PEER REVIEW, Vol. 6, Nº 9, 2024
DOI: 10.53660/PRW-2145-4001
ISSN: 1541-1389

Arquitetura de uma Ferramenta para Controle e Rastreio de Artefatos de Software

Architecture of a Tool for Controlling and Tracking Software Artifacts

Emanuel Bruno Duarte de Morais
ORCID: https://orcid.org/0009-0007-5005-353X
Universidade Federal Rural do Semi-Árido, Brasil
E-mail: emanuelbruno2018vasc@gmail.com
Taísso Reni de Souza Melo
ORCID: https://orcid.org/0009-0002-7917-2077
Universidade Federal Rural do Semi-Árido, Brasil
E-mail: taisso.melo@gmail.com
Reudismam Rolim de Sousa
ORCID: https://orcid.org/0000-0001-9728-0130
Universidade Federal Rural do Semi-Árido, Brasil
E-mail: reudismam.sousa@ufersa.edu.br

RESUMO

O controle e rastreio dos artefatos de software gerados para os requisitos elaborados para um sistema de software são tarefas necessárias no desenvolvimento de sistemas de software. No entanto, essas tarefas podem se tornar de difícil controle pelos diversos sistemas envolvidos na produção dos artefatos do software, a exemplo de ferramentas para design visual, gerência de tarefas, geração de diagramas, etc. Buscando minimizar essa problemática, a proposta deste trabalho é o Express Requirement Documents ou ExpressRD, que busca reunir esses diferentes artefatos em uma única ferramenta. Em especial, este trabalho busca propor uma arquitetura do ambiente proposto. Ao longo do trabalho, foram apresentadas três diferentes visões arquiteturais do software (Visão de Módulo, Componente & Conector e Visão de Alocação). A arquitetura foi avaliada com uma metodologia baseada em cenários e pela implementação do *back-end* do sistema. Como resultados, as avaliações demonstraram que a arquitetura do sistema está alinhada com os requisitos levantados e é viável de implementação.

Palavras-chave: Software; Sistema; Requisitos; Plataformas de Gerenciamento de Software.

ABSTRACT

Controlling and tracking the software artifacts generated for the requirements developed for a software system are necessary tasks in the development of software systems. However, these tasks can become difficult to control by the various systems involved in the production of software artifacts, such as tools for visual design, task management, diagram generation, etc. Seeking to minimize this problem, the proposal of this work is Express Requirement Documents or ExpressRD, which aims to bring together these different artifacts in a single tool. In particular, this work seeks to propose an architecture for the proposed environment. Throughout the work, three different architectural views of the software were presented (Module, Component & Connector and Allocation View). The architecture was evaluated using a scenario-based methodology and by implementing the system's back-end. As a result, the evaluations demonstrated that the system architecture is aligned with the requirements identified and is viable for implementation.

Keywords: Software; System; Requirements; Software Management Platforms.

Recebido: 03/04/2024 | Aceito: 05/05/2024 | Publicado: 08/05/2024

INTRODUÇÃO

Com a crescente complexidade dos projetos de software, é necessário buscar eficiência no planejamento da construção dos produtos (SOMMERVILLE, 2011). Nas etapas iniciais, é crucial compreender o problema, elencar estratégias para reduzir custos, realizar estimativas de tempo, identificar riscos e facilitar a comunicação entre as diversas áreas envolvidas, tornando essencial um bom planejamento para garantir a sustentabilidade do projeto, de forma a permitir que se preveja com clareza as demais fases do desenvolvimento do software (SOMMERVILLE, 2011).

Neste sentido, o planejamento é crucial para garantir a sustentabilidade da elaboração do projeto, para guiar as demais fases do desenvolvimento do software (SOMMERVILLE, 2011). Ao compreender o problema é possível montar estratégias para direcionar os esforços da equipe de desenvolvimento de maneira eficaz.

Uma das etapas principais do planejamento é o levantamento dos requisitos (SOMMERVILLE, 2011), que elenca as funcionalidades do software (requisitos funcionais) e seus atributos de qualidade e restrições (requisitos não funcionais) (BASS et al, 2012). Ao longo da gerência de um projeto de software, é necessário capturar aspectos associados a esses requisitos, tais como telas de um requisito específico, etapa do desenvolvimento de um requisito (a fazer, concluído, em teste, etc.). Para capturar esses aspectos, frequentemente, é necessário o uso de diferentes ferramentas, tais como Figma (FIGMA, 2024), Trello (TRELLO, 2024), Jira (JIRA, 2024), etc. Dessa forma, pode se tornar complexo para o gestor de projetos manter o controle dos artefatos associados a cada requisito, nessas diferentes ferramentas.

Nesse contexto, a proposta com a solução é o desenvolvimento do sistema "Express Requirement Documents" ou "ExpressRD", que visa aprimorar a gestão dos requisitos. Em específico, a proposta deste trabalho é o desenvolvimento da arquitetura deste sistema, para guiar a implementação. A utilização de um software dedicado à gestão de requisitos é fundamental para a eficácia do planejamento, possibilitando uma análise detalhada e uma compreensão das interdependências entre eles. Além disso, um software voltado para a área de requisitos contribui para um produto direcionado às expectativas do cliente e às necessidades do mercado.

Com este sistema de software é possível agrupar diversos conteúdos que agregam a uma visão ampla do projeto, como os requisitos, diagramas, histórias do

usuário e diversas outras técnicas e ferramentas, estabelecendo um desenvolvimento mais fluido ao longo de todas as etapas do ciclo de vida do software. Deste modo, o ExpressRD é a estratégia abrangente para elevar a qualidade do produto e a satisfação do cliente, que permite uma documentação padronizada rapidamente e alinhada com as expectativas do mercado.

ARQUITETURA

Nesta seção apresentamos a arquitetura do sistema proposta. Uma arquitetura denota como um sistema está estruturado, apresentando seus elementos de software, as propriedades que podem ser vistas externamente do sistema e dos relacionamentos entre esses elementos (BASS et al., 2012; SANTOS et al., 2022). A arquitetura representa uma documentação do sistema, facilitando a comunicação entre os interessados no projeto (stakeholders) e permitindo a tomada de decisão (BASS et al., 2012; SANTOS et al., 2022).

REQUISITOS FUNCIONAIS

Para o desenvolvimento da arquitetura de um software é essencial se conhecer quais funcionalidades ela deve suportar. Nesta seção, serão apresentados os requisitos funcionais do sistema. Os requisitos seguirão o seguinte padrão de nomenclatura: “RFID - Nome do Requisito Funcional”. Em tal nomenclatura, o RF refere-se a Requisito Funcional e o número indica a sequência; o nome após o colchete simboliza o nome do requisito. Os requisitos foram definidos por meio de um questionário aplicado por Melo et al. (2024) para elencar as principais necessidades dos usuários no controle e rastreio dos requisitos. Dessa forma, os requisitos serão apresentados com a prioridade destes, conforme Quadro 1.

Quadro 1 - Lista de Requisitos funcionais. E: Essencial, I: Importante e D: Desejável

Requisito	Descrição	Pri.
[RF001] Cadastrar-se	Cadastro do usuário, preenchendo informações como, nome, e-mail, login e senha.	E
[RF002] Entrar	Permitir acessar a plataforma utilizando o login e senha	E

[RF003] Visualizar dashboard	O dashboard apresentará algumas das opções principais, como projetos para ordená-los (os projetos aparecerão na tela de dashboard), ver times compartilhados e ver favoritos.	E
[RF004] Editar dados	O usuário poderá editar suas informações de acesso informadas no cadastro e adicionar foto de perfil.	I
[RF005] Listar projetos	O usuário poderá ver todos os projetos criados, sendo informado título, data da última edição e capa. Realizar ordenações pela data da edição, data de criação, visualizados recentemente e por ordem alfabética do título. Filtrar por times e favoritos. Buscar por título do projeto.	E
[RF006] Criar novo projeto	Permitir criar um novo projeto com os dados: título, imagem capa representativa do projeto, descrição. A primeira etapa do projeto é chamada de rascunho e a etapa final de produto, contendo uma caixa de texto para adicionar uma rápida criação a partir da IA a ser implementada no projeto.	E
[RF007] Editar projeto	O usuário pode editar todos os dados referente ao projeto.	E
[RF008] Arquivar/Excluir projeto	O usuário pode arquivar ou excluir o projeto caso necessite, para isso é necessário um aviso de confirmação.	E
[RF009] Emitir relatório	Permitir emitir o relatório com os dados do projeto.	I
[RF010] Criar time	Permitir criar um time. O time é composto por pessoas vinculadas pelo *login + id unique* das contas. Elas poderão receber permissões para visualizar, editar e criar projetos.	I
[RF011] Visualizar notificações	Permitir visualizar notificações do time, convites para times e projetos e aviso de mudanças nos projetos.	I
[RF012] Favoritar projeto	Permitir favoritar projetos em que necessitar um destaque, clicando na estrelinha deixando-a com uma cor diferente.	I
[RF013] Listar requisitos	O usuário poderá ver uma lista com todos os requisitos, em que verá o id, titulo e prioridade do requisito.	E
[RF014] Excluir requisito	O usuário poderá excluir o requisito.	E
[RF015] Cadastrar novo requisito	Permitir adicionar os requisitos com os dados: título do requisito, descrição, prioridade, tempo estimado, designar a equipe a ser alocada (opcional), imagens e anexos de diagramas e suas descrições.	E
[RF016] Adicionar comentário	O usuário poderá deixar um comentário, demonstrando sua opinião sobre os requisitos com as seguintes informações: comentario, autor do comentário e data do comentário	D
[RF017] Responder comentário	Permitir responder ao comentário adicionado, com os dados: resposta, autor da resposta e data da resposta. A resposta deve estar vinculada ao comentário respondido.	D
[RF018] Vincular plataforma externa	O usuário poderá adicionar plataformas externas ao projeto, deixando o projeto mais robusto de informação, podendo adicionar	I

	links, anexos ou prints dessas plataformas. Alguns exemplos de plataformas são o Trello e Draw Io.	

Fonte: Autoria própria (2024)

REQUISITOS NÃO FUNCIONAIS

Esta seção apresenta os requisitos não funcionais do ExpressRD, que denotam restrições associadas aos serviços ou funções oferecidos pelo software (SOMMERVILLE, 2011). Os requisitos seguirão o seguinte padrão de nomenclatura: "[RNFId] Nome do requisito não funcional". Em tal nomenclatura, o RNF refere-se a Requisito Não Funcional e o número indica a sequência; o nome após o colchete simboliza a categoria que o requisito se encaixa, de acordo com a qualidade do produto de software (ISO/IEC 25010).

Quadro 2 - Lista de requisitos não funcionais. U: Usabilidade, I: Integração, S: Segurança, ME: Manutenibilidade e Escalabilidade e A: Atuação.

Requisito	Descrição	Cat.
[RNF001] Tema *light/dark*	O usuário pode optar pelos temas light ou dark, de forma que se adaptem melhor ao sistema.	U
[RNF002] IA para gerar o relatório	Ao gerar o relatório com base nas informações descritas no documento a IA deve organizar gerando os requisitos com base no que o usuário informou.	I
[RNF003] Proteção e privacidade	O sistema deve proteger contra acesso não autorizado.	S
[RNF004] Incremento de novas funcionalidades	O sistema deve ser feito de forma bem organizada para competir com outras plataformas no mercado, com documentação atualizada a cada sprint.	ME
[RNF005] Alta capacidade de usuários	O sistema deve ser capaz de distribuir o acesso dos usuários sem qualquer perda de desempenho.	A

Fonte: Autoria própria (2024)

CASOS DE USO

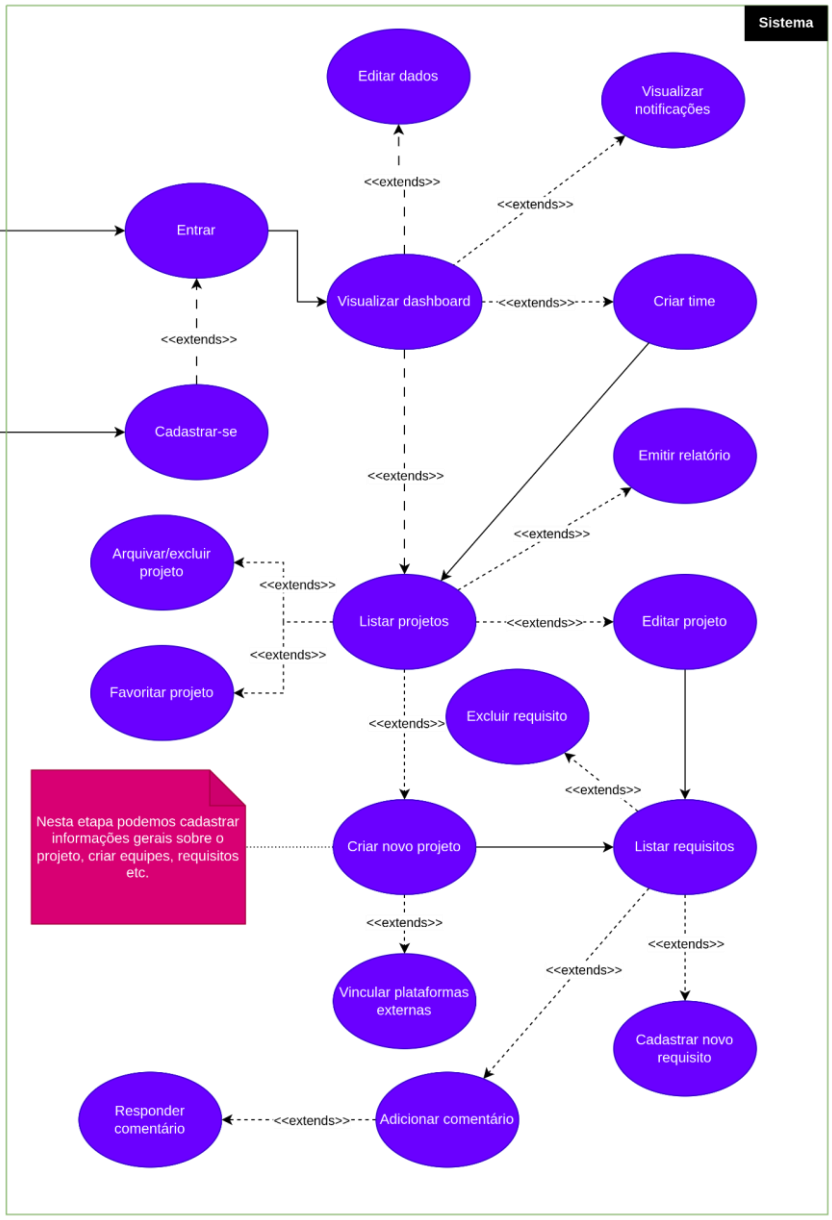
Os casos de uso representam as interações entre os atores e funcionalidades do sistema (GUEDES, 2018), sendo composto por atores, casos de uso e relacionamentos

(GUEDES, 2018). Os atores são os usuários que interagem com a aplicação, representados pelo símbolo de um boneco de palito (GUEDES, 2018). Os casos de usos das funcionalidades do sistema são geralmente construídos com verbos ou frases verbais para indicar uma ação, sendo representados pelo símbolo oval e com a frase no meio (GUEDES, 2018). Os relacionamentos são as interligações entre os atores e casos de uso (GUEDES, 2018). Na Figura 1 pode ser visto o diagrama de casos de uso para o ExpressRD.

VISÕES ARQUITETURAIS

As visões arquiteturais permitem visualizar o sistema sob diferentes perspectivas. Bass et. al. (2012) definem três visões principais (SANTOS et al., 2022): 1) Visão de Módulo, que representa como o código está estruturado e permite entender as responsabilidades dos elementos do software, a exemplo de classes e camadas (BASS et al., 2012; SANTOS et al., 2022); 2) Visão Componente & Conector, que representa os elementos do sistema em tempo de execução (serviços, clientes, servidores, etc.) e como se dá a interação entre os elementos em tempo de execução, a exemplo de mensagens, chamada de método, etc. (BASS et al., 2012; SANTOS et al., 2022); 3) Visão de Alocação, que define como os sistemas se integram a elementos que não são software, a exemplo de equipes de desenvolvimento, sistemas de arquivo, etc. (BASS et al., 2012; SANTOS et al., 2022).

Figura 1 - Diagrama de casos de uso

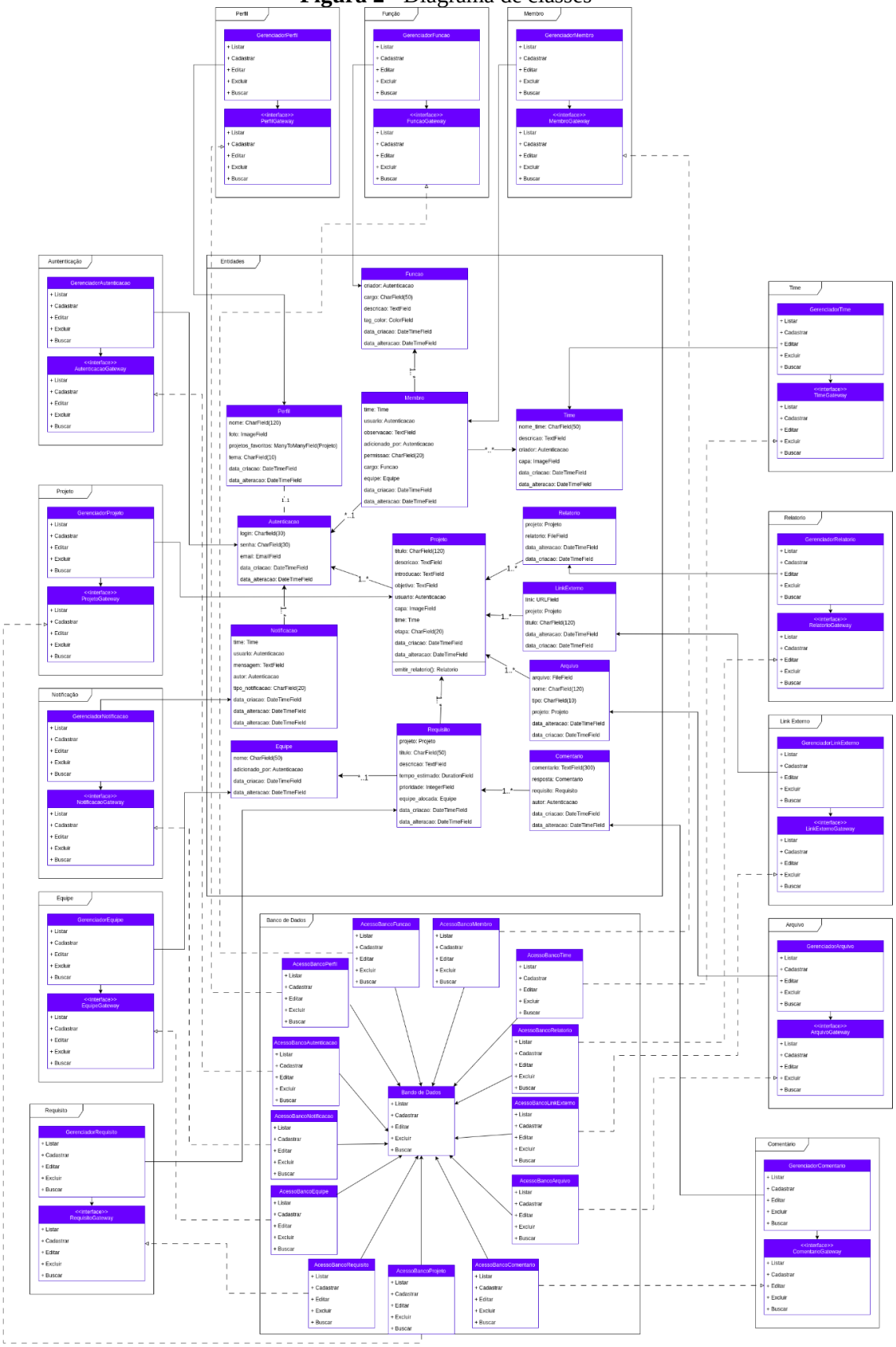


Fonte: Autoria própria (2024)

VISÃO DE MÓDULO

A Visão de Módulo pode ser representada por um diagrama de classes da UML (SANTOS et al., 2022). Com esse diagrama é possível entender diversos detalhes importantes na criação de classes, atributos, métodos e relacionamentos (GUEDES, 2018). O diagrama de classes para o ExpressRD pode ser visto na Figura 2.

Figura 2 - Diagrama de classes



Fonte: Autoria própria (2024)

O diagrama de classes possibilita visualizar os elementos estruturais, que internamente serão implementados no sistema (GUEDES, 2018). Alguns dos principais elementos estruturais de entidades do sistema são, a classe Autenticação, que permite lidar com os usuários que acessarão o sistema; a classe Projeto, que engloba o escopo de cada projeto dos usuários; a classe Requisito, associada ao projeto, que representa um requisito em um projeto. Os relacionamentos indicam como uma classe está ligada a outra (GUEDES, 2018). Esse tipo de modelagem é uma ferramenta que facilita também a comunicação entre as equipes (GUEDES, 2018). As classes de entidades possuem atributos e métodos específicos e possuem gerenciadores, para realizar operações associadas a elas. Os gerenciados podem manipular dados pelo acesso às classes que interagem com banco de dados.

VISÃO COMPONENTE & CONECTOR

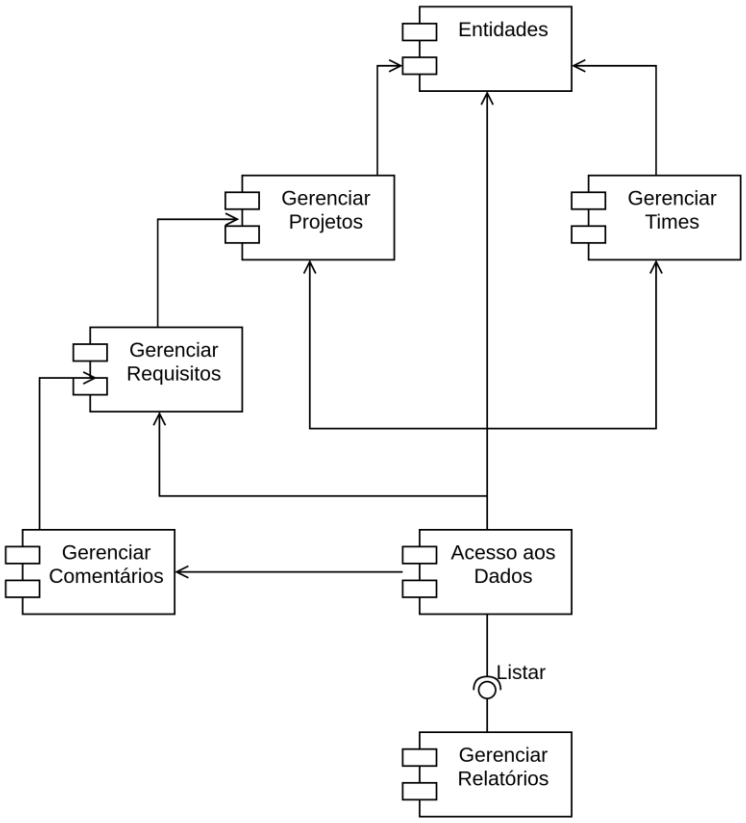
A Visão Componente & Conector pode ser representada por um diagrama de componentes, que permite visualizar os componentes de um sistema de software e seus relacionamentos (GUEDES, 2018). O diagrama de componentes para o sistema proposto pode ser visto na Figura 3.

O diagrama apresenta os componentes Entidades, Gerenciar Projetos, Gerenciar Times, Gerenciar Requisitos, Gerenciar Comentarios, Acesso aos Dados e Gerenciar Relatórios. Dessa forma, é possível se ter uma visão dos componentes e suas dependências.

O componente Gerenciar Projetos é acessado pelos usuários permitindo criar, editar, excluir e arquivar um projeto. O componente Gerenciar Times acrescenta permissões aos usuários que acessam o projeto. A gerência do time é feita pelo administrador do projeto. O componente Gerenciar Requisitos permite cadastrar, editar e excluir requisitos do projeto. O componente Gerenciar Comentários está atrelado aos requisitos; qualquer pessoa vinculada ao projeto pode criar, editar e excluir seus próprios comentários. O componente Gerenciar Relatórios permite gerar um documento contendo dados dos projetos, podendo ser acessado pelos usuários do projeto. O componente Acesso aos Dados permite manipular e recuperar dados do banco de dados. Uma vez que todos os componentes gerenciados precisam lidar com dados, há uma comunicação entre cada componente gerenciador com o componente Acesso aos Dados. Por exemplo, o

componente Gerenciar Comentários precisa listar e manipular comentários no banco de dados.

Figura 3 - Diagrama de componentes



Fonte: Autoria própria (2024)

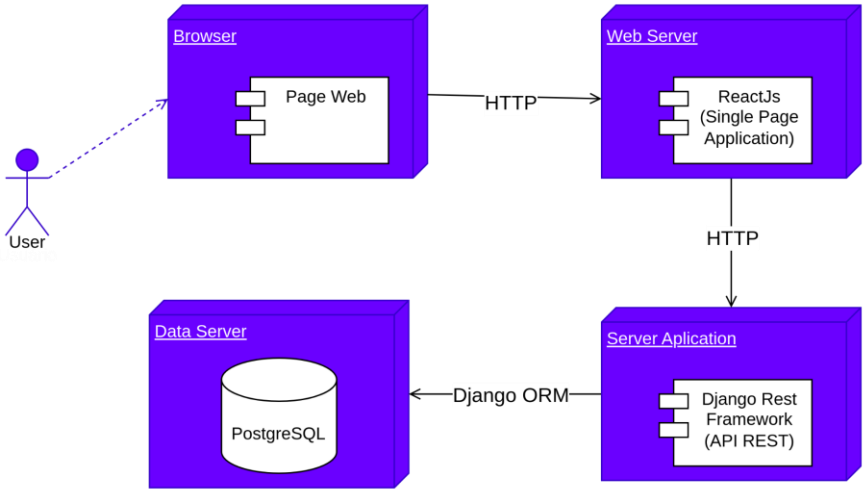
VISÃO DE ALOCAÇÃO

A Visão de Alocação pode ser modelada por diagrama de implantação. Este diagrama é uma representação visual dos componentes de configuração do sistema, definindo os artefatos físicos e virtuais e suas comunicações (GUEDES, 2018). A Visão de Alocação para o sistema proposto pode ser vista na Figura 4.

O usuário acessa o sistema por meio de uma interface WEB (por meio do navegador Web), que interage com um Servidor Web. Quando é necessário buscar ou armazenar informações no banco de dados, o Servidor Web se comunica com o Servidor de Aplicação por meio de uma API Rest, via requisições HTTP, que utiliza as funcionalidades do Django ORM para facilitar a comunicação com o banco de dados PostgreSQL. Essa abordagem melhora o desempenho do sistema, dividindo o sistema em

serviços de propósito específico, resultando em um código escalável e organizado, pela independência entre os componentes.

Figura 4 - Diagrama de implantação do ambiente



Fonte: Autoria própria (2024)

AVALIAÇÃO

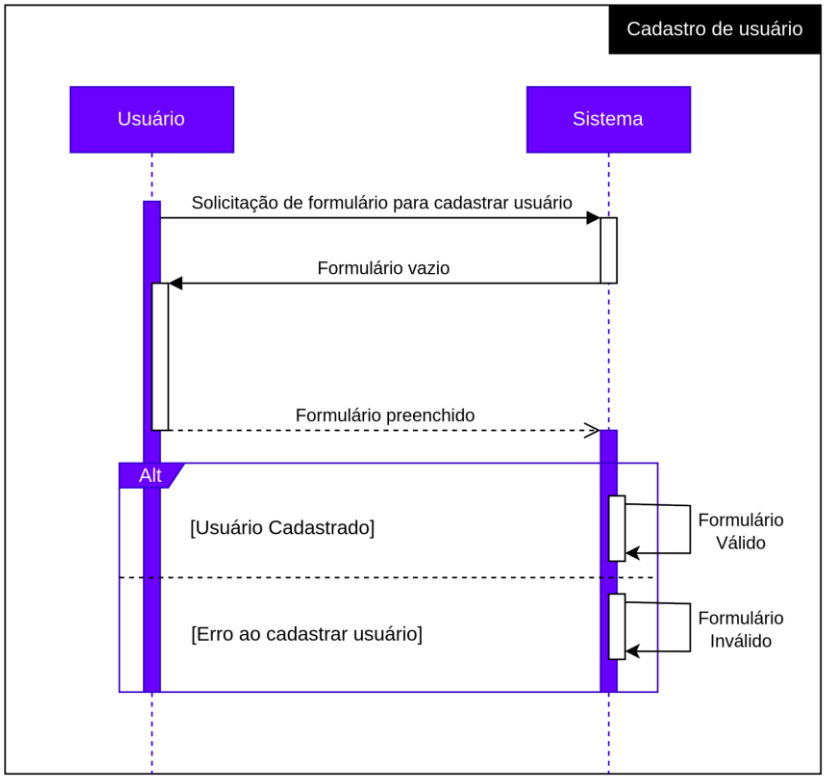
A avaliação da arquitetura de um um sistema busca identificar se o que foi projetado atende aos objetivos do projeto (BASS et al., 2012; SILVA et al., 2022). Dentre os métodos que podem ser utilizados encontra-se o Método de Análise Arquitetural baseado em Tradeoffs (Architecture Tradeoff Analysis Method) - ATAM (BASS et al., 2012). Esse método apresenta uma sequência de etapas e requer uma equipe para avaliar o projeto. Um dos elementos desse método é o uso de cenários. Neste sentido, pela limitação de pessoas envolvidas neste projeto, para avaliar a arquitetura proposta foi utilizada uma abordagem baseada em cenários. Além disso, para avaliar a capacidade da arquitetura ser desenvolvida, foi implementado o back-end do sistema.

Os cenários para a avaliação arquitetural foram elencados por meio de diagramas de interações, de três principais funcionalidades. O diagrama de interações é utilizado para representar visualmente as interações entre os componentes do sistema em tempo de execução, oferecendo uma visualização de como ocorre a troca de mensagens e métodos entre o usuário e os componentes do sistema (GUEDES, 2018). O diagrama representa, de forma temporal, as interações para se executar uma funcionalidade do sistema, tornando-a uma ferramenta útil para verificação e validação de arquiteturas/sistemas.

CENÁRIOS

O primeiro cenário modelado é o de cadastro de usuário (Figura 5). Para o cadastro, o usuário acessa o formulário de cadastro; o sistema o formulário e o usuário o preenche e submete o formulá. O sistema valida o formulário e retorna uma mensagem, revelando se foi criada com sucesso uma conta ou uma mensagem de erro (e.g., este email já foi cadastrado).

Figura 5 - Diagrama de interações - Cadastrar usuário



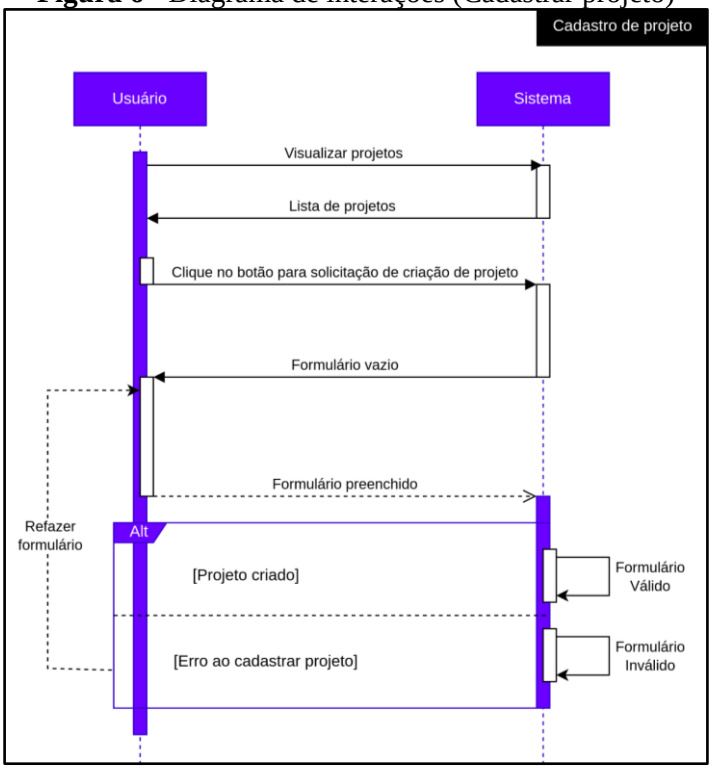
Fonte: Autoria própria (2024)

Outro cenário modelado foi o de cadastro de projetos (Figura 6), em que o usuário acessa a lista de projetos, clica em adicionar um novo projeto. O sistema direciona o usuário para um formulário, em que ele informará os dados do projeto. O formulário é preenchido e submetido para o sistema, que analisa e valida a requisição, retornando uma mensagem de projeto criado com sucesso ou um erro (e.g, título precisa conter no mínimo 1 caractere).

O último cenário descrito é o de emissão de relatório (Figura 7). Após listar o projeto, o usuário seleciona o desejado. Ao acessar as ações para o projeto, o usuário clica

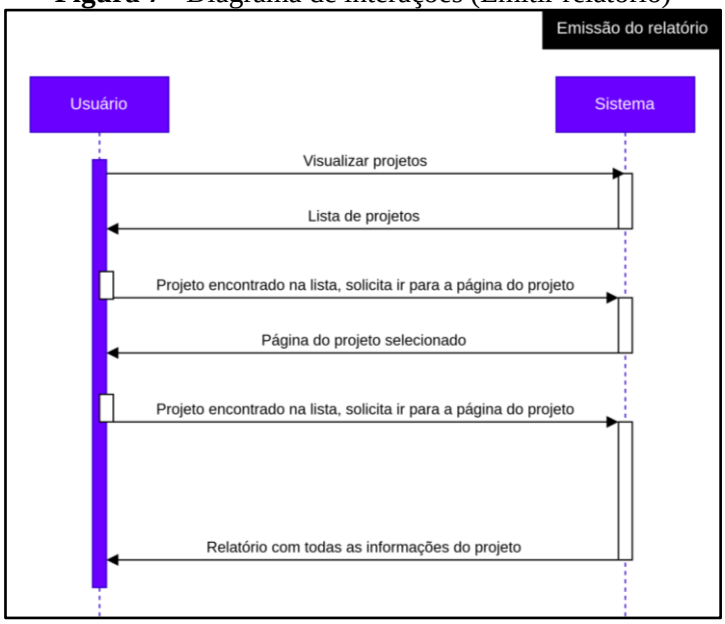
em emitir relatório e o sistema faz a coleta de todos os dados desse projeto (requisitos, diagramas, resumos, links externos, etc.) e retorna em forma de relatório para o usuário.

Figura 6 - Diagrama de interações (Cadastrar projeto)



Fonte: Autoria própria (2024)

Figura 7 - Diagrama de interações (Emitir relatório)



Fonte: Autoria própria (2024)

IMPLEMENTAÇÃO DE ARQUITETURA

A segunda parte da avaliação consiste em construir o *back-end* do ambiente, que exige esforço e tempo, necessitando de auxílio para o seu desenvolvimento. Nesse aspecto, metodologias de desenvolvimento se tornam necessárias, dentre elas as metodologias ágeis, tais como Kanban, Scrum, Lean, etc., que podem ser utilizados isoladamente ou em conjunto para chegar ao resultado almejado.

Para implementação do back-end do sistema foi escolhida a metodologia Scrum, uma abordagem ágil e iterativa (COHN et al., 2011). O Scrum oferece uma resposta rápida aos acontecimentos inesperados surgidos ao longo do desenvolvimento.

Nesta etapa da avaliação, foi possível construir o back-end o ExpressRD e o Scrum permitiu se ter um ambiente apropriado para os colaboradores, deixando claras as responsabilidades individuais e também promovendo a colaboração eficaz entre os membros da equipe, resultando em um ambiente mais ágil e adaptável às mudanças.

CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentada uma arquitetura para o sistema "Express Requirement Documents" ou ExpressRD. O sistema objetiva auxiliar na gestão dos requisitos. Em especial, o sistema visa auxiliar no controle e rastreio dos artefatos de software que são gerados a partir dos requisitos elencados. Na definição da arquitetura foram elaboradas três visões principais do software (Visão de Módulo, Componente & Conector e de Alocação). Para avaliar a arquitetura proposta foi utilizada uma metodologia baseada em cenários e na implementação do back-end do sistema proposto. O uso de cenários indicou que a arquitetura atende aos requisitos propostos e a implementação do back-end do sistema demonstrou a viabilidade de implementação da arquitetura. Em paralelo a este trabalho estão sendo desenvolvidos os demais componentes do ExpressRD. Como trabalho futuro, pretende-se realizar uma avaliação do ambiente proposto com potenciais usuários.

REFERÊNCIAS

ATLASSIAN. Atlassian | Software Development and Collaboration Tools. Disponível em: <https://www.atlassian.com/>. Acesso em: 25 de fevereiro de 2024.

COHN, M.; SILVA, A. J. C. C. DA; PRIKLADNICKI, R. Desenvolvimento de Software com Scrum: Aplicando Métodos Ágeis com Sucesso. 1a edição ed. [s.l.] Bookman, 2011.

FIGMA. Figma: the collaborative interface design tool. Disponível em: <https://www.figma.com/>. Acesso em: 25 de fevereiro de 2024.

GUEDES, G. T. A.UML 2 - Uma Abordagem Prática. Novatec Editora, 2018.

MELO, T. R. S., MORAIS, E. B. D., SOUSA, R. R. ExpressRD: Uma Ferramenta para Controle e Rastreio de Artefatos de Software. Revista Científica Multidisciplinar. 2024. To Appear.

MIRO. A Plataforma de Colaboração Visual para Todas as Equipes | Miro. Disponível em: <https://miro.com/pt/>. Acesso em 04 de março de 2024.

NOTION. Your connected workspace for wiki, docs & projects. Disponível em: <https://www.notion.so/pt-br/product>. Acesso em: 04 de março de 2024.

SANTOS, L.; SOUSA, R. R. DE; SILVA, J. W. C. Arquitetura de um ambiente para a Associação de Apoio aos Portadores de Câncer de Mossoró e Região (AAPCMR). Conjecturas, v. 22, n. 15, p. 958–976, 17 nov. 2022.

SOMMERVILLE, Ian. Engenharia de software.São Paulo: Pearson Prentice Hall, 2011.

TRELLO. Atlassian Trello. Disponível em: <https://trello.com/>. Acesso em: 25 de fevereiro de 2024.