Revista Ibero- Americana de Humanidades, Ciências e Educação- REASE

doi.org/10.51891/rease.v8i11.7771

O USO DOS FRAMEWORK FLUTTER E NEST.JS PARA DESENVOLVIMENTO DE UM FRONT-END E UM BACK-END DE UMA APLICAÇÃO DE HELP DESK

THE USE OF FRAMEWORK FLUTTER AND NEST.JS TO DEVELOP A FRONT-END AND A BACK-END OF A HELP DESK APPLICATION

Ana Karolina Santos Oliveira[1]
Edilson Carlos Silva Lima[2]
Elda Regina de Sena Caridade[3]

RESUMO: A comunicação interna das empresas é essencial para que elas apresentem um bom desenvolvimento, por isso, é necessário ficar atento aos problemas de comunicação nas empresas e sempre buscar as melhores formas de resolvê-las. Antes de buscar por soluções, entretanto, em grandes empresas onde há uma grande demanda dos mais variados tipos de serviços, é necessário categorizar e especificar os serviços e áreas da empresa para que haja uma boa comunicação, desenvolvimento e produtividade na empresa. Para isso foi desenvolvido um aplicativo com os frameworks Flutter para desenvolvimento de um front-end e Nest.Js para desenvolvimento de um back-end, buscando uma melhor comunicação e execução dos serviços internos dentro de uma empresa.

Palavras-chave: Framework. Flutter. Nest.Js. Aplicativo.

ABSTRACT: The internal communication of companies is essential for them to present a good development, so it is necessary to be aware of communication problems in companies and always seek the best ways to solve them. Before looking for solutions, however, in large companies where there is a great demand for the most varied types of services, it is necessary to categorize and specify the services and areas of the company so that there is good communication, development and productivity in the company. For this, an application was developed with the Flutter frameworks for the development of a front-end and Nest.Js for the development of a back-end, seeking better communication and execution of internal services within a company.

Keywords: Framework. Flutter. Nest.Js. Application.

INTRODUÇÃO

Manter uma boa comunicação interna traz diversos benefícios para a empresa e para os colaboradores que nela trabalham. Benefícios como aumentar o desempenho da equipe e

[1]Engenharia de Computação – UNICEUMA – São Luís – MA – Brasil
[2]Engenharia de Computação – UNICEUMA – São Luís – MA – Brasil
[3]Engenharia de Computação – UNICEUMA – São Luís – MA – Brasil

no final aumentar a satisfação do público. A comunicação interna, caso ocorra algum problema dentro da empresa, costumeiramente ocorre de forma informal e sem algum meio de comunicação fixo e em que todos da empresa tenham acesso. A forma mais comum de solicitar por suporte dentro da própria empresa, é sair do seu setor e ir procurar o setor de suporte, o que gera uma perda de tempo e desempenho dos funcionários no seu horário de trabalho.

Visto isso, algumas ferramentas são utilizadas para centralizar essa comunicação entre as partes, tais como sistemas web e aplicativos mobile. Dessa forma o presente trabalho propõe um aplicativo que será utilizado no *help desk*, que se trata de um setor, ou o conjunto de setores, técnicas, estratégias e serviços que apoiam usuários e clientes para resolver problemas. O Aplicativo foi desenvolvido utilizando tecnologias atuais e eficazes, tais como o Flutter para desenvolver o frontend, o Nest.Js para desenvolver o backend da aplicação e o PostgreSql para ser o banco de dados utilizado nesta aplicação.

Foi utilizado uma abordagem de pesquisa descritiva para analisar como a comunicação é feita dentro da universidade e os benefícios que podem ser obtidos com o uso do sistema proposto. Para isso, o estudo se baseia no cotidiano dos alunos e profissionais da instituição, e requer pesquisa bibliográfica e entrevistas com esses alunos e profissionais.

Neste trabalho vamos abordar tópicos importantes relacionados ao desenvolvimento deste aplicativo, sendo eles: a fundamentação teórica no capítulo 1 deste artigo, logo após no capítulo 2, apresenta a fundamentação teórica e as tecnologias utilizadas no desenvolvimento do aplicativo; seguido pelo capítulo 3, que irá apresentar a modelagem do aplicativo, em sequência, o capítulo 4, em que serão apresentados os resultados obtidos através do público que testou o aplicativo, e por fim no capítulo 5, será apresentado a conclusão deste trabalho.

1. Fundamentação Teórica e Tecnologias Utilizadas

Neste capítulo serão abordados os conceitos relacionados as tecnologias utilizadas no desenvolvimento do aplicativo e os passos para o desenvolvimento dele. No item 2.1 serão apresentadas considerações sobre o Nest.Js, framework escolhido para o desenvolvimento do *backend* da aplicação e por fim, no item 2.2 serão apresentadas considerações sobre o Flutter, framework escolhido para o desenvolvimento do aplicativo.

1.1 UML

Todos as aplicações a ser desenvolvidas precisam do projeto e escopos bem definidos para que sejam desenvolvidos da melhor forma. Uma ferramenta que auxilia isso, é a UML, sigla para *Unified Modeling Language* ou Linguagem de Modelagem Unificada, que dentre suas definições, uma se destaca:

> Uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação." GUEDES (2011).

A UML é composta por outros diagramas, em que cada um é responsável por representar uma parte da aplicação. Neste projeto foram utilizados o diagrama de classes e o diagrama de caso de uso, que será apresentado no item abaixo.

1.2 Diagrama de caso de uso

Para Guedes (2011), o objetivo do diagrama de caso de uso é demostrar de forma gráfica as maneiras que o usuário pode interagir com a aplicação. Ele resume os detalhes dos usuários no sistema e as interações com ele.

1.3 Diagrama de classe

Para Booch (2006), o diagrama de classes é um dos modelos mais importantes no processo de engenharia de software e é a base para outros diagramas. Isso porque além de definir os relacionamentos entre eles (como herança, composição, associação e dependências), também é usado para mapear sistemas modelando seus métodos, propriedades e classes. Seu objetivo é expressar visualmente as necessidades especificas de um sistema.

1.4 NEST.JS

Como sua própria documentação informa:

> **Nest.js** *é um framework node.js progressivo feito para criar aplicações de backend eficientes, escaláveis e confiáveis". NEST.JS (2017)*

Ele utiliza por padrão o TypeScript como linguagem e possui uma arquitetura similar ao framework Angular. Com ele é possível trabalhar com diferentes tipos de bancos de dados, tais como PostgreSql – um banco de dados relacional, até o MongoDB, um banco de dados não relacional.

A arquitetura do nestJs é baseada em módulos, em que cada modulo é composto por *controller* e *servises*. (NESTJS, 2017). Logo abaixo, serão apresentados esses conceitos.

- **Controllers:** são responsáveis por lidar com as solicitações recebidas via http request e retornar as respostas ao cliente.
- **Services:** são responsáveis por fornece os serviços para a aplicação, tais como, deletar um usuário ou listar todos os usuários de um banco de dados.

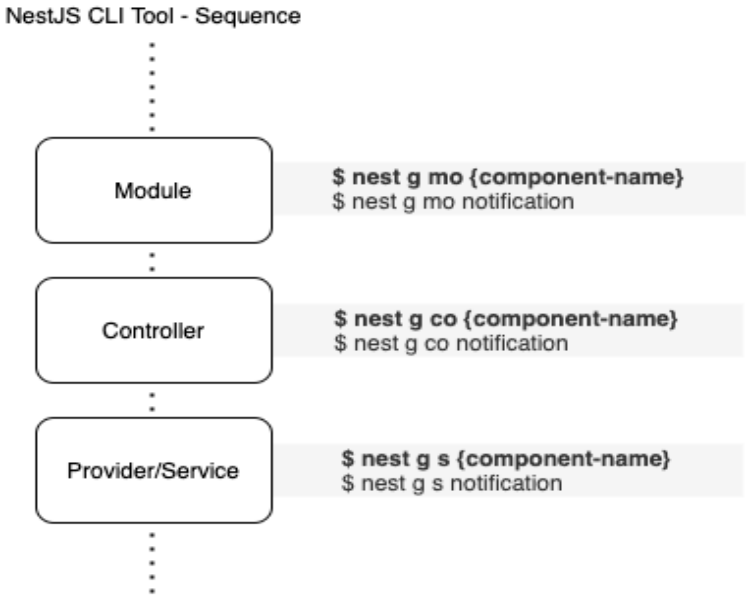


Figura 1. Arquitetura do Nest.js
Fonte: Rasim Sen, 2021.

Como visto na figura 1, o NestJs é dividido em módulos, e vale ressalvar que todas as aplicações desenvolvidas com nest.js possuem pelo menos um módulo, o módulo raiz, ele é o ponto de partida que o nest.js usa para construir o aplicativo (SEN, RASIM, 2021).

1.5 FLUTTER

Flutter é um SDK de código aberto criado pelo Google para o desenvolvimento de aplicativos multiplataforma (podem ser executados em mais de um sistema operacional). Possui como linguagem base o Dart, e é utilizado para criação de aplicações móveis, desktop e web desenvolvido pela Google. De acordo com o Google, o Flutter possibilita um desenvolvimento rápido, provê a criação de interfaces amigáveis e acessíveis, além de fornecer um código nativo para as diferentes plataformas (ALESSANDRA, 2020).

Com o Flutter é possível desenvolver aplicações para Android e IOS a partir de uma mesma base de código, na figura 2 é possível notar o mesmo aplicativo nas diferentes

plataformas, em que o Flutter abstrai as diferenças entre os componentes de cada plataforma e entrega os ícones, tipográficas e comportamentos da interface de maneira semelhante nos dois ambientes.

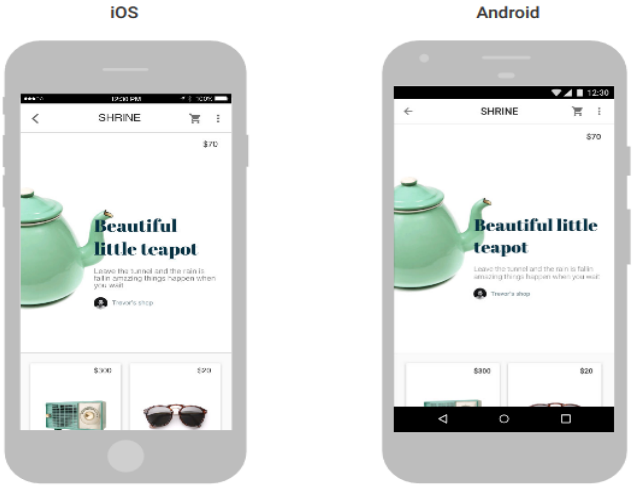


Figura 2. Aplicativo em Flutter na plataforma Android e IOS
Fonte: Luiz Marcos, 2019

Conforme a figura 3, o fluxo de desenvolvimento do flutter é baseado em Widgets, que são componentes visuais (botões, textos, imagens...) para definir a interface de um aplicativo. Os widgets possuem um design específico para a plataforma Android (Material Componentes) e uma para o IOS (Cupertino). Eles (widgets) formam uma arvore hierárquica na aplicação, em que cada um herda as propriedades do superior (ZAMMETTI, 2020).

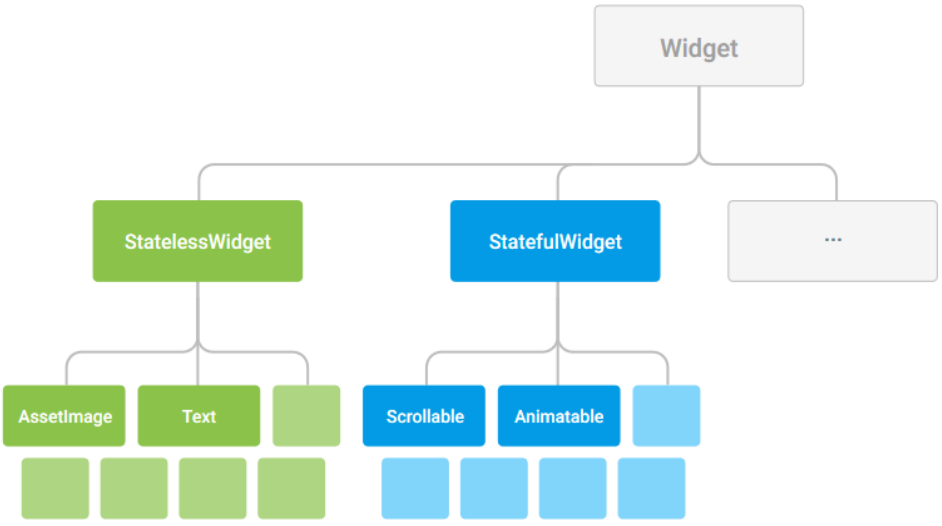


Figura 3. Hierarquia de widgets no flutter
Fonte: Flutter, 2015

A Figura 4, detalha a organização estrutural do Flutter, em que ele é composto pela *engine*, uma camada desenvolvida em C/C++, e a camada do framework implementada em Dart, é nessa camada que ocorre a implementação da aplicação, ou seja, o desenvolvimento do layout da aplicação (ZAMMETTI, 2020).

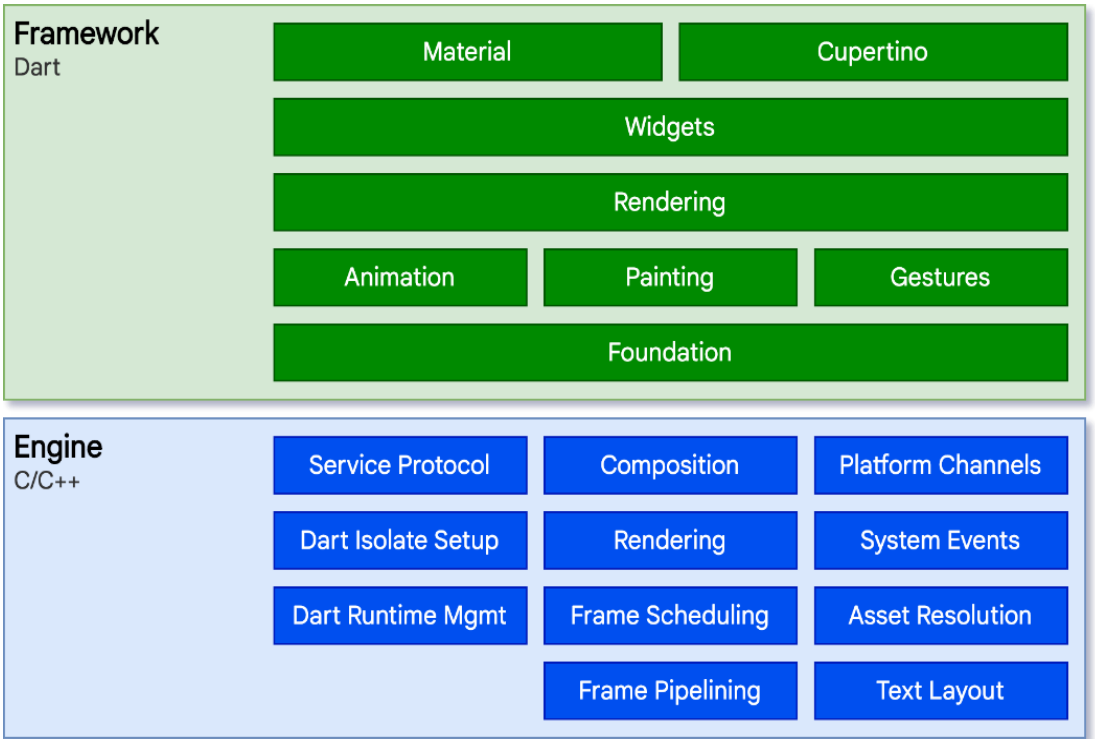


Figura 4. Arquitetura do Flutter
Fonte: Flutter, 2015

1. Benefícios de utilizar o Flutter para desenvolver aplicativos (ARAÚJO, 2021)

- **Despensa conhecimentos prévios**

  A estrutura Flutter possui recursos gráficos para criar facilmente interfaces personalizadas e complexas com segurança de aplicativos. Quase tudo o que aparece na tela pode ser personalizado, por mais complexo que seja. Para isso, utiliza o motor gráfico Skia, que possui excelente desempenho e utiliza código-fonte aberto.

  A vantagem aqui é que as atualizações de software não interferem nos layouts e designs personalizados em nenhum sistema operacional. Isso porque, diferentemente do que acontece em outros frameworks disponíveis no mercado, há pouca ou nenhuma chance de falha.

- **Reduz custos**

  Por dispensar o uso de diferentes plataformas e ferramentas de criação, o custo de desenvolvimento de um aplicativo é significativamente menor. O fato de o produto gerar empregos em todos os sistemas operacionais levou a uma redução de gastos, pois há menos necessidade de ferramentas de desenvolvimento.

- **Aumenta a produtividade**

  Os custos também são reduzidos devido à maior produtividade da equipe, que pode entregar mais em menos tempo. Por ser uma estrutura cruzada, os desenvolvedores usam a mesma base de código para ambos os sistemas e não precisam fazer versões diferentes do aplicativo.

  A compilação de código nativo do Dart e a falta de widgets OEM facilitam a comunicação entre o aplicativo e a plataforma. O fato de as visualizações responsivas serem fornecidas sem a necessidade de Javascript também beneficia os tempos de inicialização, resultando em acesso mais rápido ao sistema.

2. Estudo de Caso

Neste capítulo será apresentada a modelagem do aplicativo em estudo. No item 3.1 será apresentado uma breve introdução ao aplicativo que foi proposto, em seguida, no item 3.2 serão apresentados a modelagem do aplicativo, apresentando os requisitos funcionais e não funcionais do aplicativo, em seguida no item 3.3 serão apresentados o processo de desenvolvimento do *backend*, em seguida, no item 3.4 será apresentado o processo de desenvolvimento do *frontend*, e por fim no item 3.5 será apresentado o fluxo de utilização do aplicativo.

2.1 Aplicativo Proposto

O objetivo de aplicativo é manter uma comunicação mais centralizada entre a aera de suporte de uma faculdade e os demais setores. Em que o usuário ao encontrar um problema ou necessitar de suporte, ele pode acessar o aplicativo de solicitar os serviços, descrevendo o que precisa e em qual setor se encontra. Foi utilizado o método de pesquisa descritiva com finalidade de analisar a forma de comunicação dentro e para quem o aplicativo seria útil dentro da universidade, com isso foi possível chegar a um modelo de aplicativo desenvolvido para o *helpdesk* de uma universidade e pode ser usado pelos alunos e funcionários. Logo após, no item 3.3 serão apresentadas considerações sobre o desenvolvimento do *backend* da aplicação, seguindo no tópico 3.4 das considerações sobre o desenvolvimento do *frontend* da aplicação. E por fim, item 3.5 será apresentado o fluxo de utilização do aplicativo, sendo exemplificado por meio de ilustrações.

2.2 Modelagem do Sistema

Nesta seção será apresentada a modelagem de domínio do aplicativo, tendo em vista que o domínio é o conjunto de regras de negócio que a aplicação deve seguir, que são obtidas a partir do levantamento dos requisitos funcionais e não-funcionais. No item 3.2.1 será

apresentado as entidades, ou seja, as tabelas no banco de dados deste aplicativo, logo em seguida no item 3.2.2 serão apresentados os requisitos funcionais do aplicativo e por fim, no item 3.2.3 serão apresentados os requisitos não funcionais do aplicativo.

2.2.1 *Diagrama de caso de uso*

Neste item será apresentado o diagrama de casos de uso, que foi anteriormente explicado. O diagrama da figura 5, apresenta 2 tipos de usuários (Usuário comum e Usuário administrador) e suas respectivas funcionalidades dentro do aplicativo. O ator usuário representa as funcionalidades disponíveis para ambos os usuários. No diagrama são exibidas as funcionalidades do usuário administrador, dentre elas, ver as solicitações de todos os usuários do aplicativo, criar e listar setores e usuários e criar soluções, tais funcionalidades não estão disponíveis para usuários comuns, como demonstrado no diagrama.

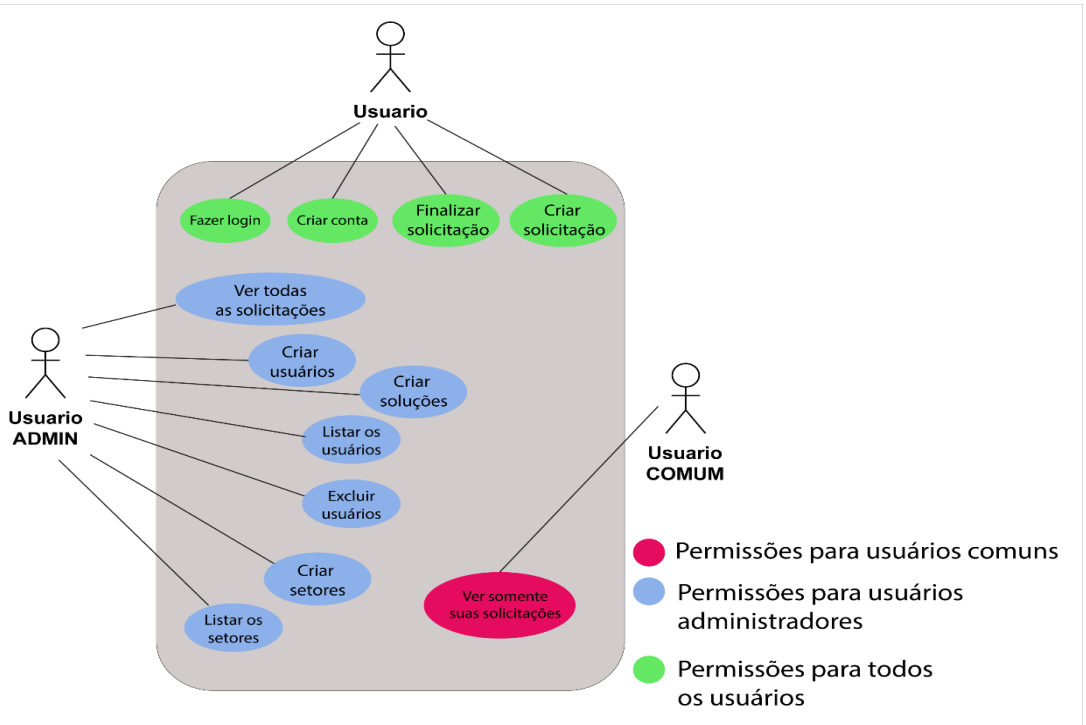


Figura 5. Diagrama de caso de uso
Fonte: Autoral, 2022.

O diagrama de casos de uso é de suma importância na modelagem de um sistema, pois ele irá especificar todas as ações do usuário dentro do sistema, e facilitar a criação da documentação do sistema (DEVMEDIA, 2012).

2.2.2 *Diagrama de classes*

O diagrama de classes é uma representação da estrutura da estrutura e relações das classes que servem de modelo para objetos (DEVMEDIA, 2016).

O diagrama da figura 6 irá representar as entidades contidas no banco de dados. Tendo isso em vista, como será apresentado na figura 6, o banco possui 4 tabelas, sendo elas usuário, solicitação, solução e setor, além dessas entidades, o modelo dessa aplicação possui dois tipos enumerados, sendo eles status, que representa o status da solicitação e o tipo de

usuário que irá representar o perfil do usuário, para que ele possa ter suas devidas permissões.

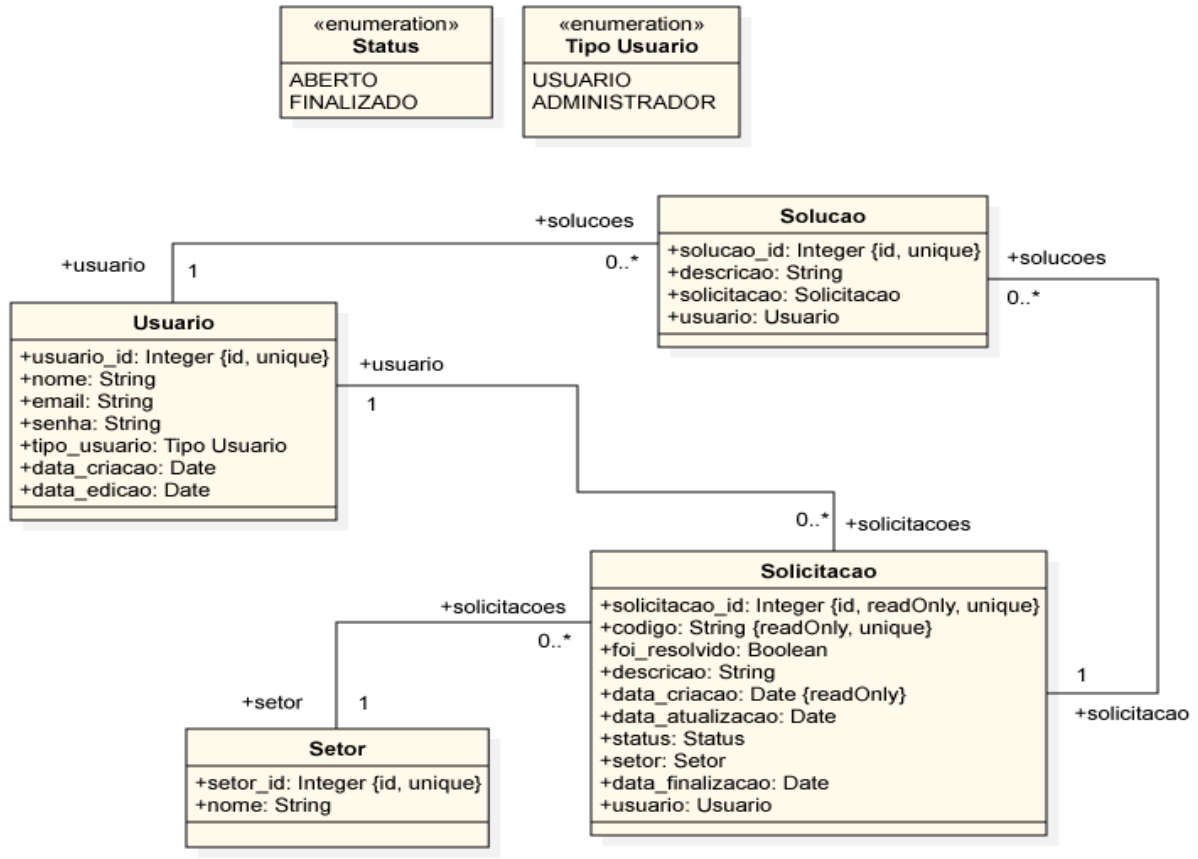


Figura 6. Diagrama de classes

Fonte: Autoral, 2022.

O diagrama de classes é útil e eficiente em várias etapas da modelagem do sistema. Dentre suas funções, ele serve para definir os relacionamentos entre as classes, ilustrar a estrutura do modelo utilizando atributos, operações e sinais, além de mostrar a hierarquia entre as classes (IBM, 2021).

2.3 Desenvolvimento do *Backend*

O servidor desse trabalho foi desenvolvido no formato API Rest, ou seja, seu objetivo é fornecer os dados necessário para o cliente da aplicação. A api rest foi desenvolvida utilizando o TypeScript, com auxílio do framework Nestjs, usando postgreSql para persistência dos dados.

2.3.1 Bibliotecas

A seguir estão listadas algumas bibliotecas utilizadas na implementação de recursos importantes para o funcionamento do sistema:

- **CryptoJs** – esta biblioteca para Javascript fornece a aplicação a criptografia de senha dos usuários;
- **TypeOrm** – é um orm (Object-Relational Mapping), ou seja, uma biblioteca que usa técnicas para aproximar o paradigma de desenvolvimento de aplicações orientadas a objetivos ao paradigma de um banco de dados relacional;
- **JsonWebToken** – uma biblioteca que utilizada para verificação de token de autenticação no sistema.

2.4 Desenvolvimento do Aplicativo Mobile

O aplicativo mobile, ou seja, o jogo em si, foi desenvolvido utilizando o framework Flutter, que utiliza como linguagem o Dart, que pode ser utilizado em plataformas Android. O objetivo do aplicativo é fornecer todas as funcionalidades para o usuário.

2.4.1 Bibliotecas

O desenvolvimento do aplicativo também fez uso de algumas bibliotecas:

- **HTTP**: biblioteca responsável por realizar requisições http para o servidor;
- **Shared Preferences Plugin:** biblioteca que permite salvar dados, com chave e valor, localmente na sessão do usuário.

2.5 Fluxo de Utilização do Aplicativo

Nesta seção será apresentada a interface do aplicativo, com a apresentação das telas, seguidas de uma breve explicação sobre o seu funcionamento. Para apresentar as telas, o sistema operacional escolhido foi o Android, tendo em vista que a distribuição do aplicativo é feita apenas por Android. Vale lembrar que o aplicativo conta com dois layouts tendo em vista que um é para os usuários comuns e o outro é para os usuários que são administradores. Primeiramente será apresentado o layout dos usuários comuns.

2.5.1 Layout para Usuários Comuns

No item 3.5.1 serão apresentadas as ilustrações referentes ao layout dos usuários comuns e no item 3.5.2 serão apresentadas as devidas ilustrações do layout dos administradores.

2.5.1.1 Tela inicial

Ao acessar o aplicativo, a tela de Figura 5 é apresentada ao usuário, oferecendo ao usuário a opção de realizar o login, ou caso não tenha uma conta, ele tem a opção de criar uma conta. Para criar uma conta o usuário deve informar seu nome, email e uma senha.

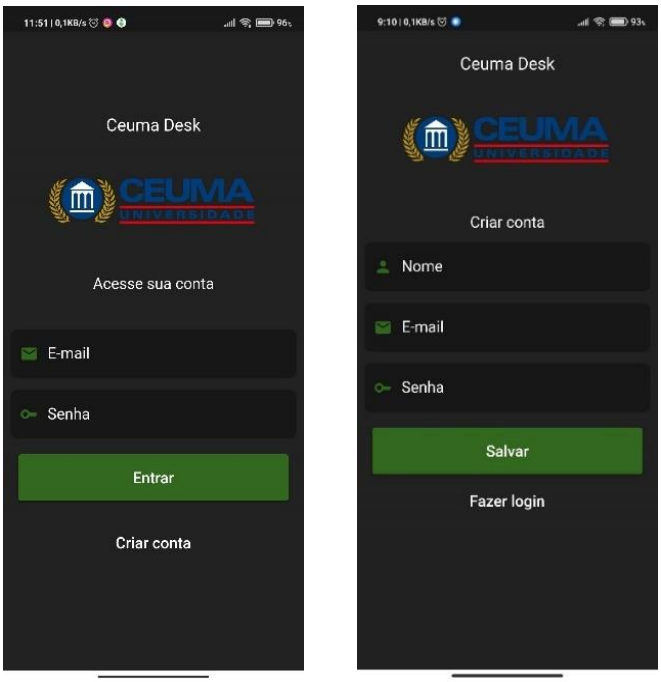


Figura 7. Tela de login e tela de criação de conta
Fonte: Autoral, 2022.

Para realizar o login o usuário deve informar apenas o email e a senha e para criar uma conta, ele deve informar seu nome, email e uma senha.

2.5.1.2 Tela de listagem de solicitações

Após realizar o login, o usuário é levado para tela principal da aplicação, onde há a listagem das solicitações que se encontram em andamento, podendo filtrar pelas solicitações que já foram finalizados. Ainda nesta tela o usuário tem a opção de abrir uma nova solicitação.

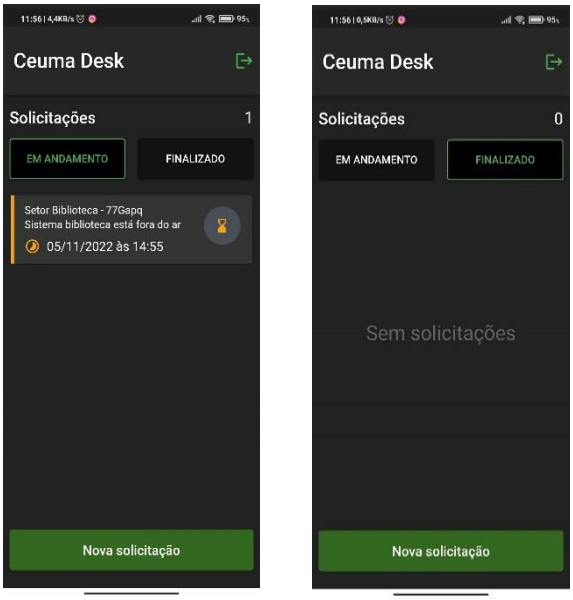


Figura 9. Tela de listagem de solicitações com status ABERTO e FINALIZADAS
Fonte: Autoral, 2022.

Conforme visto na figura 9, o usuário tem a opção de filtrar as solicitações com status em ABERTO e as solicitações com status FINALIZADO, deixando separado a listagem de cada status.

2.5.1.3 Tela para criar uma solicitação

Nesta tela o usuário deve criar uma solicitação, informando o setor onde que precisa de suporte e informar uma descrição sobre, após informar esses dados ele deve pressionar o botão 'Salvar' para que a solicitação seja salva.

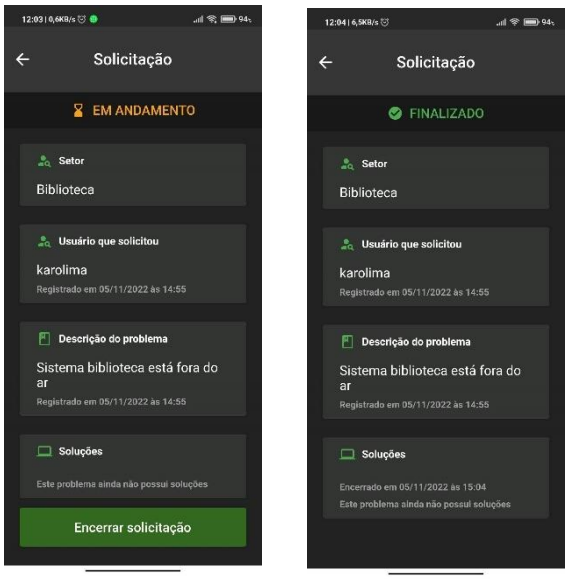


Figura 10. Tela para criar uma solicitação
Fonte: Autoral, 2022.

Após informar os dados necessários para criar a solicitação, o usuário será direcionado para a tela de listagem de solicitação (Figura 9), vale ressaltar que o usuário deve informar uma descrição rica em detalhes para que as soluções do suporte sejam mais assertivas.

2.5.1.4 Tela de detalhes de uma solicitação

Nesta tela o usuário tem todas as informações da solicitação criada, tais como status, setor, usuário que fez a solicitação, a descrição e as possíveis soluções para esta solicitação. A solicitação possui dois status, e para isso dois layouts são apresentados de acordo com o status.

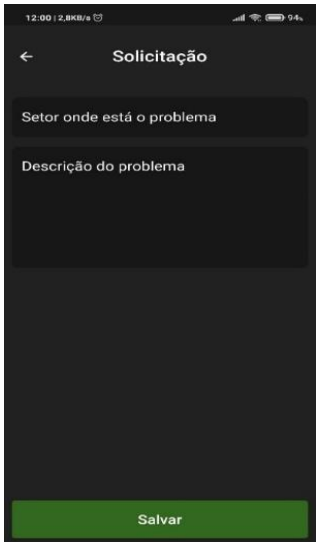


Figura 11. Tela de detalhes de uma solicitação com status em ABERTO e FINALIZADO
Fonte: Autoral, 2022.

Na tela de detalhes da solicitação, o layout deixa claro, logo no início da tela em qual status aquela solicitação está, e outras informações relacionados a solicitação também são apresentadas ao usuário, tais como, o setor, usuário que fez a solicitação, a descrição e as possíveis soluções.

2.5.2 Layout para usuários administradores

2.5.2.1 Tela de listagem de solicitações

Para ter acesso as funcionalidades que o administrador usufruir, logo da tela inicial é apresentado um botão que irá levar ele ao painel do administrador.

Figura 12. Tela de listagem de solicitações para o usuário administrador
Fonte: Autoral, 2022.

Conforme a figura 12, apenas os usuários do tipo administrador possuem a permissão para responder uma solicitação do usuário, em outra palavras, ele possui permissão para criar soluções.

2.5.2.2 Tela para criar uma solução para uma solicitação

Nesta tela o administrador pode criar uma possível solução para a solicitação, ele deve apenas escrever uma descrição sobre sua solução e após isso apertar em confirmar, e assim a solução será criada.

Figura 13. Tela para criar uma solução para uma solicitação
Fonte: Autoral, 2022.

Conforme visto na figura 13, para ir para o formulário de criação de uma solução, o administrador na tela inicial tem a opção Admin, que se trata de um botão que ao ser clicado, o usuário será direcionado para o painel do administrador.

2.5.2.3 Tela para listar os setores e usuários

Nesta tela o administrador tem acesso aos setores e usuários disponíveis na aplicação. Na mesma tela de listagem de setores o administrador tem a opção para criar um setor.

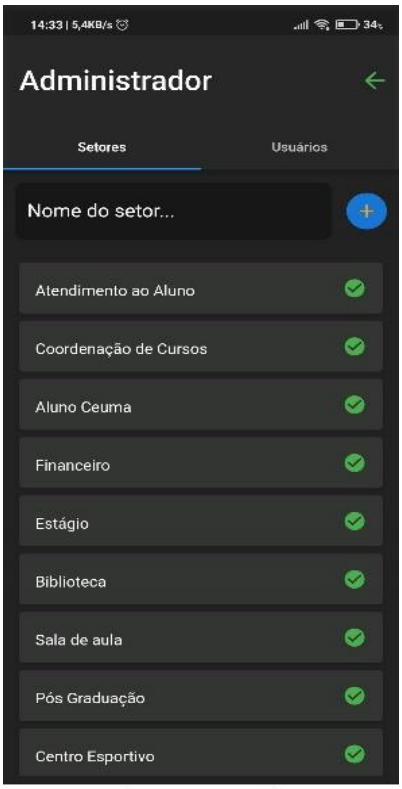


Figura 14. Tela de listagem de usuários e setores.
Fonte: Autoral, 2022.

A tela de listagem de usuário e a tela de listagem de setores são semelhantes. Na tela de listagem de usuários, o administrador tem a opção de criar um usuário, e na tela de listagem de setores, o administrador tem a opção de criar um setor.

2.5.2.4 Tela para criar um usuário

Nesta tela o administrador pode criar um usuário, ao fazer isso, este novo usuário também será um administrador.

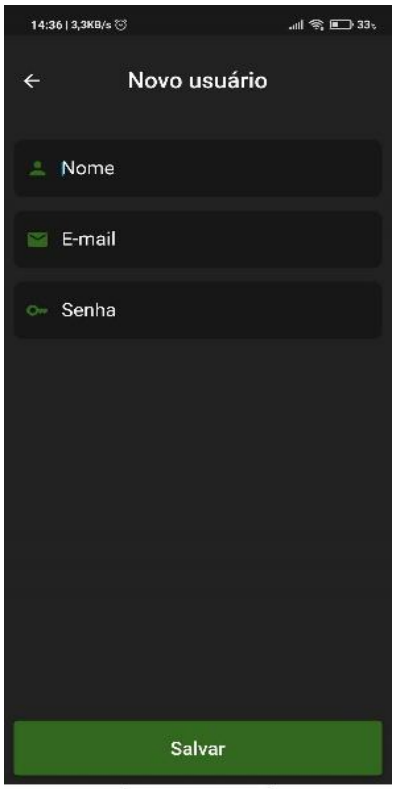


Figura 14. Tela para criar usuário.
Fonte: Autoral, 2022.

Para criar um usuário, o administrador deve informar três dados do usuário, são eles o nome, e-mail e uma senha de acesso para este usuário, depois de informar esses dados, ele deve ir à opção de salvar.

3. Resultados e Discussões

Para validar a aplicação e fazer suas devidas considerações, foram recolhidas fichas de avaliação e realizado o levantamento das respostas. O questionário de perguntas, foi constituído de 5 perguntas subjetivas, em que o público pode descrever suas experiencias com o aplicativo. Os resultados obtidos a partir de 15 entrevista dentro da universidade, mostram que os usuários aprovaram o aplicativo e que ele pode apresentar algumas melhorias. Abaixo serão apresentados os resultados para cada pergunta do questionário. Como instrumentos de coleta de dados, foi utlizado um questionário elaborado com as perguntas, disponibilizado online para os entrevistados. Nesse passo, foi utlizizado uma planilha para fazer a análide dos dados anteriormente colentados. Dessa forma foi possível obter os resultados das avaliações.

a. Dentro da faculdade, caso você precisasse de suporte técnico em algum problema, como você iria solicitar este suporte?

Figura 15. Gráfico de rosca para pergunta 1 do questionário de avaliação.
Fonte: Autoral, 2022.

Nos resultados foram observados 4 tipos de respostas, dentre essas respostas, 87% dos indivíduos dentro da universidade não sabem o local certo para solicitar o suporte técnico.

a.Você considera importante ter um canal de comunicação entre a área técnica (suporte) e os demais setores da faculdade? Descreva sua opinião

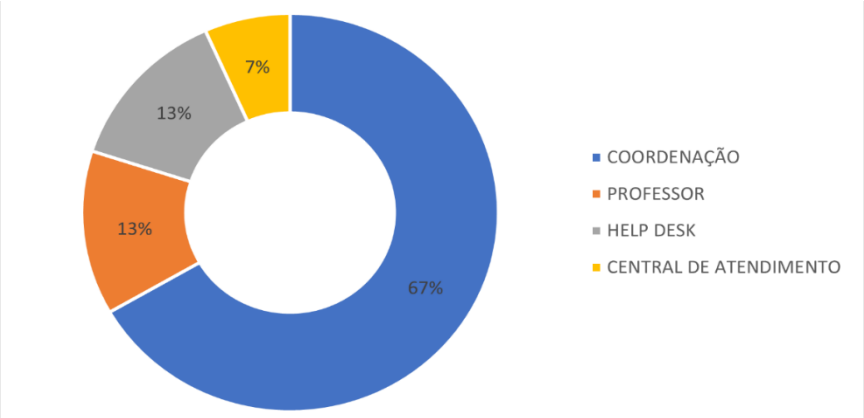
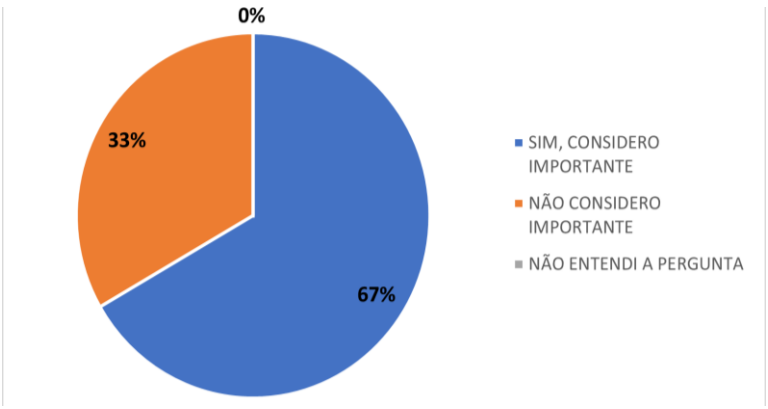


Figura 16. Gráfico de rosca para pergunta 2 do questionário de avaliação.
Fonte: Autoral, 2022.

Nos resultados foram observados que 67% dos entrevistados consideram importante ter um canal de comunicação entre o help desk da universidade e os demais setores da mesma.

a.Em relação a tela inicial deste aplicativo, você a considera intuitiva e fácil de utilizar?



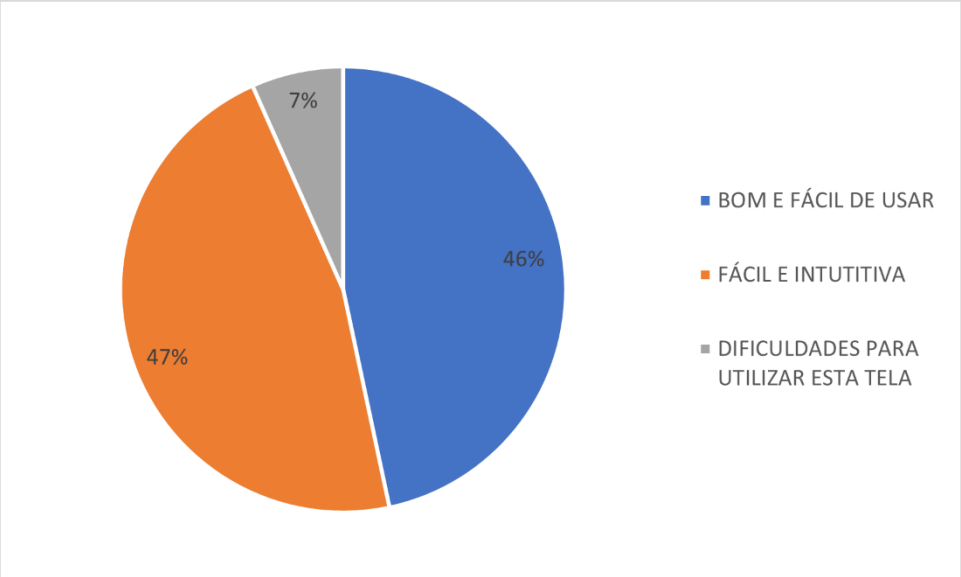


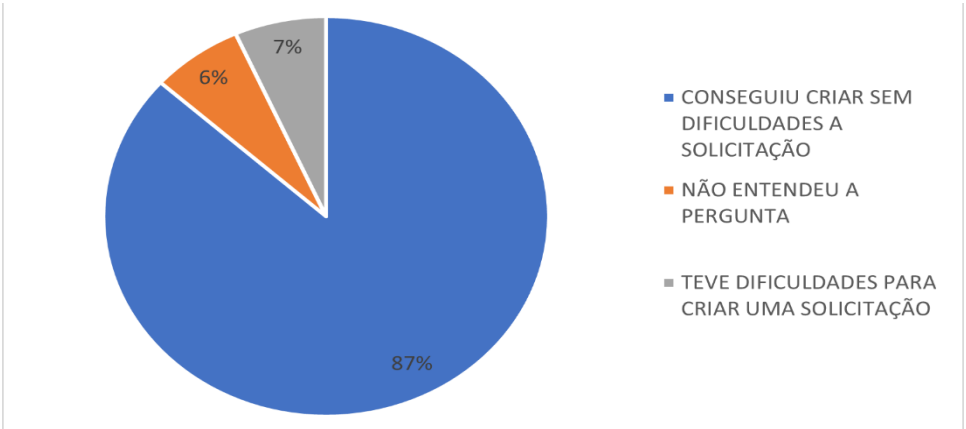
Figura 17. Gráfico de rosca para pergunta 3 do questionário de avaliação.
Fonte: Autoral, 2022.

A pergunta acima foi referente a tela de listagem de solicitações, e pode-se observar que 48% dos entrevistados à consideram fácil e intuitiva para ser usar.

2. Uma vez nesta tela, você conseguiria criar uma solicitação para o suporte da faculdade? Se não, o que você acha que pode ser acrescentado, visando uma melhoria nesta tela?

Figura 18. Gráfico de rosca para pergunta 4 do questionário de avaliação.
Fonte: Autoral, 2022.

De acordo com as repostas acima, pode-se nota que o 87% dos entrevistados conseguiram criar sem dificuldades uma solicitação, concluindo que o aplicativo tem uma interface amigável e intuitiva. Cerca de 7% do público teve dificuldades para criar a solicitação, pois não entenderam a proposta do setor.



3.Quais suas sugestões de melhorias para o aplicativo Ceuma Desk?

A última pergunta do questionário foi em relação a melhorias do aplicativo. Dentre as respostas obtidas, algumas melhorias foram apontadas, tais como:

- Inserir uma foto do problema;
- Ao criar uma solicitação ter a opção de selecionar as solicitações que ocorrem com mais frequência;
- Classificar as solicitações por prioridade.

Tais melhorias serão pensadas e elaboradas para uma próxima versão do aplicativo.

CONCLUSÃO

A elaboração deste trabalho resultou em um aplicativo móvel para ser utilizado no help desk de uma universidade, com o objetivo de centralizar a comunicação e a resolução de problemas do help desk. Fazendo com que a comunicação interna dentro da universidade seja centralizada e mais eficiente, poupando os alunos e funcionários de se deslocarem em busca do suporte, em que na maioria dos casos como foi observado, não se sabe a quem procurar quando um problema técnico ocorre.

Os resultados obtidos foram bastante positivos, como pode observar, os usuários não tiveram dificuldades em utilizar o aplicativo, e melhoria devem ser feitas, visando uma melhor usabilidade, como proposta de melhoria, que é extremamente útil para o objetivo do aplicativo, é incluir uma foto do problema ao criar uma solicitação.

O projeto nasceu de ideias e reuniões de especialistas do beneficiário. O processo de criação do projeto tem várias fases, como pré-reunião, reunião e pós-reunião. Para automatizar o processo de projeto, realizamos modelagem (análise) e desenvolvemos um aplicativo mobile de help desk. Além da modelagem completa do sistema, um primeiro protótipo foi desenvolvido para observar todos os detalhes necessários. Durante o desenvolvimento do aplicativo, encontramos a necessidade do módulo de autorização. Isso ocorre porque, embora você possa delegar a capacidade de aprovar solicitações de agendamento a outros usuários, o acesso deles é restrito. Esse problema foi resolvido criando um nível de autenticação na tabela de usuários do banco de dados, utilizando o campo tipo de usuário para conceder suas devidas permissões no aplicativo.

Foram utilizadas e exploradas tecnologias modernas que estão bem aquecidas no mercado e que são recomendadas por experientes da area, como o Flutter que permite o desenvolvimento de aplicativos nativos de alta performance, gerando assim uma boa experiencia aos usuários.

REFERÊNCIAS

ALESSANDRIA, Simone. **Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games.** Edição em Inglês. Editora: Packt Publishing. 2020

ALURA, Flutter: o que é e tudo sobre o framework. Documento eletrônico. Disponível em <https://www.alura.com.br/artigos/flutter>. Publicado em 11/2022 Acesso em 01 set. 2022.

ANDROID. Disponível em < https://android.com >. Acesso em 10 set. 2022.

FLUTTER. Flutter architectural overview. Disponível em < https://docs.flutter.dev/resources/architectural-overview >. Publicado em 2015. Acesso em 10 set. 2022.

GEEKHUNTER, Flutter, por que aprender o framework da Google é uma boa ideia em um mercado mobile crescente. Documento eletrônico. Disponível em <https://blog.geekhunter.com.br/flutter/> . Publicado em 11/2021. Acesso em 15 set. 2022.

GUEDES, Gilleanes T. A. **UML 2 – Uma abordagem prática.** Editora Novatec. 2011.

DEVMEDIA. O QUE É UML E DIAGRAMAS DE CASO DE USO: INTRIDUÇÃO PRÁTICA À UML. Disponível em < https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Publicado em 2012. Acesso em 01 jan. 2022.

DEVMEDIA. ORIENTAÇÕES BÁSICAS NA ELABORAÇÃO DE UM DIAGRAMA DE CLASSES. Disponível em < https://www.devmedia.com.br/orientacoes-basicas-na-elaboracao-de-um-diagrama-de-classes/37224>. Publicado em 2012. Acesso em 01 jan. 2022.

IBM. DIAGRAMA DE CLASSES. Disponível em < https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=structure-class-diagrams>. Publicado em março/2021. Acesso em 01 jan. 2022.

NESTJS. ***The open source platform designed for the future. Build enterprise*** . Disponível em <https://nestjs.com/>. Publicado em 2017. Acesso em 10 set. 2022.

SEN, RASIM. **Node.js Framework Series — 1.2.2. NestJS — Módulos**. Disponível em: <https://rasimsen.medium.com/node-js-framework-series-1-2-2-nestjs-modules-65908caf1bdb>. Publicado em fev/2021. Acessado em nov/2022.

ZAMMETTI, Frank. **Flutter na prática.** Editora Novatec.2020.

ARAÚJO, Everton Coimbra. **Aprofundando em Flutter: Desenvolva aplicações Dart com Widgets.** Google Books. 2021.

BOOCH, Grady; RUMBAUCH, James; JACOBSON, Ivair. **UML – Guia do usuário.** Rio de Janeiro: Campus, 2006. ISBN: 978-85-352-1784-1.

MARCOS, LUIS. Flutter, Crie Aplicativos Nativos para Android e IOS. Disponível em: ‹https://luizmarcus.com/android/flutter-crie-aplicativos-nativos-para-android-e-ios› Publicado em Janeiro/2019. Acessado em 02 de jul.2022