



## CURSO DE SISTEMAS DE INFORMAÇÃO

### **DESENVOLVIMENTO DE UM APLICATIVO MULTIPLATAFORMA PARA AUXILIO NA GESTÃO DE DISCIPLINAS E ATIVIDADES ACADÊMICAS**

JHEMYS BARROS LIMA

Palmas - TO

2021



## CURSO DE SISTEMAS DE INFORMAÇÃO

### **DESENVOLVIMENTO DE UM APLICATIVO MULTIPLATAFORMA PARA AUXILIO NA GESTÃO DE DISCIPLINAS E ATIVIDADES ACADÊMICAS**

JHEMYS BARROS LIMA

Trabalho apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do professor Esp. Frederico Pires Pinto.

Palmas - TO

2021



## CURSO DE SISTEMAS DE INFORMAÇÃO

### DESENVOLVIMENTO DE UM APLICATIVO MULTIPLATAFORMA PARA AUXILIO NA GESTÃO DE DISCIPLINAS E ATIVIDADES ACADÊMICAS

JHEMYS BARROS LIMA

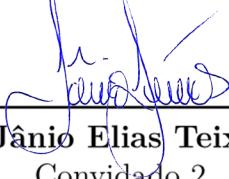
Trabalho apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do professor Esp. Frederico Pires Pinto.

  
\_\_\_\_\_  
Prof. Esp. Frederico Pires Pinto

Orientador

  
\_\_\_\_\_  
Prof. Me. Silvano Maneck Malfatti

Convidado 1

  
\_\_\_\_\_  
Prof. Me. Jânio Elias Teixeira Júnior

Convidado 2

Palmas - TO

2021

*Dedico este trabalho e essa conquista a minha mulher Amanda e aos meus filhos, Davi e Vanessa, eles foram os principais incentivadores e força motriz que permitiu o meu avanço mesmo durante os momentos mais difíceis. Agradeço do fundo do meu coração, amo vocês.*

# Agradecimentos

A minha mulher, Amanda Vargas, por todo incentivo, compreensão e por ser a principal apoiadora ao ter começado essa jornada do zero novamente.

Aos meus filhos, Davi e Vanessa, por serem minhas motivações diárias para continuar lutando, para poder proporcionar um futuro melhor pra eles.

A meu sogro e sogra, Wilson Moreira e Zenaide Vargas por terem sido verdadeiros anjos acolhedores, meus mais sinceros agradecimentos por todo apoio, incentivo e ajuda quem tem dado durante toda essa loga trajetória, espero conseguir retribuir tudo isso um dia.

Ao meu amigo Everton Júnior, por sempre ter me ajudado e tirado minhas dúvidas nos momentos que eu precisei.

Aos meus colegas de curso, pelo companheirismo e trabalho em equipe nos momentos mais difíceis.

A Universidade Estadual do Tocantins, pela oportunidade de estar fazendo o curso.

Ao meu professor orientador Frederico Pires Pinto, pela orientação, ensinamento, confiança e disponibilidade.

*“Tudo tem o seu tempo determinado,  
e há tempo para todo o propósito debaixo do céu.  
(Bíblia Sagrada, Eclesiastes 3,1)*

# Resumo

No contexto das instituições de Ensino Superior, a evasão acadêmica é um fenômeno complexo que indica a interrupção do ciclo de estudo por parte dos alunos. Sendo um fato recorrente que afeta quase todos os cursos de graduação, este problema está ainda mais presente em cursos de Tecnologia da Informação, onde dentre vários fatores, pode ser destacado a dificuldade que os acadêmicos têm em se manter organizados perante a tantas disciplinas e atividades durante sua vida acadêmica. Portanto, através da realização de uma pesquisa bibliográfica e aplicada, este trabalho tem por objetivo implementar um aplicativo para dispositivos móveis a partir da modelagem realizada no Projeto de Conclusão de Curso, visando auxiliar alunos do ensino superior a organizarem melhor seus compromissos acadêmicos. Será fornecido recursos que o auxiliarão no seu dia a dia, como: gerência das suas disciplinas, agendamento de provas e seminários, monitorar devolução de livros e ter controle sobre a frequência acadêmica. Também é possível que o aluno possa fazer anotações sobre as aulas, anexar imagens, links, etc. O aplicativo foi desenvolvido em uma linguagem híbrida, utilizando o *framework* Flutter, almejando alcançar tanto dispositivos Android quanto dispositivos iOS. Dito isto, a partir da realização de testes tanto de desempenho como de qualidade de código, os resultados apontam para a viabilidade da implantação da solução em ambiente de produção.

**Palavras-chaves:** Aplicativo. Agenda. Organização. Estudantes. Flutter.

# Abstract

In the context of Higher Education institutions, academic dropout is a complex phenomenon that indicates the interruption of the study cycle by students. It is a recurring fact that affects almost all undergraduate courses, but this problem is even more present in Information Technology courses, where among several factors, it can be highlighted the difficulty that the students have in keeping organized when facing so many subjects and activities during their academic life. Therefore, through a bibliographic and applied research, this work aims to implement an application for mobile devices from the modeling done in the Final Term Project, to help higher education students to better organize their academic commitments. It will provide resources that will help them in their daily lives, such as: managing their disciplines, scheduling tests and seminars, monitoring the return of books, and having control over academic attendance. It is also possible for the student to make notes on classes, attach images, links, etc. The application was developed in a hybrid language, using the Flutter framework, aiming to reach both Android and iOS devices. That said, based on performance and code quality tests, the results point to the viability of implementing the solution in a production environment.

**Key-words:** App. Schedule. Organization. Students. Flutter.

# Listas de ilustrações

Figura 1 – Gráfico dos sistemas operacionais moveis.	21
Figura 2 – Smartphone com Android 11	22
Figura 3 – iPhone com iOS 14.	23
Figura 4 – Nativo vs Hibrido.	24
Figura 5 – Exemplo de aplicativo Flutter.	26
Figura 6 – Relação de cursos.	30
Figura 7 – Gráfico de períodos letivos dos entrevistados.	31
Figura 8 – Arquitetura da aplicação.	33
Figura 9 – Coleção de documento do aplicativo.	36
Figura 10 – Diagrama Caso de Uso - Agenda Acadêmica	39
Figura 11 – Diagrama de Classe - Agenda Acadêmica	40
Figura 12 – Diagrama de Componentes - Agenda Acadêmica	42
Figura 13 – Tela de <i>Login</i> e cadastro de usuário	44
Figura 14 – Menu do aplicativo e tela de listagem de disciplina.	45
Figura 15 – Formulário de cadastro de disciplina e tela de listagem de professor.	46
Figura 16 – Formulário de cadastro de professor e tela de listagem de livros.	47
Figura 17 – Formulário de cadastro de livro.	48
Figura 18 – Modelo de chamadas bottom up	53
Figura 19 – Modelo de chamadas bottom up	54
Figura 20 – Gráfico Flame	54
Figura 21 – Relatório de análise do SonarQube	55
Figura 22 – Informação sobre o questionário.	61
Figura 23 – Dados sobre os entrevistados.	62
Figura 24 – Pergunta 01.	63
Figura 25 – Pergunta 02.	63
Figura 26 – Pergunta 03.	64
Figura 27 – Pergunta 04.	64
Figura 28 – Pergunta 05.	64
Figura 29 – Pergunta 06.	64
Figura 30 – Pergunta 07.	65
Figura 31 – Pergunta 08.	65
Figura 32 – Pergunta 09.	65
Figura 33 – Pergunta 10.	66
Figura 34 – Pergunta 11.	66
Figura 35 – Pergunta 12.	66
Figura 36 – Sugestão de Funcionalidades.	66

Figura 37 – SO utilizado pelos entrevistados. . . . .	67
Figura 38 – Cadastrar disciplinas cursadas no período letivo. . . . .	68
Figura 39 – Cadastrar dados dos professores. . . . .	68
Figura 40 – Cadastrar avaliações. . . . .	69
Figura 41 – Cadastrar atividades e seminários. . . . .	69
Figura 42 – Cadastrar as notas obtidas nas disciplinas. . . . .	70
Figura 43 – Criar horário de aula. . . . .	70
Figura 44 – Controle de eventos. . . . .	71
Figura 45 – Controle de frequência. . . . .	72
Figura 46 – Controle de devolução de livro. . . . .	72
Figura 47 – Exportar (fazer backup) dados e configuração do aplicativo. . . . .	73
Figura 48 – Viabilidade de implementação do aplicativo. . . . .	74

# **Lista de tabelas**

Tabela 1 – Fatores e causas da evasão no ensino superior. . . . .	19
Tabela 2 – Quadro comparativo. . . . .	20
Tabela 3 – Recursos de hardware utilizados. . . . .	28
Tabela 4 – Ferramentas de softwares utilizados. . . . .	28
Tabela 5 – Descrição do documento contendo as informações sobre disciplinas. . .	35

# **Lista de código-fonte**

1	Código Dart responsável por inserir os dados de disciplina no banco. . . . .	37
2	Código responsável por contar a lógica de <i>login</i> do aplicativo. . . . . . . . .	49
3	Código responsável por salvar dados no banco de dados. . . . . . . . . .	50
4	Código responsável por excluir dados no banco de dados. . . . . . . . .	51
5	Código responsável por fazer a leitura das informações no banco de dados.	52

# **Lista de abreviaturas e siglas**

API - *Application Programming Interface.*

App – Aplicativo.

CPU - Central Process Unit;

IDE - *Integrated Development Environment.*

MVC - *Model View Controller.*

SDK - *Software Development Kit.*

SI - Sistemas de Informação.

SO - Sistema Operacional.

UI - *User Interface.*

UML - *Unified Modeling Language.*

URL - *Uniform Resource Locator.*

VM - *Virtual machine.*

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>16</b>
<b>1.1</b>	<b>Objetivos</b>	<b>17</b>
1.1.1	Objetivo Geral	17
1.1.2	Objetivos Específicos	17
<b>1.2</b>	<b>Estrutura</b>	<b>17</b>
<b>2</b>	<b>REFERENCIAL TEÓRICO</b>	<b>18</b>
<b>2.1</b>	<b>Principais Fatores que Colaboram para a Desistência dos Acadêmicos do Ensino Superior</b>	<b>18</b>
<b>2.2</b>	<b>Trabalhos Relacionados</b>	<b>19</b>
<b>2.3</b>	<b>Dispositivos Móveis</b>	<b>21</b>
<b>2.4</b>	<b>Android</b>	<b>21</b>
<b>2.5</b>	<b>iOS</b>	<b>22</b>
<b>2.6</b>	<b>Desenvolvimento de Aplicativo Nativo vs Multiplataforma</b>	<b>23</b>
<b>2.7</b>	<b>Flutter</b>	<b>25</b>
<b>2.8</b>	<b>Dart</b>	<b>27</b>
<b>3</b>	<b>METODOLOGIA</b>	<b>28</b>
<b>3.1</b>	<b>Tipo de Pesquisa</b>	<b>28</b>
<b>3.2</b>	<b>Materiais de Hardware e Software</b>	<b>28</b>
<b>3.3</b>	<b>Etapa de Comunicação, Planejamento e Modelagem do Aplicativo</b>	<b>29</b>
3.3.1	Comunicação	29
3.3.2	Planejamento e Modelagem	29
3.3.2.1	Elicitação e Análise de Requisitos do Aplicativo	29
3.3.2.1.1	Perfil dos candidatos	30
3.3.2.2	Requisitos do aplicativo	31
3.3.2.2.1	Requisitos funcionais	31
3.3.2.2.2	Requisitos não funcionais	31
3.3.2.3	Modelagem dos Diagramas do Aplicativo	32
3.3.2.3.1	Diagrama de caso de uso	32
3.3.2.3.2	Diagrama de classe	32
3.3.2.3.3	Diagrama de componentes	32
<b>3.4</b>	<b>Etapa de Desenvolvimento do Aplicativo</b>	<b>33</b>
3.4.1	Justificativa para a Adoção do Flutter e do Dart no Desenvolvimento do Aplicativo	33
3.4.2	Arquitetura e Estrutura do Projeto	33

3.4.3	Ambiente de Desenvolvimento . . . . .	34
3.4.4	Processo de Desenvolvimento . . . . .	34
3.4.5	Armazenamento de Dados . . . . .	35
<b>3.5</b>	<b>Etapa de Testes de Desempenho e Análise de Qualidade de Código</b> . . . . .	<b>37</b>
3.5.1	Teste de Desempenho . . . . .	38
3.5.2	Análise de Qualidade de Código do Aplicativo . . . . .	38
<b>4</b>	<b>RESULTADOS</b> . . . . .	<b>39</b>
<b>4.1</b>	<b>Diagramas da UML</b> . . . . .	<b>39</b>
4.1.1	Diagrama de Cado de Uso . . . . .	39
4.1.2	Diagrama de Classe . . . . .	40
4.1.3	Diagrama de Componentes . . . . .	42
<b>4.2</b>	<b>Aplicativo Desenvolvido</b> . . . . .	<b>43</b>
4.2.1	Interface do Aplicativo Desenvolvido . . . . .	43
4.2.2	Fragments de Código do Aplicativo Proposto . . . . .	48
4.2.2.1	<i>Login</i> . . . . .	48
4.2.2.2	Salvamento de dados . . . . .	50
4.2.2.3	Exclusão dos dados . . . . .	50
4.2.2.4	Leitura dos dados do banco . . . . .	51
<b>4.3</b>	<b>Teste de Desempenho do Aplicativo com o Dart DevTools</b> . . . . .	<b>53</b>
4.3.1	CPU profiler . . . . .	53
4.3.1.1	Bottom up . . . . .	53
4.3.1.2	Call tree . . . . .	53
4.3.1.3	Flame chart . . . . .	54
<b>4.4</b>	<b>Teste de Qualidade de Código com SonarQube</b> . . . . .	<b>55</b>
<b>5</b>	<b>CONCLUSÃO</b> . . . . .	<b>56</b>
<b>5.1</b>	<b>Trabalhos Futuros</b> . . . . .	<b>56</b>
	<b>REFERÊNCIAS</b> . . . . .	<b>57</b>
	<b>ANEXOS</b> . . . . .	<b>60</b>
	<b>ANEXO A – QUESTIONÁRIO A RESPEITO DO LEVANTAMENTO DOS REQUISITOS DO APLICATIVO</b> . . . . .	<b>61</b>
<b>A.1</b>	<b>Esclarecimento a respeito do questionário</b> . . . . .	<b>61</b>
<b>A.2</b>	<b>Dados dos entrevistados</b> . . . . .	<b>62</b>
<b>A.3</b>	<b>Questionário sobre as funcionalidades do aplicativo</b> . . . . .	<b>63</b>
	<b>ANEXO B – RESULTADOS COLETADOS DO QUESTIONÁRIO</b> . . . . .	<b>67</b>

<b>B.1</b>	<b>Resultado a respeito do paradigma de desenvolvimento do aplicativo</b>	<b>67</b>
<b>B.2</b>	<b>Resultado das perguntas a respeito das funcionalidades do aplicativo</b>	<b>67</b>
B.2.1	Cadastro de disciplinas . . . . .	68
B.2.2	Cadastrar de dados dos professores . . . . .	68
B.2.3	Cadastrados de avaliações . . . . .	69
B.2.4	Cadastrar atividades . . . . .	69
B.2.5	Cadastrar as notas . . . . .	70
B.2.6	Criar horário . . . . .	70
B.2.7	Controle de eventos . . . . .	71
B.2.8	Controle de frequência . . . . .	72
B.2.9	Controle de devolução de livro . . . . .	72
B.2.10	Exportação de dados e configuração . . . . .	73
<b>B.3</b>	<b>Resultado da pesquisa sobre a viabilidade do desenvolvimento do aplicativo</b> . . . . .	<b>73</b>

## **APÊNDICES** **75**

	<b>APÊNDICE A – REQUISITOS FUNCIONAIS E REQUISITO NÃO FUNCIONAL</b> . . . . .	<b>76</b>
<b>A.1</b>	<b>Requisitos Funcionais</b> . . . . .	<b>76</b>
<b>A.2</b>	<b>Requisitos Não Funcionais</b> . . . . .	<b>89</b>

# 1 Introdução

Os *smartphones* são dispositivos móveis que têm se tornado cada vez mais indispensáveis no dia a dia de uma pessoa devido a todos os recursos e facilidades que esse aparelho pode oferecer. Um desses recursos e que tem popularizado ainda mais esses dispositivos são os aplicativos que nele pode ser instalado.

Com o passar do tempo os *smartphones* têm se mostrado cada vez mais completos e multifuncionais, isto aliado a todos os recursos que eles oferecem, no qual têm amplificado ainda mais o seu uso. Cotidianamente novos aplicativos são desenvolvidos para os mais diversos segmentos, aumentando o impacto desse dispositivo na vida das pessoas ([DIAS, 2007](#)).

Conforme Prodest ([2021](#)), os aplicativos estão presentes cada vez mais em nosso cotidiano. Este fato está relacionado a todos os recursos oferecidos e as mais diversas tarefas que eles podem desempenhar, como acesso à redes sociais, jogos, mensagens instantâneas, solicitar um carro com motorista por aplicativos de transporte privado e urbano, pedir refeição, assistir filmes e séries, e ter acesso ao banco.

Sendo assim, visando a popularidade dos aplicativos, principalmente em como eles podem contribuir significativamente na rotina de uma pessoa, buscam-se diariamente soluções de modo a favorecer diversas áreas, tais como segurança, saúde, transportes, sociedade e educação.

Dado o contexto de evasão de acadêmicos em instituições de Ensino Superior brasileiro, tem-se observado ser um problema recorrente e que atinge praticamente todos os cursos de nível superior, e tem como consequência um índice abaixo do esperado para formação de novos profissionais, assim como também acarreta na frustração por parte dos acadêmicos que não conseguem concluir o curso de graduação, gerando desperdício de recursos ([HIPÓLITO, 2011](#) apud [ZANATO; VENTURA; RIBEIRO, 2018](#)).

A evasão acadêmica pode ocorrer devido a uma série de motivos, podendo eles ser classificados de duas formas, os que estão relacionados a fatores internos a instituição, como corpo docente, assistência socioeducacional e estrutura insuficiente de apoio ao ensino, ou a fatores externos, como problemas pessoais, desvalorização da profissão, econômico e sociais da contemporaneidade ou ainda por dificuldade em se organizar perante as disciplinas propostas ([PAREDES, 1994](#) apud [ZANATO; VENTURA; RIBEIRO, 2018](#)).

Dado a exposição nos parágrafos anteriores, buscou-se realizar um estudo sobre o tema e propor uma solução para auxiliar discentes perante a dificuldade enfrentada em realizar o gerenciamento das disciplinas acadêmicas e atividades solicitadas nelas durante

a vida acadêmica, assim, este trabalho visa desenvolver um aplicativo para dispositivos móveis para auxiliar acadêmicos do Ensino Superior a fazer um melhor gerenciamento das suas atividades acadêmicas.

## 1.1 Objetivos

### 1.1.1 Objetivo Geral

Desenvolver um aplicativo multiplataforma para dispositivos móveis (Android, iOS), que auxilie discentes no gerenciamento de disciplinas e atividades acadêmicas a fim de contribuir para redução da evasão escolar.

### 1.1.2 Objetivos Específicos

Para conduzir este projeto, têm-se os seguintes objetivos específicos:

- Realizar pesquisa bibliográfica sobre a evasão acadêmica, dispositivos móveis, aplicativo nativo vs multiplataforma e das tecnologias Flutter e Dart;
- Implementar um aplicativo multiplataforma conforme os requisitos funcionais e não funcionais elicitados durante a elicitação de requisitos; e
- Realizar testes de desempenho do aplicativo com *DevTools* e análise da qualidade do código com o SonarQube.

## 1.2 Estrutura

Este trabalho de conclusão de curso está organizado em 5 capítulos. Após a introdução, os próximos capítulos seguem a seguinte estrutura:

Capítulo 2: aborda a fundamentação teórica e detalha os conceitos relacionados às escolhas tecnológicas utilizadas durante o desenvolvimento do aplicativo.

Capítulo 3: demonstra como o aplicativo foi projetado e como foi organizado o processo de desenvolvimento.

Capítulo 4: apresenta os resultados obtidos durante o processo de desenvolvimento do aplicativo.

Capítulo 5: esse último capítulo apresenta a conclusão do projeto, bem como a explanação acerca dos trabalhos futuros.

## 2 Referencial Teórico

### 2.1 Principais Fatores que Colaboram para a Desistência dos Acadêmicos do Ensino Superior

Uma das grandes motivações para o desenvolvimento desse aplicativo, foi a observação da considerável taxa de acadêmicos que abandonam seus cursos de graduação das Instituições de Ensino Superior (IES).

A evasão é um fenômeno social complexo, definido como interrupção no ciclo de estudos. É um problema que vem preocupando as instituições de ensino em geral, sejam públicas ou particulares, pois a saída de alunos provoca graves consequências sociais, acadêmicas e econômicas ([BAGGI; LOPES, 2011](#)).

A evasão acadêmica é um problema recorrente no cursos de graduação e com muita incidência histórica. Conforme apontado por Hoed ([2016](#)) em seu artigo, existem números no Brasil e em diversos outros países que apontam para taxas elevadas de evasão. Em um levantamento elaborado pelo Sindicato das Entidades Mantenedoras de Estabelecimentos de Ensino Superior no Estado de São Paulo - SEMESP ([2012](#)), cursos da área de tecnologia da informação são os que apresentam a maior taxa de evasão. Ainda segundo o SEMESP ([2012](#)), de três acadêmicos que entram em um curso de Sistemas de Informação, somente um consegue chegar até o final e receber o diploma. Esse índice chega a ser maior no curso de Ciência da Computação, onde a cada quatro acadêmicos que cursam esse curso, somente um consegue se formar.

De acordo com David e Chaym ([2019](#)), que evidencia fatores que geralmente levam os alunos a abandonar seus respectivos cursos, fatores esses que podem estar associados a escolha incorreta do curso de formação, devido a uma falsa familiaridade com a área de estudo proposta pelo curso; dificuldade de conciliar o trabalho e os estudos, problema esse que está presente em todos os cursos de formação e que é a realidade da maioria dos acadêmicos; e por último, o elevado índice de reprovação que acontece em especial nos cursos de Tecnologia da Informação (TI) e de engenharias afins.

David e Chaym ([2019](#)) relata ainda, através da Tabela 1 criada pelos próprios autores do artigo pode-se observar os possíveis fatores e causas para a evasão no ensino superior, pois é um fenômeno complexo.

Tabela 1 – Fatores e causas da evasão no ensino superior.

Fatores	Causas
Externos às instituições	O mercado de trabalho; reconhecimento social na carreira escolhida; conjuntura econômica; desvalorização da profissão; dificuldade de se atualizar perante às evoluções tecnológicas, econômicas e sociais da contemporaneidade; e políticas governamentais.
Internos às instituições	Questões peculiares à própria academia; falta de clareza sobre o projeto pedagógico do curso; baixo nível de didática-pedagógica; cultura institucional de desvalorização da docência; e estrutura insuficiente de apoio ao ensino.
Individuais dos estudantes	Habilidade de estudo; personalidade; formação universitária anterior; escolha precoce da profissão; dificuldades pessoais de adaptação à vida universitária; desencanto com o curso escolhido; dificuldades recorrentes de reprovações ou baixa frequência; e desinformação a respeito da natureza dos cursos.

**Fonte:** David e Chaym (2019).

Para Kraemer (2011), o professor também desempenha um papel fundamental no combate a evasão acadêmica. Através de metodologias de ensino bem elaboradas o educador tem o poder de incentivar o acadêmico proporcionando conhecimento de qualidade, motivação e senso crítico.

A motivação do acadêmico também é um fator determinante para o processo de aprendizagem e de permanência na instituição de ensino, como também ter o hábito de estudo constante além de se manter organizado perante a todos compromissos acadêmicos. Logo, esses agentes podem contribuir elevação do desempenho acadêmico (GIL, 2015 apud CAVALCANTE; JUNIOR, 2013).

Fica evidente através do artigo citado, que as reprovações estão geralmente associadas a problemas familiares, dificuldade que o aluno tem com o conteúdo proposto ou mesmo a inaptidão dos acadêmicos em ter que organizar as atividades acadêmicas, fazendo com que as atividades que foram solicitadas por seus docentes e conteúdo de provas se acumulem, provocando reprovação e por consequência o desânimo por parte do aluno em prosseguir a graduação.

## 2.2 Trabalhos Relacionados

Existem alguns aplicativos do mesmo gênero que se propõem a contribuir para a organização dos acadêmicos, sendo eles: Agenda do Estudante Pro - Organize-se!<sup>1</sup>, Agenda

<sup>1</sup> Disponível em: <<https://bitlyli.com/xzn3CB>>

Escolar<sup>2</sup>, Notas U - Agenda para estudantes<sup>3</sup>, e Agenda do Universitário<sup>4</sup>. Porém, os recursos que estão presentes em alguns desses aplicativos, estão ausentes em outros. Sendo assim, a proposta do aplicativo aqui desenvolvido é proporcionar a maior gama de recursos para fazer desse app mais completa em relação aos outros, conforme a Tabela 2. Algumas dessas funcionalidades são as de ter o controle de devolução de livros, controle de frequência acadêmica, horário acadêmico e outras. O objetivo é atender a demanda elicitada através de entrevista com os *stakeholders*<sup>5</sup> que aqui será representado por acadêmicos do Ensino Superior.

Tabela 2 – Quadro comparativo.

	Agenda Acadêmica	Agenda do Estudante Pro - Organize-se!	Agenda Escolar	Notas U - Agenda para estudantes	Agenda do Universitário
Login	✓	-	-	-	-
Gestão de professores	✓	-	✓	-	-
Gestão de disciplinas	✓	✓	✓	✓	✓
Gestão de atividades	✓	✓	✓	✓	✓
Gestão de anotações	✓	-	✓	-	-
Gestão de seminários	✓	-	-	-	-
Gestão de provas	✓	✓	✓	✓	✓
Gestão de notas	✓	✓	✓	✓	✓
Gestão de eventos	✓	-	✓	✓	✓
Controle de devolução de livro	✓	✓	-	✓	✓
Horário acadêmico	✓	✓	✓	✓	✓
Controle de frequência	✓	-	-	✓	✓
Versão paga do app	-	-	✓	-	✓

**Fonte:** Autor.

#### Legenda:

O símbolo ‘✓’: Significa a presença do recurso ou funcionalidade.

O símbolo ‘-’: Significa a ausência do recurso ou funcionalidade.

Conforme exposto na Tabela 2, pode ser observado que de modo geral, a maioria dos aplicativos apresentam os mesmos recursos, porém ainda há uma dispersão muito grande com relação as funcionalidades, obrigando o usuário a ter que utilizar mais de um aplicativo para ser totalmente atendido.

<sup>2</sup> Disponível em: <<https://bitlyli.com/jej1Do>>

<sup>3</sup> Disponível em: <<https://bitlyli.com/ptTLVX>>

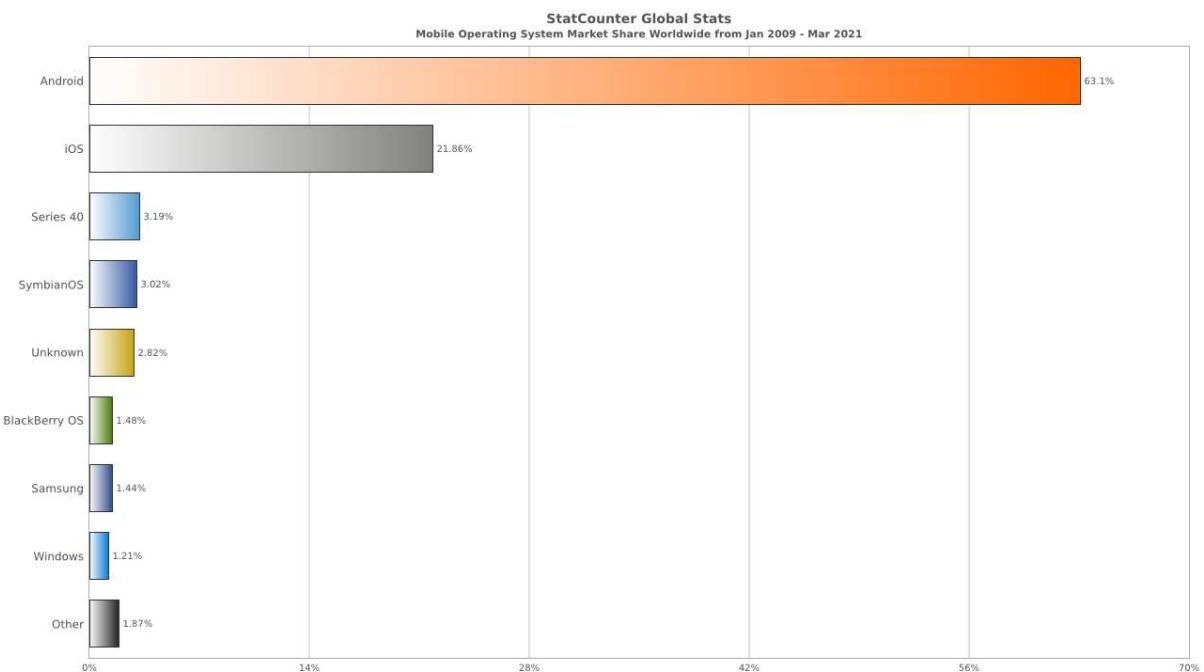
<sup>4</sup> Disponível em: <<https://bitlyli.com/QVnsdg>>

<sup>5</sup> **Stakeholder** é a junção de “stake” que significa: Interesse, participação, risco e “holder” aquele que possui. Refere-se a qualquer um que tenha interesse no sucesso de um projeto

## 2.3 Dispositivos Móveis

Os celulares fazem parte cada vez mais do nosso dia a dia e vão muito além de simplesmente fazer ligações e enviar mensagens de texto. Com eles é possível navegar pela Internet e executar softwares assim como fazemos em nossos computadores. Esses softwares, também conhecidos como aplicativos para dispositivos móveis, permitem que nós usuários possamos interagir com nossos dispositivos através da tela sensível ao toque. Existem as mais diversas funcionalidades para os aplicativos, que incluem jogos, troca de mensagens, redes sociais, para uso pessoal e negócios (SILVA; SANTOS, 2014).

Figura 1 – Gráfico dos sistemas operacionais moveis.



**Fonte:** ([GS.STATCOUNTER.COM](https://gs.statcounter.com), 2021).

No universo dos dispositivos móveis, duas grandes empresas dividem a maior porcentagem do mercado mundial quando se trata de Sistemas Operacionais (SO) (Gartner, 2018). Conforme o gráfico colhido no site Statcounter, site especialista em estatísticas globais e análise da *web*, e demostrado no Figura 1, o Android pertencente a Google, lidera o *ranking* com 63.1%, seguido pelo iOS da Apple com 21.86% do mercado global de SO móveis.

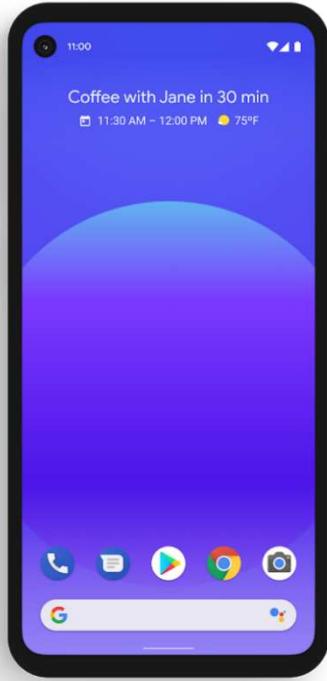
## 2.4 Android

O Android (Figura 2) é um sistema operacional que está presente na maior parte dos dispositivos móveis da atualidade. Este sistema que tem de forma progressiva tornado-se cada vez mais popular, devido ao fato de se tratar de um sistema de código aberto (*open source*). Logo, pode ser personalizado por empresas que utilizam esse SO em seus dispositivos, como a Samsung, LG, Motorola e Xiaomi (ABLESON; KING; SEN, 2012).

O Android é primariamente um esforço do Google, em colaboração com a Open Handset Alliance. Open Handset Alliance é uma aliança de dezenas de organizações comprometidas em trazer para o mercado um telefone celular “melhor” e mais “aberto” (ABLESON; KING; SEN, 2012).

O foco desse SO é ser flexível e moderno, para tornar rápido e menos trabalhoso o desenvolvimento de aplicativos. Os aplicativos desenvolvidos para essa plataforma são codificados usando a linguagem de programação Java ou Kotlin.

Figura 2 – Smartphone com Android 11



**Fonte:** ([ANDROID.COM, 2021](#)).

A loja oficial de aplicativos da Google é a PlayStore, local onde qualquer aplicativo dessa plataforma pode ser encontrado, e onde os programadores podem publicar os aplicativos que desenvolveram. Segundo o site Statista, O Google Play Store foi originalmente lançado em outubro de 2008 com o nome de Android Market. Visto que o mesmo funciona como loja de aplicativos oficial do Google, oferecendo a seus clientes uma ampla variedade de aplicativos e mídia digital, incluindo música, revistas, livros, filmes e TV. Ainda segundo o mesmo, o número de aplicativos disponíveis na Play Store atingiu recentemente a marca de 3,04 milhões de aplicativos, ultrapassando a quantia 1 milhão de aplicativos em julho de 2013 ([Statista, 2021b](#)).

## 2.5 iOS

O iOS (Figura 3) é o sistema operacional móvel que pertence à Apple e está presente nos dispositivos iPhone, iPad. Ao contrário da sua concorrente Android, o iOS é um SO proprietário, ou seja, não pode ter seu código fonte modificado e nem pode ser executado em outros dispositivos que não sejam da Apple, o que faz iOS propriedade exclusiva da Apple.

Segundo o autor Steil ([2012](#)), com o lançamento do iPhone, o sistema operacional que roda no dispositivo tinha o criativo nome de iPhone OS. Com a evolução dos dispositivos e a chegada do iPad, o sistema mudou de nome para iOS.

Figura 3 – iPhone com iOS 14.



**Fonte:** ([APPLE.COM](https://apple.com), 2021)

Os aplicativos para essa plataforma são codificados usando a linguagem de programação Swift, onde são disponibilizados na App Store, loja oficial da Apple, local onde os programadores disponibilizam os aplicativos desenvolvidos para essa plataforma. Segundo o site Statista, no primeiro trimestre de 2021, a Apple App Store foi a segunda maior loja de aplicativos com cerca de 2,22 milhões de aplicativos disponíveis para iOS ([Statista, 2021a](#)).

## 2.6 Desenvolvimento de Aplicativo Nativo vs Multiplataforma

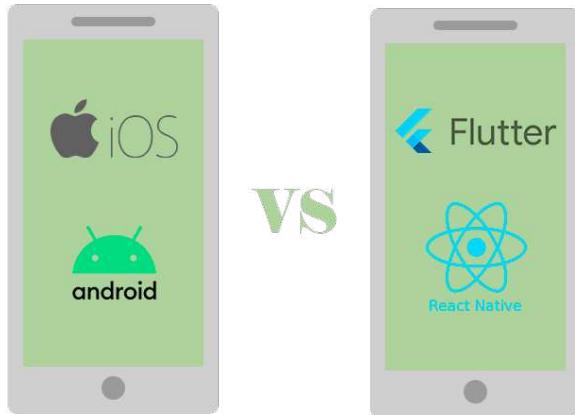
Atualmente existem duas maneiras de se desenvolver aplicativos para dispositivos móveis. A nativa, que tem como foco a criação de aplicativos que vão ser executados em dispositivos e plataformas específicos, também podem acessar os recursos nativos integrados dos *smartphones*, como câmera e microfone, por padrão e sem a necessidade de *plug-ins*. Já a multiplataforma tem como foco o desenvolvimento de aplicativos para mais de uma plataforma utilizando apenas um código, e em muitos casos apenas um código-fonte ([SILVA; SANTOS, 2014](#)).

De acordo com Corazza ([2018](#)), um dos principais objetivos dos desenvolvedores de aplicativos móveis, geralmente, é tornar seu aplicativo disponível para a maior quantidade de usuários imaginável com o menor custo possível.

De modo a ser o mais inclusivo possível, existem duas principais abordagens para essa categoria de aplicação, sendo o desenvolvimento nativo e o híbrido, também comumente conhecido como desenvolvimento multiplataforma. A princípio ambas metodologias compartilham os mesmos recursos e *design* parecidos. Mas as tecnologias utilizadas em cada tipo desenvolvimento são bem diferentes ([MARINHO, 2020](#)).

A partir da Figura 4, será explicado quais são as diferenças, vantagens e desvantagens de ambas metodologias.

Figura 4 – Nativo vs Híbrido.



**Fonte:** Autor.

Segundo Santos (2018), desenvolvimento híbrido ou multiplataforma é a metodologia que usa *frameworks* com o intuito de criar para aplicativos utilizando somente um código-fonte, ou seja, com apenas um código é possível criar aplicativos que serão publicados em mais de uma plataforma mobile, tal como Android e iOS. E essa é uma das principais diferenças no desenvolvimento nativo, que o desenvolvimento ocorrerá apenas para a plataforma específica. Há diversos *frameworks* disponível no mercado para esse tipo de desenvolvimento, dentre eles os mais conhecidos e utilizados ultimamente são React Native que utiliza a linguagem de programação JavaScript e Flutter que usa a linguagem Dart. Nesse tipo de desenvolvimento, o custo de produção é mais baixo se comparado ao desenvolvimento nativo, tendo em vista que se demanda mais recursos para desenvolver um aplicativo para cada plataforma.

De acordo com Matos (2017), é possível desenvolver para mais de uma plataforma *mobile*, a partir de um único código-fonte, o que diminui a complexidade de lidar com códigos de plataformas distintas. Outra vantagem é a possibilidade de estar desenvolvendo usando apenas um *kit* de ferramenta, tendo em vista que a *kits* específicos para cada plataforma, a exemplo do X-Code que é a IDE para desenvolver aplicativos para o iOS. E também usar apenas uma linguagem de programação, já que o Android utiliza Kotlin ou Java, enquanto iOS utiliza Swift. Em suma, a proposta do desenvolvimento multiplataforma é justamente estar disponível nas diversas plataformas mobile, o que garante que o aplicativo seja o mais abrangente possível.

O desenvolvimento nativo é a metodologia para criação de aplicativos que são executados em plataformas específicas. Conforme os dados de 2018 da International Data Corporation (IDC), os sistemas operacionais Android do Google e iOS da Apple eliminaram todos os outros sistemas operacionais móveis do mercado durante 2018. Assim, em 2019, o desenvolvimento de aplicativos móveis nativos envolve a construção de aplicativos nativos para dispositivos Android e iOS, o que significa que deverá ser feito outra versão separada do mesmo aplicativo para ir para outras plataformas. Nesta metodologia é utilizado os recursos integrados do dispositivo em que estão sendo executados, como câmera, microfone e GPS. E que geralmente proporciona uma experiência mais suave e um desempenho otimizado ao interagir com esses recursos se comparado com aplicativos multiplataforma, isso ocorre porque os aplicativos nativos são desenvolvidos com um código semelhante à linguagem de código do sistema operacional em que estão funcionando (EL-KASSAS et al., 2017).

Segundo o estudo de Matos (2017) que compara a vantagem do desenvolvimento nativo sob o desenvolvimento multiplataforma, que os aplicativos móveis nativos são mais rápidos e responsivos do que os aplicativos híbridos, tendo em vista que isso é essencialmente importante para aplicativos

centrados no desempenho, como jogos e aplicativos com muitos gráficos. Como os *apps* nativos são desenvolvidos utilizando SDK's nativas, suas UI's parecem consistentes com sua plataforma. Isso garante uma melhor experiência do usuário, pois não há discrepâncias entre o SO e o *design* do aplicativo. Logo, aplicativos nativos apresentam uma maior fluidez e um desempenho superior em relação aos aplicativos multiplataforma. Visto que usam os elementos de *layout* nativo de cada plataforma, consequentemente possuem acesso direto a todos os recursos que o dispositivo oferece, sem a necessidade de *plug-ins* e *middleware*, o que diminuem a desempenho do aplicativo. Para desenvolver *apps* para o Android usam-se as linguagens de programação Java ou o Kotlin, enquanto para o iOS utiliza-se a linguagem Swift.

Conforme Madeira (2020) a desvantagem do desenvolvimento nativo está no fato de ter que desenvolver um aplicativo para cada tipo de sistema operacional móvel, isso pode afetar o fato do aplicativo estar disponível em várias plataformas, em casos que não aja possibilidade de ter-se uma equipe de desenvolvedor para cada plataforma. Mas, resumidamente, o foco do desenvolvimento nativo é gerar *apps* com a maior compatibilidade possível e que proporcione o melhor desempenho que o dispositivo possa oferecer.

## 2.7 Flutter

O Flutter (Figura 5) é um *framework* poderoso utilizado no desenvolvimento multiplataforma que tem como linguagem de programação o Dart. Segundo Marinho (2020), flutter é o *toolkit* (*kit* de ferramentas) de IU do Google para criar aplicativos modernos e nativamente compilados para dispositivos móveis (Android e iOS), *web* e *desktop* a partir de um único código-base”.

Segundo Zammetti (2019) o Flutter existe desde 2015, quando o Google o introduziu, e permaneceu em fase beta antes de seu lançamento oficial em 2017. Desde então, Flutter tem se tornado cada vez mais popular.

De acordo com Alessandria (2020), uma aplicação Flutter é feita de *widgets*, e *widgets* são a descrição de uma parte da interface do usuário. Cada interação do usuário, e tudo o que o usuário vê ao navegar seu aplicativo, é feito de *widgets*. O próprio aplicativo é um widget. Por isso, um dos termos mais utilizado ao trabalhar com Flutter, é que "no Flutter quase tudo é um *Widget*".

Conforme Marinho (2020) em seu livro, pode-se observar que os *widgets* funcionam de maneira semelhante ao React. Onde um *widget* usa diferentes componentes para descrever a aparência da IU, em que eles podem ser com ou sem estado. Em componentes com estado, o *widget* é reconstruído devido às mudanças de estado, para acomodar o novo estado.

Figura 5 – Exemplo de aplicativo Flutter.



**Fonte:** Autor.

Corazza (2018) relata que no Flutter, todo fluxo de desenvolvimento da interface do usuário é baseada no design. Onde tem-se *widgets* para definir todo tipo de componente visual, como: botões, campos de formulários, fontes, cores além de *widgets* que definem margens e espaçamentos entre os componentes. Napoli (2019) ainda distingue os *widgets* em duas categorias no que diz respeito a construção da interface no usuário, sendo elas *StatelessWidget* e  *StatefulWidget*. O  *StatefulWidget* é utilizado quando o estado de um componente não muda, e o  *StatefulWidget* é utilizado quando o estado de um componente muda. Cada *stateless* ou *stateful* *widget* tem um método de construção com um *buildContext* que trata da localização do *widget* na árvore de *widgets*. Os objetos *buildContext* são na realidade objetos de elemento, ou seja, uma instanciação do *widget* em um local na árvore. Conforme a

Um dos pontos positivos de se utilizar o Flutter é o recurso de *hot reload*, ele permite que alterações possam ser feitas no código em tempo real, sem reiniciar o processo de construção, isso é feito ao digitar 'r' no mesmo console em que executou o *flutter run* comando para iniciar o aplicativo (MARINHO, 2020).

Segundo o site do Flutter (2021)a grandes corporações em todo mundo estão criando aplicativos com o Flutter, como o Google, Nubank, ebay, BMW, Alibaba e o Capital One. As vantagens que ele traz para as equipes de desenvolvimento o tornam um candidato promissor para a tecnologia móvel preferida em um futuro próximo. Além de que, a documentação do Flutter é muito bem detalhada e de fácil compreensão, com exemplos fáceis para casos de uso básicos (MARINHO, 2020).

## 2.8 Dart

A linguagem de programação utilizada pelo Flutter é o Dart. Dart é uma linguagem de código aberto e propósito geral, ou seja, pode ser usado para construir aplicações *web*, trabalhar do lado do servidor, criar aplicações *desktop*, mas seu principal destaque está no desenvolvimento de aplicativos móveis ([NAPOLI, 2019](#)).

O Dart foi criada pelo Google para uso interno como alguns dos seus grandes produtos. Foi disponibilizado publicamente em 2011, Dart é utilizado para construir aplicações móveis, web, e servidores. Dart é uma linguagem de programação orientada a objetos, e é baseada em classes, sua sintaxe se baseia no C e tem muita semelhança com as linguagens C, C++, Swift, Kotlin e Java ([NAPOLI, 2019](#)).

Conforme Marinho ([2020](#)) fala em seu livro, Dart transpila o código para uma linguagem comprehensível para máquinas antes da execução, o que fornece estabilidade no comportamento no código. Além de proporcionar um tempo de inicialização mais rápido quando o aplicativo é aberto.

A instalação do Dart também conta com uma VM para executar os arquivos *.dart* a partir de terminal de linha de comando. Os arquivos Dart, que são necessários para a integração entre os SO nos aplicativos Flutter, são compilados e empacotados em um arquivo binário (.apk ou .ipa) e carregados nas lojas de aplicativos.

Os principais benefícios de utilizar o Dart como linguagem de programação são:

- Dart utiliza o conceito ahead-of-time (AOT), ou seja, é compilado antes do tempo para código nativo, tornando a aplicação Flutter mais rápida. Em outras palavras, não há intermediário para interpretar uma linguagem para outra, e não há pontes. A compilação AOT é utilizada na compilação do arquivo binário (.apk ou .ipa) para ser carregado nas lojas de aplicativos, tal como Apple App Store e Google Play ([NAPOLI, 2019](#));
- Dart também usa o princípio just-in-time (JIT), o que o torna mais rápido para mostrar as alterações no código, tal como a funcionalidade de *hot reload* do Flutter, que possibilita ver as modificações em tempo real. A compilação JIT também é utilizada na depuração da aplicação quando está em execução no simulador/emulador ([NAPOLI, 2019](#));
- A renderização Flutter funciona a 60 frames por segundo (fps) e 120fps para dispositivos que são capazes de rodar a 120Hz. Isso quer dizer que quanto mais fps, mais suave é a aplicação ([NAPOLI, 2019](#)).

# 3 Metodologia

Este capítulo visa apresentar a metodologia utilizada tanto na modelagem quanto no desenvolvimento do aplicativo. A fim de uma melhor compreensão, todas as sessões serão representadas como um ciclo de desenvolvimento de software, onde terá as seguintes etapas: Comunicação, Planejamento, Modelagem, Desenvolvimento e Teste.

## 3.1 Tipo de Pesquisa

A metodologia de pesquisa utilizada neste trabalho constitui-se aplicada e bibliográfica. Segundo os autores Prodanov e Freitas (2013), pesquisa de natureza aplicada "objetiva gerar conhecimentos para aplicação prática dirigidos à solução de problemas específicos. Envolve verdades e interesses locais".

Já no que tange os procedimentos de uma pesquisa bibliográfica, ainda segundo os autores Prodanov e Freitas (2013), são feitas a partir de obras que já foram publicadas, e tem como "objetivo de colocar o pesquisador em contato direto com todo material já escrito sobre o assunto da pesquisa".

## 3.2 Materiais de Hardware e Software

Para o desenvolvimento do aplicativo, foram utilizados as seguintes configurações de hardware, conforme detalhados na Tabela 3.

Tabela 3 – Recursos de hardware utilizados.

DESCRIÇÃO	FABRICANTE	VERSÃO/MODELO
Notebook	Acer	Aspire A515-51G-71CM
SO Windows	Microsoft	Windows 10 Pro 64x
Processador	Intel	Core i7-7500U 2.90 GHz
Memória RAM	Corsair	12 Gb DDR4
SSD	WD Green	240 Gb, m.2
Smartphone	Asus	ZenFone 4 ZE554KL 4 GB RAM

**Fonte:** Autor.

E para emular e codificar o aplicativos, foram utilizados os seguintes softwares, conforme detalhados na Tabela 4.

Tabela 4 – Ferramentas de softwares utilizados.

DESCRIÇÃO	FABRICANTE	VERSÃO/MODELO
IDE	Visual Studio Code	1.62.2
Emulador Android	Android Studio	Pixel 4 API 30 Android 11

**Fonte:** Autor.

### 3.3 Etapa de Comunicação, Planejamento e Modelagem do Aplicativo

Esta seção visa esclarecer os princípios de comunicação, planejamento e modelagem que nortearam o desenvolvimento do aplicativo deste trabalho.

#### 3.3.1 Comunicação

Durante a etapa de Comunicação, a equipe identifica, reúne e define problemas, requisitos, solicitações e expectativas atuais do cliente relacionados ao aplicativo ou serviço de software ([PRESSMAN; MAXIM, 2016](#)).

A comunicação com os *stakeholders* ocorreu durante a disciplina de estágio supervisionado, onde foi realizado um estudo com o intuito de verificar a viabilidade da construção do aplicativo aqui projetado. Para validar esse estudo, foi desenvolvida uma pesquisa no formato de questionário o qual foi elaborado através da plataforma Google *Forms*. Os principais objetivos eram determinar se era ou não viável o desenvolvimento do aplicativo em questão, bem como validar e coletar sugestões com relação às funcionalidades do aplicativo ([LIMA, 2020](#)).

#### 3.3.2 Planejamento e Modelagem

Nas etapas de planejamento e modelagem, é onde os requisitos coletados são analisados e utilizados na arquitetura do software para a modelagem dos diagramas que auxiliaram no desenvolvimento do sistema ([PRESSMAN; MAXIM, 2016](#)).

Algumas atividades relacionadas à fase de coleta de requisitos podem envolver a criação de especificações de software, a criação de um plano detalhado, documentação, rastreamento de problemas e planejamento de projeto ou produto, incluindo a alocação dos recursos corretos.

##### 3.3.2.1 Elicitação e Análise de Requisitos do Aplicativo

A coleta de requisitos é um processo que determina, documenta e gerencia as necessidades e requisitos para se atender as expectativas das partes interessadas, sendo essa etapa, a primeira para atender aos objetivos do gerenciamento do projeto. A documentação que ocorre no processo de coleta de requisitos é considerada importante, pois fornece a base para definir e gerenciar o escopo do projeto. Existem várias técnicas para se coletar requisitos, algumas delas são feitas através de entrevistas, questionários, observação e *brainstorming*. A técnica que foi utilizada para eliciar os requisitos do aplicativo, conforme mencionada nos parágrafos acima, foi um questionário elaborado no Google *Forms*. Essa técnica consiste em um documento com um conjunto pré-definido de questões objetivas com suas respetivas opções, entregue para os *stakeholders* para obter suas respostas, das quais serão analisadas e a partir delas será possível determinar o grau de satisfação do participante em relação a determinado requisito.

O questionário ficou aberto para o público durante o período de 1 (um) mês e contou com a participação de 63 (sessenta e três) acadêmicos do ensino superior dos mais diversos cursos e instituições de ensino. E com ele, foi possível estabelecer os requisitos funcionais do aplicativo.

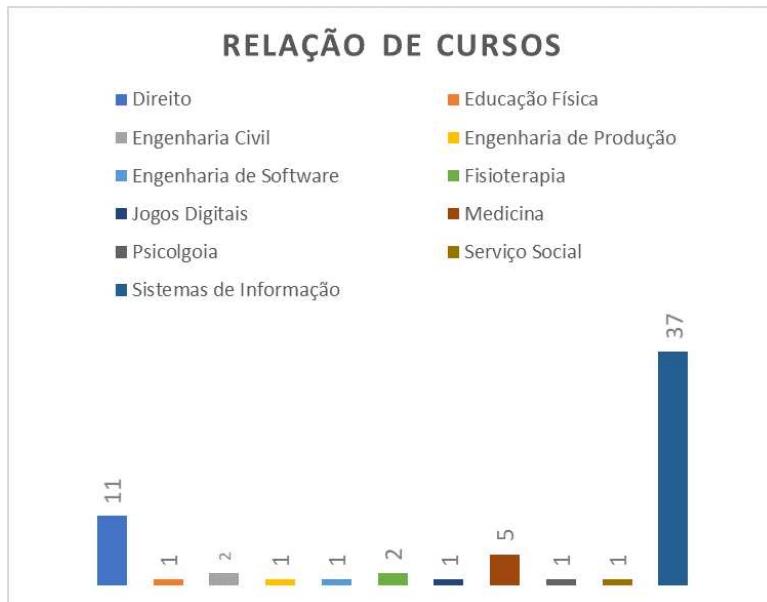
### 3.3.2.1.1 Perfil dos candidatos

Na primeira parte do questionário foram solicitados alguns dados dos usuários antes de responderem de fato às perguntas acerca do aplicativo. Foram colhidos o nome, instituição de ensino, curso, período, sistema operacional móvel utilizado pelo entrevistado e por fim foi perguntando se o acadêmico tinha interesse em participar ativamente do desenvolvimento desse projeto, dando sua opinião sobre os protótipos e versões de teste do aplicativo.

Houve uma quantidade significativa de instituições de ensino em relação aos entrevistados, ao todo foram 13 instituições, dentre elas estavam: Universidade Estadual do Tocantins, Universidade Federal de Goiás, Universidade Federal de Minas Gerais, Universidade Federal do Rio de Janeiro, Universidade Federal do Tocantins, Universidade de Ribeirão Preto – Guarujá e Universidade de Buenos Aires.

Assim, como pode ser observado no gráfico da Figura 6, houve também uma grande variedade de cursos. Seguindo a ordem do gráfico, são eles: Direito, Educação Física, Engenharia Civil, Engenharia de Produção, Engenharia de Software, Fisioterapia, Jogos Digitais, Medicina, Psicologia, Serviço Social e Sistemas de Informação.

Figura 6 – Relação de cursos.



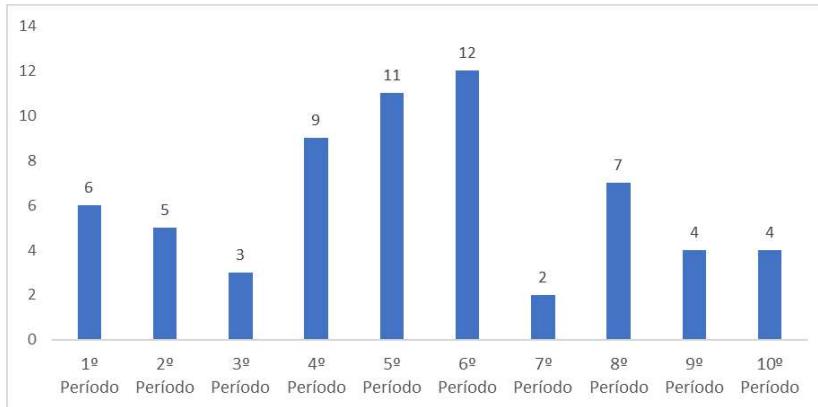
**Fonte:** Autor.

Esse resultado é essencial para determinar o quão abrangente o questionário foi. E para ter embasamento da diversidade de público o qual o aplicativo pode alcançar. Essa abrangência pode ser observada que há cursos de diversas áreas, tais como de TI, engenharias, área da saúde e área de humanas.

Outro aspecto questionado aos entrevistados foi a respeito do período letivo em que se encontravam, Figura 7.

Como pode ser observado, houve respostas de acadêmicos que estavam desde o 1º Período até o 10º Período, a maturidade dos candidatos é notável, já que os períodos que obtiveram mais respostas foram o 4º, 5º, 6º e 8º períodos.

Figura 7 – Gráfico de períodos letivos dos entrevistados.



**Fonte:** Autor.

### 3.3.2.2 Requisitos do aplicativo

Os requisitos de software são a descrição dos recursos e funcionalidades do sistema de destino. Os requisitos transmitem as expectativas dos usuários do produto de software ([SOMMERVILLE et al., 2008](#)).

O questionário que fora elaborado, era composto por 12 perguntas que estavam diretamente ligadas aos requisitos do aplicativo, e a partir das respostas que foi coletada, foi possível determinar se um requisito era ou não útil para os acadêmicos. Todas as perguntas e suas descrições podem ser vistas na Seção A.3 no anexo deste trabalho, e os dados referentes as respostas dos participantes da pesquisa podem ser observadas no Anexo B.

Ainda sobre os requisitos de um software, eles podem ser classificados como requisitos funcionais e requisitos não funcionais.

#### 3.3.2.2.1 Requisitos funcionais

Os requisitos funcionais são os requisitos que o usuário final exige especificamente como recursos básicos que o sistema deve oferecer. Todas essas funcionalidades precisam ser necessariamente incorporadas ao sistema como parte do contrato. Estes são representados ou declarados na forma de entrada a ser fornecida ao sistema, a operação realizada e a saída esperada. São basicamente os requisitos indicados pelo usuário que podem ser vistos diretamente no produto final, ao contrário dos requisitos não funcionais ([SOMMERVILLE et al., 2008](#)).

O compilado dos requisitos funcionais coletados durante o processo de elicitação de requisitos do aplicativo estão detalhadas na Seção A.1 do apêndice deste projeto.

#### 3.3.2.2.2 Requisitos não funcionais

Já os requisitos não funcionais são basicamente as restrições de qualidade que o sistema deve satisfazer conforme o contrato do projeto. A prioridade ou extensão em que esses fatores são implementados varia de um projeto para outro. Eles também são chamados de requisitos não comportamentais ([SOMMERVILLE et al., 2008](#)).

O compilado dos requisitos não funcionais definido na modelagem deste aplicativo estão detalhadas na Seção A.2 do apêndice deste projeto.

### 3.3.2.3 Modelagem dos Diagramas do Aplicativo

Nesta seção será abordado a modelagem dos diagramas do aplicativo utilizando diagramas da UML. UML é a abreviação de *Unified Modeling Language* ou Linguagem de Modelagem Unificada, é uma linguagem visual que ajuda os desenvolvedores de software a visualizar e construir novas aplicações.

UML (Unified Modeling Language – linguagem de modelagem unificada) é “uma linguagem-padrão para descrever/documentar projeto de software. A UML pode ser usada para visualizar, especificar, construir e documentar os artefatos de um sistema de software-intensivo” ([PRESSMAN; MAXIM, 2016](#)).

A UML não se trata de uma linguagem de programação, mas sim um conjunto de regras especificamente para desenhar diagramas. Embora existam vários tipos de diagramas da UML, cada diagrama possui elementos como pacotes, classes e associações que são representados com seus próprios gráficos. Neste projeto serão apresentados três tipos desses diagramas, são eles: Diagrama de Cado de Uso, na Subseção [3.3.2.3.1](#), Diagrama de Classe, na Subseção [3.3.2.3.2](#) e o Diagrama de Componentes, na Subseção [3.3.2.3.3](#).

#### 3.3.2.3.1 Diagrama de caso de uso

Um diagrama de Caso de Uso da UML, é a forma primária de requisitos de sistema para um novo software que está em desenvolvimento. Os casos de uso especificam o comportamento esperado (o que), e não o método exato de fazê-lo (como). Os casos de uso, uma vez especificados, podem ser denotados como representação textual e visual. O diagrama caso de uso serve para projetar um sistema da perspectiva do usuário final ([PRESSMAN; MAXIM, 2016](#)). Este diagrama pode ser encontrado na subseção [4.1.1](#) dos resultados.

#### 3.3.2.3.2 Diagrama de classe

Na engenharia de software, um diagrama de classes na *Unified Modeling Language* (UML) é um tipo de diagrama de estrutura estática que descreve a estrutura de um sistema, mostrando as classes do sistema, seus atributos, métodos e os relacionamentos entre os objetos. As classes em um diagrama de classes correspondem às classes no código-fonte. O diagrama mostra os nomes e atributos das classes, as conexões entre as classes e, às vezes, também os métodos das classes ([PRESSMAN; MAXIM, \(2016\)](#)). Esta representação pode ser vista na subseção [4.1.2](#) dos resultados.

#### 3.3.2.3.3 Diagrama de componentes

O diagrama de componentes da UML é um diagramas que é voltado para modelagem do software baseando-se em componentes, e tem por finalidade representar os componentes e o seus relacionamentos em forma de artefatos dos quais os componentes são feito ([VARGAS, 2007](#)). Este diagrama está localizado na subseção [4.1.3](#) dos resultados.

## 3.4 Etapa de Desenvolvimento do Aplicativo

Na etapa de desenvolvimento de software, é onde a codificação e a construção do aplicativo realmente acontecem. Utilizando-se uma linguagem de programação que geralmente é escolhida de acordo com o tipo de software que está sendo desenvolvido.

Para Pressman (2016), "a atividade de construção engloba um conjunto de tarefas de codificação e testes que conduzem ao software operacional pronto para ser entregue ao cliente e ao usuário final."

### 3.4.1 Justificativa para a Adoção do Flutter e do Dart no Desenvolvimento do Aplicativo

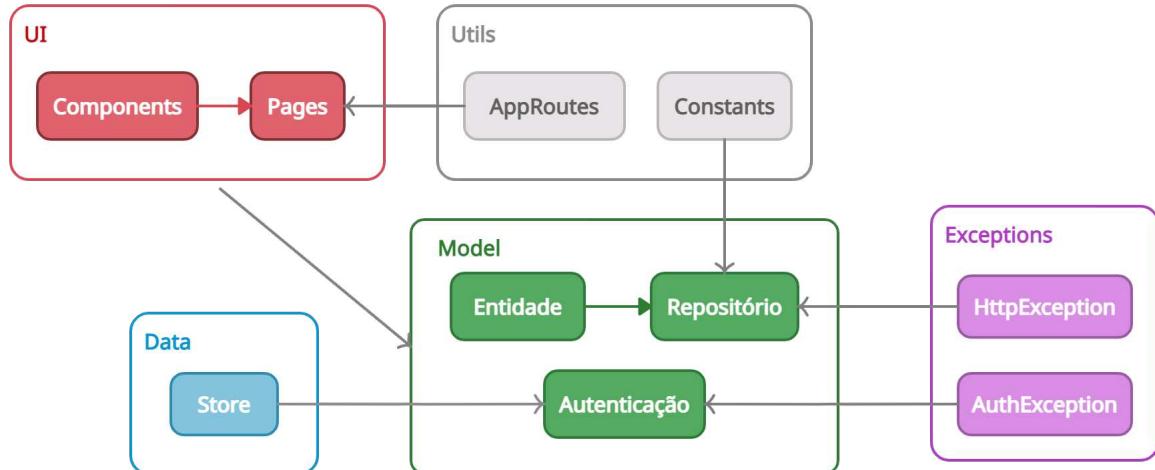
Para o desenvolvimento do aplicativo foi utilizado o *framework* Flutter para construção das telas e de toda parte visual do aplicativo e para o *back-end* foi utilizado a linguagem de programação Dart. A justificativa para utilização dessas tecnologias, deve-se principalmente ao fato da documentação tanto do Flutter quanto do Dart serem muito bem detalhada e de fácil compreensão, com vários exemplos práticos e fáceis para casos de uso básicos.

A razão para qual não foi escolhido outros *frameworks* com foco em desenvolvimento multiplataforma, tal como React Native, foi em virtude das fronteiras que o Flutter vem rompendo, principalmente na extensão do conceito de multiplataforma. Já que recentemente o Flutter (2021)b anunciou que agora além do suporte ao Android, iOS e Web, também presta suporte a desktop, o que significa os aplicativos construídos em Flutter podem ser estendidos para Windows, macOS e Linux.

### 3.4.2 Arquitetura e Estrutura do Projeto

A aplicação foi construída em 5 camadas de regras de negócio, de modo a tornar o projeto mais legível e manutenível. Sendo as camadas: *Data*, *Exceptions*, *Model*, *UI*, e *Utils*. Conforme Figura 8.

Figura 8 – Arquitetura da aplicação.



**Fonte:** Autor.

- **Data:** Em *data*, está englobada a lógica necessária para ser utilizada a autenticação da aplicação.

- **Exceptions:** Camada responsável por conter as exceções da aplicação, tanto as que estão relacionadas a autenticação quanto as de regra de negócio.
- **Model:** Esta camada contém todas as entidades com seus atributos, bem como a lógica para se conectar ao banco de dados e realizar as ações *create, read, update* e *delete* (CRUD).
- **UI (*User Interface*):** Nesta camada está contida toda parte visual da aplicação, como tela de autenticação, formulários e listagem. Também contém *widgets* personalizados, estes componentes representam os itens que serão apresentados nas telas de listagem, bem como o *widget* responsável pela exibição do menu *drawer* da aplicação.
- **Utils:** Camada responsável por conter todas as constantes relacionadas às rotas do aplicativo e URL base de comunicação com o banco de dados.

### 3.4.3 Ambiente de Desenvolvimento

Para o ambiente de desenvolvimento do aplicativo, é conforme a Tabela 4. Foi utilizado o editor de código-fonte Visual Studio Code<sup>1</sup>, que pertence à Microsoft, e é totalmente gratuito. O Visual Studio Code está disponível para vários SO's, tais como Windows, Linux e macOS. Embora o editor seja relativamente leve, ele inclui recursos poderosos que tornaram o VS Code uma ferramenta de ambiente de desenvolvimento extremamente completa, oferecendo suporte a diversas linguagens de programação, como Java, Python, PHP e C, e também presta suporte a linguagens com foco na *web*, como HTML, JavaScript e CSS. Um dos grandes diferenciais do VS Code é o suporte a extensões, o que tornam o editor mais completo. Em seu *marketplace*, dentre os pacotes de extensões da loja, há também as extensões oficiais que prestam suporte ao Flutter e ao Dart.

No desenvolvimento, foi utilizado tanto um dispositivo Android real, conforme a Tabela 3, quanto o Android *Emulator*<sup>2</sup>, pertencente à Google, que se trata de uma ferramenta capaz de simular vários dispositivos Android virtuais com *software* e *hardware*, ou seja, com ele possível emular o recebimento de chamadas e mensagens de texto, especificar o local do dispositivo através do GPS, simular rotação e outros sensores de hardware, acessar a Google Play Store e muito mais.

### 3.4.4 Processo de Desenvolvimento

De modo a manter controle sobre o fluxo de desenvolvimento e das atividades a serem desempenhadas, a construção desta aplicação foi constituída de *sprint's*, que conforme Sommerville (2011), é uma iteração de desenvolvimento que normalmente dura entre 2 e 4 semanas. As *sprint's* permitem previsibilidade e garantem a inspeção e adaptação do progresso em direção a uma aeta do produto a cada ciclo.

A primeira *sprint* foi composta pelas tarefas de desenvolvimento da parte visual do aplicativo, onde foi implementado todas as telas referentes aos formulário e listagem de itens, conforme os requisitos elicitados. Durante esta etapa, foi utilizado o *framework* Flutter para construção das telas, bem como validações de campos em brancos e quantidade de caracteres inseridas. Durante este ciclo foram implementados os requisitos funcionais: RF004, RF005, RF012, RF013, RF016, RF017, RF020, RF021, RF024, RF025, RF035, RF036, RF043, RF044. Contabilizando 14 dos 45 requisitos funcionais elicitados.

A segunda *sprint* foi dedicada a implementação das classes modelos que servirão de base para implementação dos repositórios, onde de fato irá ocorrer a interação com o banco de dados NoSQL Firebase

---

<sup>1</sup> Disponível em: <<https://code.visualstudio.com/>>

<sup>2</sup> Disponível em: <<https://developer.android.com/studio>>

*Realtime Database*, e local onde irá conter os métodos de ação para criar, alterar, listar e deletar itens na aplicação. Durante está etapa foi utilizado para o *back-end* a linguagem de programação Dart. Neste ciclo foi possível implementar os requisitos funcionais: RF003, RF006, RF011, RF014, RF015, RF018, RF019, RF022, RF023, RF022, RF026, RF034, RF037, RF042, RF045. Totalizando 15 dos 45 requisitos funcionais elicitados.

Na terceira *sprint* e final, foram implementadas as regras de negócios mais complexas da aplicação, como autenticação, notas etc. A autenticação do aplicativo foi implementada utilizando o Firebase *Authentication*, conforme o Google (2021)a proprietária desse recurso, o *Authentication* provém recursos de *back-end* fáceis de ser utilizados e conta com bibliotecas de UI prontas para serem utilizadas, possibilitando que os usuários possam realizar *login* na aplicação utilizando como e-mail/senha ou provedores de identidade federados, como Google, Facebook, entre outros. Essa implementação satisfaz o requisito funcional RF001 com a utilização de e-mail/senha para autenticação no app.

### 3.4.5 Armazenamento de Dados

O banco de dados utilizado neste projeto foi o *Realtime Database* da Google, conforme a própria empresa Google (2021)b, o Firebase *Realtime Database* é um banco de dados NoSQL a partir do qual pode-se armazenar e sincronizar os dados de aplicações em tempo real. Dentre as grandes vantagens de utilizar esse banco, podem ser destacadas a sincronização em tempo real dos dados armazenados e o uso de cache local no dispositivo para aplicar e armazenar alterações, logo, quando o dispositivo voltar a ter conexão com a internet, os dados locais são sincronizados automaticamente.

O *Realtime Database* se baseia no conceito de chave-valor, conforme a Tabela 5 descreve, os tipos de dados e os campos são armazenados no banco, junto com os dados inseridos .

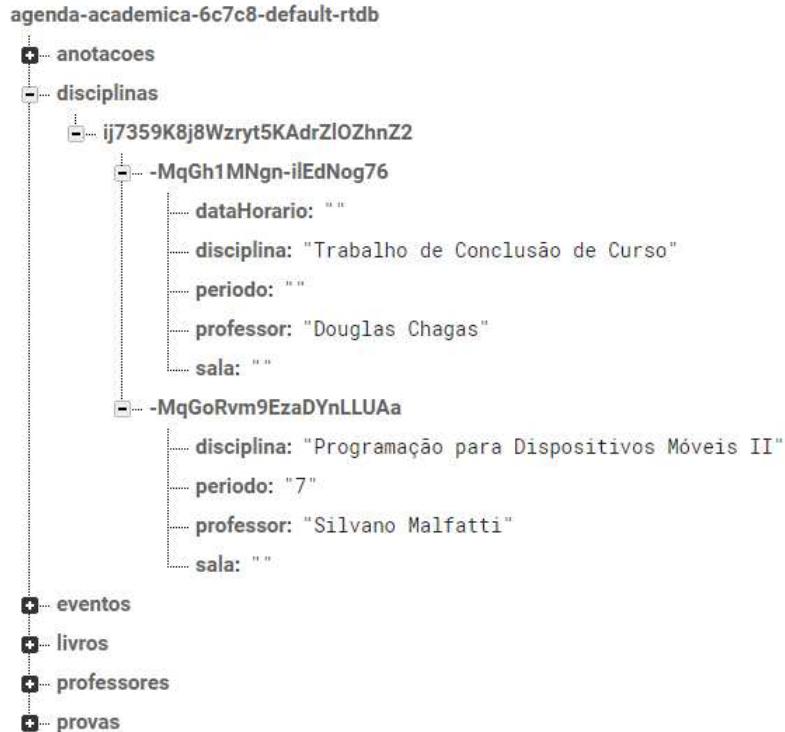
Tabela 5 – Descrição do documento contendo as informações sobre disciplinas.

<b>Tipo</b>	<b>Campo</b>	<b>Descrição</b>
Date	dataHorário	Horário de aula
String	disciplina	Nome da disciplina
String	periodo	Número do período da disciplina
String	professor	Professor da disciplina
String	sala	Local onde ocorre a disciplina.

**Fonte:** Autor.

A disposição dos dados no banco estão representados na Figura 9. Onde existem todas as coleções, e cada coleção passa a ser integrada por um usuário com o seu ID único a partir do momento que os dados desta coleção foram salvos na aplicação.

Figura 9 – Coleção de documento do aplicativo.



**Fonte:** Autor

A seguir será exibido parte do código-fonte da classe DisciplinaList que é responsável por gerenciar a entidade Disciplina. A classe DisciplinaList é responsável pela comunicação com o banco de dados via API REST através de requisições HTTP (*Hypertext Transfer Protocol*). O protocolo HTTP, segundo Tanenbaum (2003), é um protocolo de camada de aplicação projetado para transferir informações entre dispositivos em rede sendo executado no topo de outras camadas da pilha de protocolo de rede. Este protocolo provém métodos, que também são conhecidos como verbos HTTP, que indicam a ação que a solicitação HTTP espera do servidor consultado, sendo os principais verbos: GET que solicita um recurso específico do servidor; POST que cria um recurso; PUT que atualiza um recurso e por fim o DELETE que remove determinado recurso.

Na classe DisciplinaList (Código 5) há métodos assíncronos responsáveis por realizar as ações HTTP, como mostrado no trecho do código abaixo, responsável por inserir os dados passado pelo usuário no banco de dados através do verbo POST.

---

```

1 Future<void> addDisciplina(Disciplina disciplina) async {
2     final response = await http.post(
3         Uri.parse('${Constants.DISCIPLINA_BASE_URL}.json?auth=$_token'),
4         body: jsonEncode(
5             {
6                 'periodo': disciplina.periodo,
7                 'disciplina': disciplina.disciplina,
8                 'dataHorario': disciplina.dataHorario,
9                 'professor': disciplina.professor,
10                'sala': disciplina.sala,
11            },
12        ),
13    );
14
15    final id = jsonDecode(response.body)[ 'name' ];
16    _disciplinas.add(Disciplina(
17        id: id,
18        periodo: disciplina.periodo,
19        disciplina: disciplina.disciplina,
20        dataHorario: disciplina.dataHorario,
21        professor: disciplina.professor,
22        sala: disciplina.sala,
23    )));
24    notifyListeners();
25 }

```

---

Código-fonte 1 – Código Dart responsável por inserir os dados de disciplina no banco.

A interação com esse método é realizado através do formulário de cadastro de disciplina, que, após o usuário inserir todos os dados no formulário e salvar, é criado um objeto da classe Disciplina e os atributos passados são armazenados no Firebase *Realtime Database*.

### 3.5 Etapa de Testes de Desempenho e Análise de Qualidade de Código

A etapa de desenvolvimento do software está dividida em várias etapas que permitem à empresa organizar sistematicamente o seu trabalho, e construir de forma eficiente um produto com as funcionalidades necessárias dentro de um prazo e orçamento especificados.

Segundo Pressman (2016), o teste é um dos processos críticos do ciclo de vida do processo de desenvolvimento de software. Ajuda as empresas a realizar uma avaliação abrangente do software e a garantir que seu produto atenda às necessidades do cliente. Nesta etapa busca-se identificar possíveis erros e bugs de qualquer software para as empresas antes de sua implementação. Se os bugs de software não forem resolvidos ou corrigidos antes da implantação, eles afetarão gravemente os negócios do cliente.

Conforme Sommerville (2011), os testes têm por objetivo verificar e validar o bom funcionamento

da aplicação através de *scripts* de simulam o uso do programa, podendo os testes serem automatizados ou manuais.

### 3.5.1 Teste de Desempenho

Em conformidade com a seção 4.3, para o teste de desempenho do aplicativo, foi utilizado a ferramenta Dart DevTools. Segundo o Flutter (2021), o DevTools nada mais é do que um conjunto de ferramentas para desenvolvedores para desenvolvedores Flutter e Dart que consiste em ferramentas de inspeção de layout, ferramentas de desempenho, ferramentas de memória e várias outras ferramentas de depuração que você possa precisar para criar aplicações eficiente e eficaz, tudo agrupado em uma única ferramenta.

DevTools é um conjunto autônomo de ferramentas executada tanto no navegador quanto na própria IDE. Eles fornecem telemetria e funcionalidade adicionais aos quais não são suficientemente fornecidas pela IDE ou por ferramenta de depuração convencional.

Com o DevTools é possível:

- Inspecionar o layout da IU e o estado de um aplicativo Flutter;
- Diagnosticar problemas de desempenho instável da IU;
- Métricas relacionadas ao consumo da CPU;
- Métricas relacionadas ao consumo da memória RAM do dispositivo;
- Ver log geral e informações de diagnóstico.

### 3.5.2 Análise de Qualidade de Código do Aplicativo

Conforme os resultados da seção 4.4, para verificar a qualidade do código do aplicativo, foi utilizado a plataforma de código aberto SonarQube, cujo objetivo é verificar continuamente a qualidade do código da aplicação durante o desenvolvimento. O plataforma não presta suporte nativamente ao Dart e ao Flutter, portanto foi necessário utilizar um *plugin*<sup>3</sup> de terceiro.

---

<sup>3</sup> Disponível em: <<https://github.com/insideapp-oss/sonar-flutter>>

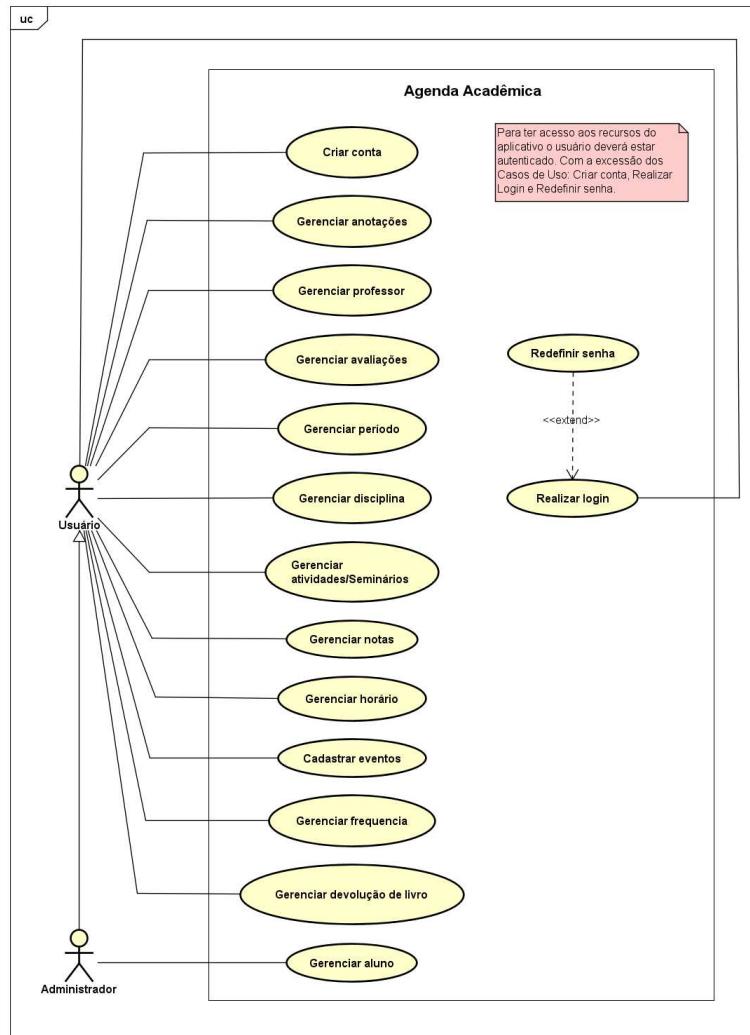
# 4 Resultados

## 4.1 Diagramas da UML

### 4.1.1 Diagrama de Caso de Uso

Este diagrama é composto por atores Administrador e Usuário, que representam o desenvolvedor do aplicativo que prestara suporte aos usuários do mesmo e os usuários finais que vão utilizar o aplicativo.

Figura 10 – Diagrama Caso de Uso - Agenda Acadêmica



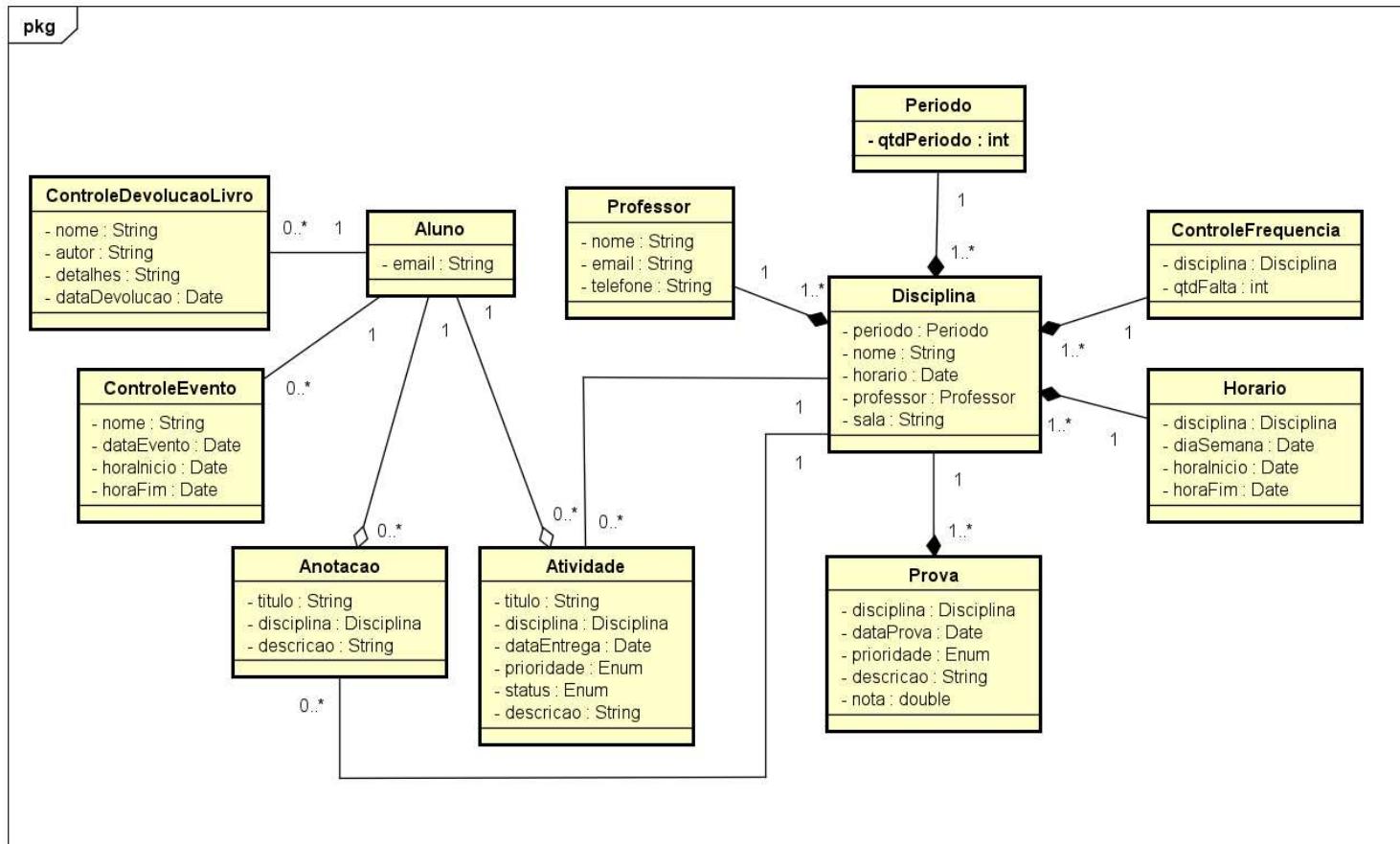
**Fonte:** Autor

O ator Administrador do aplicativo tem como caso de uso o gerenciamento dos acadêmicos que vão utilizar o aplicativo. Ele também herda todos os casos de uso de um Usuário, podendo ter acesso a todos os recursos do aplicativo. Já o ator Usuário tem como casos de uso todos os recursos do sistema, como efetuar autenticação no aplicativo, gerenciar disciplinas, professores, anotação e outros.

#### 4.1.2 Diagrama de Classe

Este diagrama produziu 11 classes, e cada uma delas possuindo seus próprios atributos e relacionamentos. A seguir será descrito a função de cada classe.

Figura 11 – Diagrama de Classe - Agenda Acadêmica



Fonte: Autor

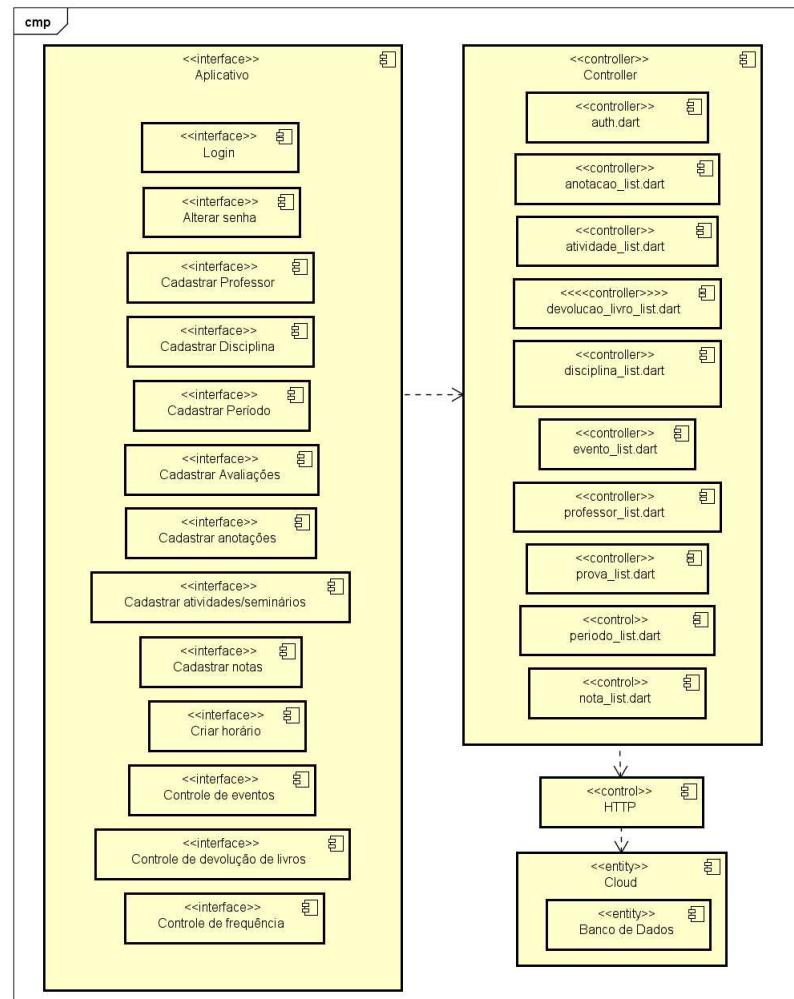
- Aluno: esta classe têm por finalidade servir como ancora para as demais classes, das quais estabelece os relacionamentos de associação com as classes ControleDevolucaoLivro e ControleEvento, como também contém os relacionamentos de agregação com as classes Anotacao e Atividade. A cardinalidade entre ControleDevolucaoLivro e ControleEvento é de associação, já com as classes Anotacao e Atividade é de agregação. Todas elas contem um Aluno, e esse Aluno pode conter 0 ou N das demais classes. A classe Aluno contém um único atributo privado que representa o e-mail do acadêmico que está se autenticando no aplicativo.
- Anotacao: classe que contém os relacionamentos de agregação com a classe Aluno e de associação com a Disciplina. Foi estabelecido os relacionamentos que possibilite que uma Anotacao pertença a um Aluno e um Aluno possa ter 0 ou N Anotacao, bem como uma Anotacao possa pertencer a uma Disciplina e uma Disciplina contenha 0 ou N Anotacao. Esta classe possui os atributos privados que têm como função detalhar uma anotação, são os atributos: titulo da anotação, qual disciplina ela pertence e um atributo para descrição.
- Atividade: esta classe contém os relacionamentos de agregação com a classe Aluno e de associação com a classe Disciplina. Foi estabelecido os relacionamentos que possibilite que uma Atividade pertença a um Aluno e um Aluno possa ter 0 ou N Atividade, bem como uma Atividade possa pertencer a uma Disciplina e uma Disciplina contenha 0 ou N Atividade. Os atributos privados dessa classe que serve para descrever uma atividade de uma disciplina são: título da atividade, disciplina a qual ela pertence, data de entrega, prioridade, *status* e descrição.
- ControleDevolucaoLivro: classe que mantém um relacionamento de associação com a classe Aluno, este relacionamento têm as cardinalidades de forma que ControleDevolucaoLivro possa pertencer a um Aluno, e um Aluno pode conter 0 ou N ControleDevolucaoLivro. Esta, contém atributos privados a fim identificar um livro: nome do livro, autor, detalhe sobre o livro e de modo a saber a data de devolução, contém o atributo dataDevolucao.
- ControleEvento: esta classe estabelece um relacionamento de associação com a classe Aluno, este relacionamento têm as cardinalidades de forma que ControleEvento possa pertencer a um Aluno, e um Aluno possa conter 0 ou N ControleEvento. A mesma, contém atributos privados que servirão para identificar um evento de interesse do aluno, os atributos são: nome do evento, data do evento e o horário de início e término.
- ControleFrequencia: classe que detém um relacionamento de composição com a classe Disciplina, cuja cardinalidade entre ambas é de que ControleFrequencia pertence a uma Disciplina e Disciplina têm um ControleFrequencia, cujos atributos privados servem para identificar a quantidade de falta em uma disciplina pelo nome da disciplina e pela quantidade de falta.
- Disciplina: esta classe estabelece os relacionamentos de composição com as classes ControleFrequencia, Horario, Periodo, Professor e Prova, cujo as cardinalidades são de que Disciplina pode conter uma das demais classes, e as outras classes podem conter 1 ou N Disciplina. Disciplina também estabelece o relacionamento de associação com as classes Anotacao e Atividade, sendo a cardinalidade de que uma Disciplina possa conter 0 ou N objetos, e as outras classes podem estar contida em uma Disciplina. Esta, contém os seguintes atributos privados: período da disciplina, nome, horário, professor e sala, cujo objetivo é caracterizar uma disciplina.
- Horario: classe que estabelece um relacionamento de composição com Disciplina, cuja cardinalidade é de que uma Disciplina está contida em um Horario e um Horario contém 1 ou N Disciplina. Os atributos privados de Horário são: disciplina, dia da semana, horário de início e fim da aula.

- Período: classe que estabelece um relacionamento de composição com Disciplina, cuja cardinalidade é de que uma Disciplina está contida em um Período e um Período contém 1 ou N Disciplina. A classe Período tem um único atributo privado cujo objetivo é armazenar a quantidade de períodos.
- Professor: esta classe detém um relacionamento de composição com Disciplina, cuja cardinalidade é de que uma Disciplina pertence a um Professor e um pode conter 1 ou N Disciplina. Os atributos privados de Professor são: nome, e-mail e telefone, e servem para caracterizar um professor.
- Prova: classe que estabelece um relacionamento de composição com Disciplina, cuja cardinalidade é de que uma Disciplina contém 1 ou N Prova e Prova contém 1 Disciplina. Esta classe possui os seguintes atributos privados: disciplina, data da prova, prioridade, descrição e nota obtida.

#### 4.1.3 Diagrama de Componentes

Este Diagrama de Componentes está constituído de 4 componentes básicos que representam o funcionamento do aplicativo. A seguir será detalhado a função de cada um.

Figura 12 – Diagrama de Componentes - Agenda Acadêmica



**Fonte:** Autor

- Aplicativo: este componente representa toda parte visual do aplicativo, local onde o usuário realizará a interação com o mesmo. Dentro deste componente há outros componentes que simbolizam as ações que podem ser realizadas no aplicativo.
- *Controller*: este componente reflete as ações internas do *back-end* que serão realizadas pelo aplicativo, essas ações correspondem ao ato de listar, cadastrar, alterar e deletar itens do aplicativo.
- HTTP: o componente HTTP, conforme explicado na Seção 3.4.5, contém os chamados verbos (HTTP) que irão requisitar recursos ao banco de dados. Este componente serve para intermediar a comunicação entre o componente *Controller* e o componente *Cloud*.
- *Cloud*: este componente é responsável por representar o armazenamento dos dados feito pelo aplicativo. A interação com o mesmo ocorre como descrito acima, através de requisições HTTP.

## 4.2 Aplicativo Desenvolvido

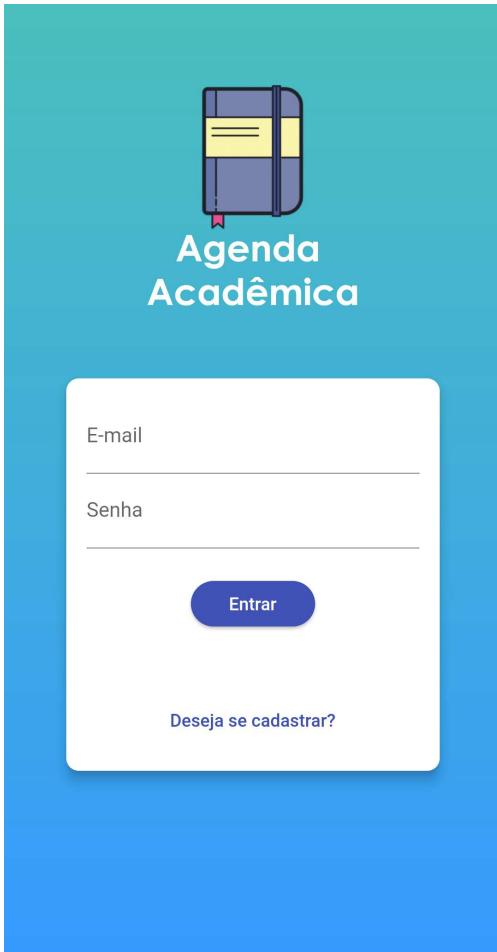
Esta seção tem por objetivo apresentar a interface do aplicativo desenvolvido com a explicação do funcionamento e importância de cada tela, assim como será apresentado uma seção contendo fragmentos de código que foi codificado durante o desenvolvimento do aplicativo. Devido a carência tanto de um emulador iOS quanto de um dispositivo real, a demonstração da interface do aplicativo será feita utilizando o SO Android, porém há total compatibilidade com ambas plataformas.

### 4.2.1 Interface do Aplicativo Desenvolvido

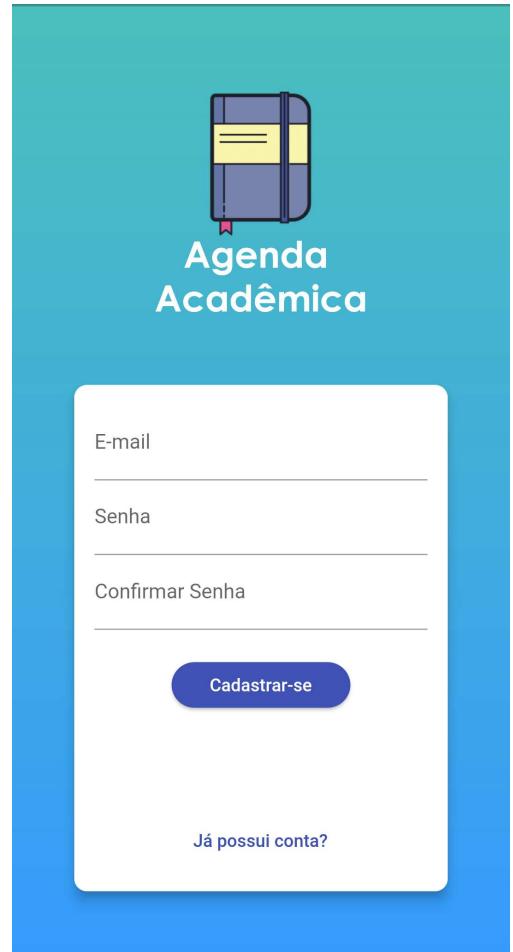
A tela de Login (Figura 13a) é o contato inicial que o usuário terá com o aplicativo. A partir dela é possível efetuar *login* pelo e-mail cadastrado ou ir para o formulário de cadastro no aplicativo.

Figura 13 – Tela de *Login* e cadastro de usuário

(a) Tela de *Login*.



(b) Tela de cadastro de usuário.



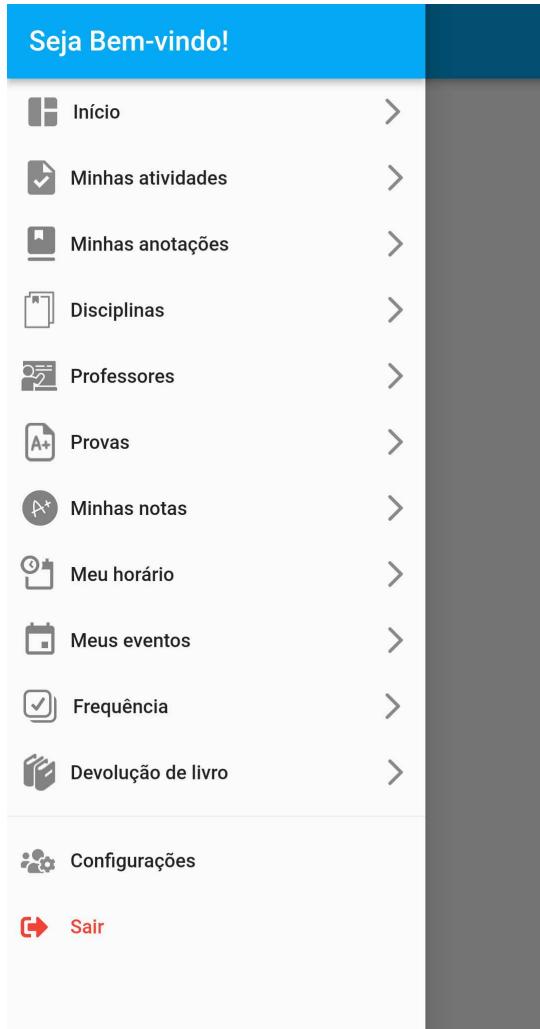
**Fonte:** Autor.

A tela de cadastro no aplicativo (Figura 13b), será por onde o usuário poderá criar sua conta para acessar o aplicativo. Para isso o usuário deverá fornecer algumas informações, como e-mail e senha.

Conforme a Figura 14a, a partir do menu do aplicativo é possível navegar para as outras telas do aplicativo, assim como como **Sair** do aplicativo.

Figura 14 – Menu do aplicativo e tela de listagem de disciplina.

(a) Tela de *Login*.



(b) Tela de cadastro de usuário.

Gerenciar Disciplinas		
Trabalho de Conclusão de Curso		
Douglas Chagas		
Simulação de Sistemas de Informação		
Marcelo Ribeiro		
Infraestrutura de Redes como Serviços		
Jocivan Suassone		
Programação para Dispositivos Móveis II		
Silvano Malfatti		

**Fonte:** Autor.

De acordo com a Figura 14b, na tela de gerência de disciplinas é possível visualizar todas as disciplinas, bem como o professor responsável por ministrar a mesma. Ainda nesta tela, é possível ir para tela de cadastro de uma nova disciplina, alterar os dados ou exclui-la.

Como ilustrado na Figura 15a, na tela de cadastro de disciplina é possível adicionar uma nova disciplina ao aplicativo, onde o usuário irá fornecer o período, nome, data, professor responsável e o local. O objetivo é utilizar essa disciplina na construção do horário, no controle de frequência e de notas.

Figura 15 – Formulário de cadastro de disciplina e tela de listagem de professor.

(a) Formulário de cadastro de disciplina.

O formulário de cadastro de disciplina tem uma barra azul superior com o título "Formulário de Disciplina". Abaixo, há campos para inserir informações: "Período" (com ícone de calendário), "Disciplina" (com ícone de lista), "Data/Horário" (com ícone de calendário), "Professor" (com ícone de usuário) e "Sala/Laboratório" (com ícone de localização).

(b) Tela de listagem de professor.

A tela de gerenciamento de professores tem uma barra azul superior com o título "Gerenciar Professores" e um ícone de mais. Abaixo, há uma lista de professores com suas respectivas informações e ícones de edição e exclusão:

Nome	E-mail	Opções
Douglas Chagas	douglas.cs@unitins.br	
Marcelo Ribeiro	marcelo.ro@unitins.br	
Silvano Malfatti	silvano.mm@unitins.br	

**Fonte:** Autor.

Tal como a Figura 15b, na tela de gerência de professores é possível visualizar todos os professores, bem como seu e-mail. A partir desta tela, é possível ir para tela de cadastro de um novo professor, alterar seus dados ou exclui-lo.

De acordo com a Figura 16a, no formulário de cadastro de professor é possível adicionar um novo professor ao aplicativo, onde o usuário irá fornecer o nome, e-mail e telefone. O objetivo é manter os dados de contato de um professor acessível para consultado sempre que houve necessidade.

Figura 16 – Formulário de cadastro de professor e tela de listagem de livros.

(a) Formulário de cadastro de professor.

The screenshot shows a mobile application interface for professor registration. At the top is a blue header bar with a back arrow icon and the text "Formulário de Professor". Below the header is a form with three fields: "Nome" (Name) with an icon of a person, "E-mail" (Email) with an icon of an envelope, and "Telefone" (Phone) with an icon of a smartphone. Each field has a horizontal line below it for input.

(b) Tela de listagem de livros.

The screenshot shows a mobile application interface for managing book returns. At the top is a blue header bar with a menu icon, the text "Gerenciar Devolução de ...", and a plus sign icon. Below the header is a table with two rows of data. Each row contains a title, an author, and edit/delete icons. The first row shows "Código Limpo" by "Robert Cecil Martin". The second row shows "Refatoração" by "Martin Fowler, Kent Beck".

Código Limpo	Robert Cecil Martin		
Refatoração	Martin Fowler, Kent Beck		

**Fonte:** Autor.

Conforme a Figura 16b, na tela de gerência de devolução de livro é possível visualizar todos os livros cadastrados, bem como seu autor. A partir desta tela, é possível ir para tela de cadastro de um novo livro, alterar seus dados ou exclui-lo.

Em conformidade com a Figura 17, é possível adicionar um novo livro ao aplicativo, onde o usuário irá fornecer o nome, autor, detalhes e a data de devolução. O objetivo é que o usuário seja notificado quando a data de devolução de um livro estiver perto de expirar.

Figura 17 – Formulário de cadastro de livro.

A imagem é uma captura de tela de um formulário de cadastro de livro. No topo, há uma barra azul com o título "Formulário de Livros" centralizado. À esquerda da barra, há um ícone de seta apontando para trás, e à direita, um ícone de smartphone com uma notificação. O formulário contém quatro campos de texto, cada um com um ícone correspondente: "Nome" (livro), "Autor", "Detalhes" e "Data de Devolução". Cada campo tem uma placeholder e uma barra horizontal para digitar o valor.

**Fonte:** Autor.

Em face do exposto, a implementação de todas as funcionalidades está em conformidade com os requisitos funcionais elicitados com os *stakeholders* através de pesquisa realizada de acordo com a Seção 3.3.

## 4.2.2 Fragmentos de Código do Aplicativo Proposto

### 4.2.2.1 *Login*

A seguir é apresentado código responsável por conter a lógica de autenticação no aplicativo, a validação é realizada através da verificação de e-mail e senha cadastrados no Firebase Authentication.

---

```

1   Future<void> _authenticate(
2     String email, String password, String urlFragment) async {
3   final url =
4     'https://identitytoolkit.googleapis.com/v1/accounts:$urlFragment?key=[API_KEY]';
5
6   final response = await http.post(
7     Uri.parse(url),
8     body: jsonEncode({
9       'email': email,
10      'password': password,
11      'returnSecureToken': true,
12    }),
13  );
14
15   final body = jsonDecode(response.body);
16
17   if (body['error'] != null) {
18     throw AuthException(body['error']['message']);
19   } else {
20     _token = body['idToken'];
21     _email = body['email'];
22     _userId = body['localId'];
23
24     _expiryDate = DateTime.now().add(
25       Duration(
26         seconds: int.parse(
27           body['expiresIn'],
28         ),
29       ),
30     );
31
32     Store.saveMap('userData', {
33       'token': _token,
34       'email': _email,
35       'userId': _userId,
36       'expiryDate': _expiryDate!.toIso8601String(),
37     });
38
39     _autoLogout();
40     notifyListeners();
41   }
42 }
```

---

Código-fonte 2 – Código responsável por contar a lógica de *login* do aplicativo.

A principal dificuldade para construção desta funcionalidade foi de encontrar uma forma de manter o usuário logado no sistema por um tempo, mesmo que o aplicativo fosse fechado. Logo, a solução

para esse problema foi encontrada na documentação oficial do Google relativa á autenticação, onde é verificado se o usuário possui um *token* válido e se o mesmo ainda está no prazo de validade, ou seja, o *token* não está expirado, está verificação é feita na linha 24 até a linha 29 do código apresentado.

#### 4.2.2.2 Salvamento de dados

A seguir temos o trecho do código responsável por salvar os dados fornecidos pelos usuários no banco de dados, neste código é feito o armazenamento dos dados de um professor. Nesta função é realizado uma lógica onde verifica se os dados fornecidos já contém um ID, caso haja, seja feita apenas á atualização dos dados, caso contrário as informações serão inseridos na base de dados.

---

```

1 Future<void> saveProfessor(Map<dynamic, Object> data) {
2     bool hasId = data['id'] != null;
3
4     final professor = Professor(
5         id: hasId ? data['id'] as String : Random().nextDouble().toString(),
6         nome: data['nome'] as String,
7         email: data['email'] as String,
8         telefone: data['telefone'] as String,
9     );
10
11    if (hasId) {
12        return updateProfessor(professor);
13    } else {
14        return addProfessor(professor);
15    }
16 }
```

---

Código-fonte 3 – Código responsável por salvar dados no banco de dados.

Quando o método *addProfessor* é invocado, os dados são inseridos na base de dados. Esta função faz um parse no *endpoint* do banco de dados na coleção de professor, e através do verbo POST os dados inseridos no banco, esses dados são identificados unicamente pelo seu ID e pelo *token* do usuário que armazenou os dados.

Quando o ID do professor não estiver nulo, é invocado o método *updateProfessor* que através do verbo PATCH será realizado á atualização dos dados do professor selecionado. Tal operação só será possível se a requisição estiver sendo feita pelo *token* que inseriu os dados.

#### 4.2.2.3 Exclusão dos dados

Neste fragmento de código é apresentado o método assíncrono responsável por excluir os dados no banco de dados.

---

```

1 Future<void> removeProfessor(Professor professor) async {
2     int index = _professores.indexWhere((p) => p.id == professor.id);
3
4     if (index >= 0) {
5         final professor = _professores[index];
6         _professores.remove(professor);
7         notifyListeners();
8
9         final response = await http.delete(
10             Uri.parse(
11                 '${Constants.PROFESSOR_BASE_URL}/$_userId/${professor.id}'.json?auth=$_token'),
12             );
13
14         if (response.statusCode >= 400) {
15             _professores.insert(index, professor);
16             notifyListeners();
17             throw HttpException(
18                 message: 'Não foi possível remover o professor.',
19                 statusCode: response.statusCode,
20             );
21         }
22     }
23 }
```

---

Código-fonte 4 – Código responsável por excluir dados no banco de dados.

Através de uma estrutura condicional é realizado uma verificação se o ID do dado armazenado corresponde ao dado que só requisitado, caso os ID sejam correspondentes, a informação será excluída do banco de dados através do verbo DELETE. Esta operação só será possível se a requisição estiver sendo feita pelo *token* que inseriu os dados.

#### 4.2.2.4 Leitura dos dados do banco

Por fim, um dos métodos muito utilizado nesta aplicação é o de leitura dos dados do banco. Como os bancos NoSQL trabalham com o formato JSON. Há a necessidade de decodificar esse formato para o formato dinâmico utilizado na aplicação, como *String* e *Datas*.

---

```
1 Future<void> loadAtividades() async {
2     _atividades.clear();
3
4     final response = await http.get(
5         Uri.parse('${Constants.ATIVIDADE_BASE_URL}/$_userId.json?auth=$_token'),
6     );
7     if (response.body == 'null') return;
8
9     Map<String, dynamic> data = jsonDecode(response.body);
10    data.forEach((atividadeId, atividadeData) {
11        _atividades.add(Atividade(
12            id: atividadeId,
13            titulo: atividadeData['titulo'],
14            disciplina: atividadeData['disciplina'],
15            dataEntrega: atividadeData['dataEntrega'] != null
16                ? DateTime.parse(atividadeData['dataEntrega'])
17                : null,
18            prioridade: atividadeData['prioridade'],
19            status: atividadeData['status'],
20            descricao: atividadeData['descricao'],
21        ));
22    });
23    notifyListeners();
24 }
```

---

Código-fonte 5 – Código responsável por fazer a leitura das informações no banco de dados.

A decodificação e leitura dos dados são feitos através de verbo GET, sendo possível apenas através do ID do usuário e o seu *token* de autenticação. Neste exemplo é feito a leitura das informações referentes as atividades que o acadêmico cadastrou no aplicativo.

## 4.3 Teste de Desempenho do Aplicativo com o Dart DevTools

### 4.3.1 CPU profiler

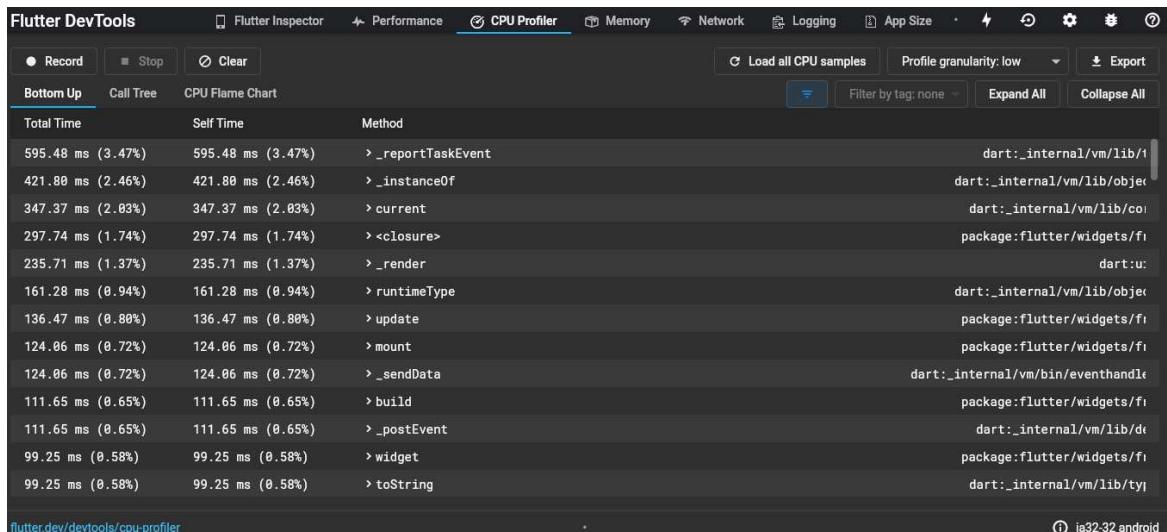
Conforme o Flutter (2021), o CPU Profiler serve para a visualização do perfilador da CPU, permitindo que você registre e crie o perfil de uma sessão de seu aplicativo Dart ou Flutter.

Comece gravando um perfil de CPU clicando em Gravar. Quando terminar de gravar, clique em Parar. Nesse ponto, os dados de criação de perfil da CPU são extraídos da VM e exibidos nas visualizações do criador de perfil (Call Tree, Bottom Up e Flame Chart).

#### 4.3.1.1 Bottom up

A visualização *bottom up* mostra o rastreamento do método para o perfil da CPU, mas, como o nome sugere, é uma representação de baixo para cima do perfil. Isso significa que cada método de nível superior na tabela é, na verdade, o último método na pilha de chamadas para uma determinada amostra de CPU.

Figura 18 – Modelo de chamadas bottom up



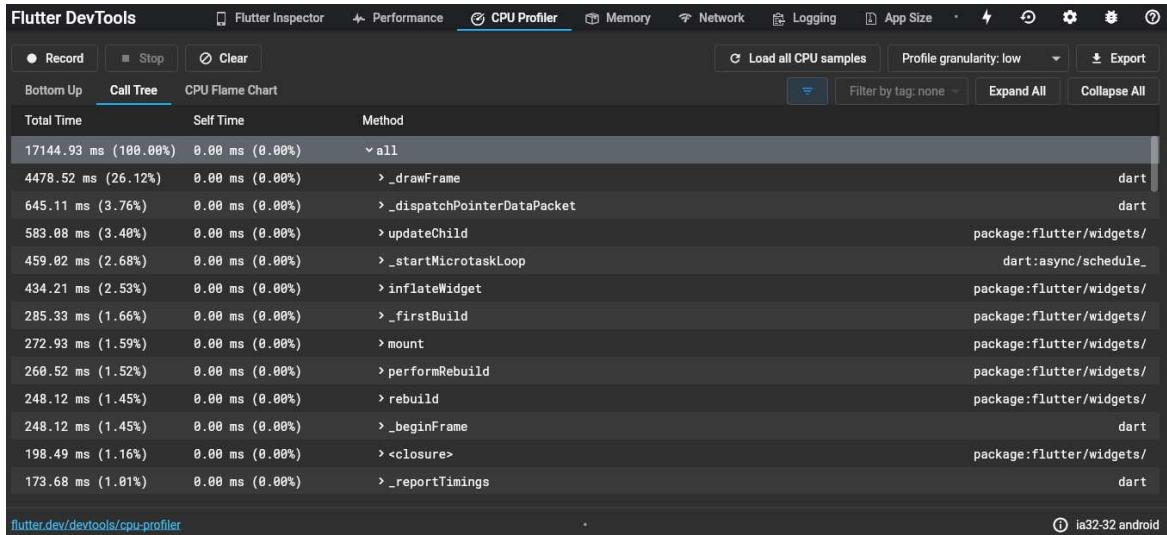
Fonte: Autor.

Neste modelo, um método pode ser expandido para mostrar seus invocadores. O **Tempo total**, é demonstrado o período que o método gasta executando seu próprio código, bem como o código de seu receptor. Já o **Tempo próprio**, mostra o tempo que os métodos de nível mais elevado da árvore ascendente, esse é o tempo que o método gasta executando apenas seu próprio código. Para subnós (os invocadores do perfil da CPU), este é o tempo próprio do receptor ao ser chamado pelo invocador. **OMétodo**, é o nome do método chamado. E por último, a **Fonte**, é o caminho do arquivo onde contém o método que foi chamado.

#### 4.3.1.2 Call tree

Neste modelo tem-se a exibição da árvore de chamadas coletada do rastreamento do método para o perfil da CPU. Esta tabela é uma representação *top-down* do perfil, o que significa que um método pode ser expandido para mostrar suas chamadas. As colunas seguem o mesmo modelo do Bottom Up 4.3.1.1.

Figura 19 – Modelo de chamadas bottom up



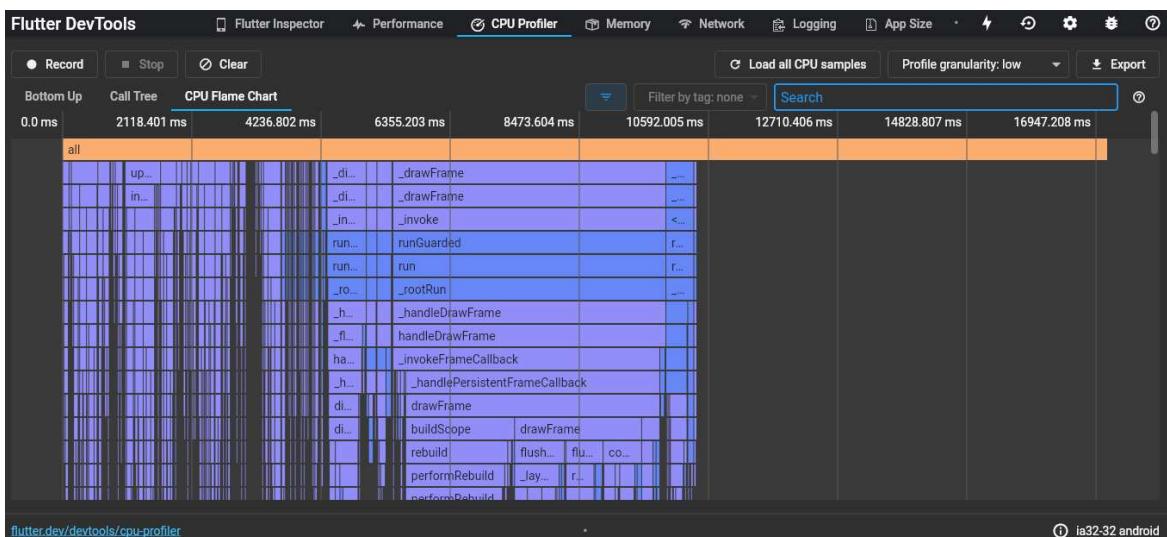
Fonte: Autor.

Este teste serve para verificar quanto tempo cada processo levou desde sua chamada. O mesmo serve para direcionar o desenvolvedor a identificar se está havendo sobrecarga nas chamadas.

#### 4.3.1.3 Flame chart

Este gráfico mostra as amostras da CPU coletadas durante a gravação. Com ele é possível ser visto como um rastreamento da pilha de execução no formato *top-down*, onde o quadro de pilha mais acima chama o que está abaixo dele.

Figura 20 – Gráfico Flame



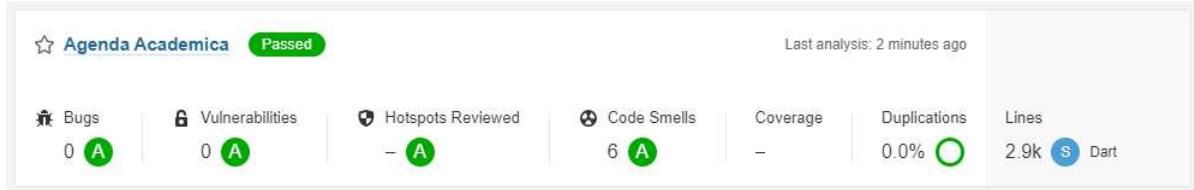
Fonte: Autor.

A largura de cada *frame* da pilha representa a quantidade de tempo que foi consumido da CPU. Os *frames* da pilha que consomem muito tempo da CPU, demonstram possíveis pontos de melhorias de desempenho.

## 4.4 Teste de Qualidade de Código com SonarQube

Conforme Ferenc (2014), o SonarQube<sup>1</sup> é uma plataforma de código aberto desenvolvida pela SonarSource, para inspeção contínua da qualidade do código. O Sonar faz análise estática do código, que fornece um relatório detalhado de *bugs*, vulnerabilidades, duplicações de código e code smell<sup>2</sup>. Mais detalhes sobre o resultado da análise pode ser observada a partir da Figura 21.

Figura 21 – Relatório de análise do SonarQube



**Fonte:** Autor.

Conforme apresentado no painel de controle de qualidade da Figura 21, os resultados obtidos pela plataforma através da análise do código do aplicativo, demonstram que foram considerados as seguintes análises:

- *Bugs*: durante a análise do código não foi encontrado nenhum bug. Logo, esta verificação recebeu o conceito A. Onde o conceito A indica que nenhum bug foi encontrado, B indica que existe pelo menos um bug mínimo. C existe pelo menos um bug principal, D que existe um bug crítico e E que pelo menos um bug está bloqueando o fluxo de funcionamento da aplicação.
- *Vulnerabilities* (Vulnerabilidades): não foi encontrada nenhuma vulnerabilidade na aplicação. Isto indica que não foi encontrado nenhum ponto de vulnerabilidade.
- *Hotspots Reviewed* (Pontos de Acesso revisados): nesse parâmetro não há nenhuma ponto de acesso para ser validado, por isso foi atribuído conceito A.
- *Code Smells*: esta verificação é uma análise profunda no código buscando má práticas durante o desenvolvimento. Mesmo sendo encontrado 6 *code smells*, foi atribuído o conceito A devido a complexidade da mudança necessária para não haver nenhum *code smell* não ser grande.
- *Coverage* (Cobertura): este parâmetro indica se a quantidade de testes na aplicação está adequada. Como não foi implementado testes, esse quesito não foi avaliado.
- *Duplications* (Duplicidade): por fim, foi verificado se havia duplicidades no código. Como não foi encontrada nenhuma linha de código duplicada, a taxa para esse item ficou em 0.0%.

<sup>1</sup> Disponível em: <<https://www.sonarqube.org/>>

<sup>2</sup> Code smell é descrito como uma indicação superficial que geralmente corresponde a um problema mais profundo no código da aplicação. O termo foi cunhado Kent Beck quando ajudava Martin Fowler a escrever seu livro Refatoração.

# 5 Conclusão

O presente trabalho teve como objetivo desenvolver uma solução para dispositivos móveis visando auxiliar discentes perante a dificuldade enfrentada em realizar o gerenciamento de disciplinas e atividades solicitadas durante a vida acadêmica. Através de uma pesquisa bibliográfica e aplicada, foi possível desenvolver um aplicativo que surgiu a partir de uma demanda real observada através de estudos realizados conforme referenciado na seção 2.1, que teve como resultado, um entendimento mais profundo sobre evasão acadêmica, entendida como a interrupção do ciclo de estudo por parte do acadêmico, sendo um fenômeno complexo e recorrente em instituições do Ensino Superior.

Para ser possível atingir o objetivo de construir uma solução multiplataforma para dispositivos móveis, e proporcionar uma ferramenta que pudesse auxiliar acadêmicos no gerenciamento de disciplinas e atividades acadêmicas, definiu-se três objetivos específicos. O primeiro consistiu em realizar uma pesquisa bibliográfica a respeito de dispositivos móveis, aplicativo nativo vs multiplataforma, das tecnologias Flutter e Dart e sobre a evasão acadêmica de modo a obter um maior entendimento sobre o assunto e que possibilitasse analisar a viabilidade da construção de um aplicativo. Para tal estudo, durante a disciplina de estágio supervisionado, foi realizado um estudo de viabilidade e elicitação de requisitos para aplicativo com 63 (sessenta e três) acadêmicos de diversos cursos e instituições de ensino. Com base no resultado deste estudo, foi possível coletar, analisar e estabelecer as funcionalidades que o aplicativo viria conter.

Em seguida, o segundo objetivo constitui-se na implementação de um aplicativo multiplataforma utilizando o *framework* Flutter, por se tratar de uma ferramenta que possibilita a partir de um único código-fonte, desenvolver soluções tanto para Android quanto para iOS. O aplicativo foi codificado em conformidade com os diagramas modelados, fundamentados nos dados elicitados com os *stakeholders*.

Por último, o terceiro objetivo constituiu-se na realização de uma série de testes tanto de desempenho quanto de qualidade de código, cujos resultados apontam para a viabilidade da implantação da solução em ambiente de produção, em outras palavras, nas lojas de aplicativos para dispositivos móveis.

Dessa maneira, o projeto aqui desenvolvido resultou na construção de um aplicativo multiplataforma com foco em dispositivos móveis, que poderá contribuir significativamente para os acadêmicos, proporcionando uma ferramenta capaz de auxiliá-los no monitoramento de prazos, gerenciamento de suas disciplinas e das suas atividades acadêmicas.

## 5.1 Trabalhos Futuros

O próximo passo com relação ao projeto:

- Estabelecer um período de testes com acadêmicos para verificar e validar as funcionalidades do aplicativo; e
- Implantar o aplicativo nas lojas de aplicativos para dispositivos móveis.

# Referências

- ABLESON, F. et al. *Android em ação*. [S.l.]: Elsevier Brasil, 2012.
- ALESSANDRIA, S. *Flutter projects : a practical, project-based guide to building real-world cross-platform mobile applications and games*. [S.l.]: Birmingham, UK : Packt Publishing, (2020).
- ANDROID.COM. *Android / A plataforma que redefine o impossível*. 2021. Disponível em: <[https://www.android.com/intl/pt-BR\\_br/](https://www.android.com/intl/pt-BR_br/)>. Acesso em: 11 mar. 2021.
- APPLE.COM. *iOS 14 - Apple (BR)*. 2021. Disponível em: <<https://www.apple.com/br/ios/ios-14/>>. Acesso em: 11 mar. 2021.
- BAGGI, C. A. D. S.; LOPES, D. A. Evasão e avaliação institucional no ensino superior: uma discussão bibliográfica. *Avaliação: Revista da Avaliação da Educação Superior (Campinas)*, SciELO Brasil, v. 16, n. 2, p. 355–374, 2011.
- CAVALCANTE, C. H. L.; JUNIOR, P. A. dos S. Fatores que influenciam o desempenho escolar: a percepção dos estudantes do curso técnico em contabilidade do ifrs–instituto federal de educação, ciência e tecnologia do rio grande do sul, campus porto alegre. *Revista Liberato*, v. 14, n. 21, p. 29–50, 2013.
- CORAZZA, P. V. Um aplicativo multiplataforma desenvolvido com flutter e nosql para o cálculo da probabilidade de apendicite. (2018).
- DAVID, L. M. L.; CHAYM, C. D. Evasão universitária: Um modelo para diagnóstico e gerenciamento de instituições de ensino superior. *RAIMED: Revista de Administração IMED*, (2019).
- DIAS, P. O impacto do telemóvel na sociedade contemporânea: panorama de investigação em ciências sociais. *Comunicação & Cultura*, Quimera, v. 3, p. 77–96, 2007.
- EL-KASSAS, W. S. et al. Taxonomy of cross-platform mobile applications development approaches. *Ain Shams Engineering Journal*, v. 8, n. 2, p. 163–190, 2017. ISSN 2090-4479. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2090447915001276>>.
- FERENC, R. et al. Source meter sonar qube plug-in. In: IEEE. *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*. [S.l.], (2014). p. 77–82.
- FLUTTER. Devtools. <https://flutter.dev/docs/development/tools/devtools/overview>. Acessado em 28 de setembro de 2021, (2021).
- Flutter. *Flutter*. (2021). Disponível em: <<https://flutter.dev/showcase>>. Acesso em: 20 de abril de 2021.
- Flutter. *Flutter on Desktop*. (2021). Disponível em: <<https://flutter.dev/multi-platform/desktop>>. Acesso em: 23 de novembro de 2021.
- FLUTTER. Using the cpu profiler view. <https://flutter.dev/docs/development/tools/devtools/cpu-profiler>. Acessado em 28 de setembro de 2021, (2021).
- Gartner. *Gartner Says Worldwide Sales of Smartphones Returned to Growth in First Quarter of 2018*. 2018. Disponível em: <<https://www.gartner.com/en/newsroom/press-releases/2018-05-29-gartner-says-worldwide-sales-of-smartphones-returned-to-growth-in-first-quarter-of-2018>>. Acesso em: 23 de setembro de 2021.
- GIL, A. C. *Didática do ensino superior*. [S.l.]: Atlas, 2015.
- Google. *Firebase Authentication*. (2021). Disponível em: <<https://firebase.google.com/docs/auth>>. Acesso em: 27 de novembro de 2021.
- Google. *Firebase Realtime Database - Armazene e sincronize dados em tempo real*. (2021). Disponível em: <<https://firebase.google.com/products realtime-database?hl=pt-br>>. Acesso em: 27 de novembro de 2021.

GS.STATCOUNTER.COM. *Mobile Operating System Market Share Worldwide*. 2021. Disponível em: <<https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-202103-bar>>. Acesso em: 11 mar. 2021.

HIPÓLITO, O. O gargalo do ensino superior brasileiro: Depoimento. 2011.

HOED, R. M. Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de computação. (2016).

KRAEMER, M. E. P. Reflexões sobre o ensino da contabilidade. *Revista Brasileira de Contabilidade*, n. 153, p. 64–79, (2011).

LIMA, J. B. Estudo de viabilidade para construção de software de agenda acadêmica. 2020.

MADEIRA, B. B. A. et al. Desenvolvimento de aplicativos: frameworks nativos versus multiplataforma (um estudo de caso da inovadigital). *Anais do Salão Internacional de Ensino, Pesquisa e Extensão*, v. 12, n. 2, (2020).

MARINHO, L. H. *Iniciando com Flutter Framework: Desenvolva aplicações móveis no Dart Side!* [S.l.]: Casa do Código, 2020.

MARINHO, L. H. *Iniciando com Flutter Framework: Desenvolva aplicações móveis no Dart Side!* [S.l.]: Casa do Código, (2020).

MATOS, B. R. D.; BRITTO, J. G. de. *Estudo comparativo entre o desenvolvimento de aplicativos móveis utilizando plataformas nativas e multiplataformas*. (2017).

NAPOLI, M. L. *Beginning Flutter: A Hands On Guide To App Development*. [S.l.]: John Wiley & Sons, (2019).

NAPOLI, M. L. *Beginning Flutter: A Hands On Guide To App Development*. [S.l.]: John Wiley & Sons, 2019.

PAREDES, A. S. *A evasão do terceiro grau em Curitiba*. [S.l.]: NUPES, 1994.

PRESSMAN, R.; MAXIM, B. *Engenharia De Software: UMA ABORDAGEM PROFISSIONAL*. MCGRAW HILL - ARTMED, 2016. ISBN 9788580555332. Disponível em: <<https://books.google.com.br/books?id=smNFvgAACAAJ>>.

PRESSMAN, R.; MAXIM, B. *Engenharia De Software: UMA ABORDAGEM PROFISSIONAL*. MCGRAW HILL - ARTMED, (2016). ISBN 9788580555332. Disponível em: <<https://books.google.com.br/books?id=smNFvgAACAAJ>>.

PRODANOV, C. C.; FREITAS, E. C. D. *Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição*. [S.l.]: Editora Feevale, (2013).

Prodest. *O uso de aplicativos na sociedade*. (2021). Disponível em: <<https://prodest.es.gov.br/o-uso-de-aplicativos-na-sociedade>>. Acesso em: 16 de agosto de 2021.

SANTOS, L. P. dos. Desempenho de aplicações móveis utilizando implementação nativa ou frameworks multiplataformas. (2018).

Semesp. *Índice de evasão de alunos é maior na área de tecnologia da informação*. (2012). Disponível em: <<http://g1.globo.com/sp/sao-carlos-regiao/noticia/2012/09/indice-de-evasao-de-alunos-e-maior-na-area-de-tecnologia-da-informacao.html>>. Acesso em: 04 abril 2021.

SILVA, M. M. da; SANTOS, M. T. P. Os paradigmas de desenvolvimento de aplicativos para aparelhos celulares. *Revista TIS*, v. 3, n. 2, 2014.

SOMMERVILLE, I. *Engenharia de software*. Pearson Prentice Hall, (2011). ISBN 9788579361081. Disponível em: <<https://books.google.com.br/books?id=H4u5ygAACAAJ>>.

SOMMERVILLE, I. et al. *Engenharia de software*. ADDISON WESLEY BRA, 2008. ISBN 9788588639287. Disponível em: <<https://books.google.com.br/books?id=ifYOgAACAAJ>>.

Statista. *Number of apps available in leading app stores as of 1st quarter 2021*. 2021. Disponível em: <<https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>>. Acesso em: 16 de outubro de 2021.

Statista. *Number of available applications in the Google Play Store from December 2009 to July 2021*. 2021. Disponível em: <<https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/>>. Acesso em: 15 de maio de 2021.

STEIL, R. *iOS. Programe Para iPhone e iPad*. [S.l.]: Casa do Código, (2012). v. 1.

TANENBAUM, A. S. *Redes de Computadores*. trad. 4 ed. Rio de Janeiro: Elsevier, (2003).

VARGAS, T. C. d. S. A história de uml e seus diagramas. [https://projetos.inf.ufsc.br/arquivos\\_projetos/projeto\\_721/artigo\\_tcc.pdf](https://projetos.inf.ufsc.br/arquivos_projetos/projeto_721/artigo_tcc.pdf). Acessado em, v. 29, n. 11, p. 2015, 2007.

ZAMMETTI, F. *Practical Flutter*. [S.l.]: Springer, (2019).

ZANATO, K. Y. da S. et al. Análise da evasão de alunos da área de tecnologia da informação por meio de um banco de dados orientado a grafos. *Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação*, v. 1, n. 8, 2018.

## Anexos

# ANEXO A – Questionário a respeito do levantamento dos requisitos do aplicativo

## A.1 Esclarecimento a respeito do questionário

Figura 22 – Informação sobre o questionário.

Levantamento de informações para o desenvolvimento de um aplicativo para gerenciamento de agenda acadêmica.

Acadêmico Jhemys Barros Lima, discente do curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS. Esse formulário tem por finalidade acadêmica, o intuito de verificar a viabilidade da implementação de uma agenda acadêmica que permitirá o usuário ter na palma de suas mãos, um aplicativo que fornecerá recursos que o auxiliarão no seu dia-a-dia acadêmico. Como por exemplo: Cadastrar disciplinas as quais o mesmo está matriculado no período letivo, sendo possível anexar informações úteis sobre tais disciplinas, como professor da disciplina, horário de aula ou mesmo atividades a serem desenvolvidas.

\*Obrigatório

Fonte: Autor

## A.2 Dados dos entrevistados

Figura 23 – Dados sobre os entrevistados.

**Dados sobre o entrevistado**

Nome \*

Sua resposta

Instituição de Ensino \*

Nome \*

Sua resposta

Instituição de Ensino \*

Sua resposta

Curso \*

Sua resposta

Período \*

Escolher ▾

Utiliza qual sistema operacional móvel \*

Android

iOS

Ambos

Gostaria de fazer parte do desenvolvimento desse projeto, dando sua opinião sobre os protótipos e versões de teste do aplicativo? Se sim, informe seu e-mail.

Sua resposta

**Fonte:** Autor

### A.3 Questionário sobre as funcionalidades do aplicativo

Figura 24 – Pergunta 01.

Levantamento de informações para o desenvolvimento de um aplicativo para gerenciamento de agenda acadêmica.  
\*Obrigatório

**Sobre o aplicativo**

As perguntas abaixo estão diretamente ligadas às funcionalidades que o aplicativo irá conter.

Quando um aplicativo lhe oferece a possibilidade fazer 'login' utilizando algumas plataformas, como (E-mail) ou redes sociais (Facebook, Twitter), qual delas você costuma utilizar? Você pode escolher mais de uma opção. \*

Gmail  
 Facebook  
 Twitter  
 Criar uma conta com seu e-mail  
 Outro: \_\_\_\_\_

**Fonte:** Autor

Figura 25 – Pergunta 02.

Dentre suas funcionalidades, o aplicativo contará com recurso de cadastrar as disciplinas que estão sendo cursadas, onde o acadêmico irá fornecer: nome da disciplina, nome do professor e sala de aula. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 26 – Pergunta 03.

Será possível também, cadastrar dados de professores, onde o acadêmico fornecerá: nome, telefone e e-mail. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 27 – Pergunta 04.

Para não ficar sem saber qual aula você terá no dia, será possível criar Horário, onde o acadêmico irá fornecer: nome da disciplina, dia da semana, hora de início e fim da aula. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 28 – Pergunta 05.

Para não perder prazos, o aplicativo terá o recurso de cadastrar atividades, onde o acadêmico fornecerá: nome da disciplina, prazo da atividade, prioridade da atividade (alta, média ou baixa) e uma campo para descrever a atividade a ser desenvolvida. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 29 – Pergunta 06.

Também terá o recurso de cadastrar avaliações, onde o acadêmico irá fornecer: nome da disciplina, data da avaliação, prioridade (alta, média ou baixa) e campo para o conteúdo a ser estudado. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 30 – Pergunta 07.

Um dos recursos fornecidos pelo aplicativo, será opção de cadastrar as notas obtidas nas disciplinas, onde o acadêmico fornecerá as notas e o sistema irá calcular a média final. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 31 – Pergunta 08.

Não perca aquele evento importante, pois, um dos recursos do aplicativo, será o controle de eventos (Seminário, Semana acadêmica, simpósio ou Palestras), onde o acadêmico fornecerá: nome do evento, dia do evento, horário de início e fim. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 32 – Pergunta 09.

Um importante recurso, será o de controle de devolução de livro emprestado da biblioteca, onde o acadêmico fornecerá: nome do livro, autor, detalhes e data para devolver o livro. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 33 – Pergunta 10.

Funcionalidade de controle de frequência, onde o acadêmico fornecerá: disciplina, quantidade máxima que pode faltar. Após isso, o acadêmico poderá ir incrementando as faltas, até o aplicativo alertar que já está perto do limite de faltas. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 34 – Pergunta 11.

O acadêmico poderá exportar (fazer backup) de todas as configurações e dados contidos no aplicativo. O quanto isso será útil para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 35 – Pergunta 12.

Após ser exposto essa série de recursos, quão útil esse aplicativo de agenda acadêmica seria para você? \*

Muito útil  
 Pouco útil  
 Nada útil

**Fonte:** Autor

Figura 36 – Sugestão de Funcionalidades.

Que recursos não mencionados nas perguntas acima, você acredita ser extremamente útil e que o aplicativo deveria ter? Descreva-o.

Sua resposta \_\_\_\_\_

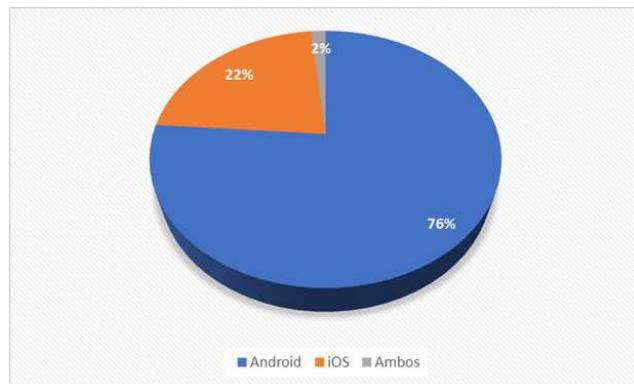
**Fonte:** Autor

# ANEXO B – Resultados coletados do Questionário

## B.1 Resultado a respeito do paradigma de desenvolvimento do aplicativo

Na primeira parte da pesquisa realizada, uma das perguntas do questionário estava diretamente ligada a qual sistema operacional os entrevistados utilizam em seu dispositivo móvel.

Figura 37 – SO utilizado pelos entrevistados.



**Fonte:** Autor

Conforme pode ser observado no gráfico da Figura 37, houve um percentual de 76% do número de entrevistados que utiliza do Android, o que corresponde a 48 entrevistados. Seguido de 22% tem o iOS, percentual que corresponde a 14 entrevistados que utilizam esse sistema operacional. Por fim, 2% dos entrevistam responderam que utilizam ambos os sistemas, o que corresponde a 1 entrevistado.

Sendo assim, a melhor metodologia para o desenvolvimento do aplicativo para abranger 100% dos entrevistados e tornar o aplicativo disponível para o maior número de usuários possível, é a híbrida. Pois, o mesmo permitirá que a partir de um único código-fonte seja criando um aplicativo tanto para o Android quanto para o iOS.

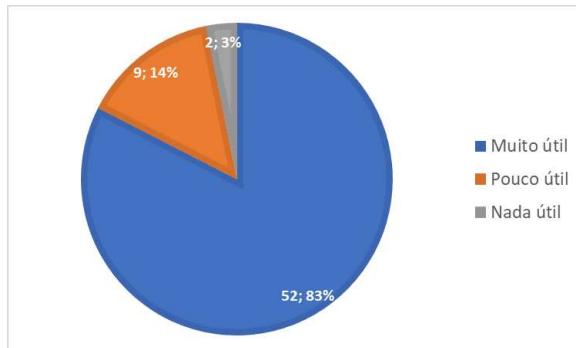
## B.2 Resultado das perguntas a respeito das funcionalidades do aplicativo

Na segunda parte do questionário houve uma série de perguntas que tiveram como objetivo validar as funcionalidades do aplicativo, onde cada pergunta poderia ser respondida através de três opções, sendo elas: “Muito útil”, “Pouco útil” e “Nada útil”. Através das respostas dessas alternativas, foi possível determinar, validar e verificar o quanto útil aquela determinada funcionalidade era para os acadêmicos entrevistados. A seguir, será apresentado através de gráficos, as respostas obtidas, bem como uma sucinta descrição a respeito da funcionalidade.

### B.2.1 Cadastro de disciplinas

O objetivo dessa funcionalidade é possibilitar que os usuários possam realizar outras atividades, tais como: vincular notas obtidas nessas disciplinas, vincular atividade/seminários, criar horário entre outros. Logo, para cadastrar disciplinas, o usuário deve fornecer os seguintes dados: nome da disciplina, nome do professor e a sala de aula.

Figura 38 – Cadastrar disciplinas cursadas no período letivo.



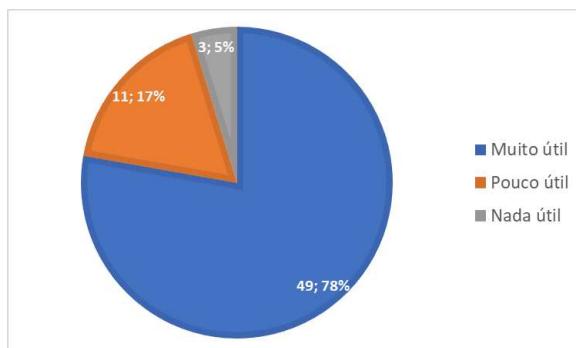
**Fonte:** Autor

Conforme pode ser observado no gráfico da Figura 38, a funcionalidade de cadastro de disciplinas é útil para 83% dos entrevistados, logo, é de vital importância que essa funcionalidade esteja presente no aplicativo.

### B.2.2 Cadastrar dados dos professores

O Objetivo dessa funcionalidade é possibilitar que o usuário possa guardar no aplicativo os dados referentes a seus professores, tal como telefone e *e-mail*. Portanto, para essa funcionalidade, o usuário deve fornecer os seguintes dados: nome do professor, telefone e o endereço de *e-mail*.

Figura 39 – Cadastrar dados dos professores.



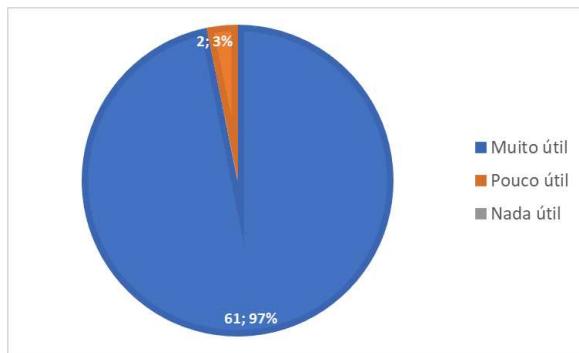
**Fonte:** Autor

Conforme pode ser observado no gráfico da Figura 39, a funcionalidade de cadastro de dados dos professores é útil para 78% dos entrevistados, logo, é de vital importância que essa funcionalidade esteja presente no corpo de funcionalidade do aplicativo.

### B.2.3 Cadastros de avaliações

O propósito dessa funcionalidade é para que usuários possam estar cadastrando suas avaliações/-testes/provas, e receber notificações quanto aos prazos que foram estipulados durante o cadastro dessa atividade. Para isso ser possível, o usuário terá que fornecer os seguintes dados: nome da disciplina, data da realização da avaliação, prioridade para essa avaliação (alta, média ou baixa) e ainda poderá inserir no campo de detalhes, o conteúdo a ser estudado.

Figura 40 – Cadastrar avaliações.



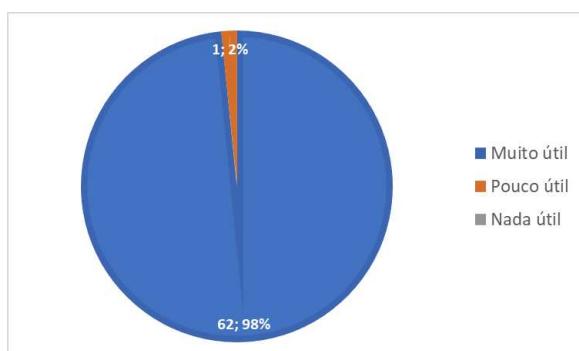
**Fonte:** Autor

De acordo com o gráfico da Figura 40, a funcionalidade de cadastro de avaliações é útil para 97% dos entrevistados, logo, é de extrema importância para os usuários, que essa funcionalidade seja implementada no aplicativo.

### B.2.4 Cadastrar atividades

A finalidade dessa funcionalidade é proporcionar ao usuário a possibilidade de cadastrar suas atividades/seminários/trabalhos ou tarefas, que também contará com o recurso de receber notificações quanto aos prazos que foram estipulados durante o cadastro dessa atividade. Portanto, para cadastrar uma atividade, o usuário deverá fornecer: nome da disciplina, prazo da atividade, prioridade da atividade (alta, média ou baixa) e também poderá inserir no campo de detalhes, o conteúdo a ser desempenhado nessa atividade.

Figura 41 – Cadastrar atividades e seminários.



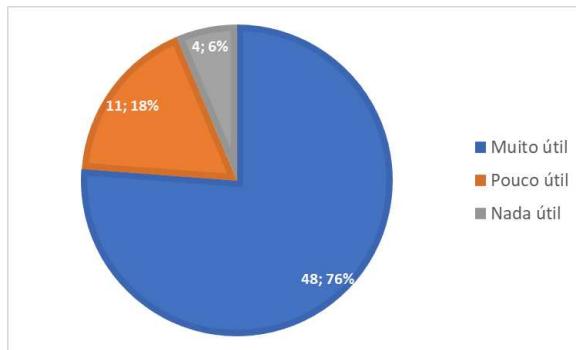
**Fonte:** Autor

Segundo o gráfico da Figura 41, a funcionalidade de cadastro de atividades/seminários é útil para 98% dos entrevistados, portanto, é de suma importância que essa funcionalidade esteja presente no corpo de funcionalidade do aplicativo.

### B.2.5 Cadastrar as notas

O objetivo de cadastrar notas, é possibilitar que os usuários possam vincular as notas obtidas durante as avaliações, e anexar nas disciplinas. Logo, o usuário terá que fornecer as notas obtidas, que o sistema irá calcular a média final.

Figura 42 – Cadastrar as notas obtidas nas disciplinas.



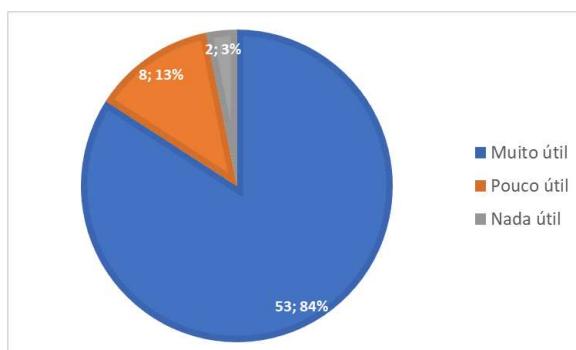
**Fonte:** Autor

Como pode ser observado no gráfico da Figura 42, a funcionalidade de cadastro de notas é útil para 76% dos entrevistados, portanto, é de suma importância que essa funcionalidade esteja presente no corpo de funcionalidade do aplicativo.

### B.2.6 Criar horário

Essa funcionalidade é o recurso que irá permitir que o usuário possa criar o horário de aula das disciplinas que ele terá durante a semana. Para isso, o usuário irá fornecer: nome da disciplina, dia da semana que terá aula da disciplina, hora de início e fim da aula.

Figura 43 – Criar horário de aula.



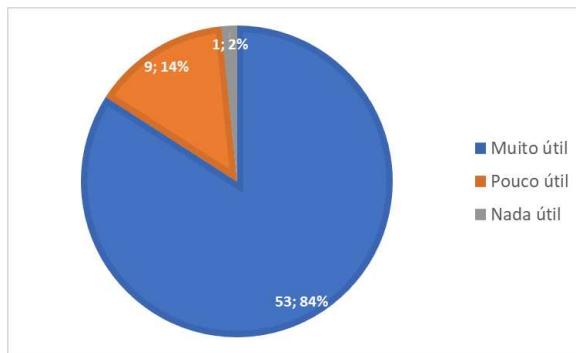
**Fonte:** Autor

Conforme pode ser observado no gráfico da Figura 43, a funcionalidade de criar horário é útil para 84% dos entrevistados, portanto, é de suma importância que essa funcionalidade esteja presente no corpo de funcionalidade do aplicativo.

### B.2.7 Controle de eventos

O propósito dessa funcionalidade, o que o usuário possa agendar eventos (Seminário, Semana acadêmica, simpósio ou Palestras) futuros e que é de seu interesse, e esse recurso ainda contará com notificações para que quando os prazos que foram estipulados durante o cadastro do evento estiverem próximos, o usuário possa ser notificado. Assim, para cadastrar um evento, o acadêmico irá preencher os campos: nome do evento, dia do evento, horário de início e fim.

Figura 44 – Controle de eventos.



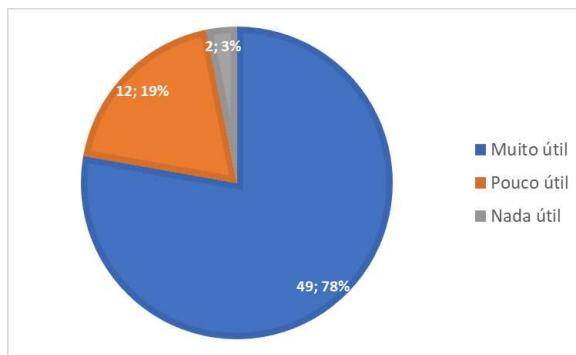
**Fonte:** Autor

De acordo com o gráfico da Figura 44, a funcionalidade de controle de eventos é útil para 84% dos entrevistados, portanto, é de suma importância que essa funcionalidade esteja presente no corpo de funcionalidade do aplicativo.

### B.2.8 Controle de frequência

O objetivo dessa funcionalidade é que os usuários possam ter controle sob suas faltas. Assim, o acadêmico irá fornecer o: nome da disciplina, quantidade máxima que se pode faltar. Após isso, o acadêmico poderá ir incrementando as faltas, até o aplicativo o alertar que já está perto do limite máximo de faltas.

Figura 45 – Controle de frequência.



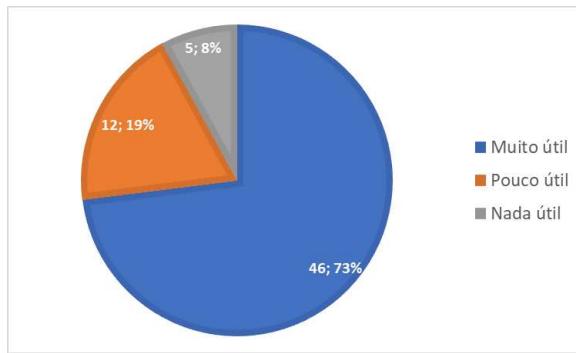
**Fonte:** Autor

Como observado no gráfico da Figura 45, a funcionalidade de controle de frequência é útil para 78% dos entrevistados, portanto, é de suma importância que essa funcionalidade esteja presente no corpo de funcionalidade do aplicativo.

### B.2.9 Controle de devolução de livro

A finalidade dessa funcionalidade é que o usuário não perca prazos quanto a devolução de livros emprestados da biblioteca. Assim, nessa funcionalidade o usuário irá fornecer: nome do livro, autor do livro, detalhes sobre o livro e a data para devolver o livro.

Figura 46 – Controle de devolução de livro.



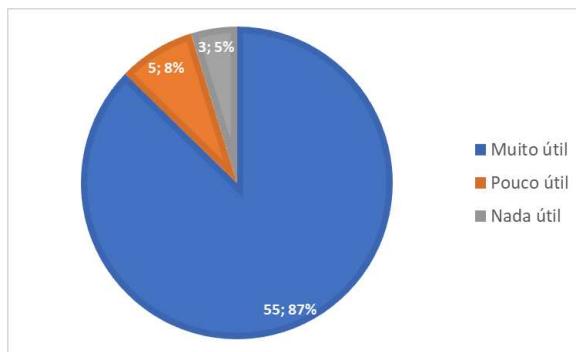
**Fonte:** Autor

Segundo o gráfico da Figura 46, a funcionalidade de controle de devolução de livro é útil para 73% dos entrevistados, portanto, é de suma importância que essa funcionalidade esteja presente no corpo de funcionalidade do aplicativo.

### B.2.10 Exportação de dados e configuração

Esse recurso irá permitir que os usuários possam possa exportar (fazer backup) de todas as configurações e dados contidos no aplicativo.

Figura 47 – Exportar (fazer backup) dados e configuração do aplicativo.



**Fonte:** Autor

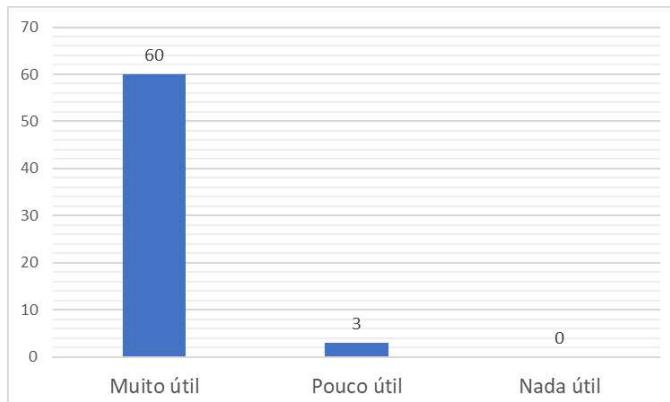
Conforme pode ser observado no gráfico da Figura 47, a funcionalidade de exportar os dados e configurações é útil para 87% dos entrevistados, portanto, é de suma importância que essa funcionalidade esteja presente no corpo de funcionalidade do aplicativo.

## B.3 Resultado da pesquisa sobre a viabilidade do desenvolvimento do aplicativo

Assim, levando em consideração todos os dados coletados durante a pesquisa realizada através do questionário no Google *Forms*, que tratou do estudo de viabilidade para a construção do um aplicativo de agenda acadêmica, bem como a elicitação e a validação dos seus requisitos. E teve a participação de 63 (sessenta e três) acadêmicos dos mais diversos cursos e instituições de ensino.

Conforme pode ser observado no gráfico da Figura 48, que corresponde a uma pergunta realizada aos acadêmicos que questionava aos mesmos, que depois de exposto uma série de recursos. O quanto útil o aplicativo de agenda acadêmica seria para auxilia-los no dia a dia, na gestão de atividades acadêmicos, controle de frequência e controle de devolução de livros.

Figura 48 – Viabilidade de implementação do aplicativo.



**Fonte:** Autor

Dessa maneira, os resultados indicam que, de um modo geral, é sim viável desenvolver o aplicativo e ele será de extrema valia e útil para que os acadêmicos possam se organizar e realizar uma melhor gestão da sua rotina acadêmica.

# Apêndices

# APÊNDICE A – Requisitos Funcionais e Requisito não funcional

## A.1 Requisitos Funcionais

### RF001 – Cadastrar usuário

**Caso de Uso:** Criar conta, Realizar login.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa criar uma conta no sistema, utilizando um e-mail e senha válidos ou possibilitar que o usuário utilize alguma conta, como gmail, facebook para realizar a autenticação.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** E-mail válido, conta do gmail ou facebook.

**Saídas e pós-condição:** Entrar no sistema

### RF002 – Redefinir senha

**Caso de Uso:** Redefinir senha.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa alterar sua senha no sistema, utilizando um e-mail para o recebimento de confirmação de alteração de senha, depois de validado no e-mail, retornar usuário para o sistema, onde ele poderá colocar a nova senha e confirmar nova senha.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** E-mail válido.

**Saídas e pós-condição:** Entrar no sistema

### RF003 – Cadastrar disciplinas

**Caso de Uso:** Gerenciar disciplina.

**Ator:** Usuário, Administrador.

**Descrição:** Para o cadastro de uma disciplina, é necessário informar o, período, nome, dia/horário, professor e sala de aula. Sendo que o campo “sala de aula” não é obrigatório.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Período/ano letivo

**Saídas e pós-condição:** Mensagem retornando a confirmação do cadastro da disciplina ou erro relacionado ao cadastro.

#### RF004 – Listar disciplinas

**Caso de Uso:** Gerenciar disciplina.

**Ator:** Usuário, Administrador.

**Descrição:** Para listagem de disciplinas é necessário autenticar o usuário e informar o ano/semestre letivo. Deve conter também, opção de buscar e editar determinada disciplina.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Período/ano letivo.

**Saídas e pós-condição:** Listagem de disciplinas com opção de editar informações.

#### RF005 – Deletar disciplinas

**Caso de Uso:** Gerenciar disciplina.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar disciplinas é necessário que o usuário esteja autenticado no sistema, depois de ter a listagem de todas as disciplinas, o usuário deverá selecionar quais disciplinas deseja excluir.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado, período/ano letivo, disciplinas que deseja excluir.

**Saídas e pós-condição:** Mensagem informando que o(s) disciplinas(es) selecionados foram excluídos.

#### RF006 – Editar disciplinas

**Caso de Uso:** Gerenciar disciplina.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar disciplinas é necessário que o usuário esteja autenticado no sistema, depois de ter a listagem de todas as disciplinas, o usuário deverá selecionar quais disciplinas deseja editar.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado, período/ano letivo, disciplinas que deseja editar.

**Saídas e pós-condição:** Mensagem informando que o(s) disciplinas(es) selecionados foram editadas.

#### RF007 – Cadastrar períodos

**Caso de Uso:** Gerenciar períodos.

**Ator:** Usuário, Administrador.

**Descrição:** Para o cadastro de período, é necessário informar dentre os períodos já cadastrados no sistema, quantos período o aluno deseja.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:**

**Saídas e pós-condição:** Mensagem retornando a confirmação do cadastro da período ou erro relacionado ao cadastro.

### RF008 – Listar períodos

**Caso de Uso:** Gerenciar períodos.

**Ator:** Usuário, Administrador.

**Descrição:** Para listar os períodos, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Estar logado no aplicativo.

**Saídas e pós-condição:** Deve conter a opção de editar determinado período.

### RF009 – Deletar períodos

**Caso de Uso:** Gerenciar períodos.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar os períodos, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Estar logado no aplicativo.

**Saídas e pós-condição:** Mensagem informando que o(s) período(es) selecionados foram excluídos.

### RF010 – Editar períodos

**Caso de Uso:** Gerenciar períodos.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar os períodos, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Estar logado no aplicativo.

**Saídas e pós-condição:** Mensagem informando que o(s) período(es) selecionados foram editados.

## RF011 – Cadastrar professor

**Caso de Uso:** Gerenciar professor.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa cadastrar um ou mais professores, informando nome, telefone e e-mail. Sendo que o campo “telefone” não é obrigatório.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Disciplina.

**Saídas e pós-condição:** Mensagem retornando a confirmação do cadastro do professor ou erro relacionado ao cadastro.

## RF012 – Listar professor

**Caso de Uso:** Gerenciar professor.

**Ator:** Usuário, Administrador.

**Descrição do caso de uso:** Para listagem de professores é necessário que o usuário esteja autenticado no sistema. Deve conter também, opção de buscar e editar determinada disciplina.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Estar logado no aplicativo.

**Saídas e pós-condição:** Deve conter a opção de editar determinado professor.

## RF013 – Deletar professor

**Caso de Uso:** Gerenciar professor.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar o professor, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Estar logado no aplicativo.

**Saídas e pós-condição:** Mensagem informando que o professor selecionado foi excluído.

## RF014 – Editar professor

**Caso de Uso:** Gerenciar professor.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar o professor, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

### RF015 – Cadastrar provas

**Entradas e pré-condições:** Estar logado no aplicativo.

**Saídas e pós-condição:** Mensagem informando que o professor selecionado foi editado.

**Caso de Uso:** Gerenciar prova.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa cadastrar um ou mais provas, informando disciplina, data da avaliação com alerta de data, prioridade da prova (Alta, média, baixa), e um campo onde o aluno poderá colocar o conteúdo a ser estudado para a avaliação.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Confirmação do cadastro da prova ou erro relacionado ao cadastro.

### RF016 – Listar provas

**Caso de Uso:** Gerenciar prova.

**Ator:** Usuário, Administrador.

**Descrição:** As avaliações cadastradas no sistema, deverá ser exibida na tela inicial do sistema após o aluno autenticar. Deve ser possível o usuário selecionar uma determinada avaliação e alterar os dados dela.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

### RF017 – Deletar prova

**Caso de Uso:** Gerenciar prova.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar a prova, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Mensagem informando que a prova selecionada foi excluída.

### RF018 – Editar provas

**Caso de Uso:** Gerenciar prova.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar a prova, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Mensagem informando que o prova selecionado foi editada.

### RF019 – Cadastrar anotações

**Caso de Uso:** Gerenciar anotação.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa cadastrar um ou mais anotações, podendo informar disciplina, e adicionar a essas anotações: textos, imagens, documentos como pdf ou doc e áudios.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Confirmação do cadastro da anotação ou erro relacionado ao cadastro.

### RF020 – Listar anotações

**Caso de Uso:** Gerenciar anotação.

**Ator:** Usuário, Administrador.

**Descrição:** As anotações cadastradas no sistema, devem ser exibidas em uma área designada para a mesmo, denominada “Minhas anotações”, para que essas anotações sejam exibidas, o usuário deverá estar autenticado no sistema. Também deve ser possível que o usuário selecionar uma determinada anotação e possa alterar os dados dela.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

### RF021 – Deletar anotações

**Caso de Uso:** Gerenciar anotação.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar uma anotação, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado, mensagem de alerta perguntando se o usuário realmente deseja excluir a anotação.

## RF022 – Editar anotações

**Saídas e pós-condição:** Mensagem informando que a(s) anotação selecionados foram excluídos.

**Caso de Uso:** Gerenciar anotação.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar a prova, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Mensagem informando que a anotação selecionada foi editada.

## RF023 – Cadastrar atividades/seminários

**Caso de Uso:** Gerenciar atividades/seminários.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa cadastrar um ou mais atividades/seminários, informando disciplina, data das atividades /seminários com alerta de data, prioridade da atividade/seminário (Alta, média, baixa), status (Pendente, Em andamento, Concluída), e um campo onde o aluno poderá colocar o conteúdo a ser estudado para a atividades/seminários.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Confirmação do cadastro da atividades/seminários ou erro relacionado ao cadastro.

## RF024 – Listar atividades/seminários

**Caso de Uso:** Gerenciar atividades/seminários.

**Ator:** Usuário, Administrador.

**Descrição:** As atividades /seminários cadastradas no sistema, deverão ser exibidas em um campo denominado “Minhas atividades /seminários”, para isso o usuário deverá estar autenticado no sistema. Deverá ser possível o usuário selecionar uma determinada atividades /seminário e alterar os dados dela.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

**Caso de Uso:** Gerenciar atividades/seminários.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar uma atividades/seminários, o usuário deve estar autenticado no aplicativo.

### RF025 – Deletar atividades/seminários

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado, mensagem de alerta perguntando se o usuário realmente deseja excluir a atividades/seminários.

**Saídas e pós-condição:** Mensagem informando que a(s) atividades/seminários selecionadas foram excluídas.

### RF026 – Editar atividades/seminários

**Caso de Uso:** Gerenciar atividades/seminários.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar as atividades/seminários, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Mensagem informando que as atividades/seminários selecionada foram editada.

### RF027 – Cadastrar notas

**Caso de Uso:** Gerenciar notas.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa cadastrar as notas das disciplinas que está cursando. Informando: disciplina, nota da A1, nota da A2 e exame final.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Confirmação do cadastro da nota ou erro relacionado ao cadastro.

### RF028 – Listar notas

**Caso de Uso:** Gerenciar notas.

**Ator:** Usuário, Administrador.

**Descrição:** As notas cadastradas no sistema, deverão ser exibidas em um campo denominado “Minhas notas”, para isso o usuário deverá estar autenticado no sistema. Deverá ser possível o usuário selecionar uma determinada nota e alterar os dados dela, o sistema deverá ser capaz de calcular a média entre as duas notas e do exame caso seja preenchido, e exibir o resultado em um campo de média final, com uma cor VERDE caso o aluno seja Aprovado, e VERMELHO caso ele seja reprovado. Para determinar o status do aluno, ele deverá obter uma média final igual ou superior a 6.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

### RF029 – Deletar notas

**Caso de Uso:** Gerenciar notas.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar uma nota, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado, mensagem de alerta perguntando se o usuário realmente deseja excluir a nota.

**Saídas e pós-condição:** Mensagem informando que a nota selecionadas foram excluídas.

### RF030 – Editar notas

**Caso de Uso:** Gerenciar notas.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar as notas, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Mensagem informando que a nota selecionada foram editada.

### RF031 – Criar horário

**Caso de Uso:** Gerenciar horário.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa criar o horário de aula das disciplinas que ele terá durante a semana. Para isso, o usuário irá fornecer: nome da disciplina, dia da semana que terá aula da disciplina, hora de início e fim da aula.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

**Caso de Uso:** Gerenciar horário.

**Ator:** Usuário, Administrador.

**Descrição:** o horário deve estar acessível a partir do menu lateral, em uma guia chamada "Meu Horário".

### RF032 – Exibir horário

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

### RF033 – Editar horário

**Caso de Uso:** Gerenciar horário.

**Autor:** Usuário, Administrador.

**Descrição:** Para editar o horário, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

### RF034 – Cadastrar eventos

**Caso de Uso:** Gerenciar eventos.

**Autor:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa cadastrar eventos do seu interesse, onde o usuário irá fornecer: nome do evento, dia do evento, horário de início e fim.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Confirmação do cadastro do evento ou erro relacionado ao cadastro.

### RF035 – Listar eventos

**Caso de Uso:** Gerenciar eventos.

**Autor:** Usuário, Administrador.

**Descrição:** Os eventos cadastrados no sistema, devem ser exibidas em uma área designada para a mesmo, denominada “Meus eventos”, para que esses eventos sejam exibidos, o usuário deverá estar autenticado no sistema. Também deve ser possível que o usuário possa selecionar um determinado evento e possa alterar os dados dele.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

## RF036 – Deletar eventos

**Caso de Uso:** Gerenciar eventos.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar um evento, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado, mensagem de alerta perguntando se o usuário realmente deseja excluir o evento.

**Saídas e pós-condição:** Mensagem informando que o evento selecionadas foi excluído.

## RF037 – Editar eventos

**Caso de Uso:** Gerenciar eventos.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar os eventos, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Mensagem informando que o evento selecionada foram editado.

## RF038 – Cadastrar frequência

**Caso de Uso:** Gerenciar frequência.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa cadastrar sua frequência acadêmica, o acadêmico irá fornecer: nome da disciplina, quantidade máxima que se pode faltar. Após isso, o acadêmico poderá ir incrementando as faltas, até o aplicativo o alertar que já está perto do limite máximo de faltas.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Confirmação do cadastro da frequência ou erro relacionado ao cadastro.

## RF039 – Listar frequência

**Caso de Uso:** Gerenciar frequência.

**Ator:** Usuário, Administrador.

**Descrição:** As frequências cadastrados no sistema, devem ser exibidas em uma área designada para a mesmo, denominada “Minha frequência”, para que as frequência sejam exibidas, o usuário deverá estar

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

autenticado no sistema. Também deve ser possível que o usuário possa selecionar um determinado evento e possa alterar os dados dela.

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

#### RF040 – Deletar frequência

**Caso de Uso:** Gerenciar frequência.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar uma frequência, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado, mensagem de alerta perguntando se o usuário realmente deseja excluir a frequência.

**Saídas e pós-condição:** Mensagem informando que a frequência selecionada foi excluída.

#### RF041 – Editar frequência

**Caso de Uso:** Gerenciar frequência.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar a frequência, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Mensagem informando que a frequência selecionada foram editada.

#### RF042 – Cadastrar devolução de livro

**Caso de Uso:** Gerenciar devolução de livro.

**Ator:** Usuário, Administrador.

**Descrição:** Permitir que o usuário possa cadastrar sua devolução de livro, o usuário irá fornecer: nome do livro, autor do libro, detalhes sobre o livro e a data para devolver o livro.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Confirmação do cadastro da devolução de livro ou erro relacionado ao cadastro.

### RF043 – Listar devolução de livro

**Caso de Uso:** Gerenciar devolução de livro.

**Ator:** Usuário, Administrador.

**Descrição:** As devoluções de livros cadastrados no sistema, devem ser exibidas em uma área designada para a mesma, denominada “Devolução de livro”, para que as frequências sejam exibidas, o usuário deverá estar autenticado no sistema. Também deve ser possível que o usuário possa selecionar uma determinada devolução de livro e possa alterar os dados dela.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:**

### RF044 – Deletar devolução de livro

**Caso de Uso:** Gerenciar devolução de livro.

**Ator:** Usuário, Administrador.

**Descrição:** Para deletar uma devolução de livro, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado, mensagem de alerta perguntando se o usuário realmente deseja excluir a devolução de livro.

**Saídas e pós-condição:** Mensagem informando que a devolução de livro selecionada foi excluída.

### RF045 – Editar devolução de livro

**Caso de Uso:** Gerenciar devolução de livro.

**Ator:** Usuário, Administrador.

**Descrição:** Para editar a devolução de livro, o usuário deve estar autenticado no aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**Entradas e pré-condições:** Usuário autenticado.

**Saídas e pós-condição:** Mensagem informando que a devolução de livro selecionada foram editada.

## A.2 Requisitos Não Funcionais

### RNF001 – Linguagem

**Descrição:** O aplicativo deve ser implementado utilizando a linguagem de programação mobile hibrida.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

### RNF001 – Linguagem

**Descrição:** O aplicativo será de vital importância para o aluno, podendo ser utilizado diariamente, logo, o sistema deve conter uma interface interativa para usuários primários, e não se tornar cansativo aos usuários mais experientes. O aplicativo deve apresentar um tutorial de utilização, como também contar com uma opção de pular tutorial.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

### RNF003 – Desempenho

**Descrição:** O aplicativo será de vital importância para o aluno, podendo ser utilizado diariamente, logo, o sistema deve conter uma interface interativa para usuários primários, e não se tornar cansativo aos usuários mais experientes. O aplicativo deve apresentar um tutorial de utilização, como também contar com uma opção de pular tutorial.

**Prioridade:**      ○ Essencial      • Importante      ○ Desejável

### RNF004 – Compatibilidade

**Descrição:** O aplicativo necessita funcionar em qualquer versão do sistema Android ou IOS.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

### RNF005 – Notificação

**Descrição:** O aplicativo deverá solicitar ao usuário permissão para notifica-lo acerca de alguns recursos do aplicativo.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável

**RNF006 – Armazenamento**

**Descrição:** O aplicativo deverá solicitar permissão ao usuário para ter acesso ao armazenamento do local do aparelho.

**Prioridade:**      • Essencial      ○ Importante      ○ Desejável