CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

ANDRE CESAR LINO

PROTÓTIPO DE APLICATIVO MÓVEL PARA ACOMPANHAMENTO ESCOLAR DO COLÉGIO UNIDAVI: MÓDULO PARA OS PAIS OU RESPONSÁVEIS

RIO DO SUL
2021

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

ANDRE CESAR LINO

PROTÓTIPO DE APLICATIVO MÓVEL PARA ACOMPANHAMENTO ESCOLAR DO COLÉGIO UNIDAVI: MÓDULO PARA OS PAIS OU RESPONSÁVEIS

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: M.e Marcondes Maçaneiro

RIO DO SUL
2021

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

ANDRE CESAR LINO

PROTOTIPO DE APLICATIVO MÓVEL PARA ACOMPANHAMENTO ESCOLAR DO COLÉGIO UNIDAVI: MÓDULO PARA OS PAIS OU RESPONSÁVEIS

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

_____

Professor Orientador: M.e Marcondes Maçaneiro

Banca Examinadora:

_____

Prof. M.e Fernando Andrade Bastos

_____

Prof. M.e Jeancarlo Visentainer

Rio do Sul, 29 de Dezembro de 2021.

RESUMO

Nos dias atuais, a maioria das pessoas possuem ao menos um dispositivo móvel no qual pode ser utilizado em inúmeras situações, tanto para o lazer como para atividades importantes como pagamento de contas, acesso a e-mails ou mesmo acompanhar a vida acadêmica dos filhos. Baseando-se nisso, o objetivo desse trabalho foi desenvolver um protótipo de um aplicativo para smartphones utilizando React Native onde é possível desenvolver apenas um código fonte para os dois sistemas operacionais mais utilizados IOS e Android. Desta forma torna possível que aos pais de alunos do Colégio Unidavi acompanhem com maior praticidade a vida acadêmica de seus filhos. Esse acompanhamento é feito através de notas e faltas, ocorrências do filho e financeiro. Hoje essas funções não possuem as informações de forma prática e clara no sistema acadêmico utilizado, tornando a experiência de uso muito difícil. Desta forma, utilizando o protótipo, os responsáveis conseguiriam usar com facilidade as funções do sistema do Colégio Unidavi, e ter um contato muito mais próximo com a instituição.

Palavras-Chave: smartphone, protótipo, notas e faltas, financeiro, ocorrências

ABSTRACT

Nowadays, most people have at least one mobile device on which they can be used at home, both for leisure and for important activities such as paying bills, accessing e-mails or even following up on their children's academic life. Based on that, the objective of this work was to develop a prototype of an application for smartphones using React Native where it is possible to develop only one source code for the two most used operating systems, IOS and Android. In this way, it is possible for the parents of students at Colégio Unidavi to monitor their children's academic life more practically. This monitoring is done through grades and absences, occurrences of the child and financial. Today these functions do not have the information in a practical and clear way in the academic system used, making the experience of use very difficult. Thus, using the prototype, those responsible would be able to easily use the functions of the Colégio Unidavi system, and have a much closer contact with the institution.

Keywords: smartphone, prototype, notes and absences, financial, occurrences

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AOT	Ahead Of Time
API	Interface de programação de aplicações
APK	Android Package
AVD	Android Virtual device
CETIC	Centro regional de estudos para o desenvolvimento da sociedade da informação
DTO	Objeto de Transferência de Dados
ES7	ECMAScript 7
GPS	Global positioning system
IDC	International Data Corporation
JIT	Just In Time
OHA	Open Handset Alliance
SDK	Kit de desenvolvimento de software
TIC	Tecnologias de informação e comunicação
UX	User experience
XML	Extensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

O uso de aplicativos para dispositivos móveis está se tornando aos poucos uma ferramenta indispensável no cotidiano das pessoas. Seu uso varia desde a utilização para transações bancárias até interações entre pessoas em redes sociais. Atividades que antes eram feitas por meio de computadores e telefones fixos, hoje podem ser executadas com alguns toques na tela do celular.

De acordo com a FGV (2020) o Brasil dispõe de 234 milhões de celulares em funcionamento para 212 milhões de habitantes em 2020, ou seja, há mais celulares que habitantes no país.

Com a tecnologia cada dia ficando mais prática e estando nos nossos bolsos, é necessário a atualização das formas de ensino para o acompanhamento escolar, que é de suma importância para pais de alunos. Como visto, ao passo que a tecnologia avança, as instituições de ensino precisam melhorar a forma de como é apresentado a informação e comunicação com os alunos e responsáveis. Diante disto é necessário que seja considerado a utilização de aplicativos para esta gestão.

Como explica CETIC (2016, p.10), com a disseminação acelerada das tecnologias de informações e comunicação (TIC) de tecnologia nas últimas décadas, trouxe desafios e oportunidades a nível de sociedade. Essas mudanças afetam em particular a educação, que precisa lidar com as metodologias para incorporar e promover as TIC como um novo instrumento na educação.

Pensando nesses aspectos tecnológicos desenvolveu-se um aplicativo para o acompanhamento das atividades escolares dos alunos do Colégio Unidavi no qual o intuito foi tornar mais acessível a forma como as informações são dispostas para os pais e responsáveis.

Neste aplicativo terá as informações já fornecidas no sistema acadêmico que possui limitações quando é colocado em dispositivos mobiles, essas informações serão de cunho financeiro, boleto e informações de histórico, informações referentes as aulas, ocorrências do aluno, notas e faltas.

1.1 PROBLEMA DE PESQUISA

Como a criação de um protótipo de aplicativo para gestão acadêmica para os responsáveis dos alunos do Colégio da Unidavi poderia melhorar seu uso com as ferramentas do Mentor?

1.2 OBJETIVOS

1.2.1 Geral

- Desenvolver um protótipo de aplicativo para gerenciamento das principais informações e consultas para os responsáveis dos alunos do Colégio Unidavi.

1.2.2 Específicos

- Implementar interface para exibição dos dados do desempenho escolar dos alunos;
- Criar acesso ao financeiro e boleto;
- Realizar avaliação de usabilidade do protótipo com um grupo de responsáveis convidados.

1.3 JUSTIFICATIVA

Tendo em vista a necessidade de uma interação maior e mais facilitada dos pais em relação ao desenvolvimento acadêmico dos filhos, bem como a situação financeira, ponderou-se a utilização de uma versão em aplicativo mobile.

A motivação partiu de uma análise sobre como os pais que têm seus filhos na Instituição utilizam aplicativos hoje em dia. No Brasil atualmente, temos mais celulares que pessoas, e o mercado mobile é um dos mercados que mais cresce.

Sendo assim uma instituição do tamanho e proporção que é a Unidavi, necessita que os pais das crianças que estudam no Colégio Unidavi sejam mais inseridos neste meio tecnológico.

O protótipo de aplicativo foi elaborado para essa finalidade, estar mais próximo aos pais, professores e administrativo para que eles tenham mais praticidade, onde no sistema atual não é viável.

2. REFERENCIAL TEÓRICO

A literatura abordada nesta sessão trata de uma breve introdução sobre a história dos celulares. Tanto dos celulares com o sistema Android quanto do iOS e a linguagem de programação React Native utilizada nesse trabalho de conclusão, e uma análise breve e teórica sobre banco de dados.

2.1 O ANDROID

Android é um sistema operacional que foi concebido para ser aplicado em dispositivos móveis, e assim a Google criou parcerias com diversas empresas do ramo de celulares. No começo do desenvolvimento para dispositivos celulares, cada fabricante possuía seu próprio sistema operacional, e cada qual tinha que desenvolver seus próprios programas e aplicativos, define Monteiro (2012).

Segundo Monteiro (2012), com o passar dos anos empresas se reuniram para o desenvolvimento de um sistema operacional padrão para todas as empresas de tecnologia, e assim lançando um kit de desenvolvimento (SDK), uma plataforma de criação única e o desenvolvimento de uma loja para a distribuição dos aplicativos desenvolvidos, abrindo espaço para qualquer desenvolvedor se assim desejar criar aplicativos e desenvolver para o novo sistema.

Como trata Lecheta (2013, p.22):

> O Android é a resposta da Google para ocupar esse espaço. Consiste em uma nova plataforma de desenvolvimento para aplicativos móveis baseada em um sistema operacional Linux, com diversas aplicações já instaladas e, ainda, um ambiente de desenvolvimento bastante poderoso, ousado e flexível.

O Android teve um grande impacto global quando ele foi anunciado, sua prerrogativa era chamar a atenção das pessoas para um sistema operacional para competir com o mercado atual. E isso apenas aconteceu pelo fato de uma gigante por trás do desenvolvimento que era a Google, onde na época já tinha revolucionado a internet. Mas não era apenas ela que estava por trás, na época existia um grupo de líderes do mercado de telefonia, como a Motorola, LG, Samsung, Sony Ericsson e outras. Este grupo era chamado de *Open Handset Alliance* (OHA), com a iniciativa de padronizar a plataforma móvel usando código aberto e livre, para atender as expetativas e tendências do mercado da época, como explica Lecheta (2013).

A plataforma Android desfruta hoje de um papel de destaque no mercado, tanto pela quantidade significativa de dispositivos produzidos como também por oferecer uma API rica. Disponibilizando fácil acesso a vários recursos de hardware, tais como Wi-Fi e GPS, além de boas ferramentas para o desenvolvedor, cita Monteiro (2012).

2.1.1 Arquitetura do Android

O sistema Android foi concebido para dispositivos móveis. Uma plataforma que possui um sistema operacional, possui um conjunto de ferramentas que incluem contatos, ferramentas de geolocalização, telefone e acesso internet, além disso ele fornece um conjunto de ferramentas APIs utilizando a linguagem Java, como explica Monteiro (2012).

Baseado no Linux, o sistema operacional Android teve seu desenvolvimento iniciado em 2003 pela empresa Android Inc. Em 2005, a empresa foi adquirida pelo Google, que atualmente lidera o desenvolvimento do Android, como afirma Monteiro (2012).

O Android desde sua concepção foi desenvolvido utilizando o *kernel* do Linux, ou seja, utiliza código aberto e o sistema utiliza a licença *Apache* 2.0, que dá direito a qualquer pessoa a acessar o código-fonte e verificar e buscar falhas de sistema, esclarece Monteiro (2012).

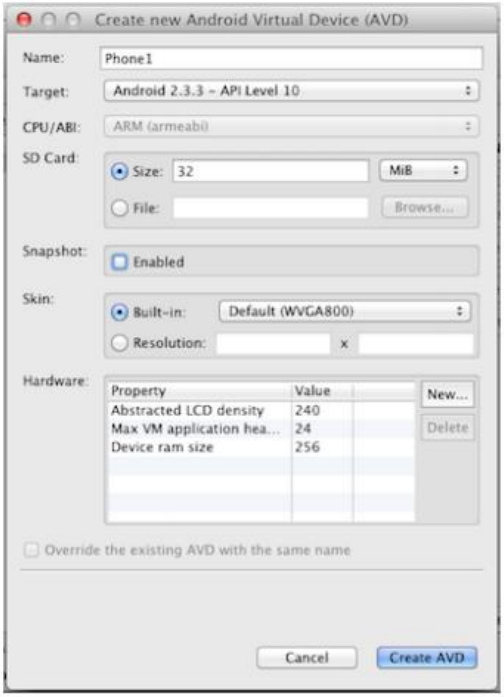
2.1.2 Desenvolvimento Android

Como explica Lecheta (2013), no desenvolvimento Android é utilizado o Android SDK. Este é um software utilizado para o desenvolvimento de aplicativos da plataforma. Nele possui um emulador para fazer uma simulação do celular para o desenvolvido o aplicativo.

Lecheta (2013) também relata que, além disso a plataforma de desenvolvimento possui uma API para a linguagem Java que contém toda as integrações já criadas com as suas classes.

O Android desde sua concepção, já passou por diversas versões onde muitos celulares foram ficando defasados e suas respectivas empresas pararam de dar suporte para atualizações, não sendo mais possível atualizar, explica Lecheta (2013).

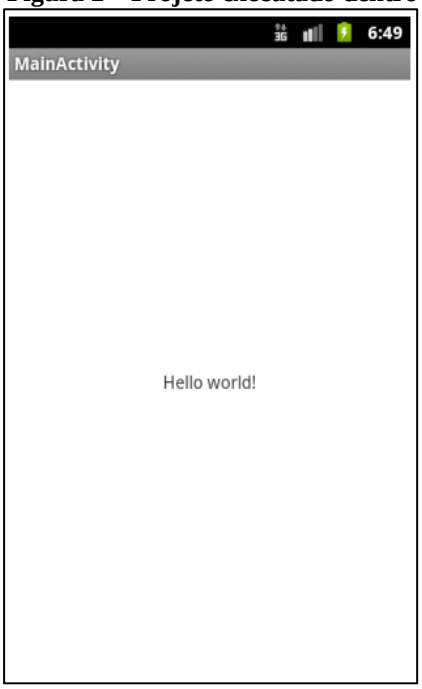
Figura 1 – Criação do ambiente virtual do Android



Fonte: Monteiro (2012, p. 18).

Como mostra Monteiro (2012), na plataforma de desenvolvimento existe a possibilidade de criar aplicativos para dar suporte a celulares de diferentes versões. Conforme pode ser observado na Figura 1, pode-se selecionar outras versões de Android, observa-se também na Figura 1 a versão 2.3.3 que foi utilizada para a criação do projeto.

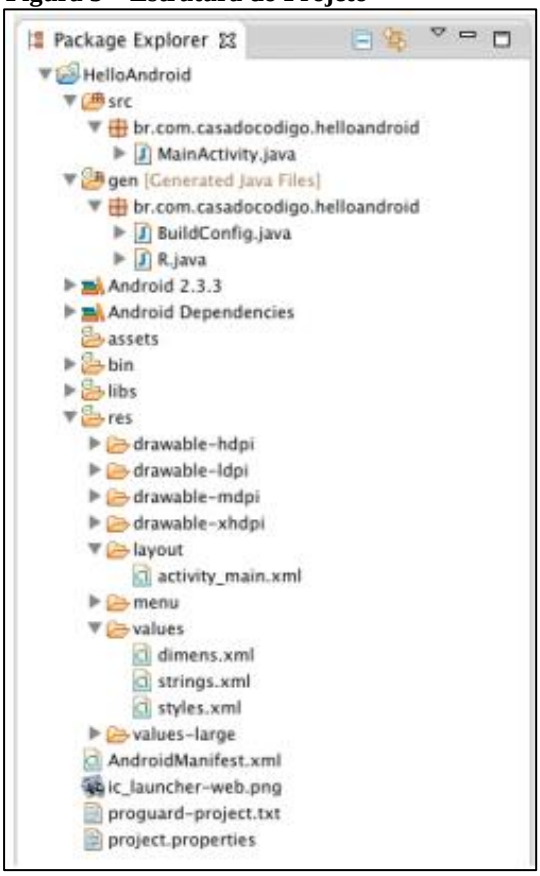
Figura 2 – Projeto executado dentro da ferramenta



Fonte: Monteiro (2012, p. 19)

E após criar o emulador chamado de Android *Virtual device* (AVD), o aplicativo padrão da ferramenta já pode ser executado e abrirá um aplicativo base padrão no dispositivo emulado, como mostra a Figura 2 no exemplo de Monteiro (2012).

Figura 3 – Estrutura do Projeto



Fonte: Monteiro (2012, p. 20)

Na Figura 3, Monteiro (2012) mostra como é organizado a estrutura de pastas dentro de um projeto para celulares Android.

Quadro 1 – Significado das pastas do projeto, conforme analisado na Figura 3

src	pasta dedicada aos armazenamentos dos códigos-fonte do projeto e será onde colocaremos as classes Java que criaremos em nossa aplicação. Repare que já existe uma MainActivity.java que foi criada automaticamente quando criamos o projeto;
res	dedicado ao armazenamento de recursos (arquivos de layout, imagens, animações e XML contendo valores como *strings*, *arrays* etc.), acessíveis através da classe R;
assets	diretório para o armazenamento de arquivos diversos utilizados por sua aplicação. Diferentemente dos recursos armazenados na pasta res, estes são acessíveis apenas programaticamente;
gen	armazena códigos gerados automaticamente pelo plugin, como a classe R que mantém referências para diversos tipos de recursos utilizados na aplicação;
Libs	pasta para armazenar bibliotecas de terceiros que serão utilizadas pela aplicação;
Bin	local utilizado pelos processos de compilação e empacotamento para manter arquivos temporários e códigos compilados

Fonte: Informação retirada de Monteiro (2012, p. 20,21)

No Quadro 1, Monteiro (2012) aborda qual é o significado de cada pasta dentro do projeto Android.

Também na pasta principal, como mostra a Figura 3, é possível observar um documento XML que é obrigatório para todas as aplicações Android. Esse documento guarda informações essenciais sobre a aplicação que está em desenvolvimento, inclui versão mínima do Android para funcionar o sistema, e descreve componentes como: (*activities*, *services*, *content providers* e *broadcast receivers*) que tem funções dentro da aplicação, como exemplifica Monteiro (2012).

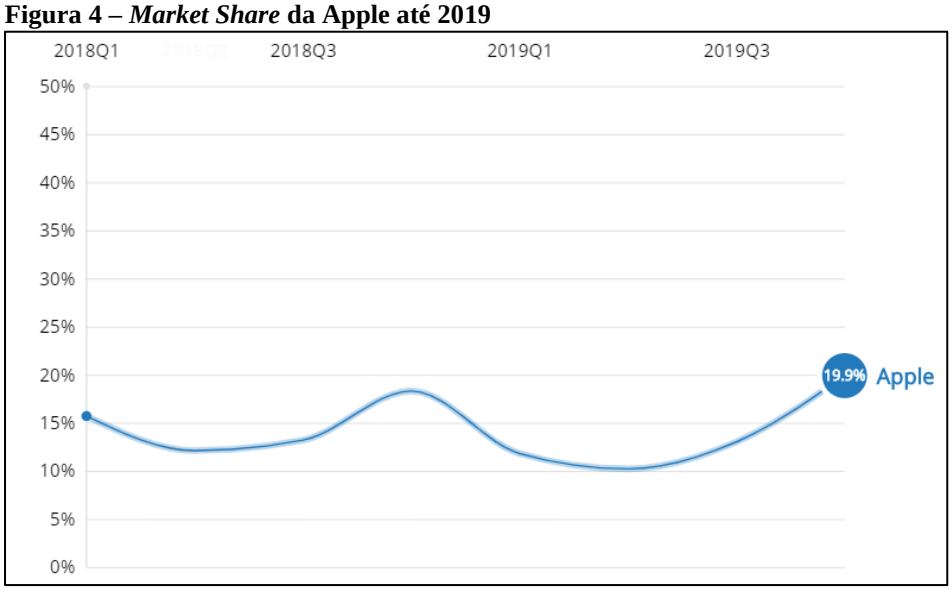
2.2 HISTÓRIA DO IOS E APPLE

Como contextualiza Steil (2012), em 2007 a Apple lançou seu próprio celular e sistema operacional, antes mesmo do Android, tinha seu sistema chamado de iPhone OS, e logo após a chegada do iPad foi trocado para iOS.

Na sua primeira versão, diferentemente do Android, não era possível desenvolver para o celular, somente no ano seguinte, o ano de lançamento do Android, desenvolveram um kit para o iOS como o kit do Android SDK, como explica Steil (2012).

Como diz Silveira e Jardim (2015), o mercado do iOS tem crescido muito no Brasil. Como mostra a Figura 4, o *Market Share* global da Apple ilustra que 19,9% dos celulares vendidos são da Apple, representando 73,8 milhões de celulares vendidos, como aponta o site *International Data Corporation* (IDC).

Figura 4 – *Market Share* da Apple até 2019



Fonte: IDC (2020)

Como complementa Steil (2012), apenas o Market Shares possui números impressionantes, porém o maior atrativo que gera ânimo aos desenvolvedores é o número de vendas na App Store, o que faz com que desenvolvam mais aplicativos para a loja da Apple.

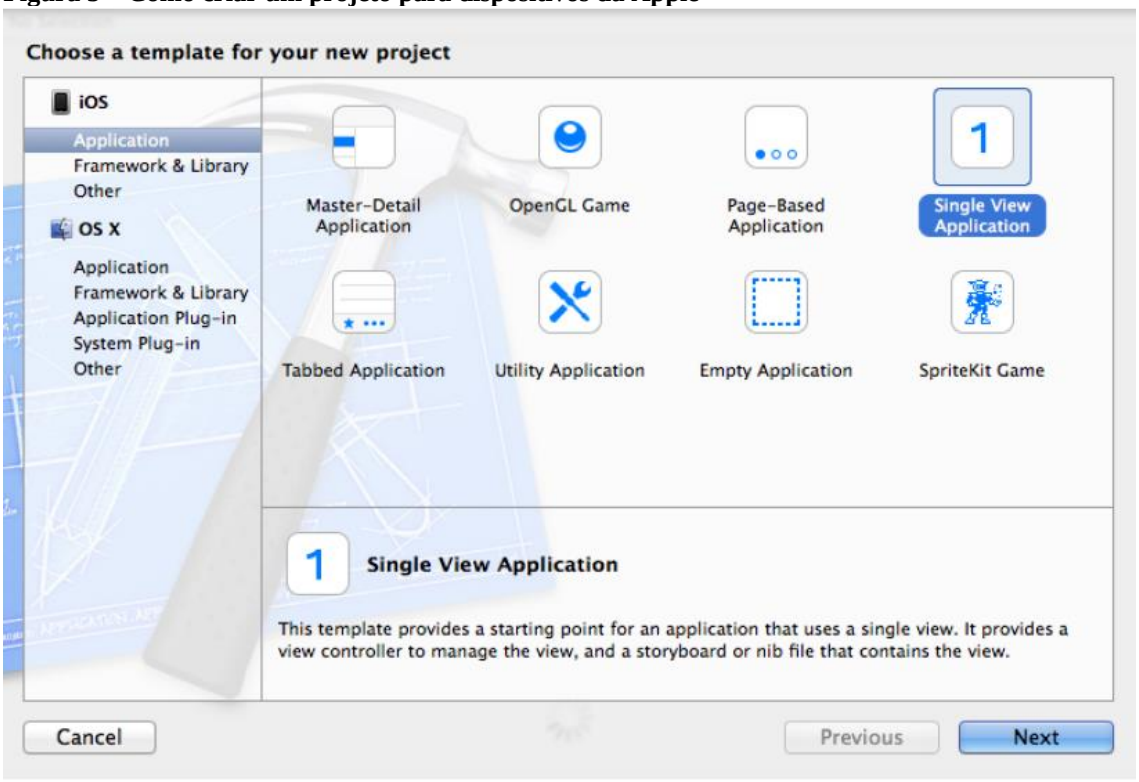
2.2.1 Desenvolvimento com Xcode

No começo dos celulares da Apple, para desenvolver para o sistema nos primórdios de 2008, era preciso comprar o kit de desenvolvimento onde o mesmo só foi se tornar gratuito em 2010, comenta Steil (2012).

Para desenvolver no sistema iOS, Steil (2012) também explica que é necessário possuir um computador que rode OS X, então para criar aplicativos para celulares Apple só é possível através de um computador da própria empresa, como exemplifica.

Também como aborda Steil (2012), a Apple atualmente disponibiliza gratuitamente o programa para desenvolver bem como sua documentação para utilizar na aprendizagem da ferramenta. A ferramenta também inclui um simulador para iPhone e iPad.

Figura 5 – Como criar um projeto para dispositivos da Apple



Fonte: Steil (2012, p. 7)

Na Figura 5, é mostrado como é criado a primeira aplicação para iOS, nesse caso é elaborado uma *single view application*, como mostra Steil (2012).

Steil (2012, p.25) explica o significado de single view application:

> "[...] *template* de projeto *single view application* (ou "aplicação com uma única *view*") não é limitado a uma única tela, mas sim que o código inicial que o Xcode irá criar para você terá uma tela inicial, ficando a cargo do desenvolvedor criar o que mais precisar."

Os outros modelos de template de escolha seguem o mesmo padrão, porém alguns modelos têm funções específicas como o *Tabbed Application* que já traz a estrutura básica de um projeto com *tab bar*", um componente que fica na barra inferior do dispositivo com alguns ícones ou *utility application* que cria um projeto com duas *views* de começo e um botão para alterar entre elas, completa Steil (2012).

2.3 DESENVOLVER EM DUAS PLATAFORMAS DISTINTAS

Como contextualiza Wu (2018), na construção de um aplicativo para a maioria dos usuários, o desenvolvimento móvel precisa ser em duas plataformas separadas, Android e iOS. Evidentemente, as diferenças entre essas duas plataformas são grandes e geralmente exigem diferentes conjuntos de habilidades para a elaboração, como Java/Kotlin para Android e Object-C/Swift apenas para iOS. Assim, desenvolvedores e empresas geralmente têm problemas para lidar com a criação em duas plataformas distintas e complexas.

2.4 JAVASCRIPT

Como explica Flanagan (2013), a grande parte de sites e aplicativos criados nos tempos atuais, usam JavaScript em seu código fonte. E todos os tipos de dispositivos seja, celulares, computadores, notebooks possuem interpretadores, "[...]tornando-a a linguagem de programação mais onipresente da história.", conclui Flanagan (2013, p. 18).

Flanagan (2013) explica que JavaScript é uma linguagem dinâmica onde é possível a utilização de orientação a objetos como forma de programação e explica que a linguagem possui um alto nível. Como conclui Noleto (2020, n.p.), "[...] cuja sintaxe é voltada ao entendimento humano[...]".

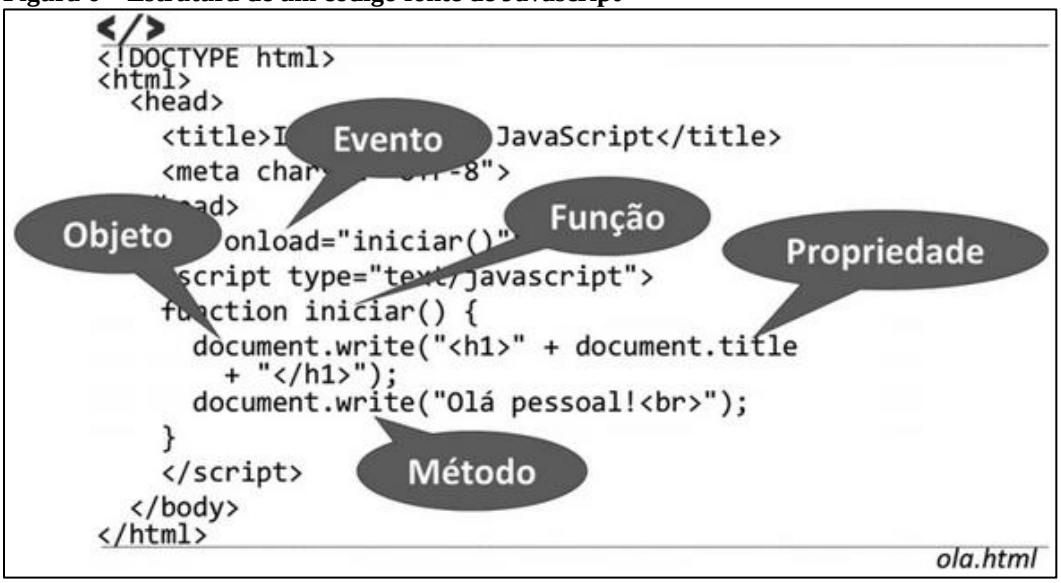
Como justifica Flanagan (2013), JavaScript não é uma linguagem relacionada a programação Java, as duas linguagens possuem apenas nomes parecidos, mas não tem vínculos

em comum. A linguagem JavaScript também com o passar do tempo deixou o termo Script de lado, onde não é apenas uma linguagem de Script, onde agora compõe um código robusto e eficiente.

2.4.1 Como funciona a linguagem JavaScript

Como explica Oliveira e Zanetti (2020), a linguagem JavaScript possui uma característica muito importante para seu comportamento e a forma que é desenvolvida, onde qualquer variável é definida como uma variante. Isso quer dizer que ela pode receber qualquer informação sem predefinir qual o tipo de dado será armazenado como é feito em outras linguagens do mercado. "A linguagem JavaScript básica define uma API mínima para trabalhar com texto, arrays, datas e expressões regulares, mas não inclui funcionalidade alguma de entrada ou saída." Define, Flanagan (2013, p.19).

Figura 6 – Estrutura de um código fonte de Javascript



Fonte: Informação retirada de Oliveira e Zanetti (2020, p. 47)

Oliveira e Zanetti (2020) exemplificam na Figura 6 que a estrutura de JavaScript é utilizado funções, objetos, eventos, propriedades e métodos da mesma forma que é usado no fundamento das linguagens de orientadas a objetos.

2.5 REACT NATIVE

A dificuldade do desenvolvimento para duas plataformas distintas Android e IOS, era necessário fazer um aplicativo para ambos separadamente, e em 2015 a equipe interna do Facebook criou o React Native, Calomeno Junior (2020).

Andrade (2020) explica como React Native é um framework baseado em outro framework, React que foi desenvolvido na linguagem JavaScript, onde era destinado a aplicações Web. React Native foi concebido para o desenvolvimento de aplicações móveis nas plataformas Android e IOS.

Para Andrade (2020), React Native possui algumas características fundamentais que são:

- Usar as informações de interface e recursos nativos dos sistemas Android e IOS usando JavaScript.
- O código fonte é muito parecido com React que é destinado a Web.
- O mesmo escopo de desenvolvimento é compartilhado entre Web e mobile.
- O código fonte é complicado para a linguagem nativa de cada sistema operacional.
- No desenvolvimento com React Native usamos apenas um código fonte para ambas as plataformas.
- O desenvolvimento é multiplataforma onde podemos usar qualquer sistema operacional para desenvolver o código fonte.

Calomeno Junior (2020) complementa que o framework React Native aumenta o desempenho dos desenvolvedores, onde torna prático o desenvolvimento de forma híbrida.

2.5.1 React Native componentes

Como aponta Calomeno Junior (2020), o React Native não utiliza tags HTML para criar seus elementos de código, mas usa formas semelhantes, onde são tags nativas de componentes.

“O React Native constrói as aplicações já com o foco e estrutura no layout. Para facilitar o desenvolvimento das interfaces, o framework trabalha com componentes e subcomponentes exibidos nas telas através do método *render()*.”, Explica Calomeno Junior (2020).

Segundo React Native (2021), os componentes mais básicos e importantes são divididos em:

Quadro 2 – Principais Componentes do React Native

View	O elemento principal, onde irá receber todos os tipos de componentes agregados
Text	Componente que recebe textos digitados na tela.
Image	Componente que mostra imagens na tela.
TextInput	Componente que recebe input de texto.
ScrollView	Componente que recebe múltiplos componentes permitindo a rolagem de tela.
StyleSheet	Componente que irá prover abstração de outros componentes para adicionar camadas similares ao CSS.

Fonte: Documentação do React Native 2021

Os componentes criados no React Native são possíveis reaproveitar de uma forma dinâmicas onde é possível ganhar muito tempo, reescrevendo código onde muito deles aparecem em diversas telas distintas, conclui Rossetti (2020).

2.5.2 ROOK: useState e useEffect

No React Native 16.8 foi inserido os *rooks* que permite que se use estados e outros recursos sem a necessidade de escrever uma classe para isso. É principalmente utilizado para controlar um estado e efeitos de um componente, como explica Jabbar (2020).

Antes para manipular um estado no React Native era necessário criar um arquivo separado e uma classe de manipulação, hoje com os *rooks*, é possível criar no próprio componente e manipular dentro do arquivo do componente, complementa Jabbar (2020).

A partir dele conseguimos criar uma variável, que irá controlar o estado, onde criamos duas variáveis, uma recebe o valor do estado e outra irá mostrar o estado, como exemplifica Hanashiro (2019).

O retorno do useState é uma matriz que sempre contêm itens, o item é variável que é armazenado sempre por convenção “setNome”, onde nome é o item em questão como conclui Negi (2019).

O useEffect serve para lidar com efeitos, assim ele recebe primeiro uma função que será executada assim que o componente renderizar como explica Hanashiro (2019).

O Hook useEffect aceita qualquer tipo de operação, isso inclui efeitos adversos, como tratamento de erros, já que ele irá executar logo após o render. Ele também recebe uma variável

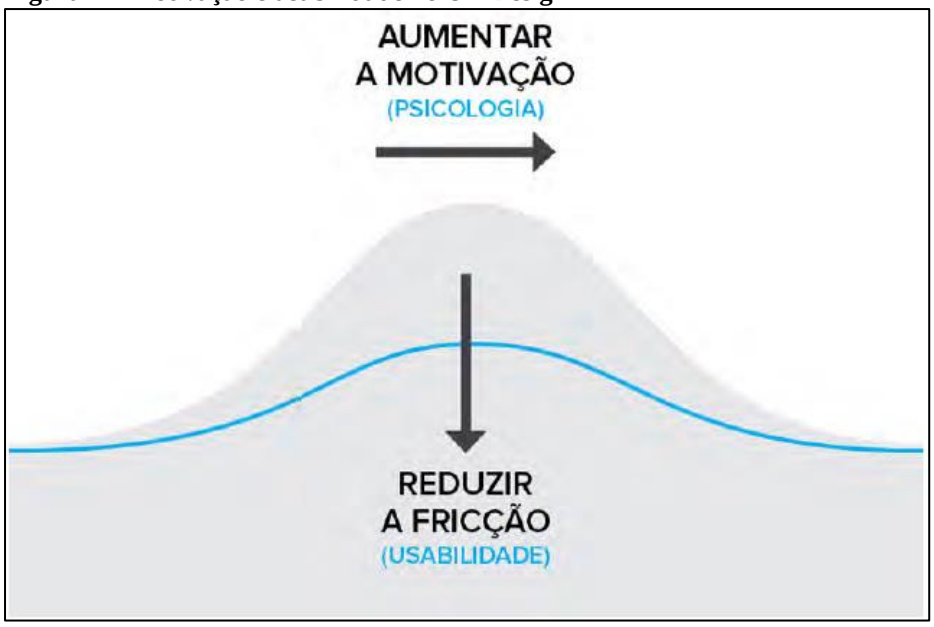
como useEffect que assim que percebe alguma mudança ele muda seu estado, como explica Negi (2019).

2.6 EXPERIÊNCIA DE USUÁRIO

Como aborda Texeira (2014), a experiência de usuário ou UX (*User eXperience*) trata da experiência de quem usa. Pode ser vista como coisas do cotidiano que faz com que você tem um conhecimento sobre as atividades exercidas.

O UX Design trabalha na construção de produtos que possuem boa usabilidade, ajudando usuários a executarem tarefas em pouco tempo e com o mínimo de obstáculo para suas atividades no aplicativo, como é demonstrado na Figura 7, no mesmo passo que apoia princípios da psicologia para incentivar o usuário a continuar usando a ferramenta, como explica Texeira (2014).

Figura 7 – Motivação e usabilidade no UX Design



Fonte: Texeira (2014, p.4)

Como aponta Texeira (2014), o trabalho de UX design é trabalhar no desenvolvimento de produtos para a área digital. “[...] o papel do UX designer é definir questões mais estratégicas[...]”, como fala Texeira (2014, p.7).

Esses aspectos de design têm como objetivo abordar questões da importância de determinadas funções, botões ou para onde será levado o usuário após a troca de tela no produto a ser desenvolvido, completa Texeira (2014).

2.7 BANCO DE DADOS

De acordo com Date (2004), um sistema de banco poderia ser comparado à um arquivamento eletrônico, algo como um recipiente para armazenamento de arquivos, dados que no caso são computadorizados.

“Um sistema de banco de dados é basicamente apenas um sistema computadorizado de manutenção de registros.”, como fala Date (2004, p.3).

Gonçalves (2015) complementa que dados antigamente eram armazenados em planilhas e arquivos de texto, onde eles eram guardados nos computadores dos usuários ou em fitas e disquetes. Para a época era uma evolução, mesmo possuindo muitas falhas e ocasionando muitos problemas.

“Muitos dos problemas estavam relacionados a duplicidade de dados e inconsistências de informações.”, sugere Gonçalves (2015, p.1-2).

2.7.1 Sistema de Banco de Dados

Date (2004) afirma que um sistema de banco de dados é um sistema computadorizado que tem como sua finalidade armazenar as informações que podem ser qualquer coisa que possua um significado, tanto para algum indivíduo quanto para a organização.

Esse modelo de banco de dados relacional pode ser interpretado como uma coleção de dados que são ordenados e adaptados com formato de tabela, interligados por chaves primárias e estrangeiras constituindo em um ambiente mais fluido e sem restrições, conforme Gonçalves (2015).

“[...] o banco pode ser utilizado por todas as aplicações relevantes sem duplicação de dados e sem a necessidade de serem definidos em programas, mas podendo ser, como composto de tabelas ou relações.”, cita Gonçalves (2015, p.10).

Machado (2004) explica que um banco de dados possui as seguintes propriedades:

- é uma coleção lógica e organizada de dados com um significado próprio; caso fosse desorganizada ela não teria um sentido, e não poderia ser referenciada;
- o banco de dados é projeto para ter um propósito de armazenamento de dados com valores específicos, dado as circunstâncias da aplicação;
- ele tem como objetivo representar o mundo real, chamado de minimundo, assim como todas as alterações efetuadas no minimundo implicam no banco de dados;

2.7.2 Modelo Conceitual

De acordo com Machado (2004), o modelo conceitual é uma descrição da realidade que constitui uma visão geral sobre o mundo e os relacionamentos, inteiramente separado dos aspectos de implementação e suas tecnologias.

"Quando falamos em modelo conceitual, estamos nos referindo aquela que deve ser sempre a primeira etapa de um projeto de banco de dados.", explica Machado (2004, p.18).

Machado (2004) esclarece que esse modelo tem como objetivo ser suficientemente compreendido pelo usuário final, para que ele entenda as regras de negócio de forma fácil. Sua definição utiliza uma linguagem de alto nível, mesmo assim visa retratar toda a realidade da organização.

2.7.3 Modelo Lógico

Segundo Machado (2004), no conceito de modelo lógico que é iniciado após o modelo conceitual estar bem definido, é onde começa a abordagem de quais tecnologias serão utilizadas no sistema de banco de dados para estruturar e estabelecer as regras e relacionamentos do modelo conceitual.

2.7.4 Modelo Físico

Como aponta Machado (2004), esse modelo físico irá incorporar e será modelado a partir do modelo lógico, sendo projetado de acordo com os requisitos de processo, sempre pensando em melhorar os recursos computacionais, como:

- tipo e tamanho de campos;
- índices;
- domínio de preenchimento desses campos;
- nomenclaturas;
- exigência de conteúdo;
- gatilhos etc.

Nessa etapa de projeto, como afirma Machado (2004), é a etapa final do projeto de banco de dados, onde será utilizada a linguagem para a construção do banco e definições de montagem.

3. METODOLOGIA DA PESQUISA

Este trabalho de conclusão de curso foi desenvolvido um protótipo de aplicativo mobile para os pais de alunos do Colégio Unidavi. Foi feita uma pesquisa-ação, a pesquisa foi efetuada diretamente com 10 usuários responsáveis do colégio que são funcionários da própria instituição. Foi aplicado uma lista de tarefas previamente montada com o intuído de testar com os usuários todas as funções do sistema e após isso foi aplicado um questionário impresso utilizando perguntas relacionadas ao uso do protótipo.

O presente trabalho buscou compreender e responder alguns problemas: Em relação a forma que é disposta a informação para escolher o estudante, é de fácil entendimento? Houve dificuldade para encontrar as ocorrências do semestre atual? Após a leitura da ocorrência notou que ela assume um status diferente, e é marcada como lida.

Para a metodologia de pesquisa foi utilizado pesquisa de campo empregando uma métrica qualitativa, considerando avaliar a usabilidade do protótipo desenvolvido.

A pesquisa foi operacionalizada da seguinte maneira: Os dados da pesquisa foram levantados a partir de um formulário entregue aos pais e responsáveis dos alunos. Juntamente com um dispositivo móvel para o teste controlado do protótipo previamente instalado, entre os dias 18 de novembro e 19 de novembro de 2021.

4. PROTÓTIPO DE APLICATIVO MÓVEL PARA ACOMPANHAMENTO ESCOLAR DO COLÉGIO UNIDAVI: MÓDULO PARA OS PAIS OU RESPONSÁVEIS

O protótipo foi montado com o intuito de facilitar a busca de informações referente aos acadêmicos do Colégio Unidavi, para que os pais tenham o controle sobre tudo referente as notas, ocorrências e financeiro no âmbito acadêmico do aluno.

Desta forma o protótipo visa tornar prático e dar usabilidade para as ferramentas que já são fornecidas pelo Colégio Unidavi para o acompanhando escolar.

4.1 ESCOPO DO PROJETO

No início do projeto foi realizada uma reunião com a direção do Colégio Unidavi. Buscando-se identificar as necessidades para o desenvolvimento do protótipo de aplicativo, visando facilitar o acesso dos pais e responsáveis dos alunos do Colégio Unidavi.

Notou-se principalmente a dificuldade na utilização do sistema atual disponibilizado em conjunto com o Sistema Acadêmico utilizado pelo Colégio Unidavi. Esse sistema foi apenas adaptado para acesso mobile, mas não foi de forma otimizada.

4.2 ESTADO DA ARTE

A empresa que desenvolveu o sistema acadêmico da Unidavi, possui um aplicativo próprio para uso, de alguns recursos, a empresa possui um WEBVIEW, que não aplica de uma forma correta a visualização e usabilidade as informações do responsável acadêmico.

O aplicativo simula um o próprio navegador, desta forma fica muito difícil achar as informações numa tela pequena como de celular. Desta forma o aplicativo não usa APIs para acessar as informações do sistema acadêmico da Unidavi, ele simula um navegador que busca as informações.

A pesquisa também se estendeu por aplicativos na loja da Google, mas não existem modelo de aplicativo com esta finalidade, que seria um aplicativo pronto para a inserção de dados de API e que seja moldável aos requisitos da instituição.

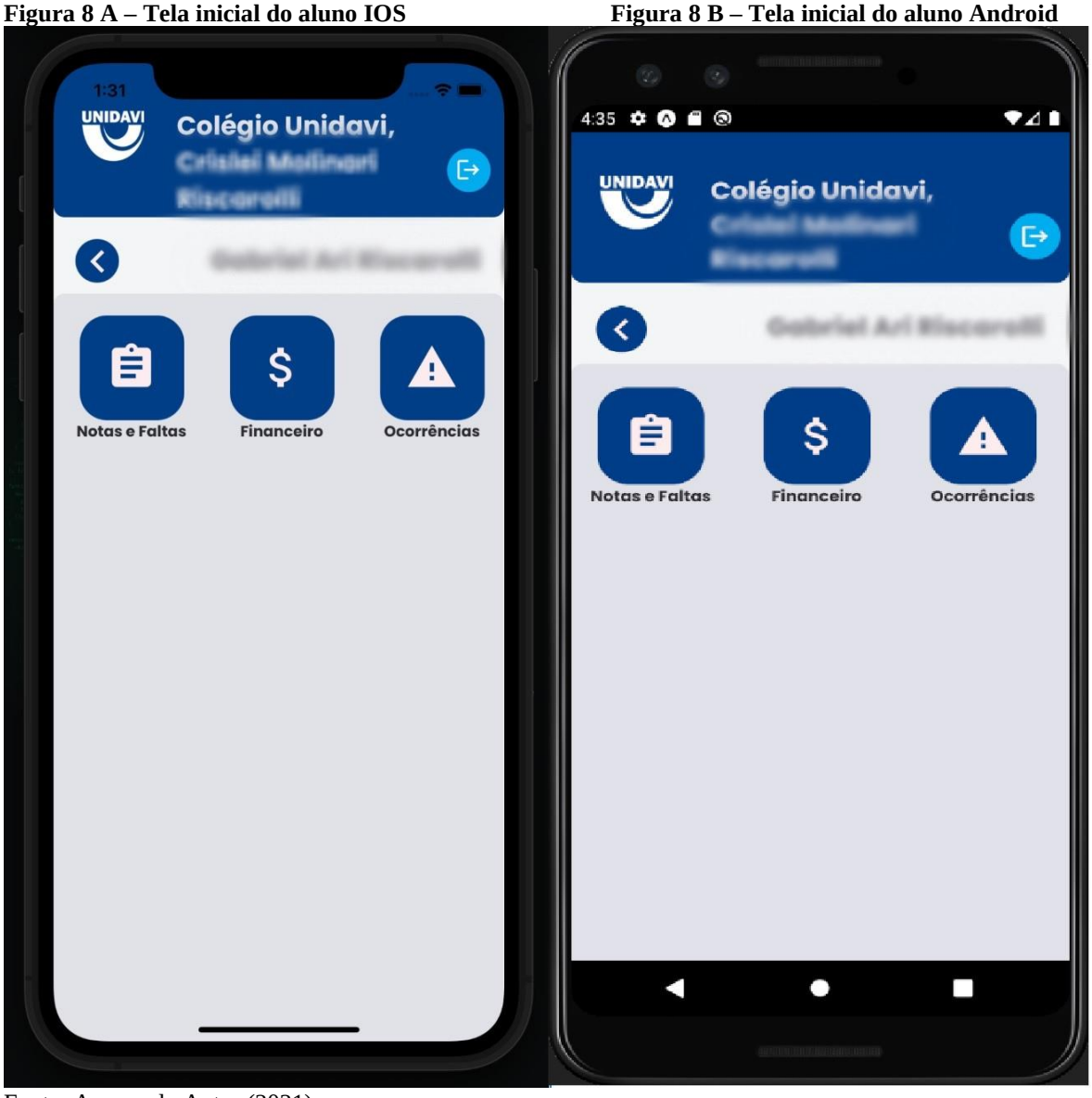
4.3 TELAS DO PROTÓTIPO

Todas as telas mostradas do protótipo contêm conteúdos reais e por esse motivo foi inserido máscara nos dados das Figuras apresentadas, os dados reais servem para dar autenticidade as informações protótipo.

Todas as telas desenvolvidas no protótipo, foram testadas em celulares IOS e Android, desta forma o projeto é compatível com todas as plataformas, utilizando apenas React Native para desenvolver para sistemas distintos.

O protótipo foi desenvolvido com três estruturas principais conforme pode ser observado na Figura 8:

Figura 8 A – Tela inicial do aluno IOS **Figura 8 B – Tela inicial do aluno Android**



Fonte: Acervo do Autor (2021)

Notas e Faltas que irá agregar toda a informação acadêmica que corresponde a sala de aula com notas, frequências, disciplinas e curso.

O Financeiro que corresponde aos boletos em aberto e o código de barra podendo ser copiado para o pagamento online.

E por último a consulta de ocorrências onde o responsável tem as informações de anotações que o colégio cria para informá-los e anotações de professores sobre o desempenho acadêmico.

4.3.1 Tela de notas e faltas

O desenvolvimento do protótipo deu seu início na definição das telas baseando-se no levantamento inicial das funcionalidades. Onde definiu-se que o protótipo teria como funcionalidades principais, uma tela que apresente as notas e frequências do(s) filho(s), onde também apresenta o status de andamento do curso.

Figura 9 – Protótipo da tela de listagem de cursos

Fonte: Acervo do Autor (2021)

A listagem conforme pode ser observada na Figura 9 representa todos os cursos já cursados pelo aluno onde o primeiro é sempre o curso atual do aluno, onde a listagem é apenas o começo das informações apresentadas relacionadas ao curso.

Nesta listagem pode também aparecer cursos de extraclasse e matérias de extensão relacionadas ao colégio como banda e atividades esportivas.

Figura 10 – Frame da listagem de disciplinas do aluno

Fonte: Acervo do Autor (2021)

Após o responsável do aluno selecionar de qual curso ele pretende visualizar as informações do filho, ele terá a visão da Figura 10 onde terá toda a listagem de todas as disciplinas.

Nesta listagem o responsável conseguirá ver as informações de nome e status da disciplina, seja ele: cursando, concluído ou reprovado entre outros status possíveis. Outra informação disponível na tela é a frequência geral do aluno desde o início do ano.

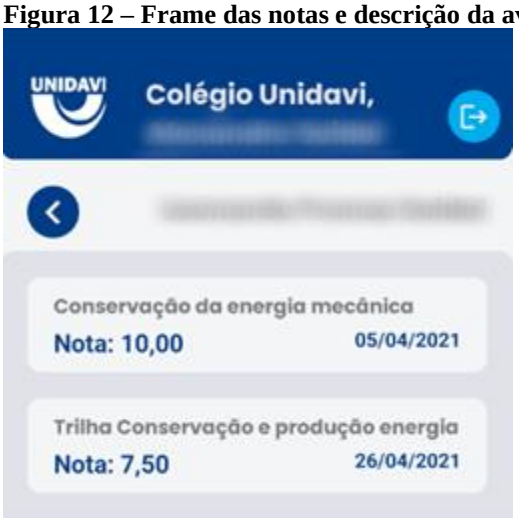
Figura 11 – Listagem dos semestres do aluno

Fonte: Acervo do Autor (2021)

Depois que é selecionado a matéria temos outros botões de avanço que são representados pelos trimestres do aluno do colégio como representado na Figura 11.

Cada trimestre terá uma média, e nesta mesma tela teremos também as informações, média dos trimestres será calculada a média entre todos os trimestres, o exame caso o aluno não atinja a nota 6 de média e por fim média final que irá calcular e trazer o resultado entre exame e média dos trimestres caso o aluno não atinja a nota mínima.

Figura 12 – Frame das notas e descrição da avaliação



Fonte: Acervo do Autor (2021)

Na tela que sucede a visualização do semestre temos a tela que representa a nota do aluno como mostra a Figura 12, esse detalhamento são todas as avaliações que foram feitas e compõem a média do trimestre detalhando o nome da avaliação, a nota obtida e a data da aplicação da avaliação.

4.3.2 Tela de financeiro

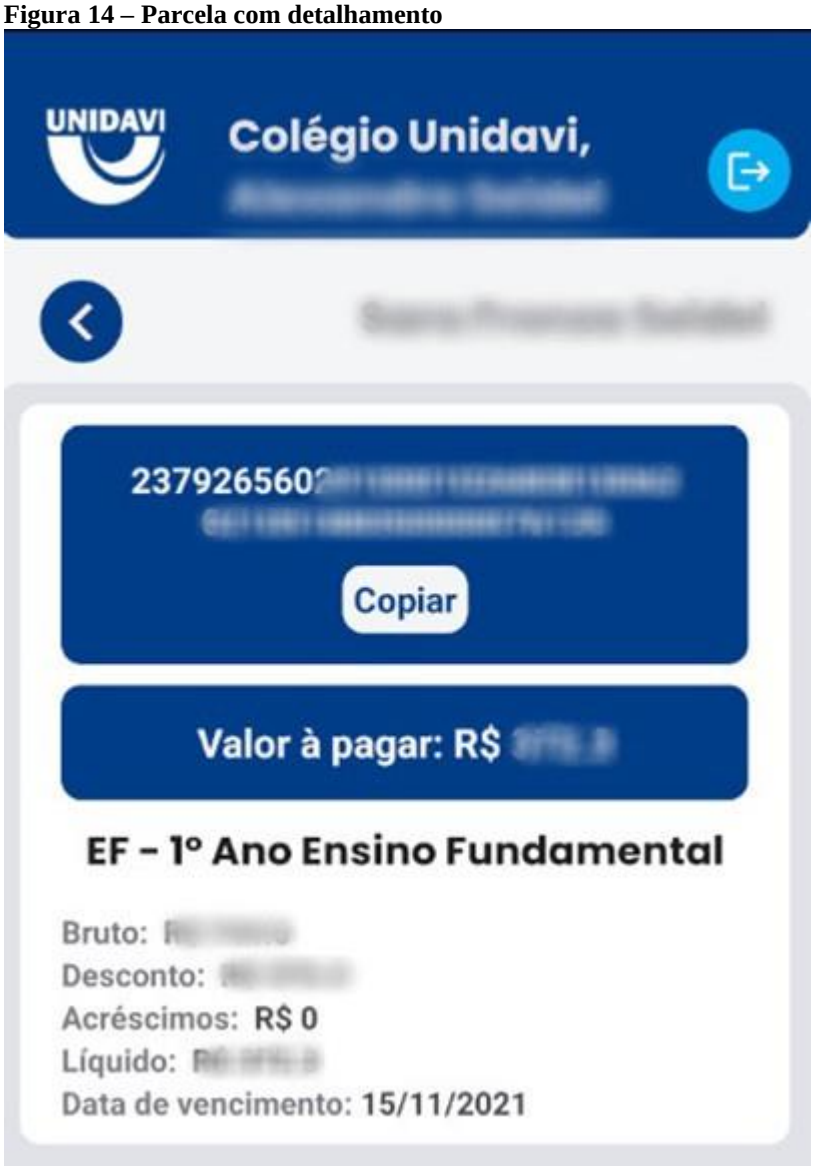
Na parte que contempla a área financeira, temos a listagem de parcelas do filho selecionado. As parcelas são geradas e controladas pelo setor financeiro da Unidavi, podendo ou não gerar para o ano inteiro.

Figura 13 – Parcelas do financeiro

Fonte: Acervo do Autor (2021)

Na visualização desta informação, referente a parcela como aparece na Figura 13, possui a pré-visualização da data de vencimento e o valor a ser pago.

Figura 14 – Parcela com detalhamento



Fonte: Acervo do Autor (2021)

A Figura 14 está representando toda a informação detalhada correspondente a parcela X do aluno Y, onde trás toda a informação financeira sobre determinado boleto, como descontos, acréscimos, valor líquido e a data de vencimento.

Todas essas informações listadas são de suma importância para a transparência com os pais e responsáveis do Colégio da Unidavi, onde eles já possuem esses conhecimentos utilizando a ferramenta através do navegador.

No topo deste boleto temos o código do boleto para ser pago e um botão logo abaixo para efetuar a cópia da informação, facilitando assim o responsável de ter que digitar o código.

4.3.3 Tela de ocorrências

Na tela de ocorrências, envolve toda a parte de comunicação entre o colégio e o responsável, é um canal de comunicação direto sobre tudo o que acontece com o aluno no ambiente escolar.

Figura 15 – Diferentes tipos de ocorrências

Fonte: Acervo do autor (2021)

Nesta tela é segmentado todos os tipos visíveis para o responsável de ocorrências do colégio, onde não é restrito a somente os que aparecem da Figura 15.

É possível que exista N tipos diferentes, dependendo apenas de a própria instituição achar necessário colocar outros modelos.

Na listagem que aparece na Figura 15 temos: comunicados (em geral para o colégio todo), conteúdos para exames (o que cairá no exame para os pais estarem cientes), direção/coordenação (são comunicados vindos desta gestão do colégio). Por fim ocorrências que são destinadas aos alunos em específico podendo ou não indicar falta justificada, problemas e situações em alguma matéria.

Todas essas formas de ocorrências podem ser destinadas a alunos específicos, turma, disciplina, curso ou toda a unidade escolar, isso depende da informação proposta e o que será repassado.

Figura 16 – Listagem de ocorrências por ano

Fonte: Acervo do autor (2021)

Quando abrimos a tela relacionada a alguma categoria de ocorrência temos a Figura 16 que mostra as ocorrências cadastrada para o aluno, mesmo que essa ocorrência seja para a turma, o vínculo é sempre no aluno.

Na tela temos quem cadastrou, que pode ser uma disciplina, curso, professor ou a própria secretaria que emitiu, nesta forma de organização o responsável consegue se situar de onde vem a informação que foi cadastrada.

O detalhamento colocado sobre do que se trata a ocorrência é colocada de uma forma sucinta para que seja clicado e lido na sua totalidade, pois pode haver textos muito grandes.

Ao abrir a ocorrência detalhada o pai ou responsável marcará que foi lido a ocorrência do filho, e a tela inicial de ocorrências irá mudar de cor para identificar ocorrências não lidas.

4.4 DESENVOLVIMENTO

O protótipo foi desenvolvido usando o React Native como a linguagem de programação, onde é uma linguagem que se assemelha muito ao CSS e HTML para seu desenvolvimento gráfico e interface, por isso tornou o projeto de fácil semelhança ao desenvolvimento web.

Figura 17 – Código referente as ocorrências

```
83          <Container>
84              <Header>
85                  <HeaderWrapper>
86                      <Image source={Logo} style={{ width: 59, height: 73 }} />
87                      <Frase>
88                          <Greetings>
89                              Colégio Unidavi,{'\n'}
90                              {user.nome}
91                          </Greetings>
92                          <ButtonLogin>
93                              <Logout onPress={logout} />
94                          </ButtonLogin>
95                      </Frase>
96                  </HeaderWrapper>
97              </Header>
98              <Body nameChild={child.childrenName} />
```

Fonte: Acervo do autor (2021)

Para demonstrar um código semelhante ao CSS temos a Figura 17 que representa a logo superior na linha 86. Logo após na linha 89 e 90 a saudação utilizando-se uma tag *{user.nome}*. Essa tag exibe a informação do nome do responsável.

O botão logout que está na linha 93 e logo baixo também temos a *tag Body* na linha 98, os dois configuram um componente importado as ocorrências da Figura 17.

Como no *HTML* dispomos de tags, que podemos estilizar da forma que bem entendermos utilizando o *CSS* do React Native. Mas diferentemente do *HTML* não temos parametrizada a estrutura HTML, que se usarmos uma tag *H2* ele irá aplicar a função parametrizada do navegador.

No caso do React Native temos que programar toda a estrutura do zero usando nossos próprios parâmetros, nesse caso é usado uso por convenção o container para englobar toda a árvore de tags subsequentes a Figura 17.

Figura 18 – Modelo de componente

```
1   import React, { ReactElement } from 'react';
2   import { useNavigation } from '@react-navigation/native';
3   import { Back } from '../Back';
4   import { Container, PersonName } from './styles';
5
6   interface Props {
7       nameChild: string;
8   }
9
10  export function Body({ nameChild }: Props): ReactElement {
11      const navigation = useNavigation();
12
13      function handleBack(): void {
14          navigation.goBack();
15      }
16
17      return (
18          <Container>
19              <Back onPress={() => handleBack()} />
20              <PersonName>{nameChild}</PersonName>
21          </Container>
22      );
23  }
```

Fonte: Acervo do autor (2021)

Os componentes são objetos muito práticos para o desenvolvimento no React Native, eles ajudam a desenvolvermos projetos mais práticos. É possível desenhar componentes genéricos para usar em diversas telas diferentes do sistema ou usar o mesmo componente na mesma tela diversas vezes dependendo apenas do projeto.

O modelo de componente evidenciado pela Figura 18 é o componente *Body*, onde temos o *export function Body* na linha 10 e que está sendo indicado a exportação para telas e outros componentes poderem importar no seu código e usar como uma parte principal ou um item novo.

Desta forma de exportação de componentes, esse Body em específico, está presente em todas as telas do sistema, tornando uma forma de reciclar o código bem dinâmica.

Nesta Figura 18 também podemos notar outra informação importante que é a *interface* na linha 6, no qual é utilizada para indicar que chegará nesse componente um *nameChild.*

Também ele será do tipo *string*, ou seja, a interface de nome *Props* (que poderia receber qualquer nome), quando chamarmos o *Body* que é nosso componente, o código exigirá que enviamos um parâmetro, este parâmetro será *nameChild*.

Podemos notar na Figura 18 que no topo da tela presente na linha 3, é feito o *import* do *Back* (um botão de voltar) que é outro componente importado dentro de Body, podendo assim termos vários componentes um dentro do outro.

4.4.1 Buscando dados da requisição API

A busca de informações da API da Unidavi é algo essencial para o protótipo pois através dele é possível mostrar as telas e informações que o responsável consegue visualizar no aplicativo.

Figura 19 – Serviço de API

```
1  import axios from 'axios';
2
3  const api = axios.create({
4      baseURL: 'https://unidavi.edu.br',
5      // baseURL: 'http://10.3.8.35:8117',
6  });
7
8  export { api };
```

Fonte: Acervo do autor (2021)

Utilizando o *axios*, uma biblioteca do React Native como mostra a Figura 19, que são feitas conexões externas de um cliente HTTP baseado em Promises para fazer requisições. Onde é possível criar a conexão entre o serviço de API da universidade com o aplicativo.

Utilizando desta forma bem prática, essa conexão é possível fazer os *GETs* e *POSTs* necessários para o aplicativo rodar e todas as respostas que são trazidas pelo *axios*, são transformadas em um JSON.

Por padronização do React Native os arquivos que representam as requisições APIs recebem o nome de api.js, e dentro do *axios* foi criado uma constante chamada API como mostra a linha 3 da Figura 19, que é exportada para fora do arquivo.

Na linha 4 temos a URL que será utilizada no sistema inteiro, no qual o *axios* concatena os parâmetros juntamente com a baseURL.

Figura 20 – Listagem de ocorrências via API

```
41  export function Occurrences(): ReactElement {
42      const { user, logout } = useAuth();
43      const navigation = useNavigation();
44      const { ocurrences, handleFetchMore, loadOcurrences } = useOcurrences();
45      const route = useRoute();
46      const { child, occurenceId } = route.params as Params;
47
48      const [selectedSemester, setSelectedSemester] = useState<SemesterDTO>({} as SemesterDTO);
49      const [semesterAll, setSemesterAll] = useState<SemesterDTO[]>([]);
50      const [loading, setLoading] = useState(true);
51
52      useEffect(() => {
53          api.get(child.links.childrenSemesters)
54              .then(response => {
55                  setSemesterAll(response.data);
56                  setSelectedSemester(response.data[0]);
57              })
58              .catch(error => {
59                  console.error(error);
60              });
61      }, [child.links.childrenSemesters]);
```

Fonte: Acervo do autor (2021)

Na Figura 20 temos a constante que armazena Semestre na linha 48, que é representada por uma função do tipo *GET* simbolizado por *selectedSemester* e *SET* representado por *setSelectedSemester*. Esta constante é a forma de controle um estado dentro *React Native*, com ele podemos definir e atualizar o estado.

Essas informações são armazenadas no momento que a tela é renderizada e fica apenas em memória enquanto o aplicativo está aberto.

Na mesma Figura 20 é possível também ver uma constante que é um useState na linha 50, há uma constante Loading, onde ela recebe *true* como parâmetro inicial, e caso não consiga mostrar os dados na tela irá receber false. Desta forma é possível controlar o que é mostrado na tela caso tenha alguma informação incorreta.

Depois de criado o estado de armazenamento dos dados referentes ao semestre, é usado um *useEffect* que é um *hook*, que efetua suas funções depois da tela ser renderizada, e a cada atualização de tela, ele se atualiza, permitindo assim caso tenha mudanças nos dados, ele faça atualizações pontuais nas informações de tela que são buscados.

Na Figura 20, o *useEffect* está buscando na API para trazer todos os semestres referentes aquele aluno e gravando, no *setSemesterAll*. Essa informação está sendo inserida para que a tela de semestres apareça as ocorrências por semestre como na Figura 16.

A outra informação que está sendo gravada, é o *setSelectedSemester*, que é preciso saber qual é o ponto inicial da busca de semestres, para que a informação sempre seja a mais recente, foi definido para pegar sempre a posição 0 que corresponde ao semestre atual cursado.

Figura 21 – Modelo de DTOs

```
1   export interface SemesterDTO {
2       semesterId: number;
3       semesterDescription: string;
4   }
```

Fonte: Acervo do autor (2021)

O estado que foi criado retratado pelo semestre recebe um modelo chamado SemesterDTO mostrado na Figura 21, são padrões predefinidos para aplicar na busca de dados e informações, onde é visto na Figura 20 que possui um DTO para a criança, que nada mais é que uma forma de achar os dados de uma forma simplificada.

Figura 22 – Modelo de dados de API

```
1  [
2      {
3          "semesterId": 978,
4          "semesterDescription": "2021-Col"
5      },
6      {
7          "semesterId": 818,
8          "semesterDescription": "2020-Col"
9      },
```

Fonte: Acervo do autor (2021)

Utilizando DTOs é possível controlar exatamente os dados que estão retornando da API utilizado na Figura 21.

Desta forma foi feito uma “cópia” dos retornos da Figura 22, das linhas 3 e 4 para que seja exatamente a informação que está sendo listada nas telas do aplicativo.

Como é possível analisar, na Figura 17 na linha 98 temos o *child.childrenName*, no qual o *child* é um objeto de um DTO, então desta forma trazemos a informação do nome da criança de dentro de *child*.

Portanto os DTOs são uma forma de representar o que está dentro de requisições APIs, onde torna muito mais fácil a sua manipulação.

4.4.2 Exibindo dados da requisição API

Para a listagem de dados vindos do *useEffect* foi utilizado um componente, que neste caso não foi criado um personalizado, foi utilizado um próprio do *React Native*, a *FlatList*.

A FlatList é mais performática pois ela renderiza apenas o tamanho total da tela, e depois vai trazendo as informações conforme a tela vai sendo rolada para baixo, caso tenha muitos dados facilita para que não quebre o aplicativo.

Figura 23 – *FlatList* das ocorrências do aluno

```
158     <FlatList
159         data={ocurrences}
160         keyExtractor={item => String(item.occurrenceChildrenId)}
161         showsVerticalScrollIndicator={false}
162         contentContainerStyle={{
163             paddingBottom: 50,
164         }}
165         onEndReachedThreshold={0.1}
166         onEndReached={({ distanceFromEnd }) => handleNextPage(distanceFromEnd)}
167         renderItem={({ item }) => (
168             <OccurrenceButton
169                 onPress={() => occurrenceComplete(item.occurrenceChildrenId)}
170                 dateVerify={item.occurrenceViewDateTime}
171                 dateOccurrence={convertDate(item.occurrenceChildrenDate)}
172                 descriptionOccurrence={item.occurrenceChildrenDescriptionShorten}
173                 disciplineOccurrence={
174                     item.occurenceChildrenDiscipline === null
175                         ? 'Colégio Unidavi'
176                         : item.occurenceChildrenDiscipline
177                 }
178             />
179         )}
180         ListEmptyComponent={() => <EmptyFlatList title="Nenhuma ocorrência cadastrada" />}
181     />
```

Fonte: acervo do autor (2021)

A *FlatList* possui diversos parâmetros que precisam ser passados para que ela seja montada, é preciso informar de onde virá os dados que serão listados, no caso da Figura 23, os dados são as ocorrências que é um *useState* que está sendo armazenado no estado e trazendo toda a listagem pelo *useEffect* faz a inserção de todos os dados no estado.

Para esse componente temos que pôr um *keyExtractor* que é a chave do dado que estamos listando, pois precisamos de índices únicos.

Depois de fornecer os dados que serão listados e a chave única de listagem, a *FlatList* possui o *renderItem* esse parâmetro que é passado pelo componente serve para propriamente trazer na tela todos os itens separando-os pelas suas respectivas chaves.

Para cada chave gerada, o parâmetro de render cria um componente chamado *OccurrenceButton*, esse componente criado, é passado alguns parâmetros que são chamados na criação de interfaces, que neste caso compõe um botão, data, descrição e disciplina, desenhando na tela a Figura 16.

4.4.3 Armazenamento de dados de login

Para o projeto foi utilizado para o desenvolvido uma aplicação API a autenticação de login e senha usando a base de login e senha do sistema acadêmico da Unidavi.

Figura 24 – Função de login

```
53      async function login({ username, password }: SignInCredencials) {
54          try {
55              const { access_token, user } = await authorization({ username, password });
56
57              api.defaults.headers.authorization = `Bearer ${access_token}`;
58              setData({ token: access_token, user });
59
60              await AsyncStorage.multiSet([
61                  ['@colegiounidavi:user', JSON.stringify(user)],
62                  ['@colegiounidavi:token', access_token],
63              ]);
64          } catch (error) {
65              console.error(error);
66          }
67      }
```

Fonte: Acervo do autor (2021)

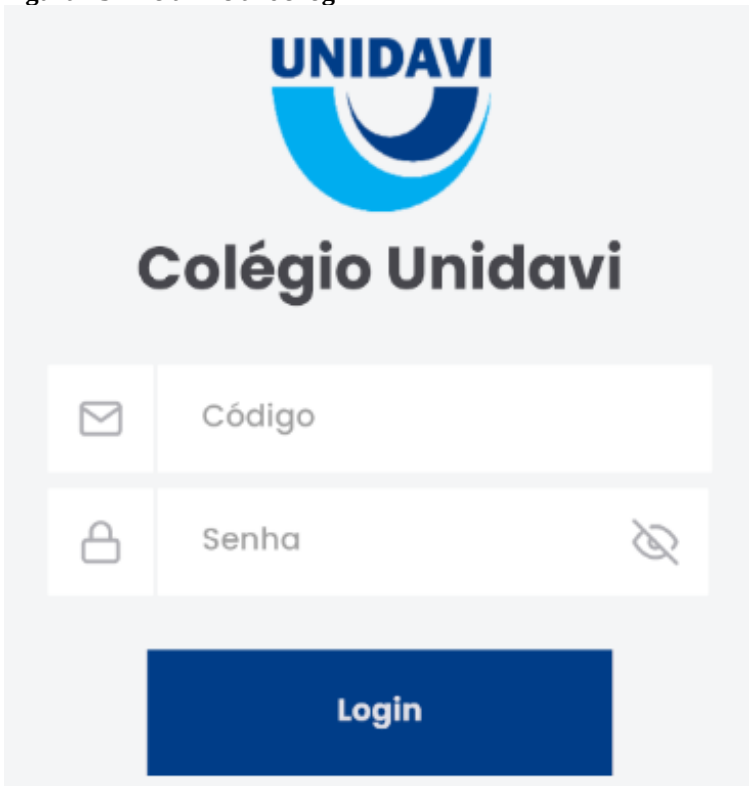
A função que engloba a parte de login do protótipo é a representada na Figura 24, que se chama login, esta função assíncrona (quando devemos aguardar a informação). Esta função é uma parte do *Hook* de autenticação onde possui, logout e outras validações.

Quando executamos a solicitação de login o user, que é uma interface de usuário com as informações de login e senha, e os dados de access_token (chave de acesso) é salvo em uma constante que está representada na linha 55.

Após armazenar estas informações na constante aplicamos o header de uma requisição API, que nada mais é que jogar para requisição o cabeçalho da requisição.

Caso isso dê certo ele irá armazenar no banco interno do celular chamado de *asyncStorage* onde ele armazena um JSON do *user* e *access_token* que está sendo mostrado na linha 61 e 62. E caso aconteça algum problema, seja a validação de login e senha ou algum outro problema de API, a função irá cair no *catch* de erros e retornar o problema.

Figura 25 - Tela inicial de login



Fonte: Acervo do autor (2021)

A função login da Figura 24 recebe usuário e senha que vem a partir do preenchimento do formulário da tela de login como mostra a Figura 25.

Esse login e senha recebe um tipo chamado de *SignInCredencials* na linha 53 da Figura 24, que é uma *interface* de que representa a inserção das informações do usuário.

4.4.4 Serviço de ocorrência

Para o desenvolvimento das ocorrências foi preciso criar um *service*, como é chamado no React Native, esse serviço é uma função separada das screens pois tem como objetivo ser uma função global.

Esta função de ocorrência é diferente das outras pois ela tem duas funções, fazer um *POST* nas ocorrências e marcar como lida e fazer sistema listar por semestres olhando informações antigas do aluno.

Quando é aberto a ocorrência ler a informação e voltar, o sistema deve atualizar a tela anterior afim de mostrar que o que acabou de ser aberto foi marcado como lido pelo sistema. Desta forma a ocorrência está como um serviço e não como funções dentro das próprias screens.

Figura 26 – Função para passar parâmetros com retorno useCallback

```
33      const loadOcurrences = useCallback(async ({ selectedSemester, child, occurenceId }: LoadOcurrenceProps) => {
34          if (selectedSemester.semesterId) {
35              try {
36                  const response = await getAll({
37                      childrenid: child.childrenId,
38                      occurrenceid: occurenceId,
39                      semesterid: selectedSemester.semesterId,
40                  });
41                  setOcurrenceData(response);
42                  setOcurrences(response.results);
43              } catch (error) {
44                  console.error(error);
45              }
46          }
47      }, []);
```

Fonte: Acervo do autor (2021)

Na Figura 26, a função representada pela constante *loadOccurences* na linha 33, recebe diversos parâmetros vindos das ocorrências, essa função global, receberá todas as informações da tela que busca as ocorrências.

O motivo da criação desta *Hook* global se é necessário pois a atualização de tela necessita que exista uma função global para que quando acesse uma tela ou outra do sistema, ela sempre seja recarregada independentemente de onde estiver.

Figura 27 – Função de busca de ocorrências

```
4   export async function getAll({ childrenid, occurrenceid, semesterid }: GetAllProps): Promise<OccurrenceResult> {
5       try {
6           const response = await api.get(
7               `/api/v1/parents/children/${childrenid}/semester/${semesterid}/occurences/${occurrenceid}`
8           );
9           return response.data;
10      } catch (error) {
11          console.error(error);
12          throw new Error('erro de autenticação');
13      }
14  }
```

Fonte: Acervo do autor (2021)

A Figura 27 demonstra uma requisição de API, que recebe parâmetros relacionado as ocorrências de determinado aluno, como é visto na linha 4, recebendo uma *Promise*.

Todos esses parâmetros são recebidos pela Figura 26, na linha 33 possui um o *useCallback*, que é utilizado para enviar e retornar uma versão nova da informação caso a informação tenha sido alterada. Desta forma, a função global irá ficar monitorando caso tenha alguma alteração e assim atualizar as informações das ocorrências.

É um objeto que está disponível para ser mostrado ou pode nunca ser mostrado, uma situação que depende de outros fatores, que nesse caso é ter ou não ocorrência para ser listado. Desta maneira, a função que busca as informações de ocorrências torna global.

Figura 28 – Função para pegar próxima página

```
49      const handleFetchMore = useCallback(() => {
50          if (ocurrenceData.next) {
51              getNextPage(ocurrenceData.next)
52                  .then(response => {
53                      setOcurrenceData(response);
54                      setOcurrences([...response.results, ...ocurrences]);
55                  })
56                  .catch(error => {
57                      console.error(error);
58                  })
59                  .finally(() => {
60                      // setLoadingMore(false);
61                  });
62          } else {
63              // setLoadingMore(false);
64          }
65      }, [ocurrenceData.next, ocurrences]);
```

Fonte: Acervo do autor (2021)

A Figura 28 está representando uma função de useCallback, onde que irá enviar e receber de volta a informação de que foi trocado de aba das ocorrências, onde ela envia um parâmetro da linha 51 que pega sempre a próxima página.

A função também recebe uma resposta, na linha 53, esta resposta então é utilizada na linha 54, ela pega todos os resultados e atualiza pelos novos. Desta forma ela organiza as informações referente as ocorrências lidas, jogando a ocorrência que foi lida para baixo e a que ainda não foi para cima.

Figura 29 – Função para trocar de página

```
16  export async function getNextPage(url: string): Promise<OccurrenceResult> {
17      try {
18          const response = await api.get(url);
19          return response.data;
20      } catch (error) {
21          console.error(error);
22          throw new Error('erro de autenticação');
23      }
24  }
```

Fonte: Acervo do autor (2021)

Para a função relacionada a troca de página separando por semestres onde todas as categorias listadas na Figura 15 possuem o sistema de paginação. No Colégio Unidavi há muitas notificações por ano letivo.

Desta forma fica mais intuitivo o sistema de paginação que é chamado quando o usuário rola a página para baixo, trazendo mais ocorrências e deixando o sistema intuitivo.

Na Figura 29 o *getNextPage* é recebe um parâmetro na linha 16, que é a URL que já está na DTO das ocorrências, essa informação então utiliza o *axios* para executar um *GET* na próxima página, e assim sucessivamente até chegar no final das páginas.

Figura 30 – Função para atualizar ocorrência

```
26  export async function updateReaded(occurenceId: number): Promise<UpdateResponse> {
27      try {
28          const response = await api.post(`/api/v1/parents/children/occurences/${occurenceId}/occurrence`);
29
30          return response.data;
31      } catch (error) {
32          console.error(error);
33          throw new Error('erro de autenticação');
34      }
35  }
```

Fonte: Acervo do autor (2021)

A Figura 30, representa uma função onde será empregado a atualização das informações referente a leitura de uma ocorrência do aluno, na linha 26. A função assíncrona está recebendo um único parâmetro, que é qual a ocorrência que está aberta no momento.

Após o recebimento do parâmetro de ocorrência, é feito um *POST* na ocorrência em específico e retornará para o aplicativo como uma ocorrência lida, desta forma o responsável e o Colégio terão o controle sobre se a mensagem chegou ao destinatário.

4.4.5 Rotas

As rotas do protótipo foram todas montadas utilizando a biblioteca do React Native *StackNavigator*, que monta as telas uma em cima da outra criando um sistema de pilha de tela.

Quando usado o *StackNavigator* temos disponível uma função chamada de *goBack()*, presente na Figura 18 e referente a linha 14. O protótipo desempilha a tela e busca a tela anterior.

Desta forma o *StackNavigator* se torna muito prático para o sistema de rotas do protótipo, ele cria diversas camadas de sobrepostas e salva em memória, tornando o aplicativo muito rápido de ser acessado.

Portanto, *StackNavigator* se faz necessário a utilização de artifícios de funções globais e *Hooks* para que seja atualizada a tela no momento que estamos desempilhando as telas.

Figura 31 – Rota de Login

```
6   export function AuthRoutes(): ReactElement {
7       const { Navigator, Screen } = createNativeStackNavigator();
8
9       return (
10          <Navigator screenOptions={{ headerShown: false }} initialRouteName="Login">
11              <Screen name="Login" component={Login} />
12          </Navigator>
13      );
14  }
```

Fonte: Acervo do autor (2021)

Para a função relacionada as rotas do protótipo, possui 2 tipos de rotas a pública e a privada, temos o *AuthRoutes* que está presenta na Figura 31 e linha 6. Esta por definição é a primeira rota a ser chamada no protótipo, ela sempre irá chamar a tela de Login de usuário, e quando é feito o logout a mesma tela é chamada.

Todas as rotas recebem uma constante com os parâmetros *Navigator* e *Screen* como é visto na linha 7, eles vêm da função *createNativeStackNavigator*. *D*esta maneira as rotas são montadas como se fosse um arquivo de HTML, onde o Navigator irá agrupar todas as Screens.

Figura 32 – Rotas das telas logadas

```
19  export function StackRoutes(): ReactElement {
20      const { Navigator, Screen } = createNativeStackNavigator();
21
22      return (
23          <Navigator screenOptions={{ headerShown: false }} initialRouteName="Choose">
24              <Screen name="Choose" component={Choose} />
25              <Screen name="Child" component={Child} />
26              <Screen name="OccurrencesMenu" component={OccurrencesMenu} />
27              <Screen name="DisciplineDetail" component={DisciplineDetail} />
28              <Screen name="Occurrences" component={Occurrences} />
29              <Screen name="OccurenceDetail" component={OccurenceDetail} />
30              <Screen name="Invoice" component={Invoice} />
31              <Screen name="InvoiceDetail" component={InvoiceDetail} />
32              <Screen name="SerieMenu" component={SerieMenu} />
33              <Screen name="Grade" component={Grade} />
34              <Screen name="GradeDescriptive" component={GradeDescriptive} />
35              <Screen name="GradeAverage" component={GradeAverage} />
36              <Screen name="DisciplineDetailDescriptive" component={DisciplineDetailDescriptive} />
37              <Screen name="Sair" component={AuthRoutes} />
38          </Navigator>
39      );
40  }
```

Fonte: Acervo do autor (2021)

Todo o restante de telas do protótipo excluindo a tela de login, estão inclusas na Figura 32, essa função de StackRoutes da linha 19, compõe todas as telas que necessitam de login para acessá-las.

Figura 33 – Verificação das rotas

```
7   import {StackRoutes} from './app.routes';
8   import {AuthRoutes} from './auth.routes';
9
10  export function Routes(){
11      const { user } = useAuth();
12  return (
13      <NavigationContainer>
14          {user ? <StackRoutes /> : <AuthRoutes/>}
15      </NavigationContainer>
16      );
17  }
```

Fonte: Acervo do autor (2021)

As rotas são importadas em outro local como mostra a linha 7 e 8 da Figura 33, nesta Figura também temos uma constante de usuário, que busca da área de login.

Na linha 14 temos uma verificação onde é visto se o usuário está logado, utilizando uma condição ternária, que verifica ser existe algum *user*, essa informação é populada ou apagada quando o usuário faz login ou logout.

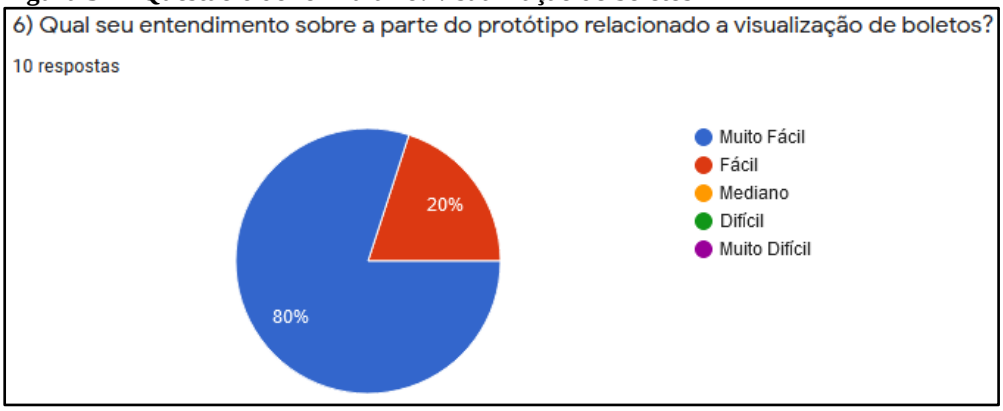
Caso exista um usuário na constante *user*, o protótipo redireciona para *StackRoutes* que possui todas as telas do sistema de logado, e se não existir *user* é encaminhado para login.

4.5 RESULTADOS DA PESQUISA

Para a pesquisa foram necessários dois dias de aplicações, entre buscar possíveis participantes e verificar se eles gostariam de participar do projeto.

Foi elencado algumas das questões aplicadas do formulário, para mostrar algumas amostrar de como foi a experiência em geral dos usuários utilizando em mãos o protótipo.

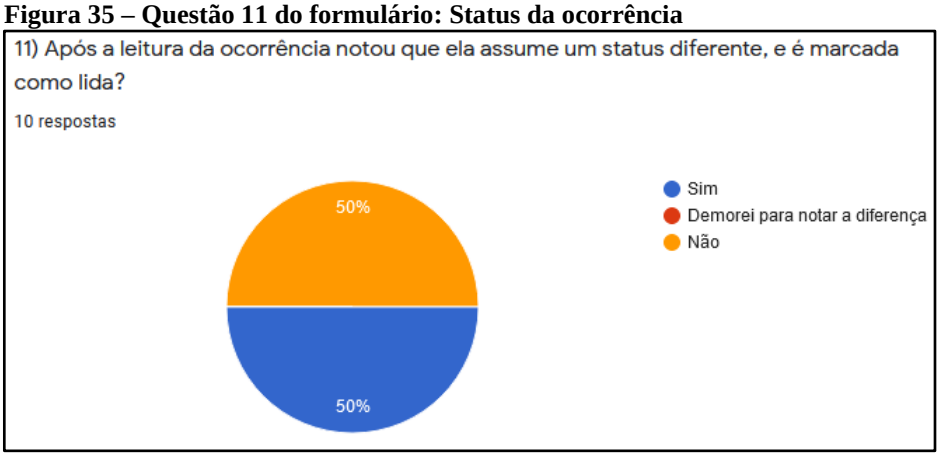
Figura 34 – Questão 6 do formulário: visualização de boletos



Fonte: Acervo do autor (2021)

Para o entendimento dos avaliados, foi muito fácil a questão de visualizar boletos, onde 80%, achou muito fácil, como aponta a Figura 34. Nesse aspecto a tela de boletos ajuda os usuários a acharem as informações necessárias para efetuar os seus pagamentos de parcelas.

Figura 35 – Questão 11 do formulário: Status da ocorrência

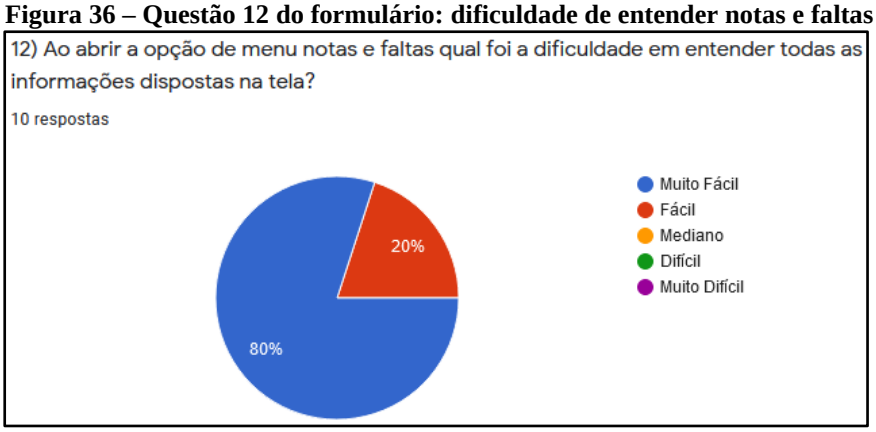


Fonte: Acervo do autor (2021)

Uma das perguntas que houve muita disparidade foi Na Figura 35, que entre as respostas ficou entre 50% sim e não.

A questão relacionada a ocorrência que assume um status de lido, onde a pessoa passava despercebido que a função trocava de cor assim como serviços de e-mail.

Figura 36 – Questão 12 do formulário: dificuldade de entender notas e faltas

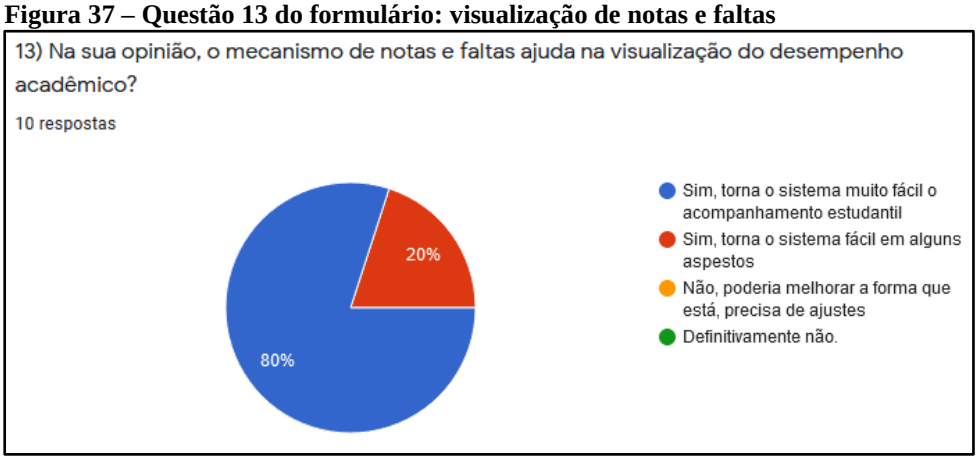


Fonte: Acervo do autor (2021)

A visualização das notas e faltas, no entendimento dos usuários segundo a Figura 36, foi muito fácil, onde muitos deles experientes em uso de dispositivos móveis foram clicando e lendo e analisando as informações dispostas.

Outros tiveram uma leve dificuldade, por ser uma ferramenta nova, ou não possuírem o manejo nos celulares, mas todos em geral se saíram muito bem e entenderam o contexto da aplicação.

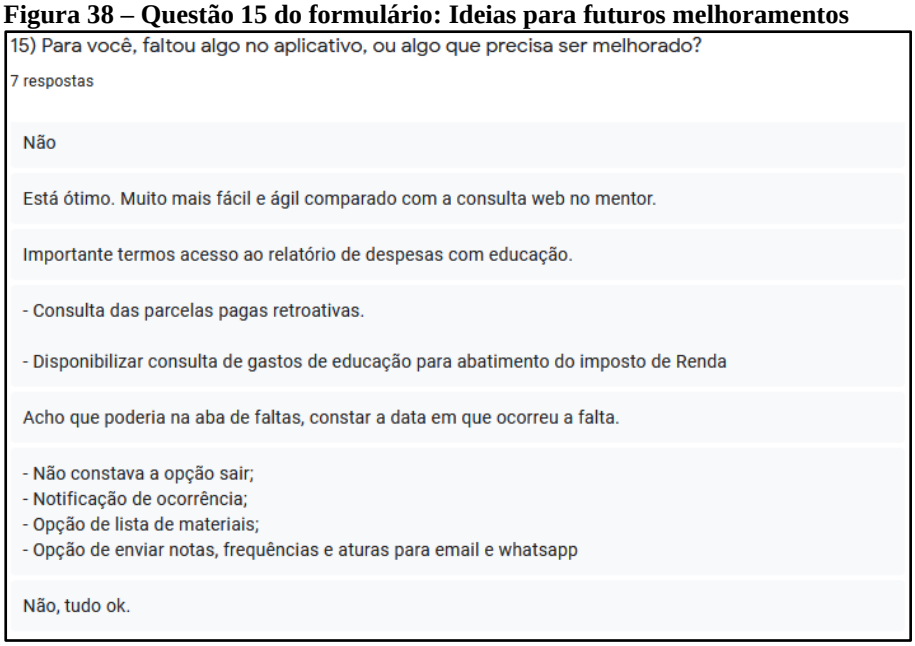
Figura 37 – Questão 13 do formulário: visualização de notas e faltas



Fonte: Acervo do autor (2021)

Outra questão elencada na Figura 37, é sobre o mecanismo para visualizar as notas e faltas, onde alguns colocaram como “torna fácil em alguns aspectos”. Onde presencialmente eles elencaram uma leve dificuldade para entender como o protótipo funciona.

Figura 38 – Questão 15 do formulário: Ideias para futuros melhoramentos



Fonte: Acervo do autor (2021)

Na Figura 38, apresenta muitas questões apontadas como interessantes para uma futura aplicação do projeto, também houve um problema com usuário onde o nome muito grande afetou o layout, este mesmo usuário deu diversas dicas para melhoramento do sistema.

Outras respostas falaram sobre a informação do informativo de imposto de renda, como uma forma de auxiliar os pais que precisam desta informação anualmente.

5. CONCLUSÃO

Este trabalho de conclusão de curso teve como principal objetivo o desenvolvimento de um protótipo de aplicativo em React Native, utilizando o mesmo código para as multiplataformas do mercado.

O intuito do projeto foi desenvolver um sistema, com usabilidade de praticidade para os responsáveis do Colégio Unidavi. Para desenvolver o protótipo foi utilizada diversas formas de programação desta, o que mais se destaca é a utilização dos componentes, uma forma de programar que traz praticidade, velocidade e uma incrível reutilização de códigos, onde facilita muito quem faz fazer manutenção em código.

Em relação aos objetivos geral, o protótipo conseguiu cumprir o seu propósito, trazendo todos os dados de um sistema acadêmico para o aplicativo, tornando a informação muito mais palpável, junto ao bolso, onde a outra forma é somente com uso de um computador utilizando navegadores web.

O projeto conseguiu cumprir todos os objetivos específicos propostos no início do trabalho, visto que ele iria desenvolver uma plataforma para acessar dados acadêmicos do aluno. Relacionado ao financeiro, notas e faltas e as ocorrências, onde todas essas informações são toda a vida acadêmica do estudante.

Todas as telas foram pensadas e desenhas utilizando experiência de usuário, onde existem diretrizes para aplicativos. Na questão de compatibilidade com diversos dispositivos, tamanhos e sistemas diferentes, nesse quesito, cada sistema possui padrões diferentes para a sua usabilidade. Neste quesito, o protótipo atende diferentes modelos de telas de celulares de IOS e Android.

Com a realização dos testes efetuados com o protótipo em uma pequena base de usuário notou-se que a demanda era necessária, onde que muitos dos usuários deram novas ideias, e a usabilidade também foi elogiada. Desta forma trouxe muita praticidade para os usuários que utilizam as ferramentas web, a migração das informações para o âmbito dos dispositivos móveis agradou a todos os usuários que testaram.

5.1 TRABALHOS FUTUROS

O próximo passo do protótipo, seria aplicar validações do aplicativo de login onde não foi aplicada nenhuma validação, caso o usuário tenha errado o login e validação caso não traga nenhuma informação ele apareça mensagem de erro.

Tornar acessíveis para todos os usuários utilizarem, desta forma teria um feedback maior, onde poderia abranger o aplicativo para todos os públicos e não somente a limitação da pesquisa efetuada.

Se faz necessário no futuro a avaliação das propostas ouvidas pela pesquisa onde algumas coisas elencadas poderiam servir para o aplicativo de forma que melhorasse ainda mais a vida de todos os usuários.

Outro ponto importante, seria possuir uma forma de notificação dos pais caso tenha uma nova inserção de ocorrência no sistema, desta forma o pai e responsável consegue ter um controle muito superior ao dado atualmente no protótipo.

REFERÊNCIAS

ANDRADE, Ana Paula. **O que é o React Native?**.2020. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-o-react-native. Acesso em: 04 out. 2021.

CALOMENO JUNIOR, Adil. **React Native: o que é, quais as funcionalidades e as vantagens desse framework**. 2020. Disponível em: https://ateliware.com/blog/react-native. Acesso em: 04 out. 2021.

CENTRO REGIONAL DE ESTUDOS PARA O DESENVOLVIMENTO DA SOCIEDADE DA INFORMAÇÃO, CETIC. **Educação e tecnologias no Brasil:** um estudo de caso longitudinal sobre o uso das tecnologias de informação e comunicação em 12 escolas públicas. 1. ed. São Paulo: Comitê Gestor da Internet no Brasil. 2016.

DATE, Christopher J. **Introdução a sistemas de bancos de dados.** 8. ed. Rio do Janeiro: Elsevier, 2004.

FLANAGAN, David. **JavaScript**: o guia definitivo. 6. ed. São Paulo: Bookman Companhia Editora Ltda, 2013.

GONÇALVES, Eduardo. **SQL. Uma abordagem para banco de dados Oracle.** São Paulo: Casa do Código, 2015. *E-book*.

HANASHIRO, Akira. **React: Conheça o Poder dos Hooks**. 2019. Disponível em: https://www.treinaweb.com.br/blog/react-conheca-o-poder-dos-hooks. Acesso em: 07 nov. 2021.

JABBAR, Gilshaan. **React Native Hooks & How To Use useState and useEffect**. 2020. Disponível em: https://gilshaan.medium.com/react-native-hooks-how-to-use-usestate-and-useeffect-3a10fd3e760c. Acesso em: 07 nov. 2021.

LECHETA, Ricardo R. **Google Android. aprenda a criar aplicações para dispositivos móveis com o android SDK.** 3. ed. São Paulo: Novatec, 2014.

MACHADO, Felipe N. F. **Banco de dados - projeto e implementação**. 3. ed. São Paulo: Saraiva, 2014.

MONTEIRO, João B. **Google Android. Crie aplicações para celulares.** São Paulo: Casa do Código, 2012.

NEGI, Manoj Singh. **UseState and useEffect explained**. 2019. Disponível em: https://medium.com/recraftrelic/usestate-and-useeffect-explained-cdb5dc252baf. Acesso em: 07 nov. 2021.

NOLETO, Cairo. **Linguagem de alto nível vs linguagem de baixo nível: definições e diferenças!** 2020. Disponível em: https://blog.betrybe.com/linguagem-de-programacao/linguagem-alto-e-baixo-nivel/. Acesso em: 04 out. 2021.

OLIVEIRA, Cláudio Luís Vieira; ZANETTI, Humberto Augusto Piovesana. **JavaScript Descomplicado**: programação para a web, iot e dispositivos móveis. São Paulo: Érica, 2020.

REACTNATIVE. **Core Components and APIs**. Disponível em: https://reactnative.dev/docs/next/components-and-apis. Acesso em: 04 out. 2021.

ROSSETTI, Micaela. **Componentes em React Native**. 2020. Disponível em: https://softdesign.com.br/blog/componentes-em-react-native. Acesso em: 04 out. 2021.

SILVEIRA, Guilherme; JARDIM, Joviane. **Swift. Programe para iPhone e iPad.** São Paulo: Casa do Código, 2015.

Smartphone Market Share. **IDC**. 2020. Disponível em: <https://www.idc.com/promo/smartphone-market-share/vendor>. Acessado em: 22 jun. 2020.
STEIL, Rafael. **IOS. Programe para iPhone e iPad.** São Paulo: Casa do Código, 2012.

TEXEIRA, Fabricio. **Introdução e boas prática em UX Design.** São Paulo: Casa do Código, 2012.

Uso da TI - Tecnologia de Informação nas Empresas. **FGV**. 2020. Disponível em: <https://eaesp.fgv.br/sites/eaesp.fgv.br/files/u68/fgvcia2020pesti-resultados.pdf>. Acesso em: 23 jun. 2020.

WU, Wenhao. **React Native vs Flutter, cross-platform mobile application frameworks**. Metropolia University of Applied Sciences. 2018. Disponível em: <https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1>. Acesso em: 22 jun. 2020.

APÊNDICE A – REVISÃO DO PROTÓTIPO

Neste item será abordado a forma que foi utilizado a pesquisa que foi aplicada. O público-alvo da pesquisa foi responsáveis de alunos matriculados no Colégio Unidavi.

A pesquisa teve como objetivo descobrir através dos usuários, a usabilidade do sistema, como o sistema se comporta e quais são possíveis melhorias que poderiam ser inclusas ao protótipo que os usuários sentiram falta.

No formulário foi incumbido dos usuários efetuarem diversas tarefas, a fim de ajudar os usuários a testarem todos os recursos do sistema e não esquecerem nenhum nos testes.

Foram efetuadas 15 questões, onde duas questões eram descritivas, que iria avaliar caso uma questão de múltipla escolha não estive em conformidade com as questões que pedia no título, e uma questão para apontar possíveis melhorias.

Outras 12 questões que irão avaliar a usabilidade bem como a forma que os dados estão sendo apresentados na tela.

Para a avaliação foi buscado dez pessoas que são responsáveis de alunos onde que para facilitar a busca foram usadas pessoas internas da própria Unidavi. No momento da aplicação foi apresentado a pesquisa de formulário impressa junto a um celular configurado com antecedência com o APK instalado no celular em questão.

Nos dias em que foi aplicado o questionário, foi feito uma abordagem perguntando primeiramente se eles aceitariam avaliar o aplicativo de forma anônima não iria ser capturado informações pessoais.

E caso eles aceitassem, poderiam avaliar conforme seu próprio critério independente do protótipo ser um produto interno ou não, a avaliação teria que ser imparcial.

Após o contexto inicial, foi mostrado que teriam 8 tarefas para serem executadas no aplicativo, e que não precisariam ser em sequência, excluído é claro o próprio login, como primeira tarefa. Depois de todas as tarefas concluídas foi orientado a responder o formulário de forma anônima.

APÊNDICE B – FORMULÁRIO DE PERGUNTAS

Protótipo de aplicativo para o Colégio da Unidavi

Projeto de protótipo de aplicativo para o Colégio da Unidavi, visando compreender se o aplicativo atende ao uso dos responsáveis.

*Obrigatório

Tarefas

1. Efetue Login com seu usuário e senha do sistema acadêmico;
2. Selecione o estudante de vínculo do seu usuário e que deseja visualizar as informações;
3. Visualize a frequência do curso atual;
4. Procure nos cursos a nota do 1° Trimestre;
5. Procure pelo detalhamento das notas e descrição deste trimestre da disciplina;
6. Visualize os boletos que estão em aberto deste semestre caso tenha;
7. Procure as ocorrências do ano atual 2021;
8. Busque pelo detalhamento da ocorrência;

1. 1) Você conseguiu executar com sucesso todos os passos descritos no roteiro do protótipo? *

   Marcar apenas uma oval.

   ◯ Sim
   ◯ Não

2. 2) Caso você não tenha executado com sucesso todos os passos, identifique o(s) e descreva o problema. *

3. 3) Para você, todas as informações de Login estavam bem claras? *

   Marcar apenas uma oval.

   - ( ) Sim
   - ( ) Parcialmente
   - ( ) Não

4. 4) Em relação ao menu, e sua de escolha do estudante, é de fácil entendimento? *

   Marcar apenas uma oval.

   - ( ) Sim
   - ( ) Parcialmente
   - ( ) Não

5. 5) Em relação ao menu principal, as informações estavam legíveis (claras e compreensíveis)? *

   Marcar apenas uma oval.

   - ( ) Sim
   - ( ) Parcialmente
   - ( ) Não

6. 6) Qual seu entendimento sobre a parte do protótipo relacionado a visualização de boletos? *

   Marcar apenas uma oval.

   - ( ) Muito Fácil
   - ( ) Fácil
   - ( ) Mediano
   - ( ) Difícil
   - ( ) Muito Difícil

7. 7) Na sua opinião, o menu de boleto auxilia na forma que você paga as parcelas mensalmente? *

   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Parcialmente
   - ◯ Não

8. 8) Você teve dificuldade para encontrar as ocorrências do semestre atual? *

   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Parcialmente
   - ◯ Não

9. 9) Caso você tenha tido dificuldade na visualização das ocorrência, descreva o problema. *

10. 10) Na sua opinião, o mecanismo de leitura de ocorrência auxilia na visualização destas informações? *

    Marcar apenas uma oval.

    - ◯ Sim, torna o sistema muito mais fácil de entender essas informações
    - ◯ Sim, torna o sistema um pouco mais fácil de entender essas informações
    - ◯ Não, ainda da forma como está, precisa de ajustes
    - ◯ No sistema de navegador é muito mais fácil

11. 11) Após a leitura da ocorrência notou que ela assume um status diferente, e é marcada como lida? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Demorei para notar a diferença
- ◯ Não

12. 12) Ao abrir a opção de menu notas e faltas qual foi a dificuldade em entender todas as informações dispostas na tela? *

Marcar apenas uma oval.

- ◯ Muito Fácil
- ◯ Fácil
- ◯ Mediano
- ◯ Difícil
- ◯ Muito Difícil

13. 13) Na sua opinião, o mecanismo de notas e faltas ajuda na visualização do desempenho acadêmico? *

Marcar apenas uma oval.

- ◯ Sim, torna o sistema muito fácil o acompanhamento estudantil
- ◯ Sim, torna o sistema fácil em alguns aspestos
- ◯ Não, poderia melhorar a forma que está, precisa de ajustes
- ◯ Definitivamente não.

14. 14) No seu ponto de vista, a apresentação das informações do protótipo está legível (clara e compreensível)? *

Marcar apenas uma oval.

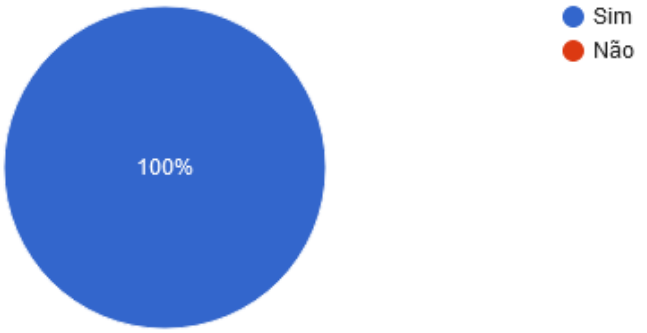
- ◯ Sim
- ◯ Parcialmente
- ◯ Não

15. 15) Para você, faltou algo no aplicativo, ou algo que precisa ser melhorado? *

APÊNDICE C – RESPOSTAS DO QUESTIONÁRIO

1) Você conseguiu executar com sucesso todos os passos descritos no roteiro do protótipo?

10 respostas



2) Caso você não tenha executado com sucesso todos os passos, identifique o(s) e descreva o problema.

0 resposta

Ainda não há respostas para esta pergunta.

3) Para você, todas as informações de Login estavam bem claras?

10 respostas

4) Em relação ao menu, e sua de escolha do estudante, é de fácil entendimento?

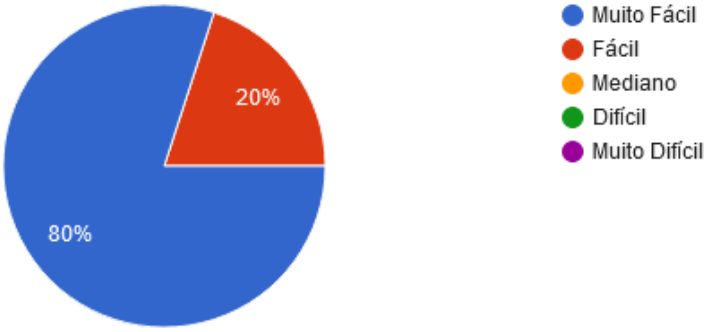
10 respostas

5) Em relação ao menu principal, as informações estavam legíveis (claras e compreensíveis)?

10 respostas

6) Qual seu entendimento sobre a parte do protótipo relacionado a visualização de boletos?

10 respostas



7) Na sua opinião, o menu de boleto auxilia na forma que você paga as parcelas mensalmente?

10 respostas

8) Você teve dificuldade para encontrar as ocorrências do semestre atual?

10 respostas

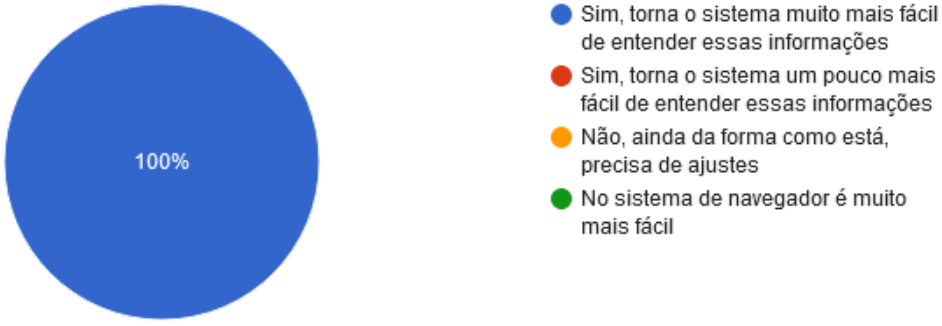
9) Caso você tenha tido dificuldade na visualização das ocorrência, descreva o problema.

0 resposta

Ainda não há respostas para esta pergunta.

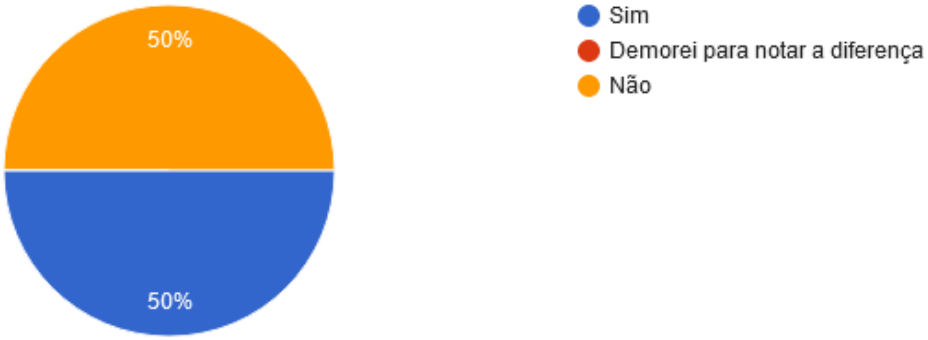
10) Na sua opinião, o mecanismo de leitura de ocorrência auxilia na visualização destas informações?

10 respostas



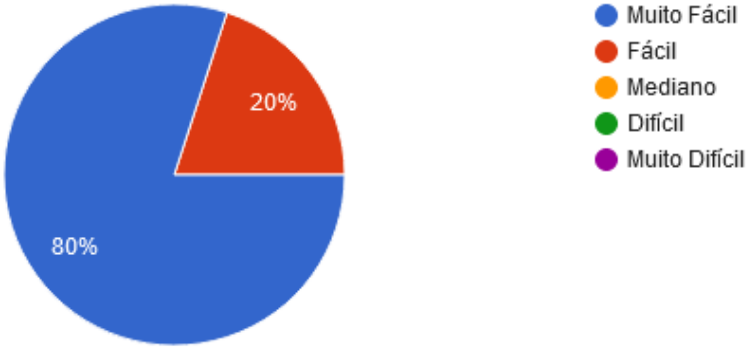
11) Após a leitura da ocorrência notou que ela assume um status diferente, e é marcada como lida?

10 respostas



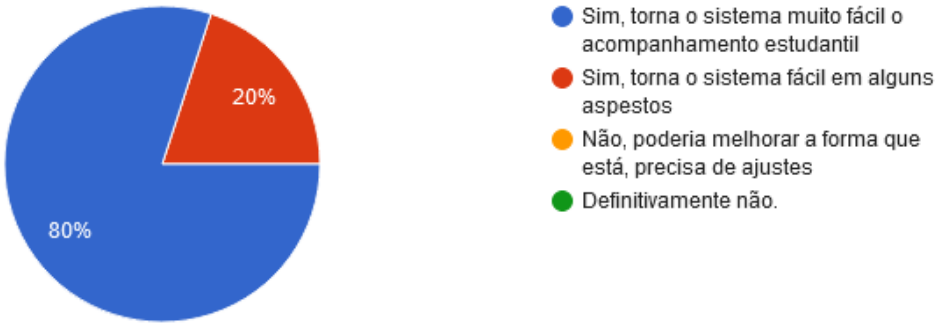
12) Ao abrir a opção de menu notas e faltas qual foi a dificuldade em entender todas as informações dispostas na tela?

10 respostas



13) Na sua opinião, o mecanismo de notas e faltas ajuda na visualização do desempenho acadêmico?

10 respostas



14) No seu ponto de vista, a apresentação das informações do protótipo está legível (clara e compreensível)?

10 respostas

15) Para você, faltou algo no aplicativo, ou algo que precisa ser melhorado?

7 respostas

Acho que poderia na aba de faltas, constar a data em que ocorreu a falta.

Está ótimo. Muito mais fácil e ágil comparado com a consulta web no mentor.

- Consulta das parcelas pagas retroativas.

- Disponibilizar consulta de gastos de educação para abatimento do imposto de Renda

- Não constava a opção sair;
- Notificação de ocorrência;
- Opção de lista de materiais;
- Opção de enviar notas, frequências e aturas para email e whatsapp

Não

Importante termos acesso ao relatório de despesas com educação.

Não, tudo ok.