

# Moov (App de Agendamento de Corridas)

---

## Resumo

Este documento apresenta a documentação técnica do aplicativo Android de agendamento de corridas para motoristas particulares (denominado como Moov). A documentação descreve motivação, objetivos, arquitetura, tecnologias utilizadas (incluindo o HERE Maps SDK), projeto de interface, fluxos de dados, funcionalidades e orientações para implantação e testes. O projeto está dividido em dois repositórios: um para o aplicativo Android (<https://github.com/spinolajunior/Project-moov>) e outro para a API REST (<https://github.com/spinolajunior/project-api-rest>).

## Introdução

### Motivação

A motivação para o desenvolvimento deste aplicativo surgiu da necessidade de atender um público-alvo da minha região que busca uma solução simples para motoristas que oferecem seus serviços de forma particular. O Moov foi idealizado para possibilitar o agendamento e o gerenciamento de corridas entre clientes e motoristas.

### Objetivos específicos

- Implementar autenticação e autorização de usuários (login/cadastro).
- Disponibilizar cadastro e gerenciamento de corridas.
- Integrar localização e mapas usando o HERE Maps SDK.
- Implementar uma API REST com Spring Boot, JPA e MySQL.
- Documentar modelo de dados, arquitetura e interface.

### Organização do trabalho

O trabalho está dividido em especificação e documentação e implementação (código-fonte nos repositórios).

## **Desenvolvimento**

### **Contextualização**

O Moov foi desenvolvido como projeto acadêmico para a disciplina projeto de inovação , para esse projeto foi desenvolvida uma api com a arquitetura RESTful e JPA. O sistema possui usuários do tipo Customer e Driver, estruturados por herança JPA, e entidade Login para autenticação. O repositório do app Android contém as Activities, layouts e integração com HERE Maps SDK. O repositório da API REST contém as entidades, repositórios, serviços e controladores.

### **Identificação do sistema**

Nome: Moov

Tipo: Aplicação distribuída (mobile + API REST)

Público-alvo: Clientes e motoristas

Plataformas: Android e servidor Java (Spring Boot).

### **Modelo de Projeto do Software**

Arquitetura baseada em camadas:

- App Android: interface em Kotlin/Java.
- API REST: Spring Boot, DTOs, Services, Controllers.
- Banco MySQL: persistência via JPA/Hibernate.

Padrões: REST, Repository (Spring Data), DTOs, Injeção de dependência, Herança JOINED no JPA.

### **Metodologia de Gestão**

Para o desenvolvimento do sistema, foi necessário utilizar ferramentas de controle de versão, como Git e GitHub, além de definir a distribuição das tarefas. Inicialmente, o processo concentrou-se no entendimento de como o sistema deveria funcionar. A partir dessa etapa, iniciou-se o desenvolvimento da API REST e, posteriormente, o desenvolvimento do aplicativo Moov.

### **Tecnologias utilizadas**

- Android (Kotlin/Java)
- HERE Maps SDK
- Spring Boot (Java)
- JPA (Hibernate)
- MySQL
- Retrofit, Gson, Coroutines (Comunicação com a api REST)
- Maven para build API
- GitHub para versionamento
- Gradle para build APP

## **Levantamento e análise de requisitos**

### **Requisitos Funcionais.**

Cadastro de usuários : O sistema deve permitir o cadastro de motoristas e clientes com dados básicos (nome, telefone, e-mail, senha).

Login de usuários : O sistema deve permitir que os usuários façam login com suas credenciais.

Solicitação de corridas : O cliente pode reservar vagas em uma corrida.

Visualização de serviços (motorista) : O motorista deve poder visualizar corridas cadastradas.

Mapa: O sistema deve exibir um mapa em que corridas cadastradas devem aparecer como um marcador interativo no local de origem da corrida.

Histórico de corridas/reservas : O sistema deve exibir um histórico das corridas já realizadas por motoristas e clientes.

Notificações básicas : O sistema deve notificar os usuários sobre confirmação de agendamentos e status das corridas.

### **Requisitos Não Funcionais.**

Compatibilidade com Android : O app deve funcionar em dispositivos Android com versões recentes e intermediárias.

Interface intuitiva : A interface deve ser simples e fácil de usar, mesmo por usuários com pouca experiência com tecnologia.

Performance : O app deve ser leve e rápido, garantindo boa experiência mesmo em celulares mais simples.

Armazenamento em nuvem : Os dados devem ser armazenados de forma segura no banco de dados MySQL, acessado via backend.

Comunicação segura : A troca de dados entre o app e o servidor deve ser feita por meio de conexões seguras (HTTPS).

Disponibilidade : O sistema deve estar disponível para uso em horário comercial.

## Fase de desenvolvimento

**Objetivo:** construir o sistema conforme as metas e a arquitetura definida.

### Desenvolvimento Back-end.

- O **back-end** do sistema será desenvolvido em Java e Kotlin, utilizando uma estrutura leve para criação de uma API REST. Essa API será responsável por gerenciar os dados dos usuários, agendamentos de corridas/fretes e o histórico de serviços. Toda a comunicação com o banco de dados MySQL será feita através desta API, garantindo que as informações fiquem centralizadas e organizadas. O foco aqui é criar um servidor simples, funcional e seguro, que atenda bem às necessidades do app.

### Desenvolvimento Front-end.

- No **front-end**, o aplicativo será feito de forma nativa para Android, usando Java ou Kotlin no Android Studio. A interface será projetada para ser prática e fácil de navegar, com telas bem definidas para cada tipo de usuário (motorista ou cliente). O uso do Google Maps vai permitir uma boa experiência visual e de localização, ajudando na hora de solicitar ou aceitar corridas.

### Integração.

- A **integração** entre front-end e back-end será feita via requisições HTTP, onde o app vai enviar e receber dados da API. Cada ação feita pelo usuário, como login, cadastro ou agendamento, será convertida em uma chamada para o servidor, que vai responder com as informações necessárias. A integração com o Google Maps também será feita diretamente no app, usando a SDK oficial para Android.

## **Entrada, processamento e saída de dados**

- Entradas: cadastro de usuário, login , criação de corridas, solicitação de reserva , edição de dados do usuário .
- Processamento: validação de login , requisições e respostas da api , Here maps
- Saídas: respostas JSON da API, atualizações de interface e notificações.

## **Projeto de interface gráfica**



Telas principais:

- Splash screen - apresentação do app
- Login - login no app
- Cadastro - cadastro de novo usuário
- Home cliente - tela principal após login específica para cliente
- Home motorista - tela principal após login específica para motorista
- Histórico - histórico de corridas para motorista e reservas para cliente
- Perfil - perfil para visualizar dados do usuário e editá-las.
- Cadastrar corrida - tela para cadastrar corrida
- Acompanhar corrida - tela para acompanhar a corrida
- Solicitar Reserva - tela para solicitação de reserva

A interface diferencia visualmente motoristas e clientes e usa HERE Maps SDK para exibir marcadores.

## Interface gráfica do usuário

O mapa exibe marcadores onde existem corridas ativas. Formulário de criação de corridas com origem/destino. Listas de corridas usando RecyclerView. Marcadores interativos com metadata.

 	<h3>Splash Screen</h3> <p>A tela splash screen do aplicativo tem a função de verificar se já existe um usuário logado. Caso exista, a lógica de autenticação será processada no back-end. Se não houver um usuário logado ou se ocorrer alguma falha na autenticação, o app redireciona o usuário para a tela de login.</p>
---	---

11:55

43%

LOGIN



Usuario

Digite seu usuario

Senha

Digite sua senha

Esqueceu sua senha?

Recuperar Senha

LOGIN

AINDA NÃO TEM CONTA?

CRIAR CONTA

## Login Screen

Na tela de login, o usuário pode acessar sua conta no aplicativo ou criar uma nova. A funcionalidade de recuperação de senha ainda não foi implementada.

11:55

43%

# Crie uma conta

Nome

Nome

Sobrenome

Sobrenome

Telefone

(DD)9NNNN-NNNN

Data de nascimento

dd/mm/aaaa

AVANÇAR

JÁ POSSUI UMA CONTA?

ENTRAR

## Criar conta Screen

A tela inicial de cadastro de novo usuário permite que ele informe seus dados básicos, como nome, sobrenome, telefone e data de nascimento.



The image shows a mobile application interface for creating a new account. At the top, there is a yellow header with the title "Crie uma conta". Below the header, there are four input fields: "Usuario" with the placeholder "usuario para login", "Senha" with masked characters "\*\*\*\*\*", "Confirme sua senha" with masked characters "\*\*\*\*\*", and "Email" with the placeholder "exemplo@moov.com". Below these fields is a grey button labeled "AVANÇAR". At the bottom, there is a link "JÁ POSSUI UMA CONTA?" and a light blue button labeled "ENTRAR". The status bar at the top shows the time "11:55" and battery level "43%".

11:55 43%

## Crie uma conta

**Usuario**

usuario para login

**Senha**

\*\*\*\*\*

**Confirme sua senha**

\*\*\*\*\*

**Email**

exemplo@moov.com

AVANÇAR

JÁ POSSUI UMA CONTA?

ENTRAR

### Criar conta 2 Screen

A tela Criar Conta 2 é responsável por coletar os dados de acesso do usuário, como nome de usuário, senha e e-mail. Antes de concluir o cadastro, o sistema realiza uma verificação no banco de dados para garantir que essas informações sejam únicas e não possuam vínculo prévio.

11:56

44%

Crie uma conta

Como voce pretende usar o App?

Motorista

Modelo do carro

ex: Corolla

Placa do veiculo

ex : ABC1D23

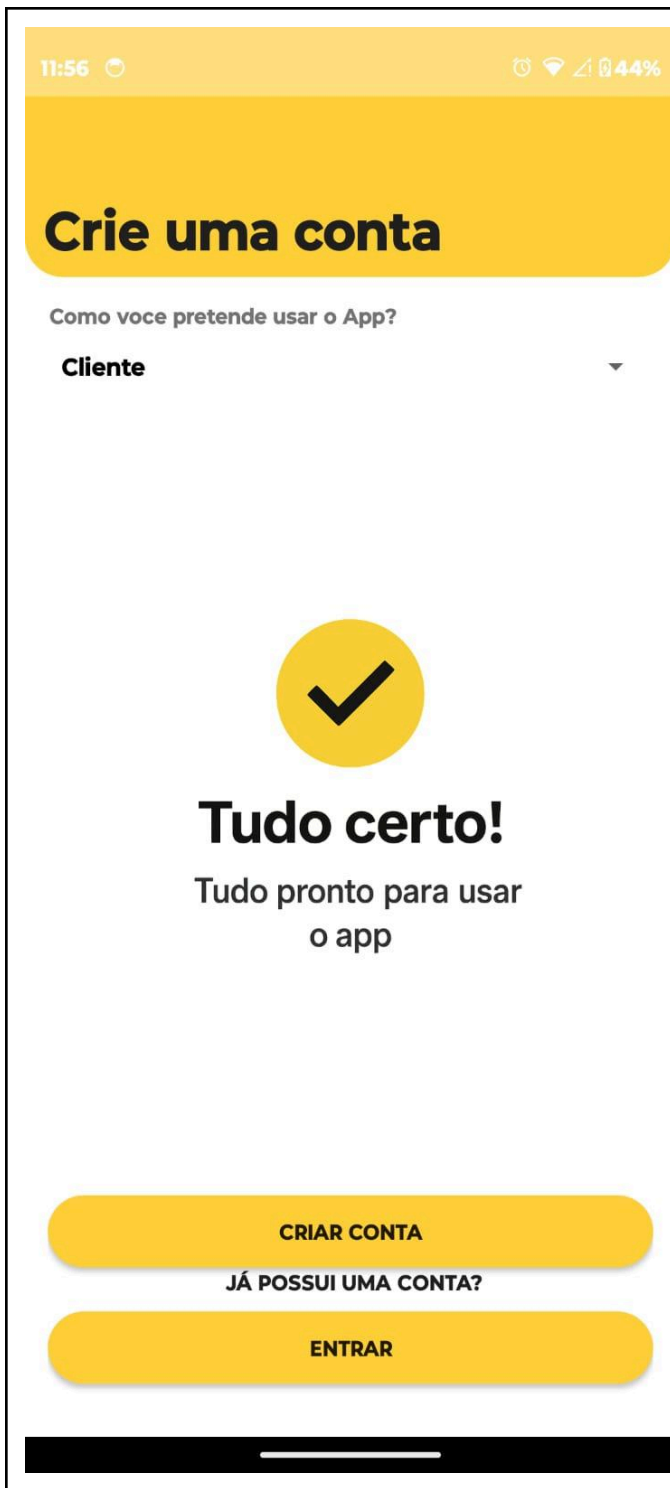
CRIAR CONTA

JÁ POSSUI UMA CONTA?

ENTRAR

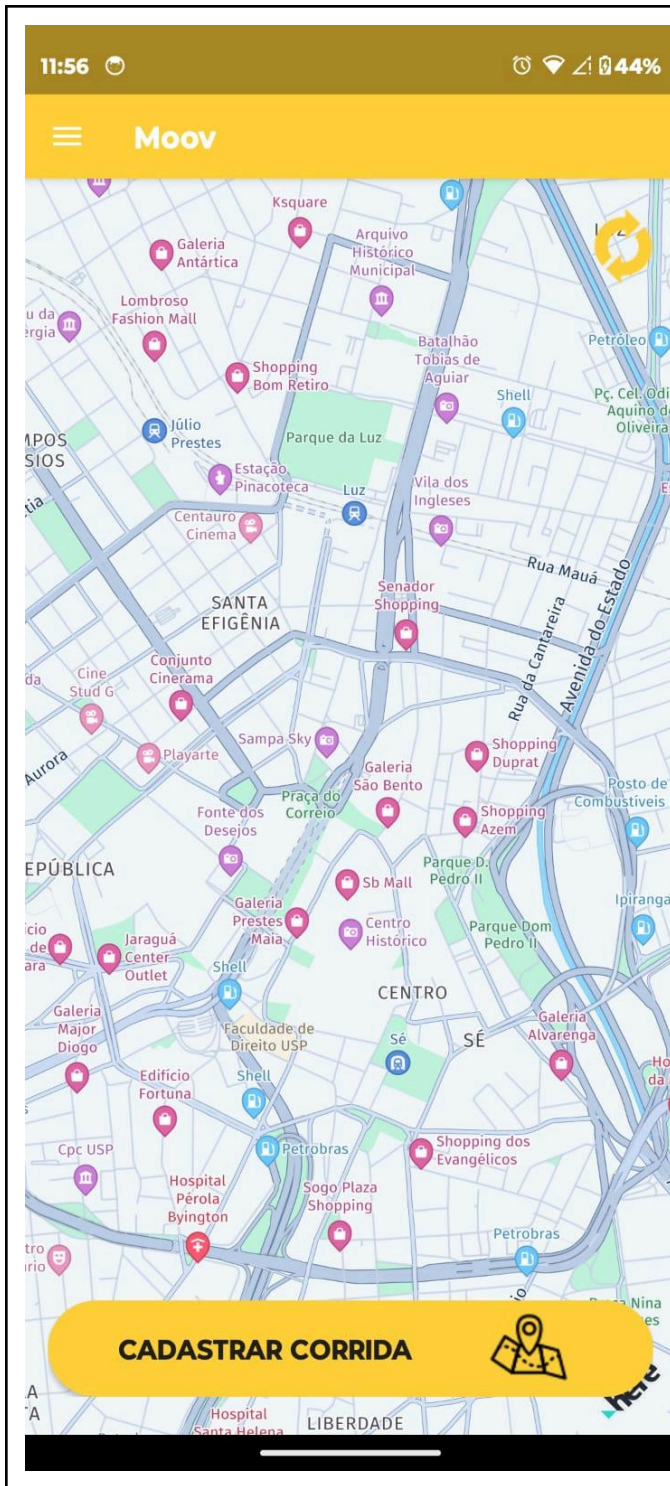
### Criar conta 3 Screen

Nesta tela, o usuário deve selecionar o tipo de conta, podendo escolher entre Motorista ou Cliente. Caso opte por ser Motorista, será necessário preencher dois campos adicionais: modelo do carro e placa do veículo.



### Criar conta 3 Screen

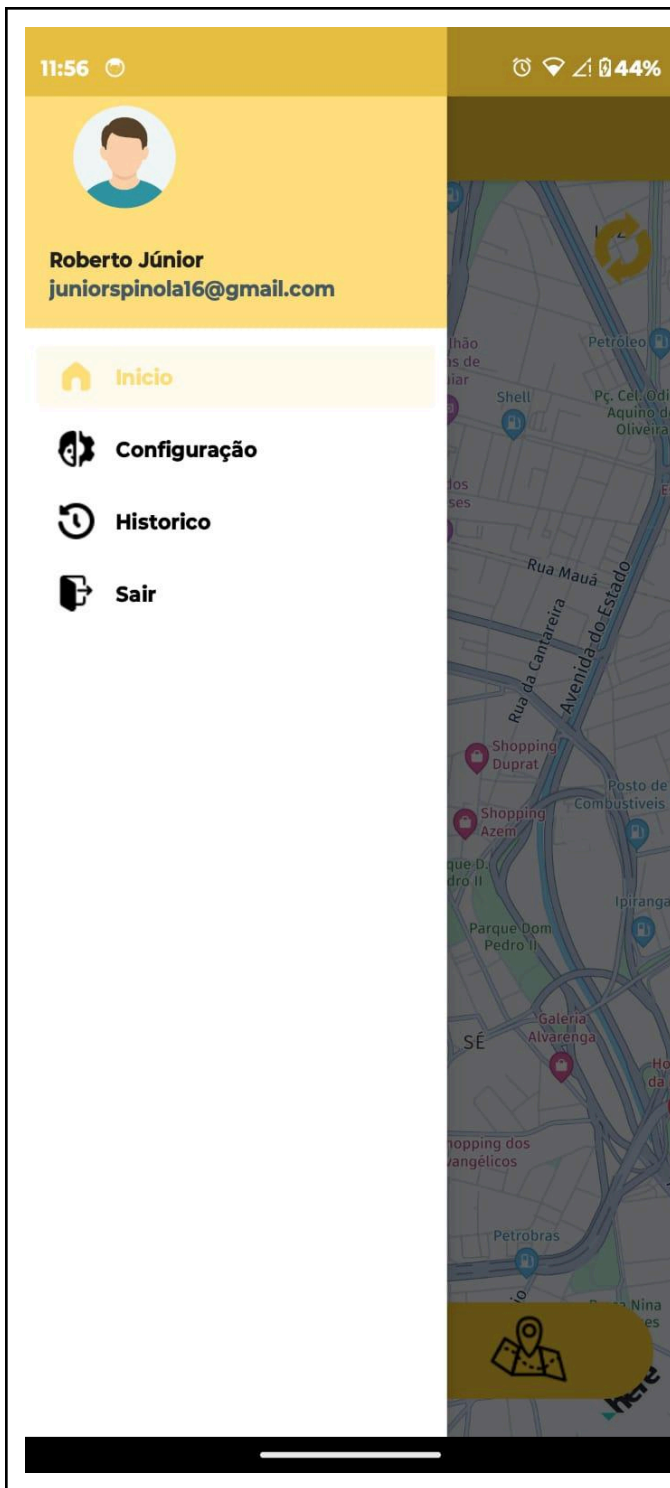
Caso o usuário escolha o tipo Cliente, não será necessário informar dados adicionais, podendo assim concluir o cadastro imediatamente.



## Home Motorista Screen

Na tela inicial do aplicativo, o conteúdo exibido varia de acordo com o tipo de usuário:

- Motorista: é apresentada a opção de cadastrar uma corrida e o mapa.
- Cliente: é exibido apenas o mapa, sem opções de cadastro de corrida.



## Navigation home

O menu de navegação apresenta informações do usuário, como nome, sobrenome e e-mail, além de oferecer acesso rápido a algumas funcionalidades:

- Tela inicial
- Configurações: permite visualizar e editar dados do perfil
- Histórico
- Sair da conta

11:57 44%

## Configurações

Alterar nome : Roberto

Alterar E-mail : juniorspinola16@gmail.com

Senha atual : Informe a senha atual

Nova senha : Informe a nova senha

Telefone : 75983717345

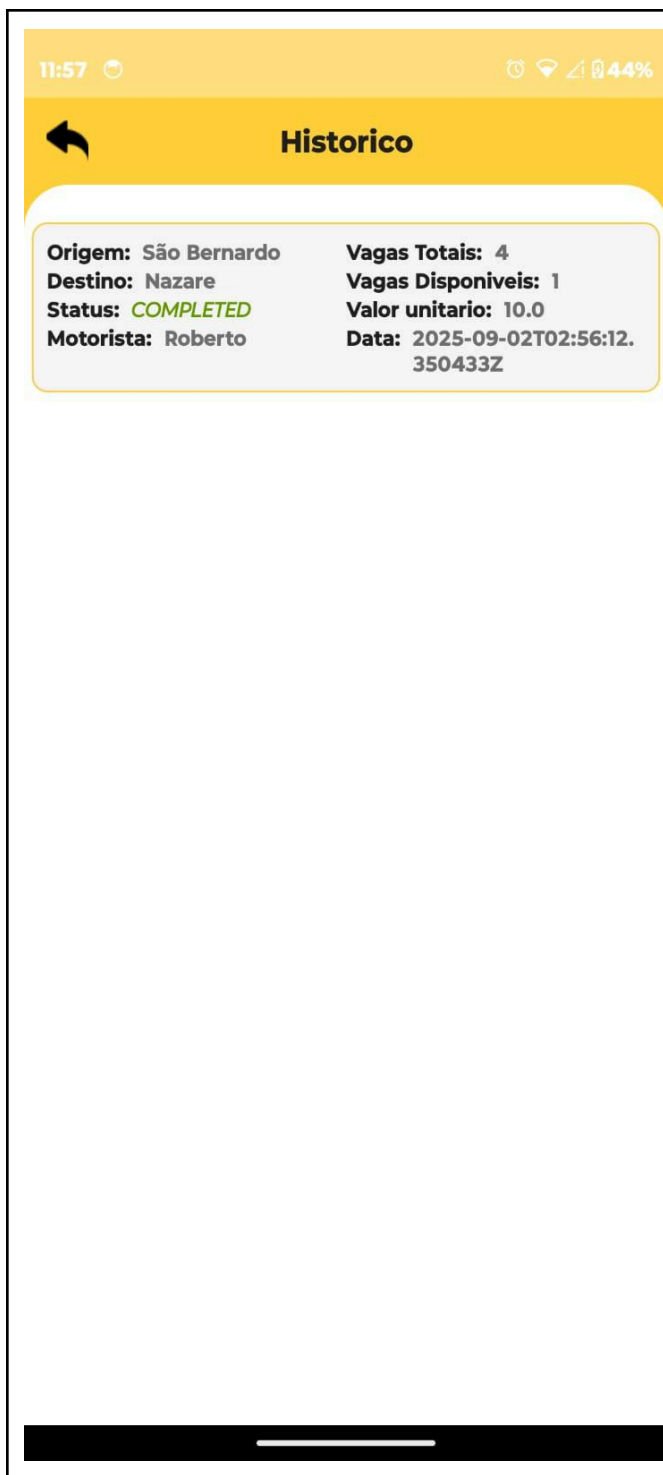
Placa do veículo : OZH7B20

Modelo do veículo : Grand Siena

SALVAR

### Perfil / Configurações Screen

Nesta tela, o usuário pode visualizar alguns de seus dados cadastrados, com a possibilidade de atualizá-los ou alterá-los conforme necessário.



## Historico Screen

- Motorista: pode visualizar todas as corridas já finalizadas.
- Cliente: pode visualizar todas as reservas realizadas.

11:57

44%

Moov

Cadastrar Corrida

Origem

Destino

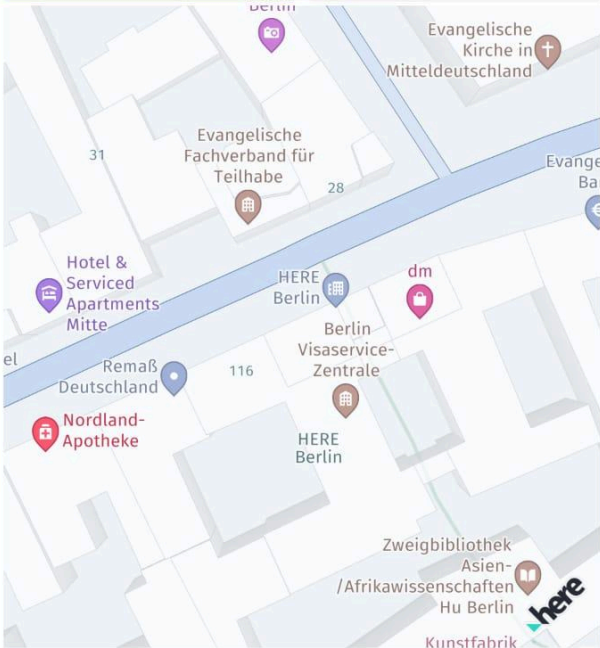
VagasPreço por vaga

2025-08-24T19:30:00Z

Descrição

VALIDAR

CRIAR VIAGEM



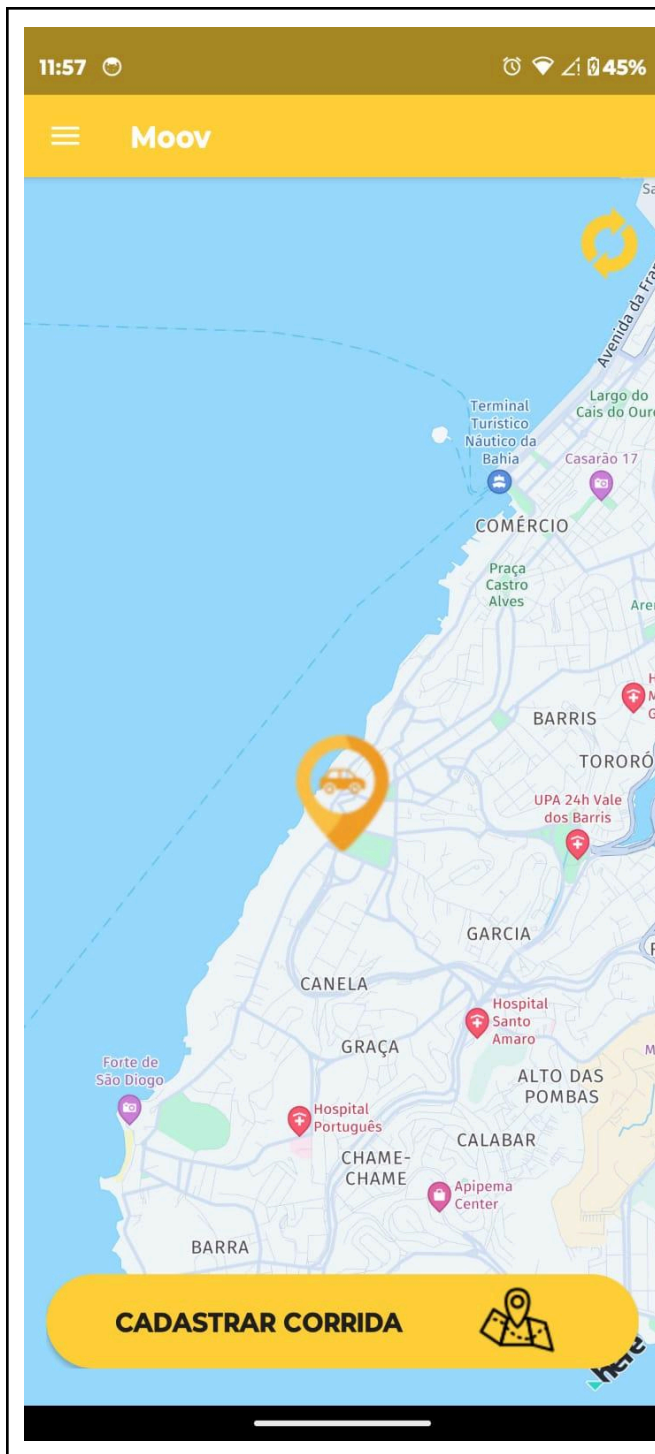
## Cadastrar Corrida Screen

Nesta tela, o **motorista** pode cadastrar uma corrida informando os seguintes dados:

- Origem
- Destino
- Número de vagas
- Preço por vaga
- Data e hora da viagem
- Descrição / Observações

Antes de criar a corrida, o usuário deve validar a **origem** e o **destino**, com a possibilidade de visualizá-los no mapa. Quanto mais detalhadas forem as informações sobre origem e destino, mais precisos ficarão os marcadores no mapa.





## Marcador Motorista

Após criar uma corrida com sucesso, no mapa do motorista serão exibidos marcadores apenas na posição de origem da corrida. Somente as corridas ativas vinculadas ao motorista logado aparecerão no mapa.

Ao tocar em um marcador, o usuário será direcionado para a tela de gerenciamento da corrida, onde poderá:

- Finalizar a corrida
- Cancelar a corrida
- Verificar as reservas realizadas

11:59

46%



## Gerenciamento

### Informações da Corrida

Origem: Salvador, BA, Brasil

Destino: Valença, BA, Brasil

Valor por vaga: R\$ 10.00

Vagas disponíveis: 0

Descrição: Salvador a Valença

### Reservas

Cliente: vanessa

Contato: 7598371745

Vagas Reservadas: 10

Valor Total: 100.0

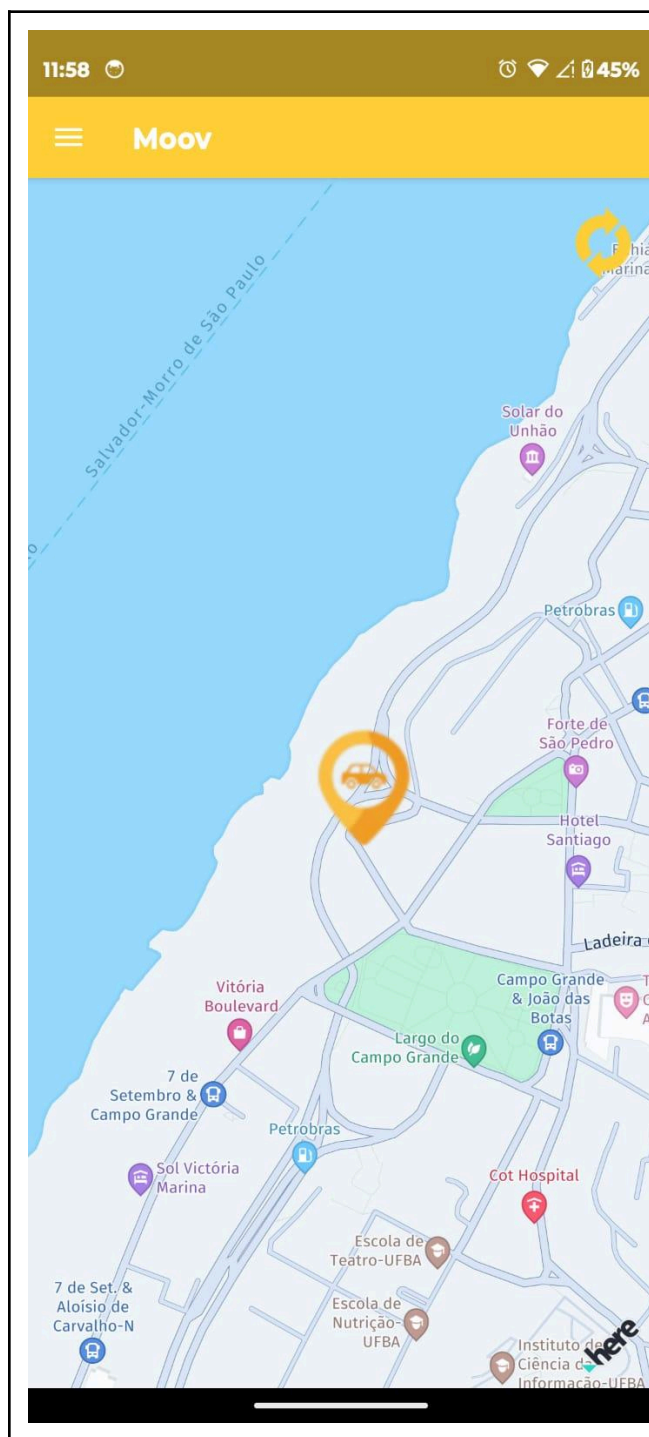
**FINALIZAR**

**CANCELAR**

## Gerenciamento de Corrida

Nesta tela, o motorista pode:

- Visualizar os dados da corrida e confirmar se estão corretos
- Verificar o número de vagas disponíveis
- Consultar as reservas realizadas
- Finalizar a corrida
- Cancelar a corrida



## Tela Inicial – Cliente

Nesta tela, o cliente visualiza no mapa todas as corridas ativas disponíveis no momento. Ao tocar em uma corrida, o usuário é redirecionado para a tela onde poderá realizar a reserva da corrida selecionada.

11:58 45%

**Moov**

**Reservar Viagem**

**Informações da Viagem**

Origem: Salvador, BA, Brasil

Destino: Valença, BA, Brasil

Valor por vaga: R\$ 10.00

Vagas disponíveis: 10

Descrição: Salvador a Valença

**Sua Reserva**

Vagas desejadas:  ex = 1

Valor total: R\$

**RESERVAR**

## Reservar viagem Screen

Nesta tela, o cliente pode visualizar informações da corrida, incluindo:

- Origem e destino
- Preço por vaga
- Número de vagas disponíveis
- Descrição da corrida

O usuário também pode reservar vagas tocando no botão Reservar. Caso a reserva seja bem-sucedida, será exibido um pop-up confirmando a ação. As reservas realizadas podem ser consultadas posteriormente na tela de histórico do cliente.

## **Funcionalidades do sistema**

- Autenticação e autorização.
- CRUD de usuários.
- Gerenciamento de corridas: criação, listagem, aceitação, atualização de status.
- Mapas e localização com HERE SDK.
- Notificações locais.

## **Considerações finais e próximos passos**

Por fim, o aplicativo cumpre o que foi proposto inicialmente, oferecendo uma solução leve e prática. Entretanto, é inegável que melhorias são necessárias, especialmente na interface do usuário. Recursos adicionais, como módulos para visualização de reservas e a possibilidade de cancelamento, devem ser implementados. Além disso, é fundamental realizar testes de segurança e garantir que toda a comunicação com a API esteja devidamente criptografada.

## Anexos

### Diagramas de Classe UML – API REST

O diagrama de classes UML, que representa o modelo objeto-relacional da aplicação, pode ser convertido em um diagrama de entidade-relacionamento (ER).

