Mariana Medeiros Ruddy Santos

xBus

Aplicativo do motorista

PROJETO FINAL

DEPARTAMENTO DE INFORMÁTICA
Programa de Graduação em Engenharia da Computação

Rio de Janeiro
Junho de 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Mariana Medeiros Ruddy Santos

xBus

Aplicativo do motorista

Relatório de Projeto Final

Relatório de Projeto Final, apresentado ao Programa de Engenharia da Computação, do Departamento de Informática da PUC-Rio como requisito parcial para a obtenção do titulo de Bacharel em Engenharia da Computação.

Orientador: Prof. Sergio Lifschitz

Rio de Janeiro
Junho de 2023

Mariana Medeiros Ruddy Santos

Graduando em Engenharia da Computação na PUC - Rio

Ficha Catalográfica

Medeiros Ruddy Santos, Mariana

xBus / Mariana Medeiros Ruddy Santos; orientador: Sergio Lifschitz. – 2023.

44 f: il. color. ; 30 cm

Projeto Final - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Transporte unversitário. 2. Desenvolvimento flutter. 3. Desenvolvimento mobile. 4. Firebase. I. Lifschitz, Sergio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Resumo

Medeiros Ruddy Santos, Mariana; Lifschitz, Sergio. **xBus**. Rio de Janeiro, 2023. 44p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O projeto xBus consiste em um sistema de transporte para alunos e professores da PUC Rio. O projeto envolveu o desenvolivmento de um sistema web para o administrador, um aplicativo para o motorista e um aplicativo para o passageiro.

Palavras-chave

Transporte unversitário; Desenvolvimento flutter; Desenvolvimento mobile; Firebase.

Sumário

1 Introdução **7**

2 Situação Atual **8**

3 Objetivos do Trabalho **10**

4 Atividades realizadas **11**
4.1 Estudos preliminares 11

5 Projeto e especificação do sistema **12**
5.1 Lista de requisitos 12
5.2 Lista de requisitos funcionais 12
5.3 Lista de requisitos não-funcionais 13
5.4 Banco de dados 13
5.5 Banco relacional vs banco não relacional 13
5.6 Diagrama de sequência 20
5.7 Sitemap 21
5.8 Wireframe 22
5.9 Casos de uso 27

6 Implementação e avaliação **31**
6.1 Desenvolvimento 32
6.2 Arquitetura 32

7 Considerações finais **34**

8 Referências bibliográficas **35**

9 Anexos **36**
9.1 Manual do usuário 36

Lista de figuras

Figura 2.1 Modais de transporte mais utilizados em 2015 9
Figura 2.2 Atributos do serviço 10

Figura 5.1 Modelagem do banco de dados 15
Figura 5.2 Diagrama de sequências 20
Figura 5.3 Diagrama de sequências da plataforma web 21
Figura 5.4 Sitemap do app do motorista 22
Figura 5.5 Wireframe login do app do motorista 23
Figura 5.6 Wireframe cadastro do app do motorista 24
Figura 5.7 Wireframe página inicial do app do motorista 25
Figura 5.8 Wireframe página viagem atual 26
Figura 5.9 Wireframe página validação de ticket 27
(a) Câmera 27
(b) Sucesso 27

Figura 9.1 Login 36
Figura 9.2 Cadastro 37
Figura 9.3 Página inicial 38
(a) Nenhuma rota selecionada 38
(b) Rota, horário e veículo selecionados 38
Figura 9.4 Página da viagem em andamento 39
Figura 9.5 Página de ticket escaneado com sucesso 40
Figura 9.6 Página inicial 41
(a) Nenhuma rota disponível 41
(b) Rota, horário e veículo selecionados 41
(c) Rota, horário e veículo selecionados 41
Figura 9.7 Página de registro com documento inválido 42
Figura 9.8 Erro na senha 43
(a) Cadastro 43
(b) Login 43
Figura 9.9 Página de ticket escaneado com erro 44

Lista de tabelas

Tabela 5.1	Caso de uso login do motorista	28
Tabela 5.2	Caso de uso cadastro de senha	29
Tabela 5.3	Caso de uso iniciar viagem	30
Tabela 5.4	Caso de uso validar ticket	31

1
Introdução

Apresento, neste documento, o projeto xBus, que visa melhorar como alunos e funcionários da PUC-Rio se deslocam da e para a universidade, oferecendo um serviço de transporte por meio de uma solução completa de software, similar a sistemas já existentes em ambientes universitários nos Estados Unidos e no interior de São Paulo, por exemplo. Usando, assim, tecnologia para trazer mais conforto e segurança para seus usuários e ajudando a diminuir a quantidade carros na cidade.

Tal solução contempla um aplicativo móvel para os passageiros, um aplicativo móvel para o motorista e um site para controle da qualidade de serviço oferecido e acompanhamento com relatórios administrativos e operacionais. Esses relatórios fornecem dados para melhorar o serviço, podendo, por exemplo, ajudar a estabelecer os horários das rotas, mudança do trajeto de rotas ou uso veículos com diferentes capacidades, conforme a demanda dos usuários.

Pretende-se trazer, com esse projeto, mais segurança, conforto, previsibilidade e pontualidade para os alunos e funcionários da PUC Rio, além de diminuir a quantidade de automóveis na rua, contribuindo, assim, com a redução dos problemas causados pelo transporte individual.

A parte de implementação, por possuir uma maior complexidade, foi divida em três partes bem definidas: web, aplicativo do passageiro e aplicativo do motorista, sendo este último o foco deste documento. O web, sistema usado pelo administrador para gerenciamento, foi desenvolvido usando Django. Já os aplicativos do passageiro e do motorista, usados durante o fluxo da viagem, foram desenvolvidos usando o framework *flutter* e a linguagem *dart*. A integração dessas três partes se faz através do *Firebase*.

O restante deste documento está estruturado da seguinte forma: na seção 2, é feita uma contextualização acerca do problema; na seção 3, são

apresentados os objetivos deste trabalho; na seção 4, são descritas as atividades realizadas durante o desenvolvimento do projeto; na seção 5, são abordadas as especificações do sistema; na seção 6, discute-se o desenvolvimento e arquitetura do sistema; na seção 7, são apresentadas as considerações finais; e na seção 8, são listadas as referências bibliográficas consultadas ao longo do trabalho. Por fim, na seção 9, encontra-se o manual do usuário, disponibilizado como anexo.

2
Situação Atual

Temos hoje, na PUC-Rio, alunos e funcionários que utilizam, para ir e voltar da faculdade, transporte público, carro particular, aplicativos de carro (como uber), entre outros. Para aqueles que utilizam o carro particular, existe um estacionamento para alunos e funcionários. Porém, esse estacionamento costuma ter filas grandes em horário de pico. Entre os usuários do transporte público, há medo da violência (infelizmente) comum na cidade do Rio de Janeiro. Além de falta de previsibilidade e pontualidade dos horários dos ônibus. Uma solução para acabar ou diminuir esses problemas poderia ser o uso de aplicativos de carros, mas isso implicaria em um custo de deslocamento muito mais elevado.

Existem soluções, como o aplicativo Moovit (LIRA, 2021), disponível para iOS e Android. O app visa facilitar o trajeto das pessoas com informações de transportes públicos e outras opções de mobilidade em tempo real. Mais recentemente, em 2017, a empresa lançou suas soluções MaaS (Mobilidade como Serviço), para planejamento, operação e avaliação de trajetos, congestionamentos, número de passageiros e outros dados através de inteligência artificial, para melhorar as condições dos serviços de transporte e auxiliar empresas a encontrarem os melhores trajetos para seus colaboradores. Porém,

suas previsões de horários ainda não são muito exatas.

Há, também, alguns condomínios na cidade, que oferecem para seus moradores, transporte exclusivo. Uma particularidade desses sistemas é que não há necessidade de funcionalidades como compra de saldo e ticket, por exemplo. Além de, oferecerem uma grade de horários limitada, geralmente diária ou semanal, sem muito espaço para adaptação conforme a demanda.

Segundo pesquisa do professor da PUC-Rio Sergio Lifschitz (figura 2.1) com alunos e funcionários da universidade, em 2015, o modal mais utilizado era o ônibus, seguido pelo carro, depois metrô, bicicleta, a pé e moto. Ainda durante essa pesquisa, perguntou-se aos entrevistados quais os atributos necessários para adotar o serviço e o resultado, visto na figura 2.2, indica previsibilidade e segurança no topo dos pedidos, seguido por acesso ao aplicativo, conforto, sustentabilidade, reserva de assento e wi-fi, nessa ordem.

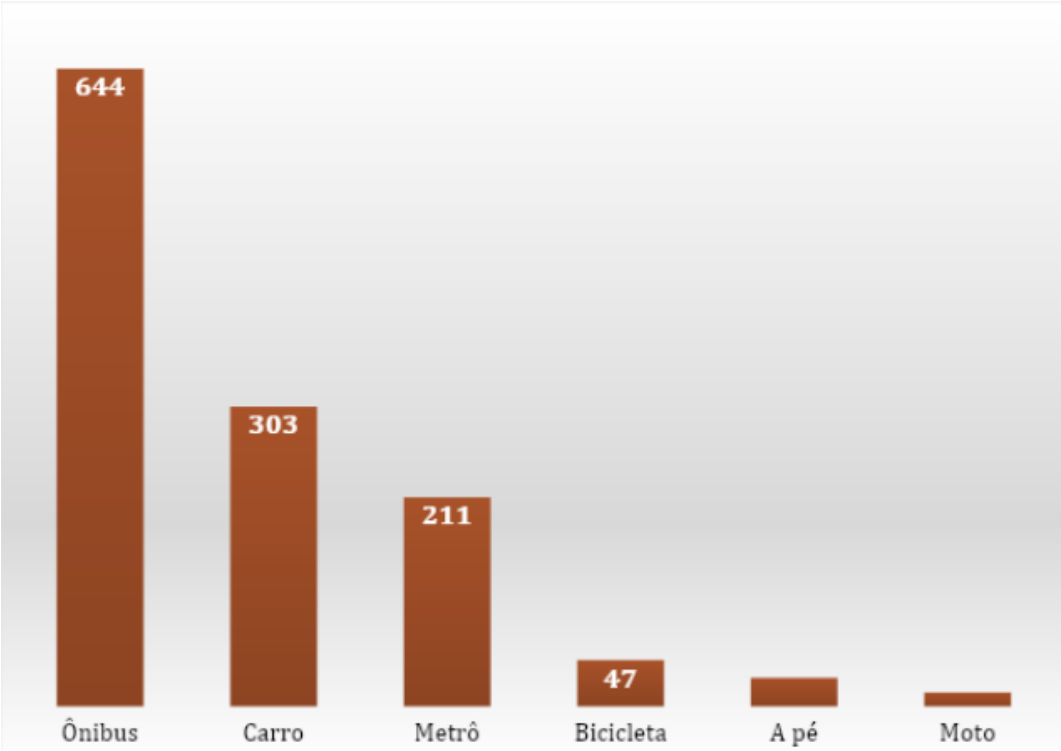


Figura 2.1: Modais de transporte mais utilizados em 2015

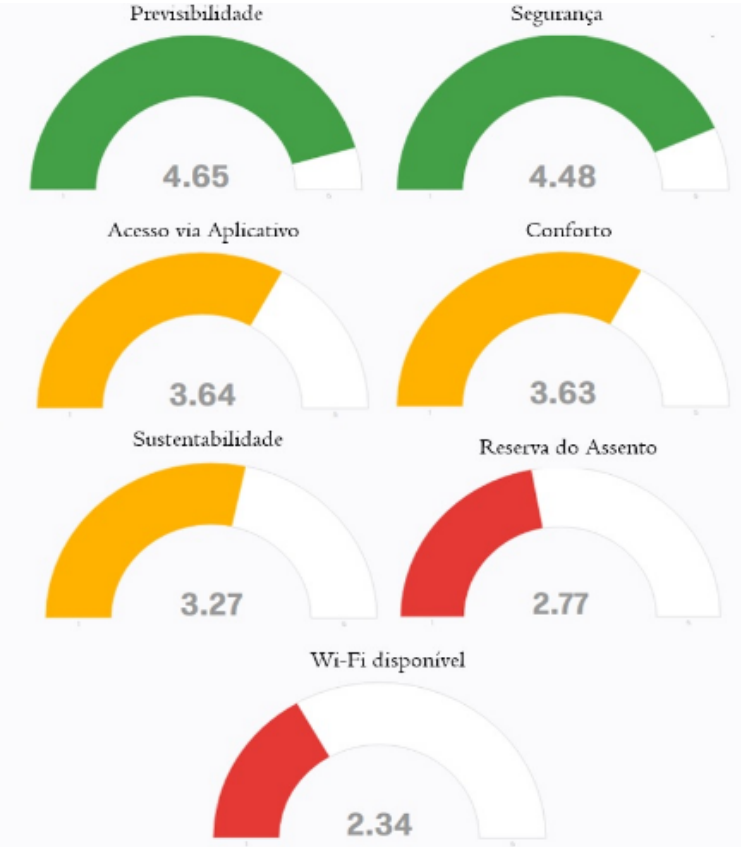


Figura 2.2: Atributos do serviço

3
Objetivos do Trabalho

Ao longo deste trabalho, pretende-se aplicar os conhecimentos adquiridos durante o curso de Engenharia da Computação. O objetivo principal é colocar em prática os conceitos teóricos e habilidades práticas adquiridas ao longo do curso, de forma a consolidar o aprendizado e ampliar a compreensão sobre o desenvolvimento de projetos na área.

Como se trata de uma prova de conceito, todas as etapas de um projeto serão percorridas, desde o planejamento até a conclusão. Isso envolve a definição clara dos requisitos, o desenvolvimento da solução, os testes e a validação dos resultados obtidos. A intenção é ter uma visão abrangente do processo de desenvolvimento de software e colocar em prática as metodologias

e boas práticas aprendidas ao longo do curso.

Além disso, o projeto em questão envolve a colaboração em equipe, uma vez que é dividido em três frentes distintas. Será necessário trabalhar em conjunto, compartilhando conhecimentos, trocando ideias e alinhando as atividades de cada membro da equipe. O gerenciamento eficiente e a comunicação clara são importantes para a realização do projeto.

Para alcançar os objetivos propostos, será necessário revisitar ferramentas e tecnologias já aprendidas, mas que não são utilizadas no dia a dia. Isso inclui explorar recursos e funcionalidades específicas dessas ferramentas, aprofundar o conhecimento e adquirir maior fluência no seu uso. Além disso, será necessário aprender novas ferramentas e tecnologias relevantes para o desenvolvimento do projeto, ampliando assim o conjunto de habilidades e competências técnicas.

O projeto também tem como objetivo aprimorar a capacidade de estipular tarefas e criar cronogramas realistas. Isso implica na habilidade de identificar e priorizar as atividades, estabelecer prazos adequados e gerenciar o tempo de forma eficiente. O cumprimento dos prazos e o monitoramento do progresso serão fundamentais para o bom andamento do projeto e para alcançar os resultados esperados.

4
Atividades realizadas

4.1
Estudos preliminares

Primeiramente, desenvolveu-se o banco de dados, diagrama de sequência, lista de requisitos, sitemap, wireframe e casos de uso, detalhados na seção 5. A partir dessas informações, deu-se início ao desenvolvimento do aplicativo.

Para o desenvolvimento do aplicativo, foi necessário adquirir conheci-

mentos de desenvolvimento mobile com *Flutter* e *dart* e conhecimentos sobre o *Firebase*. Para isso, segui o tutorial *Flutter & Firebase App Tutorial* (FLUTTER..., 2019).

Ao final desse tutorial, foi possível ter conhecimentos básicos de desenvolvimento Flutter e de como conectar um aplicativo *Flutter* com o *Firebase*, para usar os serviços *Firestore* e *Authentication*.

5
Projeto e especificação do sistema

Foi utilizado como base os passos do blog Como Fazer um App (COMO..., 2018). Com base nisso e nos conhecimentos adquiridos durante a graduação, foram desenvolvidos: lista de requisitos, banco de dados, diagrama de sequência, sitemap, wireframe e casos de uso.

5.1
Lista de requisitos

Os requisitos funcionais são uma especificação dos comportamentos e das funcionalidades que um sistema de software deve ser capaz de realizar. Os requisitos não funcionais são atributos e características do sistema que especificam qualidades, restrições e restrições de desempenho que vão além das funcionalidades básicas. Listo, a seguir, ambos os tipos de requisitos.

5.2
Lista de requisitos funcionais

- Um motorista deve poder fazer login usando usuário e senha;
- Um motorista deve selecionar um ônibus para iniciar uma viagem;
- Um motorista deve selecionar uma rota e horário para iniciar uma viagem;
- Um motorista deve poder iniciar/finalizar uma viagem;

– Um motorista deve poder verificar o ticket do passageiro;

– Um motorista deve poder ver a lotação em tempo real do ônibus.

5.3 Lista de requisitos não-funcionais

– O sistema deve executar em android;

5.4 Banco de dados

Se tratando de um sistema robusto com regras de negócio em que se faz necessário armazenar, gerenciar e recuperar informações, é preciso definir um banco de dados, como descrito no capítulo 1 do livro "Database Systems: Design, Implementation, and Management"by Carlos Coronel, Steven Morris, and Peter Rob (CORONEL; MORRIS, 2016). Ainda é necessário resolver qual tipo de banco será usado e qual ferramenta seria mais adequada para a problemática do problema, por isso serão apresentadas as opções de estudo.

5.5 Banco relacional vs banco não relacional

As principais características de banco relacional de acordo com o livro An Introduction to Database Systems (HALVORSEN, 2016): os dados são apresentados como tabelas e apenas tabelas relacionando entre si, integridade das estruturas, visto que há regras bem definidas e o aspecto de manipulação, é possível fazer pesquisas muito complexas.

De acordo com o livro NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence (SADALAGE; FOWLER, 2013), não há características que definem um banco não relacional e sim características que são mais comuns nesses bancos, as principais delas são: o não uso do modelo relacional, roda bem em clusters, não tem esquemáticos e por isso são mais dinâmicos com relação ao seus campos e são construídos para aplicações web do século 21.

Dentro das características descritas não há um claro modelo em vantagem, porém há uma ferramenta a mais a se levar em consideração dado o escopo do projeto: o *Firebase* da Google. Esse é uma plataforma digital de desenvolvimento de aplicativos que possui funcionalidades que facilitam o desenvolvimento de aplicativos e sistemas web como: *Firestore*, banco de dados não relacional do Firebase; *Authentication*, que permite a criação de usuários pelo próprio Firebase.

O *Firestore* é um banco de dados orientado a documentos, isso significa que, ao invés de possuir tabelas se relacionando, ele utiliza coleções para guardar documentos. As coleções podem ser comparadas às tabelas do banco relacional, possuem nomes únicos e podem conter vários documentos, um exemplo de coleção do projeto é a Trip, que é usada para armazenar documentos de viagens. Os documentos por sua vez são as unidades de dados que podem possuir diversas propriedades representadas por chave e valor, onde o valor pode ser de diversos tipos como string, booleanos e números.

A principal vantagem do *Firestore* para nossa prova de conceito é a integração entre os aplicativos e o sistema web, todos os dados são compartilhados entre os três sistemas. Todas as estruturas são criadas em um único lugar e todos os acessos são aos mesmos dados, dando mais confiabilidade na operação.

O Authentication permite a criação de usuários na mesma plataforma que irá gerenciar os dados desse usuário, no caso o Firebase. Essa funcionalidade gera segurança por possuir suas regras de autenticação já existentes, sem a necessidade de novas implementações.

Dessa forma decidiu-se pela utilização do Firebase como plataforma de auxílio do desenvolvimento e do *Firestore Database* como banco de dados do projeto.

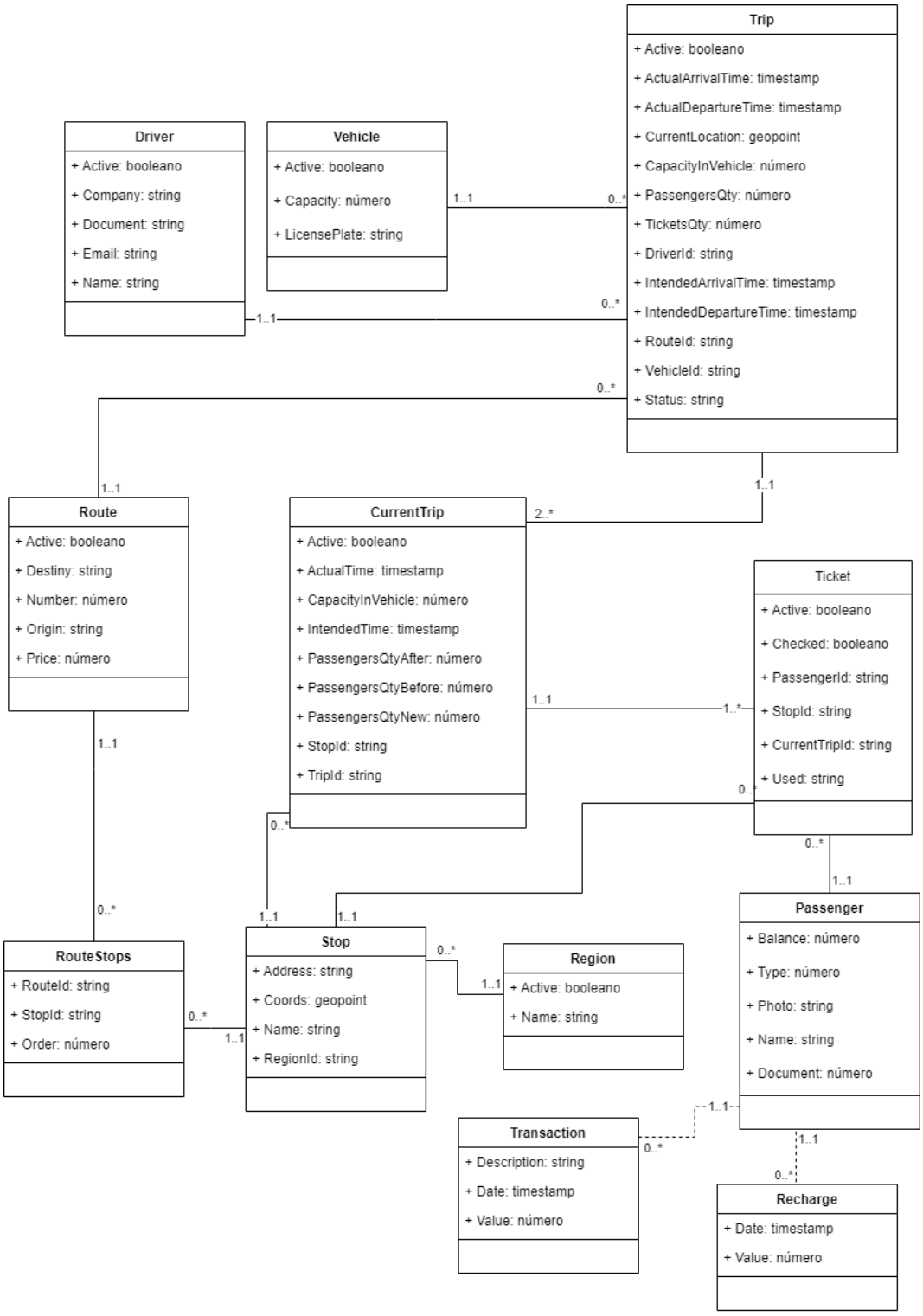


Figura 5.1: Modelagem do banco de dados

Ao utilizar o Firestore como banco de dados para armazenar dados, nos deparamos com a necessidade de agrupar certos atributos que, em um banco de dados relacional, estariam separados. Essa abordagem visa facilitar a utilização do banco de dados em aplicações.

Uma das situações com as quais nos deparamos é a realização da operação de "join", que é fundamental em um modelo de dados relacional para relacionar uma chave estrangeira (FK) com uma chave primária (PK). No modelo orientado a documentos do Firestore, temos duas opções para lidar com essa necessidade. A primeira opção é incluir em um dos campos do documento uma referência ao ID do outro documento, permitindo a busca desse documento específico na outra coleção. A segunda opção é criar subcoleções dentro dos documentos, de forma que, ao buscar um documento, possamos acessar suas sub coleções de documentos relacionados.

A seguir, vamos percorrer as coleções e exibir os atributos de cada documento para proporcionar uma melhor compreensão do nosso sistema.

É importante ressaltar que, em alguns documentos, incluímos o atributo "Active". Essa escolha foi feita com o intuito de proporcionar uma maior rastreabilidade das informações. Assim, caso um desses documentos seja excluído, não o deletaremos do banco de dados, mas definiremos esse atributo como falso, impedindo que seja referenciado por novos documentos. Com essa abordagem, garantimos a preservação das informações em nosso banco de dados, permitindo a exclusão de dados de referência que estejam sendo referenciados em outros locais.

Na coleção "Driver", organizamos as informações essenciais dos nossos motoristas. Cada documento nessa coleção contém campos como "Company", que representa a empresa em que o motorista trabalha, "Document", que guarda o número do documento do funcionário, "Email", que indica o endereço eletrônico, e "Name", que registra o nome do motorista.

A coleção "Vehicle"abrange os veículos que podem iniciar uma viagem. Em seus documentos, encontramos campos como "Capacity", que representa a quantidade de assentos disponíveis no veículo, "LicensePlate", que indica a placa do veículo, e "Active", um valor booleano para indicar se o veículo está desativado, evitando a perda de referência aos dados relacionados a ele.

Já a coleção "Region"traz informações sobre as regiões. Cada documento contém o campo "Name", que indica o nome da região, e "Active", um valor que indica se a região ainda pode ser referenciada por outros documentos.

A coleção "Stop"tem como propósito apresentar os pontos de parada para os ônibus. Em seus documentos, encontramos campos como "Address", que representa o endereço do ponto, "Coords", que são as coordenadas geográficas do ponto, "Name", que registra o nome do ponto, e "RegionId", um identificador único de uma região utilizado para referenciá-la.

Na coleção "Route", temos os campos "Destiny", que indica o ID de referência do ponto final da rota; "Number", que representa o número da rota; "Origin", que indica o ID de referência do ponto inicial da rota; "Price", que registra o valor da passagem para essa rota; e "Active", utilizado para desativar uma rota sem afetar outros documentos que a referenciam.

Na coleção "Trip", armazenamos informações cruciais para o monitoramento e gerenciamento das viagens. Cada documento dessa coleção possui campos como "ActualArrivalTime"e "ActualDepartureTime", que registram o momento exato em que a viagem começou e terminou; "CurrentLocation", que contém as coordenadas geográficas da localização atual do ônibus, sendo atualizadas em tempo real durante toda a duração da viagem; "CapacityInVehicle", que indica a quantidade de assentos disponíveis no veículo utilizado para aquela viagem. Optamos por não estabelecer uma relação direta com o veículo e obter essa informação a partir do documento do veículo, pois em casos de viagens futuras, podemos enfrentar problemas com o veículo originalmente planejado. Assim, no início da viagem, é possível associar um veículo que tenha a capacidade necessária; "PassengersQty", que registra a quantidade atual de passageiros a bordo do ônibus. Conforme os motoristas leem os bilhetes dos passageiros, esse número é atualizado. O campo "TicketsQty"indica a quantidade de bilhetes gerados para aquela viagem. À medida que os passageiros geram bilhetes, esse número é incrementado. Também utilizamos o campo

"TicketsQty"para verificar se ainda é possível emitir um novo bilhete com base na capacidade disponível no veículo; "DriverId"é o código de identificação onde podemos referenciar o motorista no banco de dados, atualizado no início da viagem; Os campos "IntendedArrivalTime"e "IntendedDepartureTime"referem-se aos horários programados para o início e término da viagem. "RouteId"é o código de referência da rota no banco de dados, enquanto "VehicleId"é o código de referência do veículo no banco de dados; "Status"indica se a viagem está em andamento, já foi concluída, não foi iniciada ou foi interrompida.

Para cada parada de uma roda da viagem, temos um documento na coleção de "CurrentTrip", que é responsável pelo acompanhamento daquela viagem no determinado ponto. Nesse documento temos os seguintes campos: "ActualTime"que indica o momento que o ônibus saiu daquele ponto naquela viagem; "CapacityInVehicle"que indica a quantidade de assentos que o veículo usado para aquela viagem tem, optamos por colocar esse atributo aqui para facilitar a trazer essa informação nas aplicações; "IntendedTime"que é a hora prevista para passar por aquele ponto aquela viagem; "PassengersQtyAfter"que é quantidade de passageiros dentro do ônibus ao sair daquele ponto; "PassengersQtyBefore"que é quantidade de passageiros dentro do ônibus ao chegar naquele ponto; "PassengersQtyNew é quantidade de passageiros que entraram no ônibus naquele ponto; "StopId"o código de identificação em que referenciamos do ponto de ônibus no banco de dados;"TripId"que é o código de referência da viagem no banco de dados.

A coleção "Ticket"armazena informações relacionadas aos tickets utilizados pelos passageiros. Cada documento nessa coleção contém campos como "Active", indicando se o ticket está ativo ou não; "Checked", indicando se o ticket já foi utilizado; "PassengerId", que é o código de identificação do passageiro no banco de dados; "StopId", que é o código de identificação do ponto de ônibus no banco de dados; "CurrentTripId", que é o código de identificação do CurrentTrip no banco de dados; e "Used", indicando se o ticket já foi usado

ou não. Esses campos permitem o controle e rastreamento dos tickets ao longo das viagens.

Por outro lado, a coleção "RouteStops"armazena informações sobre os pontos de parada em uma determinada rota. Cada documento nessa coleção possui os campos "RouteId", representando o código de identificação da rota no banco de dados; "StopId", representando o código de identificação do ponto de ônibus no banco de dados; e "Order", indicando a ordem do ponto de parada nessa rota. Esses campos permitem a organização adequada dos pontos de parada em uma sequência específica, ajudando na definição e planejamento das rotas de viagem.

A coleção "Passenger"armazena informações relacionadas ao perfil do passageiro, incluindo seu histórico de ações. Cada documento nessa coleção contém campos como "Balance", representando o saldo atual da conta do passageiro, que é atualizado quando ocorrem recargas ou emissões de tickets; "Document", representando o número de um documento do passageiro; "Name", representando o nome do passageiro; "Photo", representando uma foto associada ao perfil do passageiro; e "Type", indicando o tipo de passageiro, como funcionário, aluno ou outro.

Dentro dos documentos da coleção "Passenger", existem duas subcoleções. A primeira é a "Transaction", que representa transações financeiras relacionadas ao passageiro. Cada documento nessa subcoleção possui campos como "Value", representando o valor da transação; "Date", representando a data da transação; e "Description", fornecendo uma descrição da transação, como compra de bilhetes ou estorno de cancelamento.

A segunda subcoleção é "Recharge", que registra o histórico das recargas de crédito feitas na conta do passageiro. Cada documento nessa subcoleção possui campos como "Balance", representando o valor recarregado; e "Date", representando a data da recarga.

5.6
Diagrama de sequência

O diagrama de sequências modela as interações entre objetos de um sistema em um único caso de uso. Ele ilustra como as diferentes partes de um sistema interagem entre si para realizar uma função, e a ordem em que as interações ocorrem quando um determinado caso de uso é executado.

O diagrama de sequência (figura 5.2) indica os momentos com maior clareza, principalmente a comunicação entre passageiro, sistema do passageiro, motorista e o sistema do motorista. Com relação à plataforma web, podemos ver algo mais paralelo, sem muita necessidade de espera de momentos e processos.

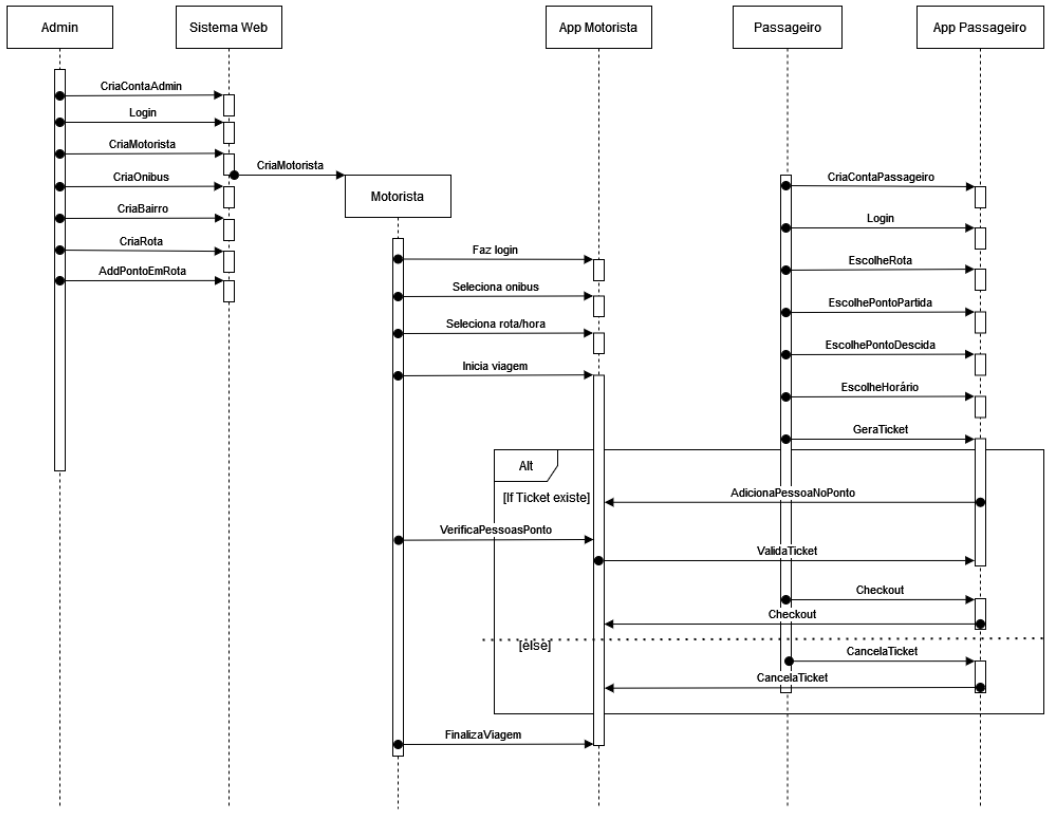


Figura 5.2: Diagrama de sequências

Como podemos ver no fluxo de web (figura 5.3) temos 3 momentos de dependência: na criação da conta admin, na criação da conta do motorista e na adição de um ponto a uma rota.

A dependência na criação de uma conta admin se dá devido ao fato de que, para podermos fazer login, é necessário que uma conta seja criada. A partir desse momento, o usuário consegue utilizar o sistema Web.

O segundo momento de dependência se encontra na criação de um motorista, para que o motorista consiga acessar seu sistema e de fato existir no sistema é preciso que o administrador crie esse motorista.

O terceiro momento seria a adição de ponto em rota, pois é preciso que uma rota seja criada e, consequentemente, um ponto.

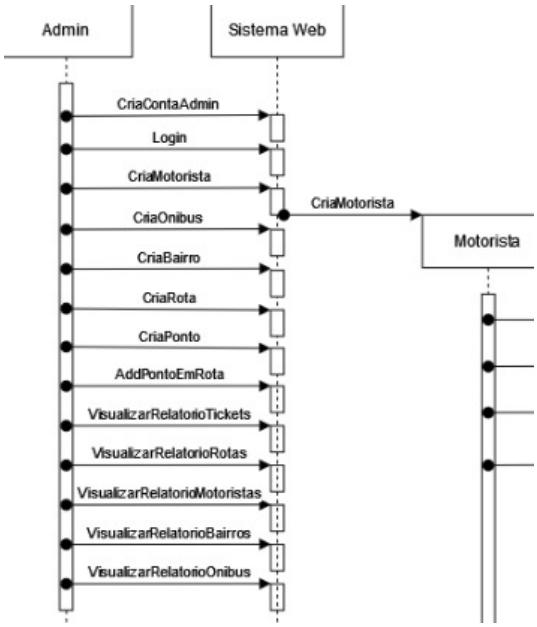


Figura 5.3: Diagrama de sequências da plataforma web

Além dessas dependências, podemos, também, ver os processos gerenciais em que as entidades são criadas e os processos de visualização dos relatórios gerados, com base nas viagens realizadas.

5.7 Sitemap

Sitemap é uma representação da estrutura de um site ou aplicativo, mostrando as diferentes telas e a navegação entre elas. Ele permite uma compreensão clara da organização e do fluxo do mesmo.

Como se observa na figura 5.4, ao abrir o aplicativo, o motorista verá a tela de login, onde pode inserir seus dados e entrar na plataforma. Caso esse

seja seu primeiro acesso, ele pode ir para a página de cadastro para uma senha de acesso.

Ao entrar no sistema, o usuário é redirecionado para a home. Nesta página, pode-se iniciar uma nova viagem sendo redirecionado para a página de viagem atual.

Na página de viagem atual, o motorista pode ir para a página de validação de ticket ou finalizar a viagem e voltar para a página inicial.

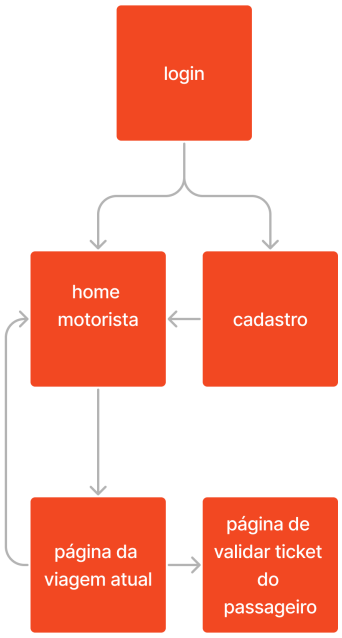


Figura 5.4: Sitemap do app do motorista

5.8 Wireframe

Segundo Felipe Memoria, no livro *Design para a internet* (MEMÓRIA, 2006), wireframe é um "rascunho das telas que posiciona a informação e navegação, incluindo-se aí agrupamento, ordem e hierarquia do conteúdo". Essa ferramenta auxilia no desenvolvimento, por disponibilizar a organização dos elementos de interface, além de servir como documentação do projeto.

No wireframe do aplicativo do motorista temos: a tela de login (figura 5.5), tela de cadastro (figura 5.6), página inicial (figura 5.7), página da viagem atual (figura 5.8) e página de validação de ticket (figura 5.9a).

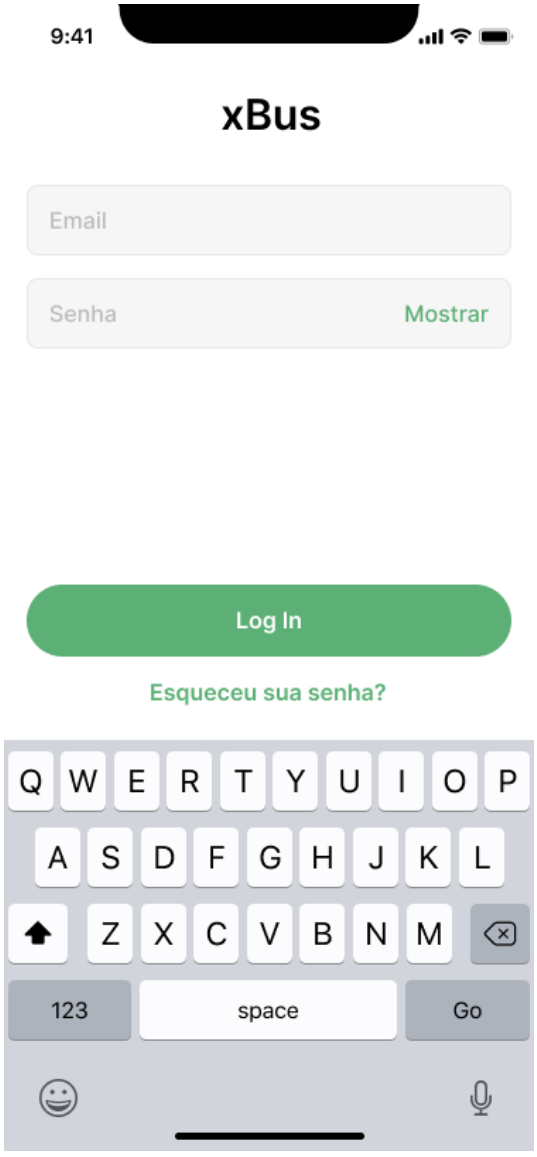


Figura 5.5: Wireframe login do app do motorista

Figura 5.6: Wireframe cadastro do app do motorista

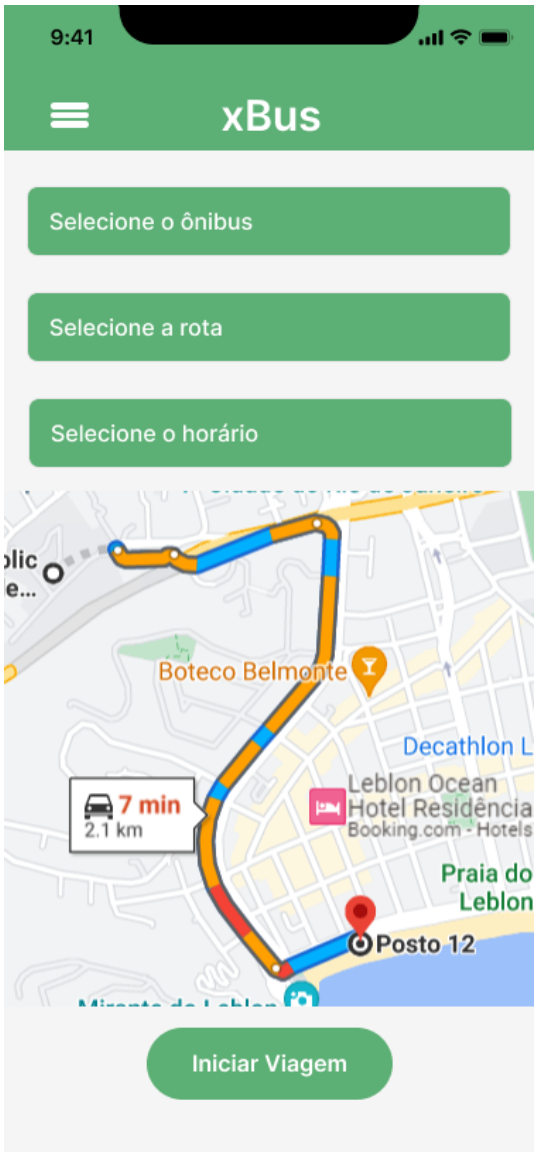


Figura 5.7: Wireframe página inicial do app do motorista

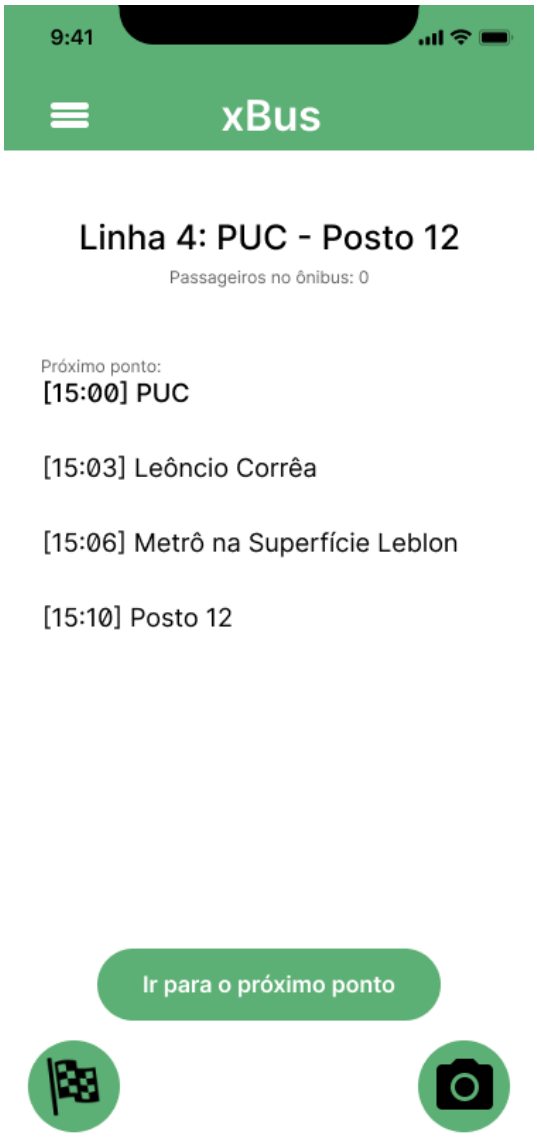
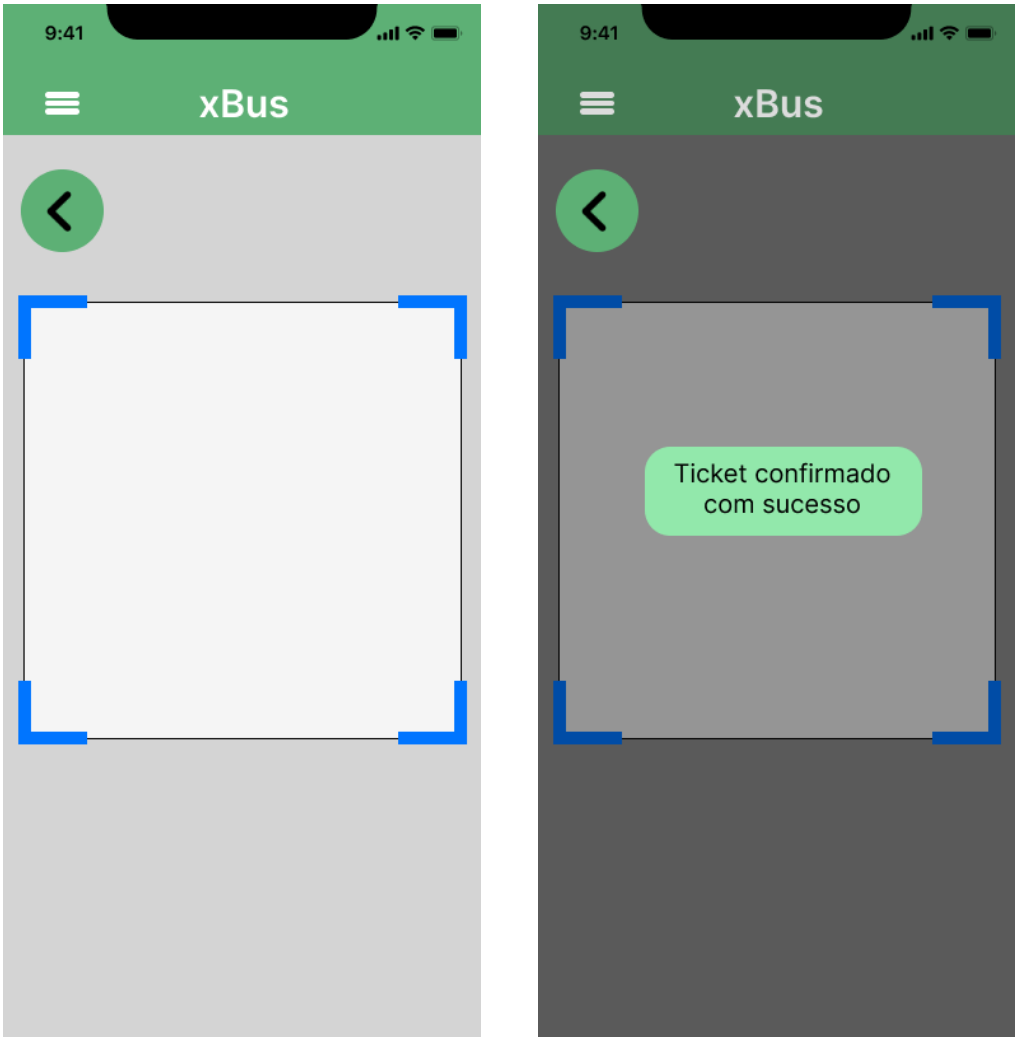


Figura 5.8: Wireframe página viagem atual



(a) Câmera (b) Sucesso

Figura 5.9: Wireframe página validação de ticket

5.9 Casos de uso

O aplicativo do motorista possui quatro casos de uso: login, cadastrar, iniciar viagem e validar ticket.

Como consta na tabela 5.1, para entrar no aplicativo, o motorista precisa de uma senha previamente cadastrada por ele. Ao fornecer sua senha e e-mail o login é efetuado com sucesso e ocorre o redirecionamento para a página inicial. Se algum dado fornecido estiver incorreto, uma mensagem de erro aparecerá e o motorista pode tentar novamente.

Caso o motorista ainda não tenha cadastrado sua senha (tabela 5.2),

Nome:	Login
Objetivo:	Entrar no sistema
Atores:	Motorista
Pré-condição:	Ter uma conta (nome de usuário e senha)
Pós-condição (cenário de sucesso):	Ter uma conta Motorista logado
Pós-condição (cenário de insucesso):	Sem acesso ao sistema
Pós-condição (Trigger):	Motorista entra na página de login
Pós-condição (Fluxo Principal):	1. Aparecem 2 caixas de texto para o nome de usuário e senha; 2.Motorista preenche com suas credenciais as caixas de texto; 3. Motorista clica no botão entrar; 4. Sistema verifica credenciais fornecidas [E1]; 5. Motorista é redirecionado para a página inicial logado;
Pós-condição (Fluxos de Exceção):	[E1] Senha ou nome de usuário inválido; 1. Sistema avisa que a senha ou o nome de usuário está inválido; 2. Retorne para o passo 2 do Fluxo Principal

Tabela 5.1: Caso de uso login do motorista

basta fornecer, na página de cadastro, o seu documento previamente cadastrado no sistema pelo administrador e uma senha a ser usada no login. Se os dados fornecidos estejam corretos, ocorrerá o redirecionamento para a página inicial. Caso contrário, uma mensagem de erro aparecerá e o motorista pode tentar novamente.

Na página inicial do aplicativo, quando o usuário já está logado, é possível iniciar uma nova viagem (tabela 5.3). Basta escolher uma rota, um horário e um veículo e dar início à viagem.

É necessário chamar atenção à três regras de negócio nesse momento.

- O motorista deve selecionar o ônibus ser na viagem, pois pode haver ônibus com diferentes ocupações (quantidade de assentos);
- O motorista deve selecionar o horário, pois pode ser que ele esteja saindo atrasado, porém, ele não deve se adiantar;

Nome:	Cadastro de senha
Objetivo:	Cadastrar uma senha e entrar no sistema
Atores:	Motorista
Pré-condição:	Ter o documento cadastrado no sistema pelo administrador
Pós-condição (cenário de sucesso):	Ter uma conta Motorista logado
Pós-condição (cenário de insucesso):	Senha não criada
Pós-condição (Trigger):	Motorista clica no botão *Cadastro* na página de login
Pós-condição (Fluxo Principal):	1. Aparecem 2 caixas de texto: documento e senha; 2. Motorista preenche as caixas de texto; 3. Motorista clica no botão *Cadastrar*; 4. Sistema verifica credenciais fornecidas [E1]; 5. Sistema salva usuário; 6. Motorista é redirecionado para a página inicial logado;
Pós-condição (Fluxos de Exceção):	[E1] Documento do usuário não consta no sistema; 1. Sistema avisa que o documento está inválido; 2. retorne para o passo 2 do Fluxo Principal

Tabela 5.2: Caso de uso cadastro de senha

– O sistema é notificado que a viagem iniciou para aparecer no sistema web e para poder atualizar o passageiro.

Nome:	Iniciar viagem
Objetivo:	Iniciar uma viagem
Atores:	Motorista
Pré-condição:	Estar logado Não ter uma viagem iniciada
Pós-condição (cenário de sucesso):	Viagem em andamento
Pós-condição (cenário de insucesso):	Viagem não iniciada
Pós-condição (Trigger):	Motorista entra na página de login
Pós-condição (Fluxo Principal):	1. Selecionar a rota; 2. Selecionar o horário; 3. Selecionar o ônibus; 4. Clicar no botão "iniciar viagem"; Sistema é notificado que a viagem iniciou; 5. Motorista é redirecionado para o mapa com a rota.
Regras de Negócio:	[RN1] O motorista deve selecionar o ônibus, pois pode haver ônibus com diferentes ocupações (quantidade de assentos); [RN2] O motorista deve selecionar o horário, pois pode ser que ele esteja saindo atrasado; [RN3] O Sistema é notificado que a viagem iniciou para aparecer no sistema web e para poder atualizar o passageiro.

Tabela 5.3: Caso de uso iniciar viagem

Enquanto o motorista tiver uma viagem em andamento ele pode validar um ticket (tabela 5.4). Na tela da viagem atual, ao clicar no botão com ícone de código QR, a câmera do celular será aberta. Quando a câmera detectar um código QR, verificará se ele está ativo e se ainda não foi usado e mostrará uma mensagem de sucesso ou falha.

Nome:	Validação de ticket
Objetivo:	Verificar o ticket do passageiro
Atores:	Motorista
Pré-condição:	Ter uma viagem iniciada
Pós-condição (cenário de sucesso):	Ticket do passageiro validado
Pós-condição (cenário de insucesso):	Ticket do passageiro não validado
Pós-condição (Trigger):	Motorista clicar no botão escanear na página do mapa com a rota atual
Pós-condição (Fluxo Principal):	1. Abre-se a câmera para escanear; 2. Quando o código QR for validado [E1] aparecerá uma mensagem na tela informando o sucesso; 3. o sistema atualizará a quantidade de passageiros naquele ônibus; 4. o ticket será marcado como usado no sistema.
Regras de Negócio:	[E1] Se o código QR sendo escaneado for inválido ou for de um ticket não ativo, aparecerá uma mensagem, informando o erro; [RN2] O motorista deve selecionar o horário, pois pode ser que ele esteja saindo atrasado; [RN3] O Sistema é notificado que a viagem iniciou para aparecer no sistema web e para poder atualizar o passageiro.

Tabela 5.4: Caso de uso validar ticket

6
Implementação e avaliação

O projeto está disponível no repositório do *github* e pode ser acessado através do link <https://github.com/marianaruddy/xbus>.

Para o desenvolvimento foi utilizado o framework *Flutter* (versão 3.3.2). Esse framework foi escolhido pela simplicidade para o desenvolvimento, por ser um framework cruzado e pela função *hot reload*.

O framework do *Flutter* consta com funções gráficas que facilitam a criação de interfaces personalizadas e sofisticadas com segurança para os aplicativos. Como dispensa o uso de diferentes plataformas e ferramentas de criação, há uma redução nítida do custo de desenvolvimento das aplicações.

Outro ponto de realce é que como se trata de um framework cruzado, os desenvolvedores utilizam a mesma base de códigos para ambos os sistemas operacionais e não precisam fazer diferentes versões das aplicações dos produtos criados, o que já leva à diminuição de despesas e de tempo de desenvolvimento. Assim unificando a criação para iOS e Android.

O framework também favorece a velocidade no desenvolvimento disponibilizando a *função hot reloaded*, na qual podemos ver as modificações feitas em tempo real no aplicativo, independente se estiver usando um emulador ou no hardware.

6.1 Desenvolvimento

Primeiramente, criou-se um projeto *Flutter* usando o comando (FLUTTER. . . ,):

```
flutter create driver
```

Esse comando cria um aplicativo de teste já estruturado. A partir desse projeto, as telas foram desenvolvidas conforme a arquitetura descrita na seção a seguir (seção 6.2).

6.2 Arquitetura

Na pasta *driver/lib* encontra-se as pastas *config*, *models*, *screens*, *service* e *shared*, além do arquivo *main.dart*, que é onde chamamos funcão *main()*, que é o entrypoint da aplicação.

Na pasta *config*, está o arquivo *google_maps_api_key*, que possui a chave secreta do *Firebase* e não deve ser commitado por motivos de segurança. Nesta pasta está também o arquivo *.gitignore*, que esconde os arquivos sensíveis no versionamento de código.

Na pasta *models* estão os arquivos que mapeiam os modelos das entidades do banco de dados, indicando tipo e nullabilidade. Por exemplo, a entidade Route.

```
class RouteModel {
  final String id;
  final String destiny;
  final int number;
  final String origin;
  final num? price;

  RouteModel ({
    required this.id,
    required this.destiny,
    required this.number,
    required this.origin,
    this.price,
  });
}
```

Como vimos na seção 5.4, O Route possui os campos *Destiny*, *Number*, *Origin* e *Price*. Na classe RouteModel, além dos campos acima citados, adiciona-se o *id*, que é o código de identificação da rota gerado automaticamente pelo banco de dados.

Na pasta *screens* estão os arquivos que implementam as telas do aplicativo.

Na pasta *service* temos os arquivos que intermedeiam os dados entre o aplicativo e o *Firebase*. Através dele, acessamos as coleções e usuários salvos nessa nuvem.

Por fim, temos, na pasta *shared*, componentes reutilizáveis, como o *Loading*.

7
Considerações finais

O projeto xBus tem o potencial de facilitar a rotina das pessoas que frequentam a PUC Rio. Traz a possibilidade de diminuir preocupações com o deslocamento permitindo que alunos foquem em seu desenvolvimento acadêmico e funcionários, em suas atividades.

Durante a produção deste trabalho, tive a oportunidade, de colocar em prática conceitos aprendidos durantes as disciplinas frequentadas tanto ao longo do curso de engenharia da computação, na PUC Rio, como ao longo do intercâmbio, no departamento de ciência da computação da Hochschule Darmstadt. Pude, também, aprender novas tecnologias, como *dart*, *Flutter* e *Firebase*. Além de praticar e aprimorar minhas habilidades de comunicação e trabalho e trabalho em equipe.

Ainda há espaço para melhorias a adição de funcionalidades, como indicar que um ponto foi visitado de forma automática, sem a necessidade do motorista apertar um botão, e o checkout, que permitiria disponibilizar um assento que não está mais ocupado, durante a viagem.

Espera-se que esse projeto tenha continuidade para que as pessoas que frequentam a PUC-Rio possam usufrir dos seus diversos benefícios em um futuro próximo.

8
Referências bibliográficas

COMO Fazer um App. 2018. Acessado em: 2022-10-10. Disponível em: <https://comofazerumapp.wordpress.com/2018/03/16/desenvolvendo-o-wireframe/>.

CORONEL, C.; MORRIS, S. **Database systems: design, implementation, & management**. [S.l.]: Cengage Learning, 2016.

FLUTTER Firebase App Tutorial. 2019. Acessado em: 2023-01-10. Disponível em: <https://www.youtube.com/watch?v=sfA3NWDBPZ4&list=PL4cUxeGkcC9j--TKIdkb3ISfRbJeJYQwC&index=3>.

FLUTTER: The Flutter command-line tool. Acessado em: 2023-06-17. Disponível em: <https://docs.flutter.dev/reference/flutter-cli>.

HALVORSEN, H. Introduction to database systems. **Notodden, Norway: University College of Southeast Norway**, 2016.

LIRA, M. **Saiba detalhes do Moovit e facilite sua mobilidade**. 2021. Acessado em: 2022-11-22. Disponível em: <https://blog.b2bstack.com.br/moovit/>.

MEMÓRIA, F. **Design para a internet**. [S.l.]: Elsevier Brasil, 2006. 36 p.

SADALAGE, P. J.; FOWLER, M. **NoSQL distilled: a brief guide to the emerging world of polyglot persistence**. [S.l.]: Pearson Education, 2013.

9
Anexos

9.1
Manual do usuário

Para rodar o projeto, basta rodar o comando:

```
flutter run
```

na pasta 'driver'.

Com o projeto rodando, a primeira tela que aparece é a de login (figura 9.1).

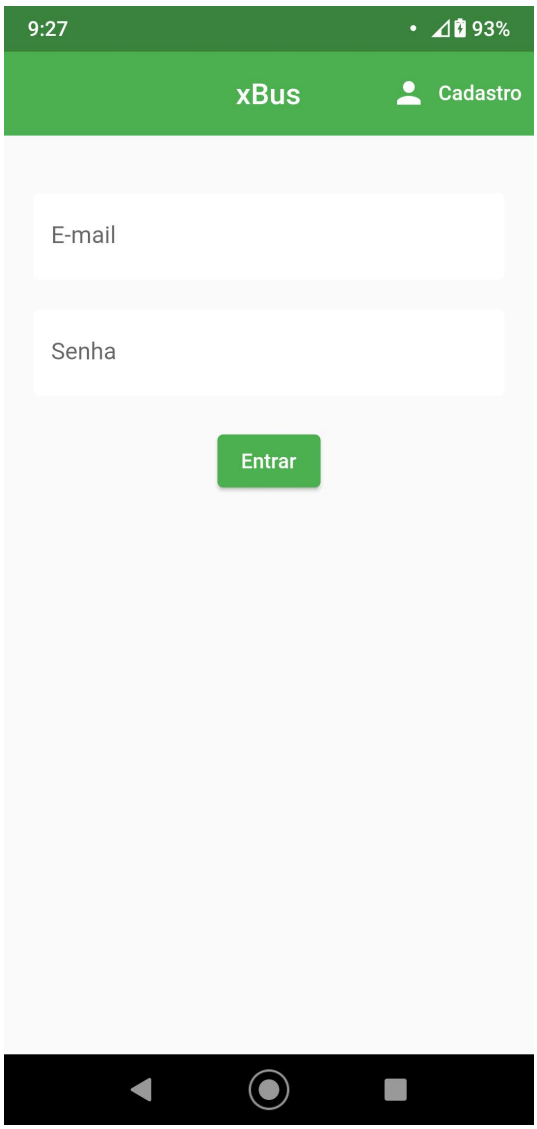


Figura 9.1: Login

Se o usuário já possuir credenciais para logar, basta fornecer os dados e clicar em *Entrar* para seguir em frente. Caso contrário, pode clicar em *Cadastrar* para criar uma senha.

Na página de cadastro (figura 9.2), o motorista pode fornecer seu número de documento para cadastrar uma senha de acesso ao app.

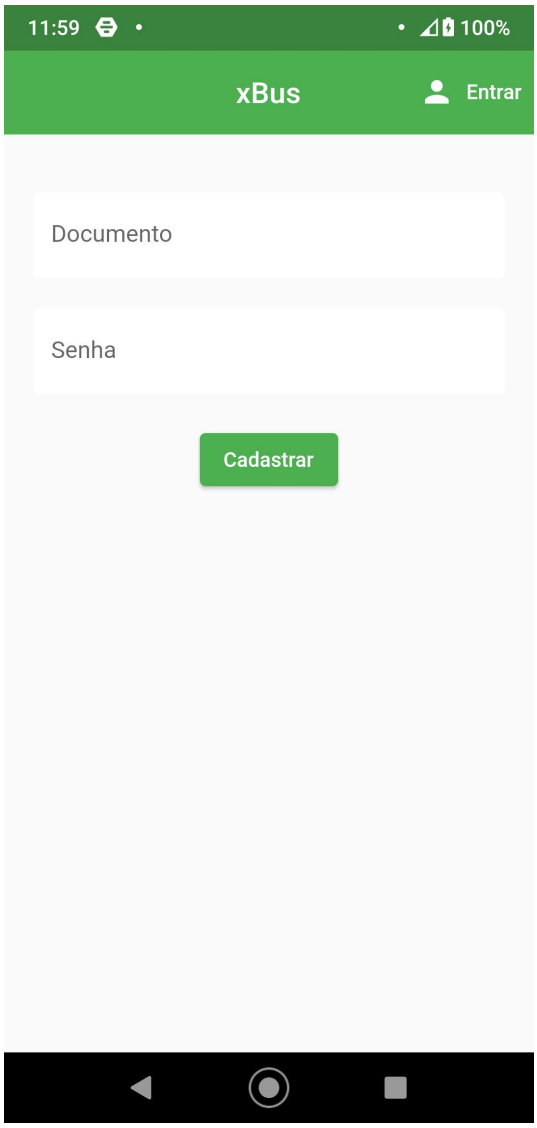
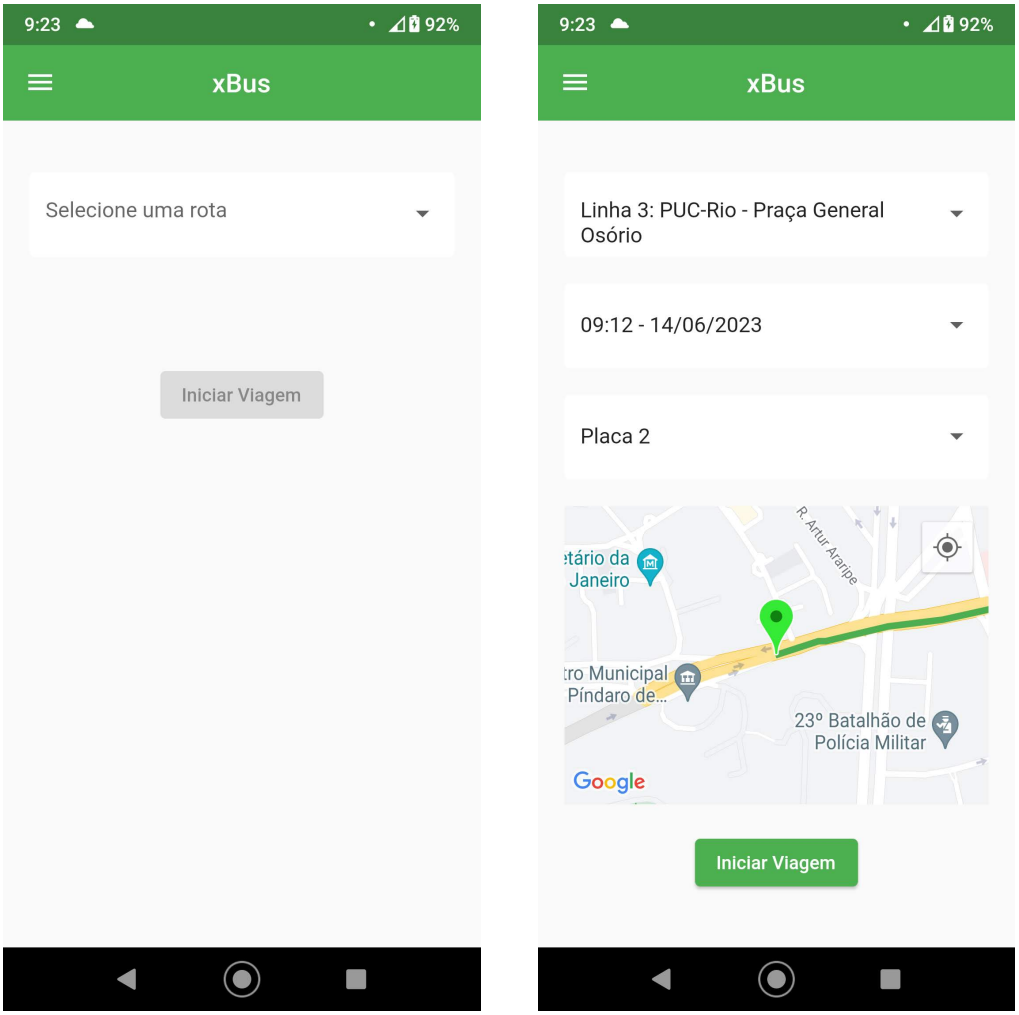


Figura 9.2: Cadastro

Na página inicial (figura 9.3a), o motorista seleciona uma rota. Aparecerá, então, um mapa com a rota destacada. Aparecerão, também, os horários programados para a rota selecionada. Ao selecionar um desses horários, aparecerá então a lista de veículos com capacidade suficiente para aquela viagem. Selecionando um veículo, pode- se, por fim, apertar o botão para iniciar a viagem (figura 9.3b).



(a) Nenhuma rota selecionada

(b) Rota, horário e veículo selecionados

Figura 9.3: Página inicial

Na página da viagem atual (figura 9.4), pode-se ver as informações de qual rota está em andamento, a quantidade de passageiros no ônibus, os pontos em ordem e quais já foram visitados. Além disso, é possível clicar no ícone no canto inferior direito para ir para a página de validação de ticket; clicar em *ir para o próximo ponto*, para marcar o ponto atual como visitado; ou clicar no ícone no canto inferior esquerdo para finalizar a viagem.

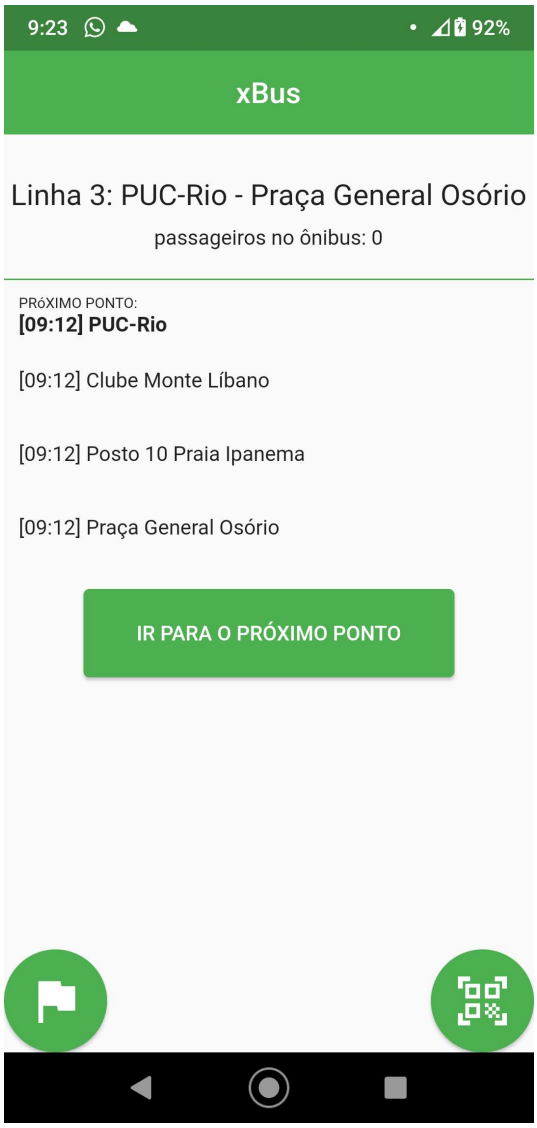


Figura 9.4: Página da viagem em andamento

Na página da validação de ticket, quando a câmera detecta um código, validade-se se o ticket está ativo e não foi usado ainda para mostrar a tela de sucesso (figura 9.5). Ao voltar para a página da viagem atual, é possível perceber que a quantidade de passageiros no ônibus foi acrescida em uma unidade.

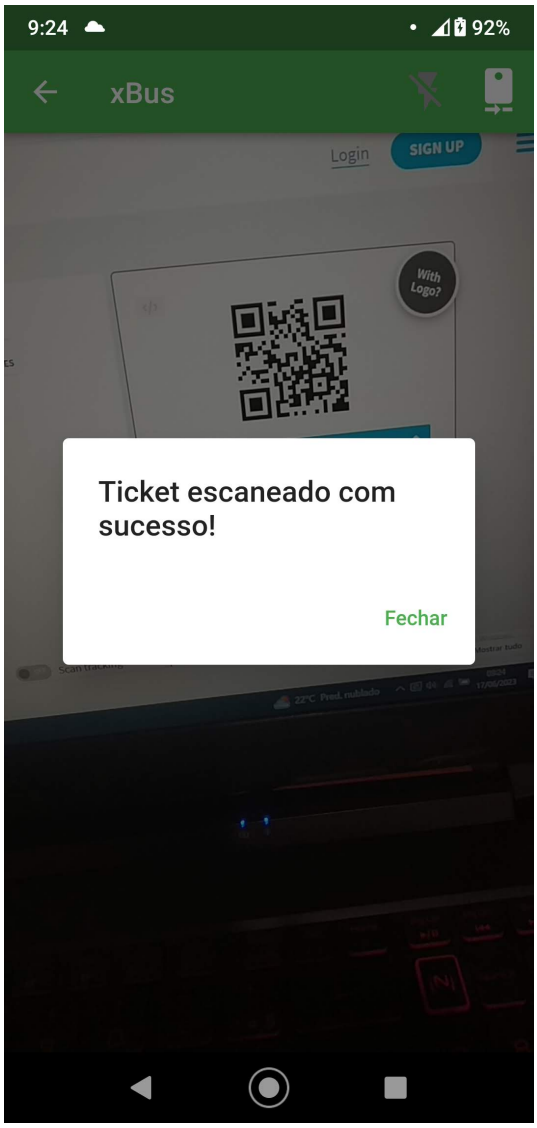
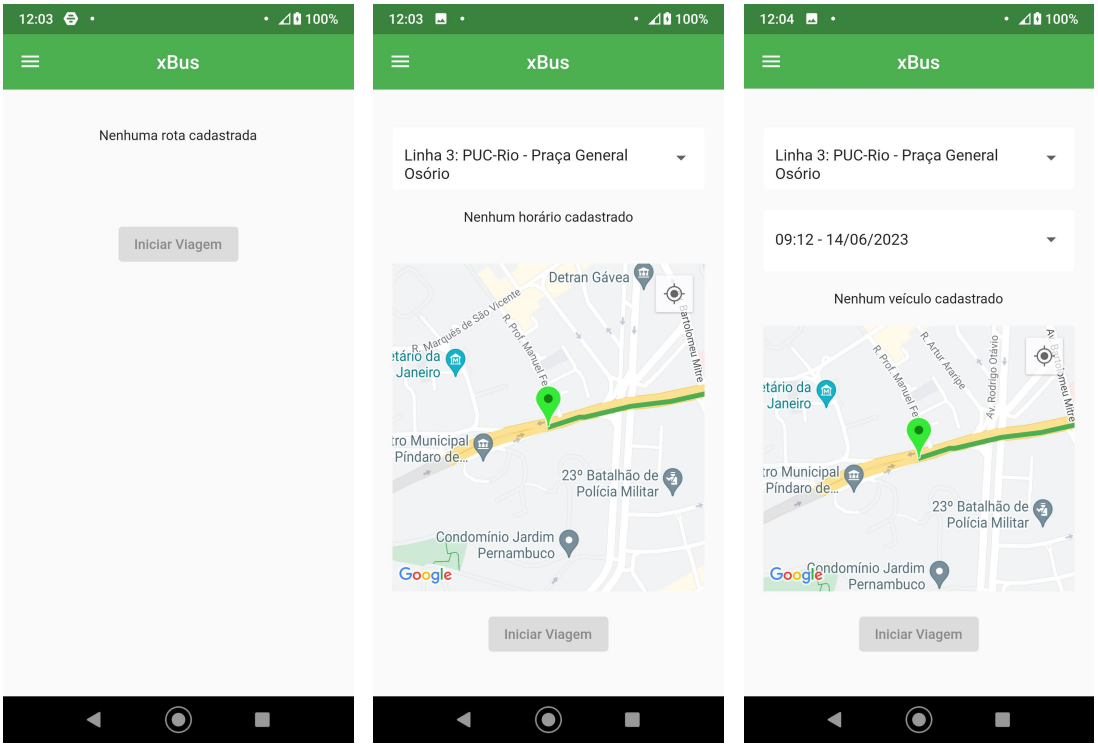


Figura 9.5: Página de ticket escaneado com sucesso

9.1.1 Casos de exceção

Na página inicial, se não houver nenhuma rota cadastrada no sistema, a tela mostrará a mensagem "Nenhuma rota cadastrada"(figura 9.6a. Caso uma rota tenha sido selecionada, mas não houver nenhum horário programado para essa rota no sistema, aparecerá a mensagem "Nenhum horário cadastrado"na tela (figura 9.6b). Caso uma rota e um horário tenham sido selecionados, mas não houver nenhum veículo cadastrado com a capacidade mínima requerida pelo administrador para tal viagem, aparecerá a mensagem "Nenhum veículo cadastrado"na tela (figura 9.6c).



(a) Nenhuma rota disponível

(b) Rota, horário e veículo selecionados

(c) Rota, horário e veículo selecionados

Figura 9.6: Página inicial

Caso o motorista, na página de cadastro, forneça um número de documento que não foi cadastrado previamente pelo administrador no sistema, a tela mostrará a mensagem "Digite um documento válido"(figura 9.7).

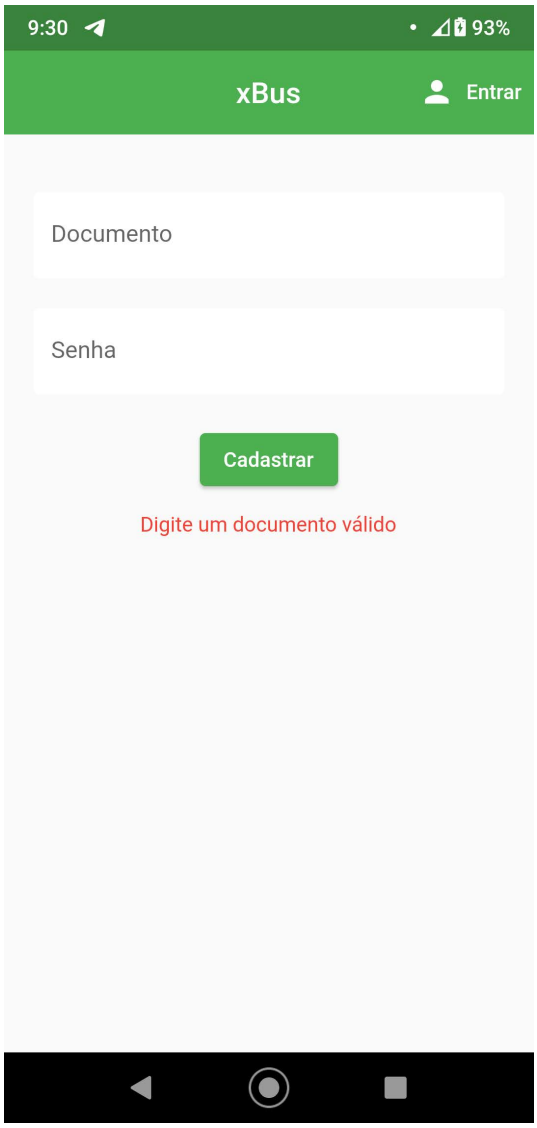
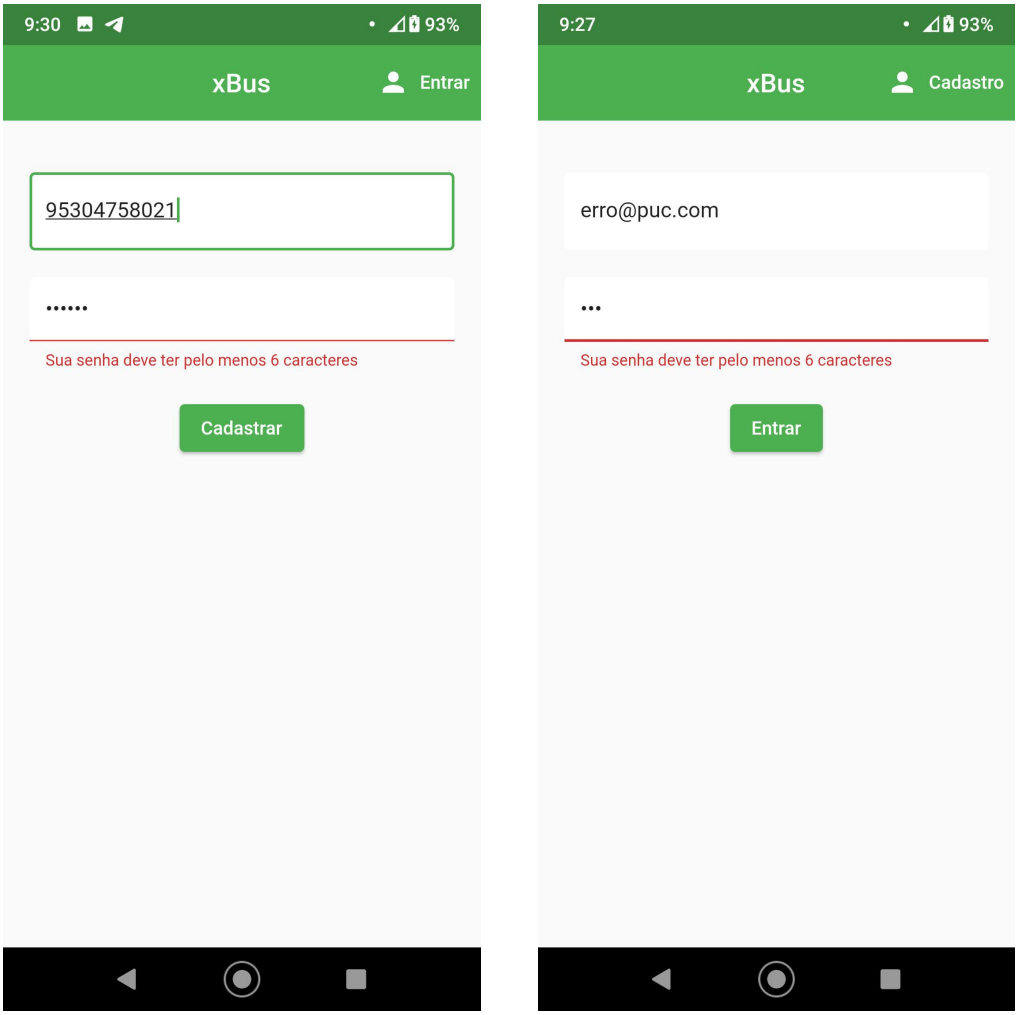


Figura 9.7: Página de registro com documento inválido

Tanto na página de login, como na página de cadastro, se o usuário fornecer, no campo senha, um valor com menos de seis caracteres, verá a mensagem "Sua senha deve ter pelo menos 6 caracteres"(figuras 9.8a e 9.8b).



(a) Cadastro (b) Login

Figura 9.8: Erro na senha

Na página da validação de ticket, se a câmera detectar um código de um ticket que não está ativo ou já foi usado, a tela de falha (figura 9.9) será mostrada.

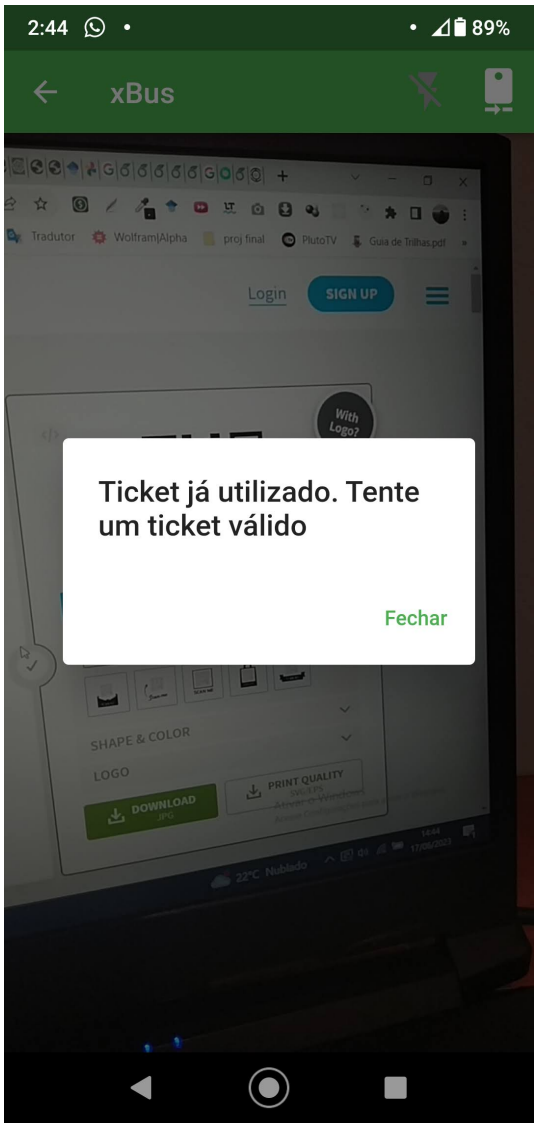


Figura 9.9: Página de ticket escaneado com erro