**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação Lato Sensu em Arquitetura de Software Distribuído**

Projeto Integrado

Relatório Técnico

Revio

Sofia Lunkes da Silva

Belo Horizonte

Março de 2021.

**Sumário**

[Introdução 3](#_Toc101121995)

[1. Cronograma do Trabalho 5](#_Toc101121996)

[2. Especificação Arquitetural da solução 7](#_Toc101121997)

[3.1 Restrições Arquiteturais 7](#_Toc101121998)

[3.2 Requisitos Funcionais 7](#_Toc101121999)

[3.3 Requisitos Não-funcionais 8](#_Toc101122000)

[3.4 Mecanismos Arquiteturais 8](#_Toc101122001)

[3. Modelagem Arquitetural 9](#_Toc101122002)

[4.1 Diagrama de Contexto 9](#_Toc101122003)

[4.2 Diagrama de Container 10](#_Toc101122004)

[4.3 Diagrama de Componentes 11](#_Toc101122005)

[4. Prova de Conceito (PoC) 12](#_Toc101122006)

[5.1 Integrações entre Componentes 13](#_Toc101122007)

[5.2 Código da Aplicação 13](#_Toc101122008)

[5. Avaliação da Arquitetura (ATAM) 15](#_Toc101122009)

[6.1. Análise das abordagens arquiteturais 15](#_Toc101122010)

[6.2. Cenários 15](#_Toc101122011)

[6.3. Evidências da Avaliação 16](#_Toc101122012)

[6.4. Resultados Obtidos 17](#_Toc101122013)

[6. Avaliação Crítica dos Resultados 18](#_Toc101122014)

[7. Conclusão 19](#_Toc101122015)

[Referências 20](#_Toc101122016)

## Introdução

Com a pandemia do Covid-19, empresas que utilizem de tecnologia para permanecerem competitivas cresceram a um passo acelerado comparado a um período prévio a Covid-19. De acordo com estudo realizado pela Accenture, estima-se que as companhias usando software e tecnologia como meio, tiveram crescimento de receita 5 vezes superior a outras que não utilizaram esses meios. Dessa forma, empresas de tecnologia ou que utilizem de departamentos de tecnologia para habilitar seus produtos e/ou serviços, estão sujeitas a entregas de códigos que possibilitem seus negócios a fim de entregar valor a seus clientes. Dentro do contexto operacional de entrega de software, existem diferentes práticas adotadas pela comunidade de desenvolvedores que permitem tanto o processo de construção quanto manutenção. Para esse contexto, uma das práticas adotadas é a revisão por pares.

A revisão por pares acontece normalmente na divulgação de artigos para publicação e passam por crivos de pessoas do ramo para que, os dados apontados e demonstrados sejam tangíveis e realísticos, provando teorias ou práticas. Assim como no meio científico, a revisão de código por pares também acontece no meio de desenvolvimento de software. Essa prática já é conhecida pela área de garantia de qualidade (quality assurance) no desenvolvimento de software e baseia-se em checagem manual de mudanças no código fonte. A primeira abordagem sobre o tema, moldada por Michael E. Fagan, trouxe popularidade a prática utilizando um método formal para encontrar defeitos, estruturado por papeis e fases, incluindo também uma reunião de inspeção (FAGAN, 1976).

A revisão de código moderno (MCR) funciona de forma mais leve e baseada em ferramentas de mudança de código. Como visão geral, ambas formas têm o objetivo de diminuir complexidades desnecessárias e a manter um código legível e limpo, com regras claras e sucintas de comportamento, bem como manter a regra de negócio escrita aderente a realidade. As principais diferenças para o modelo de Fagan são: (1) ser informal, (2) baseado em ferramenta, (3) assíncrono e (4) focado em revisão na mudança.

O MCR costuma trazer diversos benefícios, tais como: diminuição de repetição de código, menor probabilidade de inserção de comportamento anormal, disseminação de conhecimento, aumento de manutenibilidade e outros.

Contudo o processo de revisão de código dentro de uma empresa na comunidade de desenvolvedores ainda é suscetível a falhas de implementação e pode ser levado ao esquecimento ou também a inimizade ao método. Dado este contexto, o objetivo do sistema de gerenciamento de revisores - Revio, é trazer para essas empresas uma solução de revisão de código simples, porém capaz de integrar diferentes ferramentas atuais de qualidade de código, fazer o gerenciamento de desenvolvedores que podem se tornar revisores sem que a curva de maturidade do time ou a curva de entregas de funcionalidades do/de software(s) impacte negativamente o dia a dia da companhia.

O uso desse sistema ajudará a escolher revisores e atentá-los a disponibilidade de códigos a serem revisados seguindo um guia de comportamento genérico, porém adaptativo as boas práticas que a comunidade local acredita ser aderente a estilo de código.

Revio tem como objetivo **principal** monitorar as solicitações de mescla de código e acionar os revisores necessários para realizar a revisão, em adição a integração com sistemas de gestão de mudança trazendo uma nova visão sobre o risco de implementação do código realizado. Com isto em foco, o trabalho apresentado possui os seguintes objetivos específicos:

* Analisar o mercado de sistemas de controle de versão adotados pelas empresas
* Apresentar uma interface para relacionar o sistema de controle de versão.
* Apresentar uma API para lidar com as requisições feitas pela interface
* Comunicar os personas alvos do processo em tempo real.
* Retornar ao sistema de gestão de mudança o status da solicitação de mescla de código com nova funcionalidade.

## Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** | **Fase** |
| **De** | **Até** |
| 01 / 02 / 22 | 15 / 02 /22 | 1. Estudo dos sistemas de controle de versão adotados com maior frequência pelo mercado empresarial. | Descobrir qual o sistema mais utilizado em contrapartida a facilidade de integração com sistemas externos. | Primeira Entrega |
| 14 / 02 / 22 | 26 / 02 / 22 | 1. Construção do documento do projeto integrado com devidas análises. | Viabilidade do projeto. | Primeira Entrega |
| 12 / 04 / 22 | 25 / 04 / 22 | 1. Levantamento de restrições arquiteturais. | Restrições Arquiteturais | Primeira Entrega |
| 26 / 04 / 22 | 01 / 05 / 22 | 1. Levantamento de requisitos funcionais e não funcionais | Principais funcionalidades a serem desenvolvidas respeitando os limites derivados do levantamento de requisitos. | Primeira Entrega |
| 01 / 05 / 22 | 03 / 05 / 22 | 1. Construção de diagrama de contexto |  | Primeira Entrega |
| 04 / 05 / 22 | 07 / 05 / 22 | 1. Construção de diagrama de containers |  | Segunda Entrega |
| 08 / 05 / 22 | 15 / 05 / 22 | 1. Construção do diagrama de componentes |  | Segunda Entrega |
| 16 / 05 / 22 | 20 / 05 / 22 | 1. Prototipação com wireframe |  | Segunda Entrega |
| 21 / 05 / 22 | 30 / 05 / 22 | 1. Construção autenticação |  | Segunda Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Construção API CRUD e Integração AD Microsoft |  | Segunda Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Integração API com ferramenta de gestão de mudança Service Now |  | Segunda Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Integração API com Slack |  | Segunda Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Integração API com Teams |  | Segunda Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Documentação dos cenários de testes a serem realizados |  | Segunda Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Deploy dos serviços na AWS |  | Segunda Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Cenários |  | Segunda Entrega |
|  |  | 1. Evidências da avaliação |  | Terceira Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Resultados obtidos |  | Terceira Entrega |
|  |  | 1. Avaliação crítica dos resultados |  | Terceira Entrega |
| \_\_ / \_\_ / \_\_ | \_\_ / \_\_ / \_\_ | 1. Conclusão do projeto |  | Terceira Entrega |

## Especificação Arquitetural da Solução

Nesta seção, tem-se como objetivo trazer as especificações visualizadas para o sistema de gestão de revisores e notificação – Revio. Trata-se de uma visão geral sobre as necessidades contempladas e com decisões que buscam manter o sistema próximo a adesões de linguagens e arquiteturas da comunidade desenvolvedora como um todo.

## Restrições Arquiteturais

Quadro 1 – Descritivo das restrições arquiteturais.

|  |  |
| --- | --- |
| Código | Descrição |
| RA01 | O software back-end deve ser desenvolvido em Kotlin, com o framework Spring |
| RA02 | As APIs devem seguir o padrão RESTful. |
| RA03 | Autenticação deve utilizar Json Web Token |
| RA04 | O software front-end deve ser desenvolvido com Javascript, framework/biblioteca react.js |
| RA05 | Deve-se utilizar da estratégia de conteinerização com Docker. |
| RA06 | Orquestração de containeres Docker, utilizando AWS ECS |
| RA07 | A arquitetura deve ser construída na AWS |
| RA08 | Todas as peças de arquitetura devem utilizar IaaC (infrastructure as a code) com terraform. |
| RA09 | GitHub é o sistema de controle de versão a ser utilizado |
| RA10 | A pipeline de construção e publicação com GitHub Actions e AWS Codepipelines |

## Requisitos Funcionais

Quadro 2 – Requisitos funcionais do sistema.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| RF01 | O sistema deve permitir o auto cadastramento do usuário | B | A |
| RF02 | O sistema deve permitir ao administrador remover revisor da lista de revisores | M | B |
| RF03 | O sistema deve permitir ao administrador cadastrar revisores | M | B |
| RF04 | O sistema deve permitir ao administrador criar uma lista de revisores para grupos específicos ou equipe | B | B |
| RF05 | O sistema deve permitir ao administrador criar uma lista de revisores aberta | B | A |
| RF06 | O sistema deve permitir ao administrador remover um revisor da lista ou grupo | B | M |
| RF07 | O sistema deve permitir ao administrador apagar um grupo de revisores | B | M |
| RF08 | O sistema deve permitir o cadastramento em 4 áreas de atuação: SRE, backend, frontend e mobile | B | A |
| RF09 | O sistema deve ter o comportamento de separar os grupos de revisores nas áreas de atuação | M | A |
| RF10 | O sistema deve permitir exibir regras de revisão configuradas | B | B |
| RF11 | O sistema deve permitir ao usuário alterar seu nome de exibição | B | B |
| RF12 | O sistema deve permitir ao usuário alterar sua senha de ingresso | A | M |
| RF13 | O sistema deve permitir ao usuário alterar sua imagem de perfil | B | B |
| RF14 | O sistema deve permitir ao usuário ficar inativo por até 30 dias | M | M |
| RF15 | O sistema deve notificar os revisores quando tiver revisões disponíveis | M | M |
| RF16 | O sistema deve notificar o revisor somente após o *build* da solicitação de código estiver finalizado com sucesso | B | A |
| RF17 | O sistema deve mostrar todas as solicitações de código abertas e disponíveis para revisão | B | A |
| RF18 | O sistema deve remover da lista o item de revisão quando o critério de revisores for satisfeito | B | A |
| RF19 | O sistema não deve bloquear a solicitação de código ser mesclada | B | B |
| RF20 | O sistema não deve apresentar uma solicitação de código de uma área de atuação para outra área | B | A |
| RF21 | O sistema não deve notificar um revisor de uma solicitação de código de uma área diferente do revisor. | B | A |
| RF22 | O sistema deve permitir ao administrador configurar a periodicidade das notificações de solicitações de código pendentes | A | B |
| RF23 | O sistema deve permitir ao administrador definir critério de quantidade de revisores para uma solicitação | M | M |
| RF24 | O sistema deve permitir ao administrador associar repositórios a grupos de revisores | M | A |
| RF25 | O sistema deve permitir ao administrador cadastrar o revisor em 1 ou mais grupos | A | A |
| RF26 | O sistema não deve permitir ao administrador cadastrar o revisor em mais de 1 área de atuação | B | B |
| RF27 | O sistema deve permitir ao administrador cadastrar outro administrador | B | M |

\*B=Baixa, M=Média, A=Alta.

## Requisitos Não-funcionais

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | O sistema deve ter acesso ao software de controle de versão GitLab | A |
| RNF02 | O sistema web deve manter mínimo de 90% de acessibilidade pela ferramenta Lighthouse | A |
| RNF03 | O sistema deve ser acessível em qualquer região do país | B |
| RNF04 | O sistema não deve permitir que dados de uma empresa sejam acessados por outra empresa | B |
| RNF05 | A nuvem a ser utilizada deve ser a Amazon Web Services | B |
| RNF06 | O sistema deve conseguir integrar com a ferramenta de comunicação Slack | M |
| RNF07 | O sistema não deve bloquear a equipe a realizar codificações | B |
| RNF08 | O sistema deve utilizar uma base de dados não relacional | B |
| RNF09 | O sistema deve ter logs de ações para todos as APIs expostas | B |
| RNF10 | Deve ser possível rastrear a requisição feita pelo usuário entre os serviços | B |

## Mecanismos Arquiteturais

Os mecanismos arquiteturais apresentam uma visão geral dos componentes de uma arquitetura de software e são baseados em 3 estados: (1) análise, (2) design e (3) implementação. Mecanismos de análise são os estados iniciais da arquitetura, mecanismos de design representam decisões sobre tecnologias a serem usadas e mecanismos de implementação são as implementações reais baseada nas especificações definidas nos mecanismos anteriores.

No quadro abaixo, serão apresentados os mecanismos nas três categorias supracimas citadas contextualizadas no sistema Revio.

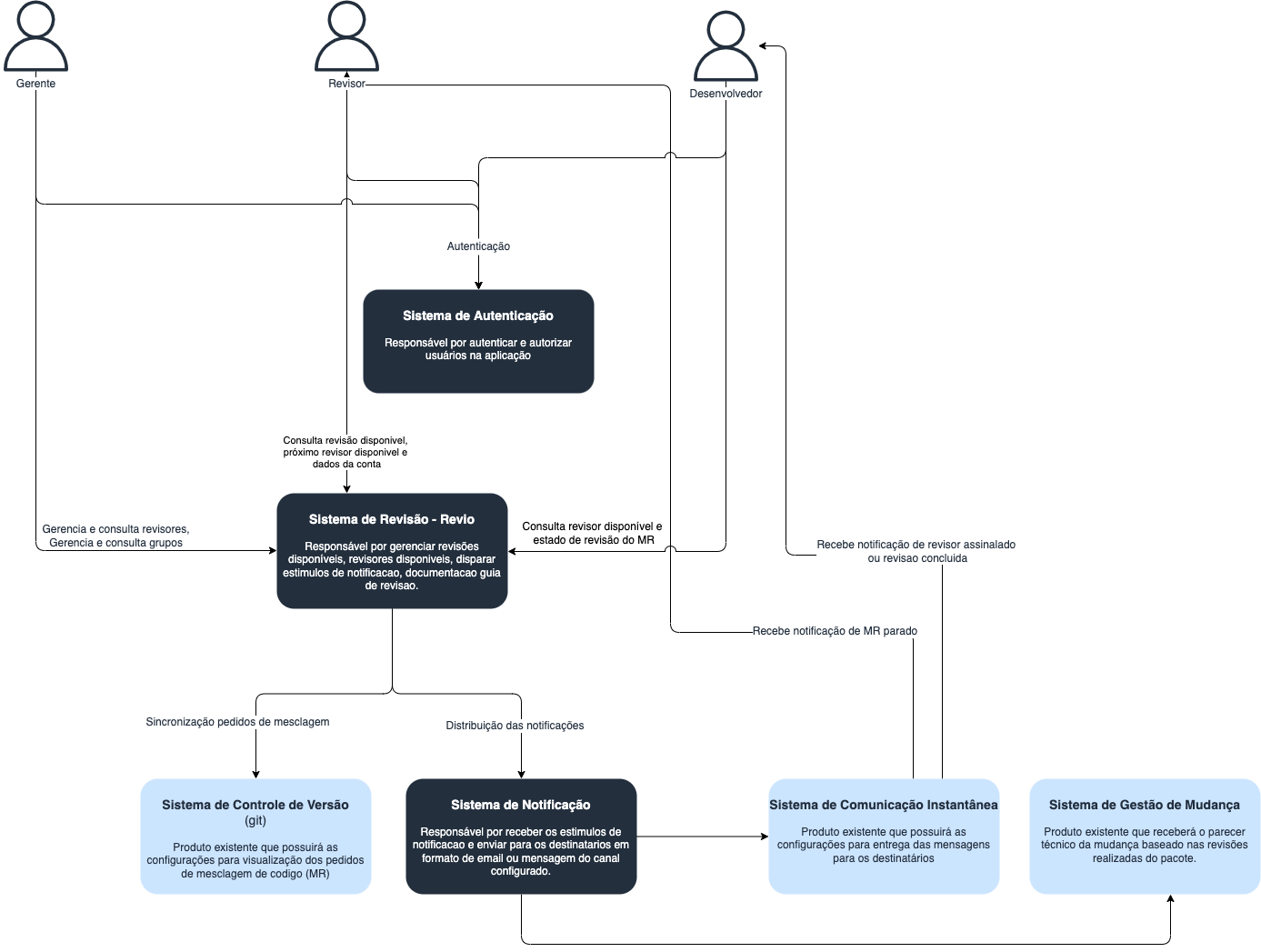
|  |  |  |
| --- | --- | --- |
| **Análise** | **Design** | **Implementação** |
| Persistência | Controle de Versão | Github |
| Persistência | ORM | Spring Framework |
| Persistência | Banco de Dados Não Relacional | MongoDB |
| Front end | Single Page Application | React.js |
| Back end | Microsserviços | Kotlin |
| Integração | Gateways | AWS API Gateway |
| Log do sistema | Gestão de logs | Cloudwatch |
| Teste de Software | Teste Unitário | JUnit |
| Teste de Software | Teste de Componente | Spring Cloud Contract |
| Distribuição | Implantação e Entrega Continua | GitHub Actions  AWS code pipeline |
| Comunicação | Mensageria | SQS/SNS |
| Comunicação | HTTP | RESTful |
| Segurança | Autenticação e Autorização | JWT |

## Modelagem Arquitetural

A modelagem arquitetural para o sistema de revisão Revio será apresentada em 3 diagramas diferentes, sendo a primeira pela visão de diagrama de contexto, diagrama de container e diagrama de componentes. O diagrama tem como objetivo fornecer um caminho e uma visão ampla de quais peças serão necessárias para o sistema funcionar como um todo, possibilitando complementar o entendimento da arquitetura.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: <https://c4model.com/> e aqui: <https://www.infoq.com/br/articles/C4-architecture-model/>. Dos quatro nível que compõem o modelo C4 três serão apresentados aqui e somente o Código será apresentado na próxima seção (5).

## 4.1 Diagrama de Contexto



**Figura 1 - Visão Geral da Solução Revio.**

A figura 1 mostra a especificação o diagrama geral da solução proposta, com todos seus principais módulos e suas interfaces.

## Referências

Exemplo:

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.

<https://exame.com/pme/numero-de-empresas-de-tecnologia-no-brasil-cresce/> - acessado 22 fevereiro de 2022.

<https://www.accenture.com/us-en/insights/technology/scaling-enterprise-digital-transformation?c=acn_glb_futuresystemsmediarelations_12144611&n=mrl_0421> – acessado 21 de fevereiro de 2022.

<https://www.codegrip.tech/productivity/how-microsoft-does-its-code-review/> - acessado 15 de março

<https://devblogs.microsoft.com/appcenter/how-the-visual-studio-mobile-center-team-does-code-review/> - acessado 16 de março

<https://dotlib.com/blog/o-que-e-revisao-por-pares> - acessado 21 de março de 2022.

<https://www.cin.ufpe.br/~rls2/processo_tg/Metodologia%20S&B/guidances/guidelines/architectural_mechanisms_374675A0.html> - acessado 17 de maio de 2022.

<https://c4model.com/> - acessado 18 de maio de 2022