Plano de Testes – Projeto Carona?

Introdução

O plano de testes do sistema **Carona?** tem como objetivo garantir a qualidade das funcionalidades críticas da aplicação voltada para o compartilhamento de caronas entre estudantes da PUC Minas. As funcionalidades serão avaliadas através de testes unitários, de integração, manuais e automatizados.

Arquitetura

- **Frontend Mobile**: React Native
- **Backend**: Java Spring Boot (API RESTful), Python (Matching das Caronas)
- **Banco de dados**: MySQL
- **Mensageria**: RabbitMQ
- **Armazenamento de imagens**: Supabase

Funcionalidades Testadas (Prioritárias)

Funcionalidade

Cadastro (Passageiro e Motorista)

Comportamento Esperado

O usuário poderá se cadastrar informando nome, e-mail, senha, confirmação de senha e papel (motorista ou passageiro). O sistema deve validar os campos e redirecionar para a tela de login em caso de sucesso.

Verificações

- Todos os campos obrigatórios preenchidos
- Validação da senha (mín. 6 caracteres)
- Confirmação de senha igual à senha
- Usuário já cadastrado
- Feedback visual (sucesso ou erro)

Critérios de Aceite

- Cadastro realizado com sucesso redireciona para login
- Mensagens de erro para campos inválidos ou repetidos
- Confirmação de senha obrigatória

Funcionalidade

Login (Passageiro e Motorista)

Comportamento Esperado

O usuário poderá acessar a plataforma com e-mail e senha válidos. Em caso de erro, deve receber feedback adequado.

Verificações

- Usuário e senha válidos
- Campo obrigatório não preenchido
- Senha incorreta
- Três tentativas inválidas consecutivas

Critérios de Aceite

- Login funcional com credenciais válidas
- Mensagens claras de erro
- Bloqueio temporário após 3 tentativas

Funcionalidade

Gerenciar Viagens (Motorista)

Comportamento Esperado

O motorista pode criar, editar, excluir ou encerrar viagens, informando data, horário, vagas.

Verificações

- Criação de nova viagem
- Edição de viagem existente
- Exclusão de viagem
- Validação de campos obrigatórios

Critérios de Aceite

- Viagens gerenciadas com sucesso
- Feedback visual após cada operação
- Campos obrigatórios validados corretamente

Funcionalidade

Gerenciar Passageiros (Motorista)

Comportamento Esperado

O motorista pode visualizar os passageiros cadastrados em sua viagem e aceitar ou remover cada um deles.

Verificações

- Lista de passageiros associada à viagem
- Aceitação de novos passageiros
- Remoção de passageiros

Critérios de Aceite

- Mudanças persistidas corretamente
- Interface atualizada com sucesso
- Mensagens de confirmação

Funcionalidade

Gerenciar Perfil (Motorista)

Comportamento Esperado

O motorista pode atualizar informações do seu perfil como nome, veículo, telefone e senha.

Verificações

- Validação de campos editáveis
- Alteração de senha com confirmação

- Atualização bem-sucedida ou falha

Critérios de Aceite

- Dados persistidos corretamente
- Feedback de sucesso ou erro
- Campos obrigatórios validados

Funcionalidade

Visualizar Mapa (Motorista)

Comportamento Esperado

O motorista pode visualizar a rota planejada e sua posição atual, utilizando a integração com o Google Maps.

Verificações

- Mapa carregado corretamente
- Exibição da rota traçada
- Localização atual visível

Critérios de Aceite

- Rotas exibidas com precisão
- Localização exibida

Funcionalidade

Aprovar Novos Usuários (Administrador)

Comportamento Esperado

O administrador poderá aprovar ou recusar cadastros pendentes.

Verificações

- Listagem de usuários pendentes
- Aprovação individual
- Rejeição individual

Critérios de Aceite

- Estado do usuário atualizado corretamente
- Interface de aprovação funcional
- Lista de pendentes na tela atualizada
- Lista de usuários cadastrados atualizada

Funcionalidade

Gerenciar Usuários (Administrador)

Comportamento Esperado

O administrador poderá visualizar, editar ou excluir usuários já aprovados.

Verificações

- Listagem de todos os usuários
- Exclusão de usuários
- Edição de informações do perfil

Critérios de Aceite

- Operações salvas corretamente
- Feedback adequado

Estratégia de Teste

- **Testes Unitários**: cobertura mínima de 60% no backend (Spring Boot).
- **Testes de Integração**: serão realizados para os principais fluxos de negócio da API REST.
- **Testes Automatizados**: uso de Cypress para testes E2E das funcionalidades de login e cadastro.
- **Testes Manuais**: todos os fluxos serão testados manualmente com base em cenários descritos.
- **Versão Beta**: liberada para 3 usuários internos antes do lançamento final.

Ambiente e Ferramentas

Ferramenta	Time	Descrição
Insomnia/Swagger	Qualidade	Ferramenta para realização de testes de API
Jest	Desenvolvimento	Framework utilizada para testes unitários
Cypress	Qualidade	Ferramenta para testes end-to-end
Lighthouse	Desenvolvimento	Avaliação de performance e acessibilidade da aplicação
Gravador de Passos	Desenvolvimento	Prover evidências dos testes

---

Classificação de Bugs

ID	Nivel de Severidade	Descrição
1	Blocker	● Bug que bloqueia o teste de uma função ou feature causa crash na aplicação. ● Botão não funciona impedindo o uso completo da funcionalidade. ● Bloqueia a entrega.
2	Grave	● Funcionalidade não funciona como o esperado ● Input incomum causa efeitos irreversíveis
3	Moderada	● Funcionalidade não atinge certos critérios de aceitação, mas sua funcionalidade em geral não é afetada ● Mensagem de erro ou sucesso não é exibida
4	Pequena	● Quase nenhum impacto na funcionalidade porém atrapalha a experiência ● Erro ortográfico ● Pequenos erros de UI

Definição de Pronto

Uma funcionalidade será considerada **Pronta** quando:

- Todos os testes definidos forem realizados e aprovados;
- Nenhum bug classificado como Blocker ou Grave permanecer aberto;
- A validação de negócio for concluída pelo time de produto.
