição com jogadores ouvintes e alguns representantes, é necessário combinar o uso de um jogo sério com ferramentas auxiliares. Isso se deve ao fato de que não existe uma solução única que atenda a todas as atividades lúdicas. Diante disso, o jogo sério proposto visa resolver esses desafios, oferecendo uma plataforma capaz de suprir todas as necessidades de forma integrada.

2 METODOLOGIA

Nesta seção, serão descritos os métodos utilizados para coletar informações relevantes sobre os requisitos do sistema, bem como a abordagem adotada para garantir que as necessidades dos usuários fossem devidamente consideradas na arquitetura proposta.

2.1 Métodos de pesquisa

Para garantir que o sistema atendesse às necessidades dos seus usuários finais, foram utilizadas duas abordagens complementares de pesquisa: entrevistas qualitativas e análise de sistemas existentes. As entrevistas qualitativas envolveram educadores e alunos, que seriam os principais usuários do sistema de *quiz*, com o objetivo de compreender as dificuldades enfrentadas no uso de ferramentas educacionais já disponíveis e identificar os requisitos funcionais e não funcionais necessários. As entrevistas foram conduzidas de forma semi-estruturada, combinando perguntas abertas e fechadas, o que proporcionou flexibilidade para os participantes compartilharem suas opiniões e sugestões de maneira mais livre e detalhada.

Além disso, a análise de sistemas existentes foi realizada com o propósito de estudar outras plataformas de jogos sérios, como *Kahoot* e Perguntados, avaliando as funcionalidades oferecidas e identificando suas limitações. Essa análise permitiu determinar quais características poderiam ser melhoradas na arquitetura proposta, de modo a garantir que o novo sistema superasse limitações observadas nas ferramentas disponíveis no mercado.

2.2 Abordagem adotada

A abordagem adotada foi a de Engenharia de Requisitos Participativa, em que os usuários finais — educadores e alunos — foram envolvidos diretamente na fase de definição dos requisitos do sistema. Esse envolvimento ativo foi essencial para garantir que a solução a ser desenvolvida atendesse às reais demandas e expectativas dos usuários.

A coleta de dados, realizada por meio de entrevistas e sessões de *brainstorming* colaborativos, foi uma aplicação direta dessa abordagem participativa. Esses encontros permitiram identificar as principais funcionalidades desejadas para o sistema, como diferentes modos de jogo (*solo*, *duo* e competição), além de especificar as condições de uso e os requisitos técnicos necessários para suportar a plataforma educacional de maneira eficiente.

O uso dessa abordagem garantiu que as decisões sobre a arquitetura do sistema fossem baseadas em necessidades dos usuários, antes de passar para as fases de *design* e implementação propriamente ditas. Com isso, foi possível definir uma estrutura sólida para o sistema, capaz de lidar com as funcionalidades e demandas apresentadas pelos futuros usuários.

2.3 Coleta e análise de dados

Os dados coletados durante as entrevistas qualitativas foram analisados por meio de análise de conteúdo, categorizando as respostas dos participantes em temas e tópicos recorrentes, como usabilidade, gestão de competições e de conteúdos. Essas categorias formaram a base para a criação dos requisitos do sistema, servindo também como critérios para avaliar a eficiência da arquitetura proposta. Além disso, os resultados da análise de sistemas existentes foram utilizados para comparar as funcionalidades de diferentes plataformas e identificar pontos que poderiam ser melhorados ou adaptados no novo sistema. Essa análise comparativa ajudou a delinear o que seria inovador na arquitetura proposta, como a introdução de um modo de competição mais interativo e inclusivo, permitindo o envolvimento simultâneo de todos os alunos.

3 ARQUITETURA

A definição da arquitetura de um sistema é fundamental, uma vez que esse instrumento bem projetado ajuda a garantir que o *software* atenda às necessidades inicialmente idealizadas e seja capaz de se flexibilizar — sem perder sua robustez — para receber mudanças. Bass, Clements e Kazman (2012) definem a arquitetura de um *software* como uma organização do sistema, evidenciando seus componentes e suas respectivas propriedades, bem como os relacionamentos entre eles. No ciclo de vida do desenvolvimento de *software*, tem-se a designação da arquitetura como sendo umas das primeiras etapas, pois este momento de identificação dos componentes e suas relações é propício para negociações com clientes, para antecipar-se a problemas futuros para ter menos retrabalho e para criar insumos (documentação) que ajudarão durante o desenvolvimento.

4 REQUISITOS DO SISTEMA

De acordo com Sommerville (2011), os requisitos descrevem o que o sistema deve fazer, os serviços que são oferecidos e quais as restrições presentes. As necessidades dos usuários devem ser enxergadas nos requisitos, sendo que estes podem ser funcionais ou não funcionais. Ainda segundo Sommerville (2011), apesar de na prática a distinção entre tais tipos não se mostrar tão clara, há uma classificação, em que os funcionais são aqueles que denotam o que o sistema deve fazer, enquanto que os não funcionais são restrições às funções.

A definição do escopo de um projeto pode influenciar no curso, no retrabalho e no grau de entendimento dos envolvidos na aplicação, além de impactar no nível de conformidade com o esperado pelos *stakeholders*. Por isso, esta etapa não pode ser negligenciada.

4.1 Requisitos funcionais

Os requisitos funcionais visam elencar quais funções a aplicação será capaz de desempenhar e como ela reage a determinadas entradas e em determinadas situações (Sommerville, 2011).

O Quadro 1 mostra os requisitos funcionais do ambiente *quiz* educacional. Para cada requisito, pode-se ver o código, seguido do nome, uma descrição, explicando a função, e a prioridade. Quanto ao código dos requisitos, "RF" foi adotado para referir-se a "Requisito funcional" e o número designa a sequência.

Referente ao grau de prioridade, este pode ser “Essencial”, “Importante” ou “Desejável”. Um requisito essencial é aquele que o sistema obrigatoriamente deve conter, pois a aplicação pode ficar descaracterizada na ausência dele; já um requisito importante é aquele que, apesar de não ser indispensável para o *software* em produção, agrega valor em potencial; por fim, um requisito desejável é aquele que pode ser entendido como um adicional e que não deve impedir de uma aplicação ser entendida como pronta e totalmente usável. O critério de priorização atribuído para tais requisitos envolveu principalmente a importância do requisito, tanto no âmbito da entrega de valor, quanto no que diz respeito à dependência entre requisitos, em que alguns requisitos naturalmente precisam vir primeiro, pois outros dependem deles.

Para se chegar a esta lista de requisitos funcionais, foi essencial o envolvimento de voluntários da escola do Alto Oeste Potiguar, cuja necessidade inicial motivou o desenvolvimento do *quiz*. Durante o processo de elicitação, foram aplicados questionários e realizadas entrevistas com os participantes para entender melhor as demandas da aplicação.

Quadro 1 - Lista dos requisitos funcionais do *quiz* educacional.

Requisito	Descrição	Prioridade
[RF001] Realizar cadastro como professor	O sistema deve permitir que o usuário se cadastre como um professor. Para isso deve informar obrigatoriamente nome, CPF, nome da escola, áreas de atuação, número da matrícula, e-mail e senha. Como opcional pode informar o telefone.	Essencial
[RF002] Realizar cadastro como jogador	O sistema deve permitir que o usuário se cadastre como um professor. Para isso deve informar nome, CPF, telefone, nome da escola, e-mail e senha. Telefone e nome da escola são opcionais.	Essencial
[RF003] Realizar login	O sistema deve permitir que um professor ou jogador realize o login informando e-mail e senha.	Essencial
[RF004] Cadastrar questão	O sistema deve permitir que o usuário com perfil “professor” cadastre questões. No cadastro, deverão ser informados enunciado, quatro alternativas, nível de ensino, visibilidade da questão e nome do conteúdo da questão. Apenas o campo de nome do conteúdo da questão é opcional.	Essencial
[RF005] Listar questões	O sistema deve permitir que o professor visualize uma listagem com todas suas questões criadas. O professor poderá filtrar por conteúdo e nível de ensino.	Essencial
[RF006] Visualizar questão	O sistema deve permitir que o professor visualize as informações de uma questão em específico. Ao visualizar poderá ser visto: Enunciado, alternativas possíveis e a alternativa correta, nível de ensino da questão, visibilidade da questão e o conteúdo a qual a questão pertence.	Essencial
[RF007] Cadastrar/jogar competição	O sistema deve permitir que o professor crie uma competição entre 2 equipes. No cadastro deverão ser informados: Nome da competição, data da realização, a lista de questões, descrição (opcional), lista de jogadores ouvintes e os dois jogadores líderes que responderão às perguntas representando suas equipes (opcional). Não há restrição quanto ao perfil do ouvinte e líder, podendo ser estudante ou professor. A competição não precisa ter início imediato.	Essencial
[RF008] Editar competição	O sistema deve permitir que o professor edite as informações de uma competição. Todos os campos poderão ser editados.	Essencial
[RF009] Listar competições criadas	O sistema deve permitir que o professor visualize uma lista com todas suas competições criadas. Na listagem, por competição, deve conter o nome da competição, data e descrição.	Essencial
[RF010] Visualizar competição	O sistema deve permitir que o professor visualize todas as informações de uma competição que seja de sua autoria. Ao visualizar poderá ser visto: Nome da competição, data da realização, descrição, questões da competição, jogadores ouvintes, jogadores líderes e o resultado da competição.	Essencial
[RF011] Jogar partida solo	O sistema deve permitir que um usuário (estudante ou professor) jogue no modo solo, podendo iniciar a partida a qualquer momento. Para a partida serão guardadas as seguintes informações: Nome do jogador, data de início do jogo, a lista de questões e o resultado da partida.	Essencial
[RF012] Jogar partida duo	O sistema deve permitir que um jogador (estudante ou professor) convide um amigo para jogar no modo duo. Para a partida serão guardadas as informações: Nome dos dois jogadores, data de início do	Essencial

		jogo, lista de questões e o resultado da partida.	
[RF013]	Encerrar partida	O sistema deve permitir que um jogador interrompa a partida a qualquer momento. Sendo no modo solo, o jogo é encerrado. Sendo no modo duo, o jogo encerra e o jogador restante é vencedor. Sendo no modo competição, caso o líder desista a equipe pode nomear outro, mas se isso não ocorrer a competição é encerrada e a equipe restante é vencedora.	Essencial
[RF014]	Excluir competição	O sistema deve permitir que o professor exclua uma competição de sua autoria.	Importante
[RF015]	Excluir questão	O sistema deve permitir que o professor exclua uma questão de sua autoria.	Importante
[RF016]	Excluir conta	O sistema deve permitir que o jogador exclua sua conta na aplicação.	Importante
[RF017]	Ver de histórico de partidas	O sistema deve permitir que o jogador consiga ver uma listagem contendo todo seu histórico de partidas (solo, duo e competição). Para cada partida na lista exibe a data, o modo de jogo e o vencedor.	Desejável
[RF018]	Ver de informações de uma partida	O sistema deve permitir que o jogador consiga ver as informações de uma partida em específico. Para cada partida deve ser exibido a data, modo de jogo, o vencedor e o desempenho geral do jogador que está visualizando (quantidade de erros e acertos).	Desejável
[RF019]	Editar informações do perfil	O sistema deve permitir que o usuário (jogador ou professor) modifique suas informações do perfil. Todos os campos são editáveis.	Desejável
[RF020]	Pausar jogo solo	O sistema deve permitir que um jogador pause um jogo solo. O progresso é salvo e pode ser retomado a qualquer momento.	Desejável

Fonte: Autoria própria (2024).

4.2 Requisitos não funcionais

A seguir serão apresentados os requisitos não funcionais do *quiz* educacional. Um requisito é classificado como não funcional quando esse vier a elucidar uma qualidade ou restrição do *software* (Bass, Clements e Kazman, 2012).

Os requisitos não funcionais estão, portanto, diretamente relacionados à qualidade da aplicação. Por isso, a fim de definir os requisitos não funcionais de maneira concisa e criteriosa, a ISO/IEC 25010 foi tomada como base, de modo que os requisitos não funcionais do *quiz* educacional foram definidos e classificados em categorias conforme os atributos de qualidade apontados neste padrão ISO.

Pelo Quadro 2 pode ser observado que cada requisito possui um código, um nome, uma descrição e uma categoria. No que diz respeito à montagem do código, o "RNF" refere-se à "Requisito não funcional" e o número aponta a sequência. Já na categoria, como mencionado de maneira breve anteriormente, elucida a qual categoria da ISO o requisito não funcional em questão se encaixa.

Quadro 2 - Lista dos requisitos não funcionais do *quiz* educacional.

Requisito	Descrição	Categoria
[RNF001] Disponibilidade	A aplicação deve ser consistente e, por isso, deve estar disponível para uso quando necessário.	Confiabilidade
[RNF002] Integridade e proteção dos dados	A aplicação deve ser capaz de garantir que os dados poderão ser acessados apenas por usuários autorizados.	Segurança
[RNF003] Capacidade de incorporação de novas funcionalidades	A aplicação deve ser capaz de ter novas funcionalidades incorporadas, sem que afete as demais já existentes.	Manutenibilidade
[RNF004] Interface com boa interação	A aplicação deve possuir uma <i>interface</i> que possibilite uma interação atraente e satisfatória, sem grandes problemas.	Usabilidade
[RNF005] Facilidade de instalação	A aplicação deve possibilitar que a instalação ou desinstalação seja feita com facilidade.	Portabilidade
[RNF006] Facilidade para operar	A aplicação deve ser fácil de ser operada e controlada.	Usabilidade

Fonte: Autoria própria (2024).

5 VISÕES ARQUITETURAIS

Um sistema pode ser visto de maneiras distintas, dependendo da perspectiva de quem o analisa, seja um desenvolvedor ou outro interessado. Nesse contexto, existem as visões arquiteturais, que auxiliam na representação de um *software* de modo a torná-lo entendível de diferentes perspectivas. Há três principais categorias, que agrupam as estruturas arquitetônicas: as estruturas de módulo, estruturas de componente & conector e estruturas de alocação (Bass, Clements e Kazman, 2012).

5.1 Visão de Módulo

A visão de módulo é capaz de definir a forma como o código-fonte do *software* será organizado em termos de elementos estruturais (Bass, Clements e Kazman, 2012). Para Bass, Clements e Kazman (2012), nesse tipo de estrutura, os elementos presentes são os módulos, que podem ser classes, camadas, etc.

O diagrama de classes da UML (*Unified Modeling Language*) pode ser usado para representar uma estrutura de módulo, e será usado no *quiz* educacional. Na Figura 1, é apresentada a organização das classes e pacotes do sistema por meio do diagrama de classes UML.

O diagrama de classes foi organizado de modo a separar por pacote os gerenciadores de cada classe, que compõem o sistema, além de um pacote contendo todas as entidades e um outro pacote com as classes responsáveis por realizar o acesso ao banco de dados.

No pacote das entidades, é possível visualizar que, cada entidade possui seus próprios atributos, e nos pacotes dos gerenciadores das classes é possível visualizar os principais métodos de cada módulo do sistema. Desta maneira, tem-se classes pequenas, mas com uma única responsabilidade (i.e., a classe de questão se preocupa unicamente com as operações a respeito das questões do jogo) e com uma maior capacidade de escalabilidade, bem como

O diagrama de componentes é comumente usado quando se quer modelar grandes sistemas, em razão da sua capacidade de permitir dividir o *software* em subsistemas gerenciáveis, de tal maneira que detalhes internos que remetem à implementação permanecem ocultos (Pilone & Pitman, 2005). Na Figura 2, é possível visualizar a visão Componente & Conector proposta para o *quiz* proposto.

Ainda de acordo com Pilone e Pitman (2005), o diagrama de componentes é útil para evidenciar de que maneira a aplicação se constitui em alto nível, dividido-a em componentes, considerando que cada componente possui distintas responsabilidades. Ademais, sua estrutura contribui para a definição das estimativas de desenvolvimento e em como a entrega do *software* pode ser fracionada. Tomando como base essas considerações, na modelagem proposta (Figura 2), os componentes são grandes módulos que objetivam gerenciar cada entidade existente no sistema (como Usuário, Partida, etc), justamente organizados, visando manter a coesão entre eles, respeitando as responsabilidades dessemelhantes.

O componente denominado “Gerenciador usuário” permite realizar algumas manipulações com o usuário da aplicação, seja ele um estudante ou professor, a exemplo de realizar *login*, realizar *logout*, cadastrar, excluir conta e editar perfil. O componente “Gerenciador partida” está relacionado com a possibilidade de jogar; e, por isso, ele gerencia operações como jogar, editar, visualizar, buscar, listar todas as partidas e encerrar partida. Já o componente Banco de dados possibilita a manipulação dos dados advindos do banco de dados. Se tratando dos demais componentes (os gerenciadores de alternativa, questão, resultado partida e área ensino), estes dispõem das operações básicas que possibilitam a manipulação de tais entidades, como cadastro, edição, visualização, listagem, exclusão e busca.

5.3 Visão de Alocação

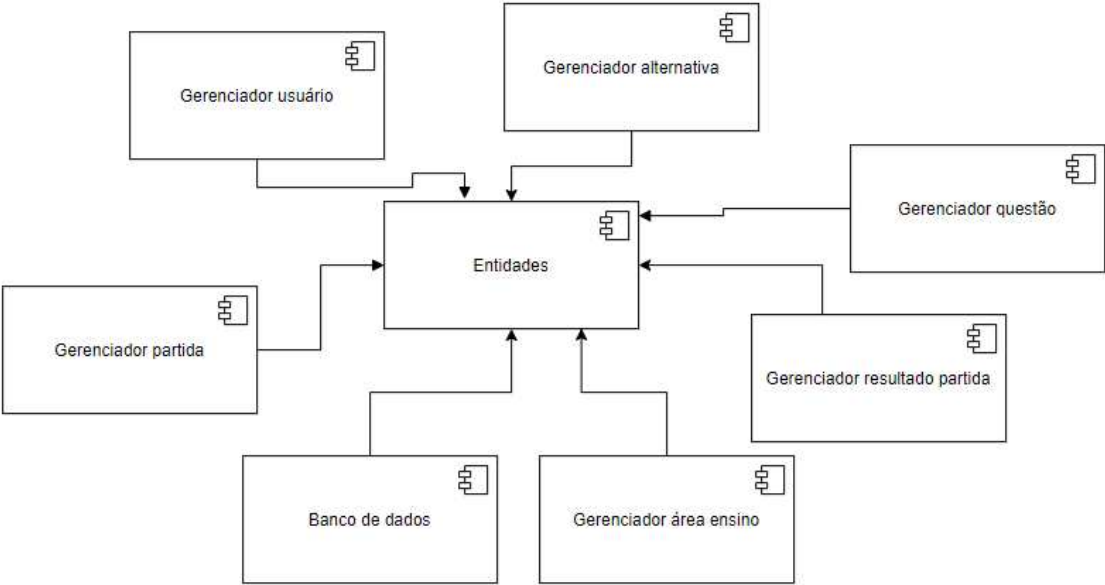
Com a visão de alocação se tem uma dimensão de como se dá o relacionamento entre o *software* e a infraestrutura de *hardware* (e.g., equipes, sistemas de arquivos, aparelhos dos usuários, etc) (Bass, Clements e Kazman, 2012). A visão de alocação pode ser retratada fazendo uso do diagrama de implantação, que foi o escolhido para modelar o *quiz* educacional, com relação a essa visão.

Na Figura 3, pode ser visualizado o diagrama de implantação proposto para a aplicação em questão, que visa explicitar de que maneira o *hardware* e o *software* envolvidos no *quiz* interagem.

A aplicação poderá ser acessada pelo usuário — que pode ser um professor ou estudante — através de um dispositivo móvel (considerando que o usuário possui a aplicação descarregada), que no diagrama pode ser visto sendo representado pelo nó denominado “Dispositivo móvel”. Já a aplicação, no que lhe concerne, interage com o nó “Servidor de aplicação”, no qual está contida toda a lógica de negócio da aplicação, os serviços de API, dentre outros. Por fim, o servidor da aplicação se relaciona com o “Servidor de banco de dados”, que objetiva armazenar os dados que se fazem necessários para o funcionamento do sistema.

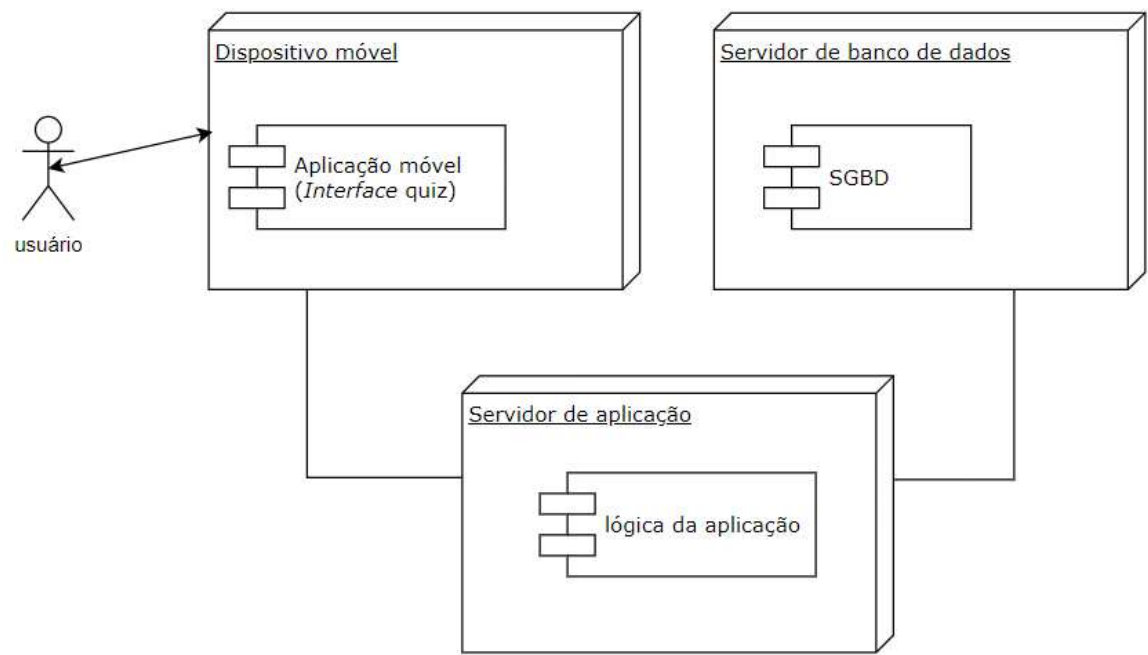
Apesar da simplicidade e concisão, os diagramas de implantação se demonstram importantes quando se deseja esclarecer em quais partes do *hardware* estarão sendo implantadas determinadas partes de *software* (Fowler, 2018).

Figura 2 – Diagrama de componentes.



Fonte: Autoria própria (2024).

Figura 3 – Diagrama de implantação.



Fonte: Autoria própria (2024).

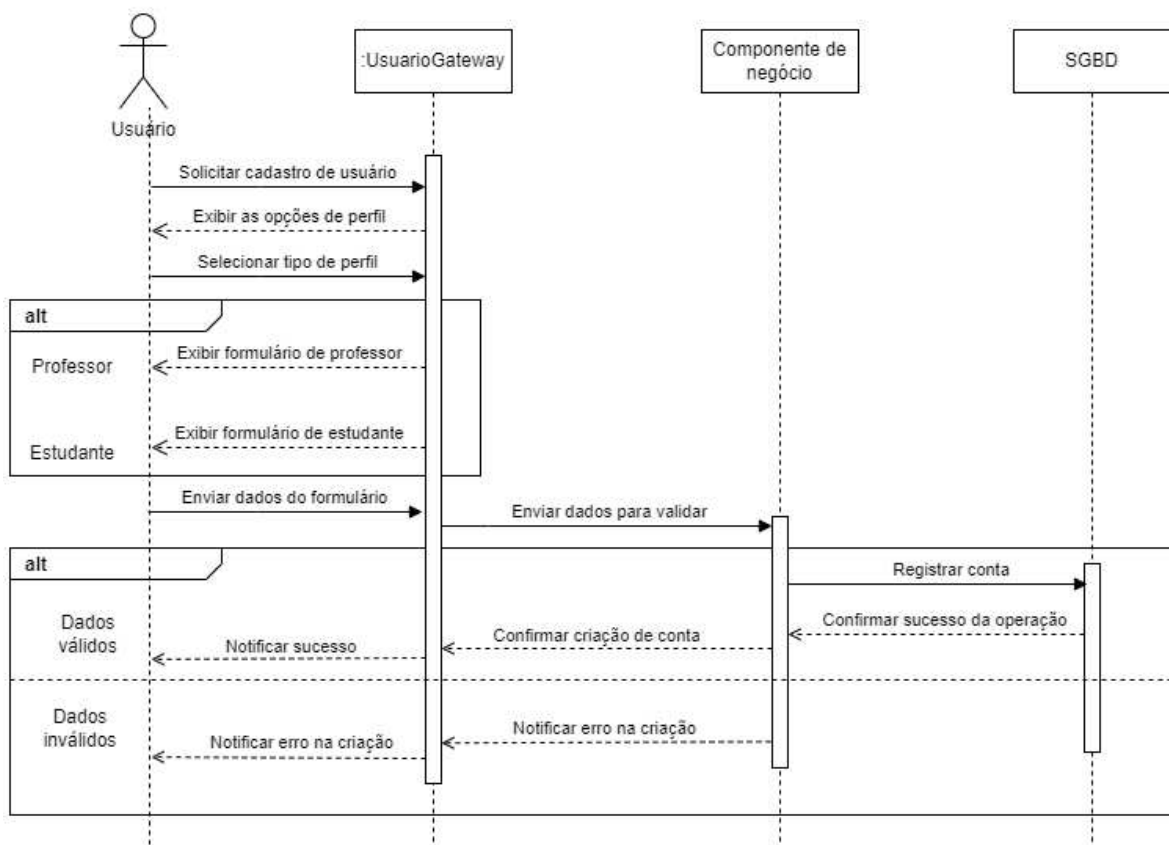
6 AVALIAÇÃO

A avaliação arquitetural busca deliberar se a arquitetura proposta para um determinado sistema está condizente com o esperado, isto é, se o que foi idealizado e posteriormente projetado, quando aplicado ao *software*, permite que este supra as necessidades dos interessados (Bass, Clements e Kazman, 2012).

A avaliação da arquitetura do *quiz* se dará fazendo uso de cenários, que é um elemento presente no método de avaliação denominado *Architecture Tradeoff Analysis Method* (ATAM), que possibilita, inclusive, que a arquitetura seja avaliada por avaliadores que não conhecem muito da arquitetura e a respeito dos objetivos de negócios (Bass, Clements e Kazman, 2012). Os cenários, representados por diagramas de sequência, serão explicados a seguir.

Na Figura 4 pode ser visualizado o diagrama de sequência que diz respeito ao cenário para cadastrar um usuário. Neste cenário, o usuário solicita o início do cadastro e deve escolher o tipo de perfil (professor ou estudante) e, em seguida, o formulário correspondente à opção de perfil escolhida é mostrado. Após o envio dos dados, o componente de negócio — responsável pela lógica de negócios e camada de processamento — valida os dados (em termos de tipo correto dos dados e para identificar se o usuário já não possui um cadastro). Após a validação, existem 2 fluxos possíveis (que no diagrama são representados pela região retangular com a *tag alt*): 1) Se os dados forem válidos, a conta do usuário do tipo escolhido é criada e o usuário recebe o *feedback* positivo; 2) Se os dados forem inválidos, o componente de negócio notifica o erro e o usuário final vai ser informado, dessa vez com *feedback* negativo, para fazer as devidas correções.

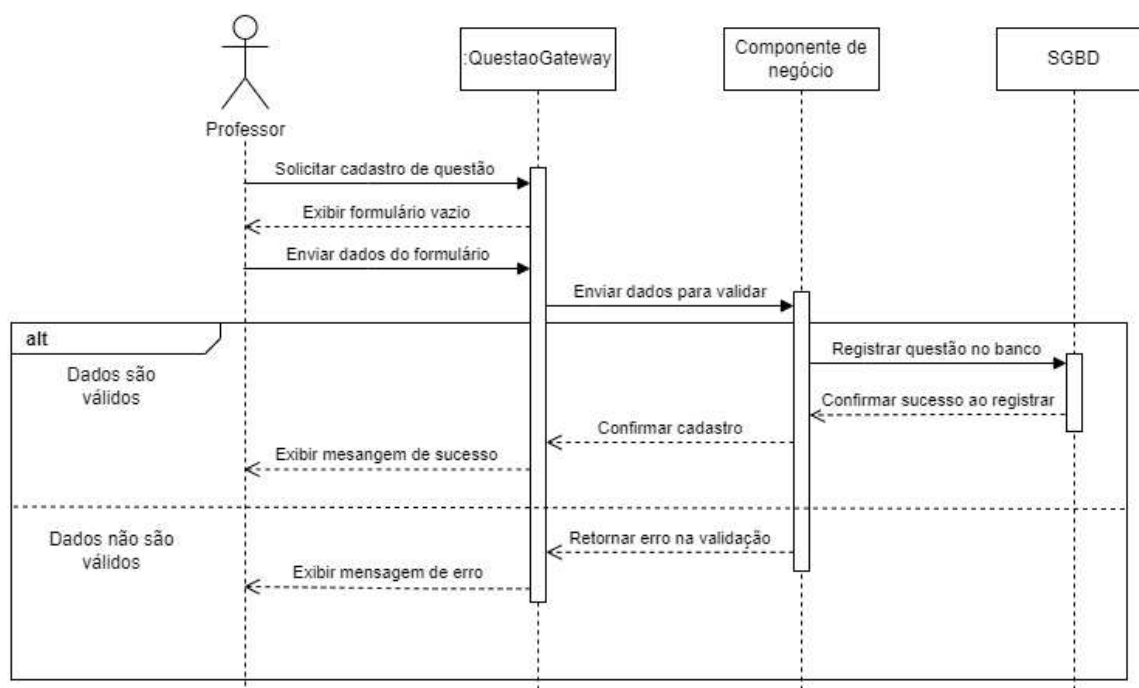
Figura 4 – Diagrama de sequência - Cadastrar professor ou estudante.



Fonte: Autoria própria (2024).

Na Figura 5 é feita a demonstração do cenário de cadastro de questões. No cenário referente ao cadastro de questão, o usuário do tipo professor acessa o formulário para cadastro que está contido na *interface* "QuestaoGateway". Uma vez que o formulário é preenchido e enviado, o componente de negócio faz a validação dos dados. Na validação ocorre semelhante ao cadastro de usuário, em que existem 2 fluxos, pois caso os dados sejam válidos, a questão é persistida no banco com sucesso, mas caso seja encontrada alguma inconsistência uma mensagem de erro é retornada e a questão não é cadastrada. Em ambos os casos, o usuário recebe um *feedback*.

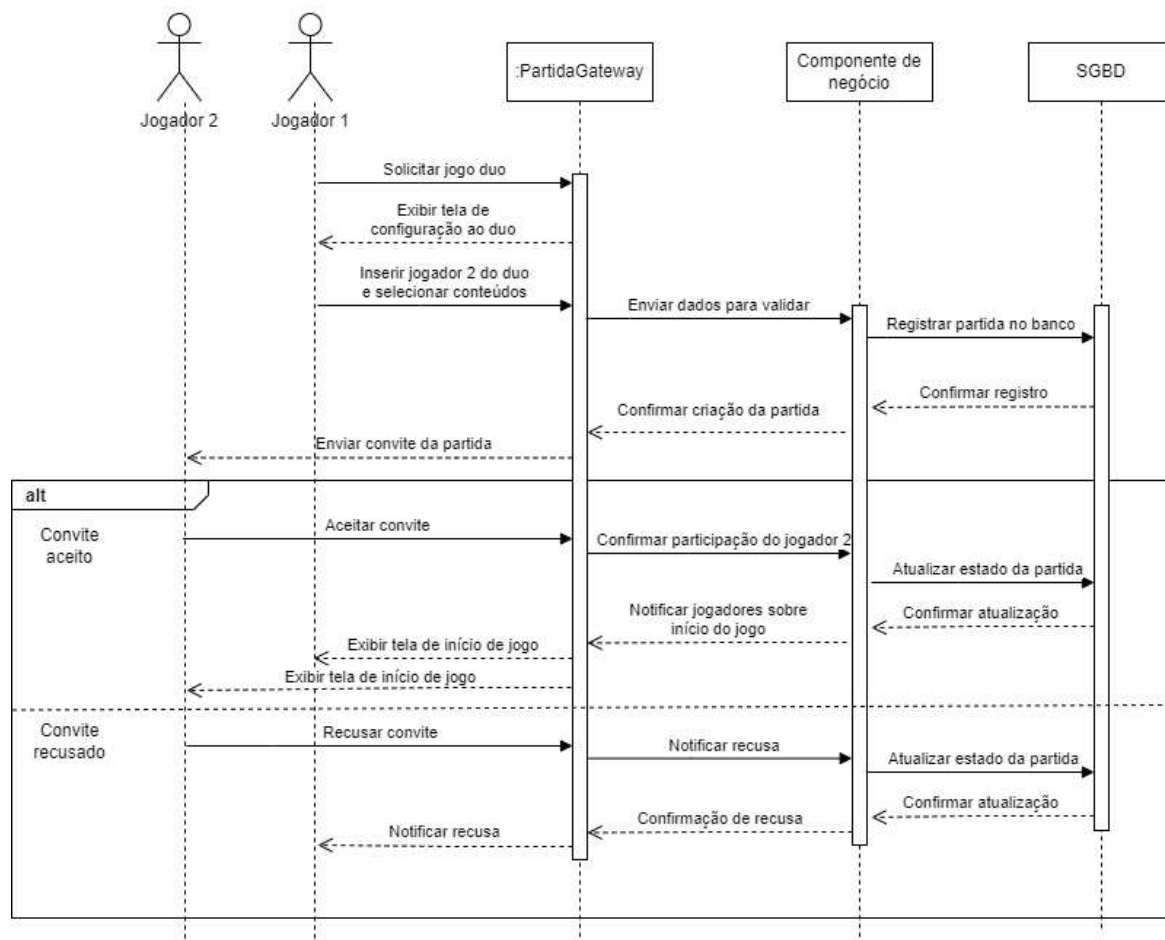
Figura 5 – Diagrama de sequência - Cadastrar questão.



Fonte: Autoria própria (2024).

Outro cenário que pode ser visto é referente à criação de uma partida no modo *duo* (Figura 6). Para começar, o jogador 1 começa criando um jogo no modo *duo* e para isso insere um jogador 2, que será convidado posteriormente e seleciona os conteúdos a serem estudados durante a partida. Após o preenchimento, os dados são validados, a partida é registrada no banco e, somente após a devida criação é que o convite é enviado para o jogador 2. O início do jogo fica condicionado ao aceite do jogador 2, por isso no diagrama se pode ver 2 fluxos, sendo que o superior considera que o convite foi aceito e inferior considera que o convite foi recusado. No primeiro caso, após ser aceito, as informações da partida são processadas no componente de negócio e atualizadas no banco, para, logo em seguida, exibir a tela de início do jogo. Já no caso da recusa, as informações da partida são atualizadas no banco e o jogador 1 é avisado.

Figura 6 – Diagrama de sequência - Criação do jogo modo duo.



Fonte: Autoria própria (2024).

7 RESULTADOS

A seguir, serão apresentados os resultados obtidos nas etapas de elicitação de requisitos, modelagem da arquitetura e avaliação da arquitetura do *quiz* educacional.

7.1 Elicitação de Requisitos

Este trabalho foi realizado em colaboração com uma escola da rede municipal de ensino localizada na região do Alto Oeste Potiguar, no estado do Rio Grande do Norte, Brasil. O objetivo principal era desenvolver um *quiz* educacional para alunos da educação básica, visando suprir as necessidades específicas identificadas na escola parceira.

Durante a fase inicial da pesquisa, foram realizadas entrevistas com educadores e alunos para entender os desafios enfrentados em sala de aula ao utilizar jogos sérios e as razões pelas quais outras ferramentas semelhantes não atendiam às suas necessidades. As entrevistas revelaram a necessidade de uma plataforma que oferecesse: 1) Múltiplos modos de jogo (solo, duo e competição); 2) Modo de competição inovador, que permite o envolvimento simultâneo de todos os alunos, com alguns atuando como representantes em tempo real durante as disputas; 3) Autonomia para professores gerenciarem questões e competições.

Além disso, foram identificados requisitos não funcionais como: Disponibilidade, integridade e proteção dos dados dos usuários, capacidade de receber novas funcionalidades,

interface intuitiva e de fácil navegação para diferentes perfis de usuários, facilidade de instalação e facilidade de operação.

7.2 Modelagem da Arquitetura

A arquitetura do *quiz* educacional foi modelada utilizando a UML, por meio de práticas consolidadas de *design* de *software*, e mostrando 3 visões distintas (visão de Módulo, Componente & Conector e de Alocação) a fim de representar visualmente os componentes do sistema e suas interações. Os diagramas de classes, componentes e alocação foram utilizados para ilustrar a estrutura e o comportamento da aplicação.

O diagrama de classes (Figura 1), que representa a Visão de Módulo do sistema, organiza as principais classes e pacotes de forma a promover a separação modular, seguindo os princípios propostos por Bass, Clements e Kazman (2012). A estrutura modular facilita a manutenção e escalabilidade, permitindo que cada classe mantenha uma única responsabilidade. A classe “Partida” se destaca como classe base, sendo estendida pelos modos *Solo*, *Duo* e *Competição*, refletindo a flexibilidade da aplicação. A arquitetura prioriza classes pequenas e coesas, alinhando-se às boas práticas defendidas por Martin (2009), o que torna o sistema mais fácil de modificar e expandir.

O diagrama de componentes (Figura 2), que representa a Visão Componente & Conector, evidencia a divisão do sistema em grandes módulos que gerenciam diferentes entidades, facilitando a coesão e clareza das responsabilidades. Essa divisão em componentes de alto nível contribui para a modularidade do *software* e permite uma entrega e desenvolvimento mais eficientes, alinhando-se às melhores práticas de organização de sistemas complexos.

O diagrama de implantação (Figura 3), que representa a Visão de Alocação, demonstra a interação entre o sistema e a infraestrutura de *hardware*, permitindo uma visão clara da distribuição dos componentes da aplicação. A arquitetura projetada promove uma comunicação eficiente entre os dispositivos dos usuários e os servidores, garantindo a escalabilidade e a fácil manutenção do sistema, conforme os requisitos técnicos e operacionais estabelecidos.

Dessa forma, os diagramas expostos não apenas facilitam a visualização da arquitetura do sistema, mas também contribuem para assegurar que o jogo sério proposto possa atender a todas as necessidades de forma integrada.

7.3 Avaliação da Arquitetura

A arquitetura proposta foi avaliada utilizando o método ATAM, que visa identificar os riscos e benefícios arquiteturais em relação aos cenários de qualidade prioritários. Os cenários de cadastro de usuário, cadastro de questão e criação de partida duo foram os selecionados para avaliação.

A Figura 4 na seção de Avaliação ilustra o cenário de cadastro de usuário. A arquitetura demonstra contemplar os fluxos principais, como a escolha do tipo de perfil (professor ou estudante), o preenchimento do formulário correspondente e a validação dos dados. A validação, por sua vez, verifica o tipo de dado e a existência de cadastros duplicados, garantindo a qualidade dos dados e a segurança das informações.

A usabilidade também é evidenciada nesse cenário de cadastro de usuário. A solicitação do tipo de perfil antes da exibição do formulário demonstra a intenção de proporcionar uma experiência mais intuitiva, simplificando o processo e evitando informações irrelevantes. Esse processo de usabilidade também é refletido em outros cenários, como o cadastro de questões.

O cenário de cadastro de questões, apresentado na Figura 5, destaca a interação do professor com a interface "QuestaoGateway" para inserir novas questões no sistema. A arquitetura repete a estrutura de validação presente no cadastro de usuário, verificando a consistência dos dados antes da persistência no banco de dados. Essa abordagem garante a confiabilidade do sistema ao evitar questões inválidas ou inconsistentes.

A Figura 6 ilustra o cenário de criação de partida no modo duo. A arquitetura contempla o processo de criação da partida pelo jogador 1, incluindo a escolha do jogador 2, a seleção de conteúdos e a validação dos dados. A gestão do convite enviado ao jogador 2, com fluxos alternativos para aceite ou recusa, demonstra a capacidade da arquitetura em lidar com a interação assíncrona entre usuários. Ademais, a atualização das informações da partida no banco de dados após o aceite ou recusa do convite garante a consistência dos dados e permite que o sistema responda adequadamente à decisão do jogador 2.

A análise dos cenários demonstra que a arquitetura proposta para o sistema de *quiz* apresenta uma estrutura sólida e capaz de suportar as funcionalidades-chave do sistema. A atenção dedicada à validação de dados, à gestão de diferentes fluxos de interação e à usabilidade reflete a preocupação em atender às necessidades dos usuários e garantir a qualidade do *software*. Além disso, a arquitetura foi projetada para assegurar a segurança dos dados dos usuários, oferecendo um sistema que não apenas é funcional, mas também confiável e intuitivo, alinhando-se aos requisitos de usabilidade e segurança definidos.

Em suma, a arquitetura proposta apresenta-se como um alicerce sólido para o desenvolvimento do sistema de *quiz*, com potencial para atender às expectativas dos usuários e alcançar os objetivos de negócio.

8 CONSIDERAÇÕES FINAIS

O presente artigo teve como objetivo propor uma arquitetura para o *quiz* educacional, um "jogo sério", que tem como objetivo contribuir com o ensino-aprendizagem, uma vez que ele possibilita que estudantes e professores possam usufruir do *software* (cada um com seu escopo próprio de responsabilidades), concedendo um ambiente virtual seguro e que propicia a aquisição de conhecimento e o desenvolvimento de habilidades, no simples ato de jogar.

Todas as necessidades e características da aplicação em questão foram levantadas, seguindo as técnicas de elicitação de requisitos, que melhor se encaixam no contexto de desenvolvimento do *quiz*, que neste caso foram as entrevistas realizadas de maneira remota e questionários. Depois, de posse das informações coletadas, foi possível organizá-las transformando-as de fato no que foi apresentado como sendo os requisitos funcionais e não funcionais. Como etapa posterior, dado que se criou uma visualização do que faz o *software*, produziu-se diferentes diagramas, como o diagrama de classes, componentes, implantação e de sequência, possibilitando a visualização da aplicação sob diferentes perspectivas. Por fim, a avaliação arquitetural foi apresentada, com a exemplificação de três cenários.

No que se refere a trabalhos futuros, é pretendido prosseguir na evolução da aplicação focando na criação da *interface* da aplicação, tomando como base a arquitetura proposta e seguindo as devidas técnicas que permeiam o processo de prototipação para identificar pontos de melhorias e entender se está em conformidade com o propósito do jogo.

9 AGRADECIMENTOS

Agradecemos ao grupo de pesquisa LIS — Laboratório de Inovações em *Software* e LISA — Laboratório de Inovações em *Software* e Automação, pelo apoio na produção deste trabalho, e à Universidade Federal Rural do Semi-Árido - UFERSA pelo financiamento, por meio da Pró-Reitoria de Pesquisa e Pós-Graduação (PROPPG) através do Edital PROPPG N° 22/2024 - Iniciação Científica – Ações Afirmativas e PROPPG N° 22/2024 - Iniciação Científica – Ações Afirmativas e PROPPG N° 21/2024 - Iniciação Científica.

REFERÊNCIAS

BASS, L.; CLEMENTS, P. e KAZMAN, R. **Software architecture in practice**. Addison - Wesley professional, 2012.

BUSARELLO, R. I. **Gamification: princípios e estratégias**. Pimenta Cultural, 2016.

DOMÍNGUEZ, A. et al. Gamifying learning experiences: Practical implications and outcomes. Gamifying learning experiences: Practical implications and outcomes. **Computers & Education**, v. 63, n. 1, p. 380–392, 2013.

FOWLER, Martin. **UML distilled: a brief guide to the standard object modeling language**. Addison-Wesley Professional, 2018.

MARTIN, Robert C. **Clean code: a handbook of agile software craftsmanship**. Pearson Education, 2009.

MITAMURA, T.; SUZUKI, Y.; OOHORI, T. Serious games for learning programming languages. **International Conference on Systems, Man, and Cybernetics**, p. 1812–1817, 2012.

MOREIRA, M. A. **Teorias de aprendizagem**. São Paulo: Editora pedagógica e universitária, 1999.

PILONE, Dan; PITMAN, Neil. **UML 2.0 in a Nutshell**. O'Reilly Media, Inc., 2005.

SOMMERVILLE, Ian. **Software engineering**. America: Pearson Education Inc, 2011.