UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANDERSON LENTZ DA SILVA

Aplicativo Móvel Android de Gestão e Acompanhamento de Alunos com Necessidades Educacionais Especiais

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Leandro Krug Wives
Coorientador: Prof. Me. Francisco Dutra dos Santos Junior

Porto Alegre
2019

AGRADECIMENTOS

Em primeiro lugar, a Deus toda honra por Ele ser doador de todas as boas dádivas que temos recebido.

Agradeço aos meus familiares, em especial minha mãe Traudi Lentz, meu pai Marcos Elias, e minha irmã Francine Lentz por todo o suporte ao longo dos anos de graduação. Nos momentos mais difíceis e alegres eles sempre estiveram ao meu lado com uma palavra de ânimo e consolação. Em especial, à minha namorada Aline de Freitas, meu agradecimento pela paciência e apoio dados durante esse período importante da minha vida.

A família Silveira Zabaleta merece toda a minha gratidão, pois por dois anos me ofereceu um lugar em seu lar. Eles foram fundamentais para que eu permanecesse em Porto Alegre e pudesse prosseguir com a faculdade.

Ao meu orientador professor Leandro Wives e co-orientador Francisco Dutra, que de maneira acessível conduziram esta orientação e deram todas as condições para que este trabalho pudesse ser realizado, meu muito obrigado por todo o apoio.

RESUMO

Aplicativos móveis fazem parte do cotidiano da grande maioria dos brasileiros e, com isso, seu uso na educação vem aumentando a cada dia. Tendo em vista os professores que possuem alunos com necessidades especiais de aprendizado, o propósito deste trabalho é desenvolver uma aplicação móvel em Android. Essa aplicação é na verdade uma continuidade de uma aplicação existente, que foi desenvolvida previamente para iOS. Desse modo, o acesso a solução desenvolvida será facilitado para toda a rede pública de educação beneficiando professores, alunos e suas respectivas famílias.

Palavras-chave: Android. Aplicativo móvel. Engenharia de software. Necessidades Educacionais Especiais.

Android Mobile Application for Managing Students with Special Educational Needs

ABSTRACT

Mobile applications are part of the daily life of the vast majority of Brazilians and their use in education is increasing every day. Because of teachers who have students with special learning needs, the purpose of this paper is to develop a mobile application on Android. This application is a continuation of a previous work, in which an iOS application was build for the same purpose. Thus, access to the developed solution will be facilitated for the entire public education network benefiting teachers, students, and their respective families.

Keywords: Android, Mobile Application, Software Engineering, Special Educational Needs.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AEE	Atendimento Educacional Especializado
AOT	Ahead-of-time
API	Application Programming Interface
APP	Application
ART	Android Runtime
FAB	Floating Action Button
GoF	Gang Of Four
HAL	Hardware Annotation Library
IDE	Integrated Development Environment
iOS	iPhone Operational System
JDK	Java Development Kit
JIT	Just-in-time
JSON	JavaScript Object Notation
NEE	Necessidades Educacionais Especiais
NoSQL	Not Only Structured Query Language
SDK	Software Development Kit
SQL	Structured Query Language
SO	Sistema Operacional
SUS	System Usability Scale
UI	User Interface
UX	User Experience
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A Constituição Federal de 1998, através do Ministério de Educação, estabelece o direito de todas as pessoas à educação. A Política Nacional de Educação Especial na Perspectiva da Educação Inclusiva (janeiro de 2008) juntamente com o Decreto Legislativo nº 186, de julho de 2008 instituiu as Diretrizes Educação Especial para o Atendimento Educacional Especializado – AEE na educação básica e regulamentado pelo Decreto nº 6.571, de 18 de setembro de 2008 (ESPECIAL, 2009).

Dessa maneira, os sistemas educacionais devem matricular os alunos com algum tipo de necessidade educacional especial (como exemplo, transtornos globais do desenvolvimento ou superdotação) nas escolas comuns do ensino regular e ofertar o atendimento educacional especializado, promovendo o acesso e as condições para uma educação de qualidade (ESPECIAL, 2009).

Nesse contexto, cabe ao professor acompanhar e prestar um atendimento especializado, por exemplo, através da elaboração de uma adequação curricular que visa a adaptação da forma de ensino de acordo com as necessidades específicas de cada aluno. Assim, durante a vida acadêmica dos alunos, os professores acabam gerando uma série de documentos, costumeiramente em formato de formulário e sem um padrão definido, contendo os planejamentos, pareceres e registros que, por exemplo, a cada final de trimestre, podem ser anexados na elaboração de um documento que informa o atendimento prestado.

Importante salientar a existência de um aplicativo iOS desenvolvido pelo Átila Silva, fruto do seu trabalho de graduação do curso de Ciência da Computação da Universidade Federal do Rio Grande do Sul, onde é proposto a facilitação do processo de captura, descrição, armazenamento de registros, geração de documentos (pareceres e documentos de adequação curricular), pelo professor, de forma segura e controlada (SILVA, 2019). Inclusive, em suas considerações finais, Silva (2019) menciona que era necessário desenvolver uma aplicação para a plataforma Android para que um número maior de pessoas pudesse ter acesso à solução e, sob essa necessidade, o desenvolvimento deste trabalho tornou-se possível.

Portanto, o aplicativo aqui proposto é uma forma complementar do trabalho de (SILVA, 2019) e tem como objetivo o desenvolvimento de uma aplicação para dispositivos móveis Android que visa a gestão e o acompanhamento, por parte do professor, aos seus alunos com necessidades especiais de aprendizado. Por ser um aplicativo na plataforma

Android, o número de professores beneficiados com a solução será aumentado através da oferta de uma alternativa que funciona de forma integrada ao aplicativo já desenvolvido para a plataforma iOS.

1.1 Estrutura

O presente trabalho está organizado em sete capítulos. Após o capítulo introdutório, os próximos capítulos seguem a seguinte disposição:

- **Capítulo 2**: apresenta a fundamentação teórica em detalhes para os principais conceitos utilizados. Também é demonstrada as tecnologias utilizadas para o desenvolvimento da aplicação.
- **Capítulo 3**: abordada a organização do projeto no seu processo de planejamento e implementação.
- **Capítulo 4**: apresenta as telas da aplicação e as suas correspondentes funcionalidades.
- **Capítulo 5**: apresenta a integração entre os aplicativos Android e iOS através de exemplos das principais telas da aplicação.
- **Capítulo 6**: apresenta o resultado do teste de usabilidade SUS.
- **Capítulo 7**: apresenta a conclusão bem como algumas questões que podem ser abordadas em trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E TECNOLOGIAS UTILIZADAS

O presente capítulo contém os conceitos relacionados ao desenvolvimento deste trabalho fornecendo a base teórica para a escolha das tecnologias utilizadas. Deste modo, é apresentado o sistema operacional Android juntamente com uma comparação com o sistema operacional iOS da Apple em termos de abrangência de mercado.

Na sequência serão apresentados os conceitos relacionados à arquitetura da aplicação, o padrão de projeto *Observer* e a concepção de banco de dados não-relacional.

Por fim, são introduzidas as tecnologias adotadas para o desenvolvimento da aplicação em termos de implementação da arquitetura, padrão de projeto e banco de dados.

2.1 Android

O Android é uma plataforma de software de código aberto e baseada em uma versão modificada do Linux. Originalmente desenvolvido pela companhia chamada Android Inc., o sistema foi adquirido em 2005 pela Google (LEE, 2012). Atualmente, o sistema operacional é amplamente utilizado em dispositivos móveis como smartphones, tablets e, além disso, outros equipamentos como televisores, painéis de automóveis e centros de mídia (JACKSON, 2012).

A adoção do Android pelo mundo tem reflexos claros na divisão de mercado que, de acordo com a International Data Corporation (IDC, 2019), prevê um aumento frente ao seu concorrente principal. Na Figura 2.1 é possível constatar que a previsão para o ano de 2019 é de um pequeno crescimento de 85.1% (2018) para 86.7% na fatia de mercado.

Figura 2.1: Fatia de Mercado para Smartphones

Year	2017	2018	2019	2020	2021	2022	2023
Android	85,1%	85,1%	86,7%	86,6%	86,9%	87,0%	87,1%
iOS	14,7%	14,9%	13,3%	13,4%	13,1%	13,0%	12,9%
Others	0,2%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
TOTAL	**100,0%**	**100,0%**	**100,0%**	**100,0%**	**100,0%**	**100,0%**	**100,0%**

Fonte: (IDC, 2019)

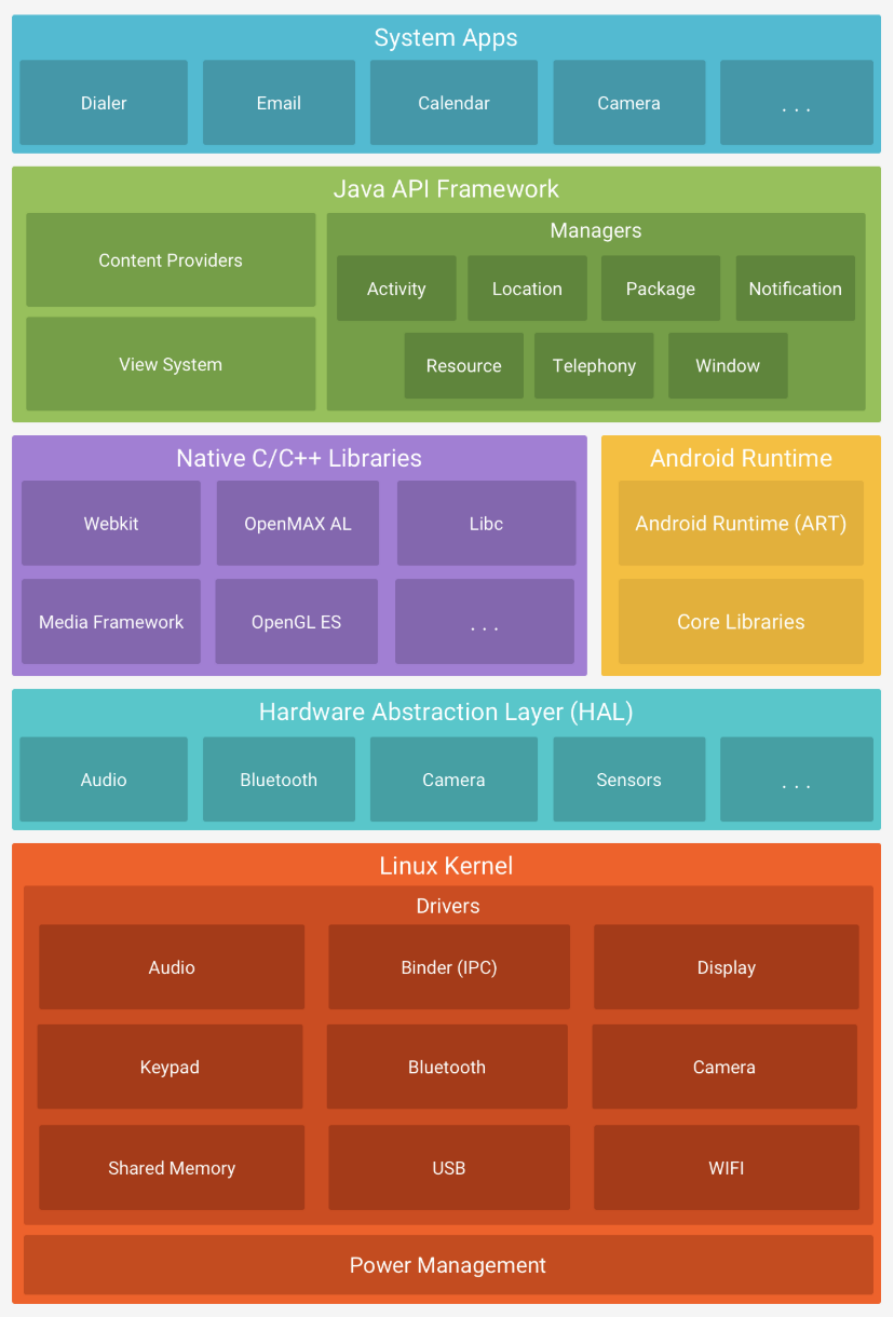
No Brasil, os dados acompanham as pesquisas mundiais, mostrando que mais de 86% da fatia de mercado é composta de dispositivos com o sistema operacional Android (STATS, 2019). Deste modo, fica claro que um número maior de usuários será alcançado

através do desenvolvimento do projeto proposto neste trabalho.

2.1.1 Arquitetura Android

A arquitetura do sistema operacional Android é uma pilha de componentes dividida em cinco seções e quatro camadas principais (POINT, 2019). Na Figura 2.2, são descritos os principais componentes da arquitetura do Android e detalhados nos parágrafos a seguir:

Figura 2.2: Arquitetura da Plataforma Android



Fonte: (ANDROID, 2019)

2.1.1.0.1 Aplicativos do sistema O Android vem com uma série de aplicativos principais como: e-mail, mensagem SMS, calendário, navegação na *internet* e contatos. Os usuários podem escolher entre usar os APPs fornecidos pelo sistema ou utilizar aplicativos de terceiros, por exemplo, para a navegação na *web*.

2.1.1.0.2 Framework JAVA API Todo o conjunto de características do sistema operacional Android é disponibilizado por meio de APIs escritas em Java as quais os desenvolvedores tem completo acesso. Assim, o desenvolvimento de aplicativos Android é simplificado pelo reuso de componentes e serviços modulares. Um exemplo de utilização destas APIs é a criação de elementos de UI como botões, caixas de texto e listas.

2.1.1.0.3 Bibliotecas C/C++ Nativas Muitas das aplicações base do Android são escritas em C ou C++ pois requerem linguagem nativa. Um exemplo de biblioteca escrita em linguagem nativa é o OpenGL ES para a manipulação e desenho de gráficos em 2D e 3D. Tais bibliotecas são expostas aos programadores através da API em Java.

2.1.1.0.4 Android Runtime Android Runtime (ART) é escrito para rodar múltiplas máquinas virtuais em dispositivos de pouca memória. As principais funcionalidades do ART são: compilação ahead-of-time (AOT) ou just-in-time (JIT), coletor de lixo otimizado, melhor suporte para depuração e, nas versões do Android 9 e superiores, conversão de pacotes no formato DEX para código de máquina mais compacto.

2.1.1.0.5 Camada de Abstração de Hardware A camada de abstração de hardware (HAL, em inglês) oportuniza uma interface padrão para a exposição das capacidades de hardware do dispositivo à camada superior (*Framework* API Java). A HAL consiste de módulos que são invocados quando um tipo específico de hardware é chamado. Portanto, quando a API chama, por exemplo, a câmera do dispositivo, o sistema carrega o módulo do *hardware* correspondente.

2.1.1.0.6 Kernel Linux O *kernel* do Linux é o fundamento da plataforma Android e é sobre esta camada que o Android Runtime delega as funcionalidades básicas como *threads* e gerenciamento de memória em baixo nível.

2.1.2 Atividade

Uma atividade, ou *Activity* em inglês, é um componente de aplicativo que fornece uma tela em que os usuários podem interagir através de uma janela (GOOGLE, 2019c). As atividades utilizam *fragments* e *views* para mostrar as informações e reagir às ações do usuário, assim, elas representam a camada de apresentação da aplicação (MEIER; LAKE,

2018).

Os aplicativos podem ter várias atividades sendo que a primeira a ser mostrada para usuário na primeira vez da execução da aplicação é denominada “principal”. É possível iniciar outras atividades para a execução de outras ações e fica a cargo do sistema em conservar a nova atividade em uma “pilha de retorno” (GOOGLE, 2019c).

Dado uma inicialização de uma nova atividade, a atividade que está sendo interrompida é notificada através de métodos de retorno de chamada do ciclo de vida da atividade (GOOGLE, 2019c). A seção a seguir relata os métodos com mais detalhes.

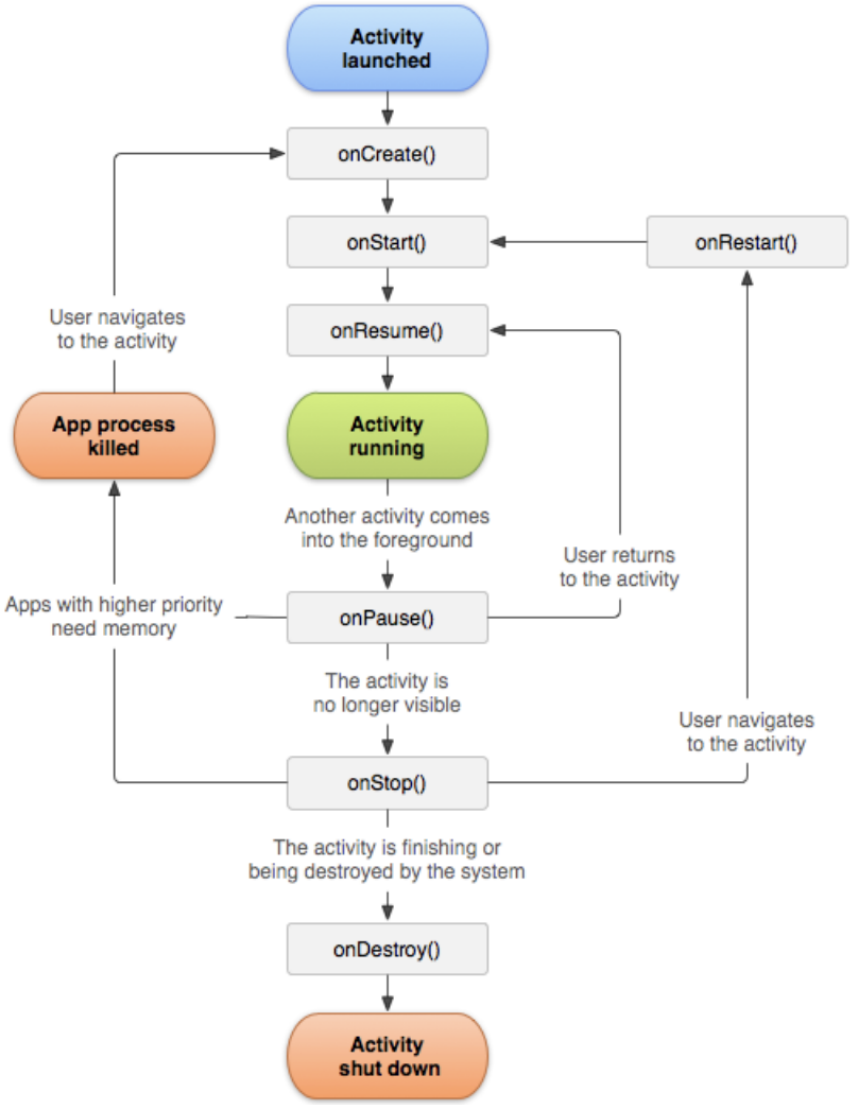
2.1.3 Ciclo de vida da Atividade

O conceito de ciclo de vida da atividade é fundamental para o desenvolvimento de aplicações móveis Android e é uma das primeiras e principais habilidades necessárias a ser dominada (THIENGO, 2019). Quando o usuário navega dentro da aplicação, as atividades transacionam entre diferentes estados. Para que a atividade tenha noção de qual estado foi alterado, a classe Activity oferece alguns *callbacks* como, por exemplo, quando o sistema está pausando uma atividade corrente (GOOGLE, 2019a).

De acordo com (SPäTH, 2018), ao contrário das tradicionais aplicações *desktop*, os aplicativos Android estão sujeitos ao sistema operacional e podem ser “mortos” a qualquer momento que o SO decidir. Portanto, cabe ao programador estar ciente destes momentos através do ciclo de vida para tomar as melhores decisões a fim de que aplicação reaja da melhor forma a fim de garantir um funcionamento correto do aplicativo.

Deste modo, a classe Activity fornece seis *callbacks* (GOOGLE, 2019a) ilustrados visualmente na Figura 2.3 e relacionados, com mais detalhes, nos parágrafos abaixo.

Figura 2.3: Ciclo de vida da atividade



Fonte: (GOOGLE, 2019a)

2.1.3.0.1 onCreate() O método é executado sempre que o sistema inicia uma atividade nova. É neste método que é feita as configurações da *activity*, por exemplo, carregar a *view* pela chamada da função *setContentView()* (GRIFFITHS; GRIFFITHS, 2017).

2.1.3.0.2 onStart() O método é executado após o *onCreate()* e no momento em que uma atividade anterior que estava em *background* volta a ter o foco (SILVEIRA, 2010).

2.1.3.0.3 OnResume() Através deste método o sistema fica a cargo de indicar que a atividade está ativa e apta para receber *input*, ou seja, a atividade está no topo da pilha e visível ao usuário (GOOGLE, 2019g).

2.1.3.0.4 OnPause() O presente método é invocado sempre que a atividade estiver saindo do primeiro plano podendo ocorrer, por exemplo, quando uma tela de diálogo estiver sendo exibida.

2.1.3.0.5 OnStop() Invocado pelo sistema, o presente método indica que o usuário está saindo da atividade e que a tela não estará visível.

2.1.3.0.6 OnDestroy() O método é chamado pelo sistema operacional toda a vez que a atividade estiver prestes a ser destruída.

2.1.4 Fragment

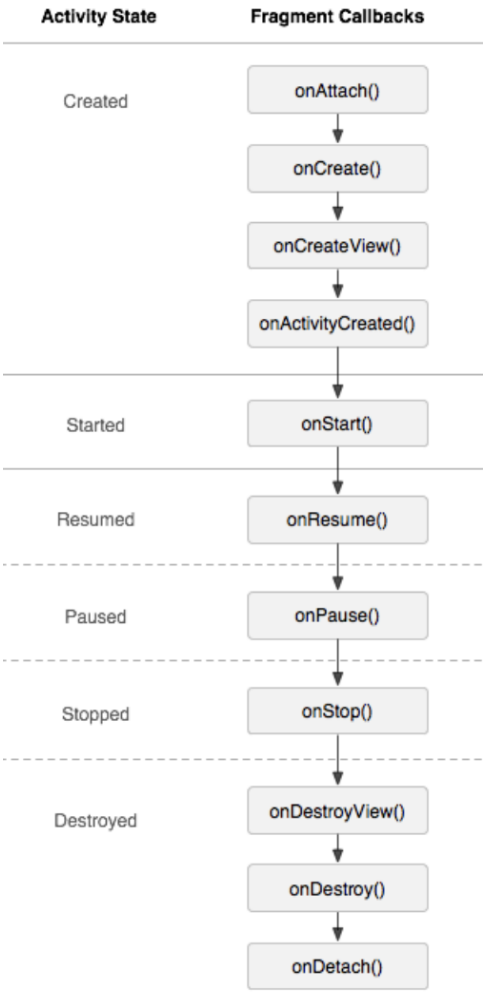
Um *Fragment*, ou Fragmento, geralmente é parte de uma interface de usuário da atividade e contribui com seu próprio *layout*. Os *fragments* podem ser adicionados ou removidos dinamicamente em uma atividade contendo partes da interface do aplicativo enquanto o mesmo está sendo executado (BAYLISS, 2018).

De acordo com a documentação oficial (DEVELOPERS, 2019b) os *Fragments* devem sempre estar hospedados em uma *Activity*, possuem seu próprio ciclo de vida que é diretamente impactado pelo ciclo de vida da atividade principal.

2.1.5 Ciclo de vida do fragmento

O ciclo de vida de um fragmento é muito semelhante ao ciclo de vida da atividade, possui *callbacks* análogos ao da *Activity* que podem ser usados para se obter o estado corrente do fragmento. A figura (2.4) relaciona os métodos de chamada para o ciclo de vida do fragmento com o estado da atividade que o contém.

Figura 2.4: Ciclo de vida do fragmento



Fonte: (DEVELOPERS, 2019b)

Nos parágrafos seguintes, segundo a documentação oficial (DEVELOPERS, 2019b), serão relatados os cinco *callbacks* adicionais que não estão presentes no ciclo de vida da atividade que estão contidos no ciclo de vida do fragmento.

2.1.5.0.1 onAttach() O método é executado quando o fragmento estiver sido associado à alguma *Activity*.

2.1.5.0.2 onCreateView() O método é chamado no momento em que a hierarquia de visualizações é associada ao fragmento.

2.1.5.0.3 onActivityCreated() O presente *callback* é invocado no momento em que o método *onCreate()* da *Activity* é retornado.

2.1.5.0.4 onDestroyView() Chamado quando a hierarquia de visualização do fragmento estiver sendo removida.

2.1.5.0.5 onDetach() Chamado no momento em que o fragmento estiver sendo desassociado da *Activity*.

2.2 Padrões de Projeto

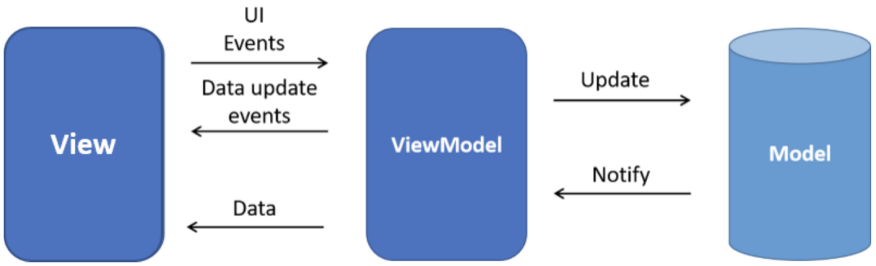
Esta seção apresenta os conteúdos teóricos relacionados a arquitetura da aplicação MVVM e do padrão de projeto *Observer* utilizados no desenvolvimento do aplicativo.

2.2.1 MVVM

O MVVM, abreviação de *Model-View-ViewModel*, é um padrão de projeto arquitetural originalmente revelado por John Gossman, arquiteto WPF e Silverlight da Microsoft, em seu blog no ano de 2005 (SMITH, 2009) que se propõe alcançar uma separação de prioridades através de três camadas representadas na Figura 2.5 e descritas abaixo (TEAM, 2019):

- **View**: responsável por mostrar a UI e informar às outras camadas sobre ações feitas pelo usuário.
- **ViewModel**: expõem informação para a *View*.
- **Model**: sabe como recuperar os dados e é responsável por expô-los para a *ViewModel*.

Figura 2.5: MVVM: interação entre as camadas



Fonte: (TEAM, 2019)

Como destaque desta arquitetura está a separação lógica entre a camada de visua-

lização (*View*) e a *Model* no sentido de que nenhuma referência direta é feita pela *Model* à *View*. A ligação entre esses dois componentes mencionados é feita através de *binding* com a *View* solicitando os dados para a *ViewModel* e ficando sob sua responsabilidade a atualização da tela com os dados recebidos.

As vantagens de se adotar este padrão arquitetural, segundo Team (2019), são:

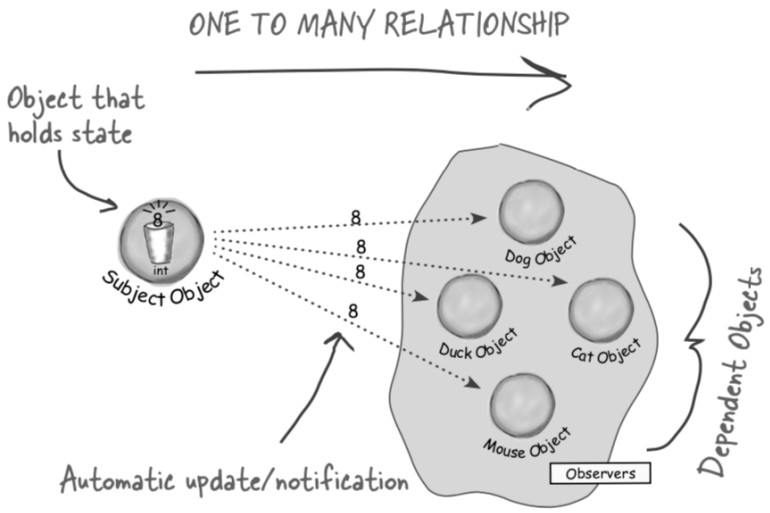
- *Controllers* menores: a adição da classe *ViewModel* permite remover código responsável pela lógica da aplicação, formatação e recuperação de dados da classe *View*. Como resultado, a *View* fica mais enxuta e focada na exibição dos dados.
- Testes: com o remanejamento de toda a manipulação de dados para a *ViewModel* a inclusão de testes unitários é facilitada já que a lógica de negócios não depende mais da *View*.
- Separação de responsabilidade: a adição da *ViewModel* visa a completa separação da *View* e, com isso, diminui a quantidade de código em outras camadas.

2.2.2 Padrão de Projeto *Observer*

O padrão de projeto *Observer* (Observador) é um padrão de comportamento, ou seja, tem por objetivo em observar como as responsabilidades são distribuídas no projeto e como a comunicação é realizada entre objetos (DOOLEY, 2017). De acordo com a Gang Of Four (GAMMA, 1995) o padrão de projeto *Observer* define uma dependência um-para-muitos entre objetos, assim quando um objeto muda seu estado todas as suas dependências são notificadas e atualizadas automaticamente.

A Figura 2.6 retrata a relação entre o objeto que mantém o estado, chamado Sujeito (*Subject* ou *Publisher*), e os objetos que recebem as atualizações de novos estados que são chamados de Observadores (*Observers*).

Figura 2.6: Relacionamento um-para-muitos do padrão *observer*



Fonte: (FREEMAN, 2004)

2.3 Android Architecture Components

Para o desenvolvimento deste trabalho as tecnologias utilizadas para a implementação da arquitetura da aplicação e do padrão de projeto *Observer* foram baseadas nos componentes de arquitetura do Android. De acordo com a documentação oficial do Google (GOOGLE, 2019b), os componentes de arquitetura fazem parte de uma coleção que auxilia a projetar uma aplicação robusta, testável e de fácil manutenção. Nas subseções a seguir serão detalhadas as tecnologias implementadas no aplicativo oriundas do *Android Architecture Compoments*.

2.3.1 ViewModel

A classe *ViewModel* é projetada para armazenar e gerenciar dados relacionados a UI de uma forma consciente do ciclo de vida (GOOGLE, 2019h). Com isso, além de se obter um maior desacoplamento de código, a classe fornecida presta um grande auxílio na medida em que ela mantém os dados a salvo em caso de mudança de configuração como, por exemplo, rotação de tela (MURPHY, 2018).

A comunicação entre a camada ViewModel com a View é feita através de dados observáveis (*LiveData*) e por meio do uso de *data binding* e são detalhados nos tópicos

seguintes.

2.3.2 LiveData

O *LiveData* é uma classe que implementa o padrão *Observer* que mantém dados observáveis e, ao contrário de observadores regulares, tem uma completa ciência do ciclo de vida da aplicação fazendo que ela respeite o ciclo de vida de outros componentes como atividades, *fragments* ou serviços (DEVELOPERS, 2019d). Portanto, fica sob responsabilidade da classe informar aos observadores as mudanças nos dados por ela mantidos e somente os observadores em estado ativo no seu ciclo de vida receberão as atualizações. Deste modo, o programador não precisa remover os observadores pois a classe faz automaticamente quando o estado da atividade entra em destruído.

De acordo com a documentação oficial (DEVELOPERS, 2019d), as vantagens de se usar a classe *LiveData* são:

2.3.2.0.1 Garantia que a UI reflete o estado do dado Como o *LiveData* segue o padrão *Observer* os objetos são notificados quando há uma mudança no seu estado. Assim, a interface pode ser atualizada mantendo a consistência com o estado corrente do objeto.

2.3.2.0.2 Sem vazamento de memória Os observáveis são ligados ao objeto *LifeCycle* e são limpados/removidos quando o seu estado associado é destruído.

2.3.2.0.3 Sem *crashes* quando a atividade é parada Se o estado do observador está inativo, estão o observador não recebe nenhuma atualização.

2.3.2.0.4 Nenhuma necessidade de manejar manualmente o ciclo de vida O programador não precisa manualmente parar ou resumir uma observação. A classe LiveData faz automaticamente este gerenciamento já que está ciente do estado do ciclo de vida do observável.

2.3.2.0.5 Dados sempre atualizados Sempre que o dado observado recebe as últimas atualizações, suponha que que a atividade esteja em *background*, ao retornar seu estado para *foreground* o dado será automaticamente atualizado.

2.3.2.0.6 Mudanças de configurações apropriadas Se houver uma mudança de configuração, por exemplo, o dispositivo é rotacionado, imediatamente o dado recebe a última atualização disponível.

2.3.2.0.7 Compartilhamento de recursos Fica fácil o compartilhamento de dados pois qualquer serviço da aplicação pode observar o recurso caso necessite.

2.3.3 Data Binding Library

A biblioteca *Data Binding Library* é fornecida para ligar componentes de UI dos layouts com as fontes de dados do APP de forma declarativa e não programaticamente (GOOGLE, 2019f) disponibilizando, assim, uma maneira mais simples do que o uso da função *findViewById()*.

Até o surgimento do *Android Architecture Compoments*, a maneira mais comum para se ligar os *layouts* com o código era através da função *findViewById()*. Todavia, a sua utilização torna o código muito repetitivo e "verboroso" (COSTA, 2018).

Como exemplo, a Figura 2.7 ilustra a utilização do método *findViewById()* para vincular um campo de texto com a propriedade *userName*.

Figura 2.7: Exemplo de uso da função findViewById

```
findViewById<TextView>(R.id.sample_text).apply {
    text = viewModel.userName
}
```

Fonte: (GOOGLE, 2019f)

Já na Figura 2.8 podemos observar que o dado é vinculado no campo de texto dentro do arquivo XML do *layout*.

Figura 2.8: Exemplo de uso do Data Binding

```
<TextView
        android:text="@{viewmodel.userName}" />
```

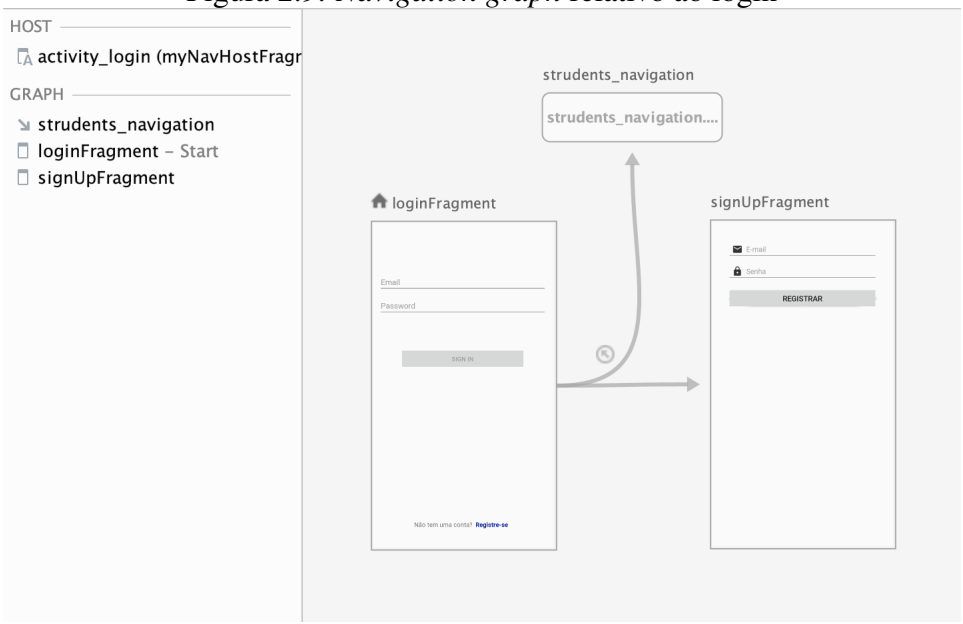
Fonte: (GOOGLE, 2019f)

2.3.4 Navigation

O *Navigation* é um novo componente de arquitetura que facilita a implementação de transições entre diferentes telas resultando em uma experiência de usuário consistente e previsível (DEVELOPERS, 2019c). Segundo a documentação oficial (DEVELOPERS, 2019c), o padrão *Navigation* consiste de três partes essenciais, a saber:

- **Navigation graph**: Um recurso em XML que mantém todas as informações relacionadas à navegação do aplicativo. A Figura 2.9 retrata mostra o arquivo de navegação para o login, contendo o fluxo para o registro de um novo usuário e fluxo restante da aplicação (uma referência a outro *navigation graph - students_navigation*).

Figura 2.9: *Navigation graph* relativo ao login



Fonte: O Autor

- **NavHost**: Um *container* vazio que é responsável por mostrar as telas de destino presentes no grafo de navegação. O componente *Navigation* possui uma implementação padrão (*NavHostFragment*) que exibe outros *fragments* de destino.
- **NavController**: Ao realizar uma navegação de uma tela contida no *NavHost* , o *NavController* é responsável por executar a ação. Como exemplo, a Figura 2.10 mostra a chamada ao *NavController* no trecho de código em que o usuário clica no botão de adicionar um novo aluno.

Figura 2.10: Exemplo de uso do *NavController*

```
binding.fab.setOnClickListener { it: View!

    findNavController().navigate(R.id.action_studentsFragment_to_newStudentFragment)

}
```

Fonte: O Autor

De modo geral, dentro do aplicativo toda a transição de telas é realizada por meio do *NavController* através da especificação de um *fragment* de destino de modo que o conteúdo seja mostrado no *NavHost*. Dentre alguns benefícios que são inerentes ao uso deste padrão de navegação, são destacados os seguintes: a facilidade de manipular as transações entre fragmentos, o gerenciamento padrão de ações *Up* e *Back* e animações entre as transições de tela realizadas por padrão.

2.4 Banco de Dados NoSQL

O armazenamento de dados é uma questão essencial para o desenvolvimento deste aplicativo. O banco de dados utilizado é do tipo não-relacional (NoSQL), termo que é usado para qualquer dado gerenciado de maneira não-relacional, e que tem duas características principais: os seus dados não são armazenados em tabelas e a linguagem do banco de dados não é SQL (MEIER; KAUFMANN; KAUFMANN, 2019).

Existem algumas tecnologias que são tidas como padrão e que diferenciam alguns tipos de banco de dados não-relacionais. Segundo Meier, Kaufmann and Kaufmann (2019) os quatro modelos NoSQL principais são: *Key-value stores*, *Column family databases*, *Document stores* e *Graph databases*. A subseção a seguir detalha o modelo *Document store* que foi o escolhido para uso na aplicação desenvolvida.

2.4.1 Document store

O modelo *Document Store* contém dados estruturados em registros que são chamados de documentos. Os documentos são armazenados sem que haja um *schema* definido antes da inserção dos dados. Assim, a responsabilidade pela organização esquemática é toda do programador, fazendo com com que seja uma ótima escolha para prototipação de soluções ao permitir com que os dados mudem ao longo do desenvolvimento (FOWLER, 2015). Os documentos são organizados de maneira estruturada e hierárquica e que con-

tém subestruturas, podendo ser semiestruturados quando os formatos de dados JSON ou XML são usados (FOWLER, 2015).

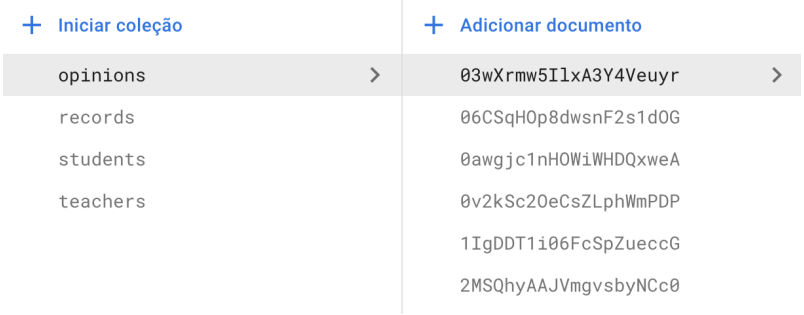
2.5 Firebase

Esta seção é relacionada ao serviço Firebase oferecido pela Google para a implementação do banco de dados utilizado no aplicativo. Foram utilizados dois serviços principais: Cloud Firestore para o banco de dados não relacional e o Cloud Storage para armazenamento de conteúdo gerado pelo usuário (fotos, vídeos e áudios).

2.5.1 Cloud Firestore

O serviço Cloud Firestore é um banco de dados não-relacional no modelo *document store* que permite o armazenamento de documentos na nuvem de forma hierarquizada, mantendo-os sincronizados oferecendo, assim, uma maneira fácil de recuperação e pesquisa em uma escala global (FIRESTORE, 2019). A Figura 2.11 mostra como está organizada a estrutura dos dados no Firebase.

Figura 2.11: Organização dos dados no Firestore



Fonte: O Autor

2.5.2 Cloud Storage

O serviço Cloud Storage permite o armazenamento de arquivos enviados por parte do aplicativo como fotos, vídeos ou imagens. Com isso, é possível relacionar os arquivos enviados pelos professores com os dados dos alunos cadastrados no banco de dados do Firestore. A Figura 2.12 mostra como é a organização dos diretórios de mídia utilizados

na aplicação.

Figura 2.12: Organização das pastas de arquivos no Storage

Fonte: O Autor

3 PROJETO E DESENVOLVIMENTO

Neste capítulo serão abordadas as escolhas adequadas do sistema de desenvolvimento e suas consequências ao projeto, a modelagem das entidades de dados, o *design* das telas da aplicação baseado no *Material Design*, as histórias de usuário implementadas e, por fim, o gerenciamento do projeto juntamente com a ferramenta Airtable.

3.1 Sistema de Desenvolvimento

Esta seção abordará a escolha do sistema do ambiente de desenvolvimento, o nível apropriado da versão do *Software Development Kit* utilizado para a programação e a linguagem escolhida para o desenvolvimento do aplicativo.

3.1.1 Android Studio

O Android Studio, desenvolvido pelo Google, é uma IDE para o desenvolvimento de aplicativos Android baseado no *IntelliJ IDEA* que foi utilizada durante a realização deste trabalho. A versão da ferramenta utilizada (3.5) disponibiliza uma série de funcionalidades que facilitam a vida do programador e, dentre algumas delas, são relacionadas as seguintes funcionalidades utilizadas no projeto:

3.1.1.0.1 Editor visual de telas Permite que o *layout* seja criado através do posicionamento dos elementos de forma visual, arrastando-os para a posição desejada. Por meio dessa funcionalidade é possível ajustar as distâncias entre os elementos via configuração de restrições resultando, assim, em telas mais complexas.

3.1.1.0.2 Emulador Permite que o aplicativo seja testado em um ambiente Android. Durante todo o projeto, sempre após alguma mudança, o APP era testado em um emulador. Com isso, era possível testar várias configurações do sistema e diferentes versões do Android.

3.1.1.0.3 Aplicador de mudanças Oferece uma maneira de aplicar as mudanças realizadas tanto em telas como em código sem que o aplicativo seja reinstalado proporcionando

um desenvolvimento mais rápido.

3.1.1.0.4 Editor de código inteligente Proporciona um desenvolvimento mais seguro, rápido e produtivo ao oferecer refatoração, análise e conclusão de código de forma avançada e eficiente.

3.1.2 Software Development Kit

Uma escolha principal que pode afetar a quantidade de usuários que a aplicação pode atingir é qual *Software Development Kit* (SDK) utilizar para o desenvolvimento do APP. Neste projeto, foi dada a prioridade por escolher um nível adequado de uma API que pudesse suportar uma grande maioria de usuários de forma a não prejudicar a programação (ao dar suporte a níveis de API defasados) fazendo com que não fosse possível utilizar *features* mais atualizadas.

A seleção do nível mínino da API a ser suportada foi baseada nos dados disponibilizados pela própria equipe do Google através de uma pesquisa sobre o número relativo de dispositivos em determinada versão do Android (DEVELOPERS, 2019a). Na Figura 3.1 podemos ver que a maioria dos aparelhos se encontra a partir da versão 16 da API.

Figura 3.1: Versões das plataformas Android

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Fonte: (DEVELOPERS, 2019a)

Portanto, foi determinada a API de versão 16 como o mínimo suportado pelo apli-

cativo fazendo com que todos os smartphones Android que possuem o sistema operacional a partir da versão 4.1 (API level 16) possam instalar a aplicação. Deste modo, em nível geral, o aplicativo abrange cerca de 99.4% dos dispositivos Android.

3.1.3 Kotlin

Para o desenvolvimento de uma aplicação nativa em Android, a plataforma disponibiliza duas linguagens principais: Java e Kotlin. Neste projeto, a linguagem Kotlin foi a escolhida por ser mais recente e por permitir uma codificação mais rápida, menos "verborosa" e ser parecida com linguagens mais modernas como o *Swift*. Segundo a documentação oficial (KOTLIN, 2019), as vantagens ao se usar a linguagem Kotlin para o desenvolvimento de aplicações Android são:

3.1.3.0.1 Compatibilidade O Kotlin é compatível com o JDK 6 fazendo com que aplicações possam rodar em dispositivos Android mais antigos sem problemas. Além disso, todas as ferramentas da linguagem são suportadas pelo Android Studio e compatíveis com o sistema de *build* do Android.

3.1.3.0.2 Performance Aplicações em Kotlin rodam tão rápido quando uma aplicação escrita em Java devido a similaridade da estrutura do *bytecode*.

3.1.3.0.3 Interoperabilidade O Kotlin é 100% interoperável com o Java. Assim, é possível utilizar todas as bibliotecas do Android escritas em Java na aplicação escrita em Kotlin.

3.1.3.0.4 Tempo de compilação O Kotlin suporta compilação incremental que, geralmente, é mais rápido do que em Java.

3.2 Modelagem de Entidades

Nas subseções abaixo serão apresentadas as entidades modeladas cujo o padrão estabelecido no trabalho do Atila (SILVA, 2019) foi mantido com o mínimo de mudanças possíveis na perspectiva de facilitar a integração com o aplicativo desenvolvido para a

plataforma iOS.

3.2.1 Teacher

Entidade que modela o professor e possui campos que ajudam a relacioná-lo com outras entidades como: aluno, parecer e registro diário. A seguir, são detalhados os campos implementados no Firebase:

- **id**: *String* gerado unicamente pelo *Firebase*. Usado para distinguir cada professor
- **name**: *String* que representa o nome do professor.
- **email**: *String* que quarda o email do professor usado para *login* no aplicativo.
- **phoneNumber**: *String* que guarda o tefefone do professor.
- **timestamp**: *Number* contendo o *timestamp* da inserção do professor no banco de dados.

3.2.2 Student

Entidade que modela um estudante e contém suas informações principais do seu perfil. O documento *Student* é gerado quando o professor adiciona um novo aluno e enviado para o *Firebase*.

- **id**: *String* gerado unicamente pelo *Firebase*. Usado para distinguir cada professor.
- **name**: *String* contendo o nome completo do aluno.
- **fatherName**: *String* contendo o nome completo do pai do aluno.
- **motherName**: *String* contendo o nome completo da mãe do aluno.
- **birthDate**: *Number* que contém o *timestamp* da data de nascimento do aluno relativo a uma data padrão.
- **generalRegistry**: *String* que representa o Registro Geral do aluno.
- **history**: *String* em que o professor pode adicionar o histórico do aluno.
- **classNumber**: *String* que contém a turma que o aluno está matriculado.
- **series**: *String* que representa a série em que o aluno atualmente se encontra.
- **shift**: *String* que contém o turno escolar do aluno.
- **phoneNumber**: *String* que armazena o telefone do responsável pelo aluno.

- **responsibleName**: *String* que armazena o nome completo da pessoa que é responsável pelo aluno.
- **relationship**: *String* que contém o grau de parentesco do responsável pelo aluno.
- **address**: *String* que armazena o endereço do aluno.
- **cid**: *String* que contém o código internacional de doenças que o aluno possui.
- **specialNeeds**: *List* contendo as necessidades especiais do aluno.
- **termsOfUse**: *Boolean* que indica se o responsável está de acordo com o uso dos dados do aluno.
- **schoolId**: *String* que indica a qual escola o aluno está cadastrado.
- **teacherId**: *String* armazena o identificador do professor que cadastrou o aluno.

3.2.3 Opinion

A entidade que mantém os documentos de pareceres ou adequações curriculares adicionadas por um professor e que são relacionados a um determinado aluno. A diferenciação entre um parecer ou uma adequação curricular é feita dentro do campo *content* por uma chave chamada *type*.

- **id**: *String* única gerada pelo Firebase no ato de inserção deste documento.
- **comment**: *String* que guarda um comentário feito pelo professor.
- **content**: *Map* chave-valor contendo os dados de um parecer ou de uma adequação curricular.
- **studentId**: *String* que contém o identificador único do aluno ao qual pertence este documento.
- **teacherId**: *String* que armazena o identificador do professor que adicionou este documento.
- **timestamp**: *Number* contendo o *timestamp* do momento da inserção do documento no Firebase.

3.2.4 Record

Entidade que modela um registro diário que representa uma espécie de evento que aconteceu com um aluno e que o professor desejou catalogar. Este documento é gerado

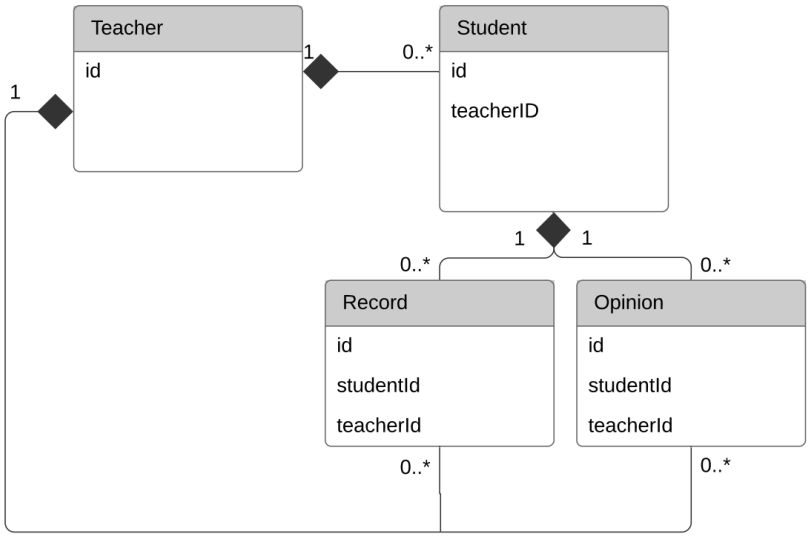
quando um professor realiza uma inserção contendo um dos tipos de mídia permitidos: texto, áudio, foto ou vídeo.

- **id**: *String* que contém o identificador único do documento que é gerado automaticamente pelo Firebase.
- **studentId**: *String* que contém o identificador único do aluno ao qual pertence este documento.
- **teacherId**: *String* que armazena o identificador do professor que adicionou este documento.
- **mediaType**: *String* que armazena o tipo de mídia que o registro diário contém.
- **comment**: *String* que contém um comentário personalizado e realizado pelo professor.
- **timestamp**: *Number* contendo a data de criação do documento.

3.2.5 Diagrama básico da modelagem de dados

As entidades do banco de dados modeladas e descritas acima podem ser visualizadas através do diagrama de entidade-relacionamento na Figura 3.2 onde apenas os campos de descrevem os relacionamentos entre os dados foram mantidos.

Figura 3.2: Diagrama entidade-relacionamento



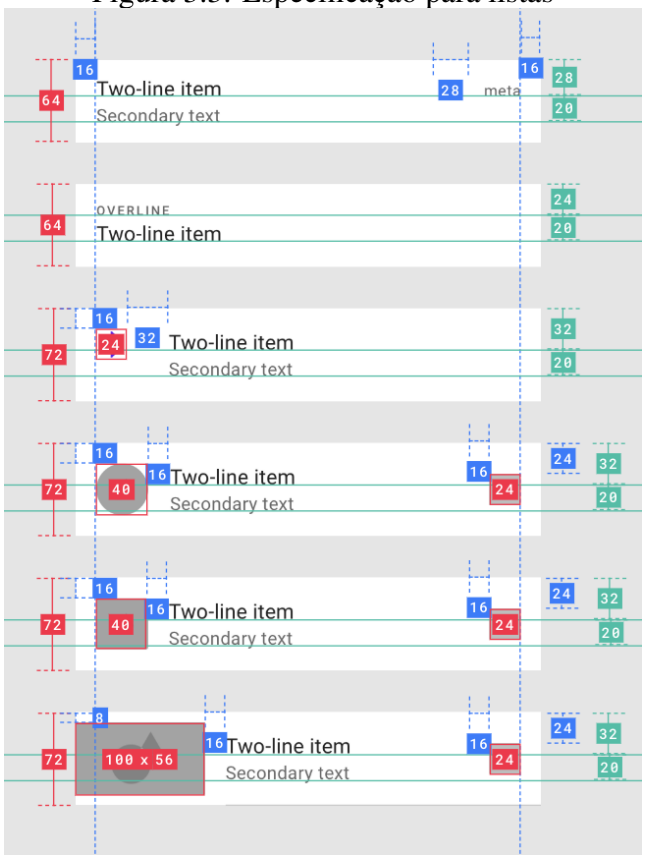
Fonte: O Autor

3.3 Material Design

O *Material Design* é uma linguagem de projeto que visa o estabelecimento de um padrão sobre a maneira com que os usuários interagem com a aplicação (BAYLISS, 2018). Para isso, existem algumas especificações de *design* que foram observadas neste trabalho fazendo com que todas as telas desenvolvidas fossem criadas seguindo essa linha de projeto de interface: cores utilizadas, formato dos botões e disposição dos elementos na tela. Desta maneira, os usuários terão uma experiência concisa de UI e UX nos moldes da plataforma Android.

Como exemplo de utilização das especificações na composição do *layout*, é ilustrada na figura 3.3 a recomendação do *Material Design* que visa o projeto de telas que contém itens em uma lista. Na figura 4.4 pode-se observar a implementação da tela de lista de estudantes que seguiu a recomendação mencionada para os itens que possuem uma imagem circular e duas linhas de texto.

Figura 3.3: Especificação para listas



Fonte: (DESIGN, 2019)

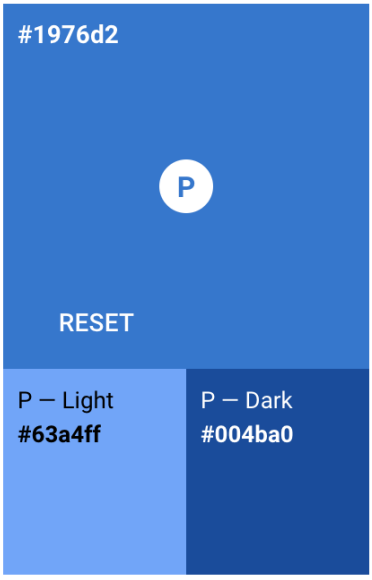
Outro tópico de suma importância e que merece muito cuidado e atenção é a de-

finição do sistema de cores. Para isso, foi utilizada a ferramenta *Color Tool* do Material Design (GOOGLE, 2019e) que ajuda a criar uma paleta de cores levando em conta, por exemplo, o contraste entre as cores de texto e botões. Duas paletas de cores foram utilizadas e são destacadas a seguir.

3.3.1 Cor Primária

A cor primária, segundo o guia do Material Design (GOOGLE, 2019d), é a cor que mais fica visível nas telas e componentes da aplicação. Como é a cor que mais tempo é mostrada, por exemplo, na barra de ação no topo de todas as telas, optou-se por uma cor que evite cansar a vista dos usuários. A paleta e as respectivas cores em hexadecimal são mostradas na Figura 3.4.

Figura 3.4: Paleta de cor primária



Fonte: (GOOGLE, 2019d)

3.3.2 Cor Secundária

A cor secundária tem função na aplicação de ressaltar alguns elementos da tela e dar contraste em relação a item coloridos usando a paleta de cor primária. Um exemplo do seu uso pode ser observado no destaque feito ao botão no canto inferior direito da tela na Figura 4.3. A paleta completa de cor secundária é visível na Figura 3.5 a seguir.

Figura 3.5: Paleta de cor secundária

Fonte: (GOOGLE, 2019d)

3.4 Histórias de Usuário

As histórias de usuário utilizadas seguem o modelo proposto por Cohn (2008) e consistem do seguinte padrão: “Eu, como um <usuário>, quero <algum objetivo>, para <algum motivo>”.

Nas tabelas a seguir, são relacionadas as histórias de usuário contendo de forma descritiva as funcionalidades básicas que foram implementadas durante o projeto. Tendo em vista a existência da aplicação para iOS desenvolvido por Silva (2019), as mesmas *user stories* foram reutilizadas neste trabalho (com exceção da primeira história de usuário que foi adaptada para o aplicativo Android).

Tabela 3.1: História de Usuário 1

Eu, como professor, quero um aplicativo em Android, para que eu possa usar no meu dispositivo.

Fonte: O Autor

Tabela 3.2: História de Usuário 2

Eu, como professor, quero adicionar, remover e editar alunos, para poder adicionar eventos aos meus alunos.

Fonte: (SILVA, 2019)

Tabela 3.3: História de Usuário 3

Eu, como professor, quero me registar usando minhas credenciais (e-mail e senha), para que eu possa manter minhas informações seguras.

Fonte: (SILVA, 2019)

Tabela 3.4: História de Usuário 4

Eu, como professor, quero excluir alunos e eventos, para que eu possa organizar o meu trabalho.

Fonte: (SILVA, 2019)

Tabela 3.5: História de Usuário 5

Eu, como professor, quero que a interface do aplicativo seja perfeita e rápida de usar, para que eu possa registrar eventos de alunos com eficiência.

Fonte: (SILVA, 2019)

Tabela 3.6: História de Usuário 6

Eu, como professor, quero poder ver os eventos dos meus alunos, para eu possa acompanhar o progresso deles.

Fonte: (SILVA, 2019)

Tabela 3.7: História de Usuário 7

Eu, como professor, quero ver os eventos dos alunos em uma linha do tempo, para que eu possa visualizar claramente quando cada evento ocorreu.

Fonte: (SILVA, 2019)

Tabela 3.8: História de Usuário 8

Eu, como professor, quero ver os conteúdos dos eventos, para que eu possa ter uma melhor compreensão do evento.

Fonte: (SILVA, 2019)

Tabela 3.9: História de Usuário 9

Eu, como professor, quero fazer o envio de áudio, vídeo e imagens das realizações de um aluno, para que eu possa registrar suas realizações.

Fonte: (SILVA, 2019)

Tabela 3.10: História de Usuário 10

Eu, como professor, quero ser capaz de criar um parecer, para que não precise fazer isso fora do aplicativo.

Fonte: (SILVA, 2019)

Tabela 3.11: História de Usuário 11

Eu, como professor, quero ser capaz de criar adequações curriculares, para que não precise fazer isso fora do aplicativo.

Fonte: (SILVA, 2019)

Tabela 3.12: História de Usuário 12

Eu, como professor, quero que a interface do aplicativo seja perfeita e rápida de usar, para registrar eventos de alunos com eficiência.

Fonte: (SILVA, 2019)

3.5 Gerenciamento do Projeto

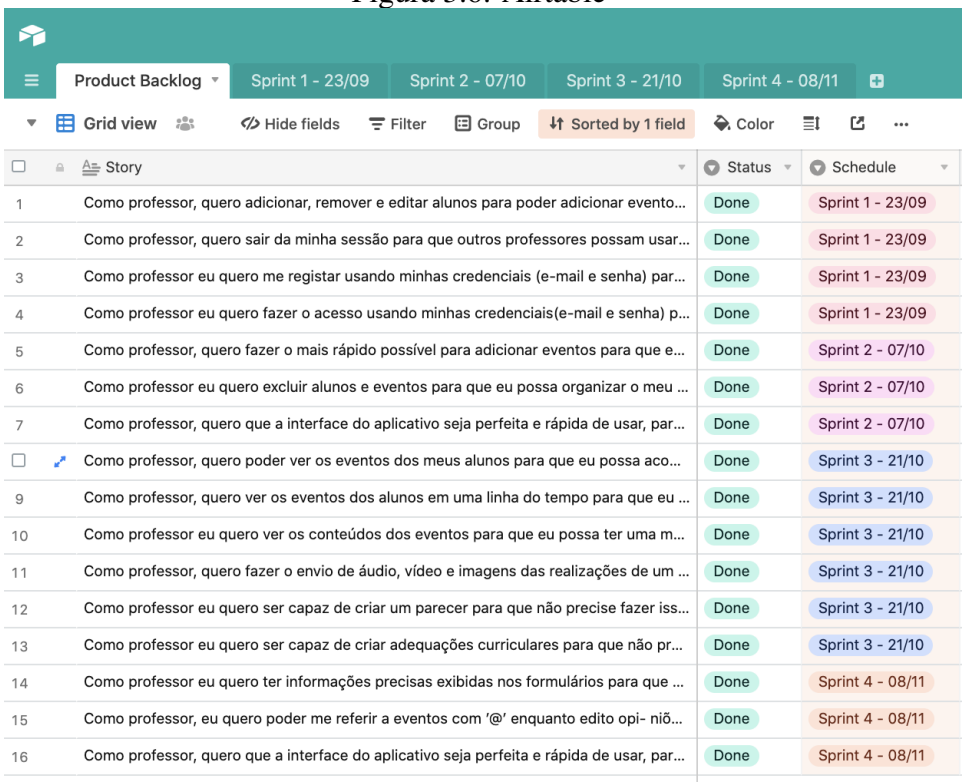
Para a organização do projeto, foi utilizada a ferramenta Airtable[1] onde foram elencadas as histórias de usuário em uma coluna denominada *Product Backlog* e gerenciadas através de quatro *sprints* com durações variadas entre duas a quatro semanas. Para cada história de usuário foram extraídas as tarefas que as compõem de modo que pudessem ser alocadas em uma das *sprints*. Assim, obteve-se uma boa imagem do que deveria ser desenvolvido em termos de programação para que cada *user story* pudesse ser aferida como realizada.

Um quadro baseado no sistema *Kanban* foi utilizado para a visualização das tarefas que deveriam ser realizadas, que estavam sendo realizadas e que já foram completadas.

[1]Disponível em: https://airtable.com

Portanto, para cada *sprint* corrente, o quadro era atualizado e fornecia uma maneira rápida de se obter o andamento em termos de tarefas realizadas. A figura 3.6 mostra uma visão geral do AirTable contendo o estado final do *produt backlog*.

Figura 3.6: Airtable



	Story	Status	Schedule
1	Como professor, quero adicionar, remover e editar alunos para poder adicionar evento...	Done	Sprint 1 - 23/09
2	Como professor, quero sair da minha sessão para que outros professores possam usar...	Done	Sprint 1 - 23/09
3	Como professor eu quero me registar usando minhas credenciais (e-mail e senha) par...	Done	Sprint 1 - 23/09
4	Como professor eu quero fazer o acesso usando minhas credenciais(e-mail e senha) p...	Done	Sprint 1 - 23/09
5	Como professor, quero fazer o mais rápido possível para adicionar eventos para que e...	Done	Sprint 2 - 07/10
6	Como professor eu quero excluir alunos e eventos para que eu possa organizar o meu ...	Done	Sprint 2 - 07/10
7	Como professor, quero que a interface do aplicativo seja perfeita e rápida de usar, par...	Done	Sprint 2 - 07/10
	Como professor, quero poder ver os eventos dos meus alunos para que eu possa aco...	Done	Sprint 3 - 21/10
9	Como professor, quero ver os eventos dos alunos em uma linha do tempo para que eu ...	Done	Sprint 3 - 21/10
10	Como professor eu quero ver os conteúdos dos eventos para que eu possa ter uma m...	Done	Sprint 3 - 21/10
11	Como professor, quero fazer o envio de áudio, vídeo e imagens das realizações de um ...	Done	Sprint 3 - 21/10
12	Como professor eu quero ser capaz de criar um parecer para que não precise fazer iss...	Done	Sprint 3 - 21/10
13	Como professor eu quero ser capaz de criar adequações curriculares para que não pr...	Done	Sprint 3 - 21/10
14	Como professor eu quero ter informações precisas exibidas nos formulários para que ...	Done	Sprint 4 - 08/11
15	Como professor, eu quero poder me referir a eventos com '@' enquanto edito opi- niõ...	Done	Sprint 4 - 08/11
16	Como professor, quero que a interface do aplicativo seja perfeita e rápida de usar, par...	Done	Sprint 4 - 08/11

Fonte: O Autor

A escolha das histórias de usuário a serem implementadas em cada *sprint* foi motivada pelas seguintes razões:

- ***Sprint* 1**: Funcionalidades básicas da aplicação como: fazer o login dado que existe um usuário no *Firebase*, *logout*, adicionar, remover e editar aluno. Foi levada em conta a configuração do banco de dados no *Firebase*.
- ***Sprint* 2**: Desenvolvimento das telas de adição e edição de um novo registro diário com conteúdo textual bem como a configuração do banco de dados para receber esta funcionalidade.
- ***Sprint* 3**: Adição de registros diários com conteúdo de áudio e vídeo, desenvolvimento das telas de novo parecer pedagógico, nova adequação curricular e a implementação no banco de dados destas funcionalidades.
- ***Sprint* 4**: Finalização da aplicação, opção de relacionar registros diários em um novo parecer pedagógico e em uma nova adequação curricular. Refatoração de

código e ajustes no *design* das telas.

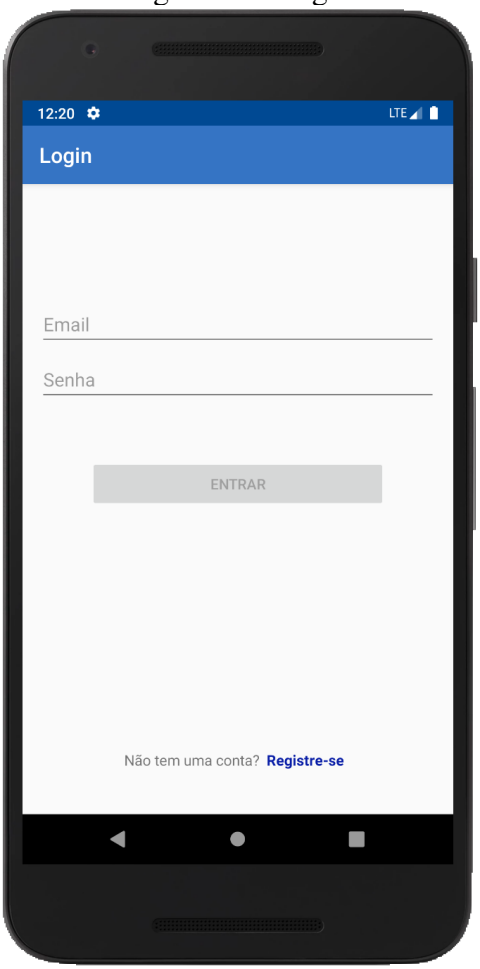
4 TELAS DO SISTEMA E FUNCIONAMENTO DO APLICATIVO

Nesta seção serão apresentadas as telas desenvolvidas para a aplicação e seus respectivos detalhes de funcionamento.

4.1 Tela de login

A primeira tela quando o usuário executa o aplicativo é a tela de login (Figura 4.1) fornecendo as opções de autenticação via e-mail e cadastro na parte inferior da tela. Caso o professor já esteja *loggado* então a tela de lista de estudantes é carregada automaticamente.

Figura 4.1: Login

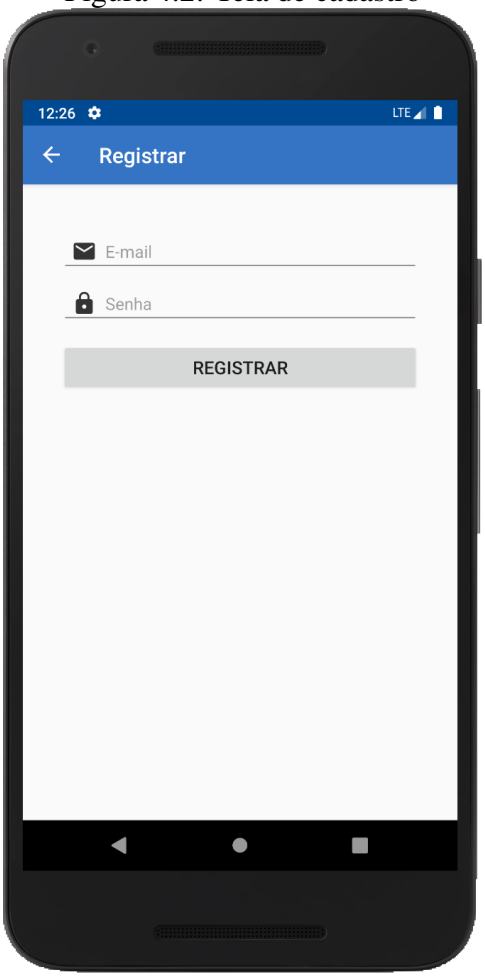


Fonte: O Autor

4.2 Cadastro

A tela de cadastro (Figura 4.2) fornece uma opção para os usuários que não possuem uma conta previamente cadastrada poderem utilizar a aplicação. Assim, mediante o preenchimento de seu e-mail e senha o sistema realizará um novo registo com os dados fornecidos através da camada de serviços do *Firebase*. Após realizado o cadastro, a tela de lista de alunos é automaticamente carregada.

Figura 4.2: Tela de cadastro



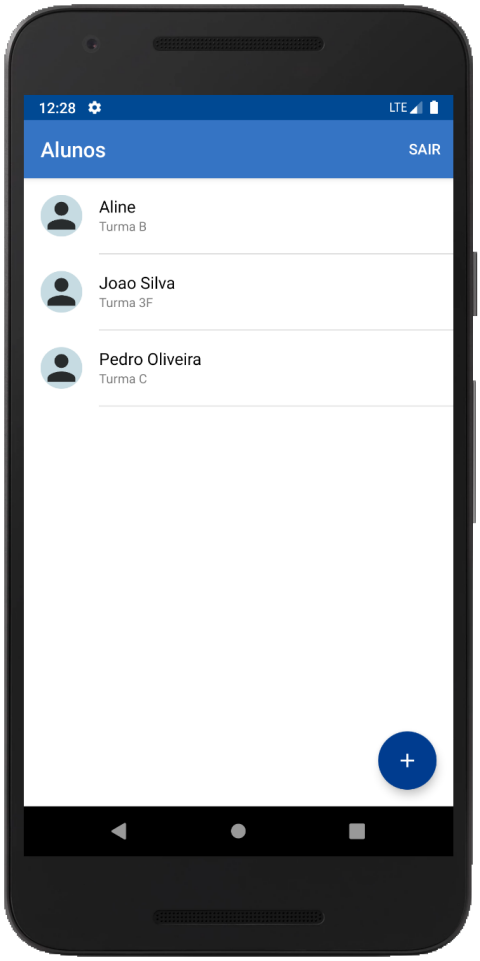
Fonte: O Autor

4.3 Lista de alunos

A tela de lista de alunos (Figura 4.3) é a interface principal do aplicativo onde o professor terá uma visão completa dos estudantes os quais acompanha. Seguindo os padrões de *design* do Google, o professor poderá cadastrar um novo aluno por meio do

botão no estilo FAB no canto inferior direito da tela. Ao clicar em um célula da lista, a tela de detalhe do aluno será exibida.

Figura 4.3: Tela de alunos

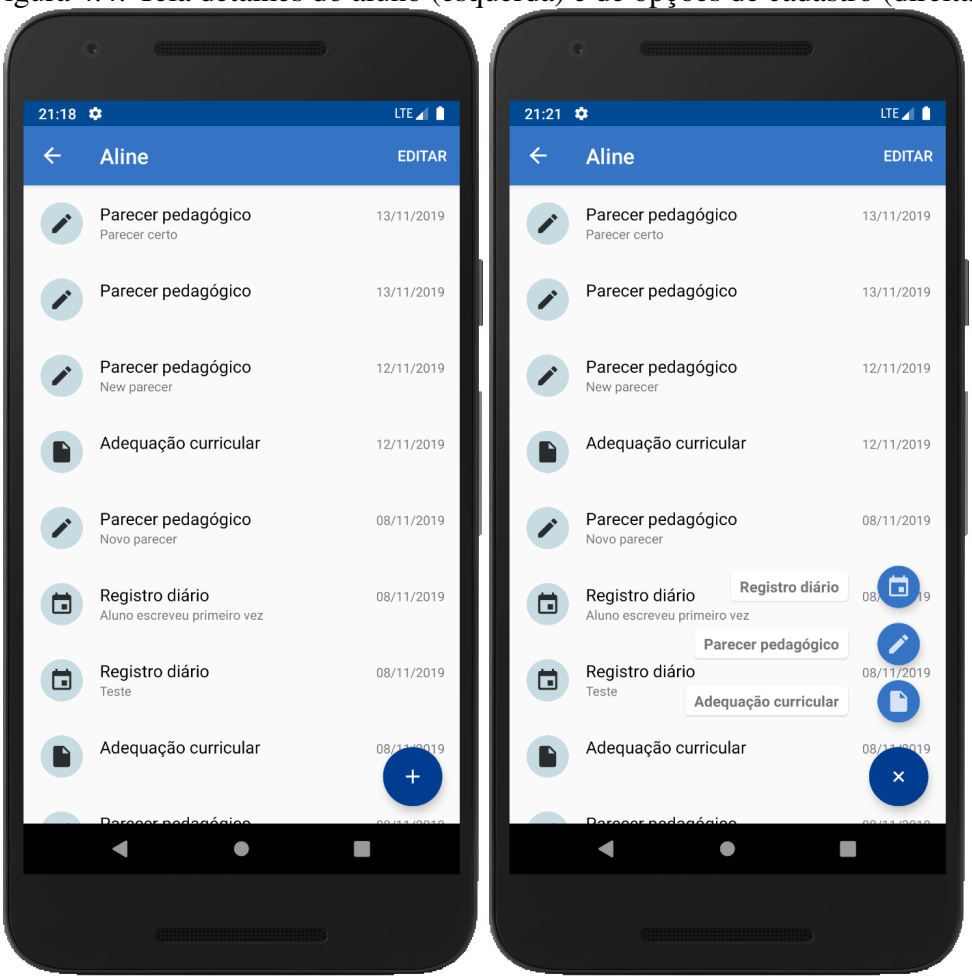


Fonte: O Autor

4.4 Detalhe do aluno

A tela de detalhe do aluno (Figura 4.3) permite ao professor ver os registros diários, pareceres e adequações curriculares cadastradas de forma organizada pela data de inserção. Ao utilizar o botão no estilo FAB no canto inferior direito, será mostrada as opções de cadastro de um novo registro diário, parecer pedagógico ou adequação curricular.

Figura 4.4: Tela detalhes do aluno (esquerda) e de opções de cadastro (direita)

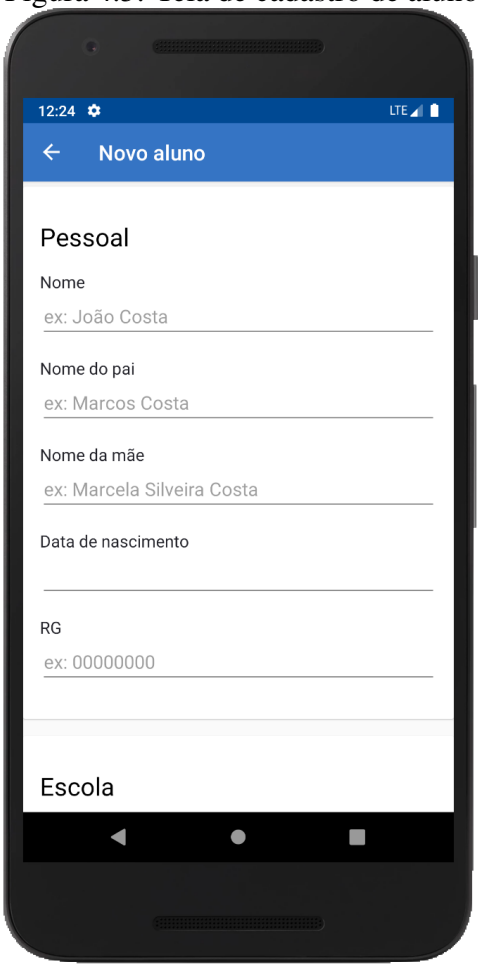


Fonte: O Autor

4.5 Cadastro de alunos

A tela de cadastro de aluno (Figura 4.5) provê as informações necessárias organizadas em *cards* que o professor poderá preencher. A tela é acessível através do botão de adicionar estudante visível no canto inferior direito na Figura 4.3.

Figura 4.5: Tela de cadastro de aluno

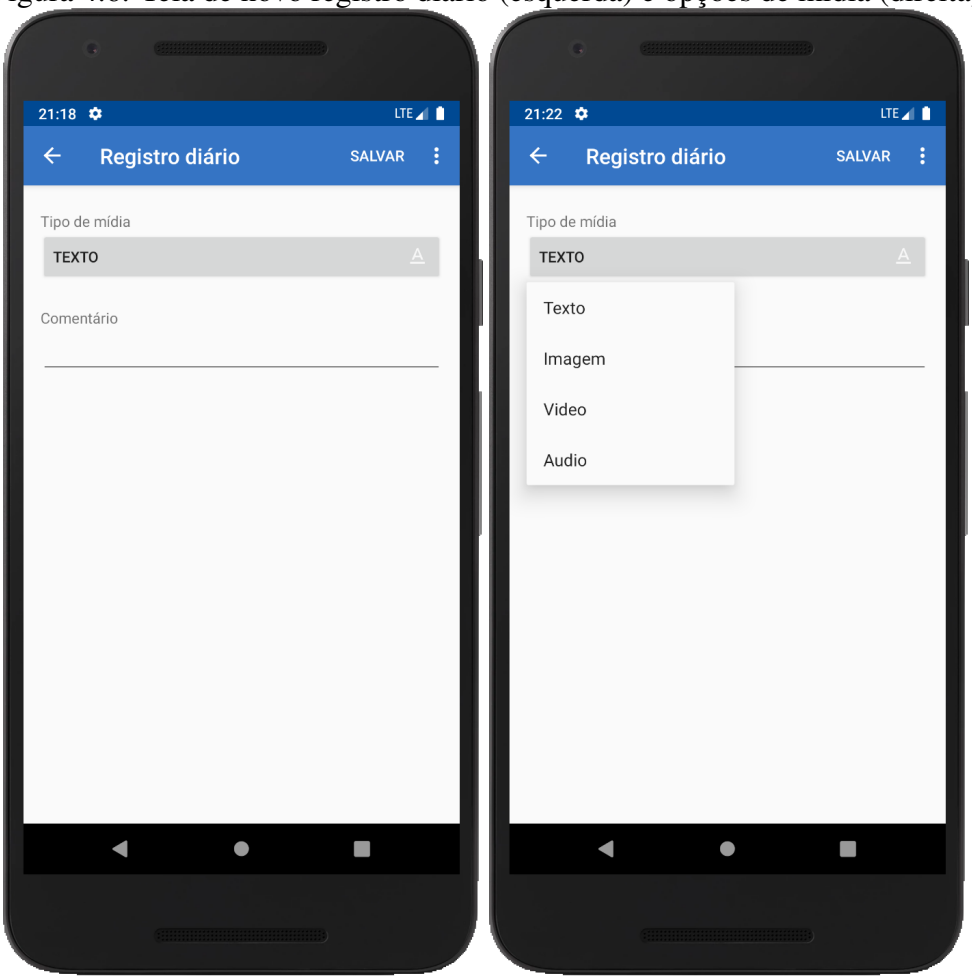


Fonte: O Autor

4.6 Cadastro de um registro diário

A tela de um novo registro diário (Figura 4.6) permite a seleção do tipo de mídia a ser incluída através de um simples botão de seleção onde o professor pode escolher os seguintes tipos: texto, áudio ou vídeo. A tela também fornece a opção do professor acrescentar um comentário em forma textual.

Figura 4.6: Tela de novo registro diário (esquerda) e opções de mídia (direita)

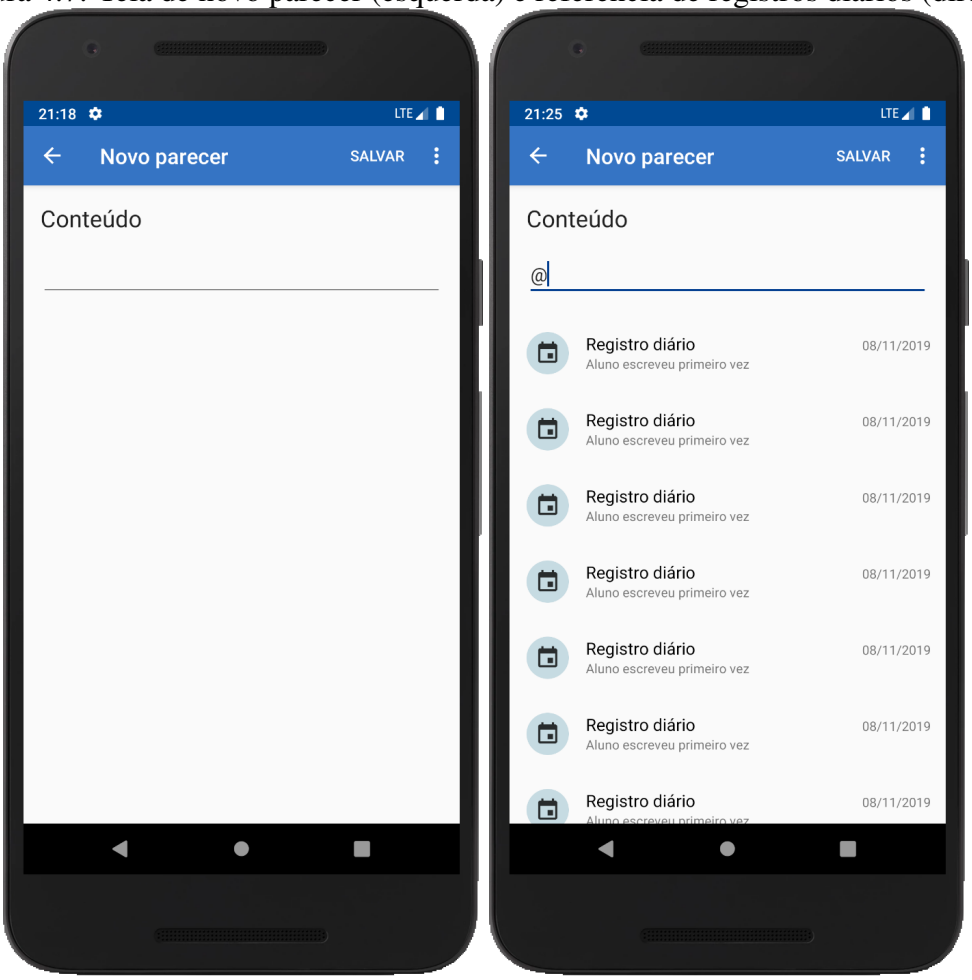


Fonte: O Autor

4.7 Cadastro de parecer pedagógico

A tela de cadastro de parecer pedagógico (Figura 4.7) possui um campo simples de texto em que o professor pode, opcionalmente, referenciar um registro diário através da inserção do caractere “@”. O campo de texto ao reconhecer o caractere “@” mostra os registros diários do aluno abaixo do campo textual.

Figura 4.7: Tela de novo parecer (esquerda) e referência de registros diários (direita)

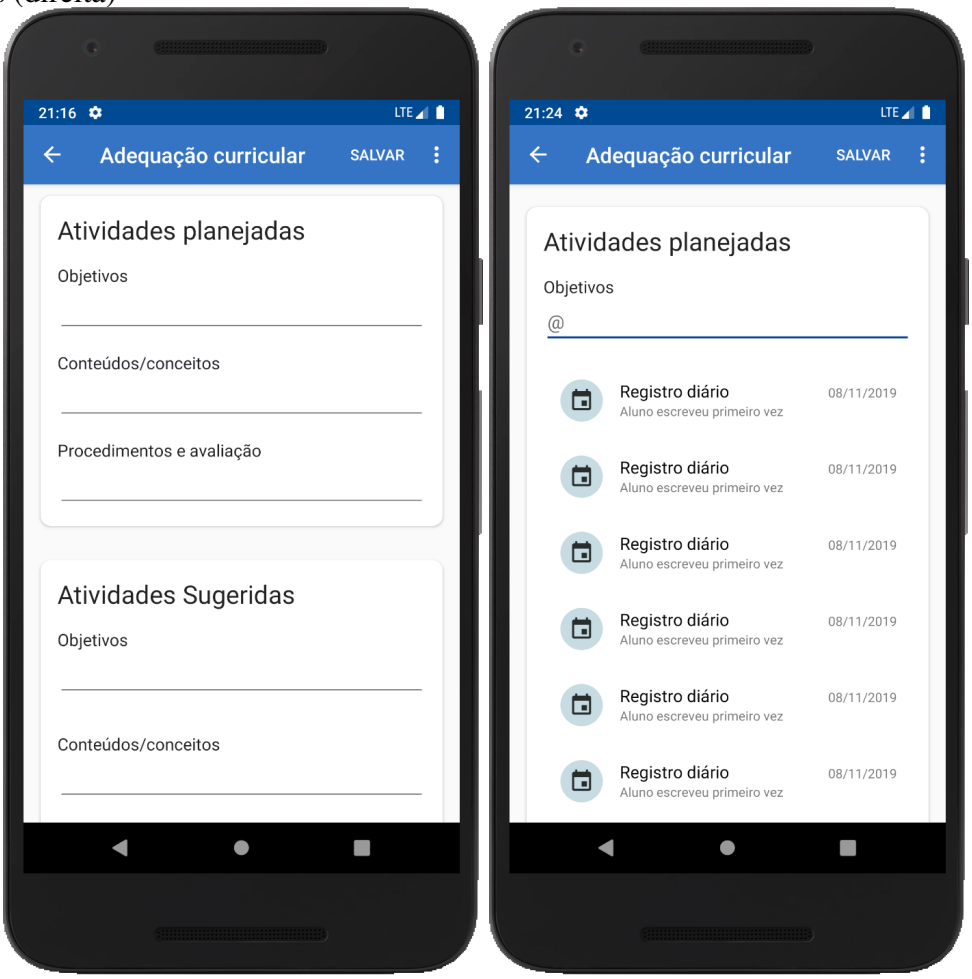


Fonte: O Autor

4.8 Cadastro de adequação curricular

A tela de cadastro de uma nova adequação curricular (Figura 4.8) possui as áreas onde o professor poderá acrescentar as atividades planejadas ou sugerir mudanças. Em todos os campos, a exemplo da tela de cadastro de um novo parecer, o professor pode relacionar os registros diários do aluno ao pressionar a tela "@".

Figura 4.8: Tela de cadastro de adequação curricular (esquerda) e referência de registros diários (direita)



Fonte: O Autor

5 INTEGRAÇÃO COM A APLICAÇÃO IOS

Durante o desenvolvimento deste trabalho, foi utilizada a mesma modelagem das entidades do banco de dados do aplicativo iOS (conforme descrito na seção 3.2) mas em uma referência distinta. Com isso, umas das etapas necessárias a realizar foi a atualização da configuração de conexão com o Firebase utilizada no desenvolvimento do aplicativo Android para a mesma referência usada pelo sistema desenvolvido para iOS.

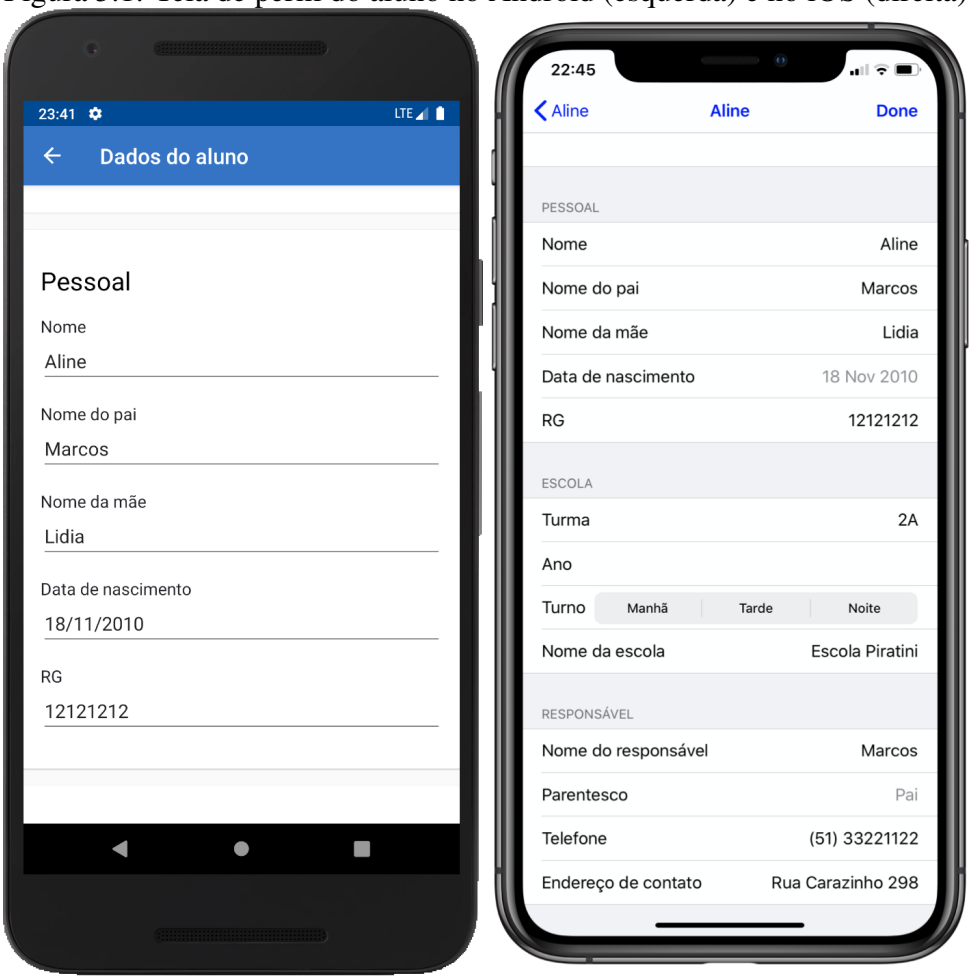
Para que a integração pudesse ser realizada com sucesso, foram mapeadas as funcionalidades implementadas por Silva (2019) na versão Android, assim, foi possível verificar que, embora em plataformas diferentes, os aplicativos compartilhavam o mesmo conjunto de dados para o mesmo professor *logado* na aplicação.

Portanto, neste capítulo será apresentada a integração com o mesmo banco de dados utilizado pela aplicação iOS desenvolvida por Silva (2019). A seguir, serão demonstradas as comparações entre as principais telas da aplicação Android e iOS. A última seção é destinada a destacar alguns detalhes da integração.

5.1 Tela de perfil do aluno

As informações cadastradas do aluno pelo professor são facilmente compartilhadas entre os aplicativos para Android e iOS. Algumas questões de integração precisam ser resolvidas: cadastro no ano escolar, turno e NEE's do aluno. Com isso, essas informações não foram salvadas no momento de criação do aluno pelo aplicativo Android pois com esses dados adicionais a aplicação no iOS parava de carregar os alunos previamente cadastrados. A figura 5.1 mostra parte dos dados salvos que referenciam o perfil de uma aluna fictícia.

Figura 5.1: Tela de perfil do aluno no Android (esquerda) e no iOS (direita)



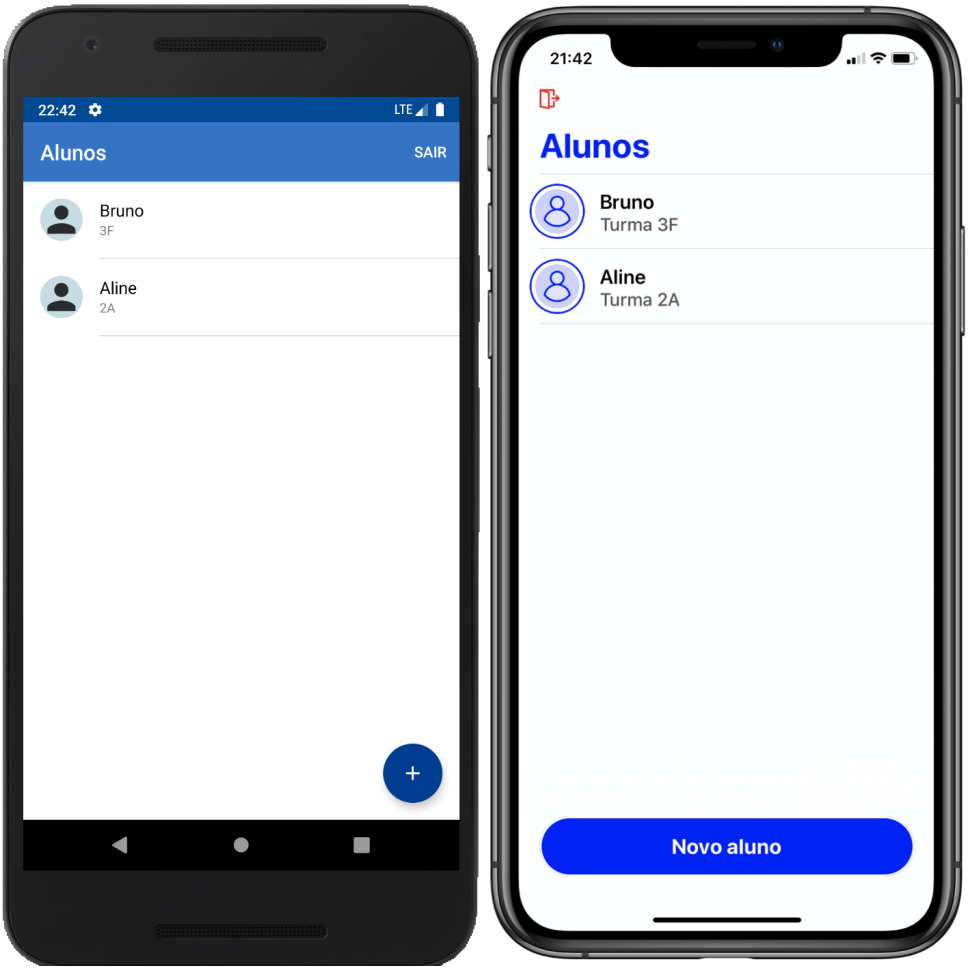
Fonte: O Autor

5.2 Tela de lista de alunos

A tela de lista de alunos mantém um padrão muito semelhante entre as aplicações desenvolvidas. Refletem os mesmos alunos cadastrados por um professor. Ambas implementações disponibilizam uma maneira do professor realizar o *logout* para terminar sua sessão no APP (botão sair - Android e ícone de uma porta - iOS). O gerenciamento de usuários foi facilitado através do uso do Firebase que dispõe de uma área para autenticação dos usuários.

A figura 5.2 mostra a comparação entre as telas de lista de alunos desenvolvidas.

Figura 5.2: Tela de lista de alunos no Android (esquerda) e no iOS (direita)



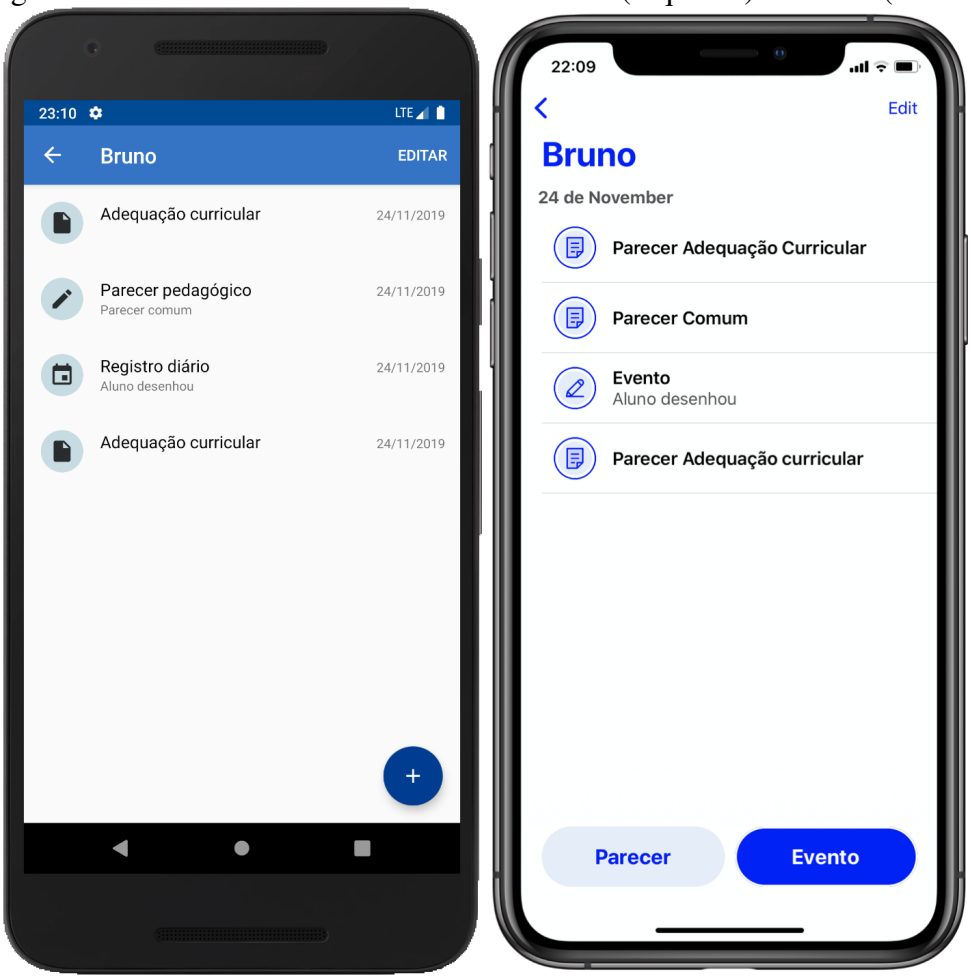
Fonte: O Autor

5.3 Tela de detalhes do aluno

A tela de detalhes do aluno contém algumas diferenças na escolha dos ícones para a representação das informações. Para a versão Android, foram alterados os títulos dos tipos de dados que o professor pode inserir. Contudo, apesar de mudanças visuais, o banco de dados é o mesmo e mantém a coerência entre as diferentes plataformas. Ambos os aplicativos dispõem das mesmas funcionalidades implementadas: novo registro diário, novo parecer e nova adequação curricular.

A figura 5.3 mostra o resultado da integração para a tela de detalhes do aluno.

Figura 5.3: Tela de detalhes do aluno no Android (esquerda) e no iOS (direita)

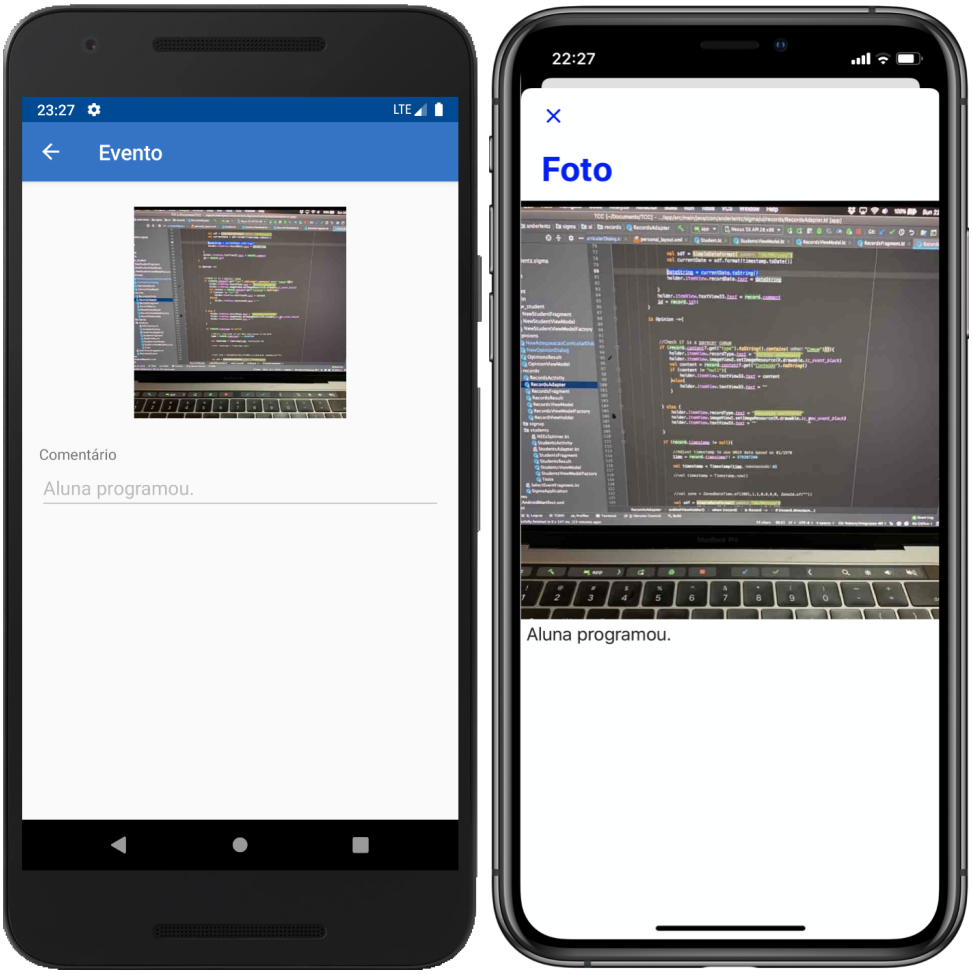


Fonte: O Autor

5.4 Tela de visualização de registros com mídia

As informações de registros diários que contêm um dos tipos de mídia permitidos podem ser visualizadas em ambas as aplicações. A saber, os tipos de mídia suportados e integrados são: imagem, áudio e vídeo. Para que as mídias fossem completamente integradas, foram utilizados arquivos compatíveis entre os aparelhos: imagem (.jpg), áudio (.mp3) e vídeo (.mp4). Esses arquivos são enviados na mesma estrutura no Firebase Cloud Storage e, caso o professor tiver permissão, acessados de forma comum entre os aplicativos. Dessa forma, ambas as plataformas, iOS e Android, possuem as mesmas funcionalidades para registro diário contendo alguma mídia. A figura 5.4 exemplifica um registro diário contendo uma foto.

Figura 5.4: Tela de visualização de registro com mídia no Android (esquerda) e no iOS (direita)



Fonte: O Autor

5.5 Detalhes da integração

Nas seções anteriores é possível notar um comparativo entre uma sequência principal da aplicação: cadastro do aluno através do preenchimento de suas informação; visualização dos alunos que um determinado professor está acompanhando, visualização da tela dos detalhes do aluno contendo os documentos (parecer, registros diários ou adequações curriculares) que o professor criou para o aluno e o exemplo de uma tela de um tipo de registro diário contendo uma imagem. Outras funcionalidades, porém, foram integradas e esta seção é destinada a destacar essas funcionalidades. A Figura 5.5 mostra o estado em que cada funcionalidade se encontra em termos da integração.

Figura 5.5: Funcionalidades integradas

Funcionalidades	Status
Login	Integrado
Logout	Integrado
Criar conta do professor	Integrado
Cadastrar aluno	Integrado
Remover aluno	Integrado
Editar perfil do aluno	Parcialmente integrado
Criar registro diário textual	Integrado
Criar registro diário com imagem	Integrado
Criar registro diário com áudio	Integrado
Criar registro diário com vídeo	Integrado
Visualizar registro diário	Integrado
Remover registro diário	Integrado
Criar parecer	Integrado
Criar adequação curricular	Integrado
Compartilhar registro diário	Não integrado

Fonte: O Autor

Conforme observado, a funcionalidade de editar o perfil do aluno está parcialmente integrada devido a uma incompatibilidade entre os campos de NEES do aluno, turno e ano escolar. Esses campos não foram salvados no momento da criação e edição de alunos, pois, ao serem incluídos no aplicativo Android, todos os alunos previamente cadastrados na aplicação iOS paravam de aparecer. Contatando o autor da aplicação iOS, um dos motivos que pode ser a causa deste problema é a falta de compatibilidade entre os campos no documento do aluno no Firebase. Assim, é possível que o sistema iOS não consiga fazer o *parse* corretamente do JSON vindo do Firebase. A funcionalidade de compartilhamento de um registro diário não foi implementada na aplicação Android, portanto, só está disponível para os usuários do aplicativo iOS, onde o professor pode gerar um link que aponta para o registro diário cadastrado no banco de dados.

Um detalhe importante que foi necessário ser implementado para a correta integração e que afeta grande parte das funcionalidades é a adequação do *timestamp*. Um problema encontrado foi que o *timestamp* gerado pelo iOS no momento da inclusão de registros diários, pareceres e adequações curriculares tem como base a data de 01/01/2001. Todavia, no Android, a data base para a geração do *timestamp* é 01/01/1970. Para que os registros criados por um professor fossem mostrados de forma ordenada pela data, foi necessário fazer com que o mesmo *timestamp* fosse utilizado em ambas as aplicações. Para isso, optou-se por modificar o *timestamp* no APP Android, assim, é subtraído o total de 978307200 segundos (diferença entre as datas bases 01/2001 e 01/2970) do *timestamp*

gerado pelo Android fazendo com que ambos os aplicativos tenham a mesma data base.

6 TESTE DE USABILIDADE

Este capítulo apresentará os resultados obtidos através da aplicação do teste SUS com cinco participantes, não especialistas, após uma série de tarefas realizadas com o objetivo de avaliar a usabilidade do aplicativo. No total, 3 mulheres e 2 homens participaram dos testes. As idades dos participantes se encontram na seguinte faixa: 25-30 anos, 3 participantes; 50-55 anos, 2 participantes.

Os resultados obtidos focam a usabilidade da aplicação e, para isso, foi elaborado um conjunto de cinco tarefas explicitadas em um formulário no Google Forms (Apêndice A) contendo as instruções para a execução do teste proposto. Além do teste realizado com os cinco participantes, a mesma avaliação foi realizada com uma professora especialista da área e os seus resultados são demonstrados na seção 6.3.

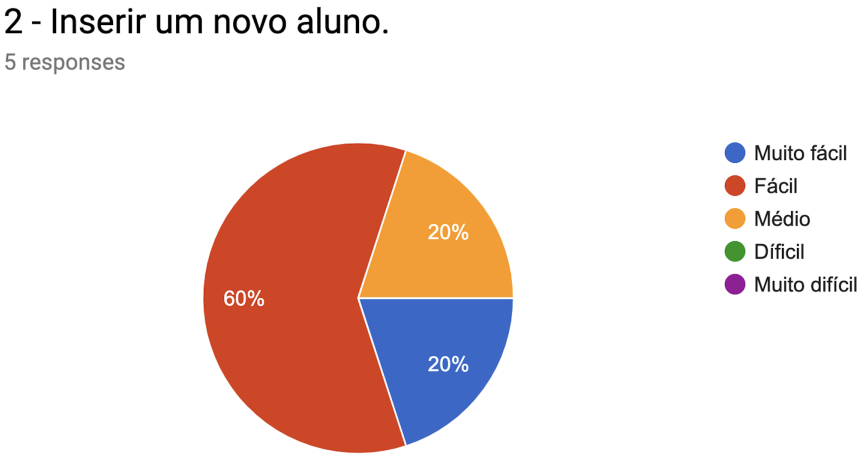
6.1 Realização das tarefas

A presente seção mostra os resultados das tarefas realizadas pelos participantes do teste. Foi testado um fluxo principal da aplicação: login, cadastro de um novo aluno, inclusão de um registro diário do tipo imagem, exclusão do registro diário e *logout* da aplicação. Para cada tarefa foi perguntado o nível de dificuldade para a execução e, a seguir, as figuras mostram os resultados obtidos.

Figura 6.1: Tarefa 1

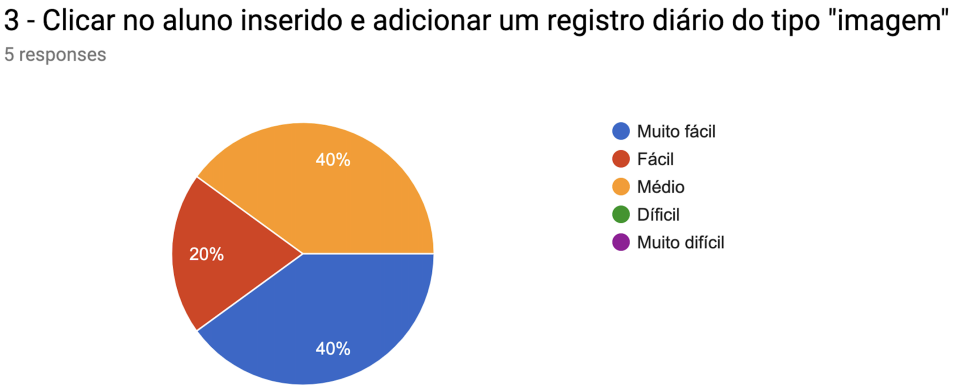
Fonte: O Autor

Figura 6.2: Tarefa 2



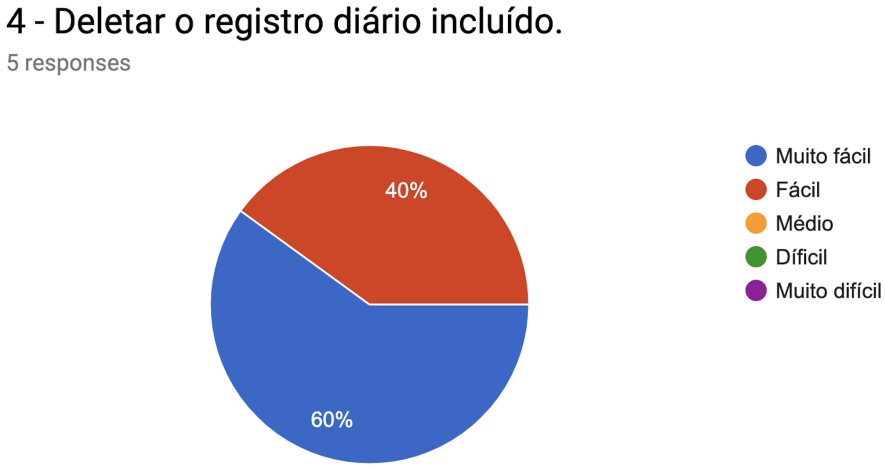
Fonte: O Autor

Figura 6.3: Tarefa 3



Fonte: O Autor

Figura 6.4: Tarefa 4



Fonte: O Autor

Figura 6.5: Tarefa 5

Fonte: O Autor

De uma maneira geral, os dados demonstram que os participantes não tiveram dificuldades excessivas nas tarefas propostas. A tarefa 2 e a tarefa 3 foram as que resultaram em um maior nível de dificuldade na realização. Um fator que pode ter contribuído é a falta de experiência com o método de inserção de dados através do botão FAB no canto inferior direito da tela ou a dificuldade de relacionar o mesmo botão com as ações que o usuário pode fazer em cada tela.

6.2 System Usability Scale

O *System Usability Scale* desenvolvido por Brooke (1996) apresenta uma maneira de verificar a usabilidade de um determinado sistema e consiste de uma série de dez perguntas onde os participantes podem indicar o grau de discordância ou concordância em cada questão.

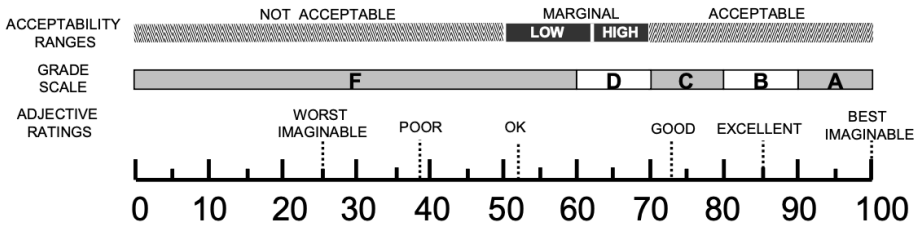
Dado as respostas das pessoas relacionadas nos testes, é realizado um cálculo para obter a pontuação final seguindo as seguintes regras:

- **para respostas ímpares**: é subtraído 1 da pontuação informada pelo participante.
- **para as respostas pares**: é subtraído ao número 5 a resposta informada pelo usuário. Por exemplo, usuário respondeu com 1. Então, o resultado é: 5 - 1 = 4.

Para o cálculo do *score* final para um participante, é realizada a soma de todas as dez respostas e multiplicado por 2,5. Desta forma, foram obtidas cinco pontuações, uma para cada participante, e foi realizada a média dos valores obtidos. De acordo com Bangor

Philip Kortum (2009), através dos estudos realizados obteve-se uma média próxima ao valor 70 e qualquer pontuação abaixo disso corresponde em uma solução com problemas de usabilidade. A escala completa é mostrada na figura a seguir:

Figura 6.6: SUS Score



Fonte: (BANGOR PHILIP KORTUM, 2009)

O aplicativo desenvolvido obteve uma média de 85,5 pontos dado o cálculo realizado com as respostas dos cinco participantes. Isto corresponde a um conceito B (excelente) quando comparado à escala da Figura 6.6. Os dados completos da pesquisa bem como o as médias para cada participante se encontram no Apêndice B.

6.3 Opinião da especialista

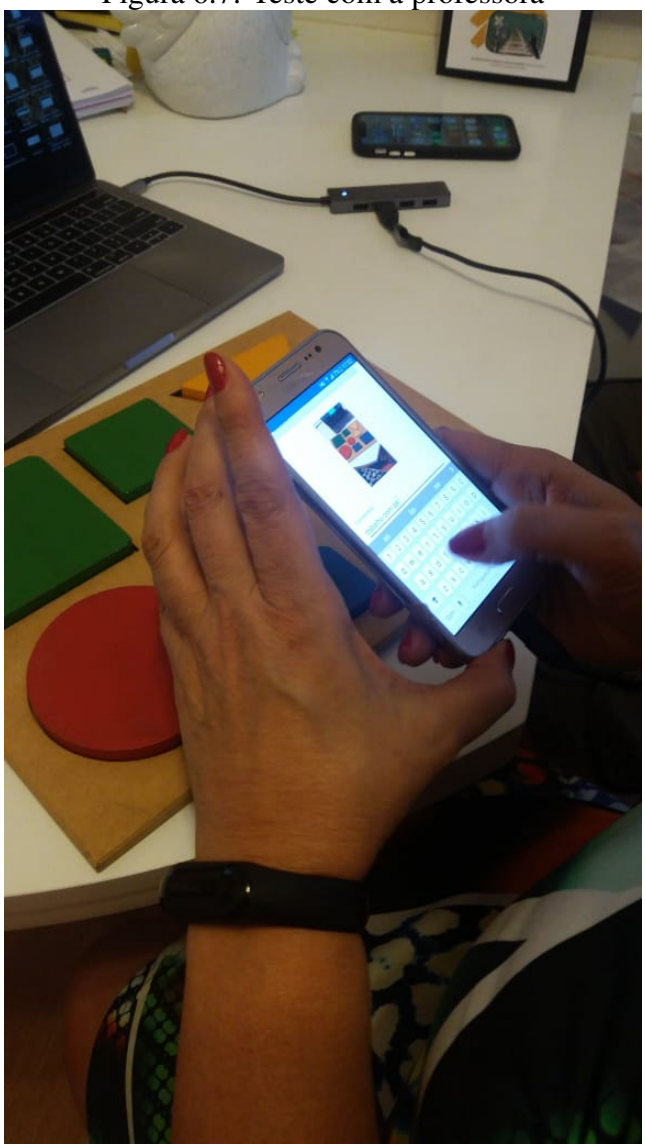
O mesmo teste SUS aplicado com usuários não especialistas na área foi utilizado com a professora Andreia Pereira Gonçalves fez parte do grupo de profissionais que iniciaram as Salas de Integração e Recursos - SIR na rede municipal de ensino de Porto Alegre. Ela teve participação direta na implementação e desenvolvimento dos processos de atendimento e intervenção neste serviço. É graduada em Pedagogia - Educação Especial, atualmente exerce suas funções como psicopedagoga e assessoria à redes públicas e privadas.

Mesmo que o teste aplicado com a professora especialista não tenha caráter quantitativo, foi possível observar, através do questionário e *in loco*, que o aplicativo foi bem recebido e avaliado. Durante os testes, diversas vezes ela demonstrou, com entusiasmo, a vontade de utilizar a solução desenvolvida em seu trabalho, inclusive externados na forma de comentários como: "já quero usar o aplicativo". A imagem na Figura 6.7 mostra a professora manuseando a aplicação em um dispositivo Android.

Sobre a usabilidade do aplicativo, a professora não demonstrou dificuldades na utilização, realizou as tarefas selecionadas com facilidade e atingiu pontuação máxima (100) no teste SUS. Ao contrário dos usuários comuns, ela concedeu nota máxima (Con-

cordo totalmente) ao item do teste SUS que pergunta aos usuários se eles desejam utilizar a aplicação com frequência. Esse resultado somado ao *feedback*, qualitativamente, denota que a aplicação tem a capacidade de fornecer uma solução útil ao trabalho do professor.

Figura 6.7: Teste com a professora



Fonte: O Autor

7 CONCLUSÃO

A realização deste trabalho oportunizou um aprendizado imenso sobre as mais recentes tecnologias para o desenvolvimento de aplicativos para Android. A utilização dos *Android Architecture Components* ajudou a criar uma aplicação modular e em compasso com o que o Google espera dos desenvolvedores. O processo de desenvolvimento através do uso das histórias de usuário e do quadro de tarefas oportunizaram a constante avaliação do andamento das tarefas a serem realizadas e resultaram em um fator decisivo para o sucesso deste projeto.

Algumas questões ficam pendentes na esperança de trabalhos futuros darem continuidade em direção a uma solução viável. Portanto, é destacado alguns pontos a serem trabalhados em futuras versões: a melhoria do relacionamento dos registros diários em um parecer ou em uma adequação curricular; inclusão de um método de pesquisa para os dados em forma de lista, a atualização em tempo real quando um novo dado é inserido em um aparelho diferente e algumas questões relacionadas a integração com o banco de dados.

Contudo, o desenvolvimento do presente aplicativo contribui para a organização e coleta de dados de maneira rápida e eficiente por parte do professor. Com a versão disponibilizada para a plataforma Android, quase 100% das pessoas que possuem algum tipo de smartphone serão beneficiadas. Embora seja uma versão embrionária, a aplicação tem muito a contribuir e evoluir como ferramenta auxiliadora e não como uma solução substitutiva do trabalho dos professores.

REFERÊNCIAS

ANDROID. **Platform Architecture**. 2019. Acessado em: 16/07/2019. Available from Internet: <https://developer.android.com/guide/platform>.

BANGOR PHILIP KORTUM, J. M. A. Determining what individual sus scores mean: Adding an adjective rating scale. In: **Journal of usability studies**. [S.l.]: Usability Professionals' Association, 2009. p. 114–123.

BAYLISS, D. **Android apprentice**. United States: Razeware LLC, 2018. ISBN 1942878494.

BROOKE, J. Sus - a quick and dirty usability scale. In: **Usability evaluation in industry**. [S.l.]: Taylor and Francis, 1996. p. 189–194.

COHN, M. **Advantages of the "As a user, I want" user story template**. 2008. Acessado: 10/11/2019. Available from Internet: <https://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template>.

COSTA, W. **Começando com Android Data Binding**. 2018. Acessado: 14/10/2019. Available from Internet: <https://medium.com/android-dev-br/começando-com-android-data-binding-d7719333eecc>.

DESIGN, M. **Material Design**. 2019. Acessado: 06/11/2019. Available from Internet: <https://material.io/components/lists/#specs>.

DEVELOPERS, A. **Distribution dashboard**. 2019. Acessado: 09/11/2019. Available from Internet: <https://developer.android.com/about/dashboards>.

DEVELOPERS, A. **Fragmentos**. 2019. Acessado: 16/11/2019. Available from Internet: <https://developer.android.com/guide/components/fragments?hl=pt-br>.

DEVELOPERS, A. **Navigation**. 2019. Acessado: 09/11/2019. Available from Internet: <https://developer.android.com/guide/navigation>.

DEVELOPERS, G. **LiveData Overview**. 2019. Acessado em: 15/07/2019. Available from Internet: <https://developer.android.com/topic/libraries/architecture/livedata>.

DOOLEY, J. F. **Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring, 2nd Edition**. 2. ed. [S.l.]: Apress, 2017. ISBN 9781484231524, 9781484231531.

ESPECIAL, S. de E. **Diretrizes Operacionais para o atendimento educacional especializado na Educação Básica, modalidade Educação Especial**. 2009. Acessado: 08/11/2019. Available from Internet: <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=428-diretrizes-publicacao&Itemid=30192>.

FIRESTORE, G. **Firestore**. 2019. Acessado: 09/11/2019. Available from Internet: <https://firebase.google.com/products/firestore/>.

FOWLER, A. **NoSQL For Dummies**. 1. aufl.. ed. New York: John Wiley Sons, 2015. ISBN 978-1-118-90562-3.

FREEMAN, E. **Head First design patterns**. Sebastopol, CA: O'Reilly, 2004. ISBN 9780596007126.

GAMMA, E. **Design patterns : elements of reusable object-oriented software**. Reading, Mass: Addison-Wesley, 1995. ISBN 978-0201633610.

GOOGLE. **Activity Lifecycle**. 2019. Available from Internet: <https://developer.android.com/guide/components/activities/activity-lifecycle>.

GOOGLE. **Android Architecture Components**. 2019. Acessado: 11/10/2019. Available from Internet: <https://developer.android.com/topic/libraries/architecture/index.html>.

GOOGLE. **Atividades**. 2019. Available from Internet: <https://developer.android.com/guide/components/activities/?hl=pt-br>.

GOOGLE. **The Color System**. 2019. Acessado: 14/07/2019. Available from Internet: <https://material.io/design/color/the-color-system.html#color-usage-palettes>.

GOOGLE. **Color Tool**. 2019. Acessado: 14/07/2019. Available from Internet: <https://material.io/tools/color>.

GOOGLE. **Data Binding Library**. 2019. Acessado: 14/10/2019. Available from Internet: <https://developer.android.com/topic/libraries/data-binding>.

GOOGLE. **onResume**. 2019. Acessado: 10/10/2019. Available from Internet: <https://developer.android.com/reference/android/app/Activity.html#onResume()>.

GOOGLE. **ViewModel**. 2019. Acessado: 14/10/2019. Available from Internet: <https://developer.android.com/topic/libraries/architecture/viewmodel>.

GRIFFITHS, D.; GRIFFITHS, D. **Head First Android Development: A Brain-Friendly Guide**. [S.l.]: O'Reilly Media, 2017. ISBN 9781491974056.

IDC. **Mobile Operating System Market Share Brazil**. 2019. Acessado em: 13/07/2019. Available from Internet: <https://www.idc.com/promo/smartphone-market-share/os>.

JACKSON, W. **Android apps for absolute beginners**. Berkeley, Calif: Apress, 2012. ISBN 978-1430247883.

KOTLIN. **Using Kotlin for Android Development**. 2019. Acessado: 25/11/2019. Available from Internet: <https://kotlinlang.org/docs/reference/android-overview.html>.

LEE. **Beginning Android 4 application development**. Indianapolis, Ind: Wrox/John Wiley & Sons, 2012. ISBN 978-1118240670.

MEIER, A.; KAUFMANN, M.; KAUFMANN, M. **SQL and NoSQL Databases - Models, Languages, Consistency Options and Architectures for Big Data Management**. 1st ed. 2019. ed. Berlin, Heidelberg: Springer, 2019. ISBN 978-3-658-24549-8.

MEIER, R.; LAKE, I. **Professional Android**. [S.l.]: Wrox, 2018. ISBN 9781118949528.

MURPHY, M. L. **Android Architecture Components**. 2018. Available from Internet: <https://commonsware.com/AndroidArch/>.

POINT, T. **Android Architecture**. 2019. Acessado em: 16/07/2019. Available from Internet: <https://www.tutorialspoint.com/android/android_architecture>.

SILVA, A. R. da C. **SIGMA : Sistema de gestão e acompanhamento móvel de alunos portadores de necessidades educacionais especiais**. Universidade Federal do Rio Grande do Sul, 2019. Trabalho de Conclusão do Curso de Ciência da Computação. Available from Internet: <http://hdl.handle.net/10183/198601>.

SILVEIRA, F. **Activity – o que é isso?** 2010. Acessado: 10/10/2019. Available from Internet: <http://www.felipesilveira.com.br/2010/05/activity-o-que-e-isso/>.

SMITH, J. **Patterns - WPF Apps With The Model-View-ViewModel Design Pattern**. 2009. Available from Internet: <https://msdn.microsoft.com/en-us/magazine/dd419663.aspx?irgwc=1&OCID=AID2000142_aff_7593_1243925&tduid=(ir__e11iyqntmskfr20akk0sohzj0m2xgjzsnfcp12au00)(7593)(1243925)(TnL5HPStwNw-Ywchy4BqjEWPFU7uZKkGog)()&irclickid=_e11iyqntmskfr20akk0sohzj0m2xgjzsnfcp12au00#id0090009?ranMID=24542&ranEAID=TnL5HPStwNw&ranSiteID=TnL5HPStwNw-Ywchy4BqjEWPFU7uZKkGog&epi=TnL5HPStwNw-Ywchy4BqjEWPFU7uZKkGog,note={Acessado:15/10/2019}.>

SPäTH, P. **Pro Android with Kotlin: Developing Modern Mobile Apps**. [S.l.]: Apress, 2018. ISBN 978-1-4842-3820-2.

STATS, G. **Mobile Operating System Market Share Brazil**. 2019. Available from Internet: <http://gs.statcounter.com/os-market-share/mobile/brazil/#monthly-201806-201906>.

TEAM raywenderlich T. **Advanced Android App Architecture (First Edition): Real-world app architecture in Kotlin 1.3**. Razeware LLC, 2019. ISBN 1942878699. Available from Internet: <https://www.xarg.org/ref/a/1942878699/>.

THIENGO, V. **Ciclo de Vida de Uma Atividade no Android**. 2019. Acessado em: 09/10/2019. Available from Internet: <https://www.thiengo.com.br/ciclo-de-vida-de-uma-atividade-no-android>.

APÊNDICE A — QUESTIONÁRIO DE AVALIAÇÃO

A.1 Instruções para a realização dos testes

Questionário de usabilidade

Olá, você está sendo convidado(a) a participar do teste de usabilidade do aplicativo. Sua colaboração é muito importante para o fechamento do meu trabalho de conclusão.

Após instalar o aplicativo, você deverá executar algumas operações básicas, descritas abaixo:

1 - Fazer o cadastro usando um email e senha.
2 - Inserir um novo aluno.
3 - Selecionar aluno inserido e adicionar um registro diário do tipo "imagem".
4 - Deletar o registro diário incluído (arrastar a célula para a esquerda).
5 - Navegar até a tela de alunos e sair da aplicação.

Após realizar as tarefas descritas, siga para a próxima seção.

Muito obrigado pela sua participação,
Anderson Lentz da Silva

NEXT

Never submit passwords through Google Forms.

A.2 Nível de dificuldade de utilização do aplicativo

Nível de dificuldade de utilização do aplicativo

Com base nas tarefas que você realizou, responda de acordo com o nível de dificuldade encontrado em cada operação.

1 - Fazer o cadastro usando um email e senha. *

- ○ Muito fácil
- ○ Fácil
- ○ Médio
- ◉ Díficil
- ○ Muito difícil

2 - Inserir um novo aluno. *

- ○ Muito fácil
- ○ Fácil
- ○ Médio
- ○ Díficil
- ○ Muito difícil

3 - Clicar no aluno inserido e adicionar um registro diário do tipo "imagem" *

- ○ Muito fácil
- ○ Fácil
- ○ Médio
- ○ Díficil
- ○ Muito difícil

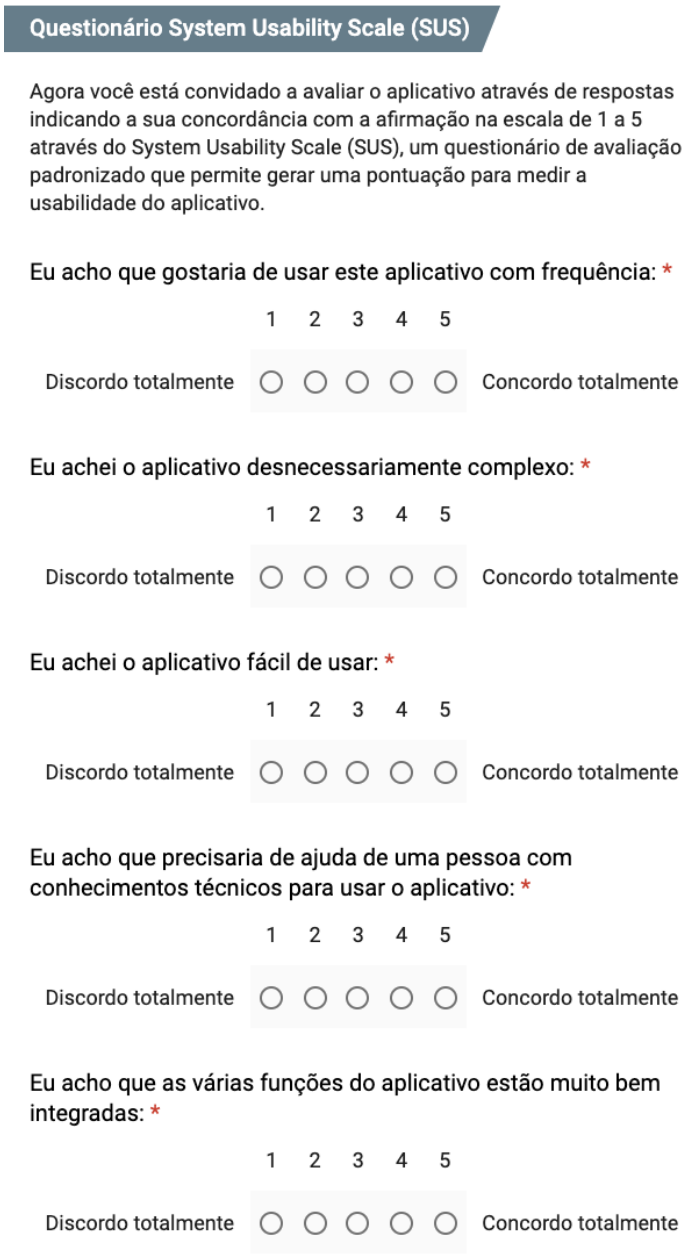
4 - Deletar o registro diário incluído (arrastar a célula para a esquerda até o final). *

- ○ Muito fácil
- ○ Fácil
- ○ Médio
- ○ Díficil
- ○ Muito difícil

5 - Navegar até a tela de alunos e sair da aplicação. *

- ○ Muito fácil
- ○ Fácil
- ○ Médio
- ○ Díficil
- ○ Muito difícil

A.3 Questionário System Usability Scale



Questionário System Usability Scale (SUS)

Agora você está convidado a avaliar o aplicativo através de respostas indicando a sua concordância com a afirmação na escala de 1 a 5 através do System Usability Scale (SUS), um questionário de avaliação padronizado que permite gerar uma pontuação para medir a usabilidade do aplicativo.

Eu acho que gostaria de usar este aplicativo com frequência: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu achei o aplicativo desnecessariamente complexo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu achei o aplicativo fácil de usar: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o aplicativo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu acho que as várias funções do aplicativo estão muito bem integradas: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu acho que o aplicativo apresenta muita inconsistência: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu imagino que as pessoas aprenderão como usar esse aplicativo rapidamente: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu achei o sistema atrapalhado de usar: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu me senti bem confiante ao usar o aplicativo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

Eu precisei aprender muitas coisas para conseguir utilizar este aplicativo: *

	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	Concordo totalmente

APÊNDICE B — RESULTADOS DA APLICAÇÃO DO SUS

B.1 Pontuação do teste SUS com usuários comuns

Participant	Statement 1	Statement 2	Statement 3	Statement 4	Statement 5	Statement 6	Statement 7	Statement 8	Statement 9	Statement 10	SU Score
Participant 1	3	1	5	2	5	1	5	1	5	1	**92,5**
Participant 2	4	1	5	1	4	1	5	1	5	1	**95,0**
Participant 3	2	1	2	2	2	2	3	3	2	1	**55,0**
Participant 4	3	1	4	1	4	1	5	1	5	1	**90,0**
Participant 5	4	1	5	1	5	1	4	1	5	1	**95,0**
										AVERAGE	**85,5**

B.2 Pontuação do teste SUS a professora especialista

Participant	Statement 1	Statement 2	Statement 3	Statement 4	Statement 5	Statement 6	Statement 7	Statement 8	Statement 9	Statement 10	SU Score
Andreia	5	1	5	1	5	1	5	1	5	1	100,0