**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 end-to-end.

**Arquitetura**

* **Frontend Mobile**: React Native (Expo)
* **Frontend Web**: React + Vite + TypeScript
* **Backend**: Java Spring Boot (API RESTful)
* **Sistema de Matching**: Python
* **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)
* Validação de formato de e-mail
* Prevenção de cadastro duplicado

**Critérios de Aceite**

* Cadastro realizado com sucesso redireciona para login
* Mensagens de erro para campos inválidos ou repetidos
* Confirmação de senha obrigatória
* E-mail deve ter formato válido
* Dados persistidos corretamente no banco

**Cenários de Teste**

1. **Cadastro com dados válidos**: Preencher todos os campos corretamente e verificar redirecionamento
2. **Cadastro com senha fraca**: Inserir senha com menos de 6 caracteres
3. **Cadastro com senhas divergentes**: Confirmar senha diferente da senha principal
4. **Cadastro com e-mail inválido**: Inserir e-mail sem formato correto
5. **Cadastro duplicado**: Tentar cadastrar e-mail já existente

**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
* E-mail não cadastrado
* Três tentativas inválidas consecutivas
* Persistência de sessão
* Logout funcional

**Critérios de Aceite**

* Login funcional com credenciais válidas
* Mensagens claras de erro
* Bloqueio temporário após 3 tentativas
* Redirecionamento para dashboard após login
* Session management adequado

**Cenários de Teste**

1. **Login com credenciais válidas**: Verificar acesso e redirecionamento
2. **Login com senha incorreta**: Verificar mensagem de erro
3. **Login com e-mail não cadastrado**: Verificar tratamento adequado
4. **Login com campos vazios**: Verificar validação obrigatória
5. **Bloqueio após tentativas**: Verificar bloqueio após 3 tentativas falhas
6. **Teste de sessão**: Verificar persistência de login

**Funcionalidade: Gerenciar Viagens (Motorista)**

**Comportamento Esperado** O motorista pode criar, editar, excluir ou encerrar viagens, informando data, horário, origem, destino e número de vagas.

**Verificações**

* Criação de nova viagem
* Edição de viagem existente
* Exclusão de viagem
* Validação de campos obrigatórios
* Validação de data/hora futura
* Limitação de vagas conforme capacidade do veículo
* Status da viagem

**Critérios de Aceite**

* Viagens gerenciadas com sucesso
* Feedback visual após cada operação
* Campos obrigatórios validados corretamente
* Apenas motorista proprietário pode editar/excluir
* Datas no passado rejeitadas

**Cenários de Teste**

1. **Criar viagem válida**: Preencher todos os campos obrigatórios
2. **Criar viagem com data passada**: Verificar rejeição
3. **Editar viagem existente**: Modificar dados e verificar persistência
4. **Excluir viagem sem passageiros**: Verificar exclusão bem-sucedida
5. **Tentar editar viagem de outro motorista**: Verificar bloqueio
6. **Criar viagem com vagas acima da capacidade**: Verificar validação

**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
* Controle de vagas disponíveis
* Notificações para passageiros

**Critérios de Aceite**

* Mudanças persistidas corretamente
* Interface atualizada com sucesso
* Mensagens de confirmação
* Vagas controladas adequadamente
* Histórico de ações mantido

**Cenários de Teste**

1. **Aceitar passageiro**: Verificar atualização de vagas
2. **Remover passageiro**: Verificar liberação de vaga
3. **Tentar aceitar acima da capacidade**: Verificar bloqueio
4. **Visualizar lista vazia**: Verificar comportamento sem passageiros
5. **Aceitar múltiplos passageiros**: Verificar controle de vagas

**Funcionalidade: Buscar Caronas (Passageiro)**

**Comportamento Esperado** O passageiro pode buscar caronas disponíveis por origem, destino, data e horário, visualizando informações do motorista e da viagem.

**Verificações**

* Filtros de busca funcionais
* Resultados precisos conforme critérios
* Informações completas da carona
* Solicitação de participação
* Status da solicitação

**Critérios de Aceite**

* Busca retorna resultados relevantes
* Filtros aplicados corretamente
* Interface responsiva e intuitiva
* Solicitações processadas adequadamente

**Cenários de Teste**

1. **Busca com filtros válidos**: Verificar resultados precisos
2. **Busca sem resultados**: Verificar mensagem adequada
3. **Solicitar participação**: Verificar envio de solicitação
4. **Busca com data passada**: Verificar tratamento
5. **Filtros múltiplos**: Verificar combinação de critérios

**Funcionalidade: Gerenciar Perfil**

**Comportamento Esperado** O usuário pode atualizar informações do seu perfil como nome, telefone, informações do veículo (motorista) e senha.

**Verificações**

* Validação de campos editáveis
* Alteração de senha com confirmação
* Upload de foto de perfil
* Atualização bem-sucedida ou falha
* Campos específicos por tipo de usuário

**Critérios de Aceite**

* Dados persistidos corretamente
* Feedback de sucesso ou erro
* Campos obrigatórios validados
* Senha alterada com segurança
* Foto salva adequadamente

**Cenários de Teste**

1. **Atualizar informações básicas**: Nome, telefone
2. **Alterar senha**: Com confirmação válida
3. **Upload de foto**: Verificar armazenamento
4. **Campos obrigatórios**: Validar preenchimento
5. **Informações de veículo (motorista)**: Placa, modelo, capacidade

**Funcionalidade: Visualizar Mapa**

**Comportamento Esperado** O usuário pode visualizar a rota planejada e localização atual, utilizando a integração com Google Maps.

**Verificações**

* Mapa carregado corretamente
* Exibição da rota traçada
* Localização atual visível
* Marcadores de origem e destino
* Tempo estimado de viagem

**Critérios de Aceite**

* Rotas exibidas com precisão
* Localização GPS funcional
* Interface responsiva
* Integração Google Maps estável

**Cenários de Teste**

1. **Carregar mapa inicial**: Verificar carregamento
2. **Traçar rota**: Entre origem e destino
3. **Localização atual**: Verificar GPS
4. **Zoom e navegação**: Funcionalidades do mapa
5. **Marcadores**: Origem, destino e pontos intermediários

**Funcionalidade: Aprovar Novos Usuários (Administrador)**

**Comportamento Esperado** O administrador poderá aprovar ou recusar cadastros pendentes de motoristas e passageiros.

**Verificações**

* Listagem de usuários pendentes
* Aprovação individual
* Rejeição individual
* Filtros por tipo de usuário
* Histórico de aprovações

**Critérios de Aceite**

* Estado do usuário atualizado corretamente
* Interface de aprovação funcional
* Lista de pendentes atualizada
* Lista de usuários cadastrados atualizada
* Notificações enviadas aos usuários

**Cenários de Teste**

1. **Aprovar usuário válido**: Verificar mudança de status
2. **Rejeitar usuário**: Verificar remoção da lista
3. **Aprovação em lote**: Múltiplos usuários
4. **Filtrar por tipo**: Motorista/Passageiro
5. **Histórico de ações**: Verificar registro

**Funcionalidade: Gerenciar Usuários (Administrador)**

**Comportamento Esperado** O administrador poderá visualizar, editar ou excluir usuários já aprovados, além de acessar relatórios e métricas.

**Verificações**

* Listagem de todos os usuários
* Exclusão de usuários
* Edição de informações do perfil
* Relatórios de uso
* Métricas de engajamento

**Critérios de Aceite**

* Operações salvas corretamente
* Feedback adequado
* Relatórios precisos
* Interface administrativa completa

**Cenários de Teste**

1. **Listar usuários**: Verificar paginação e filtros
2. **Editar usuário**: Modificar informações
3. **Excluir usuário**: Verificar remoção segura
4. **Gerar relatórios**: Verificar dados
5. **Visualizar métricas**: Dashboard administrativo

**Estratégia de Teste**

**Testes Unitários**

* **Cobertura mínima**: 70% no backend (Spring Boot)
* **Ferramentas**: JUnit 5, Mockito
* **Foco**: Lógica de negócio, validações, serviços

**Testes de Integração**

* **Escopo**: Principais fluxos de negócio da API REST
* **Ferramentas**: TestContainers, MockMvc
* **Foco**: Integração entre camadas, persistência

**Testes E2E Automatizados**

* **Mobile**: Maestro para fluxos críticos
* **Web**: Cypress para interface administrativa
* **Foco**: Jornadas completas do usuário

**Testes Manuais**

* **Escopo**: Todos os fluxos com cenários exploratórios
* **Documentação**: Casos de teste detalhados
* **Execução**: Antes de cada release

**Testes de Performance**

* **Ferramentas**: Lighthouse, JMeter
* **Métricas**: Tempo de resposta, throughput
* **Critérios**: < 2s para carregamento de telas

**Versão Beta**

* **Usuários**: 5 usuários internos
* **Duração**: 2 semanas
* **Feedback**: Formulário estruturado

**Ambiente e Ferramentas**

| **Ferramenta** | **Time** | **Descrição** |
| --- | --- | --- |
| **Jest** | Desenvolvimento | Framework para testes unitários (Backend/Frontend) |
| **Insomnia/Swagger** | Qualidade | Ferramentas para testes de API |
| **Maestro** | Qualidade | Testes E2E para aplicativo mobile |
| **Cypress** | Qualidade | Testes E2E para frontend web |
| **TestContainers** | Desenvolvimento | Testes de integração com banco real |
| **Lighthouse** | Desenvolvimento | Avaliação de performance e acessibilidade |
| **JMeter** | Qualidade | Testes de carga e performance |
| **Gravador de Passos** | Desenvolvimento | Evidências dos testes manuais |

**Classificação de Bugs**

| **ID** | **Nível de Severidade** | **Descrição** |
| --- | --- | --- |
| **1** | **Blocker** | • Bug que bloqueia o teste de uma função ou feature, causa crash na aplicação<br>• Botão não funciona impedindo o uso completo da funcionalidade<br>• Bloqueia a entrega<br>• Vulnerabilidades de segurança críticas |
| **2** | **Grave** | • Funcionalidade não funciona como o esperado<br>• Input incomum causa efeitos irreversíveis<br>• Perda de dados<br>• Falhas de integração críticas |
| **3** | **Moderada** | • Funcionalidade não atinge certos critérios de aceitação, mas sua funcionalidade em geral não é afetada<br>• Mensagem de erro ou sucesso não é exibida<br>• Problemas de usabilidade significativos |
| **4** | **Pequena** | • Quase nenhum impacto na funcionalidade, porém atrapalha a experiência<br>• Erro ortográfico<br>• Pequenos erros de UI<br>• Problemas cosméticos |

**Critérios de Qualidade**

**Cobertura de Código**

* **Backend**: Mínimo 70%
* **Frontend**: Mínimo 20%
* **Críticos**: Mínimo 90%