UNISAL Americana
Engenharia de Computação
Projeto Integrador e Projeto de Vida I
1S2023

FinApp

Alysson Roberto Fabreti, alyssonfr626@gmail.com
Igor Santiello Gonçalves Silvério, igor.santiello17@gmail.com
Pedro Augusto da Silva Abel, pedro11.abel@gmail.com
Thiago Crochat, crochatthiago@gmail.com
Vinicius Felipe Basso, vinifbasso2@gmail.com

Resumo *– O projeto em questão visa criar um aplicativo com o objetivo de oferecer apoio financeiro às pessoas, auxiliando-as no controle de suas finanças. Para alcançar esse objetivo, serão adotadas algumas heurísticas durante o desenvolvimento do software. O aplicativo proporcionará aos usuários uma ferramenta eficiente para gerenciar seus gastos e auxiliar no desenvolvimento de uma vida financeira saudável.*
Para a implementação desse projeto, será utilizada a framework Flutter, que é amplamente reconhecida e utilizada para criar aplicativos móveis. Essa escolha permitirá a criação de versões do aplicativo compatíveis com dispositivos móveis.

Palavras-chave*: Aplicativo, controle, financeiro, flutter.*

I. INTRODUÇÃO

O objetivo deste projeto é desenvolver um aplicativo móvel para controle de gastos pessoais, com recursos para auxiliar os usuários a gerenciar suas finanças de forma mais eficiente e prática.

Infelizmente, muitas pessoas enfrentam dificuldades em gerenciar suas finanças pessoais e acabam gastando mais do que podem. Essa situação pode levar à acumulação de dívidas, problemas financeiros e até mesmo estresse e ansiedade.

Para combater esse problema, o aplicativo de controle de gastos pessoais será desenvolvido com uma série de recursos que permitirão aos usuários gerenciar suas finanças de forma mais consciente e eficiente.

Entre as funcionalidades do aplicativo, estão o cadastro de despesas diárias, definição de metas de economia, visualização de gráficos e relatórios detalhados, notificações de faturas a vencer e planejamento de despesas futuras. O aplicativo também terá funções como categorização automática de despesas, alertas de orçamento, integração com contas bancárias, análise de tendências, metas financeiras, calculadora de dívida, comparação de preços, integração com sistemas de investimentos e relatórios de despesas.

O aplicativo será desenvolvido utilizando o framework Flutter e a linguagem de programação Dart, que permitem a criação de aplicativos nativos para Android a partir de um único código base e possivelmente integrado para plataformas iOS. A arquitetura utilizada será baseada no padrão de design MVC (Model-View-Controller), que separa a lógica de negócios, a interface do usuário e a comunicação entre os dois.

Ainda, espera-se que o aplicativo possa impactar positivamente a sociedade, contribuindo para a educação financeira, a saúde financeira e a inclusão financeira das pessoas. Ao oferecer uma ferramenta fácil e prática para controlar as finanças pessoais, o aplicativo pode ajudar as pessoas a compreender melhor seus gastos e a desenvolver hábitos financeiros mais saudáveis. Por exemplo, o aplicativo pode disponibilizar gráficos e relatórios que detalham os gastos do usuário, permitindo uma análise mais aprofundada dos hábitos de consumo. Oferecendo dicas e sugestões personalizadas para ajudar o usuário a reduzir seus gastos e economizar dinheiro.

Outro ponto importante é que o aplicativo pode ajudar as pessoas a evitar dívidas e a reduzir suas despesas desnecessárias, contribuindo para uma maior estabilidade financeira. O aplicativo também pode ser uma ferramenta importante para o planejamento financeiro a longo prazo, ajudando os usuários a economizar para alcançar objetivos como a compra de uma casa ou a aposentadoria.

Por fim, o aplicativo pode ser uma ferramenta importante para a inclusão financeira, permitindo que pessoas que não têm acesso a serviços financeiros mais complexos possam controlar seus gastos e planejar suas finanças com mais facilidade. Isso pode contribuir para uma maior equidade social e econômica, já que mais pessoas terão acesso a recursos para melhorar sua qualidade de vida. O aplicativo também pode incentivar os usuários a poupar dinheiro, investir em suas metas financeiras e se preparar para o futuro, promovendo ainda mais estabilidade financeira e bem-estar para a sociedade. Com possibilidade para cadastrar despesas diárias, definir metas de economia, visualizar gráficos e relatórios detalhados, receber notificações de faturas a vencer e planejar despesas futuras, o aplicativo pode ajudar os usuários a tomar decisões financeiras mais conscientes e a desenvolver hábitos financeiros mais saudáveis.

Para o desenvolvimento deste aplicativo, foi utilizado o framework Flutter, que permite criar aplicativos nativos para a plataforma Android. Será utilizada a linguagem de programação Dart, que é nativa do Flutter e tem uma sintaxe clara e simples para realizar regras de negócios necessárias para o desenvolvimento da aplicação.

Para a estruturação do aplicativo, foi utilizada uma arquitetura baseada no padrão de design MVC (Model-View-Controller) ajustando a arquitetura de uma forma que temos os princípios fundamentais do padrão na aplicação, mas também adaptando para as necessidades do aplicativo, que separa a lógica de negócios, a interface do usuário e a comunicação entre os dois.

II. IMPACTO NA SOCIEDADE

A educação financeira desempenha um papel fundamental na vida das pessoas, capacitando-as a tomar decisões conscientes e responsáveis em relação ao dinheiro. No contexto brasileiro, a falta de conhecimento financeiro é uma realidade que afeta indivíduos e famílias, contribuindo para endividamento excessivo, dificuldades financeiras e falta de planejamento para o futuro.

Diversos estudos científicos têm demonstrado a relação positiva entre a educação financeira e melhores resultados financeiros. Pesquisas revelam que indivíduos com maior conhecimento financeiro tendem a acumular mais poupanças, reduzir dívidas, investir de forma mais eficiente e evitar práticas financeiras de risco. Exemplo de um estudo relevante é o trabalho de Lusardi e Mitchell (2014), que evidencia os benefícios da educação financeira em termos de resultados financeiros de longo prazo.

A educação financeira é um componente essencial para o desenvolvimento econômico e social de um país. No Brasil, é crucial investir na promoção da educação financeira, utilizando evidências científicas para embasar políticas e práticas educacionais. A tecnologia desempenha um papel importante, oferecendo ferramentas acessíveis e interativas para facilitar a aprendizagem financeira. Através de esforços conjuntos, envolvendo governo, instituições educacionais e sociedade civil, é possível superar os desafios e aproveitar as oportunidades para promover uma educação financeira sólida e abrangente para todos os brasileiros, capacitando-os a tomar decisões financeiras informadas e melhorar sua qualidade de vida.

No Brasil, a educação financeira ainda enfrenta desafios significativos. A falta de inclusão da educação financeira nas grades curriculares escolares, a escassez de recursos e a falta de formação adequada para os professores são alguns dos obstáculos a serem superados. No entanto, também existem oportunidades para avançar nesse campo. A implementação de políticas públicas que incentivem a inclusão da educação financeira nas escolas, a promoção de parcerias entre instituições públicas e privadas e o desenvolvimento de programas de capacitação para educadores são algumas das oportunidades que podem impulsionar a educação financeira no Brasil.
A tecnologia tem desempenhado um papel significativo na promoção da educação financeira, oferecendo soluções acessíveis e interativas para os indivíduos. Aplicativos móveis, plataformas online e ferramentas digitais têm se mostrado eficazes na disseminação de conhecimentos financeiros e no desenvolvimento de habilidades práticas de gestão financeira. Estudos mostram que o uso de tecnologia na educação financeira pode aumentar a motivação, o engajamento e a compreensão dos conceitos financeiros por parte dos estudantes, permitindo que eles apliquem o conhecimento adquirido em situações reais.

Com o acompanhamento dos gastos em tempo real, o usuário tem uma visão clara da sua situação financeira e pode tomar decisões mais conscientes em relação ao seu dinheiro.

O aplicativo pode ajudar as pessoas a evitar dívidas e a reduzir suas despesas desnecessárias, contribuindo para uma maior estabilidade financeira. Isso pode ter um efeito positivo na sociedade como um todo, já que uma população financeiramente saudável tem mais recursos para investir em áreas como educação, saúde e bem-estar. O aplicativo também pode ser uma ferramenta importante para o planejamento financeiro a longo prazo, ajudando os usuários a economizar para alcançar objetivos como a compra de uma casa ou a aposentadoria.

III. DESENVOLVIMENTO

No caos diário do mundo moderno, manter o controle das finanças pessoais pode ser um verdadeiro desafio. Felizmente, a tecnologia veio para simplificar essa tarefa. Com um aplicativo de controle de gastos pessoais, você terá uma solução prática e eficiente ao alcance das mãos. Neste artigo, vamos explorar as funcionalidades-chave desse aplicativo revolucionário e como ele pode transformar a maneira como você lida com suas finanças.

No mundo agitado em que vivemos, é fácil se perder nas despesas do dia a dia. Com um aplicativo de controle de gastos pessoais, você pode finalmente ganhar o controle sobre suas finanças e transformar sua vida financeira de uma vez por todas. Vamos conhecer algumas das principais funcionalidades que tornam esse aplicativo uma ferramenta indisponível.

1. Cadastro de despesas diárias: Uma das principais funcionalidades de um aplicativo de controle de gastos pessoais é a capacidade de registrar e categorizar nossas despesas diárias. Ao inserir as informações sobre nossos gastos, como compras no supermercado, transporte e outras despesas, o aplicativo nos fornece uma visão clara de como estamos utilizando nosso dinheiro. Essa funcionalidade nos ajuda a identificar padrões de gastos, eliminar desperdícios e tomar decisões mais conscientes sobre o uso do nosso dinheiro.

2. Definição de metas de economia: Estabelecer metas de economia é fundamental para alcançar a independência financeira. Com um aplicativo de controle de gastos pessoais, podemos definir metas específicas, como economizar uma porcentagem do nosso salário todo mês. O aplicativo acompanha o nosso progresso em relação a essas metas, oferecendo insights sobre nosso desempenho e incentivando-nos a manter o foco. Ao estabelecer metas realistas e monitorar nosso progresso, podemos alcançar a liberdade financeira mais rapidamente.

3. Visualização de gráficos e relatórios detalhados: Compreenda seus gastos tendo uma visão clara de onde está gastando seu dinheiro sem ter que analisar pilhas de recibos e extratos bancários. Com um aplicativo de controle de gastos pessoais. O aplicativo oferece gráficos e relatórios detalhados sobre nossos gastos, permitindo uma análise visual e intuitiva de nossos hábitos financeiros. Podemos identificar facilmente categorias de gastos que estão consumindo uma parcela significativa de nossos recursos e tomar medidas para ajustar nosso orçamento.

4. Notificações de faturas a vencer: Evitar atrasos no pagamento de faturas é crucial para manter uma boa saúde financeira. Um aplicativo de controle de gastos pessoais pode ser seu aliado nessa tarefa, enviando notificações e lembretes sobre faturas a vencer. Seja a fatura do cartão de crédito, contas de serviços públicos ou empréstimos, o aplicativo garantirá que você não esqueça nenhuma data importante, evitando assim juros e multas desnecessárias.

5. Planejamento de despesas futuras: Conquiste seus sonhos todos temos sonhos e objetivos financeiros, como uma viagem especial ou a compra de um carro novo. Com um aplicativo de controle de gastos pessoais, podemos planejar e acompanhar o progresso dessas despesas futuras. O aplicativo nos ajuda a estabelecer um plano realista, estipulando quanto precisamos economizar e por quanto tempo, para alcançar nossos objetivos financeiros. Dessa forma, estamos mais próximos de realizar nossos sonhos, enquanto mantemos uma visão clara e organizada de nossas finanças.

IV. ARQUITETURA

A estruturação de um aplicativo financeiro com cadastro de despesas diárias, definição de metas de economia, visualização de gráficos e relatórios detalhados, e planejamento de despesas futuras requer uma arquitetura em camadas bem definida. A adoção da arquitetura MVC, juntamente com componentes de conexão com o banco de dados, serviços de lógica de negócios, modelos de dados, DAO, uma API bem definida, injeção de dependências, middlewares e pipelines, pode facilitar o desenvolvimento, a manutenção e a escalabilidade do aplicativo. Essa estruturação permite uma separação clara de responsabilidades, promove a reutilização de código, torna o aplicativo mais modular e flexível, e oferece recursos avançados, como autenticação, autorização e login.

1. Definindo a arquitetura em camadas: Uma arquitetura em camadas adequada é essencial para desenvolver um aplicativo escalável, de fácil manutenção e reutilização de código. Recomenda-se a utilização da arquitetura MVC (Model-View-Controller), que divide a aplicação em três camadas principais

    1.1. Camada de Modelo (Model): Responsável por representar e manipular os dados da aplicação. Os modelos definem a estrutura dos dados, como despesas, metas de economia, gráficos e relatórios. Eles fornecem métodos para acessar e manipular esses dados.

    1.2. Camada de Visualização (View): Responsável por exibir a interface do usuário e interagir com o usuário. Nessa camada, são apresentadas as telas para o cadastro de despesas diárias, definição de metas de economia, visualização de gráficos ou relatórios, e planejamento de despesas futuras. Ela recebe as entradas do usuário e as transmite para a camada de controle para processamento.

    1.3. Camada de Controle (Controller): Atua como intermediária entre a camada de modelo e a camada de visualização. Recebe as entradas do usuário da camada de visualização, processa a lógica de negócios, interage com a camada de modelo para obter ou atualizar os dados necessários e retorna os resultados para a camada de visualização. Os controladores processam as operações de cadastro de despesas, definição de metas de economia, geração de gráficos e relatórios detalhados, e planejamento de despesas futuras.

2. Gerenciamento de Dados: Para lidar com o armazenamento e recuperação de dados, é necessário estabelecer conexões com o banco de dados. A camada de modelo pode incluir componentes que gerenciam a comunicação com o banco de dados, realizando consultas, atualizações e outras operações relacionadas ao armazenamento de dados. Esses componentes fornecem uma interface para a camada de controle acessar os dados necessários.

3. Lógica de Negócios: A lógica de negócios da aplicação é encapsulada nos serviços. Os serviços executam as operações principais da aplicação, como cálculos relacionados a metas de economia, informação de gráficos e detalhes, cálculos para o planejamento de despesas futuras e validações de dados. Eles oferecem uma interface para que outras partes da aplicação possam interagir com essa lógica.

4. Acesso a Dados: O acesso aos dados é realizado através dos Data Access Objects (DAO). O DAO fornece uma interface abstrata para a camada de acesso a dados, encapsulando a lógica de acesso ao banco de dados. Eles fornecem métodos para realizar operações de leitura e escrita nos modelos de dados da aplicação, como selecionar despesas, atualizar metas de economia ou deletar transação.

5. Exposição da Funcionalidade: A funcionalidade do aplicativo financeiro será exposta através de uma API (Application Programming Interface). A API define os endpoints e os formatos de dados utilizados nas comunicações. Ela permite que a aplicação interaja com o aplicativo financeiro, realizando requisições de dados necessários. A API também pode oferecer autenticação e autorização para garantir a segurança das operações.

6. Gerenciamento de Dependências: O padrão de injeção de dependências (DI) pode ser utilizado para facilitar a criação de componentes independentes, aumentar a modularidade e facilitar os testes unitários. O DI permite injetar dependências em objetos em tempo de execução, garantindo que cada componente receba as dependências necessárias para funcionar corretamente. Isso reduz o acoplamento entre os componentes e torna o código mais flexível e fácil de manter.

V. FRONTEND

A página inicial de um aplicativo de gestão financeira é uma parte crucial para fornecer aos usuários uma visão clara e concisa de sua situação financeira. O frontend dessa página desempenha um papel fundamental ao apresentar as informações de forma organizada e intuitiva. Vamos explorar os principais elementos que podem ser incluídos nessa página.

O elemento mais importante é o saldo atual. Ele representa a quantia de dinheiro disponível para o usuário e geralmente é exibido de forma proeminente no topo da página inicial. O saldo pode ser destacado com um design de destaque, como uma fonte maior ou uma cor diferenciada, para chamar a atenção do usuário.

Junto do saldo, é útil fornecer informações sobre as despesas e as receitas do usuário. O valor das despesas pode ser associado a um bloco separado, entre despesas como alimentação, moradia, transporte, entre outras. Isso ajuda o usuário a entender como seu dinheiro está sendo gasto. Da mesma forma, o valor das rendas pode ser exibido a uma associação separado, junto a fontes de renda, como salário, investimentos, aluguéis, entre outros.

Uma lista de transações recentes também é uma parte valiosa da página inicial. Ela exibe as transações mais recentes do usuário, incluindo o tipo de transação (despesa ou renda), o nome da transação e o valor. Essa lista pode ser apresentada em um formato de lista, com a opção de classificar as transações por tipo (Renda ou Despesa). É benéfico incluir recursos de pesquisa e filtragem para permitir que o usuário encontre transações específicas com facilidade.

Além desses elementos principais, existem outras informações importantes para o frontend da página inicial de um app de gestão financeira. A interface deve ser responsiva, adaptando-se a diferentes tamanhos de tela, como smartphones, tablets e computadores. O design deve ser limpo e intuitivo, com uma hierarquia clara de informações e uma paleta de cores que transmita uma sensação de confiança e segurança. Gráficos visuais também podem ser usados para apresentar informações financeiras de forma mais visual e compreensível.

Figura 1 – Tela Principal

Fonte: Aplicação Flutter

VI. BACKEND

A camada de backend desempenha um papel fundamental no desenvolvimento de uma aplicação fullstack, sendo responsável pela lógica de negócios, manipulação de dados e comunicação com serviços externos. Ao analisar o planejamento dessa camada, fica evidente como a sua estrutura foi concebida com o objetivo de garantir a modularidade, escalabilidade e manutenibilidade da aplicação como um todo.

No processo de planejamento, é crucial estabelecer uma arquitetura robusta que atenda às necessidades da aplicação e identifique quais recursos serão utilizados. Uma abordagem comum é a adoção de padrões arquiteturais, como o Model-View-Controller (MVC) ou o Modelo de Domínio Driven (DDD), que orientam a organização e a separação de responsabilidades entre os diferentes componentes do backend. Durante o planejamento do backend, é de suma importância considerar a modularidade como um princípio fundamental. A divisão em diretórios e pastas específicos permite uma organização clara, facilitando a localização e a manutenção dos arquivos relacionados a cada aspecto da aplicação.

O módulo fundamental da nossa aplicação é a infraestrutura, que desempenha um papel essencial na estrutura e funcionamento do sistema. Um dos componentes cruciais desse módulo é o banco de dados MySQL, que é responsável por armazenar e gerenciar os dados utilizados pela aplicação. Ele fornece uma conexão confiável e eficiente para que nossa aplicação possa acessar e manipular esses dados de forma segura. Além do banco de dados, também utilizamos a técnica de Injeção de Dependências dentro da infraestrutura. Essa técnica desempenha um papel importante na arquitetura do nosso sistema, pois permite abstrair o conteúdo e as funcionalidades de um objeto no qual outros objetos dependem. Com a Injeção de Dependências, podemos separar as abstrações e responsabilidades dos objetos, o que resulta em um código mais modular e fácil de manter.

Ao utilizar a Injeção de Dependências, os serviços que dependem de um objeto não precisam conhecer os detalhes de sua construção. Em vez disso, podem simplesmente utilizar uma dependência fornecida por um arquivo externo. Essa abordagem traz benefícios significativos, como a separação de preocupações e a facilitação da gestão da infraestrutura da aplicação. Dessa forma, podemos modificar e atualizar componentes da infraestrutura sem afetar diretamente os serviços que dependem deles, promovendo uma maior flexibilidade e escalabilidade do sistema.

Além disso, dentro do módulo de infraestrutura, temos um submódulo dedicado à segurança. Esse submódulo desempenha um papel fundamental na proteção da aplicação contra ameaças e na garantia de que apenas usuários autorizados tenham acesso aos recursos. Uma das principais técnicas utilizadas é o JWT (JSON Web Token), um padrão aberto que permite a autenticação e autorização de usuários. O JWT é um token criptografado que contém informações sobre o usuário e suas permissões. Ele é enviado pelo cliente em cada requisição e, com base em sua validação, o backend pode permitir ou negar o acesso a determinadas rotas e funcionalidades.

Além da autenticação e autorização, o submódulo de segurança também inclui a implementação de outras funcionalidades importantes, como o gerenciamento de permissões. Isso envolve a definição de papéis e privilégios para diferentes tipos de usuários, permitindo um controle granular sobre quais recursos cada um pode acessar. Essa abordagem contribui para a segurança e a integridade dos dados da aplicação. É essencial investir em um robusto submódulo de segurança para garantir que a aplicação esteja protegida contra vulnerabilidades e ataques maliciosos. Ao adotar práticas sólidas de segurança, é possível aumentar a confiança dos usuários, proteger informações sensíveis e preservar a reputação da aplicação no mercado.

Dentro do módulo TO (Transfer Object), temos a capacidade de acessar dados de outros objetos fora de seu contexto. Esse módulo tem como objetivo definir objetos de transferência de dados relacionados à autenticação da aplicação, como os campos do Usuário, que seguem a mesma estrutura do modelo de Usuário. Esses objetos atuam como representações simplificadas dos dados durante a comunicação entre os diferentes componentes do backend. Essa abordagem facilita a transferência de informações específicas entre os componentes, evitando o compartilhamento desnecessário de dados confidenciais e promovendo a modularidade e a separação de preocupações na aplicação.

O módulo de utilidades abriga arquivos utilitários que fornecem funções ou configurações auxiliares para a aplicação. Um exemplo relevante da utilização desse módulo é a construção de variáveis de ambiente, onde é possível armazenar configurações personalizadas do ambiente de desenvolvimento. Isso permite que a aplicação se adapte a diferentes contextos, e garante a segurança do armazenamento de valores importantes, como as configurações de conexão do banco de dados, que podem ser importadas conforme necessário.
Esses utilitários proporcionam uma abordagem modular para a implementação de funcionalidades auxiliares, facilitando a reutilização de código.

Um dos módulos de extrema importância em um sistema de aplicação é o DAO (Data Access Object), responsável pelo acesso aos dados. O DAO atua como uma camada intermediária que se comunica diretamente com o banco de dados, permitindo lidar com operações como INSERT, DELETE e SELECT no MySQL. Para cada entidade da aplicação, é criado um DAO correspondente, que pode ser modificado conforme as necessidades surgirem.
A separação do acesso aos dados por meio do DAO é fundamental, pois permite encapsular as operações de acesso em classes dedicadas, tornando o código mais seguro. Essa abordagem ajuda a manter cada módulo com seu objetivo específico, particionando o código de forma organizada. Essa estrutura facilita a manutenção do sistema e contribui para sua escalabilidade, pois alterações em uma parte específica não afetam o funcionamento de outras partes.

O módulo de Services desempenha um papel crucial na arquitetura da aplicação, pois é responsável por fornecer os serviços necessários para encontrar elementos, encontrar todos os elementos, salvar e deletar. Ele pode ser visto como a camada de lógica de negócios da aplicação, que se conecta ao DAO para realizar as operações desejadas.
Uma prática comum na criação dos serviços é a implementação de um serviço genérico, que contém funcionalidades genéricas utilizadas por diferentes partes da aplicação. Essas funcionalidades podem ser implementadas por serviços específicos, como os serviços de login e de transações. Dessa forma, a arquitetura do sistema se beneficia da reutilização de código e da modularidade, tornando-o mais flexível e de fácil manutenção.

Os serviços específicos são responsáveis por implementar serviços especializados, como autenticação de usuários e manipulação de transações. Cada serviço específico tem a responsabilidade de executar a lógica de negócios correspondente à sua área de atuação. Essa abordagem permite que cada funcionalidade seja desenvolvida e testada de forma independente, simplificando a manutenção e o teste do sistema como um todo.

O módulo Models desempenha um papel crucial na definição dos objetos de dados utilizados pela aplicação e é um dos pilares do padrão MVC (Model-View-Controller), onde construímos as entidades correspondentes ao banco de dados no módulo de Infraestrutura.

Os arquivos de tipo model são compostos por classes que não contêm informações sobre como os dados serão tratados ou utilizados. Eles são responsáveis por descrever a estrutura e o comportamento desses objetos. Essa abordagem permite uma separação clara das responsabilidades no sistema, onde o módulo Models é responsável por representar e manipular os dados de forma consistente.

Ao definir modelos, podemos estabelecer as propriedades e os métodos necessários para interagir com os dados. Esses modelos são utilizados em toda a aplicação, fornecendo uma representação consistente dos dados em diferentes partes do sistema. Isso facilita a manipulação e a validação dos dados, evitando a duplicação de código e mantendo a consistência dos objetos em toda a aplicação.

Os modelos também podem ser enriquecidos com métodos adicionais que encapsulam lógicas específicas relacionadas aos dados. Esses métodos podem incluir validações, cálculos, formatações ou qualquer outra manipulação específica necessária para o correto funcionamento da aplicação.

Ao utilizar o módulo Models, estamos adotando uma abordagem orientada a objetos para representar os dados da aplicação. Isso traz diversos benefícios, como reutilização de código, modularidade e facilidade na manutenção do sistema. A consistência dos modelos contribui para a integridade dos dados e para a correta interação com outros componentes do sistema, como os módulos de Services e DAO.

Um dos módulos essenciais em muitas aplicações é o módulo relacionado a APIs, que concentra os arquivos responsáveis pela comunicação com serviços externos. Cada arquivo específico dentro desse módulo, como login API ou transactions API, implementa a lógica necessária para interagir com a API correspondente. Essa abordagem permite que diferentes partes da aplicação acessem os serviços externos de forma coesa e reutilizável.

Ao utilizar esse módulo de APIs, a aplicação pode se comunicar com serviços externos por meio de diversos tipos de requisições, como consultas, atualizações, deleções ou criações de dados. Essas requisições são feitas por meio de endpoints, que são os pontos de entrada definidos na API externa. Ao definir esses endpoints e implementar a lógica de comunicação, podemos estabelecer uma integração eficiente entre a nossa aplicação e os serviços externos.

No caso específico do ambiente em que a aplicação está hospedada, foi escolhida a Google Cloud Platform (GCP) e uma máquina virtual (VM) com o Sistema Operacional Linux SMP Debian instalado. Nessa VM, estão configurados o ambiente de desenvolvimento Dart e o banco de dados MySQL. Dessa forma, é possível inicializar o banco de dados e iniciar o servidor em um IP externo. Essa configuração permite abstrair as informações da API externa, fornecendo um ambiente controlado para o desenvolvimento e teste das integrações.

A utilização de um módulo de APIs e a escolha de uma plataforma de hospedagem e infraestrutura adequada são elementos-chave para garantir a comunicação eficiente e segura entre a aplicação e os serviços externos. A modularização da lógica de comunicação por meio dos arquivos específicos de cada API facilita a manutenção, o teste e a reutilização do código.

A divisão em módulos é um aspecto fundamental que simplifica a manutenção do código ao longo do tempo. Quando surge a necessidade de modificar ou adicionar uma funcionalidade, os desenvolvedores podem facilmente localizar os arquivos pertinentes e realizar as alterações necessárias, sem afetar outras partes do sistema. Isso contribui para uma maior estabilidade e extensibilidade da aplicação.

A abordagem modular adotada no planejamento do backend proporciona a oportunidade de reutilizar componentes e funcionalidades em diferentes partes da aplicação. Por exemplo, a criação de um serviço genérico pode ser utilizado por várias funcionalidades do sistema, evitando a duplicação de código e promovendo uma maior consistência e padronização.

Ao planejar o backend de uma aplicação mobile em Dart, é fundamental considerar a escalabilidade e a manutenibilidade do código. A estrutura organizacional dos diretórios e pastas, juntamente com a definição clara das responsabilidades de cada componente, contribui para a modularidade e a coesão do sistema como um todo.

Por meio da estrutura definida no planejamento, os desenvolvedores podem trabalhar de forma mais eficiente e colaborativa, já que cada diretório ou pasta possui um propósito claro e delimitado. Além disso, a modularidade permite que diferentes partes da aplicação sejam desenvolvidas e testadas independentemente, facilitando a identificação e a correção de problemas específicos.

Por fim, a estrutura do backend também facilita a colaboração entre diferentes membros da equipe de desenvolvimento. Cada diretório ou pasta possui um escopo específico, o que permite que os desenvolvedores trabalhem em paralelo em diferentes partes da aplicação sem conflitos ou dependências excessivas.

Em resumo, o planejamento da camada de backend de uma aplicação mobile em Dart deve levar em consideração a modularidade, a escalabilidade e a manutenibilidade do código. A estrutura organizacional dos diretórios e pastas, bem como a definição clara das responsabilidades de cada componente, contribui para a eficiência e a qualidade do desenvolvimento. Ao adotar essa abordagem, os desenvolvedores podem construir uma aplicação mobile robusta, fácil de manter e com potencial para expansão futura.

VII. DIAGRAMAS

DIAGRAMA DE SEQUÊNCIA

O diagrama de sequência ilustra o fluxo de interações entre o ator Usuário, a Interface do projeto, o Servidor (backend) criado e a API implantado na VM (Máquina Virtual). A sequência começa quando o usuário inicia a interação com a interface da aplicação.

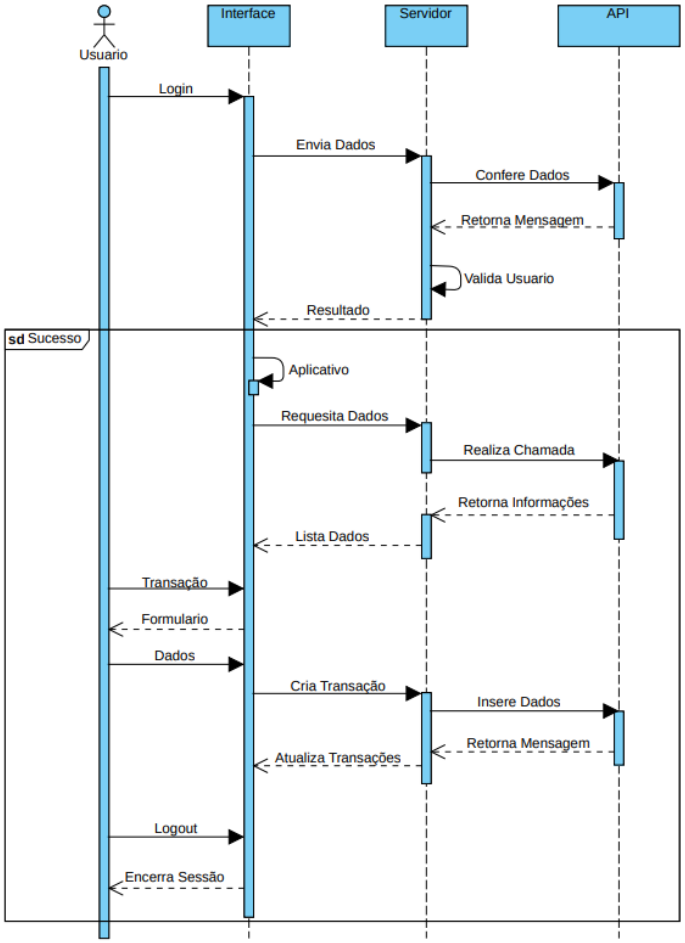
Quando o usuário realiza uma ação, como realizar o login, a Interface captura essa interação e encaminha a solicitação para o Servidor. A Interface atua como um intermediário entre o Usuário e o Servidor, traduzindo a ação em uma chamada compreensível pelo sistema.

O Sistema, por sua vez, recebe a solicitação da Interface e inicia o processo de comunicação com a API. Ele envia uma mensagem à API, solicitando um serviço ou acesso a determinados dados. Essa mensagem é transmitida através de um protocolo de comunicação, como HTTP.

A API recebe a solicitação do sistema e processa-a de acordo com a sua lógica interna. Isso pode envolver consultas a bancos de dados ou autenticação de usuário. Após o processamento, a API retorna uma resposta ao sistema contendo os dados solicitados ou o resultado da operação requisitada.

Ao receber a resposta da API, o Servidor atualiza seu estado interno, processa os dados recebidos e realiza as ações necessárias para atender à solicitação do Usuário. Essas ações podem envolver atualizações no banco de dados, criação de transações ou outras operações da aplicação.

Figura 2 - Diagrama de Sequência



Ferramenta: Visual Paradigm

MÁQUINA DE ESTADO

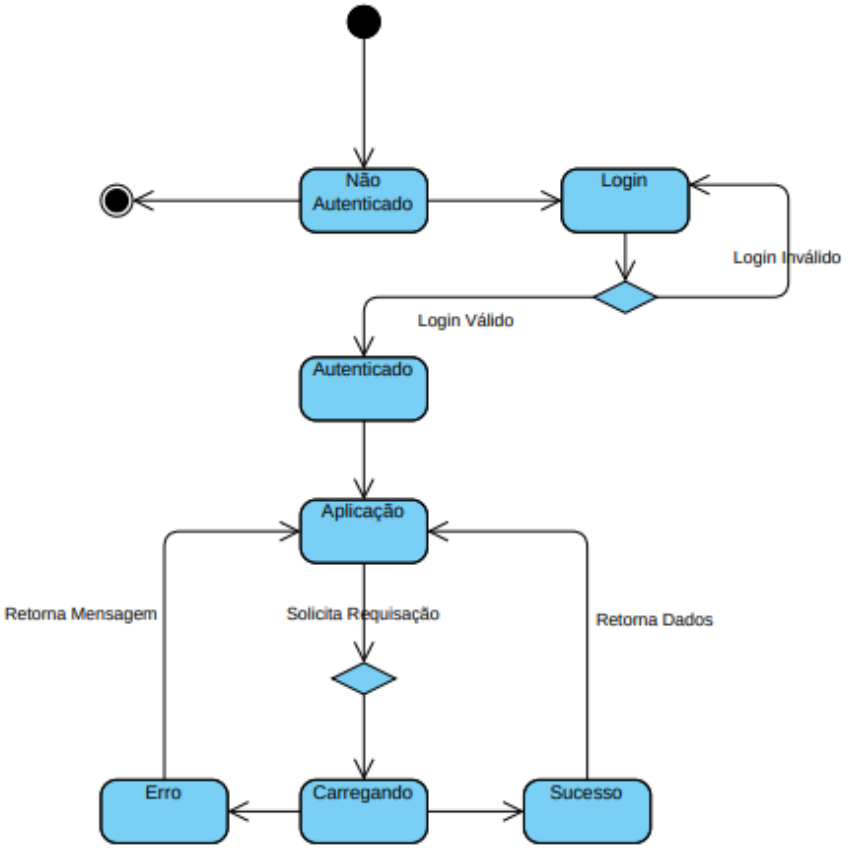
A máquina de estados inicia-se com estado "Não autenticado", onde o sistema aguarda o usuário fornecer suas credenciais de login. Assim que o usuário insere suas informações, a transição ocorre para o estado "Login", onde o sistema verifica a autenticidade das credenciais fornecidas.

Se as credenciais forem validadas com sucesso, a transição ocorre para o estado "Autenticado", indicando que o usuário foi autenticado com êxito. Neste estado, o sistema permite que o usuário acesse a aplicação e interaja com seus recursos. Enquanto o usuário estiver usando a aplicação, a máquina de estados permanece no estado "Aplicação". Aqui, o usuário pode realizar várias ações e navegar pelas funcionalidades disponíveis.

No entanto, em determinadas circunstâncias, pode ocorrer a necessidade de carregar dados ou executar alguma tarefa em segundo plano. Nesse caso, a máquina de estados transita para o estado "Carregando", indicando que o sistema está ocupado realizando as operações solicitadas.

Após o término do carregamento, a transição ocorre para o estado "Sucesso" se todas as operações forem concluídas com êxito. Neste estado, o sistema exibe os resultados ao usuário de forma satisfatória.
Por outro lado, se ocorrer algum erro durante o carregamento ou durante a execução de uma tarefa, a máquina de estados transita para o estado "Erro". Nesse estado, o sistema trata o erro e exibe uma mensagem ou feedback apropriado para informar o usuário sobre a falha.

Figura 3 – Máquina de Estado



Ferramenta: Visual Paradigm

VIII. RESULTADO

O projeto em questão foi desenvolvido com o objetivo de criar um aplicativo financeiro abrangente, fornecendo aos usuários uma solução eficiente para gerenciar suas finanças pessoais. Durante o desenvolvimento, foram implementadas diversas funcionalidades essenciais, com resultados satisfatórios e um potencial promissor para atender às necessidades dos usuários.

Ao início do processo de desenvolvimento, foram definidas metas claras e realistas para criar um aplicativo financeiro funcional e intuitivo. A equipe de desenvolvimento concentrou-se em criar uma base sólida para o projeto, priorizando a segurança dos dados, a autenticação do usuário e a facilidade de uso.
As expectativas em relação ao projeto eram altas, visando criar uma plataforma que permitisse aos usuários acompanhar e controlar suas finanças pessoais de forma eficaz. Os resultados alcançados até o momento são significativos, com as funcionalidades implementadas, como o login, a tela principal, as transações e os relatórios, proporcionando aos usuários uma experiência satisfatória.

No entanto, é importante ressaltar que o projeto está aberto para evolução e espaço de expansão e aprimoramento para atender às expectativas e necessidades do início do projeto citadas na parte de Desenvolvimento. A resolução envolvia a implementação de funcionalidades adicionais, como metas financeiras, configurações personalizáveis, categorias de gastos, integrações com outras plataformas financeiras e recursos de suporte ao cliente.

IX. CONCLUSÃO

O aplicativo de controle de gastos pessoais desenvolvido neste projeto tem um enorme potencial para contribuir significativamente para a educação financeira e a saúde financeira das pessoas, além de promover a inclusão financeira e a equidade social. Com a capacidade de cadastrar todas as despesas diárias, definir metas de economia, gerar gráficos e relatórios detalhados, receber notificações de faturas a vencer e planejar despesas futuras, o aplicativo pode ajudar os usuários a tomar decisões financeiras mais conscientes e a desenvolver hábitos financeiros mais saudáveis.

Por exemplo, ao registrar todas as suas despesas diárias no aplicativo, os usuários podem obter uma visão clara de onde estão gastando seu dinheiro e onde podem fazer cortes para economizar. Eles podem usar as metas de economia para estabelecer objetivos alcançáveis e acompanhar seu progresso ao longo do tempo. Os gráficos e relatórios detalhados fornecidos pelo aplicativo podem ajudá-los a entender melhor seus hábitos de consumo e identificar áreas onde eles podem ser mais conscientes com seus gastos. Além disso, as notificações de faturas a vencer podem ajudá-los a evitar atrasos de pagamento e possíveis cobranças de juros, e o planejamento de despesas futuras pode ajudá-los a se preparar para despesas maiores, como impostos ou férias.

Com todos esses recursos disponíveis, o aplicativo pode se tornar uma ferramenta valiosa para aqueles que buscam se tornar mais financeiramente conscientes e saudáveis. A inclusão financeira e a estabilidade social também são promovidas, pois o aplicativo é acessível para qualquer pessoa com um smartphone, independentemente de sua renda ou status financeiro. Em resumo, este aplicativo tem o potencial de ser um grande diferencial na vida financeira das pessoas, fornecendo recursos, informações e incentivos para que elas tomem decisões mais inteligentes e saudáveis com seu dinheiro.

REFERÊNCIAS

LUSARDI, Annamaria; MITCHELL, Olivia. The Economic Importance of Financial Literacy: Theory and Evidence. Washington, DC: Global Financial Literacy Excellence Center (GFLEC), George Washington University School of Business, 2015. Disponível em: **https://gflec.org/wp-content/uploads/2015/09/Annamaria-Lusardi-Paper.pdf**.

LUSARDI, Annamaria; MITCHELL, Olivia. Financial literacy around the world: an overview. Washington, DC: Global Financial Literacy Excellence Center (GFLEC), George Washington University School of Business, 2015. Disponível em: https://gflec.org/wp-content/uploads/2014/12/economic-importance-financial-literacy-theory-evidence.pdf .

VALENTE, Marco. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade. 1ªed. Minas Gerais: 2020

O'REILLY, John. DAO Object Model - The Definitive Reference. 1ª ed. Sebastopol: O'Reilly, 2000.

SEEMANN, Mark. Dependency Injection in .NET. 2ª edição. Shelter Island, NY: Manning Publications, 2019.

GEEWAX, JJ. API Design Patterns: Creating Web APIs That Developers Love. Sebastopol, CA: O'Reilly Media, 2020.

IETF. JSON Web Token (JWT). Disponível em: https://datatracker.ietf.org/doc/html/rfc7519.

KHONONOV, Vlad. Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy. Sebastopol, CA:O'Reilly Media, 2021

GOÉS, Wilson. Aprenda UML por Meio de Estudos de Caso.1ªedição, Novatec Editora, 2014