UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BEATRIZ ALICE ALVES SANTOS AZEVEDO

CLEAN CODE E SOLID NA CONSTRUÇÃO DA APLICAÇÃO ORATIO: MELHORANDO A MANUTENIBILIDADE E ESCALABILIDADE

CAMPINA GRANDE - PB

2023

BEATRIZ ALICE ALVES SANTOS AZEVEDO

CLEAN CODE E SOLID NA CONSTRUÇÃO DA APLICAÇÃO ORATIO: MELHORANDO A MANUTENIBILIDADE E ESCALABILIDADE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharela em Ciência da Computação.

Orientador: Professor Dr. Tiago Lima Massoni

CAMPINA GRANDE - PB

2023

BEATRIZ ALICE ALVES SANTOS AZEVEDO

CLEAN CODE E SOLID NA CONSTRUÇÃO DA APLICAÇÃO ORATIO: MELHORANDO A MANUTENIBILIDADE E ESCALABILIDADE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharela em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Tiago Lima Massoni

Orientador – UASC/CEEI/UFCG

Professor Dr. Jorge Cesar Abrantes de Figueiredo

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 14 de Fevereiro de 2023.

CAMPINA GRANDE - PB

ABSTRACT

Oratio is a web application created to facilitate the work of professors responsible for the Work and Conclusion of Course (TCC) discipline of Computer Science at the Federal University of Campina Grande. It allows the professor to easily allocate evaluators and have control of all information related to a project and a student. Developed in Flutter and Java, Oratio was designed with the Clean Code and SOLID philosophy, thus ensuring the quality, efficiency, and scalability of the code. Although the Oratio project followed good programming practices, not all clean code concepts were used, but rather a modification of them to improve project development and maintenance. This was done to ensure that SOLID features are met, aiming to increase code maintenance and flexibility and make the project more robust and easily scalable in a short development time.

Clean Code e SOLID na construção da aplicação Oratio: Melhorando a manutenibilidade e escalabilidade

Beatriz Alice Alves Santos Azevedo
beatriz.azevedo@ccc.ufcg.edu.br
Unidade Acadêmica de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba

Resumo

Oratio é uma aplicação web criada para facilitar o trabalho dos professores responsáveis pela disciplina de Trabalho e Conclusão de Curso (TCC) de Ciência da Computação da Universidade Federal de Campina Grande . Ela permite que o professor aloque facilmente avaliadores e tenha o controle de todas as informações relacionadas a um projeto e a um aluno. Desenvolvida em Flutter e Java, Oratio foi projetada com a filosofia de Clean Code e SOLID, garantindo assim a qualidade, a eficiência e a escalabilidade do código. Embora o projeto Oratio tenha seguido boas práticas de programação, não foram utilizados todos os conceitos do clean code, mas sim uma modificação dos mesmos para melhorar o desenvolvimento e a manutenção do projeto. Isso foi feito para garantir que as características do SOLID sejam atendidas, visando aumentar a manutenção e a flexibilidade do código e tornar o projeto mais robusto e de fácil escalabilidade em um curto tempo de desenvolvimento.

Palavras-chave:

Aplicação Web, Clean Code, SOLID, Flutter, Boas práticas .

Repositórios do projeto:

- front-end (https://github.com/ibiaalice/oratio-front)
- back-end (https://github.com/ibiaalice/oratio-api)

1. Introdução

A avaliação de projetos de TCC é uma etapa importante no processo de conclusão de curso, pois permite que os alunos apresentem seus trabalhos para avaliação e obtenham feedback dos professores orientadores. No entanto, a avaliação pode ser complicada e desorganizada, especialmente devido ao uso excessivo de formulários e à acumulação de informações em diferentes lugares. É aí que entra o Oratio, uma aplicação desenvolvida para facilitar a avaliação de projetos de TCC. Com o Oratio, é possível centralizar todas as informações importantes em um único lugar, tornando o processo mais fácil e organizado para todas as partes envolvidas. Além disso, o Oratio permite que as informações sejam acessadas de forma mais rápida e eficiente, evitando o acúmulo de informações e a necessidade de procurar em diferentes lugares.

O Oratio é uma aplicação web desenvolvida com as tecnologias Flutter e Java que visa auxiliar professores de disciplinas de Trabalho e Conclusão de Curso (TCC) na alocação de avaliadores e no gerenciamento de informações relacionadas aos projetos e alunos. Além disso, a aplicação foi projetada seguindo os princípios do Clean Code e SOLID. Clean Code é um conjunto de boas práticas de programação que buscam produzir código de fácil leitura, manutenção e evolução. O objetivo é tornar o desenvolvimento mais eficiente e seguro, diminuindo erros e facilitando a manutenção do software, bibliotecas, imagens entre outros elementos do projeto. A utilização do Clean Code é objeto de diversas opiniões entre os profissionais de tecnologia. Enquanto alguns acreditam que é fundamental para garantir a qualidade do código, outros argumentam que pode ser limitante para a criatividade e flexibilidade do desenvolvedor. De acordo com Robert C. Martin, autor do livro "Clean Code: A Handbook of Agile Software Craftsmanship", o Clean Code é uma prática fundamental para garantir que o código seja fácil de evoluir [1]. Já outros especialistas argumentam que as regras e padrões impostos pelo Clean Code podem ser rígidos e prejudicar a capacidade de solução de problemas de forma criativa. David Heinemeier Hansson, criador do Ruby on Rails, afirmou: "O Clean Code é uma bela visão, mas é uma armadilha para seguir à risca. Em vez de se concentrar nas melhores práticas, é melhor concentrar-se na melhor comunicação"[2]. Já o SOLID é uma sigla que representa os cinco princípios de design de software que visam ajudar na construção de aplicações mais coesas, coerentes e escaláveis. Os princípios do SOLID são considerados uma boa prática para seguir na construção de aplicações de software, ajudando a garantir a qualidade e a manutenção do código. Estes princípios ajudam a garantir que o software seja escalável, mantido e fácil de ser testado.

A combinação desses conceitos com o desenvolvimento do Oratio garante que a aplicação seja de fácil manutenção, escalável e de alta qualidade. A implementação de Clean Code e SOLID também ajuda a preservar a integridade dos dados e a segurança das informações fornecidas pelos usuários.

O artigo apresenta o projeto Oratio, a aplicação desenvolvida com o objetivo de facilitar a avaliação de projetos de TCC (Trabalho de Conclusão de Curso). A plataforma foi criada para solucionar os problemas encontrados nos processos de avaliação atuais,

como o uso excessivo de formulários, acúmulo de informações e informações separadas em diferentes lugares.

2. Solução

2.1 Visão geral

A avaliação do TCC começa com o acompanhamento dos alunos durante o semestre, com o professor orientador fornecendo direcionamentos e feedback para aprimorar o projeto. Quando o texto está pronto, ele é submetido para a banca examinadora, que avalia e avalia a qualidade do projeto e se ele atende aos critérios estabelecidos. Finalmente, após a avaliação da banca, o resultado da avaliação é compartilhado com o aluno. O Oratio permite ao professor ter um controle completo sobre as informações dos projetos, incluindo título, descrição, o link com o arquivo de pré-projeto e detalhes dos orientadores e avaliadores, como mostrado abaixo (figura 1).

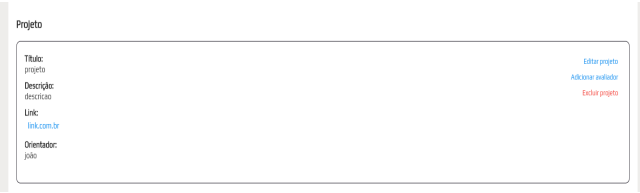

Figura 1. Detalhamento de projeto

O processo de avaliar trabalhos de conclusão de curso (TCCs) pode ser desafiador para os professores da disciplina. Isso porque existe a necessidade de revisar formulários longos, coletar informações em vários locais e monitorar o progresso dos projetos. Essa sobrecarga de tarefas pode levar a atrasos na avaliação e dificultar a garantia da qualidade dos trabalhos avaliados. Com as informações organizadas e de fácil acesso, o professor tem mais tempo para se concentrar nas atividades pedagógicas, ao invés de se perder na gestão de informações, como demonstrado na imagem (Figura 2).

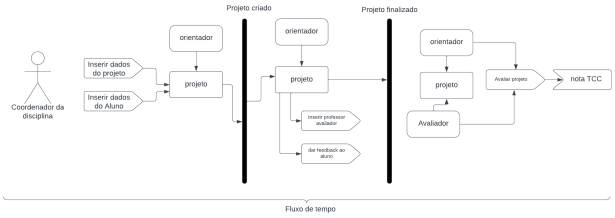

Figura 2. linha do tempo do projeto

No Oratio, existem diferentes atores que desempenham papéis importantes na avaliação de projetos de TCC. Estes incluem o professor responsável pela disciplina, que é o principal avaliador dos trabalhos e tem a tarefa de revisar e avaliar os projetos, além de fornecer orientações e feedbacks aos estudantes. Também há os estudantes que estão desenvolvendo os projetos de TCC, por fim, pode haver outros membros da comunidade acadêmica que podem ajudar a fornecer suporte e orientação aos estudantes, como outros professores. Todos esses atores trabalham juntos para garantir que o processo de avaliação dos projetos de TCC seja tão eficiente e eficaz quanto possível.

Além do registro do progresso do aluno, este recurso permite que o professor identifique possíveis dificuldades ou problemas o mais cedo possível, o que possibilita a tomada de decisões e ajustes importantes antes que seja tarde demais. A capacidade de fornecer feedback e orientações aos alunos é fundamental para garantir que todos os projetos sejam bem-sucedidos e alcancem suas metas.

2.2 Descrição

O Oratio é importante por oferecer uma solução eficiente e organizada para os profissionais que precisam gerenciar trabalhos de conclusão de curso. É uma ferramenta que aglomera informações importantes sobre os projetos, seus respectivos orientadores e avaliadores em um só lugar, o que permite ao administrador ter uma visão mais clara e completa do andamento. Além de possibilitar a adição de feedbacks aos alunos durante o processo de desenvolvimento, o que ajuda na melhor gerência do projeto e na construção de uma base sólida para a avaliação do professor responsável pelo trabalho. O uso da aplicação também promove a organização de informações, tornando mais fácil para os administradores localizar informações relevantes e tomar decisões precisas e informadas.

2.3 Funcionalidades

A lista de funcionalidades do Oratio inclui:

1. Registro e Login: permite que usuários criem suas contas na plataforma, fornecendo informações pessoais e do e-mail do usuário. ilustrados nas imagens abaixo tela de registro, (Figura 3) tela de login.

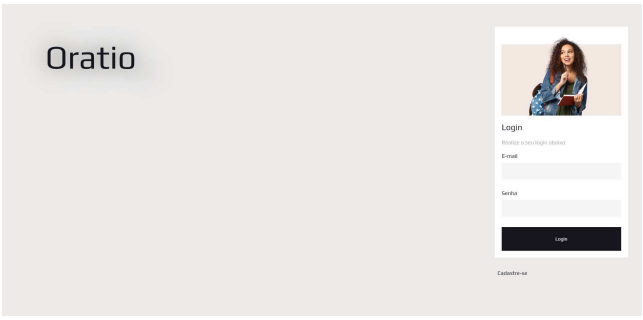

Figura 3. Tela de login

2. Abertura e Fechamento de Semestre: permite ao usuário administrador controlar o acesso aos projetos de alunos, fechando o semestre quando desejado. Como

demonstrado na imagem abaixo, (figura 4) abertura de semestre.

Figura 4. Tela de abertura de semestre

3. Cadastro Individual de Alunos e Professores: permite que usuários adicionem informações sobre alunos e professores, incluindo informações pessoais e de contato.
4. Cadastro de Alunos e Professores por Tabela do Google Sheet: permite que usuários importem informações sobre alunos e professores a partir de uma tabela do Google Sheet. (Figura 5).

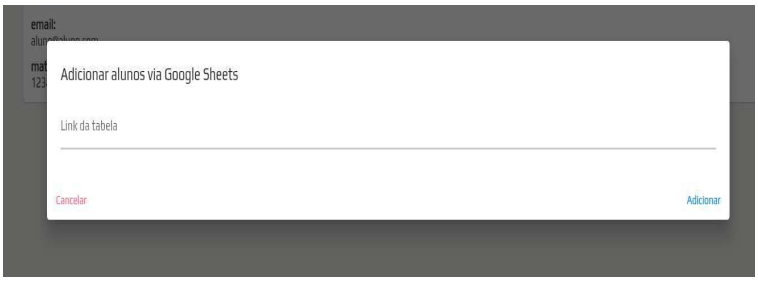

Figura 5. Opção de adicionar alunos por tabela Google Sheets

5. Remoção de Alunos e Professores: permite que usuários excluam informações sobre alunos e professores.
6. Adição, Edição e Exclusão de Acompanhamento de Alunos: permite que usuários acompanhem o desenvolvimento dos projetos dos alunos, adicionando, editando e excluindo informações. (figura 6) abaixo.

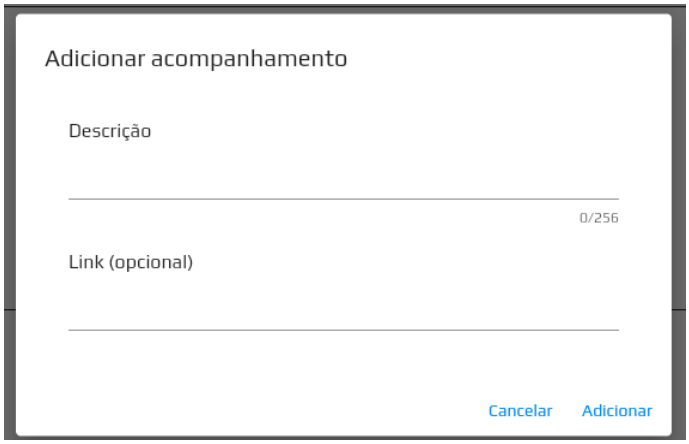

Figura 6. Opção de adicionar acompanhamento ao aluno.

2.4 Arquitetura

A arquitetura do Oratio foi desenvolvida com o objetivo de proporcionar uma boa performance, escalabilidade e manutenção do software. Para isso, foi utilizada uma combinação de tecnologias de front-end (Flutter) e back-end (Java), criando uma estrutura em camadas para garantir a separação de responsabilidades e uma melhor organização do código. Além disso, foram aplicados conceitos de clean code e SOLID na construção da aplicação, para garantir que o código seja fácil de entender, modificar e manter. Esta arquitetura permite a integração com outras ferramentas, como o Google Sheets, para facilitar a gestão de informações.

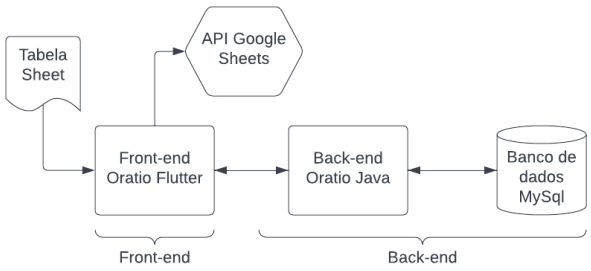

Figura 7. Arquitetura da aplicação

2.4.1 Back-end

A arquitetura backend do Oratio é baseada na linguagem de programação Java, utilizando as tecnologias Spring Security, Lombok, Spring Boot, JPA Repository e Javax. O objetivo é fornecer uma camada de segurança sólida para a aplicação, além de simplificar o processo de desenvolvimento, tornando-o mais eficiente.

O Spring Security é utilizado para garantir a autenticação e autorização de usuários, protegendo as informações sensíveis [4]. O Lombok é uma biblioteca que ajuda a reduzir a verbosidade do código, tornando-o mais conciso [5].

Já o Spring Boot é um framework que oferece uma série de recursos para facilitar o desenvolvimento de aplicações, incluindo a configuração automática do servidor e a geração de artefatos prontos para uso [6]. O JPARepository é uma interface que fornece acesso ao banco de dados através de operações básicas como salvar, atualizar, excluir e buscar informações [7].

Por fim, o Javax é uma biblioteca de classes e interfaces padrões do Java, utilizada para fornecer suporte a diferentes tecnologias e funcionalidades [8]. O banco de dados relacional escolhido é o Mysql, que é uma das opções mais utilizadas e confiáveis na área de tecnologia [9].

A arquitetura backend do Oratio foi projetada para oferecer segurança, facilitando o armazenamento de dados confidenciais. Além disso, a arquitetura foi projetada para ser simples e fácil de entender, tornando mais fácil para os desenvolvedores trabalharem com a aplicação. A eficiência foi uma preocupação importante na criação da arquitetura, o que garante que a aplicação funcione de forma rápida e eficiente. Por fim, a arquitetura foi desenvolvida para ser escalável, permitindo que a aplicação cresça e evolua de acordo com as necessidades dos usuários. Para garantir a realização dessas metas, tecnologias comprovadas e com suporte ativo foram utilizadas na criação da arquitetura backend do Oratio.

2.4.2 Front-end

A arquitetura front-end do Oratio é baseada no framework Flutter, versão 3.3. O Flutter é um framework de desenvolvimento de aplicativos móveis desenvolvido pelo Google, que permite criar aplicativos nativos para Android, iOS, desktop e web. Além disso, o Flutter utiliza a linguagem de programação Dart, tornando a escrita de código mais fácil e eficiente.

Ao desenvolver a arquitetura front end do Oratio, foi escolhido utilizar o Flutter Web, que é uma tecnologia que permite que aplicativos Flutter sejam executados em navegadores web. Isso possibilita a criação de uma aplicação de alta qualidade e com boa performance, sem a necessidade de plugins extras, como Adobe Flash ou Java.

Além do Flutter Web, o Oratio também utiliza dois packages importantes, o MobX e o Gsheets. O MobX é uma biblioteca de estado gerenciada para o framework, que ajuda a criar páginas mais responsivas e com menos código. Já o Gsheets é uma biblioteca que facilita a integração com o Google Sheets, possibilitando a importação de dados de forma rápida e segura.

3. Metodologia

A metodologia de desenvolvimento do Oratio foi baseada em uma abordagem ágil, onde os requisitos do projeto foram levantados com o professor responsável da cadeira de trabalho de conclusão de curso. O objetivo era garantir que as necessidades e expectativas do professor estivessem claras e pudessem ser atendidas durante o processo de desenvolvimento.

Para garantir o sucesso do projeto, foram estabelecidas entregas de 15 em 15 dias, acompanhadas por uma reunião semanal com o professor. Essas reuniões eram importantes para discutir o andamento do projeto, identificar questões e tomar decisões sobre eventuais correções ou mudanças. O processo foi baseado em rituais Scrum adaptados, o que permitiu a atualização e entrega de partes do projeto durante as reuniões. Foram necessárias 6 reuniões durante o espaço de tempo de 4 meses, uma reunião em cada Sprint, totalizando o projeto ao final do tempo decorrido.

Durante o desenvolvimento, foi necessário fazer algumas modificações nos requisitos iniciais levantados. Uma das modificações realizadas foi a remoção da possibilidade do aluno inserir seus projetos com o login no Oratio, bem como as opções do professor avaliador. Além disso, limitamos a abertura do semestre a apenas um semestre por vez, sem a separação por salas de aulas. Estas mudanças foram feitas para simplificar o funcionamento da plataforma e garantir uma experiência mais direcionada ao objetivo do projeto. Isso foi necessário para que o escopo do projeto conseguisse ser entregue dentro do prazo estabelecido. Alguns requisitos prévios foram cortados, mas isso foi feito com o objetivo de garantir a qualidade e a entrega do projeto dentro do prazo.

A abordagem ágil permitiu que o projeto evoluísse de forma dinâmica e que as necessidades do professor fossem atendidas de forma mais eficiente. Além disso, a realização de entregas frequentes e a comunicação constante com o professor permitiram que o projeto fosse ajustado de acordo com as necessidades da cadeira de trabalho de conclusão de curso e que o mesmo conseguisse ser entregue dentro do prazo estabelecido.

4. Resultados

O resultado do projeto Oratio é uma ferramenta de gestão de projetos de TCC que permite aos professores e coordenadores gerenciarem as atividades dos alunos de maneira eficiente. Além de permitir o registro e login, a abertura e fechamento de semestres, o cadastro individual de alunos e professores, o Oratio também permite a inserção de dados dos alunos e professores pelas tabelas do Google Sheet.

Durante o desenvolvimento do Oratio, foi aplicada uma metodologia de trabalho ágil, sendo adequada ao tempo do orientador e do desenvolvedor, com levantamento de requisitos e entregas periódicas. Mesmo com algumas mudanças e cortes em requisitos ao longo do desenvolvimento, o projeto conseguiu ser entregue dentro do prazo estabelecido.

O Oratio foi desenvolvido com um objetivo claro de seguir princípios de Clean Code e utilizar conceitos do SOLID, o que ajuda a garantir a qualidade e a manutenção do código ao longo do tempo. Essas boas práticas de programação foram aplicadas rigorosamente durante todo o processo de desenvolvimento do Oratio.

Começando com a nomenclatura de variáveis e métodos, elas foram escritas de forma clara e descritiva para garantir que qualquer pessoa que leia o código possa facilmente entender o propósito de cada uma delas. Além disso, a nomenclatura foi

mantida coerente ao longo do projeto para garantir consistência e facilidade de leitura.

A indentação também foi uma preocupação importante durante o desenvolvimento do Oratio. Ela foi aplicada de forma consistente para destacar as estruturas de controle e facilitar a leitura do código. Isso permite que o desenvolvedor tenha uma visão clara do fluxo do código, o que é fundamental para a correção de bugs e para a implementação de novos recursos.

Refatoração é outra prática importante que foi aplicada no Oratio. O código foi revisado regularmente para garantir sua legibilidade e mantê-lo simples e enxuto. Isso ajuda a garantir que o código seja fácil de manter e melhorar ao longo do tempo, o que é fundamental para o sucesso de um projeto de software a longo prazo.

No projeto Oratio, os princípios SOLID foram aplicados para garantir a qualidade e a flexibilidade do software. Por exemplo, o princípio da Responsabilidade Única foi implementado para garantir que cada classe tenha uma única responsabilidade, o que torna o código mais claro e fácil de manter. O princípio da Aberto-Fechado foi utilizado para que as classes sejam projetadas de tal forma que possam ser estendidas sem precisar modificá-las, tornando o código mais flexível e adaptável. Além disso, o princípio da Substituição de Liskov foi aplicado para garantir que as subclasses pudessem ser utilizadas em lugares onde a classe base é esperada, sem causar problemas. Por exemplo, a classe de controle Store foi utilizada como classe base para todas as classes de controle de estado, garantindo que os métodos de escrita não precisassem ser reescritos em cada classe. Isso resultou em um código mais organizado e fácil de manter, pois a lógica de escrita foi centralizada na classe Store, e as subclasses puderam se concentrar na funcionalidade específica de cada controle de estado. Como exemplo de responsabilidade única, na arquitetura do Oratio, foi criada uma classe específica para gerenciar o Login, onde todas as responsabilidades relacionadas a autenticação dos usuários foram centralizadas. Dessa forma, a classe Login possui apenas uma responsabilidade única, garantindo a clareza e a manutenção do código. Esses são alguns exemplos da aplicação do SOLID no Oratio, que resultou em um código de alta qualidade e fácil de manter.

A abordagem destes dois conceitos permitiu a criação de um código de fácil manutenção e escalabilidade, garantindo a continuidade e evolução do projeto no futuro.

Em resumo, o projeto Oratio trouxe uma solução eficiente e fácil de usar para a gestão de projetos de TCC, facilitando a vida dos professores e coordenadores. Além disso, o projeto também permitiu aprimorar conhecimentos e habilidades em tecnologias e metodologias de desenvolvimento de software.

5. Experiência e lições aprendidas

Nesta seção será descrito o percurso proporcionado pelo processo de desenvolvimento do projeto Oratio, bem como os desafios e tarefas futuras.

5.1 Processo de desenvolvimento

O projeto Oratio foi desenvolvido através de uma série de etapas e ações planejadas. Primeiramente, foi realizada uma análise dos requisitos do projeto, levados pelo professor da disciplina, incluindo as necessidades e expectativas do usuário. Em seguida, definiu-se a arquitetura e as ferramentas a serem utilizadas, incluindo o framework Flutter e o banco de dados Mysql. Levando em conta o conhecimento prévio do desenvolvedor. Durante o processo de desenvolvimento, foram realizadas entregas a cada 15 dias, acompanhadas por reuniões semanais com o professor responsável pela cadeira de trabalho de conclusão de curso.

O desenvolvimento do Oratio envolveu a utilização de boas práticas de codificação, incluindo a implementação de Clean Code e SOLID, para garantir a qualidade e a manutenção do código. Além disso, o uso do Flutter Web permitiu que o aplicativo fosse adaptado para atender ao objetivo do projeto, que era de permitir a inserção de dados de alunos e professores através de tabelas do Google Sheets.

Ao longo do processo, foram enfrentados desafios relacionados à adaptação do Flutter Web para atender aos requisitos do projeto, especialmente no que diz respeito à integração com outras ferramentas e tecnologias.

Em resumo, o processo de desenvolvimento do Oratio foi uma jornada desafiadora, mas também extremamente gratificante e enriquecedora de desenvolvimento. Ao final do projeto, foi possível constatar que os objetivos foram alcançados e que o Oratio é uma ferramenta útil e de grande valor para os usuários.

5.2 Principais desafios enfrentados

Neste projeto de desenvolvimento, foram enfrentados desafios importantes devido à implementação de boas práticas de programação, como Clean Code e SOLID, bem como a utilização de tecnologias de backend, as quais não eram familiarizadas pelo desenvolvedor responsável.

Um dos desafios encontrados durante o desenvolvimento foi a utilização do Flutter Web. Embora essa tecnologia tenha permitido a criação de aplicativos para a web de forma mais rápida e eficiente, a equipe teve que se adaptar a algumas limitações e dificuldades técnicas durante o desenvolvimento. No entanto, o resultado final foi uma ferramenta de gestão de projetos de TCC fácil de utilizar.

Apesar destes obstáculos, houve uma dedicação intensa na pesquisa e aprendizado dessas metodologias, o que permitiu a evolução constante no desenvolvimento do projeto. Além disso, o desenvolvedor teve que lidar com o desafio de atuar como full-stack, ou seja, desenvolver tanto o frontend quanto o backend sem ter contato anterior com as tecnologias envolvidas da parte do backend.

Com estes desafios, é possível destacar a importância da adaptação constante e da vontade de aprender no desenvolvimento de projetos tecnológicos, além da relevância de seguir boas práticas de programação para garantir a qualidade e a escalabilidade do software.

5.3 Limitações

Durante o processo de desenvolvimento do projeto Oratio, uma das principais limitações encontradas foi a falta de conhecimento em infraestrutura por parte do único desenvolvedor envolvido no projeto. Embora esse desenvolvedor tenha atuado como fullstack, não possuía contato anterior com tecnologias de backend. Isso impossibilitou a publicação e disponibilização do projeto na web, uma vez que não havia tempo hábil para realizar pesquisas e aprendizado. Infelizmente, devido ao curto período de tempo destinado ao desenvolvimento da aplicação Oratio, não foi possível realizar uma avaliação completa durante a disciplina. O foco principal foi aprender novas tecnologias do backend, e por isso, a equipe decidiu priorizar o desenvolvimento do MVP (Produto Mínimo Viável) e entregá-lo para avaliação posterior.

5.4 Trabalhos futuros

O projeto Oratio teve como objetivo inicial criar uma aplicação básica que pudesse ser usada como exemplo. No entanto, há muitas possibilidades de melhorias futuras. Por exemplo, é possível adicionar novos tipos de usuários, como professores orientadores e alunos, a fim de obter feedback e aprimorar a plataforma. Além disso, pode ser desenvolvido telas responsivas para diferentes dispositivos e expandido para incluir a criação de aplicativos para diferentes plataformas, como Android e iOS, usando as capacidades da multiplataforma do Flutter. Finalmente, com as melhorias mencionadas, o projeto Oratio pode ser publicado e disponibilizado para o público em geral. Há ainda muitos trabalhos futuros a serem realizados, como integração com outros sistemas, criação de relatórios e implementação de novas funcionalidades pedidas pelos usuários, a fim de aprimorar ainda mais a plataforma e oferecer uma excelente experiência aos usuários.

Agradecimentos

Gostaria de expressar minha gratidão a todos aqueles que participaram ativamente no desenvolvimento do projeto Oratio, oferecendo comentários e direcionamentos valiosos. Agradeço aos meus amigos de faculdade, que sempre estiveram dispostos a ajudar ao longo do projeto, e à minha família, que apoiou incondicionalmente. Agradeço também ao professor Tiago por suas orientações e interesse no trabalho. E por fim, gostaria de destacar meu agradecimento especial a meu pai, que ao longo de toda a sua vida, me apoiou e me motivou a seguir em frente em busca de meus sonhos.

REFERÊNCIAS

[1] CLEAN Code: Prequel and Principles. In: CLEAN Code: A Handbook of Agile Software Craftsmanship: Writing clean code is what you must do in order to call yourself a professional. There is no reasonable excuse for doing anything less than your best.. 2. ed. San Francisco: Prentice Hall, 2008. cap. 1, p. 15.

[2] HANSSON, David Heinemeier. Code: Optimize for Happiness. In: GETTING Real: The smarter, faster, easier way to build a successful web application. [S. l.: s. n.], 2006. p. 107. E-book.

[3] MCCONNELL, Steve. Design in Construction: Design Practices. In: MCCONNELL, Steve. Code Complete: A Practical Handbook of Software Construction. 2. ed. A Division of Microsoft Corporation: Microsoft Press, 2004. cap. 5, p. 110. ISBN 0-7356-1967-0. E-book.

[4] SPRING Security. [S. l.], 2023. Disponível em: https://spring.io/projects/spring-security. Acesso em: 26 jan. 2023.

[5] LOMBOK. [S. l.], 2023. Disponível em: https://projectlombok.org/setup/maven. Acesso em: 26 jan. 2023.

[6] SPRING boot. In: Spring boot. [S. l.], 2023. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 26 jan. 2023.

[7] JPA Repository. In: Jpa Repository . [S. l.], 2023. Disponível em: https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html. Acesso em: 3 fev. 2023.

[8] JAVAX. In: Javax. [S. l.], 2023. Disponível em: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html. Acesso em: 3 fev. 2023.

[9] MYSQL. In: MySql. [S. l.], 2023. Disponível em: https://docs.oracle.com/en-us/iaas/mysql-database/doc/overview-mysql-database-service.html. Acesso em: 3 fev. 2023.

Sobre o autor:

Beatriz Alice Alves Santos Azevedo é uma estudante de Ciência da Computação na Universidade Federal de Campina Grande, no momento cursando o 10º período. Atualmente, trabalha como Desenvolvedor Mobile na empresa Softdesign.