UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS -
BACHARELADO EM SISTEMAS
DE INFORMAÇÃO

OTÁVIO EDÉSIO IZIDORO LINO DA SILVA
VINÍCIUS BERNARDES CREPALDI

Sistema de gestão do aluno

BAURU
Dezembro/2023

OTÁVIO EDÉSIO IZIDORO LINO DA SILVA
VINÍCIUS BERNARDES CREPALDI

Sistema de gestão do aluno

Monografia apresentada para a disciplina Trabalho de Conclusão de Curso II do Curso de Bacharelado em Sistemas da Informação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru, sob orientação do Prof. José Remo Ferreira Brega.

BAURU
Dezembro/2023

Silva, Otávio Edesio Izidoro Lino da
S586s Sistema de gestão do aluno / Otávio Edesio Izidoro Lino da Silva, Vinicíus Bernardes Crepaldi. -- Bauru, 2023 - 36 p.
Trabalho de conclusão de curso (Bacharelado - Sistemas de Informação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientador: José Remo Ferreira Brega
1. chatGPT. 2. Quiz. 3. Frequência.
I. Crepaldi, Vinicius Bernardes II. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a)

RESUMO

A aplicação tem como objetivo proporcionar uma ferramenta eficiente e prática para a gestão da rotina acadêmica dos alunos, tornando-a mais organizada e produtiva, permitindo que o aluno gerencie suas faltas, e utilize a funcionalidade de “quiz” para estudar através das recomendações, referente aos assuntos registrados, ou temas diversos na qual o usuário pode digitar e escolher qual a melhor opção para o seu estudo. O sistema possui uma interface de fácil utilização, podendo ser acessado por dispositivos móveis, sendo seu desenvolvimento dividido em diferentes módulos para otimizar a aplicação. Os objetivos a serem alcançados com essa solução foram a melhora do desempenho escolar dos alunos, evitando problemas acadêmicos relacionados a faltas e aumentando a interação entre os usuários do aplicativo e os conteúdos referentes às matérias registradas de uma forma dinâmica e prática.

PALAVRAS-CHAVES

ChatGPT, Sistema de Gestão, Controle de faltas, Mobile, Android, Kotlin, Java, Quiz dinâmico, Desafios estudantis, Quiz usando ChatGPT, Requisição e resposta API ChatGPT, Banco de dados MongoDB, Impacto econômico na educação.

ABSTRACT

The application aims to provide an efficient and practical tool for managing students' academic routines, making them more organized and productive. It allows students to manage their absences and use the quiz functionality to study based on recommendations related to recorded subjects or various topics, where users can type and choose the best option for their study. The system features a user-friendly interface accessible through mobile devices, and its development is divided into different modules to optimize the application. The achieved objectives with this solution include improving students' academic performance, preventing academic problems related to absences, and enhancing interaction between application users and dynamically practicing the content related to registered subjects.

Keywords

ChatGPT, Management System, Attendance Control, Mobile, Android, Kotlin, Java, Dynamic Quiz, Student Challenges, Quiz using ChatGPT, ChatGPT API Request and Response, MongoDB Database, Economic Impact on Education.

LISTA DE ILUSTRAÇÕES

Figuras

Quadros

SUMÁRIO

1. Introdução

A gestão eficaz do tempo e a organização acadêmica são desafios constantes enfrentados pelos estudantes, refletindo não apenas em seu desempenho individual, mas também nas dinâmicas educacionais e sociais. Estudos recentes destacam a preocupante frequência com que os estudantes se deparam com problemas relacionados à falta de organização, resultando em consequências adversas para seu progresso acadêmico.

A desorganização dos estudantes, muitas vezes manifestada na falta de controle sobre suas presenças em sala de aula, tem sido objeto de investigação e análise por parte de pesquisadores. Conforme apontado por [Inoue et al., 2020], a falta de uma estrutura organizacional eficiente pode levar a um aumento significativo nas taxas de faltas, afetando diretamente o aprendizado e comprometendo o desenvolvimento acadêmico.

Outros estudos, como o realizado por [Silva e Santos, 2018], corroboram essa perspectiva, evidenciando uma correlação entre a falta de organização dos estudantes e a queda no desempenho escolar. A ausência frequente em aulas pode resultar em lacunas no entendimento do conteúdo, comprometendo a assimilação de conhecimento e prejudicando o alcance de metas educacionais.

Nesse contexto, emerge a necessidade premente de soluções inovadoras que auxiliem os estudantes na gestão eficiente de sua rotina acadêmica. Este trabalho propõe uma contribuição significativa para a resolução desse problema, apresentando uma aplicação prática e eficaz para o gerenciamento de faltas, cujo desenvolvimento foi fundamentado nas lacunas identificadas por pesquisas anteriores.

Ao longo deste trabalho, serão explorados os impactos da desorganização estudantil, assim como a eficácia da solução proposta na promoção da organização e no aprimoramento do desempenho acadêmico. Através dessa abordagem, almejamos não apenas preencher uma lacuna na literatura acadêmica, mas também

oferecer uma contribuição tangível para a melhoria da experiência educacional dos estudantes.

1.1 Estruturação do documento

O documento começa com um resumo do trabalho e uma introdução, na qual é apresentado o problema que será abordado no trabalho e os objetivos da proposta. Em seguida, é feita uma detalhada descrição do problema, destacando sua importância e a necessidade de soluções para o mesmo. São apresentadas soluções existentes, mas o foco principal é na proposta do projeto, que é descrito em detalhes. As tecnologias pesquisadas para a implementação do projeto também são descritas. Depois, são apresentados os procedimentos de funcionamento do projeto e uma análise dos escopos de códigos, e finalmente, uma conclusão que resume o trabalho realizado e uma lista de referências.

2. Detalhamento do Problema

2.1 Problema que busca-se resolver

O problema que se almeja resolver neste trabalho é a falta de organização e controle das faltas dos alunos, que pode levar a consequências acadêmicas, como reprovação em disciplinas. É comum que os alunos tenham dificuldade em acompanhar sua frequência nas aulas, e muitas vezes percam o controle das faltas acumuladas ao longo do semestre. Além disso, a falta de organização pode levar a outros problemas, como falta de conhecimento dos critérios de avaliação, médias das provas e trabalhos e dados de contato do professor. A motivação foi proporcionar aos alunos uma ferramenta que auxiliasse no gerenciamento eficiente de suas faltas, além de proporcionar uma funcionalidade que ajudasse na fixação do conteúdo referente às matérias, permitindo que ficassem atualizados sobre sua frequência e evitassem problemas acadêmicos.

Soluções existentes para esse problema incluíam sistemas de gerenciamento de faltas em aplicativos, permitindo que os alunos registrassem suas faltas e acompanhassem sua frequência.

Uma alternativa desenvolvida foi a criação de um aplicativo específico para o gerenciamento de faltas dos alunos. A principal diferença reside no fato de que os alunos podem acompanhar, sem depender de atualizações dos professores ou do acesso a plataformas de ensino online, já que eles próprios realizam o cadastro. Além disso, o aplicativo inclui outras funcionalidades úteis, como o registro de informações referentes a cada matéria, espaço para anotações e uma ferramenta de quiz com perguntas de múltipla escolha sobre temas definidos pelo próprio aluno.

As tecnologias e ferramentas utilizadas incluem linguagens de programação como Java, Kotlin e Xml para o desenvolvimento de aplicativos móveis, plataformas de desenvolvimento como Android Studio, Intellij, Xcode, Visual Studio e banco de dados MongoDB para o armazenamento dos dados dos alunos.

2.2 Consequências

A questão afeta diretamente os alunos, que podem sofrer consequências acadêmicas, como reprovação em disciplinas, caso ultrapassem o limite de faltas permitido pela instituição de ensino. Além disso, a falta de organização pode prejudicar o desempenho acadêmico dos alunos, interferindo na qualidade do aprendizado. Portanto, a solução do problema contribui para o desenvolvimento acadêmico dos alunos, aumentando a qualidade do ensino e diminuindo a evasão escolar.

Nota-se o impacto econômico da falta de organização e controle das faltas dos alunos, que pode ser observado tanto no nível individual, quanto no nível institucional. Para o aluno, o impacto pode ser representado pelo aumento do tempo necessário para a conclusão do ensino, além de gastos com aulas extras ou com a necessidade de pagar novamente disciplinas reprovadas. Para as instituições de ensino, a falta de controle das faltas pode levar à queda na qualidade do ensino, o que pode impactar a reputação da instituição, e consequentemente, a captação de novos alunos.

A ausência de um controle eficiente da frequência e a falta de recursos para pesquisar temas de interesse e questões discutidas nos dias de ausência impõem desafios substanciais aos estudantes universitários. A desconexão resultante com o conteúdo acadêmico pode prejudicar a compreensão global dos temas abordados, dificultando o acompanhamento das aulas subsequentes e comprometendo o desempenho acadêmico. A incapacidade de recuperar informações perdidas também limita o engajamento dos alunos, afetando negativamente sua participação ativa e troca de ideias. Essa lacuna no aprendizado, somada à preparação inadequada para exames, pode resultar em desmotivação e desinteresse, impactando significativamente a experiência educacional e o comprometimento dos estudantes com seus estudos. Portanto, é imperativo abordar esses problemas de forma abrangente para criar um ambiente acadêmico mais favorável e promover o sucesso dos alunos.

3. Objetivos da Proposta

O objetivo desta proposta de TCC foi desenvolver um sistema inovador e intuitivo que permita aos alunos gerenciarem suas matérias de forma eficiente, de acordo com o dia da semana. O sistema conta com um calendário personalizado para cada aluno, no qual poderá registrar suas presenças para cada disciplina. A principal vantagem é que a responsabilidade pelo controle da frequência será atribuída ao aluno, não dependendo apenas da instituição de ensino, proporcionando maior autonomia no acompanhamento de suas obrigações acadêmicas.

Além disso, o sistema também oferece uma funcionalidade única: a possibilidade de adicionar as matérias que os alunos perderam em campos específicos dos dias em que faltaram, criando uma forma estruturada de compensar o conteúdo perdido. Dessa maneira, os alunos terão uma ferramenta para se organizarem e reforçarem os tópicos perdidos, garantindo um aprendizado mais completo e consistente.

Por fim, a inclusão de uma aba dedicada ao estudo com quizzes foi proposta, os quais serão fornecidos pelo sistema ChatGPT. Essa funcionalidade permite que os alunos testem seus conhecimentos de forma interativa e estimulante. O uso de quizzes gerados pelo ChatGPT proporcionará uma variedade de perguntas e abordagens, tornando o processo de aprendizado mais interessante e atraente. Assim, os estudantes poderão revisar os conteúdos de forma mais efetiva, consolidando o aprendizado adquirido em sala de aula e reforçando seu conhecimento em diferentes disciplinas. Com esse sistema abrangente, espera-se que os alunos alcancem um melhor desempenho acadêmico e desenvolvam habilidades de autogestão essenciais para o sucesso em sua jornada educacional.

4. Soluções Existentes

4.1 Notion

Notion é uma ferramenta de produtividade tudo-em-um que permite aos usuários criar notas, tarefas, calendários, bancos de dados entre outras coisas, tudo em uma mesma aplicação. Com o Notion, os usuários podem organizar suas informações pessoais e de trabalho em um espaço digital altamente personalizável e colaborativo.

Seu funcionamento é baseado em blocos, o que significa que os usuários podem criar, arrastar e soltar vários tipos de blocos, como notas, tarefas, imagens, tabelas. Eles podem ser organizados em páginas que podem ser compartilhados com outras pessoas para colaboração no projeto.

A aplicação também oferece muitos modelos e integração de aplicativos para ajudar os usuários a personalizar seus espaços de trabalho. Ele permite criar páginas, que podem conter vários tipos de blocos e podem ser organizadas em uma estrutura hierárquica. Por exemplo, é possível criar uma página principal para um projeto e várias subpáginas para cada fase do projeto.

Além disso, o Notion permite criar bancos de dados personalizados que podem ser usados para gerenciar informações, como lista de tarefas, orçamentos e catálogo de produtos. Os bancos de dados podem ser criados a partir de modelos ou do zero, e é possível definir propriedades para cada registro, como tipo de dado, cor, status entre outros atributos. Eles também podem ser visualizados em diferentes formatos, como tabela, lista, calendário e galerias.

Possui recursos de colaboração, permitindo que os usuários compartilhem suas páginas e bancos de dados com outras pessoas. É possível conceder diferentes níveis de acesso, como leitura, escrita ou administração, e colaborar em tempo real com outras pessoas em uma página ou banco de dados.

O Notion oferece ainda diversas integrações com outros aplicativos, como Slack, Google Drive, Trello, Asana, GitHub e Zapier. Isso permite que os usuários conectem suas informações em diferentes plataformas e automatize tarefas repetitivas.

Como aspectos negativos cita-se que o Notion possui um nível de complexidade relevante para usuários que não estejam acostumados a certos conceitos utilizados no aplicativo como por exemplo o de banco de dados. Além disso, embora haja um plano gratuito disponível, muitos dos recursos mais avançados do Notion são exclusivos para os planos pagos, que podem ter um custo elevado para usuários individuais ou pequenas empresas, também há o fato do plano gratuito do Notion ter um limite de armazenamento de 1000 blocos, o que pode ser rapidamente atingido se o usuário criar muitos bancos de dados e páginas com conteúdo rico em mídia, como imagens e vídeos.

4.2 Eduqo

O Eduqo é uma solução de gestão escolar que oferece diversos recursos para escolas, incluindo o controle de frequência dos alunos. O seu funcionamento pode variar um pouco dependendo das necessidades da escola e do plano escolhido, mas em geral, ele funciona da seguinte maneira:

Cadastro de alunos: Os administradores da escola cadastram os alunos no sistema, inserindo informações como nome, data de nascimento, série, turma, entre outras.

Controle de frequência: Com o cadastro de alunos concluído, os professores podem registrar as faltas dos alunos diretamente no sistema, por meio de um aplicativo ou plataforma online. É possível registrar faltas por aula, por período ou por dia, dependendo da preferência da escola.

Análise de frequência: Com os dados de frequência registrados no sistema, os administradores da escola podem gerar relatórios para análise. Os relatórios podem

mostrar a frequência de cada aluno ao longo do tempo, a frequência da turma como um todo, entre outras informações relevantes.

Comunicação com pais e responsáveis: O Eduqo também oferece recursos para comunicação com pais e responsáveis, como o envio de notificações sobre as faltas dos alunos. Com essa funcionalidade, os pais podem ser alertados imediatamente sobre a ausência do filho e tomar as medidas necessárias para garantir que ele não perca conteúdo importante.

Outros recursos: Além do controle de frequência, o Eduqo oferece outros recursos de gestão escolar, como o controle de notas, diários de classe eletrônicos, agendamento de atividades, entre outros. A plataforma é altamente customizável, permitindo que a escola escolha os recursos que melhor se adequam às suas necessidades.

4.3 Quadro comparativo

Quadro 1 - Comparação entre aplicativos

	Notion	**Eduqo**	**Nossa aplicação**
Controle de frequência		**x**	**x**
Aluno no controle das informações	**x**		**x**
Gestão de informações	**x**	**x**	**x**
Aba para reforço de estudo com quiz			**x**

O quadro comparativo acima destaca as características distintivas entre a aplicação desenvolvida, Eduqo e Notion, evidenciando uma vantagem significativa

proporcionada pela nossa solução. A principal diferenciação reside no foco específico em elementos-chave para o sucesso acadêmico.

Em primeiro lugar, a aplicação desenvolvida se destaca ao oferecer um controle de frequência integrado, proporcionando aos estudantes uma ferramenta vital para gerenciar sua presença nas aulas. Esta funcionalidade não apenas auxilia na prevenção de reprovações devido à falta excessiva, mas também promove uma participação mais consistente e eficaz.

Além disso, a autonomia do aluno na gestão de informações é uma característica distintiva. Permitir que os estudantes acessem detalhes específicos sobre disciplinas em um formato acessível por dia coloca o controle diretamente em suas mãos, facilitando a recuperação de informações perdidas e promovendo um aprendizado autodirigido.

Enquanto o Eduqo oferece funcionalidades abrangentes para organização de tarefas e comunicação, nossa aplicação proporciona uma abordagem mais direcionada para evitar reprovações devido à falta excessiva, oferecendo abas específicas para reforço de estudo com quizzes interativos. A autonomia do aluno na gestão de informações, com espaços dedicados para salvar dados relevantes sobre disciplinas, contribui para uma experiência mais focada e organizada, destacando a nossa aplicação como uma ferramenta mais especializada para abordar desafios acadêmicos específicos.

Notion é conhecido por sua flexibilidade e versatilidade como uma ferramenta de organização e produtividade, a nossa aplicação se destaca por sua especialização em atender às necessidades específicas dos estudantes. Uma das principais diferenças reside na ênfase na gestão acadêmica, com destaque para o controle de frequência integrado. Essa funcionalidade específica oferece aos estudantes uma abordagem prática para evitar reprovações devido à falta excessiva, algo que o Notion não oferece de maneira tão direta.

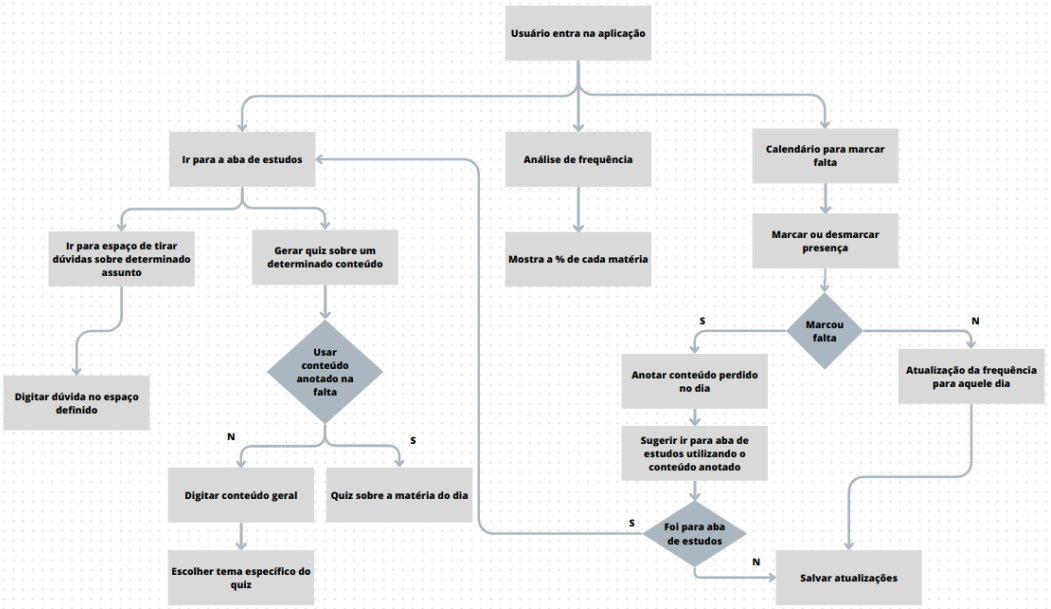
5. Descrição do Projeto Proposto

Este projeto consiste no desenvolvimento de um Sistema de Controle de Frequência e Informações Acadêmicas para facilitar o acompanhamento da presença dos alunos, gerenciar informações das disciplinas e fornecer uma experiência de aprendizado aprimorada. A aplicação inclui seções para marcar faltas, registrar tópicos abordados em aula, visualizar a frequência dos alunos e gerar quizzes dinâmicos usando a API do ChatGPT. A integração com o ChatGPT proporciona a geração automática de quizzes com base nos tópicos das aulas, oferecendo uma abordagem inovadora para envolver os alunos no processo de aprendizagem.

Neste trabalho, utilizou-se serviços que integram o conceito de Linguagem de Modelagem de Linguagem (LLM) para aprimorar a interação entre usuários e o sistema e o modelo integrado foi do ChatGPT. A escolha do ChatGPT se fundamentou na familiaridade com sua utilização, destacando-se pela interface de conexão API da openAi, a qual se revelou simplificada e direta. Essa facilidade de integração foi um fator crucial para a decisão final da LLM utilizada, proporcionando um ambiente de desenvolvimento mais acessível em comparação com outras opções disponíveis no universo de Inteligência Artificial (IA).

5.1 Diagrama de blocos

Figura 1 - Diagrama de blocos



Na figura 1, a tela inicial foi exibida para permitir ao usuário cadastrar suas disciplinas de segunda a sexta-feira, inserindo informações relevantes como nome do professor, matéria, e-mail, entre outros dados. Caso a opção do usuário seja direcionada ao calendário, nesta tela, é possível cadastrar faltas e presenças, além de fazer anotações sobre tópicos perdidos em dias específicos.

Ao optar pela tela de análise de frequência, são disponibilizadas informações sobre os dias de presença para cada matéria, proporcionando ao aluno um controle efetivo de sua frequência acadêmica. Por outro lado, ao escolher a tela de quiz, o aluno pode pesquisar perguntas para reforçar o conteúdo, levando em consideração os tópicos previamente perdidos ou registrados. Adicionalmente, há a facilidade de realizar pesquisas em outros tópicos simplesmente digitando as palavras-chave desejadas.

5.2 Funcionalidades e formas de interação

O sistema permite que os alunos possam gerenciar suas faltas de forma eficiente, de modo que possam acompanhar sua frequência, cadastrar informações por disciplina e evitar problemas acadêmicos. Os alunos poderão realizar o cadastro das disciplinas em que estão matriculados e terão a opção de visualizar a quantidade de faltas e a frequência já acumuladas em cada disciplina.

O aplicativo móvel desenvolvido utiliza Kotlin no front-end e Java no back-end, com o objetivo de fornecer um sistema abrangente para o gerenciamento de frequência e aprendizado personalizado. Com suas três telas principais - login, calendário de frequência e quiz de conteúdo - e a integração com a API do ChatGPT, o sistema oferecerá aos estudantes uma experiência única de aprendizado, permitindo o acompanhamento da frequência e a revisão interativa dos conteúdos perdidos.

Comparado ao Notion e Eduqo, o sistema proposto apresenta vantagens significativas, fornecendo uma solução focada nas necessidades específicas dos

usuários. Enquanto o Notion oferece uma ampla gama de recursos de organização e colaboração, o sistema proposto se concentra especificamente no gerenciamento de frequência e revisão de conteúdos perdidos. Além disso, a integração com a API do ChatGPT oferece uma abordagem personalizada de aprendizado, gerando respostas adaptadas às necessidades específicas dos estudantes.

Por outro lado, o Eduqo é um aplicativo educacional, mas não possui recursos específicos para o gerenciamento de frequência. O sistema proposto preenche essa lacuna, permitindo que os usuários registrem sua presença nas aulas e acompanhem sua frequência de maneira eficiente.

Descrição dos blocos para o projeto:

1. Interface:

Esse bloco é responsável pela interface do sistema e pela comunicação com o usuário. Ele apresentará as informações para o usuário e receberá as entradas realizadas por meio de dispositivos móveis. Ele utilizará Kotlin no front-end para projetar e implementar as telas, priorizando a usabilidade, design centrado no usuário e a consistência visual em todas as interações.

A tela de login deve fornecer campos para inserção de dados diversos sobre as matérias em determinado dia da semana, enquanto o calendário de frequência exibe um layout intuitivo para selecionar e registrar presenças, sendo possível registrar tópicos para estudar, interagindo com a tela de reforço do conteúdo. Já a tela do quiz de conteúdo deverá apresentar perguntas e respostas de forma clara e interativa, utilizando elementos visuais adequados.

O Material Design oferece uma linguagem visual consistente, com o uso de elementos como botões, ícones e tipografia específicos. Por meio de bibliotecas, como o Material Components for Android, é possível implementar facilmente esses elementos visuais em um aplicativo Kotlin desenvolvido no IntelliJ. Além disso, o IntelliJ oferece um conjunto robusto de recursos e ferramentas para o desenvolvimento Kotlin mobile. Ele fornece suporte completo ao Kotlin, incluindo sugestões de código, refatoração, depuração e testes. O IntelliJ também permite a

integração com sistemas de controle de versão, como o Git, facilitando o gerenciamento do código-fonte do aplicativo.

2. Banco de Dados:

Este bloco armazena as informações inseridas pelos usuários no sistema, como cadastro de disciplinas, dados de alunos, informações de chamadas, entre outros. É responsável pela integridade dos dados e pela sua disponibilidade, a integração com o MongoDB proporciona uma solução prática e eficiente para a interação com o banco de dados.

Utilizando o MongoDB, é possível armazenar, sincronizar e acessar os dados do aplicativo em tempo real, oferecendo uma experiência atualizada e consistente para os usuários.

Para estabelecer a interação com o banco de dados MongoDB, é necessário configurar o projeto para utilizar o driver oficial do MongoDB em conjunto com uma biblioteca de ORM (Object-Relational Mapping, usou-se o MongoOperations) para operações mais complexas.

Em seguida, é possível utilizar as classes e métodos disponibilizados pelo driver e pela biblioteca ORM para realizar operações de leitura, escrita e sincronização dos dados.

O MongoDB também oferece recursos avançados, como a possibilidade de criar índices para acelerar o acesso a determinados campos e a capacidade de distribuir e replicar dados de forma escalável. Além disso, é possível definir permissões de acesso aos dados com base em regras personalizadas, ajudando a proteger os dados e garantir que apenas os usuários autorizados possam ler e gravar informações específicas.

3. Módulo Aluno: Este bloco é responsável pela gestão das informações de cada aluno, como o registro das matérias dos alunos no sistema, dados voltados às matérias que o aluno está cursando, email, telefone dos professores, as médias e um campo livre para o aluno anotar informações sobre a matéria.

4. Módulo Disciplina: Este bloco é responsável pelo gerenciamento das informações de cada disciplina e a atualização das informações referentes à frequência.

5. Módulo Estudar: Este bloco é responsável por tratar os assuntos armazenados de tópicos ou temas diversos, que foram perdidos nos dias de falta do aluno, por meio de quizzes tirados do ChatGpt para reforçar o conhecimento do aluno.

Desenvolvimento dos módulos:

1. Interface: A interface foi desenvolvida utilizando linguagens de programação para desenvolvimento mobile, como Kotlin e Java.

2. Banco de Dados: O banco de dados foi desenvolvido utilizando um sistema de gerenciamento de banco de dados não relacional por meio do MongoDB.

3. Módulo Aluno: O módulo aluno foi desenvolvido utilizando linguagens de programação como Java e Kotlin e se comunica com o banco de dados para realizar as operações de cadastro, inserção e visualização das informações.

4. Módulo Disciplina: O módulo disciplina foi desenvolvido utilizando linguagens de programação como Java e Kotlin, com informações como disciplina e os dias que consta falta ou presença para ela, com informações com as anotações nos dias que o aluno faltou.

5. Módulo Estudar: O módulo estudar é desenvolvido em software, utilizando linguagens de programação como Java e Kotlin, trazendo questões para o aluno responder sobre assuntos anotados na tela de registro de frequência, ou outros temas.

6. Tecnologias Pesquisadas:

A tecnologia abrange conhecimentos, ferramentas, técnicas e recursos utilizados no desenvolvimento e implementação de soluções em diversas áreas. No contexto do desenvolvimento de aplicativos, englobam linguagens de programação, frameworks, bibliotecas e outras ferramentas. A tecnologia concorrente refere-se às alternativas disponíveis no mercado para alcançar os mesmos objetivos.

No aplicativo desenvolvido, o uso de Kotlin no front-end e Java no back-end é uma opção tecnológica, proporcionando vantagens específicas, como recursos avançados, interoperabilidade e suporte a programação assíncrona. Cada tecnologia tem suas próprias características e trade-offs, não existindo uma abordagem universalmente melhor.

A escolha da tecnologia adequada depende do contexto e requisitos específicos, e a complexidade e erros de teste podem variar conforme a qualidade do código, o processo de teste adotado e a familiaridade da equipe de desenvolvimento com as tecnologias utilizadas. É apresentado algumas possíveis tecnologias como opção a ser utilizadas em cada bloco:

1. Bloco de Interface com o Usuário:
- Tecnologia: Desenvolvimento de aplicativo mobile para Android, utilizando as linguagens de programação Kotlin e Java, respectivamente.
- Tecnologias concorrentes: Outras linguagens de programação como React Native e Xamarin, Swift .
2. Bloco de Armazenamento de Dados:
- Tecnologia: Banco de dados não relacional MongoDB que trabalha com um contexto de documentos permitindo maior dinâmica ao lidar com os dados.
- Tecnologias concorrentes: Outros bancos de dados não relacionais, como CosmosDB e Cassandra.
3. Bloco de Controle de Frequência, matérias e estudar:
- Tecnologia: Cálculo de frequências em java, com código e interfaces em Kotlin para visualização, para controlar a contagem de faltas e se comunicar com o aplicativo mobile.
- Tecnologias concorrentes: Outros microcontroladores, como Arduino e Raspberry Pi, ou módulos de contagem de faltas pré-fabricados.

7. Implementação da aplicação

1. Lógica de presença:

Figura 2 - Lógica de presença

```
public UserPresenceResponse updatePresence(final UserPresenceRequest request) {

        final UserPresence userPresenceMap = userPresenceRequestToUserPresenceMapper.map(request);

        final var itsNewClass = userFrequencyService.verifyNewCurrentClass(userPresenceMap);

        final UserPresence userPresence = userPresenceDAO.update(userPresenceMap);

        if(Objects.isNull(userPresence) ||  !userFrequencyService.updateFrequency(userPresence, itsNewClass)){
            return  null;
        }

        return userPresenceToUserPresenceResponseMapper.map(userPresence);

}

1 usage   Otavio Izidoro
public UserPresenceResponse updateContent(final ClassContentRequest request) {

    final UserPresence classContent = userPresenceDAO.updateClassContent(request);

    if(Objects.isNull(classContent)) {
        return null;
    }
    return userPresenceToUserPresenceResponseMapper.map(classContent);
}
```

A figura 2 representa a função “updatePresence” que é responsável por fazer a verificação se o usuário está inserindo uma nova aula, com presença ou falta, ou se ele está modificando de presença para falta e vice versa, para aquele dia específico. Caso a atualização no banco de dados através do método “userPresenceDAO.update” ocorra corretamente, então é chamado o método “userFrequencyService.updateFrequency” para atualizar a % de frequência do usuário, o parâmetro “itsNewClass” identifica que é a primeira vez que uma presença é marcada para aquele dia e será usado como validação para diferenciar o fluxo na hora de calcular a frequência.

A função “updateContent” recebe uma requisição para alterar o conteúdo referente àquele dia e atualiza no banco de dados do usuário.

2. Lógica de frequência:

Figura 3 - Lógica de frequência

```
public boolean updateFrequency (final UserPresence userPresence,
                                final boolean itsNewClass) {

    final var oldFrequency = userFrequencyDAO.findById(userPresence.getAccountId(), userPresence.getDay());

    if(oldFrequency.isEmpty()){
        System.out.println("Materia nao registrada!");
        return false;
    }
    final var frequencyToSave = oldFrequency.get();

    if(itsNewClass){
        return updateNewFrequency(frequencyToSave);
    }


    final UserFrequency invertedFrequency;
    if(userPresence.isPresence()){
        invertedFrequency = invertPresenceAndAbsence(frequencyToSave, presenceValue: 1, absenceValue: -1);
    }else{
        invertedFrequency = invertPresenceAndAbsence(frequencyToSave, presenceValue: -1, absenceValue: 1);
    }
    return updateNewFrequency(invertedFrequency);

}
```

Como visto na figura 3, primeiro busca-se a presença que precisa ser atualizada no banco, se ela não existir é retornado “false” para indicar que houve um problema ao atualizar a frequência. Em seguida é feito a verificação se a chamada do método foi feita para registrar uma nova presença com o parâmetro “itsNewClass” dessa forma apenas calcula-se sua frequência e então é salvo no banco através do método “updateNewFrequency”, caso o parâmetro seja “false” indicando que aquele registro não é uma nova marcação e sim uma atualização de uma aula que já foi marcada, então verifica-se se o usuário marcou presença ou falta para aquele dia, e inverte-se a quantidade de presenças e faltas do usuário para aquela matéria (ex: se ele tinha registrado 3 dias como presença e 1 dia como falta para aquela matéria e decidiu alterar um desses dias, inverte-se para que ele fique com 2 dias de presença e 2 dias de falta) e em seguida calcula-se o seu novo valor de frequência sempre baseado na quantidade de aulas que foram ministradas para aquela matéria.

3. Lógica do quiz utilizando ChatGPT

Figura 4 - Lógica quiz usando o ChatGPT

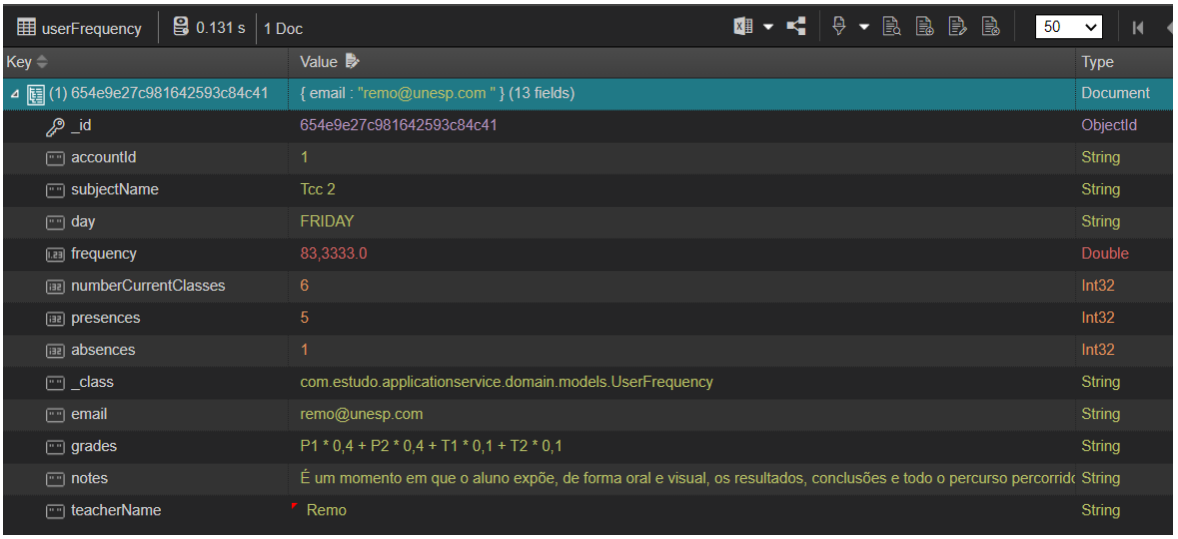
```java
public QuizResponse chatQuiz(final OpenaiRequest request) {
    request.setModel(this.model);
    final var choices = Arrays.stream(openaiClient.chat(baseUrl, apiKey, request).getChoices()).toList().get(0);

    return getQuiz(choices);
}

1 usage  Otavio Izidoro
private QuizResponse getQuiz(final Choice choice){
    final String content = choice.getMessage().getContent();
    final List<String> splitContent = Arrays.stream(content.split( regex: "\n\n")).toList();
    return buildQuizResponse(splitContent);
}

1 usage  Otavio Izidoro
private QuizResponse buildQuizResponse (final List<String> content){
    final QuizResponse response = new QuizResponse();
    final var LAST_ELEMENT_POSITION = content.size()-1;

    final var answers = Arrays.stream(content.get(LAST_ELEMENT_POSITION).split( regex: ",")).toList();

    response.setAnswers(answers);
    response.setQuestions(content.subList(0,LAST_ELEMENT_POSITION));
    return response;
}
```

Na figura 6 nota-se que a função “ChatQuiz” recebe uma requisição contendo o prompt que será utilizado para fazermos a chamada API do ChatGPT. O retorno da API é um Json contendo diversas informações, mas a resposta que espera-se está sendo extraída e armazenada na variável choices. O método “getQuiz” divide a resposta e a armazena em um List chamado “splitContent” de tamanho “n”, tendo em mente que as posições do array vão de 0 até n-1, as questões do quiz ficam armazenadas da posição 0 até a posição n-2 enquanto a última posição armazena as respostas das respectivas perguntas do quiz.

Dessa forma no método “buildQuizResponse” cria-se o objeto de resposta que será enviado para o frontEnd contendo dois campos, o campo “questions” é a lista de questões múltipla escolha e o campo “answers” é uma lista em que cada posição se refere a resposta da questão respectivamente. Abaixo está um exemplo de requisição e resposta para esse endpoint:

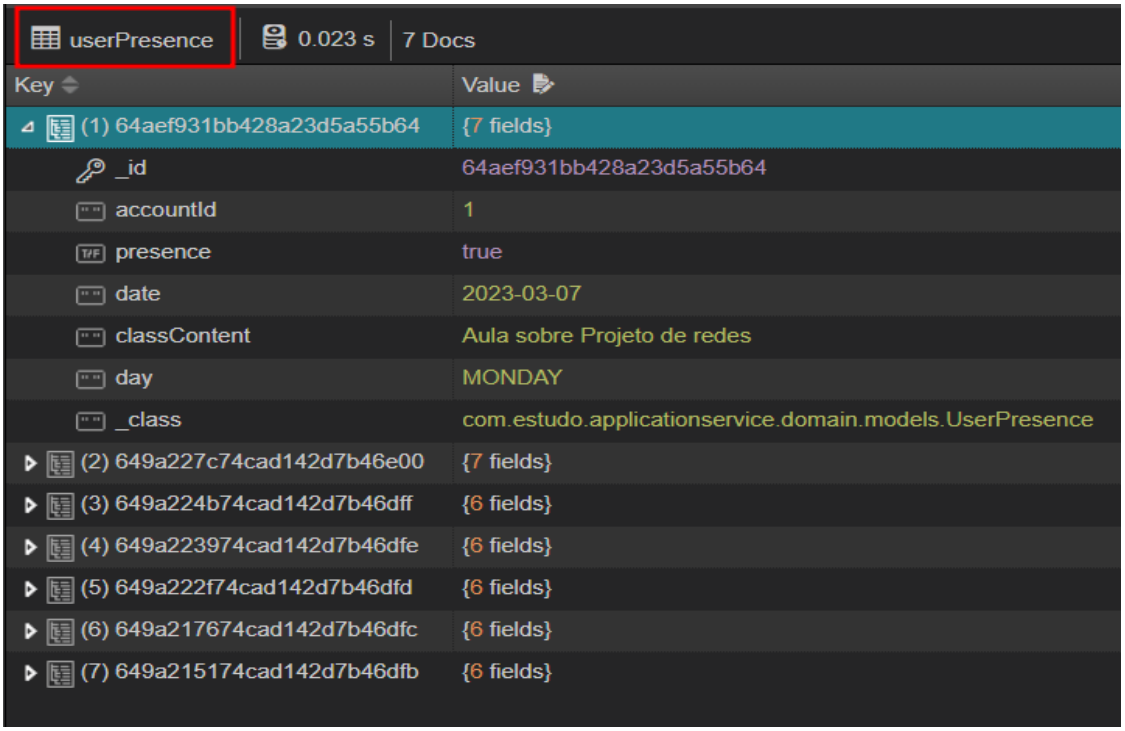
4. Collections no banco de dados mongoDB
    a. userFrequency: Contém o dia da semana, o nome da matéria, a frequência, número de aulas ministradas, número de presenças e de faltas, o campo accountID para fazer o gerenciamento e distinguir as matérias entre as contas existentes na aplicação, além das informações específicas nome do professor, email do professor, cálculo das notas e anotações gerais sobre a matéria.

Figura 5 - Banco de dados, userFrequency



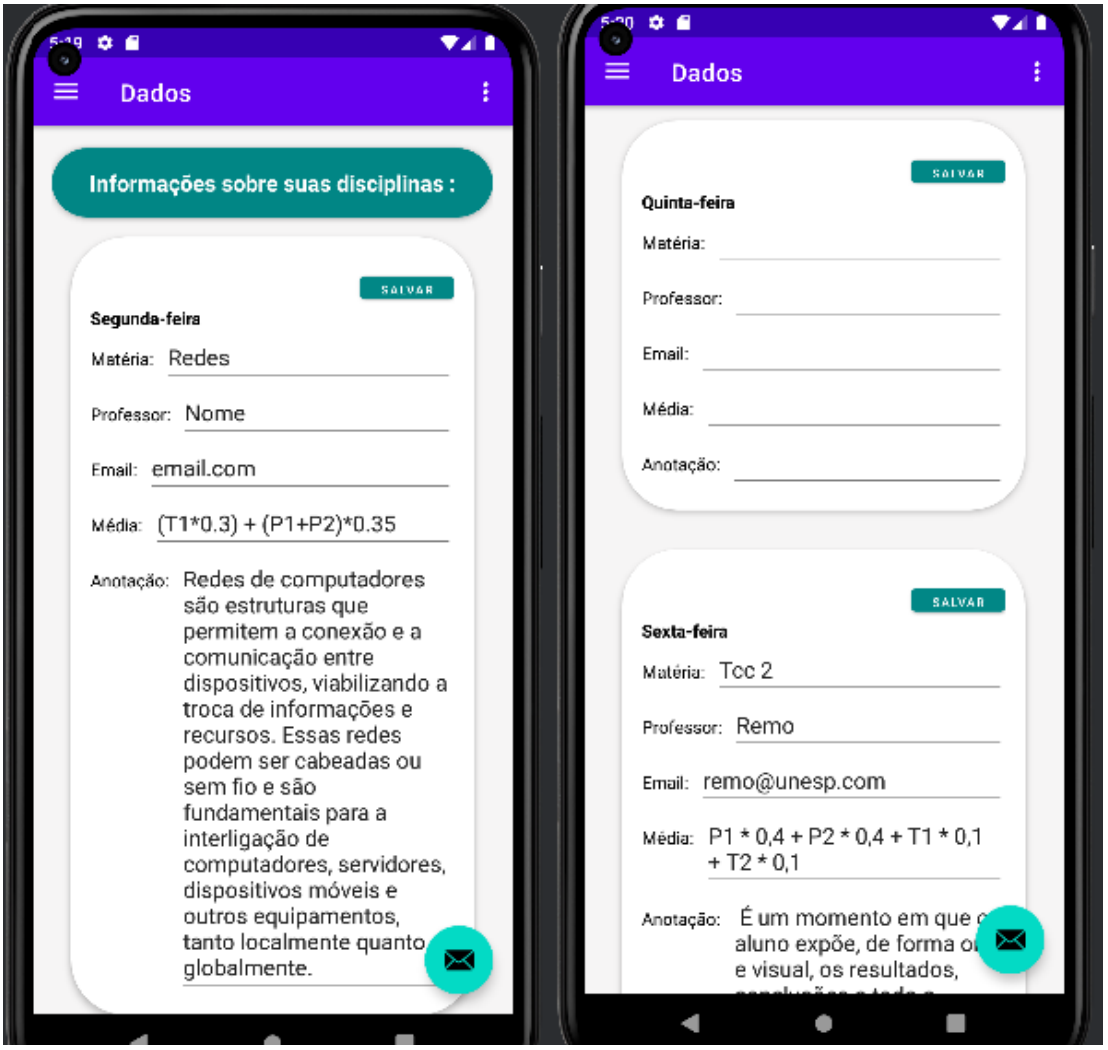
    b. userPresence: Contém informações referentes a um dia específico e uma matéria específica, no exemplo abaixo se refere a segunda-feira, marcado como “true” para o campo “presence” indicando que o usuário marcou presença para aquele dia, “date” é o data, usada como campo de validação para evitar que exista dois documentos para a mesma accountID e o mesmo dia (ou seja uma account só pode ter 1 documento para cada data, indicando que ele não poderia por exemplo marcar presença e falta para o mesmo dia/mês daquele ano) e “classContent” é o campo adicionado pelo usuário naquele dia em específico, que será usado como base para construir o prompt a ser enviado para gerar o quiz utilizando a API do ChatGPT, ele se refere ao conteúdo da matéria daquele dia.

Figura 6 - Banco de dados, userPresence



5. Registro dos dados da matéria

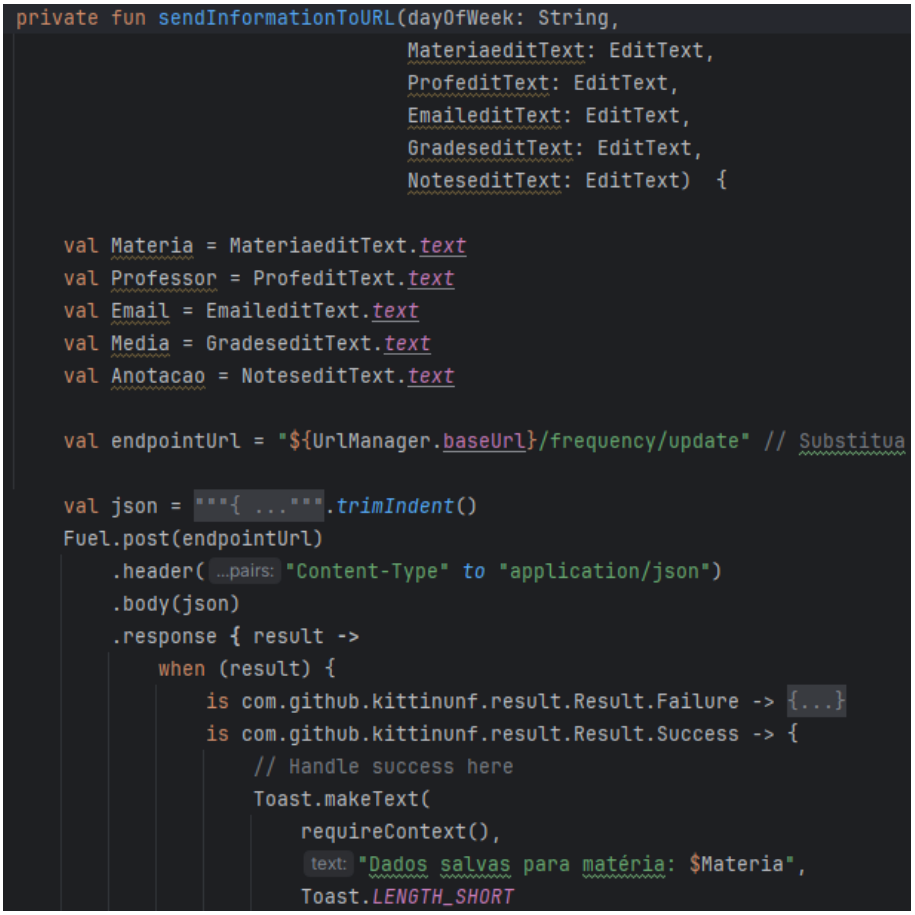
Figura 7 - Design do registro dos dados por dia da semana



A funcionalidade de cadastro de disciplinas no aplicativo, evidenciada pelos campos de inserção e anotações em cada dia da semana no fragmento, desempenha um papel crucial na integração entre as diversas telas do programa. Permitindo que o usuário registre, após salvar, informações como o nome da matéria, professor, e-mail e notas, o aplicativo facilita a organização e visualização centralizada de dados acadêmicos.

Ao oferecer a capacidade de anotar observações sobre a matéria, o aplicativo não apenas se adapta às necessidades individuais dos usuários, mas também promove um ambiente interativo e adaptável, proporcionando um espaço prático para registrar informações específicas e relevantes para cada disciplina.

Figura 8 - Lógica do registro dos dados por dia da semana



```
private fun sendInformationToURL(dayOfWeek: String,
                                 MateriaeditText: EditText,
                                 ProfeditText: EditText,
                                 EmaileditText: EditText,
                                 GradeseditText: EditText,
                                 NoteseditText: EditText)  {

    val Materia = MateriaeditText.text
    val Professor = ProfeditText.text
    val Email = EmaileditText.text
    val Media = GradeseditText.text
    val Anotacao = NoteseditText.text

    val endpointUrl = "${UrlManager.baseUrl}/frequency/update" // Substitua

    val json = """{ ..."""".trimIndent()
    Fuel.post(endpointUrl)
        .header( ...pairs: "Content-Type" to "application/json")
        .body(json)
        .response { result ->
            when (result) {
                is com.github.kittinunf.result.Result.Failure -> {...}
                is com.github.kittinunf.result.Result.Success -> {
                    // Handle success here
                    Toast.makeText(
                        requireContext(),
                        text: "Dados salvas para matéria: $Materia",
                        Toast.LENGTH_SHORT
```

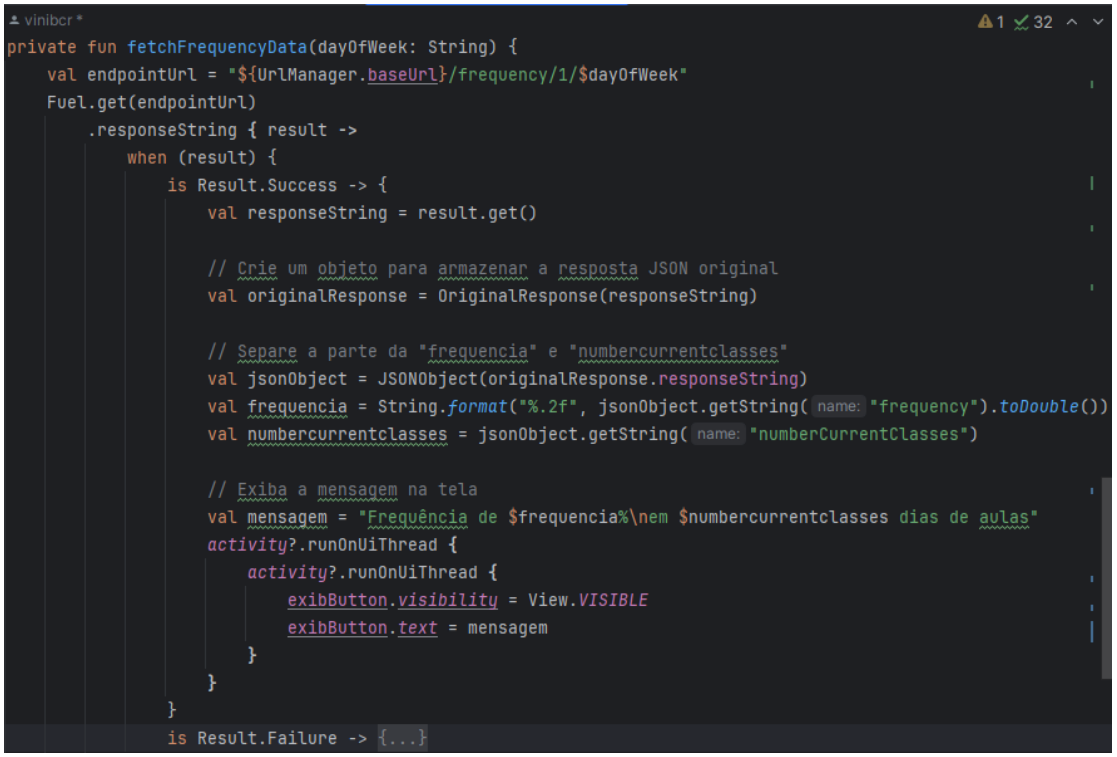
A interface do usuário na figura 7 exibe cinco conjuntos de campos para inserção e visualização de dados, como nome da disciplina, nome do professor, e-mail, notas e outras informações relacionadas a cada dia da semana. Os dados são obtidos e enviados para um servidor remoto por meio de requisições HTTP, utilizando a biblioteca Fuel para facilitar as chamadas.

6. Visualização da frequência por dia da semana

Figura 9 - Design da visualização da frequência

O código apresentado corresponde a um fragmento de um aplicativo Android que exibe informações de frequência em aulas para diferentes dias da semana. O fragmento contém botões representando cada dia útil, e ao clicar em um botão, é realizada uma chamada de API para obter dados de frequência associados a esse dia. A resposta da API é processada para extrair informações relevantes, como a porcentagem de frequência e o número de aulas realizadas até o momento, que são então exibidos dinamicamente na interface do usuário.

Figura 10 - Lógica da visualização da frequência



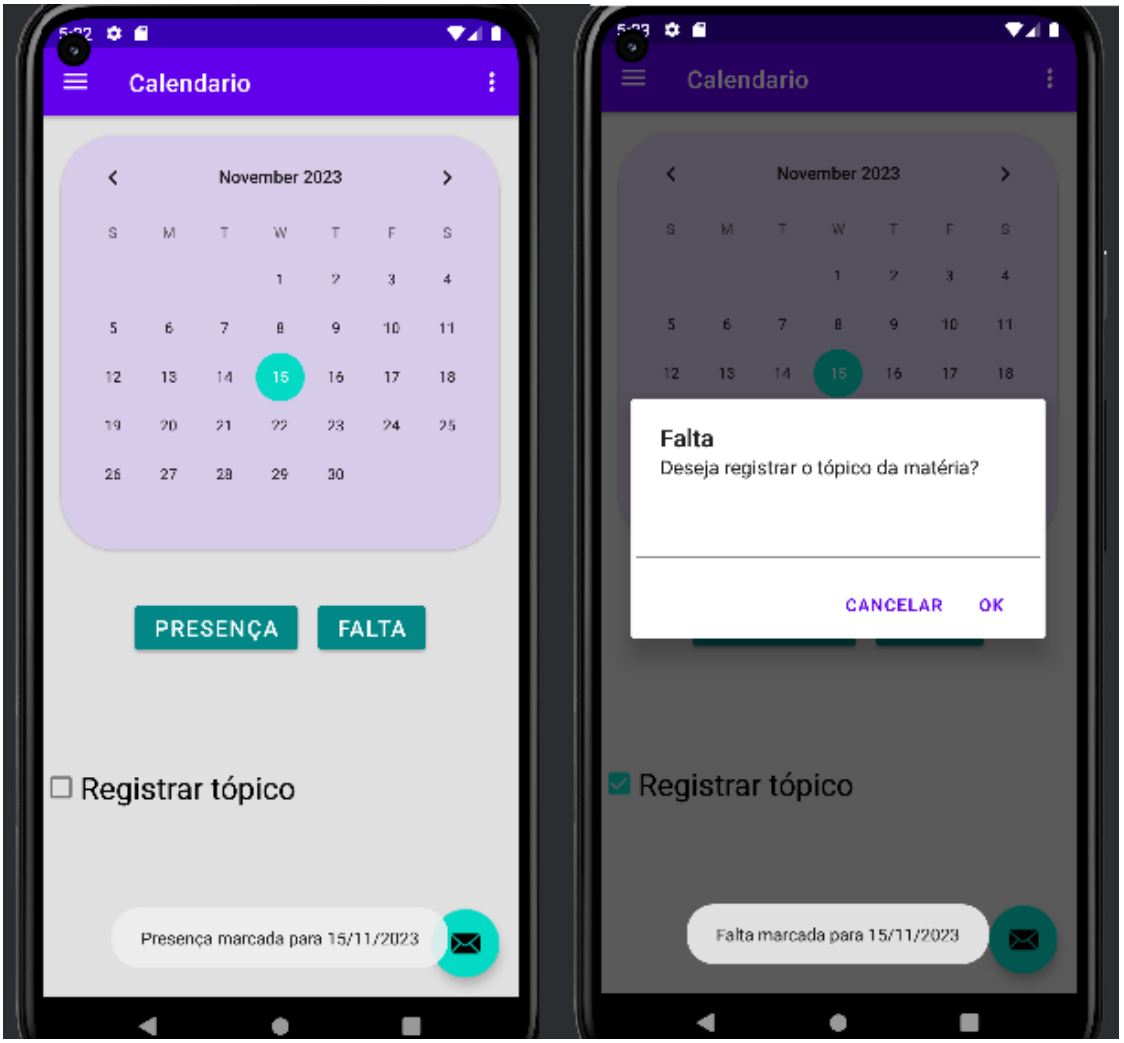
```
vinibcr *                                                                    1  32
private fun fetchFrequencyData(dayOfWeek: String) {
    val endpointUrl = "${UrlManager.baseUrl}/frequency/1/$dayOfWeek"
    Fuel.get(endpointUrl)
        .responseString { result ->
            when (result) {
                is Result.Success -> {
                    val responseString = result.get()

                    // Crie um objeto para armazenar a resposta JSON original
                    val originalResponse = OriginalResponse(responseString)

                    // Separe a parte da "frequencia" e "numbercurrentclasses"
                    val jsonObject = JSONObject(originalResponse.responseString)
                    val frequencia = String.format("%.2f", jsonObject.getString( name: "frequency").toDouble())
                    val numbercurrentclasses = jsonObject.getString( name: "numberCurrentClasses")

                    // Exiba a mensagem na tela
                    val mensagem = "Frequência de $frequencia%\nem $numbercurrentclasses dias de aulas"
                    activity?.runOnUiThread {
                        activity?.runOnUiThread {
                            exibButton.visibility = View.VISIBLE
                            exibButton.text = mensagem
                        }
                    }
                }
                is Result.Failure -> {...}
```

Ao clicar em botões correspondentes aos dias da semana, a função fetchFrequencyData é chamada para realizar uma requisição HTTP, obter dados relacionados à frequência do aluno no servidor e exibir as informações na tela. O código também trata erros de requisição, como falhas de conexão, exibindo mensagens de erro através de pop-ups.

Os dados relevantes são extraídos e apresentados de forma formatada na interface do usuário. A estrutura do código segue o padrão de arquitetura MVVM (Model-View-ViewModel), utilizando a classe FrequenciaViewModel para gerenciar a lógica de negócios relacionada à frequência. O aplicativo é destinado a presença do aluno em diferentes dias da semana, proporcionando uma experiência interativa e informativa.

7. Calendário de registro de presença

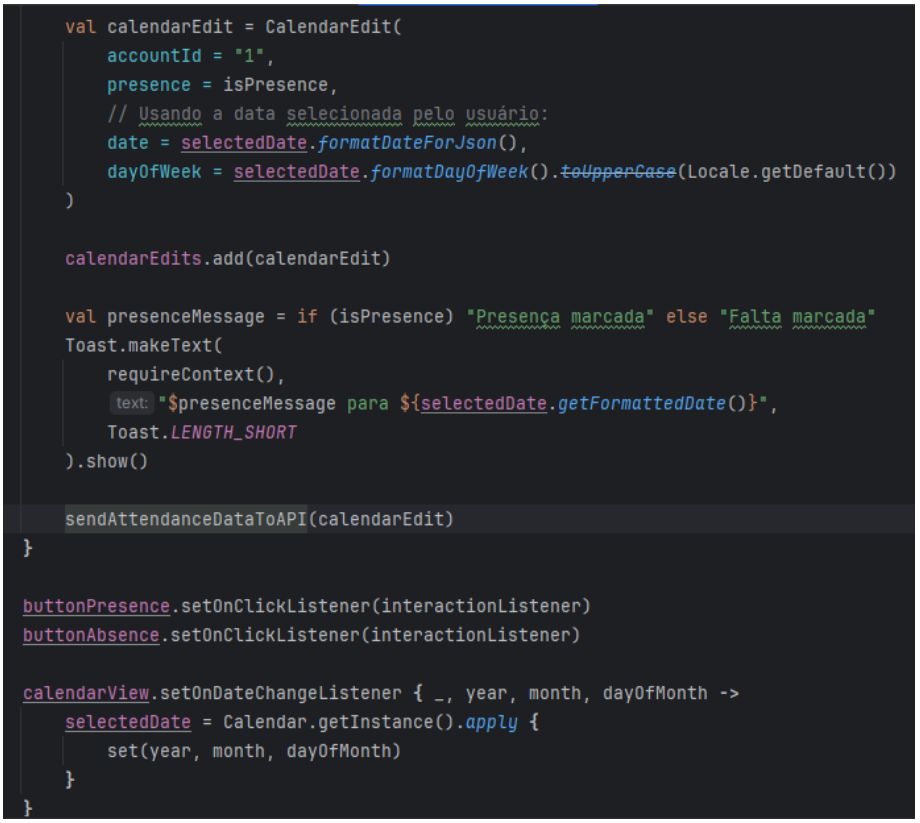
Figura 11 - Design do calendário de frequência



A tela Android em questão oferece aos usuários autonomia para registrar suas presenças ou faltas. Ao interagir com o calendário, os usuários podem selecionar a data desejada e, além de marcar a presença ou falta, têm a opção de registrar o tópico específico da matéria associada àquela data.

Esse recurso visa não apenas auxiliar na gestão de frequência, mas também proporciona uma oportunidade valiosa para reforçar o conteúdo por meio de quizzes, estabelecendo uma integração eficiente entre o registro de presença, a revisão de tópicos e a avaliação do conhecimento.

Figura 12 - Lógica do calendário de frequência



```
    val calendarEdit = CalendarEdit(
        accountId = "1",
        presence = isPresence,
        // Usando a data selecionada pelo usuário:
        date = selectedDate.formatDateForJson(),
        dayOfWeek = selectedDate.formatDayOfWeek().toUpperCase(Locale.getDefault())
    )

    calendarEdits.add(calendarEdit)

    val presenceMessage = if (isPresence) "Presença marcada" else "Falta marcada"
    Toast.makeText(
        requireContext(),
        text: "$presenceMessage para ${selectedDate.getFormattedDate()}",
        Toast.LENGTH_SHORT
    ).show()

    sendAttendanceDataToAPI(calendarEdit)
}

buttonPresence.setOnClickListener(interactionListener)
buttonAbsence.setOnClickListener(interactionListener)

calendarView.setOnDateChangeListener { _, year, month, dayOfMonth ->
    selectedDate = Calendar.getInstance().apply {
        set(year, month, dayOfMonth)
    }
}
```

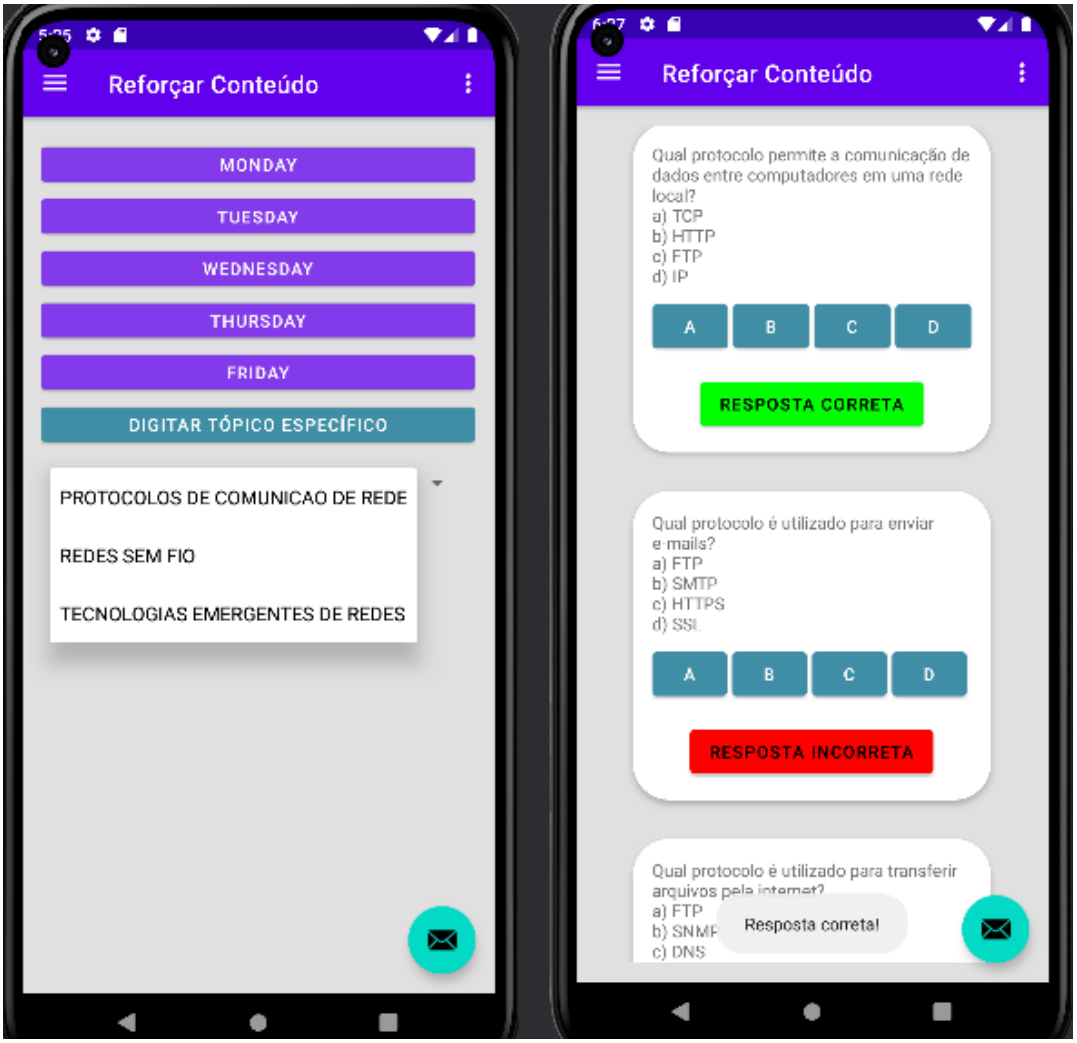
A funcionalidade principal na figura 12, ocorre quando o usuário seleciona uma data no calendário. As ações de presença ou falta são registradas no servidor, e uma mensagem informativa é exibida. Se a opção de registrar tópicos estiver marcada, uma caixa de diálogo é exibida para que o usuário insira um tópico relacionado à presença ou falta. Esse tópico é então registrado junto com a informação de presença ou falta no servidor.

Essa lógica garante que as informações de presença ou falta, juntamente com qualquer tópico associado, sejam registradas no servidor quando o usuário interage com o calendário. Os botões de presença e falta, assim como a seleção de datas, estão todos interligados para fornecer uma experiência de registro intuitiva para o usuário.

As respostas do servidor são tratadas, exibindo mensagens de sucesso ou informando sobre falhas. A estrutura do código é organizada em funções auxiliares para facilitar a leitura e manutenção do código, abordando aspectos como formatação de datas, conversão de objetos para JSON e envio de dados para o servidor.

8. Quiz para reforçar conteúdos

Figura 13 - Design da Integração dos tópicos com o Quiz



A figura 13 faz parte de uma funcionalidade em um aplicativo que permite aos usuários gerar quizzes com base nos tópicos salvos em determinados dias da semana selecionado, ou clicando no botão para gerar uma caixa de texto, onde a pessoa pode pesquisar outros temas usando a caixa de texto.

O código está dentro de uma tela que oferece a funcionalidade de criar quizzes com temas personalizados, trazidos do ChatGPT. Os quizzes são formados a partir de tópicos salvos para cada dia da semana. Os usuários podem selecionar um dia da semana específico e escolher um tópico associado a esse dia para criar um quiz, dando a opção para o usuário inserir manualmente outro tópico específico.

Funções são chamadas quando o usuário pressiona um botão de resposta no quiz, compara a resposta correta com a resposta selecionada pelo usuário e, com base no resultado, ajusta a aparência de um botão de envio e exibe um feedback visual.

A funcionalidade envolve a interação do usuário com botões correspondentes com os tópicos das matérias de determinados dias da semana e botões de resposta para cada pergunta do quiz. Quando um usuário responde a uma pergunta, o código verifica se a resposta está correta e fornece feedback visual imediato na interface do usuário, com cores, indicando se a resposta foi correta ou incorreta.

8. Conclusão

Este trabalho de conclusão de curso apresenta uma aplicação acadêmica abrangente e eficiente, destacando-se pela implementação cuidadosa da lógica de presença, frequência e quiz com o ChatGPT. A integração com o banco de dados MongoDB assegura a organização e recuperação eficaz dos dados, enquanto a interface do usuário reflete uma abordagem intuitiva para o registro de informações acadêmicas. A aplicação não apenas simplifica a gestão da frequência e presença, mas também inova ao incorporar o ChatGPT para criar quizzes dinâmicos, enriquecendo a experiência educacional do usuário. Essa abordagem integrada, aliada à interatividade e praticidade proporcionadas pela aplicação, representa um avanço significativo no contexto acadêmico.

No desenvolvimento da aplicação, a lógica de registro de dados por dia da semana e a visualização da frequência demonstram adaptabilidade às necessidades individuais dos usuários. A integração dos tópicos com o quiz reforça o compromisso com a eficiência, criando uma conexão coerente entre o registro de presença, revisão de tópicos e avaliação do conhecimento. Em resumo, este TCC destaca-se pela criação de uma solução integrada e versátil que não apenas atende às demandas acadêmicas, mas também eleva a experiência do usuário ao proporcionar uma gestão acadêmica mais organizada, interativa e produtiva.

Referências

ALVES, E. A. A evasão escolar: um problema complexo que exige soluções integradas. *Cadernos de Pesquisa*, v. 39, n. 136, p. 687-701, 2009.

BORTOLOZZO, A. S.; SANTOS, R. S. Aplicação mobile para controle de frequência escolar. *Revista Ciência em Extensão*, v. 13, n. 1, p. 28-37, 2017.

COSTA, E. M.; SILVA, M. A. Gerenciamento de tarefas no ensino superior: análise de aplicativos móveis. In: Anais do Congresso Nacional de Educação, v. 3, n. 1, p. 1-13, 2015.

FARIA, R. T.; SILVA, M. V. Gerenciamento de tarefas em ambientes virtuais de aprendizagem: uma revisão integrativa. In: Anais do Congresso Brasileiro de Informática na Educação, v. 27, n. 1, p. 1-10, 2016.

FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Tese de doutorado, University of California, Irvine, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.

GALDINO, R. A. Sistema de controle de frequência e notas para instituições de ensino. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação). Universidade Federal de Mato Grosso, Campus Cuiabá, 2014.

GUIMARÃES, F. L. Aplicativo mobile para comunicação entre alunos e professores. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação). Universidade Federal de Minas Gerais, Campus Belo Horizonte, 2015.

INOUE, K.; SMITH, J.; BROWN, A. Challenges of Student Time Management in Higher Education. *Journal of Educational Research*, v. 45, n. 2, p. 123-145, 2020.

JetBrains. *IntelliJ IDEA.* Disponível em: <https://www.jetbrains.com/idea/>.

Kotlin. *Kotlin Programming Language.* Disponível em: <https://kotlinlang.org/>.

Oracle. *Java SE Documentation.* Disponível em: <https://docs.oracle.com/en/java/javase/index.html>.

SILVA, M. A.; SANTOS, P. Q. The Impact of Student Disorganization on Academic Performance. *International Journal of Educational Psychology*, v. 30, n. 4, p. 567-589, 2018.