UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE ENSINO SUPERIOR DO SERIDÓ
DEPARTAMENTO DE COMPUTAÇÃO E TECNOLOGIA
CURSO DE SISTEMAS DE INFORMAÇÃO
PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Aluna: Raquel Lima Fernandes

Relatório 02: Aplicativo para Gestão de Transporte Universitário “Tuni”

1 Visão Geral

O aplicativo "Tuni" está sendo desenvolvido para auxiliar motoristas e estudantes na gestão do transporte universitário. Ele oferece funcionalidades para acompanhar rotas e horários, possibilita a comunicação entre alunos e motoristas, e permite o envio de notificações relevantes para melhorar a experiência de transporte.

2 Estrutura de Pastas

A estrutura de pastas do aplicativo é organizada de maneira a facilitar a manutenção e a escalabilidade do código. Abaixo, está a descrição detalhada de cada diretório e seus arquivos:

2.1 Descrição das Pastas e Arquivos

- **assets/**
    - **fonts/**: Este diretório deve conter todas as fontes personalizadas utilizadas na aplicação.
    - **images/**: Este diretório deve conter todas as imagens utilizadas na aplicação.
- **src/**
    - **components/**
        - **CustomerTabBar.js**: Componente personalizado para a barra de navegação por abas.
    - **data/**
        - **users.js**: Arquivo contendo dados fictícios dos usuários, incluindo

informações sobre mensagens, notificações, turnos, reservas, tipos de usuários e instituições.

- **navigation/**
  - **AppStack.js**: Configuração da pilha de navegação das telas principais do aplicativo.
  - **AppTabs.js**: Configuração da navegação por abas do aplicativo.
  - **AuthStack.js**: Configuração da pilha de navegação das telas de autenticação.
  - **RootNavigator.js**: Arquivo principal de navegação que integra todas as pilhas e abas de navegação.
- **screens/**
  - **App/**
    - **ChatDetails.js**: Tela com os detalhes de um chat específico.
    - **Chats.js**: Tela com a lista de conversas.
    - **Clocks.js**: Tela com os horários das rotas.
    - **Feed.js**: Tela com o feed de informações, incluindo rotas e serviços.
    - **Notifications.js**: Tela com as notificações recebidas pelos usuários.
    - **ProfileDetails.js**: Tela com os detalhes do perfil do usuário.
    - **ScheduleDetails.js**: Tela com os detalhes de um agendamento específico.
    - **Schedules.js**: Tela com a lista de agendamentos.
    - **SearchChats.js**: Tela de pesquisa de chats.
  - **Auth/**
    - **Login.js**: Tela de login do usuário.

- **App.js**: Arquivo principal da aplicação, onde a aplicação é inicializada.
- **package.json**: Arquivo de configuração do projeto, contendo dependências, scripts e informações gerais sobre o projeto.

3 Descrição dos Components

3.1 Componente CustomerTabBar

A tela CustomerTabBar é um componente personalizado de barra de navegação por

abas, implementada utilizando React Native e Expo. Este componente fornece uma interface interativa para os usuários navegarem entre diferentes seções do aplicativo "Tuni".

3.1.1 Estrutura do Código e Imagem

CustomerTabBar

3.1.2 Importações

- **React**: Importa o módulo React para usar componentes funcionais.
- **React Native Components**: Importa componentes nativos como View, Text, StyleSheet para a construção da interface e TouchableOpacity.
- **Icones**: Importa ícones de MaterialCommunityIcons da biblioteca @expo/vector-icons para os ícones da barra de navegação.
- **state**: Representa o estado da navegação, incluindo informações sobre as rotas atuais e a aba ativa.
- **descriptors**: Descritores das rotas, fornecendo informações adicionais sobre cada rota.
- **navigation**: Objeto de navegação para controlar a navegação entre abas.

3.1.3 Mapeamento das Rotas

O componente mapeia sobre *state.routes* para criar uma aba para cada rota definida. Determina o label de cada aba com base nas opções da rota ou no nome da rota.

3.1.4 Funções de Navegação

- **onPress:** Navega para a aba correspondente ao ser pressionada.
- **onLongPress**: Emite um evento quando a aba é pressionada por um longo período.

3.1.5 Icones Dinâmicos

Define os ícones para cada rota, alternando entre a versão preenchida e contornada com base no estado de foco.

3.1.6 Estilização Condicional

Estiliza a aba com um fundo e cor diferentes se estiver focada e adiciona uma borda

arredondada e padding ao ícone da aba.

3.2 Descrição da Navigation

3.2.1 Navegação AppStack

O componente AppStack é uma estrutura de navegação baseada em pilha (stack navigator) do React Navigation, que gerencia a navegação entre diferentes telas do aplicativo "Tuni". Este componente define as várias telas e suas respectivas opções de cabeçalho, proporcionando uma navegação fluida entre as seções principais do aplicativo.

3.2.1.1 Estrutura do Código

AppStack.js.pdf

3.2.1.2 Importações

- **React**: Importa o módulo React e o hook useState.
- **React Navigation**: Importa createNativeStackNavigator do React Navigation para criar a navegação baseada em pilha.
- **Componentes do Aplicativo**: Importa componentes das telas do aplicativo, incluindo AppTabs, SearchChats, ChatDetails, Schedules, ScheduleDetails, e ProfileDetails.

3.2.1.3 Definição do Stack Navigator

- **Stack**: Cria uma instância do stack navigator usando createNativeStackNavigator.
- **Screen Options**: Define as opções de estilo para o cabeçalho de cada tela na pilha, incluindo a cor de fundo (headerStyle), visibilidade da sombra do cabeçalho (headerShadowVisible), e a cor do texto do cabeçalho (headerTintColor).

3.2.1.4 Telas Definidas na Pilha

- **Feed**: A tela principal, renderizada pelo componente *AppTabs*.
- **SearchChats**: Tela de pesquisa de conversas.
- **ChatDetails**: Tela de detalhes de uma conversa específica.
- **Schedules**: Tela de horários.

- **ScheduleDetails**: Tela de detalhes de um horário específico.
- **ProfileDetails**: Tela de detalhes do perfil do usuário.

3.2.2 Navegação AppTabs

O componente AppTabs é um elemento crucial do aplicativo "Tuni", fornecendo a navegação principal através de abas na parte inferior da tela. Ele usa o createBottomTabNavigator do React Navigation para gerenciar a navegação entre diferentes seções do aplicativo, como Feed, Horários, Notificações e Conversas.

3.2.2.1 Estrutura do Código

AppTabs.js.pdf

3.2.2.2 Importações

- **React**: Importa o módulo React e o hook useState.
- **React Native Components**: Importa componentes nativos como Text, View, StyleSheet, TouchableOpacity, Image, e Modal.
- **Ícones**: Importa ícones da biblioteca expo/vector-icons para uso nos botões e na navegação.
- **React Navigation**: Importa createBottomTabNavigator e useNavigation para criar a navegação por abas.
- **Moment.js**: Importa a biblioteca Moment.js para manipulação de datas, configurada para o idioma português.
- **Calendário**: Importa o componente *Calendar* da biblioteca react-native-calendars.

3.2.2.3 Definição do Navigator

- **Tab**: Cria uma instância do tab navigator usando createBottomTabNavigator.
- **Estados**: Usa useState para gerenciar o estado do modal e a imagem de perfil do usuário atual.

3.2.2.4 Configuração das Telas

Feed: Define a tela inicial com um cabeçalho personalizado, incluindo a imagem de perfil do usuário e um ícone de calendário.
Clocks: Define a tela de horários com um botão para adicionar um novo agendamento.
Notifications: Define a tela de notificações com botões de filtros.
Chats: Define a tela de conversas com um botão para pesquisa.

3.2.3 Navegação HootNavigator

O componente RootNavigator é um componente principal de navegação dentro do aplicativo React Native, configurado utilizando a biblioteca @react-navigation/native. Ele serve como o ponto de entrada da navegação para o aplicativo, definindo a estrutura de navegação de pilha (stack navigation) e gerenciando as telas principais do app através das stacks AppTabs e AppStack.

3.2.3.1 Estrutura do Código

RootNavigation.js.pdf

3.2.3.2 Importações e Dependências

- **React**: A biblioteca principal para criar componentes funcionais em React Native.
- **NavigationContainer**: Um contêiner obrigatório fornecido pelo @react-navigation/native, que engloba toda a estrutura de navegação do aplicativo.
- **createNativeStackNavigator**: Função da biblioteca @react-navigation/native-stack que cria um objeto de navegação em pilha, permitindo a transição entre telas com uma navegação estilo "stack".
- **AppTabs** e **AppStack**: Componentes que encapsulam as principais rotas e telas do aplicativo. Estes são passados como componentes para as Screens dentro do Stack.Navigator.

3.2.3.3 Componente RootNavigator

O componente RootNavigator é uma função que retorna um JSX contendo o

NavigationContainer, que por sua vez contém o Stack.Navigator. Dentro do Stack.Navigator, duas Screens são definidas: AppTabs e AppStack.

- **AppTabs**: Representa uma navegação por abas na parte inferior do aplicativo.
- **AppStack**: Contém outras telas que requerem navegação empilhada e são acessíveis de forma hierárquica a partir de AppTabs.

3.2.3.4 Opções de Navegação

Ambos AppTabs e AppStack têm a opção ‘headerShown: false’ configurada, o que significa que o cabeçalho padrão fornecido pela navegação em pilha não será exibido para estas telas.

3.3 Feed Screen

O componente Feed é uma tela que exibe as rotas do dia em um mapa interativo e oferece uma lista de serviços adicionais disponíveis para os usuários. Ele combina elementos de UI como mapas, listas horizontais, ícones, e alertas, oferecendo uma interface rica e funcional.

3.3.1 Estrutura do Código e Imagem

Feed.js.pdf

3.3.2 Importações e Dependências

- **React**: A biblioteca principal usada para construir a interface.
- **View, Text, FlatList, StyleSheet, TouchableOpacity, Alert**: Componentes do React Native para estrutura e interatividade da interface.
- **MaterialCommunityIcons**: Uma biblioteca de ícones usada para exibir ícones personalizados.
- **MapView**: Um componente da biblioteca react-native-maps que integra um mapa interativo, permitindo a exibição de rotas e pontos de interesse.

3.3.3 Layout e Renderização

- **Mapa com Instituições**: O primeiro bloco renderiza um mapa interativo usando o componente MapView. A área do mapa é delimitada pela região inicial configurada

com latitude e longitude específicas de (Caicó-RN).

- **Lista de Serviços**: O segundo bloco renderiza uma lista horizontal de serviços usando FlatList. Cada item da lista é um botão (TouchableOpacity) que, ao ser pressionado, exibe um alerta com o nome do serviço selecionado.

3.4 Clocks Screen

O componente Clocks foi criado para exibir uma lista de estudantes organizados por turnos (manhã, tarde e noite). O componente permite que o usuário filtre a exibição dos estudantes com base no tipo de horário (Ida, Volta ou Todos) e navegue para uma tela de detalhes de agendamento.

3.4.1 Estrutura do Código e Imagem

Clocks.js.pdf

3.4.1 Importações e Dependências

- **React**: A biblioteca principal usada para construir a interface.
- **View, Text, FlatList, StyleSheet, TouchableOpacity, Alert**: Componentes do React Native para estrutura e interatividade da interface.
- **MaterialCommunityIcons**: Uma biblioteca de ícones usada para exibir ícones personalizados.

3.4.2 Funções Utilizadas

- **groupStudentsByShif**t: Agrupa os estudantes com base nos turnos e horários de ida/volta.
- **filterStudents**: Filtra a lista de estudantes com base no filtro selecionado (Todos, Ida, Volta).
- **renderStudentItem**: Renderiza os avatares dos estudantes dentro dos cards.
- **renderShiftSection**: Cria a estrutura visual para cada turno, exibindo os horários e os estudantes associados.
- **FilterBar**: Renderiza a barra de filtros, permitindo que o usuário altere a visualização.

3.5 Notifications Screen

O componente Notifications do aplicativo Tuni foi desenvolvido para exibir as notificações recebidas pelos usuários, permitindo a filtragem dessas notificações com base em diferentes critérios, como "Todas" e "Não lidas". A tela proporciona uma interface limpa e intuitiva, onde os usuários podem visualizar informações importantes relacionadas ao serviço de transporte.

3.5.1 Estrutura do Código e Imagem

3.5.2 Importações

- **React:** Biblioteca principal utilizada para a construção do componente.
- **Componentes do React Native:**
- useState, useEffect: Hooks utilizados para gerenciar o estado e os efeitos colaterais no componente.
- **StyleSheet, Text, View, ScrollView, StatusBar, Image, TouchableOpacity:** Componentes usados para estruturar e estilizar a interface.
- **dataBase:** Um arquivo local que contém as informações dos usuários, incluindo notificações e dados relacionados.

3.5.3 Layout e Renderização

- **Cabeçalho de Filtro:** Inclui botões para alternar entre "Todas" e "Não lidas". O botão ativo é destacado visualmente para indicar o filtro em uso.
- **Corpo das Notificações:** As notificações são exibidas em cartões individuais (notificationCard), que contêm a imagem do usuário, nome, texto da notificação e o horário. Se nenhuma notificação estiver disponível para o filtro selecionado, uma mensagem padrão é exibida.

3.6 Chats Screen

O componente Chats no aplicativo Tuni é projetado para exibir uma lista de conversas recentes, permitindo que os usuários visualizem, pesquisem e interajam com suas mensagens. A interface é simples e eficaz, proporcionando uma navegação fluida entre as diferentes conversas. Além disso, o componente inclui uma função de longo pressionamento para excluir conversas, aumentando a capacidade de gerenciamento das interações.

3.6.1 Estrutura do Código e Imagem

Chats.js.pdf

3.6.2 Importações

- **React e React Native:** useState, useEffect: Hooks para gerenciar o estado e efeitos colaterais.
- **Text, View, StatusBar, TouchableOpacity, Image, ScrollView, Alert, StyleSheet:** Componentes usados para estruturar, estilizar e interagir com a interface.
- **Icones:** Feather (Icon) e MaterialIcons (MIcon): Bibliotecas de ícones usadas para adicionar ícones de pesquisa e status de visualização.
- **useNavigation:** Hook do React Navigation para navegação entre telas.
- **usersData:** Dados fictícios importados que contêm informações sobre os usuários e suas mensagens. Estrutura do Código e Imagem

3.6.3 Layout e Renderização

- **Botão de Pesquisa:** Um botão de pesquisa no topo da tela permite que o usuário navegue para uma tela de pesquisa de conversas. Ele exibe um ícone de lupa e um texto indicativo.
- **Lista de Conversas:** Cada conversa é exibida como um cartão contendo a imagem do usuário, nome, hora da última mensagem, e uma prévia da mensagem. A conversa pode ser clicada para acessar os detalhes completos ou pressionada longamente para ser removida.
- **Estado de Digitação e Visualização:** Se o usuário estiver digitando, o texto "digitando..." é exibido. Além disso, se a mensagem foi enviada pelo usuário, ícones de confirmação indicam se a mensagem foi visualizada ou não.

3.6 ProfileDetails Screen

O componente ProfileDetails exibe as informações detalhadas do perfil do usuário, incluindo a imagem de perfil, nome, e-mail e instituição. Ele também oferece opções para acessar diferentes seções do aplicativo, como atualizar o perfil, visualizar conversas, consultar agendamentos e ajustar configurações. Além disso, o componente inclui uma opção de logout.

3.6.1 Estrutura do Código e Imagem

ProfileDetails.js.pdf

3.6.2 Importações e Dependências

- **React e React Native:** useState, useEffect: Hooks para gerenciamento de estado e efeitos colaterais.
- **View, Text, Image, StyleSheet, TouchableOpacity, ScrollView: C**omponentes usados para estruturação e estilização da interface.
- **useNavigation:** Hook do React Navigation para gerenciar a navegação entre telas.
- **useFonts:** Hook da Expo Fonts para carregar fontes personalizadas.
- **FontAwesome (FAIcons) e MaterialCommunityIcons (MCI):** Bibliotecas de ícones para adicionar ícones de interface.

3.6.3 Layout e Renderização

- **Cabeçalho da Tela:** headerLeft: Adiciona um botão de retorno com um ícone de seta para a esquerda e um título "Perfil" no cabeçalho da tela.
- **Informações do Perfil:** ProfileHeader: Contém a imagem do perfil, nome do usuário, e-mail e instituição, centralizados na tela. profileImage: Exibe a imagem do perfil com um estilo circular. userName, userEmail, userInstitution: Exibem o nome, e-mail e instituição do usuário, estilizados com a fonte personalizada.
- **Botões de Navegação:** editButton: Cada botão permite navegar para diferentes seções do aplicativo, como atualizar o perfil, visualizar conversas, consultar agendamentos e acessar configurações. logoutButton: Oferece uma opção para o logout, com um ícone de saída.
- **Separador:** separator: Adiciona uma linha de separação antes do botão de logout, melhorando a organização visual dos elementos.

3.7 Shedules Screen

O componente Schedules é responsável por gerenciar e exibir os horários de ida e volta disponíveis para um estudante, permitindo que ele selecione e confirme um horário para suas reservas. Ele também configura a navegação e a interface do usuário para a tela de agendamentos.

3.7. 1 Estrutura do Código e Imagem

Shedules.js.pdf

3.7.2 Funcionalidades

- **Seleção de Horários:** Permite ao usuário selecionar horários de ida e volta. Destaca o horário selecionado para melhor visualização.
- **Confirmação de Reserva:** O usuário pode confirmar sua seleção de horários. Exibe um alerta confirmando a reserva ou notificando se horários não foram selecionados.
- **Configuração da Navegação:** Configura o título da tela, o botão de retorno e o botão de menu na barra de navegação.
- **Estilo e Layout:** Utiliza um layout responsivo e estilizado para exibir horários de ida e volta. Aplica estilos consistentes para botões e textos, com suporte a estados selecionados e não selecionados.

3.8 ClocksDetails Screen

ClocksDetails.js.pdf

3.9 ChatsDetails Screen

ChatDetails.js.pdf

3.10 SearchChats Screen

SearchChat.js.pdf