UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

Adson Pereira Leal
adsonpleal@gmail.com

Angelo Manoel de Matos Leal
angelommleal@gmail.com

Scholar: Desenvolvimento de um aplicativo móvel genérico de apoio acadêmico a estudantes em universidades

Volume único

Florianópolis
2019

Adson Pereira Leal
adsonpleal@gmail.com

Angelo Manoel de Matos Leal
angelommleal@gmail.com

Scholar: Desenvolvimento de um aplicativo móvel genérico de apoio acadêmico a estudantes em universidades

Volume único

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de informação.
Orientador: Prof. Dr. Leandro Komosinski.

Florianópolis
2019

RESUMO

Com a sociedade cada vez mais imediatista, é notório o surgimento de cada vez mais aplicativos que visam facilitar as diversas tarefas do cotidiano das pessoas. Para um aluno universitário por exemplo, seria interessante obter informações relevantes a sua vida acadêmica de forma centralizada e rápida, tais como horário e local das aulas, controle de faltas, cardápio do Restaurante Universitário, atestado de matrícula, entre outras informações. A falta de tais aplicativos em muitas instituições de ensino acaba tornando esta tarefa trabalhosa e maçante para muitas pessoas, visto que muitas soluções existentes não foram projetadas pensando em dispositivos móveis.

O desenvolvimento de um aplicativo móvel genérico de código aberto, que possa ser usado em várias instituições com pouco esforço para alteração no código, foi a solução encontrada para sanar este problema. O aplicativo, batizado de Scholar, foi desenvolvido utilizando a tecnologia Flutter e foi publicado inicialmente somente para Android e Web, devido ao custo da publicação para iOS, e foi modelado de forma a abstrair os dados provenientes das instituições, tornando possível a utilização por diferentes entidades apenas utilizando os *endpoints* fornecidos pelas mesmas, requerendo o mínimo possível de mudança no código.

Ao fim do projeto, obteve-se uma aplicação totalmente funcional cumprindo todos os requisitos levantados inicialmente, atingindo mais de quinhentos usuários após a última etapa de divulgação, com usuários ativos diariamente. Com a liberação do código fonte como código aberto no Github, espera-se que eventualmente o trabalho possa ser continuado utilizando outras instituições como fonte de informação.

Palavras Chave: Auxílio estudantil. Aplicativo de apoio acadêmico. Aplicativo genérico. Flutter. Dart. Tecnologia híbrida.

SUMÁRIO

1 INTRODUÇÃO 8
1.1 JUSTIFICATIVA 9
1.2 OBJETIVOS 9
1.2.1 OBJETIVO GERAL 9
1.2.2 OBJETIVOS ESPECÍFICOS 10
1.2.3 PREMISSAS E RESTRIÇÕES 10
1.3 METODOLOGIA 10
1.4 ESTRUTURA DO TEXTO 12

2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 APLICAÇÕES HÍBRIDAS 14
2.2 APIs WEB 15
2.2.1 PROTOCOLO HTTP 16
2.3 METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO 16
2.3.1 SCRUM 17

3 SOLUÇÃO PROPOSTA 20
3.1 SOLUÇÕES EXISTENTES 20
3.1.1 Minha UFSC 20
3.1.2 GraduApp 21
3.1.3 Moodle 23
3.2 COMPARAÇÃO AO SCHOLAR 23
3.3 REQUISITOS FUNCIONAIS 24
3.3.1 Diagrama de casos de uso 27

4 DESENVOLVIMENTO 28
4.1 TECNOLOGIAS UTILIZADAS 28
4.1.1 FRAMEWORK FLUTTER 28
4.1.1.1 LINGUAGEM DART 29
4.1.1.2 WIDGETS 29
4.1.2 FIREBASE 30
4.1.3 BLoC 31
4.2 SPRINTS 32
4.2.1 SPRINT 1 (Autenticação, Integração com idUFSC, Controle de faltas, Telas iniciais) 33
4.2.1.1 Planejamento dos requisitos 33
4.2.1.2 Desenvolvimento 33
4.2.1.3 Resultados 34
4.2.2 SPRINT 2 (Cardápio dos RUs, Grade de horários, Eventos, Notificações) 37
4.2.2.1 Planejamento dos requisitos 37
4.2.2.2 Desenvolvimento 37

4.2.2.3 Resultados 40
4.2.3 SPRINT 3 (Integração com idUFSC, Conclusão do MVP, Lançamento da versão Android, Manutenções corretivas) 43
4.2.3.1 Planejamento dos requisitos 43
4.2.3.2 Desenvolvimento 43
4.2.3.3 Resultados 44
4.2.4 SPRINT 4 (Integração com idUFSC, Lançamento da versão Android) 45
4.2.4.1 Planejamento de requisitos 45
4.2.4.2 Desenvolvimento 46
4.2.4.3 Resultados 46
4.2.5 SPRINT 5 (Visualizar detalhes de eventos, Editar ou remover eventos, Manutenções corretivas) 47
4.2.5.1 Planejamento de requisitos 47
4.2.5.2 Desenvolvimento 48
4.2.5.3 Resultados 49
4.2.6 SPRINT 6 (Histórico síntese, Atestado de matrícula, Currículo do curso, Melhoria nas notificações, Manutenções corretivas) 50
4.2.6.1 Planejamento de requisitos 50
4.2.6.2 Desenvolvimento 51
4.2.6.3 Resultados 52
4.2.7 SPRINT 7 (Versão Web) 52
4.2.7.1 Planejamento de requisitos 52
4.2.7.2 Desenvolvimento 53
4.2.7.3 Resultados 54

5 RESULTADOS OBTIDOS 56
5.1 PESQUISA DE SATISFAÇÃO 60
5.2 Dados das atividades dos usuários 66
5.2.1 Usuários Ativos 67
5.2.2 Retenção de usuários 68
5.2.3 - Informações demográficas 68
5.2.4 - Engajamento diário médio 69
5.2.5 - Falhas 69

6 CONCLUSÃO 71
6.1 TRABALHOS FUTUROS 72

REFERÊNCIAS BIBLIOGRÁFICAS 74

APÊNDICE A - Código Fonte 79

APÊNDICE B - Questionário de Satisfação 80

APÊNDICE C - Artigo 83

LISTA DE FIGURAS

Figura 1 - Comunicação entre cliente e servidor (Fonte: CSE - IIT Kanpur) 15
Figura 2 - Funcionamento do Scrum (Fonte: PEREIRA;ISIDORO;MOREIRA;ÁVILA, 2013) 19
Figura 3 - Página da Playstore do aplicativo Minha UFSC (Fonte: Google Playstore) 21
Figura 4 - Página da Playstore do aplicativo GraduApp (Fonte: Google Playstore) 22
Figura 5 - Diagrama de casos de uso (Fonte: Autores) 27
Figura 6 - Hierarquia de widgets no Flutter (Google, 2019b) 30
Figura 7 - Diagrama da arquitetura BLoC (Fonte: Autores) 32
Figura 8 - Classe "AuthRepository", mostrando as assinaturas dos métodos de autenticação (Fonte: Autores) 34
Figura 9 - Resultados obtidos no Sprint 1 (Fonte: Autores) 36
Figura 10 - Implementação da classe NotificationsService e a assinatura de seus métodos (Fonte: Autores) 38
Figura 11 - Método _fetchSubjects da classe UfscService (Fonte: Autores) 39
Figura 12 - Inicialização das variáveis _apiUrl e _userTimeGrid (Fonte: Autores) 39
Figura 13 - Função parseRUTrindade (Fonte: Autores) 40
Figura 14 - Resultados obtidos na segunda sprint (Fonte: Autores) 42
Figura 15 - Tela de seleção de campus (Fonte: Autores) 45
Figura 16 - Página da Google Play do aplicativo Scholar (Fonte: Autores) 47
Figura 17 - Tela de detalhes de eventos com ícone de exclusão (Fonte: Autores) 50
Figura 18 - Código para buscar os dados do usuário logado na versão web (Fonte: Autores) 53
Figura 19 - Código para buscar os dados do usuário logado na versão mobile (Fonte: Autores) 53
Figura 20 - Aplicação rodando diretamente em um browser (Fonte: Autores) 55
Figura 21 - Diagrama de classes (Fonte: Autores) 56
Figura 22 - Perfil (Fonte: Autores) 57
Figura 23 - Agenda (Fonte: Autores) 57
Figura 24 - Detalhes de eventos (Fonte: Autores) 58
Figura 25 - Controle de faltas (Fonte: Autores) 58
Figura 26 - Ajuda no controle de faltas (Fonte: Autores) 59
Figura 27 - Grade de horários (Fonte: Autores) 59
Figura 28 - Cardápio (Fonte: Autores) 60
Figura 29 - Respostas da primeira pergunta do questionário de satisfação (Fonte: Autores) 61
Figura 30 - Respostas da segunda pergunta do questionário de satisfação (Fonte: Autores) 61
Figura 31 - Respostas da terceira pergunta do questionário de satisfação (Fonte: Autores) 62
Figura 32 - Respostas da quarta pergunta do questionário de satisfação (Fonte: Autores) 62
Figura 33 - Respostas da quinta pergunta do questionário de satisfação (Fonte: Autores) 63
Figura 34 - Respostas da sexta pergunta do questionário de satisfação 64
Figura 35 - Respostas da sétima pergunta do questionário de satisfação (Fonte: Autores) 64

FIGURA 36 - RESPOSTAS DA OITAVA PERGUNTA DO QUESTIONÁRIO DE SATISFAÇÃO 65
FIGURA 37 - RESPOSTAS DA NONA PERGUNTA DO QUESTIONÁRIO DE SATISFAÇÃO 65
FIGURA 38 - USUÁRIOS ATIVOS POR PERÍODO (FONTE: FIREBASE DO PROJETO) 67
FIGURA 39 - RETENÇÃO DE USUÁRIOS POR DIA DA AQUISIÇÃO (FONTE: FIREBASE DO PROJETO) 68
FIGURA 40 - INFORMAÇÕES DEMOGRÁFICAS DOS USUÁRIOS DO SCHOLAR (FONTE: FIREBASE DO PROJETO) 69

LISTA DE ABREVIAÇÕES

UFSC - Universidade Federal de Santa Catarina

CAGR - Sistema de Controle Acadêmico da Graduação

SDK - Software Development Kit (Kit de desenvolvimento de software)

HTTP - HyperText Transfer Protocol

AUP - Agile Unified Process

TDD - Test-Driven Development

FDD - Feature-Driven Development

XP - Extreme Programming

PO - Product Owner

PR - Push Request (Github);

1 INTRODUÇÃO

Com o mundo se tornando cada vez mais interconectado, tanto economicamente quanto socialmente, a adoção de tecnologias continua sendo um dos fatores decisivos no progresso humano (Poushter, 2016). Somente no Brasil, temos 220 milhões de celulares inteligentes, mais de 1 por habitante. Somando os notebooks e tablets, são 306 milhões de dispositivos portáteis em maio de 2018, ou seja, 1,5 dispositivo móvel por habitante (29ª Pesquisa Anual do Uso de TI, 2018). A partir dessas constatações, é facilmente perceptível que cada vez mais é crescente a presença das tecnologias móveis no cotidiano das pessoas, servindo múltiplos propósitos. Com esse crescimento gigantesco no mercado de dispositivos móveis, é cada vez mais comum o surgimento de aplicativos, principalmente de smartphones, com vários propósitos, que vão desde o entretenimento até transações financeiras ou monitoramento de saúde. Partindo deste fato conhecido, surgiu a ideia do desenvolvimento deste trabalho.

Um sistema de informação adequado se torna indispensável em um ambiente acadêmico, pois, a necessidade da integridade e do acesso rápido à informação se torna evidente (LEITE, 2017), sendo assim, existem atualmente vários aplicativos que visam auxiliar a vida acadêmica de estudantes em diversas universidades. Podemos tomar como exemplo, o aplicativo Minha UFSC (MAFRA, GASPARIN, 2013), desenvolvido na Universidade Federal de Santa Catarina no ano de 2013, como um trabalho de conclusão de curso. O problema encontrado na pesquisa é que muitas universidades não possuem um aplicativo próprio onde o estudante pode facilmente acessar suas notas, grade curricular, grade de horários, contatos dos professores, histórico curricular, entre outras informações relevantes à vida estudantil.

É fato que com a popularidade da Internet e a facilidade de acesso à rede nos dias de hoje fez com que as Universidades e instituições de ensino buscassem alternativas para disponibilizar online as informações acadêmicas para seus alunos, professores e funcionários em geral. Essa necessidade instigou as instituições a buscarem soluções já existentes ou a desenvolverem aplicações próprias para centralizar o conteúdo ministrado nas aulas e facilitar o acesso à essas informações por parte dos interessados. Porém, mesmo com toda a popularidade das aplicações móveis nos dias de hoje, poucas são as alternativas que possibilitam o uso nativo em sistemas móveis, como Android e iOS, forçando o usuário a recorrer ao navegador de seu dispositivo móvel para acessar as informações. Acessar sites na

Web que não foram pensados para dispositivos móveis, como Smartphones, pode ser uma experiência bastante desagradável.

Pensando em facilitar a vida de universitários em todo Brasil, surgiu a ideia deste projeto. Desenvolver um aplicativo genérico que possa ser integrado às fontes de dados das diversas instituições de ensino ao redor do país, mantendo um mesmo padrão na exibição dos dados ao usuário, com uma navegação facilitada e intuitiva. A adoção de políticas "open-source" no desenvolvimento do protótipo, visa facilitar a posterior implementação de possíveis interessados em utilizar o modelo, em diversas outras instituições de ensino.

1.1 JUSTIFICATIVA

A motivação do trabalho foi a dificuldade que os usuários de universidades encontram para acessar informações acadêmicas pertinentes, quando são necessárias, de uma forma rápida e centralizada. Esse problema foi identificado pelos autores tanto por passarem por isso durante toda sua vida acadêmica, quanto pela conversa com outros estudantes, que também tinham reclamações referentes a este problema.

Tomando a UFSC como exemplo: todos os dados acadêmicos dos alunos estão disponíveis no CAGR, que é um site não responsivo, funcionando bem apenas em computadores, mas não em dispositivos móveis. Nestes dispositivos, por não ser responsivo, o site se comporta exatamente igual como se estivesse sendo acessado de um computador, fazendo o usuário ter que dar zoom e por vezes alterar a visualização do *smartphone* para modo paisagem, para poder acessar as informações com mais clareza.

Dessa forma, decidiu-se realizar este trabalho para solucionar este problema, desenvolvendo um aplicativo móvel que centralizasse estas informações acadêmicas em um só lugar, de uma forma simples e fácil de visualizar, buscando facilitar a vida dos estudantes universitários. A ideia inicialmente é desenvolver o aplicativo tomando a UFSC como base, mas desenvolver o aplicativo de forma modular, que possa ser facilmente adaptado para qualquer outra universidade.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

O objetivo geral deste trabalho é, através da aplicação de metodologias ágeis de análise e desenvolvimento, desenvolver um protótipo de aplicativo móvel genérico para auxílio da vida acadêmica de estudantes em universidades.

1.2.2 OBJETIVOS ESPECÍFICOS

Por meio da coleta de dados dos potenciais usuários e da análise dos mesmos, os objetivos específicos deste trabalho são:

- Agregar em único local as principais informações do aluno referentes a sua vida acadêmica, tais como notas, grades de horários, entre outras;
- Tornar prático o acesso a tais informações através de um aplicativo projetado especificamente para dispositivos móveis;
- Desenvolver um aplicativo de forma genérica, encapsulando as informações provenientes de fontes de dados diferentes, para que possa ser utilizado por outras instituições de ensino com o mínimo de mudança no código;
- Utilizar uma abordagem de software livre no desenvolvimento, disponibilizando publicamente o aplicativo após seu término.

1.2.3 PREMISSAS E RESTRIÇÕES

Como premissa deste projeto tem-se que o aplicativo objetivado a ser desenvolvido será um protótipo e dependerá da disponibilidade das informações por parte das instituições onde porventura possa vir a ser aplicado. Este projeto tem como maior restrição a possível falta de APIs de integração de dados nas instituições, inviabilizando o uso da aplicação.

1.3 METODOLOGIA

A pesquisa para a realização deste trabalho pode ser classificada como exploratória, tendo em vista que o desenvolvimento do aplicativo em questão demanda entrevistas com possíveis usuários para alinhar expectativas e obter informação para melhor compor o projeto, além da realização de testes e levantamento de material bibliográfico de apoio.

Para o desenvolvimento deste trabalho, será aplicada a metodologia multimétodo, dividida em etapas:

Etapa 1 - Síntese dos conceitos fundamentais e teóricos: O foco desta etapa gira em torno da análise e definição dos conceitos básicos que servirão de base para o desenvolvimento do

projeto como um todo. A síntese dos conceitos será focada principalmente em metodologias ágeis sendo o Scrum a escolhida, Framework Flutter e suas ferramentas, Firebase e suas ferramentas, Business Logic Component (BLoC) e suas ferramentas, conceituação sobre aplicações Web, conceituação sobre metodologias ágeis em geral. Para melhor definir o trabalho, os conceitos técnicos foram separados e agrupados no Desenvolvimento (Capítulo 3). Esta etapa compreende as seguintes atividades:

1. Síntese dos conceitos sobre aplicações Web, elencando as principais características e como funcionam estes tipos de aplicações;
2. Síntese dos conceitos referentes a metodologias ágeis, com foco no Scrum, abordando aplicações móveis e adaptada para uma equipe de apenas duas pessoas;

Etapa 2 - Análise, implementação e testes: O foco desta etapa é a realização das três principais tarefas do projeto, de modo que esta etapa compreende as seguintes tarefas:

1. **Análise**: Nesta tarefa, objetiva-se todo o processo analítico referente ao projeto em questão, compreendendo as seguintes fases:
    a. Entrevistas: O objetivo desta fase da análise é obter informações diretamente dos possíveis usuários através de questionários e entrevistas, que servirão como base de conhecimento para melhor projetar as telas da aplicação na fase seguinte;
    b. Prototipação: Nesta fase do projeto, toda a informação obtida até então, fornecida pelos potenciais usuários, é compilada com o intuito de nortear a criação dos primeiros protótipos de telas da aplicação, a serem validados consultando os possíveis usuários;
    c. Levantamento de requisitos: Nesta fase, os requisitos levantados para a implementação do projeto são concebidos, levando em consideração toda a informação obtida até então nas fases anteriores. São eles que guiarão a etapa de Implementação e serão a base das funcionalidades constantes no aplicativo.
2. **Implementação**: O objetivo desta fase é a produção do artefato principal do projeto, que é o aplicativo. O desenvolvimento se dará com base nas informações obtidas na etapa de Análise, isto é, o feedback transmitido pelos potenciais usuários, as telas prototipadas e os requisitos levantados. A fase de implementação se dará em duas etapas:

a. Desenvolvimento do *back-end*:

O desenvolvimento do *back-end* consiste em projetar três partes: o banco de dados, a API e os *parsers* dos restaurantes universitários. Para redução de custos do projeto e facilidade de implementação será utilizada a ferramenta de banco de dados e API **Firebase**, de propriedade do **Google,** que possui um plano grátis para projetos de pequeno porte. A API e os *parsers* serão escritos em Javascript, NodeJS, desse modo tornando o projeto mais fácil de ser mantido.

b. Desenvolvimento do *front-end*:

O *front-end* consiste numa aplicação mobile desenvolvida em Flutter, tecnologia criada pelo Google Flutter é uma tecnologia que permite a criação de apps nativos a partir de um código escrito na linguagem Dart.

3. **Teste**: Finalmente, após a conclusão do aplicativo, serão efetuados testes exploratórios mais aprofundados, para validar o correto funcionamento do mesmo e a confiabilidade das informações apresentadas.

1.4 ESTRUTURA DO TEXTO

A estrutura do texto deste trabalho se organiza da seguinte forma:

- Capítulo 2 Fundamentação teórica: Neste capítulo e em seus sub-tópicos serão descritos todos os conhecimentos teóricos utilizados necessários para o desenvolvimento deste trabalho;
- Capítulo 3 Desenvolvimento: Neste capítulo e em seus sub-tópicos serão descritos primeiramente os conhecimentos tecnológicos e técnicos utilizados neste trabalho e na sequência, como o trabalho foi organizado e realizado, destacando cada sprint realizada;
- Capítulo 4 Resultados obtidos: Aqui serão descritos todos os resultados obtidos ao final do desenvolvimento do projeto, descrevendo a situação final do aplicativo criado;
- Capítulo 5 Conclusão: Neste capítulo foi realizado um compilado das principais conclusões obtidas, levando em conta as experiências vivenciadas no decorrer do

projeto, tais como experiências com usuários, framework utilizado, situação final do aplicativo, etc.;

- Por fim, as Referências utilizadas no desenvolvimento deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Tomando como base a UFSC, o acesso aos dados do CAGR via dispositivos móveis é um verdadeiro desafio. Não existe um aplicativo móvel nativo para o sistema, de modo que o mesmo deve ser acessado via o navegador do dispositivo, possuindo o mesmo formato que sua versão Web acessada via um computador por exemplo. Esse problema estende-se à outras universidades, as quais não possuem aplicativos móveis para apresentar aos seus estudantes as informações acadêmicas necessárias aos mesmos, como atestado de matrícula, grade de horários, histórico acadêmico, currículo do curso, etc. O desenvolvimento proposto por este trabalho, de um modelo de aplicativo genérico, que possa ser usado por diferentes instituições, tende a facilitar o acesso à estes dados por parte dos estudantes.

2.1 APLICAÇÕES HÍBRIDAS

Aplicações multiplataforma, ou híbridas, são aquelas desenvolvidas com uma determinada linguagem, utilizando algum *framework* de auxílio, que geram uma aplicação final capaz de rodar em múltiplos dispositivos e plataformas diferentes, utilizando a mesma base de código que foi implementada uma única vez. Esses tipos de aplicações são bastante interessantes porque geram economia de recursos para empresas de desenvolvimento de software, visto que apenas uma equipe é necessária para desenvolver aplicações para diferentes plataformas. Além disso, a aplicação final fica idêntica independente da plataforma utilizada, exceto pelas nuances de cada sistema operacional.

Para funcionar, uma aplicação híbrida precisa ser desenvolvida tendo como base algum framework de apoio. Definimos uma framework multiplataforma como um conjunto de arquivos de código fonte, bibliotecas e ferramentas que suportam pelo menos duas plataformas diferentes e que permitem o desenvolvimento sem ramificações do código fonte (FREIRE; RIBEIRO, 2013).

Existem atualmente inúmeros frameworks que possibilitam o desenvolvimento de aplicações híbridas, como por exemplo: Cordova, Ionic, React Native, Xamarin, Unity, Adobe Air, Flutter, entre outros. Para o desenvolvimento deste trabalho, o framework escolhido foi o Flutter.

Independente do framework, sua função para com o código gerado, é bastante semelhante. O framework é o responsável por fazer a ponte entre a aplicação e o dispositivo, através de suas APIs. Também é ele o responsável por empacotar a aplicação e gerar o executável para cada plataforma diferente.

2.2 APIs WEB

API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A sigla API refere-se ao termo em inglês "*Application Programming Interface*" que significa em tradução para o português "Interface de Programação de Aplicativos". Esse tipo de aplicação utiliza a internet para a transferência de dados e processa e armazena informações relevantes para o funcionamento de um produto ou serviço.

Esse tipo de aplicação possui seu funcionamento baseado na conversação entre cliente e servidor, através de um protocolo de comunicação, que fica responsável por transpor os dados entre um e outro. A Figura 1 demonstra esse fluxo de informações. Os passos presentes na figura são descritos logo abaixo.

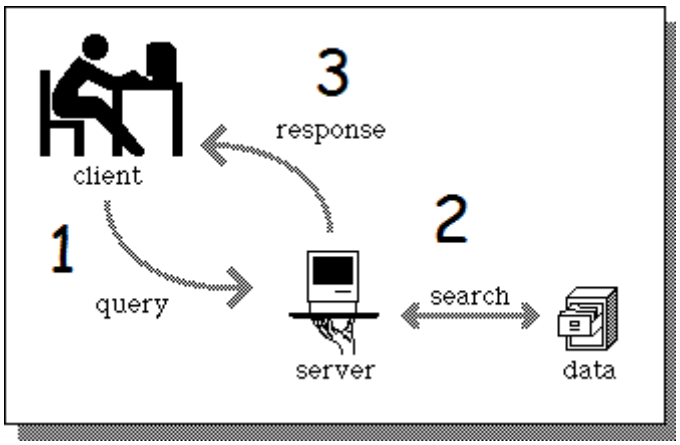

Figura 1 - Comunicação entre cliente e servidor (Fonte: CSE - IIT Kanpur)

- **Passo 1 (*query*)**: O cliente gera uma solicitação ao servidor, geralmente através do protocolo HTTP. Seja solicitando informações do usuário, ou informando credenciais para acesso de login por exemplo. O servidor então recebe esta requisição, avalia o tipo da mesma e faz o seu tratamento.
- **Passo 2 (*search*)**: Caso a solicitação feita pelo cliente precise de acesso ao banco de dados, o servidor se encarrega de realizar este acesso para obter as informações necessárias. Este acesso pode ser somente uma consulta, para o caso de uma verificação de credenciais de login, ou de inclusão, para o caso de inserção de um novo usuário por exemplo. Após realizar as consultas (geralmente em linguagem SQL), o servidor obtém então a resposta do banco de dados e passa para o passo 3;
- **Passo 3 (*response*)**: A resposta obtida pelo servidor, via acesso ao banco de dados ou não, é finalmente encaminhada ao cliente.

2.2.1 PROTOCOLO HTTP

HTTP é um protocolo que permite a obtenção de recursos, tais como documentos HTML. É a base de qualquer troca de dados na Web e um protocolo cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, geralmente um navegador da Web. Um documento completo é reconstruído a partir dos diferentes sub-documentos obtidos, como por exemplo texto, descrição do layout, imagens, vídeos, scripts e muito mais (MDN, 2019a).

Este protocolo é bastante extensível, por meio de seus “headers”, introduzidos no HTTP/1.0. Estes cabeçalhos permitem que o cliente e o servidor passem informações adicionais com a solicitação e a resposta HTTP (MDN, 2019b).

2.3 METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO

As definições modernas de desenvolvimento de software ágil evoluíram a partir da metade de 1990 como parte de uma reação contra métodos "pesados", caracterizados por uma pesada regulamentação, regulamentação e micro gerenciamento usando o modelo em cascata para desenvolvimento (WAZLAWICK, 2013). Foi criado então o Manifesto Ágil.

O Manifesto Ágil é uma declaração de valores e princípios essenciais para o desenvolvimento de software. Ele foi criado em fevereiro de 2001, quando 17 profissionais, que já praticavam métodos ágeis como XP, DSDM, Scrum, FDD etc, se reuniram nas montanhas nevadas do estado norte-americano de Utah (CAMARGO, 2018).

Para melhor compreensão dessa nova filosofia de desenvolvimento, o grupo também criou uma lista com doze princípios que devem ser seguidos. São eles (BECK, 2001):

- Satisfação do cliente como prioridade mais alta, através de entregas contínuas e adiantadas de software com valor;
- Dar boas vindas a mudanças de requisitos, mesmo que no fim do desenvolvimento;
- Entregar software funcionando frequentemente, de preferência com uma curta escala de tempo;
- Pessoas relacionadas à negócios e desenvolvedores devem trabalhar juntos diariamente durante o projeto;
- Construir projetos com indivíduos motivados, dando o ambiente e o suporte que precisam e acreditando que farão seu trabalho;

- A forma mais eficiente e efetiva de trocar informações durante um projeto é por meio de conversas face a face;
- Software funcional é a principal medida de progresso;
- Desenvolvimento sustentável. Patrocinadores, usuários e desenvolvedores devem manter um ritmo constante indefinidamente;
- Atenção contínua à excelência técnica e bom design;
- Simplicidade - a arte de maximizar a quantidade de trabalho que não precisa ser feito;
- As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizáveis;
- Em intervalos regulares, a equipe deve refletir sobre como se tornar mais eficiente.

A partir daí, várias metodologias descritas como ágeis foram emergindo, tais como: *Agile Unified Process* (AUP), *Test-Driven Development* (TDD), *Feature-Driven Development* (FDD), entre outras. Além destas, duas dessas metodologias se tornaram mais notórias no mercado: *Scrum* e *Extreme Programming* (XP). Neste trabalho, decidimos por utilizar a metodologia *Scrum* para o desenvolvimento, mas adaptando-a à nossa realidade.

2.3.1 SCRUM

O *Scrum* é atualmente uma das metodologias ágeis mais adotadas por profissionais da área de desenvolvimento de software, tanto em pequenas quanto em grandes empresas do ramo. O *survey* global realizado em 2018 pela ScrumAlliance, "*State of Scrum Report*", demonstra isso (SCRUMALLIANCE, 2018).

Segundo o guia "*The Scrum Guide*", o *Scrum* é um framework no qual as pessoas podem resolver problemas adaptativos e complexos, enquanto entregam produtos com o maior valor possível, de forma produtiva e criativa (SCHWABER; SUTHERLAND, 2017). Ainda segundo o mesmo guia, esta metodologia tem como principais características ser: leve, simples de entender, difícil de dominar.

As equipes que trabalham utilizando o *Scrum* são chamadas de *Scrum Team*, e dividem-se primariamente em três funções: o *Product Owner* (PO), o *Development Team* (Time de Desenvolvimento) e o *Scrum Master*. As atribuições e definições de cada integrante do time, são descritas a seguir, ainda segundo o "*The Scrum Guide*" (SCHWABER; SUTHERLAND, 2017):

- *Product Owner* (PO): responsável por maximizar o valor do produto resultante do trabalho do time de desenvolvimento. A forma como isso deve ser feito varia

amplamente entre organizações e times. Além disso, o PO é a pessoa responsável por gerenciar o *backlog* do produto, de modo que os itens estejam claros para todos e a ordenação dos mesmos vise o cumprimento dos objetivos da melhor forma possível. É importante que as decisões do PO sejam respeitadas por toda a organização, para que o trabalho tenha sucesso;

- Time de Desenvolvimento: consiste em profissionais que trabalham para entregar um incremento visando o produto "pronto" no fim de cada sprint. Cada time deve ter autonomia para se auto-gerenciar e organizar o trabalho da melhor forma possível, visando resultados. Um time deve ser multifuncional, ou seja, cada membro deve possuir habilidades complementares uns aos outros, de forma que todas as habilidades necessárias para alcançar os objetivos estejam presentes. O tamanho do time também é um fator importante, idealmente deve conter entre três e nove membros;
- *Scrum Master*: é o responsável por promover e dar suporte ao funcionamento do Scrum no time e na organização, junto aos outros *Scrum Masters*. Seu trabalho é ajudar todos a entenderem a teoria do Scrum, boas práticas, regras e valores, de modo que a metodologia seja um sucesso.

Tratando-se de um modelo iterativo e incremental, o *Scrum* divide o projeto em vários *sprints* (ciclos curtos de desenvolvimento) consecutivos que ocorrerão de acordo com a prioridade do product owner (proprietário de produto). Cada período de *sprint* é definido, geralmente, entre duas e quatro semanas. Durante esse tempo, o *scrum team* se dedica ao máximo para ter um pequeno conjunto de funcionalidades codificadas e testadas (PAGOTTO, 2016).

De forma a entender melhor o funcionamento da metodologia *Scrum* aplicada ao dia-a-dia de uma empresa de desenvolvimento de software, podemos tomar como exemplo a imagem da Figura 2.

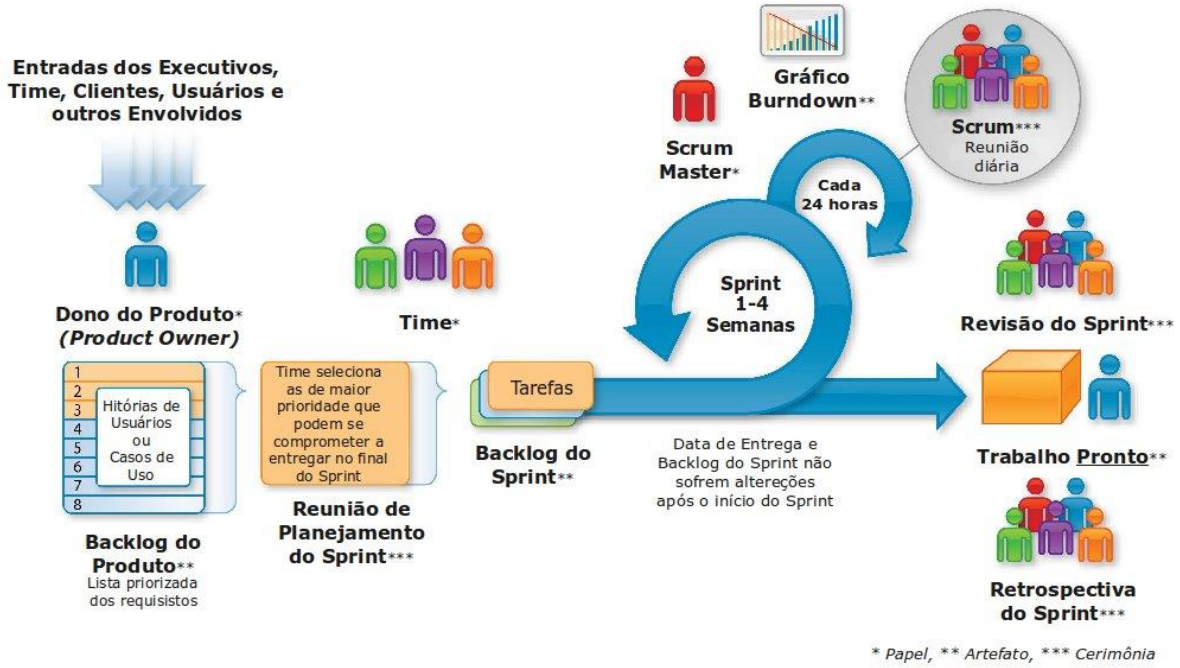

Figura 2 - Funcionamento do Scrum (Fonte: PEREIRA;ISIDORO;MOREIRA;ÁVILA, 2013)

Primeiramente, o PO ordena em uma lista de prioridades, todos os requisitos que foram avaliados pela equipe para o desenvolvimento de um determinado projeto. Essa lista de requisitos é chamada de *Product Backlog*. A partir desta lista, o time realiza uma reunião, chamada de *Sprint Planning*, que serve para avaliar e selecionar os requisitos os quais podem se comprometer a entregar no final desta *Sprint.* É formada então outra lista, chamada de *Sprint backlog*, que é uma versão minimizada do *Product backlog*, contendo todas as tarefas que o time deve realizar na *Sprint.* Diariamente é feita outra reunião, chamada de *Daily Scrum*, onde o time alinha as tarefas e define um plano a ser seguido no próximo dia. Ao final de cada *Sprint* é feita um outro tipo de reunião chamada *Sprint Review* para adaptar e avaliar o *Product Backlog* caso haja necessidade. Também ao fim da *Sprint* é feita mais uma reunião chamada *Sprint Retrospective* para autoavaliação do time onde são sugeridas melhorias para aprimorar o próximo *Sprint* (FERREIRA, 2017).

O desenvolvimento do Scholar foi feito um baseado-se em algumas das ideias principais do *Scrum*, sendo a principal, a divisão dos ciclos de desenvolvimento em *sprints* bem definidas e com escopo definido anteriormente. Como a equipe é formada por apenas dois membros, a definição de papéis do *Scrum* foi deixada de lado. Os artefatos primários ainda foram mantidos, que são o *Product Backlog* e o *Sprint Backlog*, onde o primeiro será formulado de acordo com a análise de requisitos realizada no início do projeto, e o segundo de acordo com a prioridade definida de acordo com a importância de cada requisito.

3 SOLUÇÃO PROPOSTA

Para moldar o aplicativo, foi necessário pesquisar soluções previamente existentes e quais eram as características de cada uma delas, para ser possível modelar o Scholar da melhor forma possível. Três soluções existentes foram encontradas e são detalhadas logo abaixo. Além disso, este capítulo também engloba a elaboração dos requisitos levantados e o diagrama de casos de uso proveniente deste levantamento.

3.1 SOLUÇÕES EXISTENTES

Este capítulo abrange uma síntese de três outras aplicações semelhantes ao aplicativo Scholar, proposto neste trabalho. Dois deles (Minha UFSC e GraduApp) também foram frutos de TCCs passados, porém infelizmente deixaram de funcionar. Estes dois aplicativos foram os que motivaram o desenvolvimento deste TCC. Boa parte das funcionalidades oferecidas por eles também estarão presentes no Scholar.

Outra opção, porém que foge um pouco ao escopo do planejado, é o aplicativo fornecido pelo Moodle. Porém, nem todas as universidades que utilizam o Moodle oferecem suporte para o aplicativo do mesmo.

3.1.1 Minha UFSC

O aplicativo Minha UFSC foi uma solução desenvolvida pelos alunos Marlon Mafra e Ygor Gasparin como parte integrante de seu TCC, no curso de Sistemas de Informação na UFSC, em 2013.

Dentre as principais funcionalidades do mesmo estavam:

- Consulta da grade de horários;
- Histórico escolar;
- Cardápio do RU;
- Estatísticas de IAA;
- Avisos da Universidade;
- Consulta a pendências de livros com a BU.

O app funcionou muito bem nos primeiros anos após seu lançamento, porém depois de um tempo deixou de funcionar devido a mudanças no sistema de autenticação unificada por parte da SeTIC, o que impossibilitava os usuários de efetuarem login no sistema. Isso

pode ser refletido pelas avaliações extremamente negativas do aplicativo na Playstore (Figura 3), que se dão quase que exclusivamente graças a este problema.

Este aplicativo foi a principal inspiração para a elaboração deste trabalho.

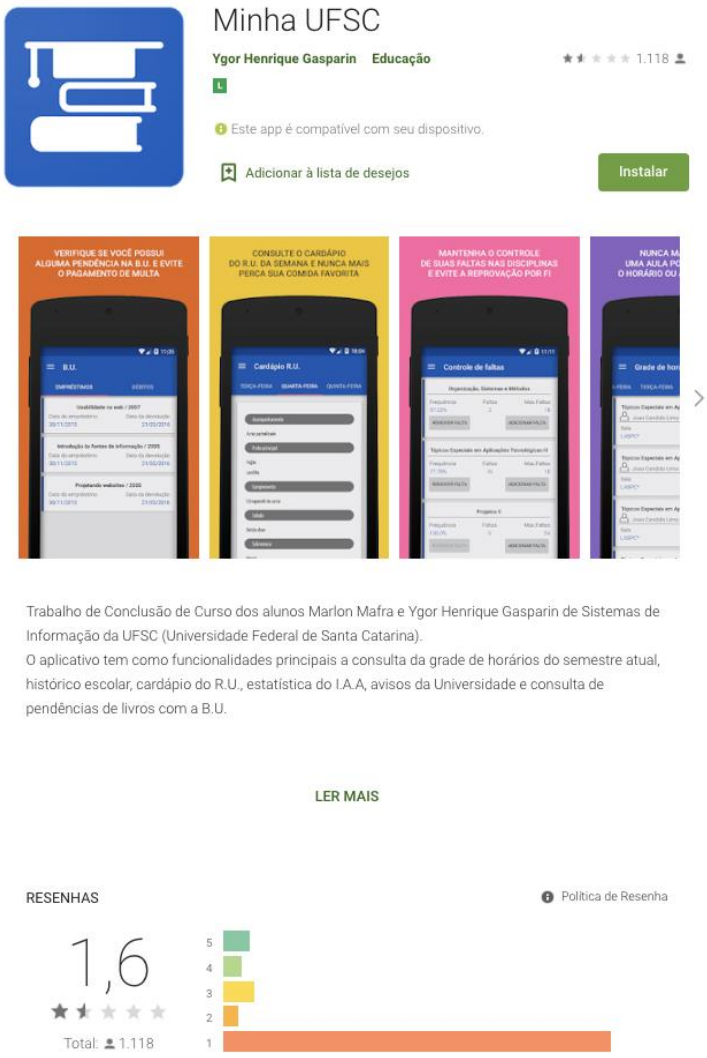

Figura 3 - Página da Playstore do aplicativo Minha UFSC (Fonte: Google Playstore)

3.1.2 GraduApp

O GraduApp é um aplicativo desenvolvido por alunos do curso de Engenharia de Controle e Automação na UFSC. Apesar de ser mais novo que o Minha UFSC, ele acabou sofrendo dos mesmos problemas que o anterior, devido às mudanças no sistema de autenticação unificada da UFSC, realizadas recentemente pela SeTIC.

Como pode ser observado na Figura 4, as avaliações do aplicativo também estão bastante negativas devido aos problemas de conexão que impedem os usuários de utilizar o mesmo.

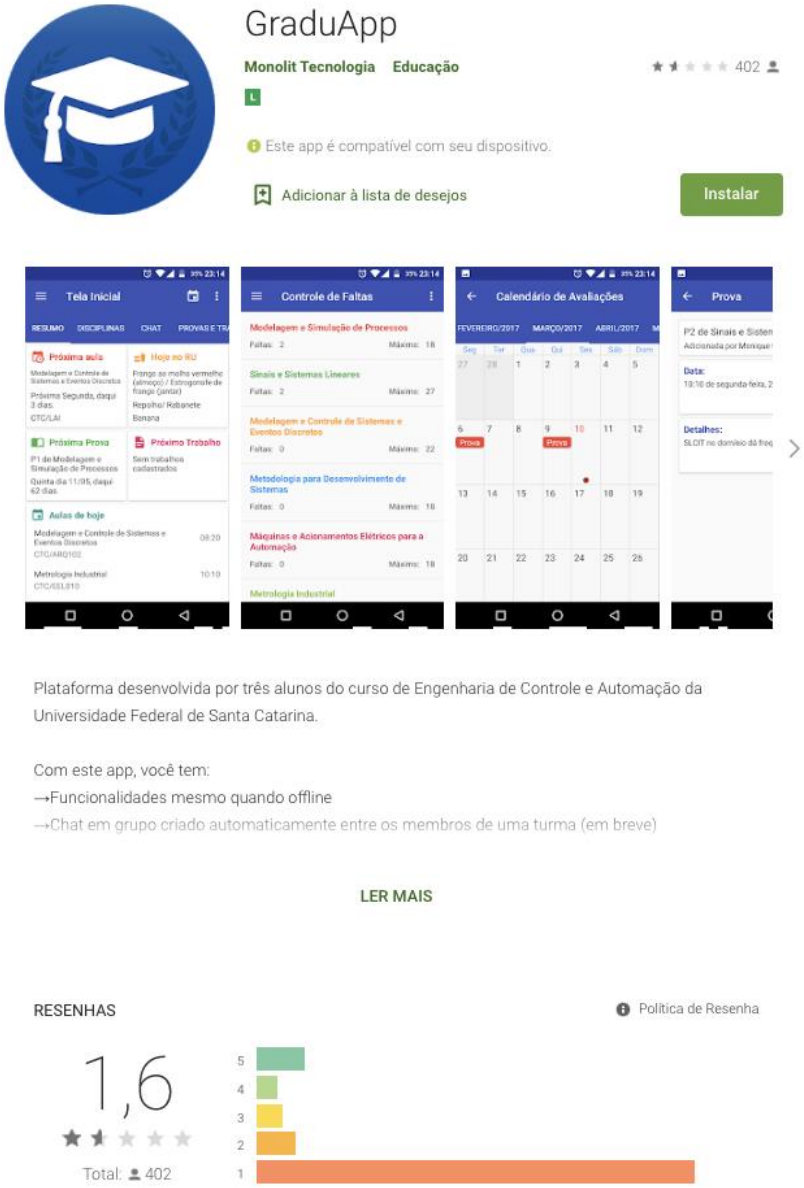

Figura 4 - Página da Playstore do aplicativo GraduApp (Fonte: Google Playstore)

Dentre as principais funcionalidades deste app, estavam:

- Cardápio do RU;
- Grade de horários;
- Avaliações adicionadas individualmente ou por colegas;
- Controle de faltas;
- Calendário de provas;
- Histórico escolar;
- Colegas de curso.

3.1.3 Moodle

O Moodle, utilizado na sua versão web pela UFSC para assuntos relacionados às disciplinas ofertadas, também fornece uma solução Mobile com algumas funcionalidades interessantes, por exemplo:

- Navegar pelo conteúdo oferecido pelos cursos;
- Receber notificações sobre eventos cadastrados no Moodle;
- Encontrar e contactar pessoas do curso;
- Fazer upload de arquivos do seu smartphone;
- Ver as notas dos cursos;
- Entre outros.

Apesar de oferecer soluções interessantes, para que o usuário possa utilizar o aplicativo, primeiramente é necessário que o administrador do sistema configure o ambiente para que seja permitida a utilização do aplicativo. O aplicativo funciona na UFSC, porém possui funcionalidades limitadas.

Para este trabalho, foi levantada a hipótese de, através da autenticação unificada da UFSC, conectar com o *back-end* do sistema do Moodle, para fazer a integração entre Scholar e Moodle. Assim, seria possível integrar algumas funcionalidades como controle de presença, agendamento de trabalhos e provas, etc. Entretanto, o Moodle não possui uma interface de comunicação para disponibilização dos dados, sendo necessário o desenvolvimento prévio desta interface para que esta integração fosse possível. Isto demandaria um trabalho extra considerável, que não se adequaria ao escopo deste projeto.

Apesar de tudo, o Moodle oferece serviços voltados para as disciplinas cursadas, que é um escopo diferente do planejado para o projeto. O Scholar visa facilitar o acesso a informações gerais para auxiliar o estudante, como grade de horários, documentos fornecidos pelo CAGR, cardápio do restaurante, ou seja, informações que independem de quais disciplinas que o usuário está matriculado.

3.2 COMPARAÇÃO AO SCHOLAR

Tomando como base as três aplicações listadas, podemos listar as principais funcionalidades semelhantes entre o aplicativo Scholar, desenvolvido neste trabalho, com os seus antecessores.

As funcionalidades dos antigos aplicativos da UFSC (GraduApp e MinhaUFSC) que foram remodeladas e também inclusas no Scholar foram:

- Grade de horários;
- Cardápio do RU;
- Histórico escolar.

Algumas funcionalidades eram oferecidas apenas no GraduApp e também estão inclusas no Scholar:

- Avaliações adicionadas individualmente ou por colegas;
- Controle de faltas;
- Calendário de provas;

Por fim, existem algumas funcionalidades que existiam nos antigos apps mas que foram deixadas de fora do escopo deste trabalho, como por exemplo:

- Avisos da Universidade;
- Consulta a pendências de livros com a BU.

Já o Moodle *mobile* oferece muitas funcionalidades interessantes que, se integradas ao aplicativo Scholar, promoveriam um resultado ainda mais satisfatório com relação a experiência do usuário. Mas devido a falta de uma API disponível para fazer esta integração, de acordo com a SeTIC, esta possibilidade de integração entre as duas aplicações foi deixada de lado.

3.3 REQUISITOS FUNCIONAIS

O propósito da aplicação, como descrito anteriormente, é servir de apoio aos estudantes que acessam informações referentes a sua universidade. Sendo assim, nosso público alvo, e por assim dizer, nosso cliente, seriam os estudantes. Dessa forma, foram feitas entrevistas informais com os potenciais usuários a fim de levantar os principais requisitos para a aplicação.

Após analisar as respostas obtidas, foi possível traduzir as principais necessidades em requisitos funcionais que a aplicação deveria atender, a fim de suprir as necessidades dos usuários. Os requisitos mapeados foram:

R1 Criar conta

Criar uma conta, com e-mail e senha, para fazer login no aplicativo. Surgiu a ideia de criar este requisito, visto que o propósito da aplicação é de ser um aplicativo genérico. Dessa forma o usuário conecta-se ao aplicativo criando uma conta para ele, e posteriormente faz a

conexão com o sistema da UFSC (ou futuramente, de alguma outra universidade), para fazer a obtenção dos dados acadêmicos daquele aluno.

R2 Recuperar senha

Recuperar senha de uma conta previamente cadastrada, através do e-mail utilizado no cadastro.

R3 Acessar com idUFSC (Acesso unificado)

Conectar a aplicação ao idUFSC do usuário, através do acesso unificado da UFSC, para possibilitar o acesso às informações acadêmicas do mesmo.

R4 Realizar controle de faltas

Controlar o número de faltas em cada disciplina cursada no semestre corrente, por meio da atualização manual do usuário do número de faltas em determinada disciplina.

R5 Acessar Home

Independentemente de qual tela o usuário se encontra dentro do aplicativo, deve ser possível acessar a tela “Home”.

R6 Acessar Perfil

Independentemente de qual tela o usuário se encontra dentro do aplicativo, deve ser possível acessar a tela “Perfil”.

R7 Acessar Agenda

Independentemente de qual tela o usuário se encontra dentro do aplicativo, deve ser possível acessar a tela “Agenda”.

R8 Ativar Notificações

Ativar ou desativar as notificações no smartphone, sempre que o usuário desejar, por meio da tela “Perfil”. As notificações devem vir ativadas por padrão.

R9 Visualizar grade de horários

Visualizar a grade de horários do semestre, mostrando a disciplina, o período de cada aula, a sala onde será ministrada e o professor. Retratando o que é mostrado no CAGR.

R10 Visualizar cardápio do RU

Visualizar o cardápio semanal do Restaurante Universitário de forma prática, podendo escolher o cardápio de qual campus que gostaria de acessar.

R11 Listar eventos

Listar os eventos relacionados às disciplinas cursadas, mostrando os eventos próprios do usuário, bem como os compartilhados entre os alunos que cursam tal disciplina.

R12 Criar e compartilhar eventos

Criar eventos na agenda, tais como provas, entrega de trabalhos, avaliações, etc., e fornecendo a possibilidade de compartilhar este evento com a disciplina em questão, para que os demais alunos que a cursam, também tenham acesso ao evento criado.

R13 Editar ou remover eventos

Editar, ou remover, eventos criados previamente.

R14 Visualizar detalhes de eventos

Visualizar a tela com os detalhes de cada evento presente na “Agenda”.

R15 Visualizar histórico síntese

Visualizar o histórico síntese do aluno diretamente no aplicativo.

R16 Fazer download do atestado de matrícula

Fazer download o atestado de matrícula atualizado, diretamente no aplicativo.

R17 Fazer download do currículo do curso

Fazer o download o currículo do curso.

R18 Receber notificações

Receber notificação 10 minutos antes de uma aula. Receber notificação quando um usuário adiciona uma prova/trabalho na mesma turma.

3.3.1 Diagrama de casos de uso

O diagrama de casos de uso é o diagrama mais geral e informal da UML, utilizado na fase de levantamento e análise de requisitos para representar as principais funcionalidades (usos) do sistema/software. A técnica utilizada para construir um diagrama é identificar os atores (usuários ou outros sistemas) e os serviços que o sistema fornecerá aos atores, conhecidos como casos de uso. O diagrama é construído organizando os atores e casos de uso e especificando os seus relacionamentos (SILVA;PANSANATO;FABRI, 2010).

Com base na aplicação Scholar, foi identificado somente um tipo de usuário final, identificado simplesmente como “Usuário”, visto que não há nenhuma funcionalidade do tipo “Administrador” ou algo semelhante. Todo usuário faz login no sistema e tem acesso às mesmas funcionalidades. O diagrama gerado pode ser observado logo abaixo:

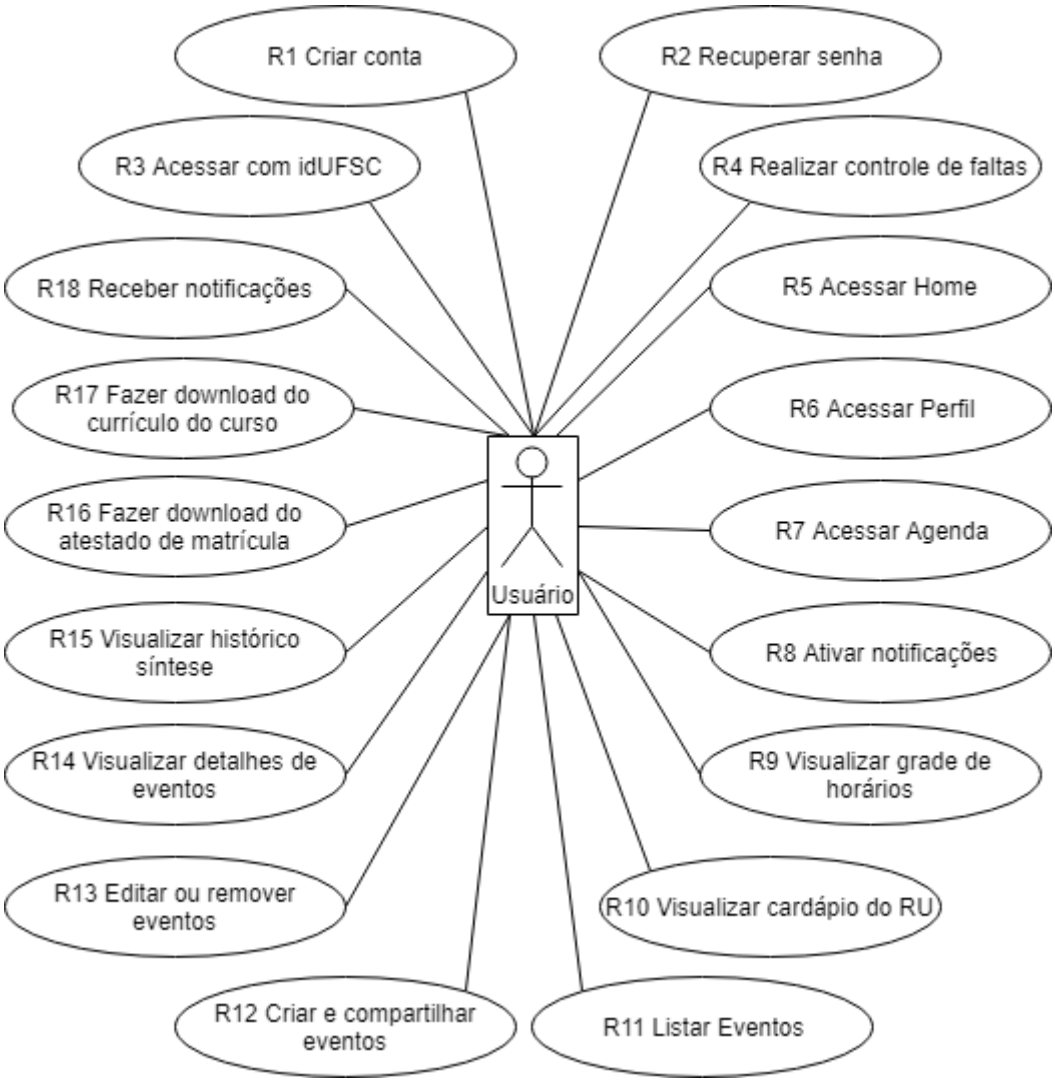

Figura 5 - Diagrama de casos de uso (Fonte: Autores)

4 DESENVOLVIMENTO

Neste capítulo serão descritos todos os tópicos referentes a como foi executado o desenvolvimento deste trabalho, incluindo as tecnologias utilizadas e como foram utilizadas, além da descrição do que foi realizado em cada Sprint.

4.1 TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do aplicativo, realizamos uma pesquisa sobre as possíveis linguagens a serem adotadas. A ideia inicial seria utilizar a React Native para o desenvolvimento, porém, após uma recomendação e um pouco de pesquisa, optamos por utilizar uma ferramenta nova e com bastante potencial, fornecida pela Google, chamada Flutter. Além disso utilizamos o Firebase para o *back-end* do projeto, e o BLoC como padrão de arquitetura do projeto.

4.1.1 FRAMEWORK FLUTTER

Flutter é um framework utilizado para criação de apps de alta performance e fidelidade para as plataformas iOS e Android (Google, 2019a). Foi desenvolvido pela Google e lançado publicamente em maio de 2017, com sua primeira versão pública sendo a Alpha v0.0.6. As aplicações são escritas utilizando a linguagem Dart, também criada pela Google, que é bastante semelhante ao conhecido Javascript. Além da linguagem, uma outra característica é que ele se baseia em “widgets”. Toda a interface de um app utilizando Flutter, é um widget.

Um aspecto diferenciado deste framework é que ao invés de utilizar “web views” ou depender dos widgets nativos do próprio dispositivo, o Flutter renderiza todo componente de suas “views” utilizando sua própria engine de alta performance. Este comportamento possibilita a construção de aplicações que são tão performáticas quanto as próprias aplicações nativas tem a capacidade de ser (Wenhao Wu, 2018).

De forma similar ao React Native, o Flutter tem a funcionalidade “Hot Reload”, a qual possibilita que a aplicação seja rapidamente recarregada após qualquer alteração no código, mantendo o estado da última seção. Este fator auxilia e muito nos processos de desenvolvimento e testes, visto que não é necessário parar e rodar novamente toda a aplicação a cada mudança no código fonte.

4.1.1.1 LINGUAGEM DART

Dart é uma linguagem de programação de propósito geral originalmente desenvolvida pelo Google. Inicialmente revelada na conferência GOTO em Aarhus, Dinamarca, 12 de outubro de 2011. A linguagem pode ser utilizada para o desenvolvimento Web, servidor, desktop e aplicações móveis. É orientada a objetos, possui "Garbage Collector" (coletor de lixo) e uma sintaxe do estilo das linguagens derivadas do C. Pode ser transpilada para Javascript, possibilitando o desenvolvimento Web.

Dart pode ser compilado tanto JIT ("Just In Time") quanto AOT ("Ahead of time"). Linguagens que executam JIT são interpretadas no momento da execução, um exemplo é o Javascript rodando no Node JS. Linguagens AOT são pré-compiladas para uma linguagem de máquina (virtual ou física) antes da execução da aplicação, um exemplo de linguagem com compilação AOT é o C. A Linguagem Dart possui quatro padrões de compilação (PROANDROIDDEV, 2018):

- Script: Modo JIT. Como no Node.js, o código pode ser executado diretamente através da ferramenta de linha de comando da máquina virtual do Dart.
- Script Snapshot: Modo JIT. Diferente do modo "Script", compacta o código em conjuntos de "tokens". O código compactado faz o compilador ganhar tempo pois facilita a análise léxica do código.
- Application Snapshot: Modo JIT. O modo “Application Snapshot” funciona como uma cópia do tempo de execução. Inclui classes e funções parseadas do código fonte, assim o carregamento e a execução podem ser executados antes. Esse tipo de "snapshot" é dependente da arquitetura, "snapshots" gerados para IA_32 não podem ser executados em ARM, por exemplo.
- AOT: Modo AOT. Nesse modo o código Dart é traduzido em arquivos assembly, o assembler converte os arquivos assembly em código binário para as diferentes arquiteturas.

4.1.1.2 WIDGETS

Como dito anteriormente, os blocos fundamentais da construção da interface de um app utilizando Flutter, são os widgets. Diferentemente de outros frameworks que separam views, controladores de views, layouts e outras propriedades, o Flutter possui um modelo consistente e unificado, que é o widget (Google, 2019b). Os widgets são os responsáveis por definir a estrutura de um elemento (se é um texto, um botão, uma lista, etc), o estilo (como a

cor ou tipo de fonte), o aspecto (margens, paddings, etc) e o modo como devem responder a diferentes eventos.

Os widgets são ordenados de forma hierárquica, de modo que eles são agrupados sob um outro widget pai, herdando características do mesmo. Os próprios widgets são normalmente compostos por outros widgets menores, que ditam suas características, como cor, alinhamento, tamanho da fonte, etc. A figura 1 mostra a hierarquia dos widgets no Flutter:

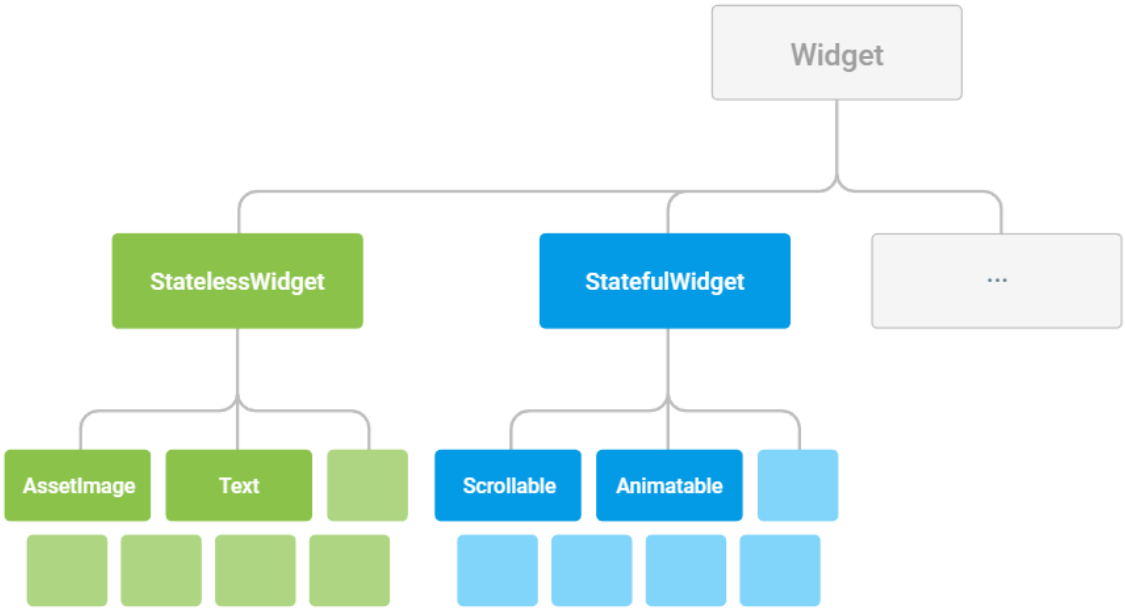

Figura 6 - Hierarquia de widgets no Flutter (Google, 2019b)

Como mostrado na Figura 1, existem dois tipos de widgets no Flutter: “Stateless Widget” e “Stateful Widget”. Os widgets do tipo “Stateless”, não modificam seu estado, ou seja, são imutáveis independentemente das ações do usuário, de modo que são indicados para construção de interfaces concretas, que não sofrerão mudanças ou alterações no decorrer da utilização. Já os do tipo “Stateful”, são passíveis de alterações e são normalmente utilizados em atividades assíncronas, que requerem interações com o usuário.

4.1.2 FIREBASE

Firebase é uma plataforma *BaaS (Backend-as-a-Service)*, atualmente pertencente à *Google*, que possui diversas ferramentas para auxílio de desenvolvimento *Web* e *mobile*. Com

ela é possível desenvolver para *iOS*, *Android* ou *Web* já que todo o *Backend* será configurado e gerenciado pelo *Firebase* (Moribe, 2016).

Um *Backend-as-a-Service* é um serviço de computação em nuvem que serve como *middleware*. O mesmo fornece aos desenvolvedores uma forma para conectar suas aplicações *mobile* e *web* a serviços na nuvem a partir de *APIs* e *SDKs* (Batschinski, 2016). Esse tipo de serviço auxilia os desenvolvedores de forma que possam se concentrar em pontos mais específicos e de maior valia ao usuário da aplicação, como o *frontend*. Isso é possível pois este tipo de serviço possibilita ao desenvolvedor a abstração da infraestrutura necessária pelo *backend*, gerando economia de trabalho e por sua vez, acelerando a construção da aplicação.

O *Firebase* possui diversos recursos que auxiliam no desenvolvimento, na qualidade e no crescimento das aplicações, dentre os quais podemos destacar:

- *Realtime Database*: Banco de dados *NoSQL* hospedado na nuvem. Os dados são armazenados como *JSON* e sincronizados em tempo real com todos os clientes conectados;
- *Firebase Hosting*: Hospedagem de conteúdo *Web* para desenvolvedores, que possibilita a implantação de *Apps* com um único comando, contando ainda com uma conexão segura (*SSL*), entrega rápida de conteúdo no mundo todo, e *rollbacks* fáceis de executar em caso de erros.
- *Firebase Crashlytics*: Ferramenta de relatório de falhas em tempo real que permite monitorar, priorizar e corrigir problemas de estabilidade no aplicativo. Permite descobrir se uma falha está afetando muitos usuários, realiza o envio de alertas quando a gravidade de um problema aumentar, e permite a descoberta de quais linhas de código estão causando os problemas.
- Monitoramento de desempenho: Serviço que fornece informação sobre as características de desempenho dos aplicativos, tais como tempo de inicialização, solicitações de rede, atividades em segundo plano, renderização de dados pela tela, etc. (Firebase, 2019).

4.1.3 BLoC

BLoC (*Business Logic Component*) é um padrão de arquitetura de projeto apresentado por Paolo Soares e Cong Hui, desenvolvedores da Google, na DartConf 2018. Tem como objetivo separar a lógica de negócio da UI (*User Interface*).

Fortemente baseado na programação reativa, o padrão BLoC utiliza de facilidades da linguagem Dart para o desenvolvimento de aplicações com separação bem definida entre a lógica de negócios, a camada de apresentação e as chamadas ao back-end.

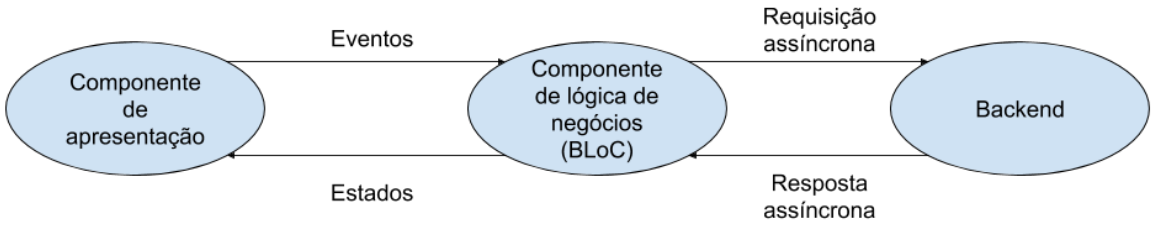

Figura 7 - Diagrama da arquitetura BLoC (Fonte: Autores)

Em Dart conseguimos implementar o padrão BLoC utilizando *Streams*. *Stream* é um fluxo contínuo de dados, que pode ser disparado assincronamente. O estado do padrão BLoC nada mais é do que um *Stream* do estado, assim podemos escutar as mudanças desse estado na *View*, observando o objeto *Stream*. Da mesma forma que o estado, os eventos também são implementados com um *Stream*, cada evento é disparado num *Stream* de eventos que é interpretado pelo objeto BLoC.

4.2 SPRINTS

Como explicado anteriormente, a metodologia ágil que guiou a criação do aplicativo foi o *Scrum*, e portanto, o desenvolvimento do projeto se dividiu em *sprints*. Cada *sprint* possui uma lista de requisitos estabelecidos previamente, com base na importância dos mesmos, tendo como foco a conclusão de um MVP já nos primeiros *sprints*. Logo que o aplicativo atingiu os requisitos mínimos esperados, foi gerado uma versão Beta e liberada para os usuários, a fim de obter *feedback* da sua utilização.

Foi acordado entre a equipe que os *Sprints* teriam a duração de duas semanas, e que as tarefas a serem desenvolvidas em cada um deles, levariam em consideração o máximo aproveitamento desse tempo.

4.2.1 SPRINT 1 (Autenticação, Integração com idUFSC, Controle de faltas, Telas iniciais)

4.2.1.1 Planejamento dos requisitos

Para o primeiro *Sprint*, foi planejado primeiramente experimentar o ambiente de desenvolvimento Flutter e a linguagem Dart no contexto da aplicação a ser desenvolvida. Ambas as tecnologias eram novidade para a equipe, por isso a necessidade de alocar algum tempo da primeira *Sprint* para realizar testes e conhecer melhor a linguagem.

Após esse primeiro momento em que a equipe utilizou todo o tempo extra para realizar estudos e adquirir conhecimento a respeito das tecnologias, foi planejado para o primeiro *Sprint*, realizar todo o processo de início de um projeto, desde a criação do mesmo, até a criação das telas principais e o link entre elas. Como esse processo é relativamente rápido, resolvemos incluir no planejamento do primeiro *Sprint* também, uma das funcionalidades que muito provavelmente seria a que demandaria mais tempo e aplicação, visto que não dependeria exclusivamente da equipe: a integração com o Acesso Unificado da UFSC.

Formalmente, os requisitos a serem atacados nesse primeiro *sprint*, inicialmente, foram:

- R1 Criar conta;
- R2 Recuperar senha;
- R3 Acessar com idUFSC (Acesso Unificado);
- R5 Acessar Home;
- R6 Acessar Perfil;
- R7 Acessar Agenda.

Os seis requisitos planejados precisaram de menos tempo do que fora previsto, de forma que ao fim da implementação de todos eles foi possível acrescentar mais um já nesse primeiro *sprint*. Este requisito adicional foi o da funcionalidade de Controle de Faltas, considerado pelos potenciais usuários como uma das principais funções do sistema. Sendo assim, o requisito "*R4 Realizar controle de faltas*", foi acrescentado aos objetivos deste sprint, totalizando sete a serem atacados.

4.2.1.2 Desenvolvimento

Para tornar possível a autenticação, primeiro foi necessário realizar a configuração do *Firebase*, que como explicado anteriormente na seção 3.1.2, foi a ferramenta escolhida para

ser responsável tanto pelo banco de dados da aplicação através do *Firebase Firestore*, quanto pelo gerenciamento da autenticação através do *Firebase Auth*.

O framework Flutter possui uma biblioteca chamada **firebase_auth**, desenvolvida pela Google, que facilita a integração com a aplicação *Firebase Auth.* Foi desenvolvida uma classe de abstração chamada "AuthRepository" que encapsula as chamadas à biblioteca **firebase_auth.**

```
class AuthRepository {
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;

  Future<FirebaseUser> get currentUser => _firebaseAuth.currentUser();

  Future<String> signIn(String email, String password) async {…

  Future<String> signUp(String email, String password) async {…

  Future<void> resetPassword(String email) async {…

  Future<void> signOut() async {…

  Future<void> sendEmailVerification() async {…

  Future<bool> isEmailVerified() async {…
}
```

Figura 8 - Classe "AuthRepository", mostrando as assinaturas dos métodos de autenticação (Fonte: Autores)

Na sequência, foram implementadas as telas de login do usuário, de criação de conta e de recuperação de senha, além das páginas principais da aplicação que são a Home, Agenda e Perfil. A tela de perfil foi modelada de forma a mostrar o e-mail do usuário logado no sistema, além do botão de "Sair", do botão "Conectar UFSC" para fazer a integração com o sistema da universidade e buscar as informações do usuário logado, além de uma opção para permitir as notificações no *smartphone*.

Foi escolhido um padrão de desenvolvimento reativo, onde as telas ficam observando as mudanças em *Streams*, assim alterando os estados quando alguma mudança é notificada. Foi adotado o padrão BLoC (*Business Logic Component*), abordado com mais detalhes anteriormente na seção 3.1.3.

4.2.1.3 Resultados

Nesse primeiro *sprint*, o maior desafio foi a implementação da integração com a UFSC. A API do CAGR fornecida pela SeTIC se mostrou bastante problemática no início,

não possibilitando a conexão e a integração com o sistema, mesmo com a autenticação funcionando e com o acesso liberado aos *endpoints* da aplicação. Mesmo após várias tentativas, não foi possível a conclusão deste requisito. Dessa forma, optou-se por fazer um mock dos dados, simulando os dados que seriam obtidos com a integração, somente para fins de teste, até o problema ser resolvido e essa parte ser finalizada com sucesso.

A funcionalidade do Controle de Faltas faz a verificação da quantidade máxima possível de faltas que o aluno pode ter em determinada disciplina cursada no semestre atual, e possibilita que o mesmo marque no próprio aplicativo quantas vezes já faltou. Essa informação é armazenada no banco de dados *Firestore*, tornando possível que o usuário se conecte em qualquer dispositivo e ainda obtenha a informação das faltas, de forma transparente. Seria possível obter a informação referente às faltas diretamente do aplicativo do Moodle, mas segundo a SeTIC, eles não possuem uma API disponível para fazer esta comunicação com o Moodle. Com essa resposta da parte deles, resolvemos optar por este caminho mais simplificado, pois seria necessário desenvolver toda a interface de comunicação primeiramente.

Mesmo com o pouco conhecimento dos desenvolvedores sobre as tecnologias utilizadas, as tarefas progrediram significativamente. Somente o requisito R3, que fora planejado para este *sprint*, não pôde ser concluído devido aos problemas encontrados. A Figura 9 mostra os principais resultados alcançados nessa etapa inicial.

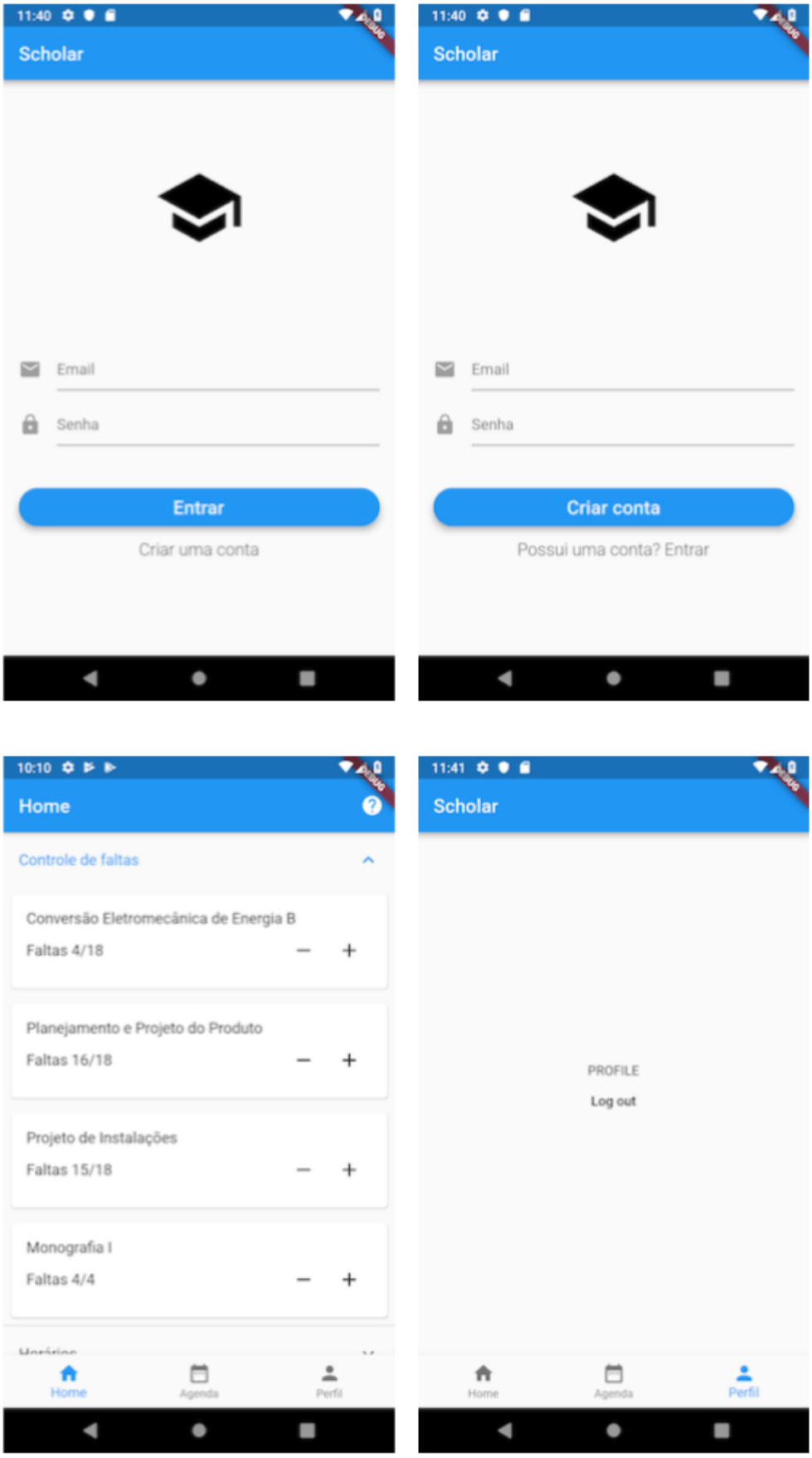

Figura 9 - Resultados obtidos no Sprint 1 (Fonte: Autores)

4.2.2 SPRINT 2 (Cardápio dos RUs, Grade de horários, Eventos, Notificações)

4.2.2.1 Planejamento dos requisitos

De modo a aproveitar melhor o tempo de duas semanas de cada *sprint*, nesta segunda etapa foi decidido abordar mais alguns requisitos importantes de forma a aproveitar melhor o tempo, levando em conta que os membros da equipe já estavam familiarizados com as tecnologias utilizadas e o bom desempenho obtido no primeiro *sprint*. Por fim, o planejamento do segundo *sprint* levou a decisão de implementar mais seis requisitos, sendo eles:

- R8 Ativar Notificações;
- R9 Visualizar grade de horários;
- R10 Visualizar cardápio do RU;
- R11 Listar eventos;
- R12 Criar e compartilhar eventos;
- R18 Receber Notificações.

4.2.2.2 Desenvolvimento

Abordando o primeiro requisito deste *sprint*, foi implementado todo o módulo de Notificações com base nas disciplinas, notificações estas que o usuário pode ativar ou desativar na tela de Perfil. Foram utilizadas duas bibliotecas para a implementação das notificações:

- *firebase_messaging*: Biblioteca da aplicação *Firebase Messaging* da Google. Utilizada para receber notificações enviadas do servidor.
- *flutter_local_notifications*: Biblioteca utilizada para gerar as notificações localmente, utilizando o relógio do sistema para disparar essas notificações.

Foi criada uma classe chamada *NotificationsService* que encapsula essas bibliotecas.

```dart
class NotificationsService {
  final _flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();
  final _payloadStreamController = StreamController<String>();
  final _firebaseMessaging = FirebaseMessaging();

  Stream<String> get payloadStream => _payloadStreamController.stream;

  NotificationsService() {…

  Future _onSelectNotification(String payload) async {…

  Future<String> get token => _firebaseMessaging.getToken();

  void scheduleWeeklyNotification({
    Time time,
    Day weekDay,
    String title,
    String content,
  }) async {…

  void dispose() {…

  void addNotifications(List<Subject> subjects) {…

  void removeAllNotifications() {…
}
```

Figura 10 - Implementação da classe NotificationsService e a assinatura de seus métodos (Fonte: Autores)

O próximo requisito implementado foi a funcionalidade de visualizar a grade de horários, conforme as disciplinas cursadas pelo usuário no semestre corrente. Para isso, é feita uma consulta diretamente à API do CAGR, de modo a obter as disciplinas que o usuário está cursando, os horários, salas onde serão as aulas, e o professor que ministra a disciplina. Em posse destes dados, é montada uma tabela, mostrando ao usuário estas informações.

```dart
Future<void> _fetchSubjects(String code) async {
  final accessUri = _buildAccessTokenUri(code);
  final response = await http.get(accessUri);
  final accessToken = _parseAccessToken(response.body);
  final timeGrid = await _performGet(_userTimeGrid, accessToken);
  final subjects = decodeSubjects(timeGrid);
  _userData.replaceSubjects(subjects);
  final schedules = await _userData.schedules;
  final settings = await _userData.settings;
  if (settings.allowNotifications) {
    _notifications.addNotifications(schedules);
  }
  await _userData.saveSettings(
    settings.rebuild((b) => b
      ..restaurantId = 'trindade'
      ..accessToken = accessToken
      ..connected = true),
  );
}
```

Figura 11 - Método _fetchSubjects da classe UfscService (Fonte: Autores)

```dart
const _apiUrl = 'https://ws.ufsc.br/rest/CAGRUsuarioService/';
const _userTimeGrid = 'getGradeHorarioAluno';
```

Figura 12 - Inicialização das variáveis _apiUrl e _userTimeGrid (Fonte: Autores)

Como podemos ver nas Figuras 11 e 12, o método *_fetchSubjects* faz uma requisição HTTP utilizando as variáveis *_apiUrl* e *_userTimeGrid* para buscar a grade de horários do aluno. Após finalizada a requisição a função *decodeSubjects* converte o resultado JSON para uma lista de objetos *Subject*.

A visualização dos cardápios dos restaurantes, precisou superar alguns problemas, visto que cada Campus da UFSC possui um RU, e cada um deles apresenta os cardápios de forma diferente. Alguns são páginas HTML normais, outros são arquivos PDF, etc. Para tornar a visualização padrão no aplicativo, foi necessário a criação de *parses*. Um *parse* é basicamente um tradutor, que converte uma informação apresentada, para algum tipo desejado, visando um padrão. Com os *parses* implementados, foi possível buscar, traduzir e

padronizar os cardápios dos diferentes restaurantes universitários, para apresentar no aplicativo conforme a seleção do usuário. Além da criação dos *parses* foi necessário implementar também a tela que apresenta os cardápios, de uma forma legível, fácil de navegar e fácil de entender.

```
async function parseRUTrindade() {
    const { document } = (await JSDOM.fromURL("http://ru.ufsc.br/ru/")).window
    const rows = Array.from(document.querySelector('table').querySelectorAll('tr'))
    const getCellContent = (tr) => Array.from(tr.querySelectorAll('td'))
        .flatMap((td) => td.innerHTML.replace(/<p><\/p>/, '\n')
            .replace(/(<\/?p>)|(<\/?strong>)/g, '').replace(/ /, ' ').split('\n'))
    const contentToDayAndPlates = ([date, ...plates]) => ({
        date: moment(date, 'DD/MM').toDate(),
        plates: plates.flatMap(i => i.split('/').map(p => p.trim())).filter(i => i !== '')
    })
    const menu = rows.map(getCellContent).map((c) => contentToDayAndPlates(c.slice(2)))
    await saveMenu({ menu }, 'trindade', 'Campus Trindade')
}
```

Figura 13 - Função parseRUTrindade (Fonte: Autores)

Como podemos ver no exemplo da Figura 13 o restaurante do campus Trindade possui um cardápio no formato HTML na página "http://ru.ufsc.br/ru/". Para obtermos as informações do cardápio foi necessário navegar pelo DOM da página e extrair os dados desejados.

Finalmente, os últimos requisitos a serem implementados neste *sprint* foram a listagem de eventos e a criação e compartilhamento destes eventos. Ambas estas funcionalidades são interligadas e fazem uso da funcionalidade de Notificações para avisar o usuário quando algum evento se aproxima.

4.2.2.3 Resultados

As notificações foram implementadas com sucesso nesta sprint e estão fortemente ligadas à tela de Agenda e à grade de horários do usuário. Quando um novo evento é cadastrado em uma disciplina cursada pelo usuário, uma notificação é enviada avisando sobre aquele evento. Além disso, foi implementada também a notificação com base nos horários das aulas, que avisa o usuário quando uma aula está prestes a começar. Esse tempo por padrão é de dez minutos antes do início da aula, mas pode ser configurado pelo usuário como bem entender. Todas essas notificações podem ser ativadas ou desativadas, conforme especificado no sprint anterior, na tela de Perfil do usuário.

Todo o módulo de eventos foi criado neste *sprint*, e é utilizado através da tela Agenda. Com esse módulo, é possível que o usuário cadastre um novo evento em uma determinada disciplina, tal como a data de entrega de um trabalho ou a data de uma prova, e compartilhe esse evento com os outros usuários que também estão cursando aquela determinada disciplina naquele semestre. O Moodle, como mencionado na seção 3.1.3, já registra eventos e os disponibiliza, porém devido a falta de API disponível para integração, foi necessário criar um módulo independente.

Através da alimentação das informações por parte dos usuários, é possível que os calendários de cada disciplina estejam sempre atualizados. Da mesma forma, é possível simplesmente visualizar a agenda e verificar os eventos já criados com base nas diferentes disciplinas cursadas.

A grade de horários das aulas cursadas pelo usuário no semestre atual também foi implementada com sucesso, permitindo ao usuário que consulte suas aulas sempre que quiser de forma rápida e simples. De forma semelhante, o cardápio dos restaurantes universitários também foi implementado e também está disponível no aplicativo, sempre atualizado para facilitar a vida do estudante que utiliza os RUs. Nesse primeiro momento, somente o cardápio do restaurante do campus Trindade foi concluído. Os cardápios dos demais campus serão implementados em uma *sprint* futura.

Essa *sprint* também teve a duração de duas semanas e não foi constatado nenhum problema no desenvolvimento das funcionalidades, visto que a integração via idUFSC já estava funcionando corretamente, possibilitando a obtenção de todos os dados necessários.

O resultado no aplicativo, proveniente desta *sprint*, pode ser observado na Figura 14.

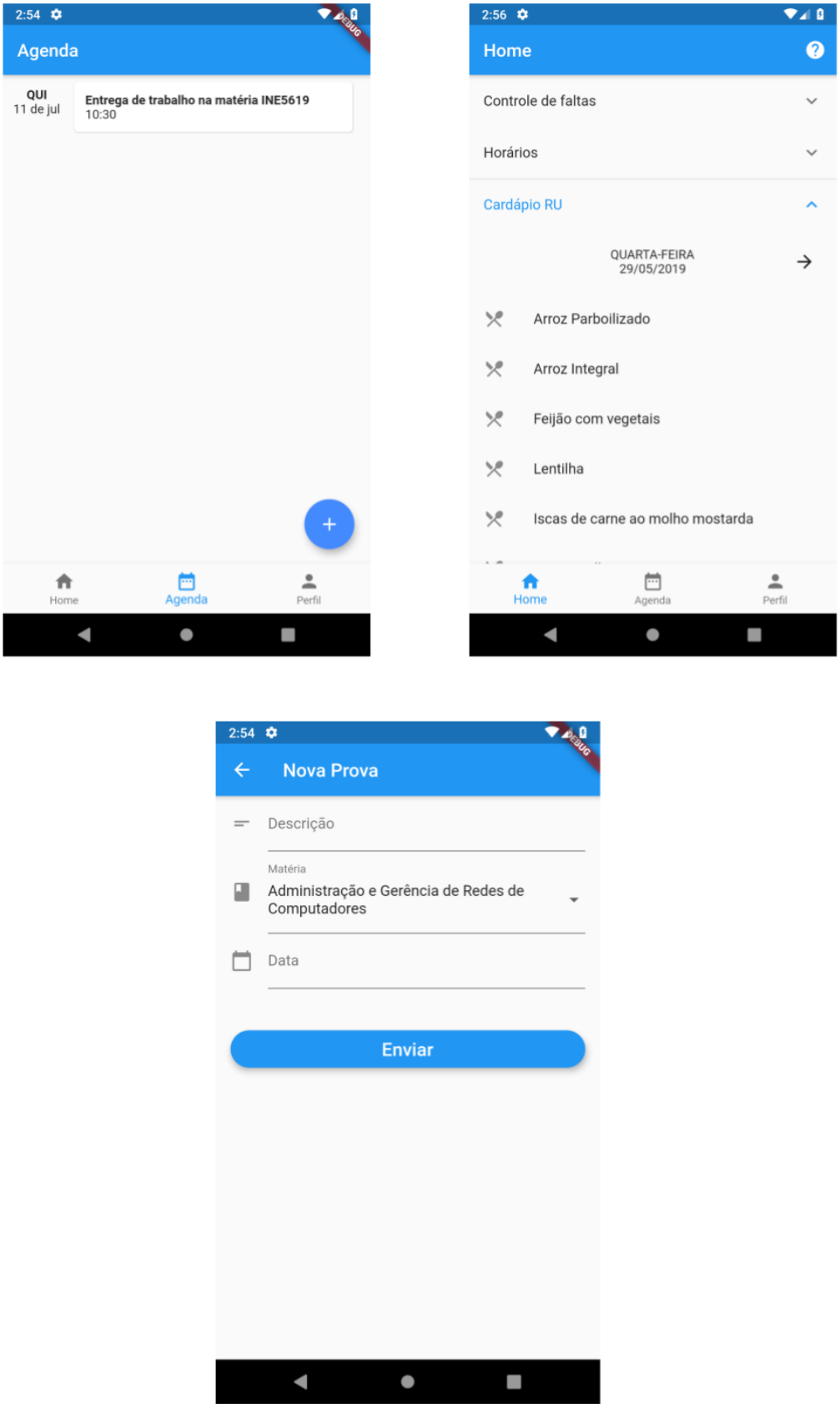

Figura 14 - Resultados obtidos na segunda sprint (Fonte: Autores)

4.2.3 SPRINT 3 (Integração com idUFSC, Conclusão do MVP, Lançamento da versão Android, Manutenções corretivas)

4.2.3.1 Planejamento dos requisitos

Os objetivos do terceiro sprint foram de continuar o desenvolvimento da aplicação, visando a finalização de um *MVP* (*Minimum Viable Product*), com o objetivo de lançar essa versão mínima do aplicativo para que alguns poucos usuários pudessem utilizá-lo e fornecer *feedback* à equipe.

O requisito *R3 Acessar com idUFSC (Acesso unificado)* voltou a ser atacado neste *sprint*, sendo o principal foco. Visto que não foi possível concluí-lo no *Sprint* 1 por dificuldades técnicas enfrentadas.

O lançamento da versão do sistema, para dispositivos Android, também é um dos objetivos desta *sprint*, visto que ao final dele é esperado a conclusão de um MVP, como mencionado anteriormente. Desta forma, o lançamento seria possível. Neste primeiro momento, a aplicação seria liberada somente a alguns usuários selecionados, através de um link privado da Google Play Store, como forma de obter *feedback* dos usuários acerca das funcionalidades já implementadas.

Antes do início desta *sprint*, foi constatado que o *script* desenvolvido para obter e traduzir as informações contidas nas páginas dos RUs também precisava de alguns ajustes, visto que em alguns casos não estava conseguindo realizar a tradução, o que fazia com que a opção “Cardápio RU” desaparecesse da tela *Home*. Este, e outros problemas identificados em testes internos da equipe, também foram foco deste *sprint*.

4.2.3.2 Desenvolvimento

As tarefas de desenvolvimento planejadas para este *sprint*, obtidas a partir dos objetivos elencados, foram:

- Finalizar *script* de coleta de dados do RU;
- Integração com a API da SETIC;
- Corrigir a tela de grade de horários;
- Corrigir a tela de cardápios;
- Configurar o app na Google Play Store;
- Fazer o lançamento do app para um grupo controlado de usuários.

Novamente, os esforços de desenvolvimento foram voltados para a parte da integração com o Acesso Unificado da UFSC, visto que as demais tarefas eram relativamente

simples. Após várias tentativas e diferentes abordagens para tentar solucionar o problema, as suspeitas de que algo externo estava impedindo a conexão foram confirmadas, após abertura de um chamado com a SeTIC.

O *script* de tradução dos cardápios também foi melhorado nesta *sprint*. Além do erro mencionado anteriormente ter sido corrigido, agora é possível selecionar o cardápio de qualquer campus, bastando selecionar qual deseja através da tela “Perfil”.

4.2.3.3 Resultados

Após alguns dias de contato frequente com a SeTIC, descobriu-se que houvera um problema interno com a API, após uma atualização realizada por eles, que impossibilitava a conexão e a obtenção dos dados dos alunos. Mesmo com toda a estrutura do aplicativo pronta para receber estes dados, novamente não foi possível concluir o requisito no tempo deste *sprint*.

As demais tarefas planejadas para desenvolvimento foram todas concluídas. Porém, devido a impossibilidade de conexão devido ao problema listado, o lançamento do aplicativo teve que ser adiado, até que o problema fosse solucionado, o que independia da equipe. Sendo assim, os objetivos “Integração com a API da SETIC” e “Fazer o lançamento do app” não puderam ser concluídos.

A Figura 15 mostra os resultados obtidos ao fim desta *sprint.*

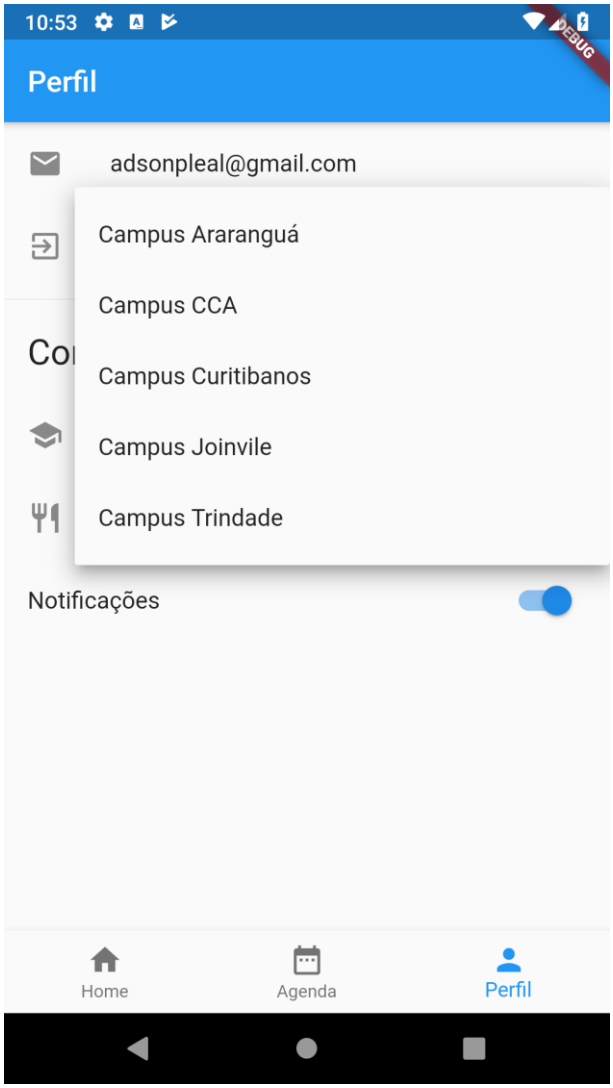

Figura 15 - Tela de seleção de campus (Fonte: Autores)

4.2.4 SPRINT 4 (Integração com idUFSC, Lançamento da versão Android)

4.2.4.1 Planejamento de requisitos

Nesta quarta *sprint*, o foco foi novamente a integração com a API da SeTIC e a preparação para o lançamento de uma versão do app, que eram requisitos da terceira *sprint*, porém não puderam ser concluídos devido a problemas técnicos.

Como já mencionado anteriormente, inicialmente o aplicativo só estará disponível para o sistema Android, visto que a liberação de aplicativos neste sistema é muito mais simples e barata se comparada a aplicativos para iOS, cuja licença para publicar aplicativos na loja custa um valor considerável. A possibilidade de divulgação do App também para dispositivos iOS não foi descartada, e pode vir a acontecer em um momento futuro, caso seja constatado que há essa demanda por parte dos usuários.

Os objetivos desta *sprint* então, por fim, foram: “Concluir integração com API da SeTIC” e “Lançamento da versão Android”.

4.2.4.2 Desenvolvimento

Após a solução do problema por parte da SeTIC com a atualização da API, finalmente foi possível concluir a integração com o acesso unificado da UFSC para obter os dados acadêmicos dos alunos. Como toda a estrutura já estava pronta e somente aguardando a correção, este requisito levou pouco tempo para ser implementado, o que favoreceu o desenvolvimento da segunda parte da *sprint*.

Para realizar o lançamento do aplicativo, primeiro foram necessárias algumas configurações específicas nas ferramentas que seriam utilizadas. Primeiramente o aplicativo precisou ser configurado na loja da Google Play para poder ser lançado, sendo primeiramente necessário definir alguns parâmetros como qual seria a versão do lançamento, qual seria o público alvo, além de criar toda a página do aplicativo na loja com descrições e prints das telas, o que é obrigatório. Após a configuração da loja, foi necessário configurar o ambiente de CI (*Continuous Integration*), o que facilita e muito o lançamento de atualizações e novas versões do App. O ambiente de CI inicia uma tarefa (*Job*) após verificar alterações no repositório GIT, essa tarefa roda os testes, verifica possíveis erros de compilação e gera uma nova versão do app. Essa versão pode ser disponibilizada na loja diretamente do painel da aplicação no Google Play.

Por fim, as tarefas identificadas e realizadas nesta sprint foram:

- Configurar o aplicativo na Google Play Store;
- Configurar o ambiente de CI;
- Finalizar integração com a API da SeTIC.

4.2.4.3 Resultados

Após resolvido o problema por parte da superintendência, a autenticação e integração com a UFSC finalmente começou a funcionar, possibilitando a busca das matérias cursadas pelo usuário a fim de implementar praticamente todas as funcionalidades da aplicação, inclusive o Controle de Faltas.

A página da loja do aplicativo e toda a configuração necessária para o lançamento da primeira versão do App também foram concluídas com sucesso. Um ponto importante a ressaltar foi que nesse momento, após algumas discussões prévias a respeito, finalmente foi decidido o nome do aplicativo, de forma oficial. Decidiu-se por chamá-lo de “Scholar”,

devido ao aspecto acadêmico do aplicativo. A logo também foi finalizada neste Sprint, após refinamento de algumas ideias prévias. A página da loja do aplicativo pode ser visualizada logo abaixo na Figura 16.

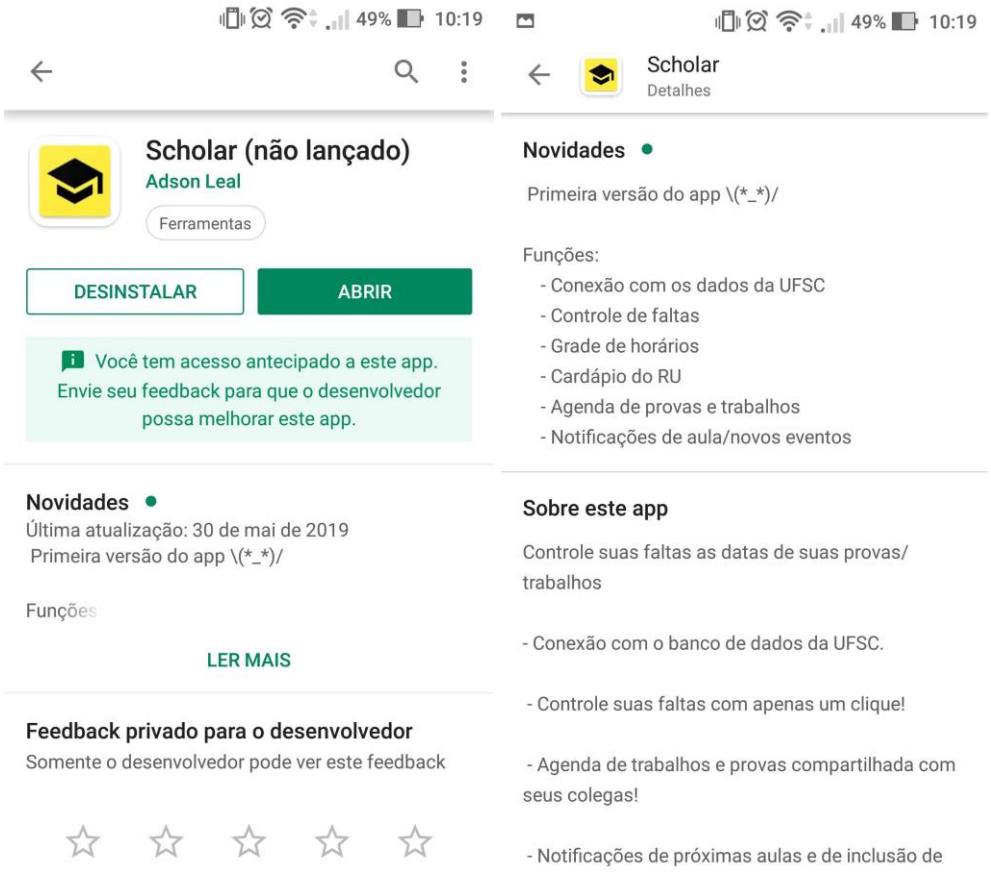

Figura 16 - Página da Google Play do aplicativo Scholar (Fonte: Autores)

4.2.5 SPRINT 5 (Visualizar detalhes de eventos, Editar ou remover eventos, Manutenções corretivas)

4.2.5.1 Planejamento de requisitos

Para a *sprint* de número cinco, foi planejado atacar mais dois requisitos principais do projeto, sendo eles:

- R13 Editar ou remover eventos;
- R14 Visualizar detalhes de eventos.

Esses requisitos foram incluídos no projeto depois do *feedback* recebido após os primeiros testes realizados pelos usuários selecionados, após o lançamento do App ao final da sprint anterior. De acordo com o *feedback* obtido, é desejado que o usuário possa clicar sobre um evento criado ou compartilhado e visualizar os detalhes do mesmo em uma outra janela, mostrando mais claramente as informações.

É desejável também que se possa editar um evento, modificando seu nome, disciplina ou data de entrega, caso haja algum erro na hora da criação do mesmo, ou alguma mudança na data estipulada por exemplo. Junto com a possibilidade de edição, foi planejado também a remoção de eventos, em caso de cancelamento por parte do professor, ou após o usuário aceitar o compartilhamento de um evento indesejado.

Com o lançamento do aplicativo para teste controlado na *sprint* anterior, algumas solicitações de melhoria foram feitas. Após analisá-las, resolveu-se separar as mais relevantes para incluir nesta *sprint*, além do desenvolvimento dos requisitos. Entre as solicitações de melhoria apontadas, as selecionadas foram:

- Na tela Home, mudar o nome da função “Horários” para “Grade de horários”;
- Acrescentar código da sala onde ocorrem as aulas, bem como o professor que ministra;
- Na Agenda, alterar a exibição dos eventos para que mostre o nome da disciplina ao invés do código;
- Adicionar a opção de poder voltar nos dias da semana no cardápio, para visualizar cardápios de dias passados na semana.

Além das melhorias, foi detectado um *bug* na data das notificações que, no caso de matérias com mais de uma aula por dia, enviava apenas uma notificação de aviso de aula para a segunda aula.

4.2.5.2 Desenvolvimento

Seguindo o planejamento dos requisitos feitos para essa *sprint*, pode-se desmembrar os requisitos estabelecidos em algumas tarefas principais de desenvolvimento, sendo elas:

- Implementar a tela de detalhes dos eventos;
- Implementar a remoção de eventos;
- Implementar a edição de eventos;
- Implementar melhorias selecionadas no planejamento.

Para implementar a remoção de eventos utilizamos a função de remoção do firebase, quando removemos um evento de um usuário os eventos criados a partir dele, por meio do compartilhamento de eventos da turma, não são removidos. Isso se deve ao fato de nosso banco de dados ser um banco não relacional, o que significa que cada usuário mantém uma cópia completa do seus próprios dados, os mesmos não estão referenciados por uma ligação com chave privada, como acontece num banco relacional. Assim podemos remover dados de um usuário sem se preocupar com o efeito que isso acarreta nos outros usuário.

Para implementação da tela de detalhes dos eventos, não houve complicações, visto que as informações apresentadas são inseridas pelo usuário na hora do cadastro. Bastou que fosse montada a tela e que essas informações cadastradas previamente fossem apresentadas, após o usuário clicar sobre um determinado evento.

Foi decidido que as funcionalidades de edição e de exclusão de um evento, deveriam estar na própria tela de detalhes dos mesmos. Sendo assim, foram acrescentados dois ícones no canto superior direito da tela, um sendo para exclusão do evento caso seja desejado, e outro para edição, para o usuário alterar alguma informação cadastrada na hora da criação, como a descrição do evento ou a data.

As melhorias abordadas foram simples de solucionar, visto que a maioria dizia respeito a alteração nas *labels*. Quanto a correção do *bug* das notificação, o problema foi identificado e logo corrigido. Tratava-se de um problema de sincronização com a data onde o servidor da aplicação está hospedado. Após ajustar isto, foi solucionado o problema.

4.2.5.3 Resultados

Ao fim da *sprint*, foi possível concluir todos os requisitos planejados inicialmente. A tela de detalhes de eventos, após implementada, já foi integrada ao código do aplicativo, juntamente com a possibilidade de remover um evento. O ícone para remoção do evento, após uma análise realizada pela equipe e uma conversa com alguns usuários, foi colocado na própria tela de detalhes, tornando mais limpo o design da tela “Agenda”, sem incluir novas funcionalidades nela, já que o intuito principal da mesma é apenas visualização. Logo abaixo na Figura 17 está um print da nova tela de detalhes implementada.

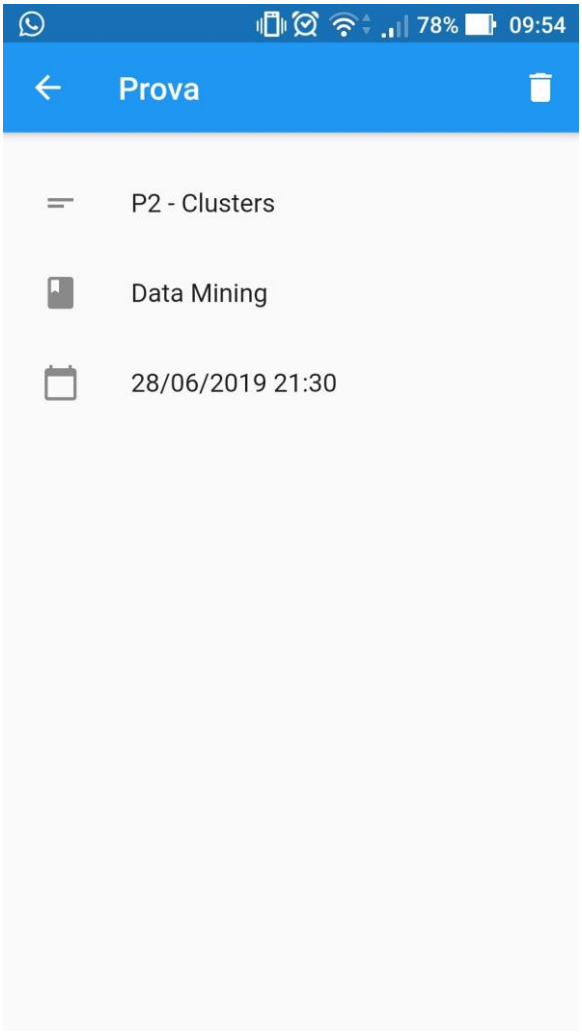

Figura 17 - Tela de detalhes de eventos com ícone de exclusão (Fonte: Autores)

As demais sugestões, que na maioria foram alterações de *labels*, também foram corrigidas sem problemas nessa *sprint*, bem como o *bug* detectado na data das notificações, que também foi corrigido nesta *sprint*. Após essas correções, uma nova *build* foi gerada para o aplicativo e foi disponibilizada uma atualização na Google Play Store, já com as correções e novas funcionalidades integradas, bastando o usuário atualizar seu aplicativo para ter acesso a estas.

4.2.6 SPRINT 6 (Histórico síntese, Atestado de matrícula, Currículo do curso, Melhoria nas notificações, Manutenções corretivas)

4.2.6.1 Planejamento de requisitos

Finalmente a última *sprint* planejada do projeto, englobando todos os requisitos restantes, além de algumas melhorias estudadas e de algumas correções de problemas

identificados. Para esta sprint, foi planejado atacar os seguintes requisitos, que ainda não haviam sido implementados:

- R15 Visualizar histórico síntese;
- R16 Fazer download do atestado de matrícula;
- R17 Fazer download do currículo do curso.

Além destes requisitos finais, foi identificada uma melhoria que era desejada pelos potenciais usuários. Segundo o *feedback* recebido, é desejado que o sistema envie notificações não somente para aulas que estão prestes a começar, mas também para trabalhos e provas que se aproximam. Desse modo, decidiu-se implementar três níveis de notificações referentes a provas e trabalhos, apresentando a notificação no *smartphone* do usuário com a mensagem "*Falta/m X dia/s para a/o prova/trabalho #título do evento conforme Agenda#*". A mensagem muda conforme o nível da notificação e o tipo de evento a qual se refere, da seguinte forma:

- Primeira notificação: Sete dias antes do evento;
- Segunda notificação: Três dias antes do evento;
- Terceira e última notificação: Um dia antes do evento.

Dessa forma, o aluno pode ser lembrado sobre a data da realização de uma prova ou da entrega de um trabalho, minimizando as chances de que ocorra um esquecimento por parte do aluno que por vezes acaba fazendo com que comece a estudar para a prova, ou a trabalhar no trabalho proposto, muito em cima da hora, gerando resultados ruins.

Além da melhoria nas notificações, foi identificado um *bug* na seleção das datas dos eventos, onde ao cadastrar um evento para a data atual, o sistema mostrava uma data e hora completamente diferentes ao realizar este cadastro. A correção deste *bug* também entrou no escopo desta *sprint* final.

4.2.6.2 Desenvolvimento

Seguindo o planejamento da *sprint*, pode-se dividir os requisitos planejados nas seguintes tarefas de desenvolvimento:

- Implementar visualização do histórico síntese do aluno, adicionando um novo botão na tela *Home*;
- Implementar o download do atestado de matrícula, adicionando um novo botão na tela *Home*;
- Implementar o download do currículo do curso, adicionando um novo botão na tela *Home*;

- Implementar e testar os três níveis diferentes das notificações;
- Corrigir *bug* com a seleção da data atual nos eventos;

Os três requisitos principais foram simples de realizar, graças ao acesso à API do CAGR. Existem três *endpoints* que são responsáveis exatamente por estas funcionalidades, bastando que o usuário anteriormente faça a conexão com o sistema da UFSC através da tela "Perfil". Dessa forma, a disponibilização das três novas informações se torna possível.

4.2.6.3 Resultados

A apresentação das três novas informações, segundo os requisitos R15, R16 e R17, foram bastante simples de resolver, visto que com o acesso possibilitado aos devidos *endpoints* do CAGR, bastou conectá-los ao aplicativo e fazer a criação dos novos botões na tela de *Home*. Sendo assim, quando o usuário desejar acessar alguma dessas informações, basta que esteja conectado ao sistema da UFSC, para que as opções sejam mostradas na tela inicial, a partir daí, basta clicar no botão desejado e a informação está disponível normalmente.

Os níveis de notificações foram um pouco mais complicados de se resolver, porque deveriam levar em conta o compartilhamento dos eventos entre os usuários. Ao aceitar a inclusão de um novo evento cadastrado por outro aluno, este evento passa a constar na agenda do usuário conectado ao aplicativo. Porém, não foi ele o criador do evento. Foi preciso tratar esta situação para que mesmo não sendo o usuário criados, assim que um evento entrasse na agenda, essas notificações passassem a entrar em vigor também para aquele evento.

Em síntese, apesar das dificuldades encontradas, todos os requisitos planejados para esta *sprint* final foram implementados com sucesso dentro do tempo da mesma. Após a conclusão da implementação, uma nova *build* do aplicativo foi gerada e enviada para a Google Play Store na forma de uma nova atualização. Para ter acesso às correções e as novas funcionalidades, bastou que os usuários realizassem a atualização de seus aplicativos.

4.2.7 SPRINT 7 (Versão Web)

4.2.7.1 Planejamento de requisitos

Logo após a finalização da Sprint 6, que foi a última *sprint* que adiciona funcionalidades ao projeto, a Google liberou a versão do Flutter que possibilita a criação de

aplicativos Web com a mesma base de código dos aplicativos mobile. Sendo assim, optou-se por fazer mais uma *sprint* “bônus”.

Muitos usuários estavam pedindo a versão iOS da nossa aplicação, mas pelo caráter acadêmico do nosso trabalho, sem fim lucrativo, nós decidimos criar a versão web no lugar de lançar a versão iOS. O maior empecilho ao lançamento da versão iOS foi o custo de manter o app na loja, que é de 99 dólares por ano. Como esse valor foge do nosso orçamento para esse trabalho nós optamos pela versão Web, que pode ser acessada por qualquer dispositivo independente de sistema operacional.

4.2.7.2 Desenvolvimento

O desenvolvimento dessa etapa do trabalho foi relativamente fácil, toda a parte do código referente às telas não necessitou alterações para a versão Web. A única alteração necessária foi na parte do Firebase, por algum motivo a API da versão Web da biblioteca do Firebase é diferente da versão mobile. Então tivemos que alterar algumas classes da camada de serviços da nossa aplicação. Como implementamos o princípio da responsabilidade única para as nossas classes não tivemos dificuldade na realização dessa tarefa.

Abaixo nas Figuras 18 e 19, estão alguns exemplo das alterações que foram necessárias.

```
Future<fs.DocumentReference> get userDocument async {
  final user = await _auth.currentUser;
  return _store.collection('users').doc(user?.uid);
}
```

Figura 18 - Código para buscar os dados do usuário logado na versão web (Fonte: Autores)

```
Future<DocumentReference> get userDocument async {
  final user = await _auth.currentUser;
  return _store.collection('users').document(user?.uid);
}
```

Figura 19 - Código para buscar os dados do usuário logado na versão mobile (Fonte: Autores)

Como podemos ver nas Figuras 18, o método para buscar o documento de uma coleção na versão *Web* da biblioteca do *firebase* é "*doc*(<ID_DOCUMENTO>)" já na versão *mobile* (Figura 19) é "*document*(<ID_DOCUMENTO>)". Esse mesmo problema também acontece com outros métodos dessa biblioteca. Como a versão *Web* do *Flutter* é muito

recente é comum encontrarmos esse tipo de problema pois o compilador do *Dart* para *Web* na verdade é um transpilador que converte o código *Dart* em *Javascript*, assim as bibliotecas *Web* para *Dart* são fortemente amarradas a suas versões *Javascript*, o que pode gerar algumas inconsistências com suas versões *mobile*.

Outra mudança necessária foi na parte de autenticação do usuário com o serviço da UFSC. Quando o usuário se conecta com a UFSC e autoriza nossa aplicação um "*redirect_uri*" é chamado, esse URI para a versão *mobile* é "*tccleal://tccleal.setic_oauth.ufsc.br*" já para a versão *WEB* precisamos que ele retorne para o próprio *browser* por isso alteramos o URL de retorno para "*https://web.app-scholar.xyz*". Para fazer essa alteração não basta alterar no lado da aplicação, precisamos dessa mudança também na parte do serviço da UFSC, por isso abrimos um ticket com a SETIC requisitando essa mudança. Até a data da finalização deste trabalho não obtivemos resposta a essa solicitação.

Após essas pequenas mudanças para adequar o código à versão *Web* foi necessário disponibilizar a aplicação num serviço de hospedagem de sites para possibilitar o acesso aos nosso usuários. Coincidentemente o *Firebase* também disponibiliza esse serviço, assim conseguimos manter toda nossa aplicação na mesma plataforma.

O Firebase disponibiliza um *endpoint* padrão após a configuração do serviço de hospedagem, mas como esse *endpoint* não é amigável nós decidimos comprar o domínio "app-scholar.xyz". Decidimos manter a página "*app-scholar.xyz*" com os links para a loja de aplicativos e a página "*web.app-scholar.xyz*".

4.2.7.3 Resultados

A versão Web é idêntica à versão *mobile*, a única diferença é o tipo de acesso que não necessita a instalação de um aplicativo no dispositivo do usuário. Qualquer pessoa pode acessar a aplicação diretamente pelo endereço "*web.app-scholar.xyz*".

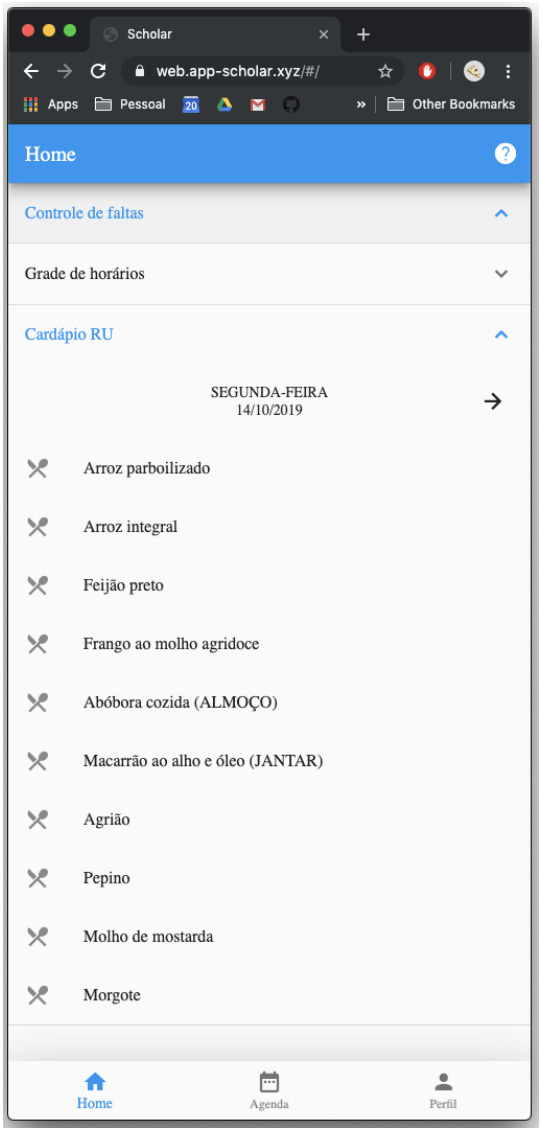

Figura 20 - Aplicação rodando diretamente em um browser (Fonte: Autores)

Na Figura 20 a aplicação está rodando em um *browser* Chrome em uma máquina com MacOs. A versão *Web* é mais acessível do que a versão *mobile*, que atualmente está disponível apenas para dispositivos Android.

Até a data da finalização deste trabalho não divulgamos a versão *Web* para nossa base de usuários pois eles não conseguiriam conectar suas contas da UFSC com nossa aplicação. Necessitamos de uma pequena alteração por parte da SETIC que ainda não foi disponibilizada.

5 RESULTADOS OBTIDOS

Ao final das seis sprints realizadas no decorrer do trabalho, obteve-se um aplicativo totalmente funcional cumprindo todos os requisitos que foram estabelecidos previamente. Decidiu-se por nomear o aplicativo de “Scholar”, que remete a assuntos acadêmicos, já que o objetivo do aplicativo é reunir as informações acadêmicas do aluno conectado.

Após a conclusão do aplicativo, o diagrama de classes do mesmo ficou da seguinte maneira:

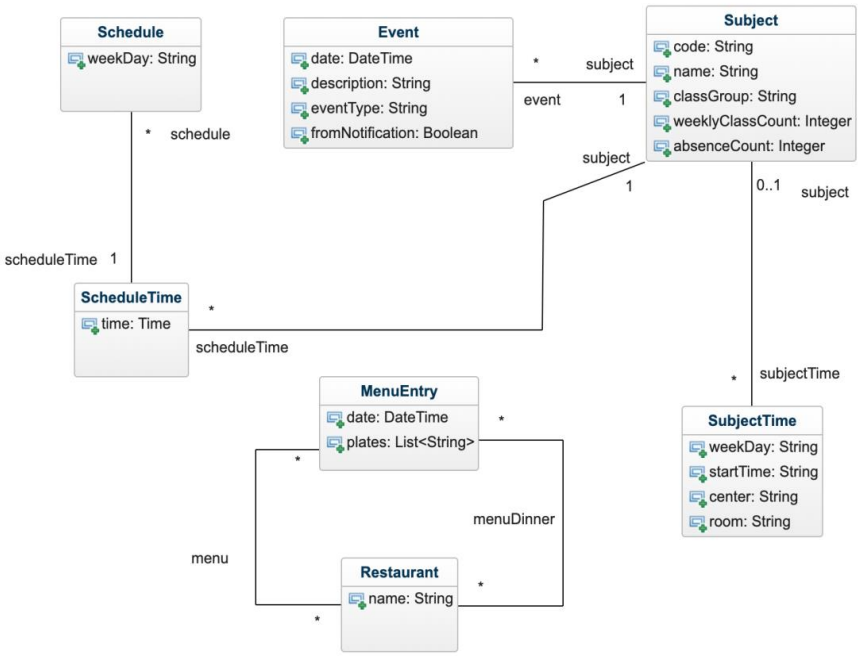

Figura 21 - Diagrama de classes (Fonte: Autores)

O Scholar, na sua versão concluída, possui várias funcionalidades que antes só podiam ser acessadas através do site do CAGR. A API de integração fornecida pela SeTIC possibilitou a extração e apresentação destes dados diretamente no aplicativo. Além disso, o aplicativo também apresenta os cardápios dos RUs dos diferentes campus, sendo esta uma das funcionalidades mais solicitadas pelos usuários.

Nas Figuras 22 à 28, podemos ver *prints* da tela de um *smartphone* rodando o aplicativo, mostrando algumas das principais telas:

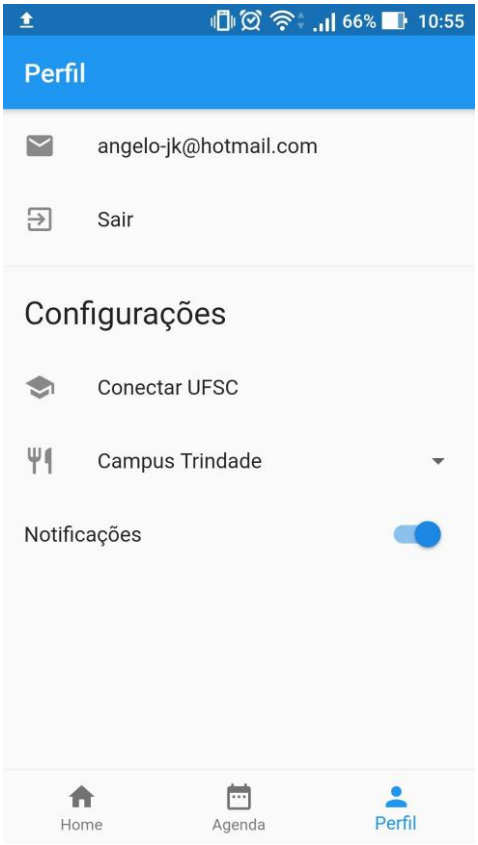

Figura 22 - Perfil (Fonte: Autores)

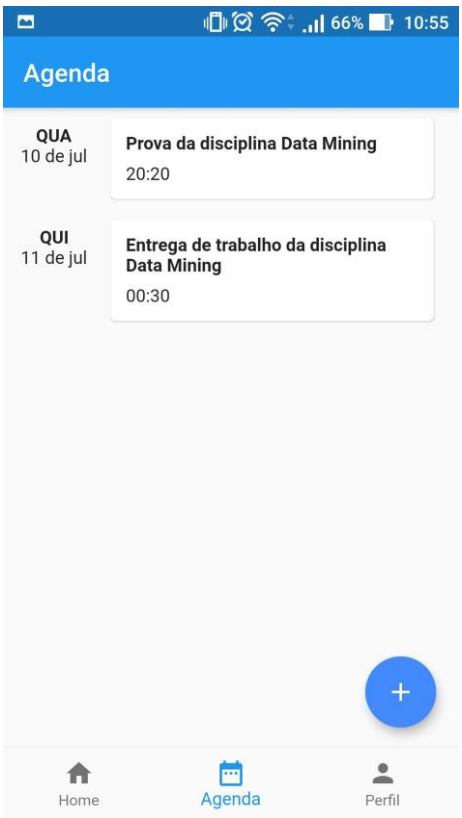

Figura 23 - Agenda (Fonte: Autores)

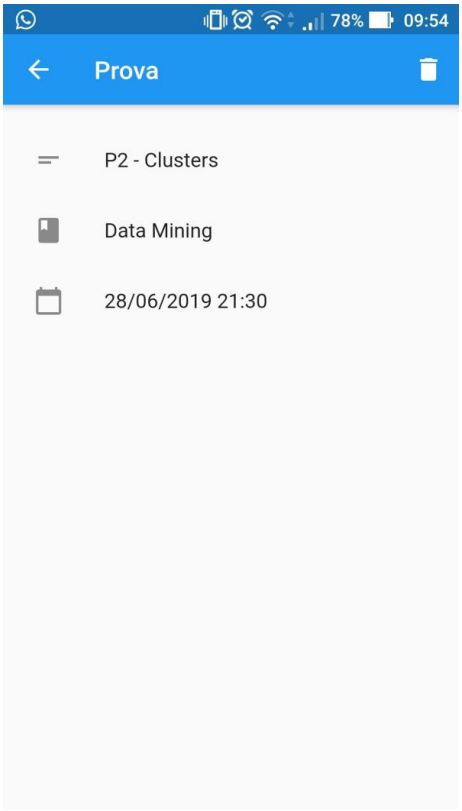

Figura 24 - Detalhes de eventos (Fonte: Autores)

Figura 25 - Controle de faltas (Fonte: Autores)

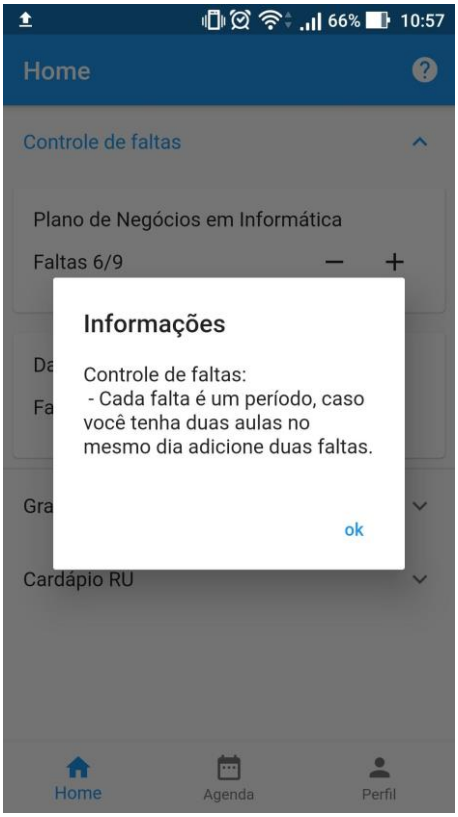

Figura 26 - Ajuda no controle de faltas (Fonte: Autores)

Figura 27 - Grade de horários (Fonte: Autores)

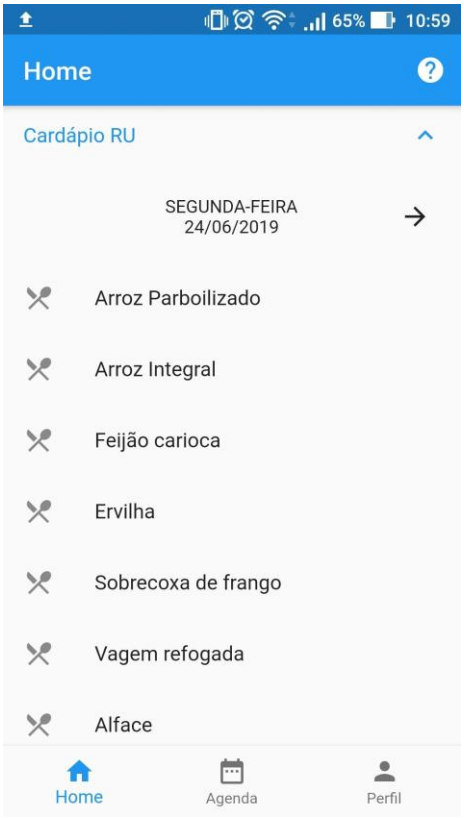

Figura 28 - Cardápio (Fonte: Autores)

5.1 PESQUISA DE SATISFAÇÃO

Após o lançamento do aplicativo, foi feita a divulgação do mesmo em algumas listas de e-mail da universidade. Depois de algumas semanas de aplicativo publicado, decidimos enviar um formulário de avaliação, opcional, para os usuários através da ferramenta de formulários do Google. O questionário consiste em 10 perguntas, sendo 9 questões objetivas e uma questão aberta sobre possíveis melhorias no aplicativo. A transcrição para o formulário, bem como o link de acesso ao mesmo, encontram-se no fim deste trabalho no Apêndice B.

Foi tomado o cuidado de mencionar o atual problema relacionado a SeTIC, que impede que novos usuários utilizem o aplicativo devido aos problemas de conexão com a API, solicitando que somente quem não estivesse enfrentando este problema respondesse este formulário. O mesmo foi enviado para toda a base de usuários e conseguimos um total de 45 respondentes. A seguir analisamos as respostas deste questionário.

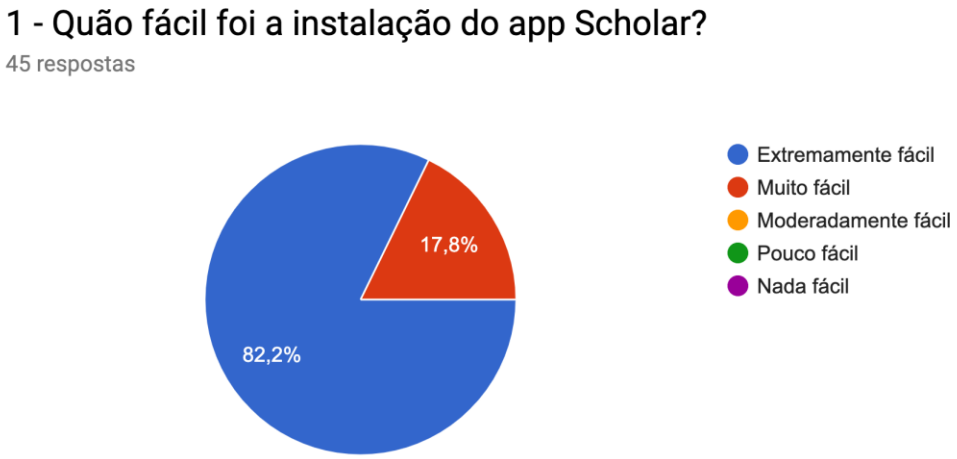

Figura 29 - Respostas da primeira pergunta do questionário de satisfação (Fonte: Autores)

Como podemos ver na Figura 29 a grande maioria dos usuários respondentes acredita que a instalação do aplicativo foi extremamente fácil. Acreditamos que esse resultado positivo se dá ao fato de utilizarmos a loja oficial de aplicativo do Google, a Play Store, para distribuir nossa aplicação. A maioria dos usuários já está acostumada a instalar aplicativos diretamente da loja de aplicativos.

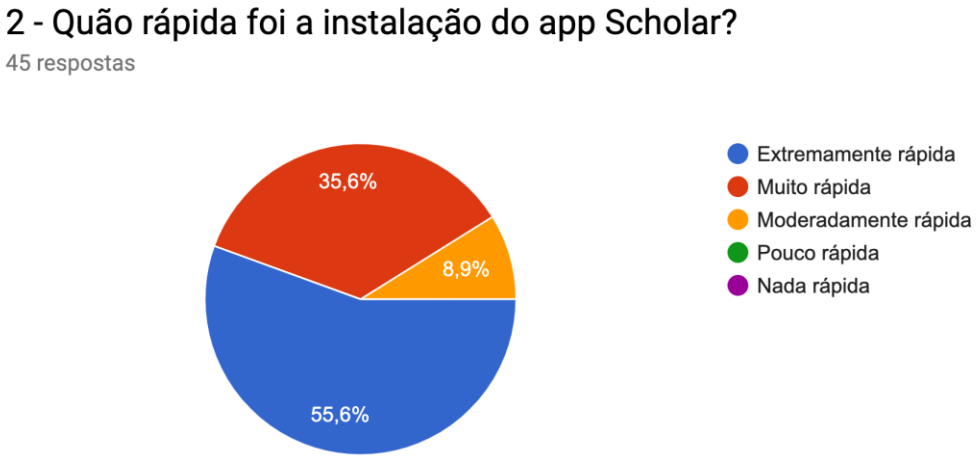

Figura 30 - Respostas da segunda pergunta do questionário de satisfação (Fonte: Autores)

Como podemos ver na Figura 30, 55.6% dos usuários respondentes acredita que a instalação do aplicativo foi extremamente rápida, 35,6% acredita que foi muito rápida e apenas 8,9% acredita que foi moderadamente rápida. Esse bom resultado se dá devido aos

nossos esforços de manter baixo o tamanho da aplicação, que após lançada ficou em torno de 10 MB.

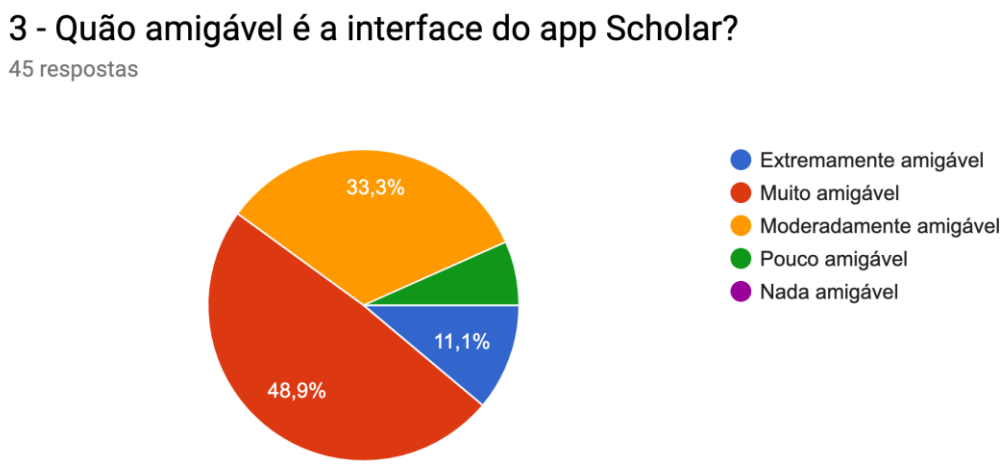

Figura 31 - Respostas da terceira pergunta do questionário de satisfação (Fonte: Autores)

De acordo com a Figura 31, 11,1% dos usuários respondentes acredita que a interface do aplicativo é extremamente amigável, 48,9% que é muito amigável, 33,3% que é pouco amigável. Nessa questão já notamos que alguns usuários não estão tão satisfeitos com a interface do aplicativo. Esse resultado já era esperado, pois tentamos manter a interface o mais simples possível. A melhora do design e usabilidade do aplicativo está listada no tópico 6.1, sobre possíveis trabalhos futuros.

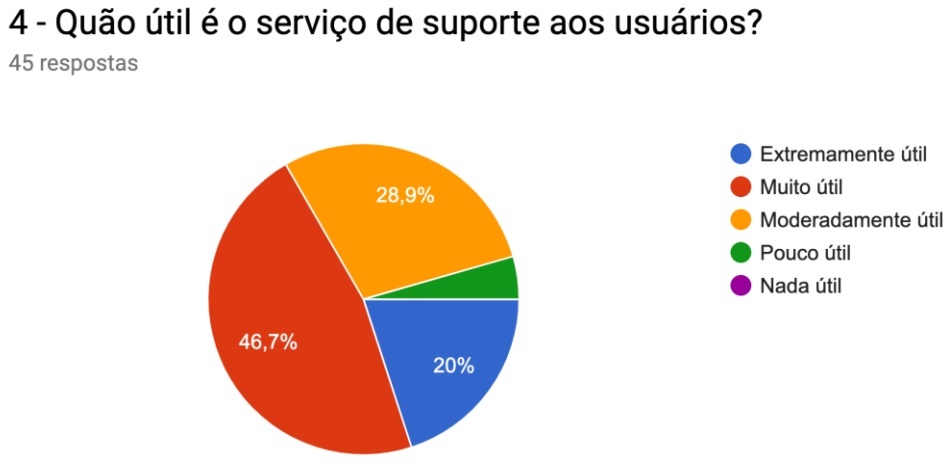

Figura 32 - Respostas da quarta pergunta do questionário de satisfação (Fonte: Autores)

Como podemos ver na Figura 32, 20% dos usuários respondentes acreditam que o serviço de suporte do aplicativo foi extremamente útil, 46,7% que foi muito útil, 28,9% que foi moderadamente útil e 4,4% que foi pouco útil. Esse resultado nos mostra que aproximadamente 70% dos usuários ficaram muito ou extremamente satisfeitos com o nosso serviço de suporte. Tentamos responder todos os emails que foram enviados para nosso canal de suporte, recebemos algumas mensagens reclamando de *bugs* ou falhas no sistemas que foram corrigidas prontamente. Também recebemos algumas mensagens com sugestões, algumas foram adicionadas ao aplicativo, outras não couberam no escopo deste trabalho e foram adicionadas na seção "Trabalhos Futuros".

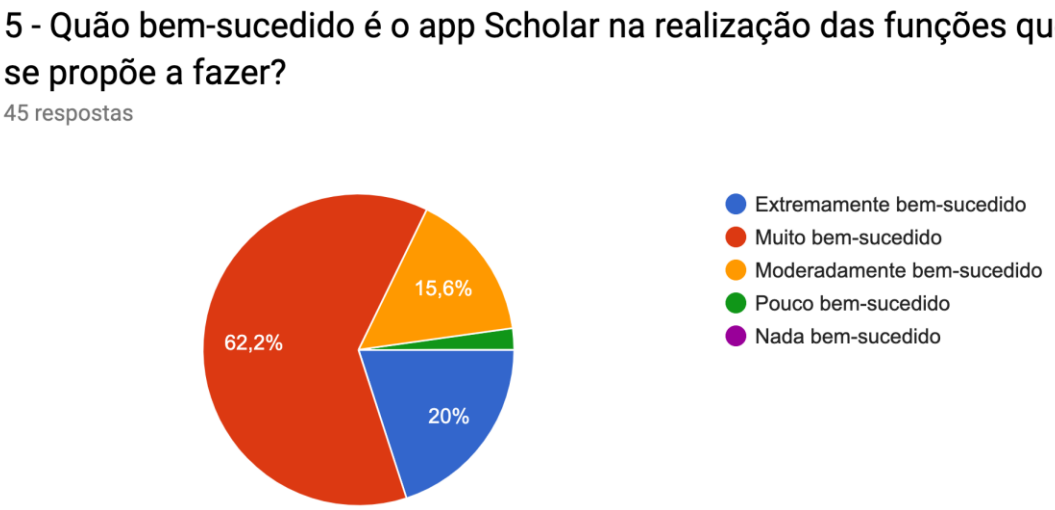

Figura 33 - Respostas da quinta pergunta do questionário de satisfação (Fonte: Autores)

De acordo com a Figura 33, aproximadamente 80% dos usuários respondentes acredita que o aplicativo é muito ou extremamente bem-sucedido na realização das funções que se propõe a fazer. Apesar do resultado extremamente positivo acreditamos que temos alguns pontos de melhoria, principalmente na parte da usabilidade e design das funcionalidades atuais.

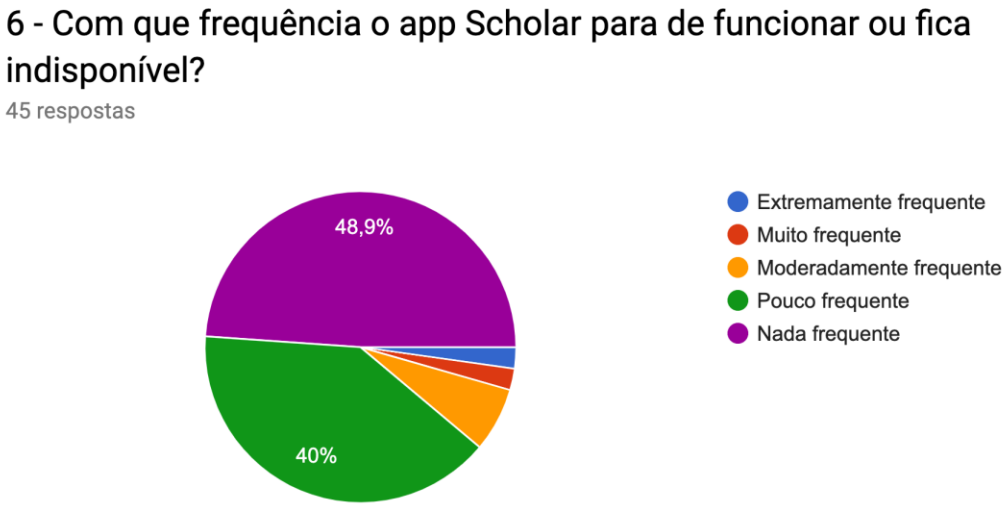

Figura 34 - Respostas da sexta pergunta do questionário de satisfação

Como pode ser observado na Figura 34, aproximadamente 90% dos usuários respondentes verificou que a aplicação ficou indisponível ou parou de funcionar com pouca ou nenhuma frequência. Esse resultado positivo se dá ao fato da resposta ativa fornecida através do nosso canal de suporte e do contínuo monitoramento do serviço de análise de erros e falhas. Assim, foi possível mitigar os erros que apareceram com relativa velocidade.

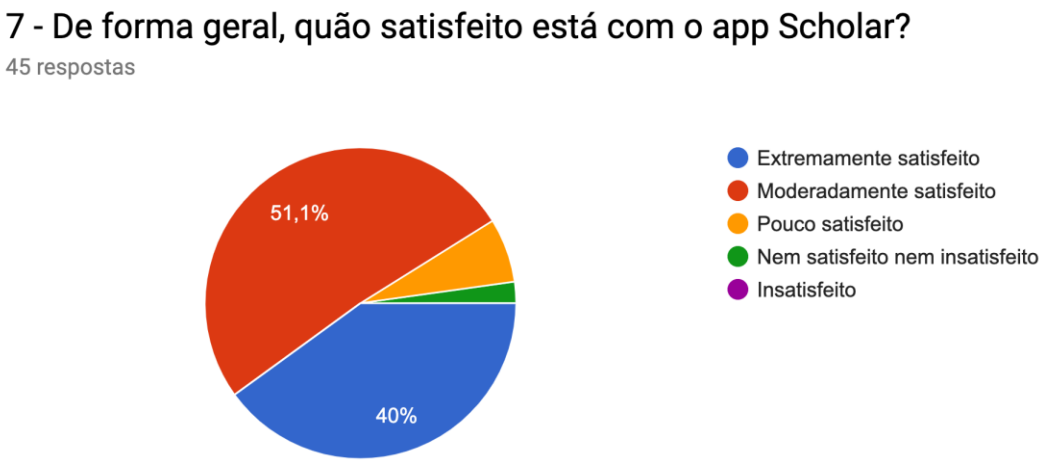

Figura 35 - Respostas da sétima pergunta do questionário de satisfação (Fonte: Autores)

De acordo com a Figura 35, aproximadamente 91% dos usuários respondentes estão moderadamente ou extremamente satisfeitos, de forma geral, com o aplicativo. Acredita-se que esse bom resultado se dá a diversos fatores, tais como o nicho aberto por apps anteriores

como GraduApp e Minha UFSC que pararam de funcionar, a qualidade do suporte e a facilidade de obtenção e uso do aplicativo

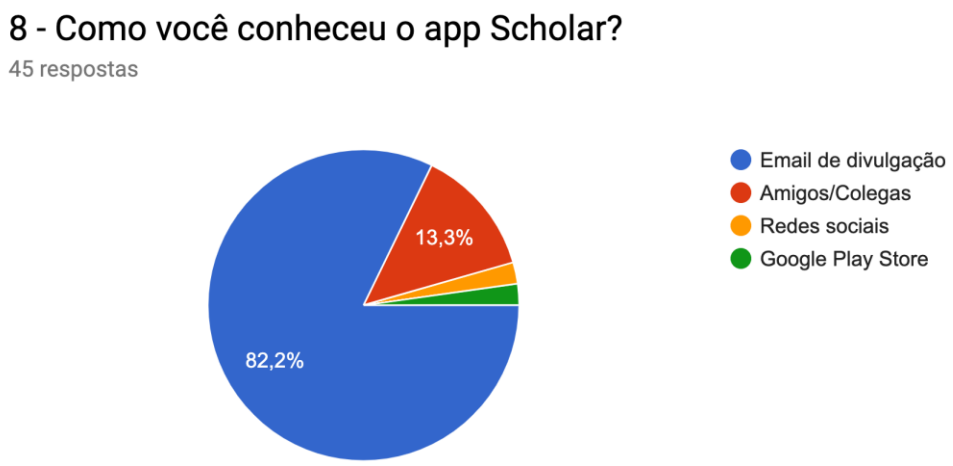

Figura 36 - Respostas da oitava pergunta do questionário de satisfação

Como podemos observar na Figura 36, a grande maioria dos usuários respondentes conheceu o app através dos e-mails de divulgação. Esse resultado já era esperado, e este gráfico reflete claramente o aumento da aquisição de usuários logo após as divulgações por e-mail.

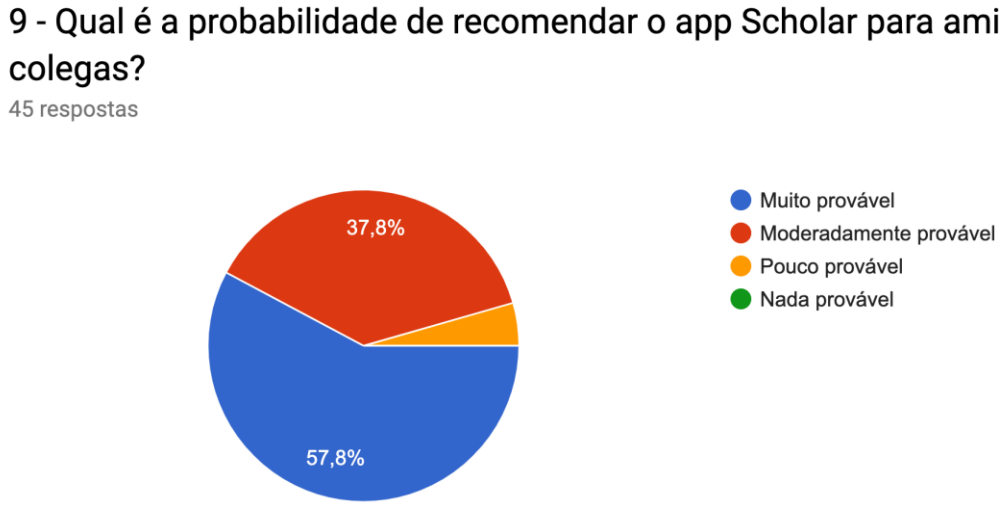

Figura 37 - Respostas da nona pergunta do questionário de satisfação

Como podemos verificar na Figura 37, aproximadamente 95% dos usuários respondentes afirma que é moderada ou muito provável a recomendação para um colega ou amigo. Esse resultado demonstra a aceitação dos usuários para com o aplicativo.

A décima e última pergunta do questionário foi "*Alguma sugestão para a melhoria do app Scholar?*". Essa pergunta era opcional, por esse motivo obtivemos um número menor de respostas com relação às últimas perguntas. Abaixo estão listadas as respostas mais relevantes:

- "Para alunos que já possuem uma graduação, como meu caso, o app traz somente os dados do primeiro curso e não é possível alterar. Talvez seja por conta dos dados que o sistema da UFSC envia, todos os dados relacionados ao aluno. Porém deve haver alguma forma ou local que indique o status do curso. O meu calendário traz as matérias do curso que já concluí, e não do curso que está em andamento."
- "Adicionar espaço para as notas de provas e trabalhos, fazendo com que a gente insira a equação para calcular a média de acordo com cada matéria."
- "Design, e questão dos horários que acaba ficando poluído por não juntar aulas faixas."
- "Adição do IA."
- "Melhorar a estética da interface e personalizar o aplicativo para cursos com diferentes regimes, como por exemplo o curso trimestral de engenharia de materiais."
- "Apresentação do horário das aulas em grade, como no cagr."
- "Agreguem mais funções úteis, e.g., matrículas."
- "Melhorar a aparência do aplicativo e adicionar algumas funções, como um cronograma com as aulas que pudesse ser editado."
- "Poderiam adicionar um widget para adicionar a tela da área de trabalho."
- "Poderiam colocar a base de dados da BU, assim poderíamos ver os livros que emprestamos e se temos multas e o prazo para devolução."

5.2 Dados das atividades dos usuários

Durante o período de coleta de dados realizamos um teste interno com cerca de 10 usuários. Após o lançamento do aplicativo realizamos duas divulgações para a comunidade acadêmica. A primeira divulgação foi restrita aos alunos do curso de graduação de Sistemas de Informação através do fórum de graduação do CAGR. A segunda divulgação foi restrita aos alunos de graduação de todos os cursos do Centro Tecnológico (CTC). Havia a

expectativa da realização de uma terceira divulgação contemplando todos os alunos da UFSC, mas a mesma não foi possível devido a diversos fatores externos.

Foram coletados diversos dados sobre a atividade desses usuário utilizando as ferramentas de Analytics do Google Play Store e do Firebase. Com essas ferramentas foi possível coletar dados do uso do app como tempo de sessão, eventos de visualização de tela e número de usuários ativo. Também foi possível coletar informações sobre os erros e falhas da aplicação utilizando a ferramenta Crashlytics, também parte do Firebase. Nas próximas seções serão apresentados os dados coletados.

5.2.1 Usuários Ativos

A métrica de usuários ativos foi coletada através da ferramenta Google Play. Com essa métrica podemos verificar a quantidade de usuários ativos por dia. Essa métrica se refere aos usuários que mantêm uma instalação ativa em pelo menos um dispositivo.

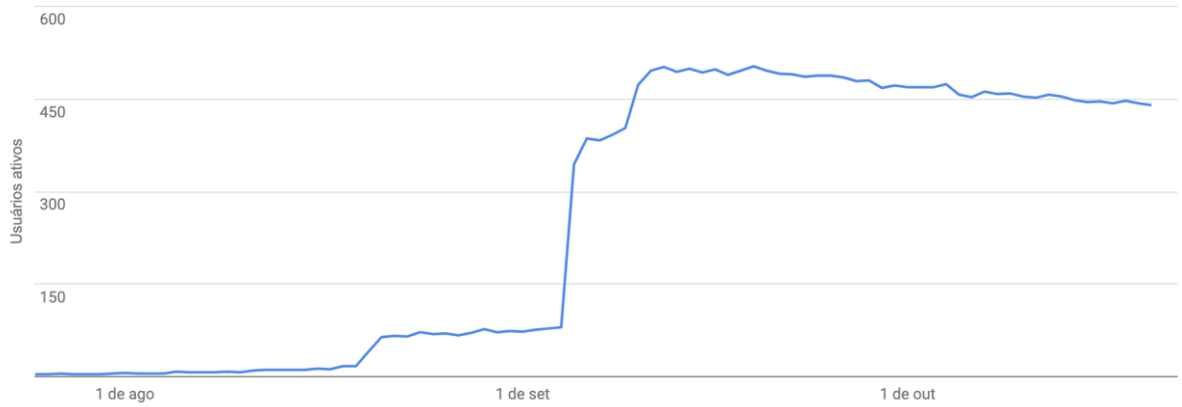

Figura 38 - Usuários ativos por período (Fonte: Firebase do projeto)

A figura 38 representa a variação dos usuários ativos num período de 180 dias. Através desse gráfico podemos notar claramente os períodos de divulgação da aplicação, a primeira divulgação aconteceu no dia 20/08/2019 e a segunda divulgação ocorreu no dia 05/09/2019. Algumas semanas após as divulgações notamos uma leve queda no número de usuários ativos, isso se deve ao fato de não haver novas divulgações e o crescimento orgânico não ser suficiente para vencer a quantidade de usuários que deixaram de usar a aplicação no mesmo período.

5.2.2 Retenção de usuários

A métrica de retenção de usuários se refere a porcentagem de usuários que retornam a cada dia. Essa métrica foi coletada através da ferramenta de analytics do Firebase. Com essa métrica podemos ter uma ideia da porcentagem de usuários que utilizam o app diariamente.

Figura 39 - Retenção de usuários por dia da aquisição (Fonte: Firebase do projeto)

A figura 39 é um gráfico da retenção de usuários por dia da aquisição. O que significa a porcentagem de usuários que permaneceram ativos pela quantidade de dias que eles baixaram a aplicação. Como podemos verificar pelo gráfico há uma ligeira queda logo após o primeiro dia, e a quantidade de usuários ativos diariamente se mantêm na faixa de 15% durante todo o período avaliado.

5.2.3 - Informações demográficas

O gráfico na figura 40 mostra a proporção de usuários com base no sexo e na faixa etária. Podemos perceber claramente que a grande maioria dos usuários é do sexo masculino. Isso se deve ao fato de que a maior divulgação do Scholar que foi possível de ser feita, foi através da lista de e-mails de alunos do CTC. A maioria dos alunos deste centro tende a ser do sexo masculino, por isso a grande diferença demonstrada no gráfico. Se tivesse sido possível a divulgação em toda a universidade, essa proporção poderia estar mais balanceada. Outro dado importante mostrado na figura 40 é a faixa etária dos usuários do aplicativo que está mais concentrada nas idades de 18 a 24 anos, o que faz todo sentido no contexto dos cursos de graduação.

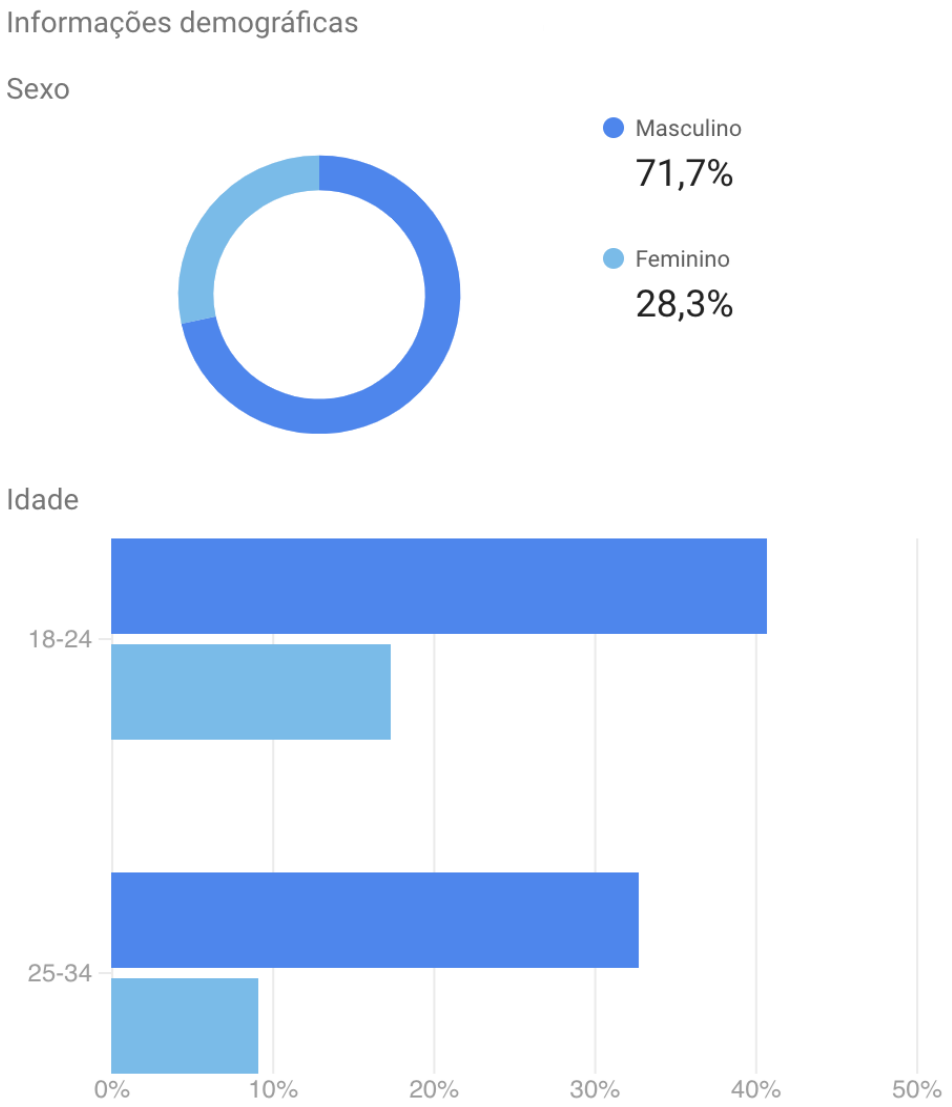

Figura 40 - Informações demográficas dos usuários do Scholar (Fonte: Firebase do Projeto)

5.2.4 - Engajamento diário médio

Engajamento diário médio é uma métrica que se refere ao tempo médio que os usuário passam diariamente na aplicação. O valor obtido no período apurado foi de 1 minuto e 10 segundos. Como o propósito da aplicação é fornecer informações rápidas e somente quando for conveniente, podemos considerar essa métrica um bom número.

5.2.5 - Falhas

Foram monitoradas 5 falhas fatais, aquelas que ocasionam o fechamento da aplicação, que afetaram apenas 2 usuários. Também foram monitoradas as falhas não fatais, aquelas que

não ocasionam o fechamento da aplicação, essa falhas foram registrada 60 vezes, afetando um total de 14 usuários.

Vale ressaltar que foi incluída uma funcionalidade na aplicação para permitir aos usuários entrar em contato diretamente conosco, informando eventuais problemas. Muitos dos problemas relatados já foram investigados e solucionados.

6 CONCLUSÃO

Com o desenvolvimento deste trabalho, pôde-se concluir que uma aplicação centralizadora de informações acadêmicas, ainda mais se tratando de um aplicativo móvel, é de fundamental importância em universidades e demais instituições de ensino. A pesquisa realizada previamente ao trabalho, mostrou que o mundo globalizado de hoje em dia, com cada pessoa a todo momento com um dispositivo móvel nas mãos, acabou mudando os paradigmas de desenvolvimento de software até então utilizados.

O aplicativo Scholar, uma vez concluído, é um bom exemplo dessas mudanças. No caso da UFSC, muitos universitários reclamam do modo como as informações são centralizadas em um site não responsivo, ainda nos dias de hoje. Essa característica negativa acaba atrapalhando e muito a obtenção dessas informações através de *smartphones* e demais dispositivos móveis. Com o desenvolvimento do aplicativo, pôde disponibilizar essas informações em um local de fácil acesso, literalmente na palma da mão do usuário. Com poucos toques na tela, é possível obter informações tais como atestados de matrícula, grades de horários, cardápios dos restaurantes, entre outras. Mesmo em sua fase de testes fechados, o aplicativo já se mostrava bastante utilizado, com os usuários inclusive relatando pequenos problemas e fazendo sugestões de melhoria, o que mostra que ainda há uma grande gama de serviços que pode ser acoplado ao aplicativo.

A utilização do framework Flutter da Google se mostrou mais do que satisfatória. A ferramenta se mostrou bastante versátil e poderosa, atendendo todas as expectativas da equipe durante o desenvolvimento do projeto. A curva de aprendizado para quem já possui alguma experiência com linguagens programação web ou mobile é muito baixa, e a oferta de bibliotecas é bastante vasta, possibilitando muitas coisas de serem construídas sem demandar um esforço absurdo. Se tratando de um *framework* de código aberto, qualquer pessoa pode realizar manutenções ou melhorias no código e enviar para integração. Como o próprio *framework* e as bibliotecas são relativamente novos, contribuições para o desenvolvimento são muito bem-vindas e encorajadas. Os próprios autores contribuíram para o desenvolvimento de um gerador de código para a biblioteca BLoC.

A versão estável lançada até o momento, já cumpre todos os requisitos especificadas no trabalho. O lançamento gradativo das funcionalidades e o auxílio de alguns colegas que se propuseram a testar o app, foi de grande ajuda para desenvolvimento da versão final, onde alguns problemas encontrados já desde cedo, puderam logo ser solucionados. O Scholar ainda possui bastante potencial para melhorias futuras, inclusive abrangendo novas funcionalidades

que já foram identificadas e poderão futuramente ser desenvolvidas. Se não pela equipe, mas em um trabalho futuro realizado por outra pessoa, visto que o aplicativo foi desenvolvido visando totalmente o código aberto e essa possível expansão por parte de colaboradores interessados.

Infelizmente, nas últimas semanas de desenvolvimento, enfrentamos problemas relacionados a API da UFSC, fornecida e gerenciada pela SeTIC. Devido a esse problema, o aplicativo ficou incapacitado de contactar a API para requisitar informações novas dos alunos. Foi aberto um chamado junto a SeTIC para solução deste problema, bem como envios insistentes de e-mails, porém até o momento o problema não foi solucionado. Este problema impede que novos usuários se conectem ao Scholar, pois não permite a obtenção de novas informações. Usuários antigos continuam podendo utilizar o aplicativo normalmente, entretanto, assim que o semestre mudar e as informações precisarem ser atualizadas, este problema, se não for solucionado, impedirá completamente o funcionamento do aplicativo.

6.1 TRABALHOS FUTUROS

Podemos afirmar que o desenvolvimento do aplicativo, no geral, foi um sucesso, quando olhamos para o que foi planejado no início do projeto: desenvolver um aplicativo modular genérico, de apoio aos estudantes em assuntos acadêmicos, que pudesse ser facilmente adaptado a outras instituições.

Algumas pendências, porém, podem ser observadas. Inicialmente a intenção era publicar o aplicativo tanto para Android quanto para iOS. Porém, devido aos altos custos para se publicar um aplicativo na Applestore, esse requisito foi deixado de lado. Para trabalhos futuros, seria o ideal que esta publicação fosse realizada, visto que vários usuários gostariam do lançamento desta versão. Como um paliativo, foi lançada a versão Web do Scholar, que pode ser acessada de qualquer dispositivo, independentemente do sistema operacional.

Existem algumas funcionalidades que foram pensadas durante o decorrer do projeto, mas que demandariam um esforço adicional muito grande que não caberia no escopo do que foi estimado, principalmente com relação ao tempo, tendo em vista o quanto de tempo foi necessário para finalmente conseguir acesso as APIs fornecidas pela SeTIC. Uma delas seria a integração com o sistema da BU, possibilitando aos usuários verificar quais livros estão locados para si e realizar renovações destes empréstimos. Um contato inicial a respeito desta funcionalidade foi feito com a SeTIC, que informou que o sistema da BU é independente e que eles não poderiam ajudar com isso. Então, para que esta funcionalidade seja

implementada, seria necessário entrar em contato diretamente com o setor técnico da biblioteca.

Uma outra funcionalidade interessante para os usuários seria a possibilidade de fazer a rematrícula, bem como ajustes na mesma, através do aplicativo. Esta *feature* também foi pensada somente mais ao fim do projeto, e como não caberia no escopo, também foi deixada de lado para um possível trabalho futuro. Similar a isto estaria a integração direta com o Moodle, que também foi pensada mas foi abandonada, por falta de uma interface que possibilite a integração e obtenção dos dados. Para que fosse executada esta integração, seria necessário desenvolver toda esta interface, o que poderia ser trabalho suficiente para um outro TCC.

O Scholar pode ser um ótimo ponto inicial para qualquer outro trabalho futuro, visto que seu código é aberto e está disponível no Github (ver Apêndice A). A UFSC oferece diversos serviços que atualmente estão muito dispersos. Uma unificação destes serviços facilitaria e muito a vida dos usuários.

Um ponto importante que vale ressaltar para qualquer trabalho futuro que dependa da SeTIC é que os desenvolvedores deste projeto se adiantem o máximo possível com relação as solicitações que dependam deles. O maior empecilho para que o aplicativo não tenha ficado exatamente como foi pensado, foi devido a extrema dificuldade de obter auxílio e resposta da SeTIC, seja por e-mails diretos ou por chamados abertos, que simplesmente ficam esquecidos por meses sem resposta.

REFERÊNCIAS BIBLIOGRÁFICAS

POUSHTER, Jacob. (2016). Smartphone ownership and internet usage continues to climb in emerging economies. *Pew Research Center* 22 (2016): 1-44. Disponível em <http://www.pewglobal.org/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/>. Acesso em: Fevereiro de 2019.

MEIRELLES, Fernando S. (2018). 29ª Pesquisa Anual do Uso de TI. Disponível em <https://eaesp.fgv.br/sites/eaesp.fgv.br/files/pesti2018gvciappt.pdf>. Acesso em: Fevereiro de 2019.

LEITE, Mattheus Rodrigo Marzola. (2017). Portal do docente para dispositivos móveis SIGA UFPR. Monografia de Especialização Digital, Universidade Federal do Paraná. Disponível em <https://acervodigital.ufpr.br/handle/1884/55847>. Acesso em: Fevereiro de 2019.

MAFRA, Marlon; GASPARIN, Ygor Henrique. (2013). Uma aplicação Android para o portal Minha UFSC. Trabalho de Conclusão de Curso de Sistemas de Informação, UFSC. Disponível em <https://repositorio.ufsc.br/handle/123456789/184683>. Acesso em: Março de 2019.

PACHECO, Pedro Henrique; SILVA, Isaac Luiz da. (2018). ANÁLISE E DESENVOLVIMENTO DE UM APLICATIVO DE AUXÍLIO DE TOMADA DE DECISÃO NA ESCOLHA DE GRADE DE HORÁRIOS UTILIZANDO IONIC E JEKYLL. Trabalho de Conclusão de Curso de Sistemas de Informação, UFSC. Disponível em <https://repositorio.ufsc.br/handle/123456789/187883>. Acesso em: Março de 2019.

QUINTELA, Jorge Manuel da Silva. (2016). UPT Mobile: Aplicação Móvel da Universidade Portucalense. Dissertação de Mestrado em Informática, Universidade Portucalense. Disponível em <http://repositorio.uportu.pt/xmlui/handle/11328/1694>. Acesso em: Março de 2019.

FREIRE, Pedro J.; RIBEIRO, Rui. (2014). Revisão de Literatura de Frameworks de Desenvolvimento Móvel Multiplataforma. In: Atas da Conferência da Associação Portuguesa de Sistemas de Informação, p. 386-398. Disponível em:

<http://revista.apsi.pt/index.php/capsi/article/download/38/33>. Acesso em: Dezembro de 2019.

HEFLIN, Houston; SHEWMAKE, Jennifer; NGUYEN, Jessica. (2017). Impact of mobile technology on student attitudes, engagement, and learning. Computers & Education, Volume 107, 2017, Pages 91-99, ISSN 0360-1315. Disponível em <https://www.sciencedirect.com/science/article/pii/S0360131517300064?via%3Dihub>. Acesso em: Março de 2019.

RODRIGUES, Timóteo de Almeida. (2016). LUSÓFONA-PROF MOBILE: Aplicação móvel de apoio aos docentes da Universidade Lusófona. Dissertação de Mestrado em Sistemas de Comunicação Multimédia, Universidade Lusófona de Humanidade e Tecnologias. Disponível em <http://recil.grupolusofona.pt/handle/10437/7455>. Acesso em: Março de 2019.

GOOGLE. (2019a). Flutter Technical Overview, What is Flutter. Disponível em: <https://flutter.dev/docs/resources/technical-overview#what-is-flutter>. Acesso em: Junho de 2019.

GOOGLE. (2019b). Flutter Technical Overview, Everything is a widget. Disponível em: <https://flutter.dev/docs/resources/technical-overview#everythings-a-widget>. Acesso em: Junho de 2019.

WU, Wenhao. (2018). React Native vs Flutter, Cross-platforms mobile application frameworks. Disponível em: <https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1>. Acesso em: Junho de 2019.

PROANDROIDDEV.COM. (2018). Flutter's Compilation Patterns. Disponível em: <https://proandroiddev.com/flutters-compilation-patterns-24e139d14177>. Acesso em: Junho de 2019.

MORIBE, Fernando. (2016). Firebase – Vantagens de um BaaS para sua Startup. Disponível em:<https://medium.com/@fgmoribe/firebase-vantagens-de-um-baas-para-sua-startup-38fd3891329a>. Acesso em: Junho de 2019.

FIREBASE. (2019). Guias do Firebase. Disponível em: <https://firebase.google.com/docs/perf-mon/?hl=pt-br>. Acesso em: Junho de 2019.

ZAMBALDI, Leonardo Flores. (2016). Aplicação web escalável e customizável para sistema educacional. Disponível em: <https://repositorio.ufsc.br/handle/123456789/171484>. Acesso em: Junho de 2019.

MDN WEB DOCS, Mozilla. (2019a). Uma visão geral do HTTP. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Acesso em: Junho de 2019.

MDN WEB DOCS, Mozilla. (2019b). Cabeçalhos HTTP. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers>. Acesso em: Junho de 2019.

WAZLAWICK, Raul. (2013). Engenharia de Software: Conceitos e Práticas. Disponível em: <https://books.google.com.br/books/about/ENGENHARIA_DE_SOFTWARE.html?id=Qtg4VUkE0V0C&redir_esc=y>. Acesso em: Outubro de 2019.

CAMARGO, Robson. (2019). Manifesto Ágil: entenda como surgiu e conheça os 12 princípios. Disponível em: <https://robsoncamargo.com.br/blog/Manifesto-Agil-entenda-como-surgiu-e-conheca-os-12-principios>. Acesso em: Junho de 2019..

BECK, Kent. (2001). Manifesto for Agile Software Development. Disponível em: <http://agilemanifesto.org/>. Acesso em: Junho de 2019.

SCRUMALLIANCE. (2018). 2017-18 State of Scrum Report. Disponível em: <https://www.scrumalliance.org/learn-about-scrum/state-of-scrum>. Acesso em: Junho de 2019.

PAGOTTO, Tiago; FABRI, José Augusto; LERARIO, Alexandre; GONÇALVES, José Antonio. (2016). Scrum solo: Software process for individual development. *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*, Las Palmas, 2016, pp. 1-6. Disponível em: <https://ieeexplore.ieee.org/document/7521555>. Acesso em: Junho de 2019.

FERREIRA, Joao Luiz Mafra. (2017). Aplicação Web para gerenciamento de campeonatos de futebol. Disponível em: <https://repositorio.ufsc.br/handle/123456789/177703>. Acesso em: Junho de 2019.

PROANDROIDDEV. (2018). Flutter's Compilation Patterns. Disponível em: <https://proandroiddev.com/flutters-compilation-patterns-24e139d14177>. Acesso em: Junho de 2019.

SILVA, Christiane E.; PANSANATO, L. T.; FABRI, José A. (2010). Ensinando diagramas UML para estudantes cegos. XVIII CIESC–XXXVI CLEI. Disponível em <http://paginapessoal.utfpr.edu.br/luciano/publicacoes/publicacoes/2.8_34_CIESC2010-%20EnsinandoDiagramasUMLparaEstudantesCegos.pdf>. Acesso em: Maio de 2019.

CSE IIT Kanpur, Dr. Dheeraj Sanghi. Computer Networks (CS425). Disponível em: <https://www.cse.iitk.ac.in/users/dheeraj/cs425/lec17.html>. Acesso em: Maio de 2019.

PEREIRA, Alexandre; ISIDORO, Eduardo; MOREIRA, Filipe; ÁVILA, Renato de. (2013). Curso gratuito de SCRUM, Unifenas Virtual. Disponível em <http://ned.unifenas.br/cursosgratuitos/201302/scrum/funcionamento.html>. Acesso em: Maio de 2019.

SURI, Sagar. (2018). Architect your Flutter project using BLOC pattern. Disponível em <https://medium.com/flutterpub/architecting-your-flutter-project-bd04e144a8f1>. Acesso em: Junho de 2019.

ANGELOV, Felix. (2019). A predictable state management library that helps implement the BLoC design pattern. Disponível em <https://github.com/felangel/bloc/>. Acesso em: Junho de 2019.

BOELENS, Didier. (2018). Reactive Programming - Streams - BLoC. Disponível em <https://www.didierboelens.com/2018/08/reactive-programming---streams---bloc/>. Acesso em: Junho de 2019.

APÊNDICE A – Código Fonte

O código fonte do aplicativo encontra-se disponível no repositório do Github, podendo ser acessado pelo link a seguir.

Código fonte: https://github.com/adsonpleal/Scholar

APÊNDICE B – Questionário de Satisfação

Este apêndice é uma transcrição do formulário de satisfação enviado para os usuários, utilizando a ferramenta Google Forms. Foram 9 questões objetivas (resposta única) e uma única questão aberta, para possíveis sugestões (questão 10).

Link de acesso ao formulário no Google Forms: https://forms.gle/12Um78ANLCY8DyCV8

Questionário de satisfação app Scholar

1. Quão fácil foi a instalação do app Scholar?
    a. Extremamente fácil
    b. Muito fácil
    c. Moderadamente fácil
    d. Pouco fácil
    e. Nada fácil
2. Quão rápida foi a instalação do app Scholar?
    a. Extremamente rápida
    b. Muito rápida
    c. Moderadamente rápida
    d. Pouco rápida
    e. Nada rápida
3. Quão amigável é a interface do app Scholar?
    a. Extremamente amigável
    b. Muito amigável
    c. Moderadamente amigável
    d. Pouco amigável
    e. Nada amigável
4. Quão útil é o serviço de suporte aos usuários?
    a. Extremamente útil
    b. Muito útil
    c. Moderadamente útil
    d. Pouco útil
    e. Nada útil

5. Quão bem-sucedido é o app Scholar na realização das funções que ele se propõe a fazer?
    a. Extremamente bem-sucedido
    b. Muito bem-sucedido
    c. Moderadamente bem-sucedido
    d. Pouco bem-sucedido
    e. Nada bem-sucedido
6. Com que frequência o app Scholar para de funcionar ou fica indisponível?
    a. Extremamente frequente
    b. Muito frequente
    c. Moderadamente frequente
    d. Pouco frequente
    e. Nada frequente
7. De forma geral, quão satisfeito está com o app Scholar?
    a. Extremamente satisfeito
    b. Moderadamente satisfeito
    c. Pouco satisfeito
    d. Nem satisfeito nem insatisfeito
    e. Insatisfeito
8. Como você conheceu o app Scholar?
    a. Email de divulgação
    b. Amigos/Colegas
    c. Redes sociais
    d. Google Play Store
9. Qual é a probabilidade de recomendar o app Scholar para amigos e colegas?
    a. Muito provável
    b. Moderadamente provável
    c. Pouco provável
    d. Nada provável
10. Alguma sugestão para a melhoria do app Scholar?

APÊNDICE C - Artigo

Scholar: Desenvolvimento de um aplicativo móvel genérico de apoio acadêmico a estudantes em universidades

Adson P. Leal[1], Angelo M. de Matos Leal[2]

[1]Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina

[2]Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina

adson.leal@grad.ufsc.br, angelo.leal@grad.ufsc.br

Abstract. *A common problem in educational institutions is the lack of centralization of information. Developing a generic open source mobile application that can be used across multiple institutions with little effort to change code was the solution found to remedy this problem. The application, named Scholar, was developed using Flutter technology and was modeled to abstract data from institutions, making it possible for different entities to use only the endpoints provided by them, requiring the least possible change in code.*

Resumo. *Um problema comum em instituições de ensino é a falta de centralização das informações. O desenvolvimento de um aplicativo móvel genérico de código aberto, que possa ser usado em várias instituições com pouco esforço para alteração no código, foi a solução encontrada para sanar este problema. O aplicativo, batizado de Scholar, foi desenvolvido utilizando a tecnologia Flutter e foi modelado de forma a abstrair os dados provenientes das instituições, tornando possível a utilização por diferentes entidades apenas utilizando os endpoints fornecidos pelas mesmas, requerendo o mínimo possível de mudança no código.*

1. Introdução

Com o mundo se tornando cada vez mais interconectado, tanto economicamente quanto socialmente, a adoção de tecnologias continua sendo um dos fatores decisivos no progresso humano (POUSHTER, 2016). Um sistema de informação adequado se torna indispensável em um ambiente acadêmico, pois, a necessidade da integridade e do acesso rápido à informação se torna evidente (LEITE, 2017), sendo assim, existem atualmente vários aplicativos que visam auxiliar a vida acadêmica de estudantes em diversas universidades. Podemos tomar como exemplo, o aplicativo Minha UFSC (MAFRA, GASPARIN, 2013), desenvolvido na Universidade Federal de Santa Catarina no ano de 2013, como um trabalho de conclusão de curso. O problema encontrado na pesquisa é que muitas universidades não possuem um aplicativo próprio onde o estudante pode facilmente acessar suas notas, grade curricular, grade de horários, contatos dos professores, histórico curricular, entre outras informações relevantes à vida estudantil.

Pensando em facilitar a vida de universitários em todo Brasil, surgiu a ideia deste projeto. Desenvolver um aplicativo genérico que possa ser integrado às fontes de dados das diversas instituições de ensino ao redor do país, mantendo um mesmo padrão na exibição dos dados

ao usuário, com uma navegação facilitada e intuitiva. A adoção de políticas “open-source” no desenvolvimento do protótipo, visa facilitar a posterior implementação de possíveis interessados em utilizar o modelo, em diversas outras instituições de ensino.

2. Conceitos Básicos

2.1. Aplicações Híbridas

Aplicações multiplataforma, ou híbridas, são aquelas desenvolvidas com uma determinada linguagem, utilizando algum framework de auxílio, que geram uma aplicação final capaz de rodar em múltiplos dispositivos e plataformas diferentes, utilizando a mesma base de código que foi implementada uma única vez. Esses tipos de aplicações são bastante interessantes porque geram economia de recursos para empresas de desenvolvimento de software, visto que apenas uma equipe é necessária para desenvolver aplicações para diferentes plataformas. Além disso, a aplicação final fica idêntica independente da plataforma utilizada, exceto pelas nuances de cada sistema operacional.

Para funcionar, uma aplicação híbrida precisa ser desenvolvida tendo como base algum framework de apoio. Definimos uma framework multiplataforma como um conjunto de arquivos de código fonte, bibliotecas e ferramentas que suportam pelo menos duas plataformas diferentes e que permitem o desenvolvimento sem ramificações do código fonte (FREIRE; RIBEIRO, 2013).

Existem atualmente inúmeros frameworks que possibilitam o desenvolvimento de aplicações híbridas, como por exemplo: Cordova, Ionic, React Native, Xamarin, Unity, Adobe Air, Flutter, entre outros. Para o desenvolvimento deste trabalho, o framework escolhido foi o Flutter.

Independente do framework, sua função para com o código gerado, é bastante semelhante. O framework é o responsável por fazer a ponte entre a aplicação e o dispositivo, através de suas APIs. Também é ele o responsável por empacotar a aplicação e gerar o executável para cada plataforma diferente.

2.2. APIs Web

API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A sigla API refere-se ao termo em inglês "Application Programming Interface" que significa em tradução para o português "Interface de Programação de Aplicativos". Esse tipo de aplicação utiliza a internet para a transferência de dados e processa e armazena informações relevantes para o funcionamento de um produto ou serviço.

Esse tipo de aplicação possui seu funcionamento baseado na conversação entre cliente e servidor, através de um protocolo de comunicação, que fica responsável por transpor os dados entre um e outro. A Figura 1 demonstra esse fluxo de informações. Os passos presentes na figura são descritos logo abaixo.

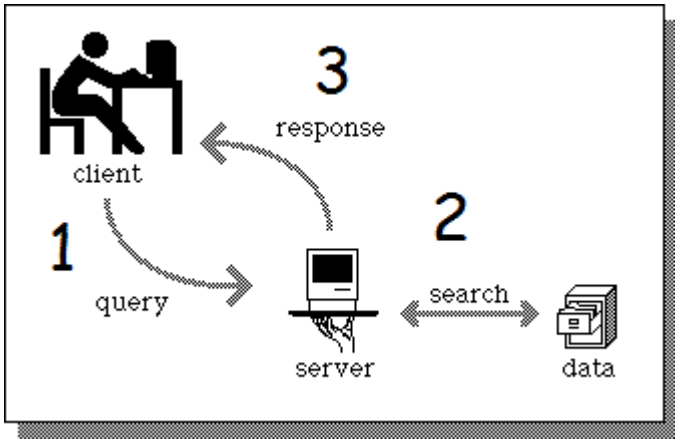

Figura 1. Comunicação entre cliente e servidor (Fonte: CSE - IIT Kanpur)

- Passo 1 (query): O cliente gera uma solicitação ao servidor, geralmente através do protocolo HTTP. Seja solicitando informações do usuário, ou informando credenciais para acesso de login por exemplo. O servidor então recebe esta requisição, avalia o tipo da mesma e faz o seu tratamento.
- Passo 2 (search): Caso a solicitação feita pelo cliente precise de acesso ao banco de dados, o servidor se encarrega de realizar este acesso para obter as informações necessárias. Este acesso pode ser somente uma consulta, para o caso de uma verificação de credenciais de login, ou de inclusão, para o caso de inserção de um novo usuário por exemplo. Após realizar as consultas (geralmente em linguagem SQL), o servidor obtém então a resposta do banco de dados e passa para o passo 3;
- Passo 3 (response): A resposta obtida pelo servidor, via acesso ao banco de dados ou não, é finalmente encaminhada ao cliente.

3. Trabalhos Correlatos

3.1. Minha UFSC

O aplicativo Minha UFSC foi uma solução desenvolvida pelos alunos Marlon Mafra e Ygor Gasparin como parte integrante de seu TCC, no curso de Sistemas de Informação na UFSC, em 2013.

Dentre as principais funcionalidades do mesmo estavam:

- Consulta da grade de horários;
- Histórico escolar;
- Cardápio do RU;
- Estatísticas de IAA;
- Avisos da Universidade;
- Consulta a pendências de livros com a BU.

O app funcionou muito bem nos primeiros anos após seu lançamento, porém depois de um tempo deixou de funcionar devido a mudanças no sistema de autenticação unificada por parte da SeTIC, o que impossibilitava os usuários de efetuarem login no sistema.

3.2. GraduApp

O GraduApp é um aplicativo desenvolvido por alunos do curso de Engenharia de Controle e Automação na UFSC. Apesar de ser mais novo que o Minha UFSC, ele acabou sofrendo dos mesmos problemas que o anterior, devido às mudanças no sistema de autenticação unificada da UFSC, realizadas recentemente pela SeTIC.

Dentre as principais funcionalidades deste app, estavam:

- Cardápio do RU;
- Grade de horários;
- Avaliações adicionadas individualmente ou por colegas;
- Controle de faltas;
- Calendário de provas;
- Histórico escolar;
- Colegas de curso.

3.3. Moodle

O Moodle, utilizado na sua versão web pela UFSC para assuntos relacionados às disciplinas ofertadas, também fornece uma solução Mobile com algumas funcionalidades interessantes, por exemplo:

- Navegar pelo conteúdo oferecido pelos cursos;
- Receber notificações sobre eventos cadastrados no Moodle;
- Encontrar e contatar pessoas do curso;
- Fazer upload de arquivos do seu smartphone;
- Ver as notas dos cursos;
- Entre outros.

Apesar de oferecer soluções interessantes, para que o usuário possa utilizar o aplicativo, primeiramente é necessário que o administrador do sistema configure o ambiente para que seja permitida a utilização do aplicativo. O aplicativo funciona na UFSC, porém possui funcionalidades limitadas.

Para este trabalho, foi levantada a hipótese de, através da autenticação unificada da UFSC, conectar com o back-end do sistema do Moodle, para fazer a integração entre Scholar e Moodle. Assim, seria possível integrar algumas funcionalidades como controle de presença, agendamento de trabalhos e provas, etc. Entretanto, o Moodle não possui uma interface de comunicação para disponibilização dos dados, sendo necessário o desenvolvimento prévio desta interface para que esta integração fosse possível. Isto demandaria um trabalho extra considerável, que não se adequaria ao escopo deste projeto.

4. Scholar

O propósito do aplicativo Scholar é servir de apoio aos estudantes que acessam informações referentes a sua universidade. Sendo assim, nosso público alvo, e por assim dizer, nosso cliente, seriam os estudantes. Dessa forma, foram feitas entrevistas informais com os potenciais usuários a fim de levantar os principais requisitos para a aplicação.

Após analisar as respostas obtidas, foi possível traduzir as principais necessidades em requisitos funcionais que a aplicação deveria atender, a fim de suprir as necessidades dos usuários. As funcionalidades presentes no aplicativo Scholar são:

- Criar uma conta, com e-mail e senha, para fazer login no aplicativo.
- Recuperar senha de uma conta previamente cadastrada, através do e-mail utilizado no cadastro.
- Conectar a aplicação ao idUFSC do usuário, através do acesso unificado da UFSC, para possibilitar o acesso às informações acadêmicas do mesmo.
- Controlar o número de faltas em cada disciplina cursada no semestre corrente, por meio da atualização manual do usuário do número de faltas em determinada disciplina.
- Visualização da grade de horários do aluno.
- Visualização do cardápio do restaurante universitário.
- Criar e gerenciar eventos, como provas e trabalhos, relacionados às disciplinas do aluno.
- Visualizar o histórico síntese do aluno diretamente no aplicativo.
- Fazer download o atestado de matrícula atualizado, diretamente no aplicativo.
- Fazer o download o currículo do curso.
- Receber notificação 10 minutos antes de uma aula.
- Receber notificação quando um usuário adiciona uma prova/trabalho na mesma turma.

5. Resultados

Ao final das seis sprints realizadas no decorrer do trabalho, obteve-se um aplicativo totalmente funcional cumprindo todos os requisitos que foram estabelecidos previamente. Decidiu-se por nomear o aplicativo de “Scholar”, que remete a assuntos acadêmicos, já que o objetivo do aplicativo é reunir as informações acadêmicas do aluno conectado.

Após a conclusão do aplicativo, o diagrama de classes do mesmo está representado na Figura 2.

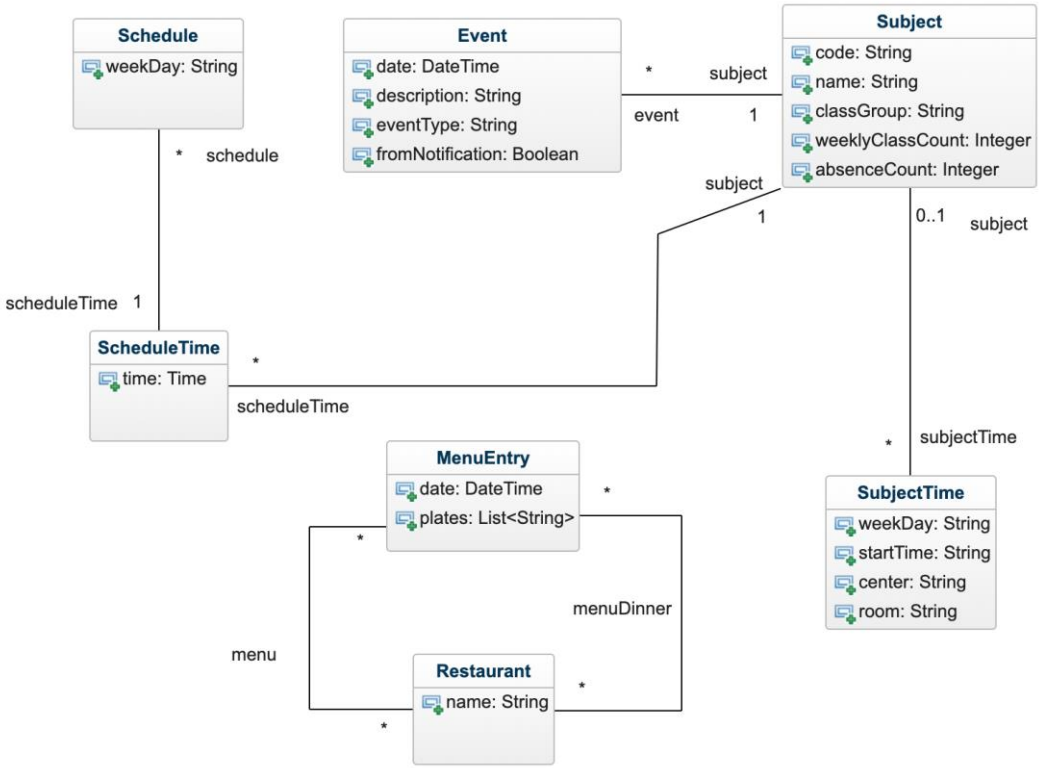

Figura 2. Diagrama de classes (Fonte: Autores)

O Scholar, na sua versão concluída, possui várias funcionalidades que antes só podiam ser acessadas através do site do CAGR. A API de integração fornecida pela SeTIC possibilitou a extração e apresentação destes dados diretamente no aplicativo. Além disso, o aplicativo também apresenta os cardápios dos RUs dos diferentes campus, sendo esta uma das funcionalidades mais solicitadas pelos usuários.

5.1 PESQUISA DE SATISFAÇÃO

Após o lançamento do aplicativo, foi feita a divulgação do mesmo em algumas listas de e-mail da universidade. Depois de algumas semanas de aplicativo publicado, decidimos enviar um formulário de avaliação, opcional, para os usuários através da ferramenta de formulários do Google. O questionário consiste em 10 perguntas, sendo 9 questões objetivas e uma questão aberta sobre possíveis melhorias no aplicativo.

Foi tomado o cuidado de mencionar o atual problema relacionado a SeTIC, que impede que novos usuários utilizem o aplicativo devido aos problemas de conexão com a API, solicitando que somente quem não estivesse enfrentando este problema respondesse este formulário. O mesmo foi enviado para toda a base de usuários e conseguimos um total de 45 respondentes. A seguir analisamos as respostas deste questionário.

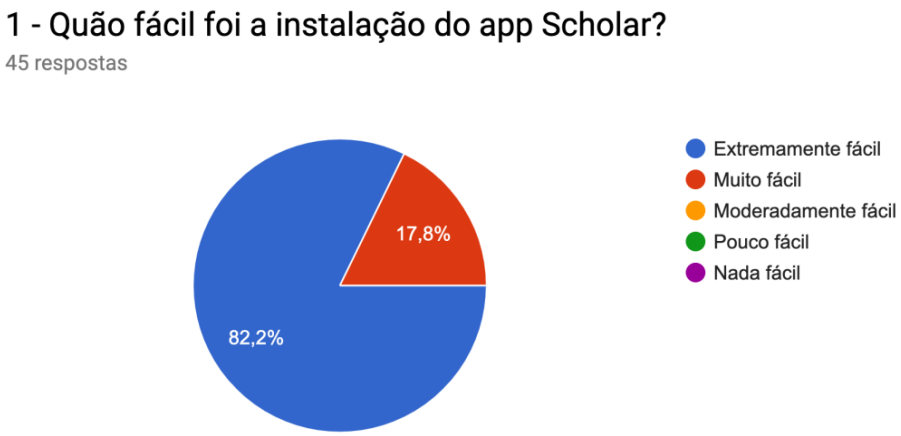

Figura 3 - Respostas da primeira pergunta do questionário de satisfação (Fonte: Autores)

Como podemos ver na Figura 3 a grande maioria dos usuários respondentes acredita que a instalação do aplicativo foi extremamente fácil. Acreditamos que esse resultado positivo se dá ao fato de utilizarmos a loja oficial de aplicativo do Google, a Play Store, para distribuir nossa aplicação. A maioria dos usuários já está acostumada a instalar aplicativos diretamente da loja de aplicativos.

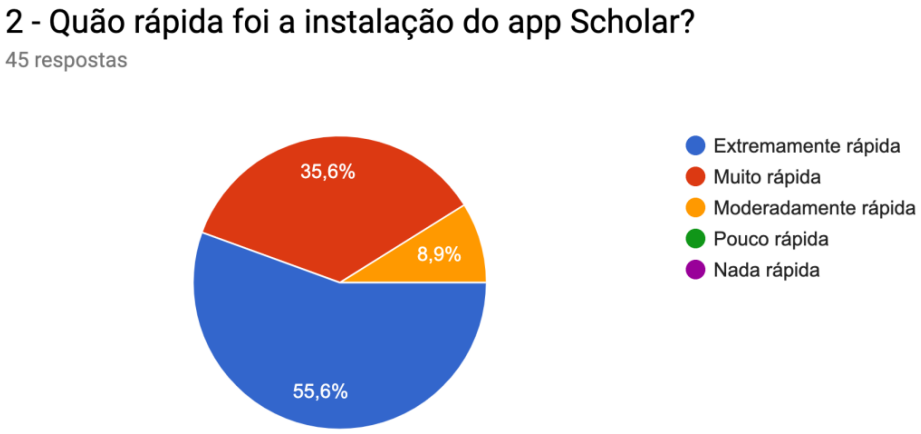

Figura 4 - Respostas da segunda pergunta do questionário de satisfação (Fonte: Autores)

Como podemos ver na Figura 4, 55.6% dos usuários respondentes acredita que a instalação do aplicativo foi extremamente rápida, 35,6% acredita que foi muito rápida e apenas 8,9% acredita que foi moderadamente rápida. Esse bom resultado se dá devido aos nossos esforços de manter baixo o tamanho da aplicação, que após lançada ficou em torno de 10 MB.

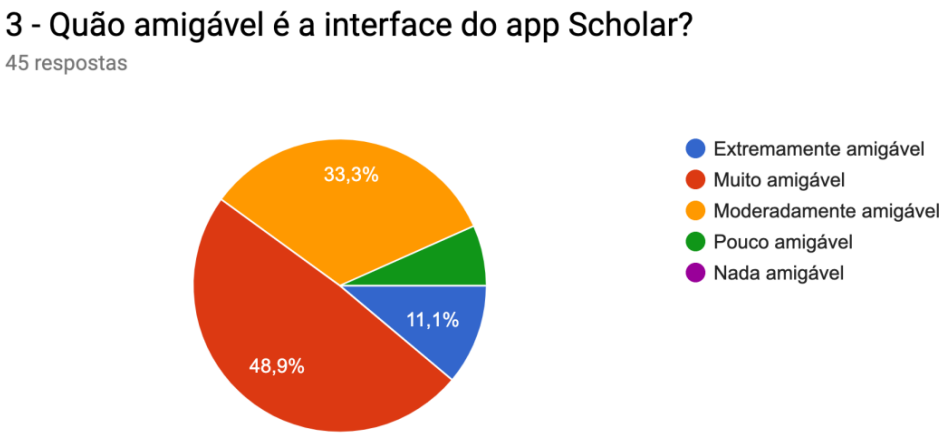

Figura 5 - Respostas da terceira pergunta do questionário de satisfação (Fonte: Autores)

De acordo com a Figura 5, 11,1% dos usuários respondentes acredita que a interface do aplicativo é extremamente amigável, 48,9% que é muito amigável, 33,3% que é pouco amigável. Nessa questão já notamos que alguns usuários não estão tão satisfeitos com a interface do aplicativo. Esse resultado já era esperado, pois tentamos manter a interface o mais simples possível. A melhora do design e usabilidade do aplicativo está listada no tópico 6.1, sobre possíveis trabalhos futuros.

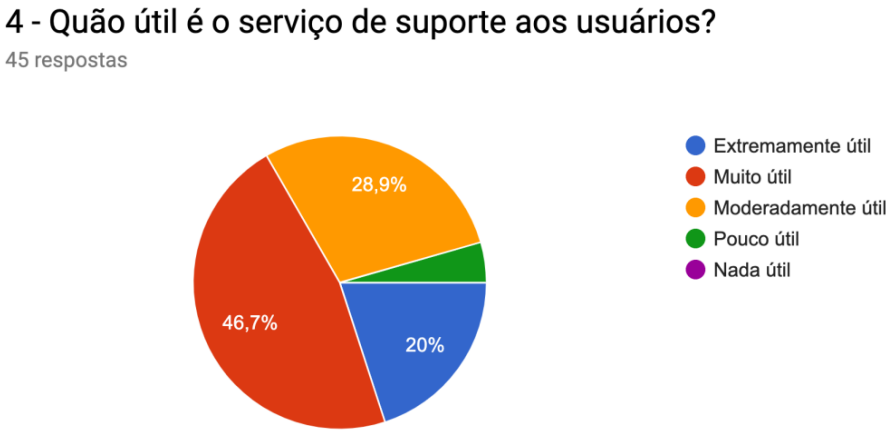

Figura 6 - Respostas da quarta pergunta do questionário de satisfação (Fonte: Autores)

Como podemos ver na Figura 6, 20% dos usuários respondentes acreditam que o serviço de suporte do aplicativo foi extremamente útil, 46,7% que foi muito útil, 28,9% que foi moderadamente útil e 4,4% que foi pouco útil. Esse resultado nos mostra que aproximadamente 70% dos usuários ficaram muito ou extremamente satisfeitos com o nosso serviço de suporte. Tentamos responder todos os emails que foram enviados para nosso canal de suporte, recebemos algumas mensagens reclamando de bugs ou falhas no sistemas que foram corrigidas prontamente. Também recebemos algumas mensagens com

sugestões, algumas foram adicionadas ao aplicativo, outras não couberam no escopo deste trabalho e foram adicionadas na seção “Trabalhos Futuros”.

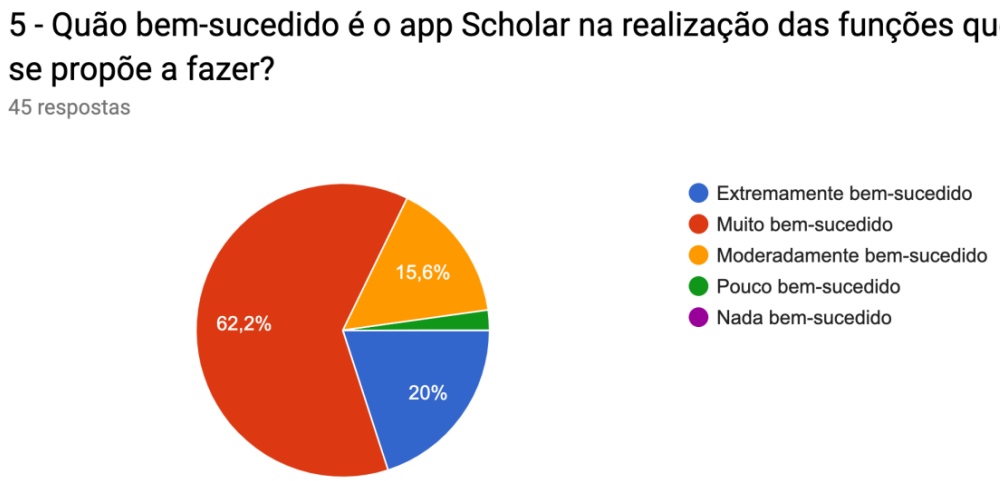

Figura 7 - Respostas da quinta pergunta do questionário de satisfação (Fonte: Autores)

De acordo com a Figura 7, aproximadamente 80% dos usuários respondentes acredita que o aplicativo é muito ou extremamente bem-sucedido na realização das funções que se propõe a fazer. Apesar do resultado extremamente positivo acreditamos que temos alguns pontos de melhoria, principalmente na parte da usabilidade e design das funcionalidades atuais.

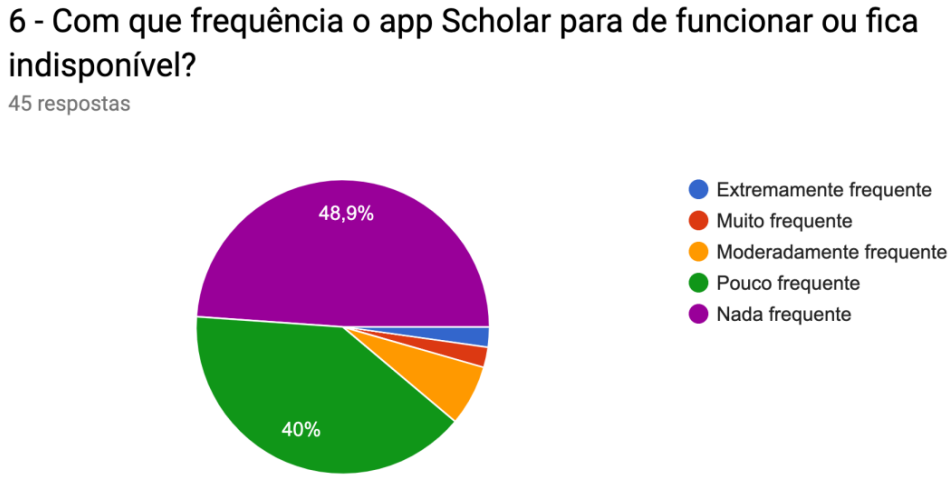

Figura 8 - Respostas da sexta pergunta do questionário de satisfação

Como pode ser observado na Figura 8, aproximadamente 90% dos usuários respondentes verificou que a aplicação ficou indisponível ou parou de funcionar com pouca ou nenhuma frequência. Esse resultado positivo se dá ao fato da resposta ativa fornecida através do nosso canal de suporte e do contínuo monitoramento do serviço de análise de erros e falhas. Assim, foi possível mitigar os erros que apareceram com relativa velocidade.

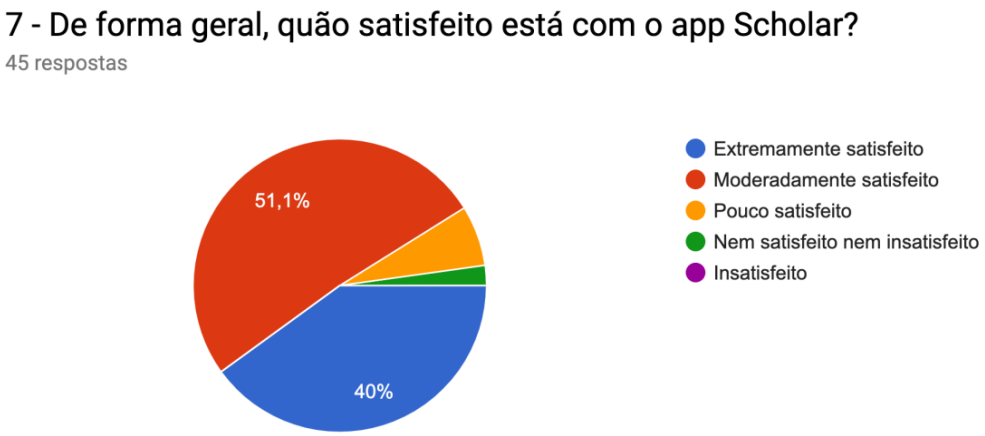

Figura 9 - Respostas da sétima pergunta do questionário de satisfação (Fonte: Autores)

De acordo com a Figura 9, aproximadamente 91% dos usuários respondentes estão moderadamente ou extremamente satisfeitos, de forma geral, com o aplicativo. Acredita-se que esse bom resultado se dá a diversos fatores, tais como o nicho aberto por apps anteriores como GraduApp e Minha UFSC que pararam de funcionar, a qualidade do suporte e a facilidade de obtenção e uso do aplicativo

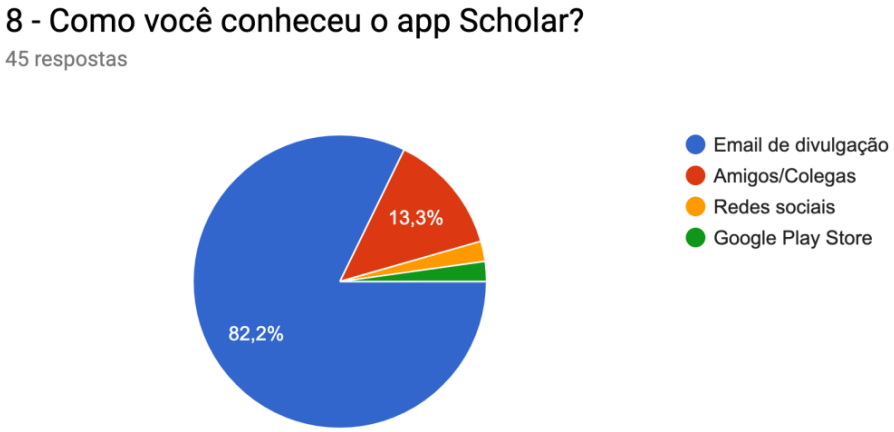

Figura 10 - Respostas da oitava pergunta do questionário de satisfação

Como podemos observar na Figura 10, a grande maioria dos usuários respondentes conheceu o app através dos e-mails de divulgação. Esse resultado já era esperado, e este gráfico reflete claramente o aumento da aquisição de usuários logo após as divulgações por e-mail.

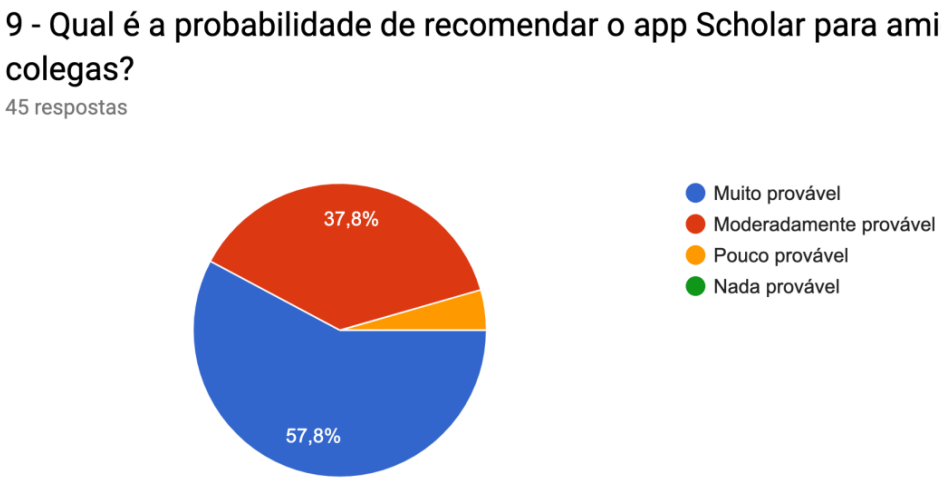

Figura 11 - Respostas da nona pergunta do questionário de satisfação

Como podemos verificar na Figura 11, aproximadamente 95% dos usuários respondentes afirma que é moderada ou muito provável a recomendação para um colega ou amigo. Esse resultado demonstra a aceitação dos usuários para com o aplicativo.
A décima e última pergunta do questionário foi "Alguma sugestão para a melhoria do app Scholar?". Essa pergunta era opcional, por esse motivo obtivemos um número menor de respostas com relação às últimas perguntas. Abaixo estão listadas as respostas mais relevantes:

- "Para alunos que já possuem uma graduação, como meu caso, o app traz somente os dados do primeiro curso e não é possível alterar. Talvez seja por conta dos dados que o sistema da UFSC envia, todos os dados relacionados ao aluno. Porém deve haver alguma forma ou local que indique o status do curso. O meu calendário traz as matérias do curso que já concluí, e não do curso que está em andamento."
- "Adicionar espaço para as notas de provas e trabalhos, fazendo com que a gente insira a equação para calcular a média de acordo com cada matéria."
- "Design, e questão dos horários que acaba ficando poluído por não juntar aulas faixas."
- "Adição do IA."
- "Melhorar a estética da interface e personalizar o aplicativo para cursos com diferentes regimes, como por exemplo o curso trimestral de engenharia de materiais."
- "Apresentação do horário das aulas em grade, como no cagr."
- "Agreguem mais funções úteis, e.g., matrículas."
- "Melhorar a aparência do aplicativo e adicionar algumas funções, como um cronograma com as aulas que pudesse ser editado."
- "Poderiam adicionar um widget para adicionar a tela da área de trabalho."
- "Poderiam colocar a base de dados da BU, assim poderíamos ver os livros que emprestamos e se temos multas e o prazo para devolução."

5.2 Dados das atividades dos usuários

Durante o período de coleta de dados realizamos um teste interno com cerca de 10 usuários. Após o lançamento do aplicativo realizamos duas divulgações para a comunidade acadêmica. A primeira divulgação foi restrita aos alunos do curso de graduação de Sistemas de Informação através do fórum de graduação do CAGR. A segunda divulgação foi restrita aos alunos de graduação de todos os cursos do Centro Tecnológico (CTC). Havia a expectativa da realização de uma terceira divulgação contemplando todos os alunos da UFSC, mas a mesma não foi possível devido a diversos fatores externos.

Foram coletados diversos dados sobre a atividade desses usuário utilizando as ferramentas de Analytics do Google Play Store e do Firebase. Com essas ferramentas foi possível coletar dados do uso do app como tempo de sessão, eventos de visualização de tela e número de usuários ativo. Também foi possível coletar informações sobre os erros e falhas da aplicação utilizando a ferramenta Crashlytics, também parte do Firebase. Nas próximas seções serão apresentados os dados coletados.

5.2.1 Usuários Ativos

A métrica de usuários ativos foi coletada através da ferramenta Google Play. Com essa métrica podemos verificar a quantidade de usuários ativos por dia. Essa métrica se refere aos usuários que mantêm uma instalação ativa em pelo menos um dispositivo.

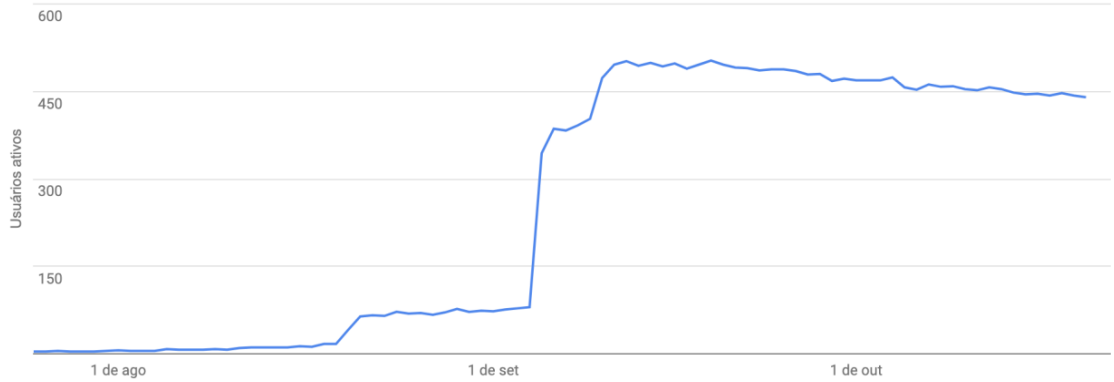

Figura 12 - Usuários ativos por período (Fonte: Firebase do projeto)

A figura 12 representa a variação dos usuários ativos num período de 180 dias. Através desse gráfico podemos notar claramente os períodos de divulgação da aplicação, a primeira divulgação aconteceu no dia 20/08/2019 e a segunda divulgação ocorreu no dia 05/09/2019. Algumas semanas após as divulgações notamos uma leve queda no número de usuários ativos, isso se deve ao fato de não haver novas divulgações e o crescimento orgânico não ser suficiente para vencer a quantidade de usuários que deixaram de usar a aplicação no mesmo período.

5.2.2 Retenção de usuários

A métrica de retenção de usuários se refere a porcentagem de usuários que retornam a cada dia. Essa métrica foi coletada através da ferramenta de analytics do Firebase. Com essa métrica podemos ter uma ideia da porcentagem de usuários que utilizam o app diariamente.

Figura 13 - Retenção de usuários por dia da aquisição (Fonte: Firebase do projeto)

A figura 13 é um gráfico da retenção de usuários por dia da aquisição. O que significa a porcentagem de usuários que permaneceram ativos pela quantidade de dias que eles baixaram a aplicação. Como podemos verificar pelo gráfico há uma ligeira queda logo após o primeiro dia, e a quantidade de usuários ativos diariamente se mantêm na faixa de 15% durante todo o período avaliado.

5.2.3 - Informações demográficas

O gráfico na figura 14 mostra a proporção de usuários com base no sexo e na faixa etária. Podemos perceber claramente que a grande maioria dos usuários é do sexo masculino. Isso se deve ao fato de que a maior divulgação do Scholar que foi possível de ser feita, foi através da lista de e-mails de alunos do CTC. A maioria dos alunos deste centro tende a ser do sexo masculino, por isso a grande diferença demonstrada no gráfico. Se tivesse sido possível a divulgação em toda a universidade, essa proporção poderia estar mais balanceada. Outro dado importante mostrado na figura 14 é a faixa etária dos usuários do aplicativo que está mais concentrada nas idades de 18 a 24 anos, o que faz todo sentido no contexto dos cursos de graduação.

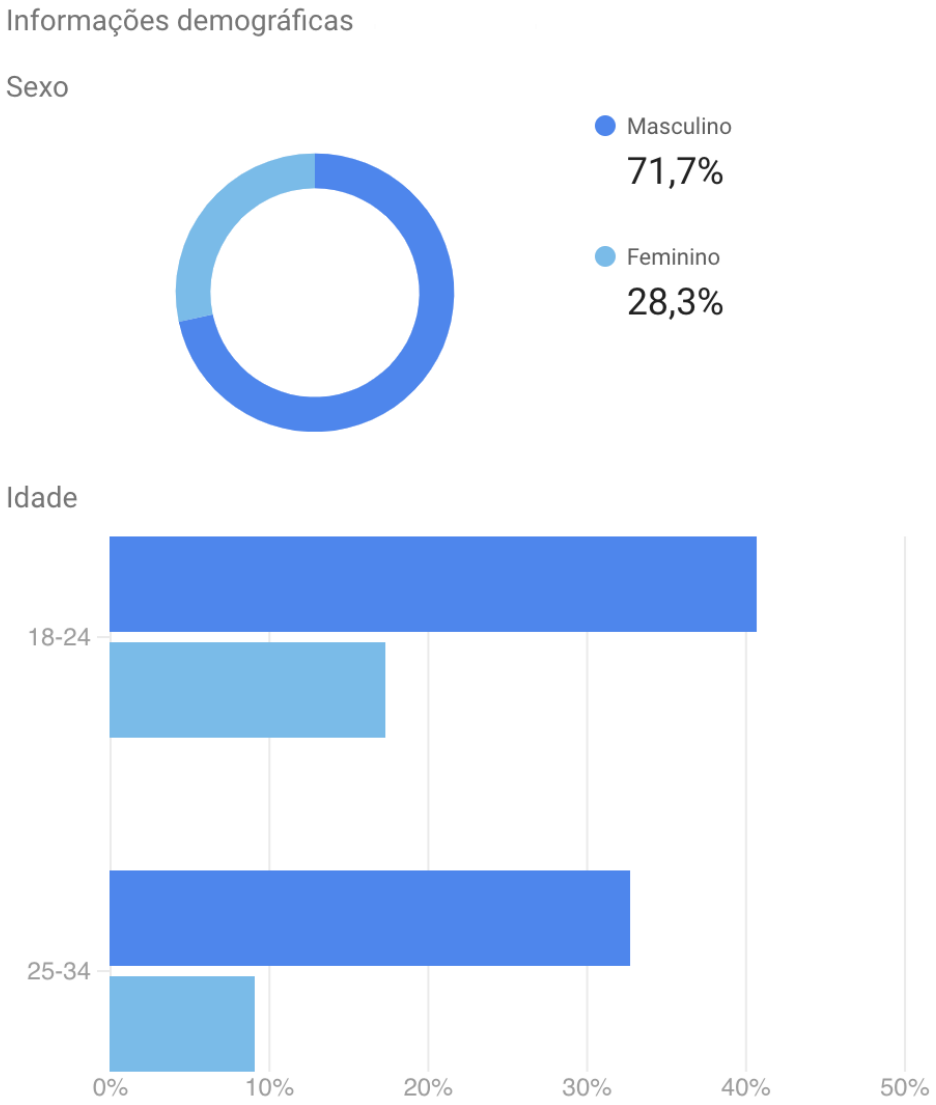

Figura 14 - Informações demográficas dos usuários do Scholar (Fonte: Firebase do Projeto)

5.2.4 - Engajamento diário médio

Engajamento diário médio é uma métrica que se refere ao tempo médio que os usuário passam diariamente na aplicação. O valor obtido no período apurado foi de 1 minuto e 10 segundos. Como o propósito da aplicação é fornecer informações rápidas e somente quando for conveniente, podemos considerar essa métrica um bom número.

5.2.5 - Falhas

Foram monitoradas 5 falhas fatais, aquelas que ocasionam o fechamento da aplicação, que afetaram apenas 2 usuários. Também foram monitoradas as falhas não fatais, aquelas que não ocasionam o fechamento da aplicação, essa falhas foram registrada 60 vezes, afetando um total de 14 usuários.

Vale ressaltar que foi incluída uma funcionalidade na aplicação para permitir aos usuários entrar em contato diretamente conosco, informando eventuais problemas. Muitos dos problemas relatados já foram investigados e solucionados.

6. Conclusões

Com o desenvolvimento deste trabalho, pôde-se concluir que uma aplicação centralizadora de informações acadêmicas, ainda mais se tratando de um aplicativo móvel, é de fundamental importância em universidades e demais instituições de ensino. A pesquisa realizada previamente ao trabalho, mostrou que o mundo globalizado de hoje em dia, com cada pessoa a todo momento com um dispositivo móvel nas mãos, acabou mudando os paradigmas de desenvolvimento de software até então utilizados.

O aplicativo Scholar, uma vez concluído, é um bom exemplo dessas mudanças. No caso da UFSC, muitos universitários reclamam do modo como as informações são centralizadas em um site não responsivo, ainda nos dias de hoje. Essa característica negativa acaba atrapalhando e muito a obtenção dessas informações através de smartphones e demais dispositivos móveis. Com o desenvolvimento do aplicativo, pôde disponibilizar essas informações em um local de fácil acesso, literalmente na palma da mão do usuário. Com poucos toques na tela, é possível obter informações tais como atestados de matrícula, grades de horários, cardápios dos restaurantes, entre outras. Mesmo em sua fase de testes fechados, o aplicativo já se mostrava bastante utilizado, com os usuários inclusive relatando pequenos problemas e fazendo sugestões de melhoria, o que mostra que ainda há uma grande gama de serviços que pode ser acoplado ao aplicativo.

A utilização do framework Flutter da Google se mostrou mais do que satisfatória. A ferramenta se mostrou bastante versátil e poderosa, atendendo todas as expectativas da equipe durante o desenvolvimento do projeto. A curva de aprendizado para quem já possui alguma experiência com linguagens programação web ou mobile é muito baixa, e a oferta de bibliotecas é bastante vasta, possibilitando muitas coisas de serem construídas sem demandar um esforço absurdo. Se tratando de um framework de código aberto, qualquer pessoa pode realizar manutenções ou melhorias no código e enviar para integração. Como o próprio framework e as bibliotecas são relativamente novos, contribuições para o desenvolvimento são muito bem-vindas e encorajadas. Os próprios autores contribuíram para o desenvolvimento de um gerador de código para a biblioteca BLoC.

A versão estável lançada até o momento, já cumpre todos os requisitos especificadas no trabalho. O lançamento gradativo das funcionalidades e o auxílio de alguns colegas que se propuseram a testar o app, foi de grande ajuda para desenvolvimento da versão final, onde alguns problemas encontrados já desde cedo, puderam logo ser solucionados. O Scholar ainda possui bastante potencial para melhorias futuras, inclusive abrangendo novas funcionalidades que já foram identificadas e poderão futuramente ser desenvolvidas. Se não pela equipe, mas em um trabalho futuro realizado por outra pessoa, visto que o aplicativo foi desenvolvido visando totalmente o código aberto e essa possível expansão por parte de colaboradores interessados.

Infelizmente, nas últimas semanas de desenvolvimento, enfrentamos problemas relacionados a API da UFSC, fornecida e gerenciada pela SeTIC. Devido a esse problema, o aplicativo ficou incapacitado de contactar a API para requisitar informações novas dos alunos. Foi aberto um chamado junto a SeTIC para solução deste problema, bem como

envios insistentes de e-mails, porém até o momento o problema não foi solucionado. Este problema impede que novos usuários se conectem ao Scholar, pois não permite a obtenção de novas informações. Usuários antigos continuam podendo utilizar o aplicativo normalmente, entretanto, assim que o semestre mudar e as informações precisarem ser atualizadas, este problema, se não for solucionado, impedirá completamente o funcionamento do aplicativo.

6.1. Trabalhos Futuros

Podemos afirmar que o desenvolvimento do aplicativo, no geral, foi um sucesso, quando olhamos para o que foi planejado no início do projeto: desenvolver um aplicativo modular genérico, de apoio aos estudantes em assuntos acadêmicos, que pudesse ser facilmente adaptado a outras instituições.

Algumas pendências, porém, podem ser observadas. Inicialmente a intenção era publicar o aplicativo tanto para Android quanto para iOS. Porém, devido aos altos custos para se publicar um aplicativo na Applestore, esse requisito foi deixado de lado. Para trabalhos futuros, seria o ideal que esta publicação fosse realizada, visto que vários usuários gostariam do lançamento desta versão. Como um paliativo, foi lançada a versão Web do Scholar, que pode ser acessada de qualquer dispositivo, independentemente do sistema operacional.

Existem algumas funcionalidades que foram pensadas durante o decorrer do projeto, mas que demandariam um esforço adicional muito grande que não caberia no escopo do que foi estimado, principalmente com relação ao tempo, tendo em vista o quanto de tempo foi necessário para finalmente conseguir acesso as APIs fornecidas pela SeTIC. Uma delas seria a integração com o sistema da BU, possibilitando aos usuários verificar quais livros estão locados para si e realizar renovações destes empréstimos. Um contato inicial a respeito desta funcionalidade foi feito com a SeTIC, que informou que o sistema da BU é independente e que eles não poderiam ajudar com isso. Então, para que esta funcionalidade seja implementada, seria necessário entrar em contato diretamente com o setor técnico da biblioteca.

Uma outra funcionalidade interessante para os usuários seria a possibilidade de fazer a rematrícula, bem como ajustes na mesma, através do aplicativo. Esta feature também foi pensada somente mais ao fim do projeto, e como não caberia no escopo, também foi deixada de lado para um possível trabalho futuro. Similar a isto estaria a integração direta com o Moodle, que também foi pensada mas foi abandonada, por falta de uma interface que possibilite a integração e obtenção dos dados. Para que fosse executada esta integração, seria necessário desenvolver toda esta interface, o que poderia ser trabalho suficiente para um outro TCC.

O Scholar pode ser um ótimo ponto inicial para qualquer outro trabalho futuro, visto que seu código é aberto e está disponível no Github. A UFSC oferece diversos serviços que atualmente estão muito dispersos. Uma unificação destes serviços facilitaria e muito a vida dos usuários.

Um ponto importante que vale ressaltar para qualquer trabalho futuro que dependa da SeTIC é que os desenvolvedores deste projeto se adiantem o máximo possível com relação às solicitações que dependam deles. O maior empecilho para que o aplicativo não tenha ficado exatamente como foi pensado, foi devido a extrema dificuldade de obter auxílio e

resposta da SeTIC, seja por e-mails diretos ou por chamados abertos, que simplesmente ficam esquecidos por meses sem resposta.

Referências

POUSHTER, Jacob. (2016). Smartphone ownership and internet usage continues to climb in emerging economies. Pew Research Center 22 (2016): 1-44. Disponível em <http://www.pewglobal.org/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/>. Acesso em: Dezembro de 2019.

LEITE, Mattheus Rodrigo Marzola. (2017). Portal do docente para dispositivos móveis SIGA UFPR. Monografia de Especialização Digital, Universidade Federal do Paraná. Disponível em <https://acervodigital.ufpr.br/handle/1884/55847>. Acesso em: Dezembro de 2019.

MAFRA, Marlon; GASPARIN, Ygor Henrique. (2013). Uma aplicação Android para o portal Minha UFSC. Trabalho de Conclusão de Curso de Sistemas de Informação, UFSC. Disponível em <https://repositorio.ufsc.br/handle/123456789/184683>. Acesso em: Dezembro de 2019.

FREIRE, Pedro J.; RIBEIRO, Rui. (2014). Revisão de Literatura de Frameworks de Desenvolvimento Móvel Multiplataforma. In: Atas da Conferência da Associação Portuguesa de Sistemas de Informação, p. 386-398. Disponível em: <http://revista.apsi.pt/index.php/capsi/article/download/38/33>. Acesso em: Dezembro de 2019.