Planejamento de Requisitos - CrashGuardian

1. Objetivo da Aplicação

Descreva o propósito da aplicação.

Melhorar o sistema de emergência com o foco em ocorrências de acidentes de trânsito.

Para isso, primeiro, pense no problema. Qual problema ela resolve?

Resposta de emergência mais rápida, Congestionamento, Segurança no trânsito.

Quem são os usuários-alvo?

Acima de 18 anos com habilitação.

Problema:

Acidentes de trânsito têm um impacto significativo na sociedade, tanto em termos de custos econômicos quanto humanos. Todos os anos, milhões de pessoas são feridas ou mortas em acidentes de trânsito em todo o mundo, tornando-se um problema urgente de saúde pública e segurança.

Solução (Propósito da Aplicação):

Ter uma aplicação para dispositivos mobile para ajudar os órgãos de segurança no acionamento e deslocamento de ocorrências relacionadas ao trânsito.

Usuários-Alvo:

- Pessoas acima de 18 anos com habilitação.

2. Funcionalidades Principais

Liste as principais funcionalidades que a aplicação deve oferecer.

- Autenticação:
- Cadastro de usuários.

- Login e Logout.
- Recuperação de senha.

- Cadastro veicular automotor.

- **Funcionalidades para Usuários:**
- Acionamento de socorro.
- Acionamento de contato de emergência pessoal.
- GPS.
- Localização em tempo real.

- Administração:
- Gerenciar usuários.
- Monitorar acidentes de trânsito.

3. Requisitos Não Funcionais

Estes são os aspectos que afetam o desempenho, a segurança e a experiência geral da aplicação.

- Performance:
- Tempo de resposta de até 2 segundos para ações de envio de mensagens.
- Suporte a até 1000 usuários simultâneos.

- Segurança:
- Criptografia de senhas.
- Controle de acesso seguro para usuários e administradores.
- Proteção contra ataques comuns (ex.: SQL Injection, XSS).

- Compatibilidade:
- Compatível com Android e IOS.
- Responsividade para dispositivos móveis.

4. Requisitos Técnicos

Nesta parte, você deve definir as tecnologias que serão utilizadas, tanto no frontend quanto no backend, além do banco de dados e hospedagem.

- Frontend:
- Framework: React ou Vue.js.
- Biblioteca de componentes: Material UI ou Bootstrap.
- Responsividade: CSS Grid/Flexbox.

- Backend:
- Linguagem: Node.js com Express.js.
- APIs: REST (ou GraphQL).

- Banco de Dados:
  - Tipo: Relacional (MySQL ou PostgreSQL) ou NoSQL (MongoDB).
  - Estrutura: Tabelas ou coleções para usuários, mensagens, etc.

- Hospedagem e Infraestrutura:
  - Hospedagem: AWS, DigitalOcean ou Heroku.
  - Banco de Dados: RDS ou serviço gerenciado de banco de dados.
  - CDN: Cloudflare para acelerar a entrega de conteúdo.

5. Definição das APIs

Estas são as APIs que a aplicação vai fornecer para que o frontend se comunique com o backend. Elas permitem que os usuários interajam com o sistema (ex.: login, envio de mensagens). Se você ainda não aprendeu a respeito, deixe esta parte em branco.

5.1 APIs Internas

Método	Rota	Descrição	Parâmetros	Resposta
POST	/login	Autenticar usuário	{ email, senha }	Token de autenticação
POST	/register	Registrar novo usuário	{ nome, email, senha }	Confirmação de registro
GET	/messages	Obter mensagens recebidas	Token de autenticação	Lista de mensagens
POST	/messages	Enviar nova mensagem	{ conteúdo, desenho }	Confirmação de envio
PUT	/profile	Atualizar informações de perfil	{ nome, preferências }	Perfil atualizado

5.2 APIs de Terceiros (Externas)

- API de Autenticação (ex.: Auth0, Firebase Auth): Facilita o login seguro dos usuários.

- API de Armazenamento de Arquivos (ex.: AWS S3, Firebase Storage): Armazena imagens ou desenhos.

- API de Envio de Emails (ex.: SendGrid, Mailgun): Envia emails de confirmação ou recuperação.

- API de Geolocalização (ex.: Google Maps API): Mostra ou verifica localização de usuários.

6. Modelo de Dados

Nesta parte, defina as tabelas ou coleções que serão usadas no banco de dados. Vocês devem planejar o banco como nos modelos abaixo, criar o Diagrama de

Entidade-Relacionamento e, após, o Modelo Físico (ou seja, os comandos para criar o banco utilizando o SGBD escolhido)

Tabela de Usuários

Cadastro	Tipo	Descrição
id	INT	Identificador unico do usuário
nome	VARCHAR(50)	Nome do usuario
Sobrenome	VARCHAR(100)	Sobrenome do usuario
CPF	VARCHAR(255)	CPF do usuario
Data de nascimento	DATETIME	Data de nascimento do usuario
Genero	VACRCHAR(20)	Genero do usuario
Telefone	INT	Numero do usuario
Senha	VARCHAR(100)	Senha criptografada

Tabela de Carro User

Campo	Tipo	Descrição
id	INT	Identificador único da mensagem
Marca	VARCHAR(40)	Marca do veiculo do usuario
Modelo	VARCHAR(30)	Modelo do veículo do usuário
Ano	INT	Ano do veiculo
Versão	VARCHAR(20)	Versão veicular
Cor	VARCHAR(20)	Cor veicular
Placa	VARCHAR(10)	Placa veicular

Tabela Emergencia

Campo	Tipo	Descrição
id	INT	Identificador unico da mensagem
Police_Number	INT	Numero policia
PRF_Number	INT	Numero PRF
Samu_Number	INT	Numero Samu
Bombeiro	INT	Numero Bombeiro

Tabela de Emergencia User

Campo	Descrição	Tempo Estimado
ID	INT	Identificador unico da mensagem

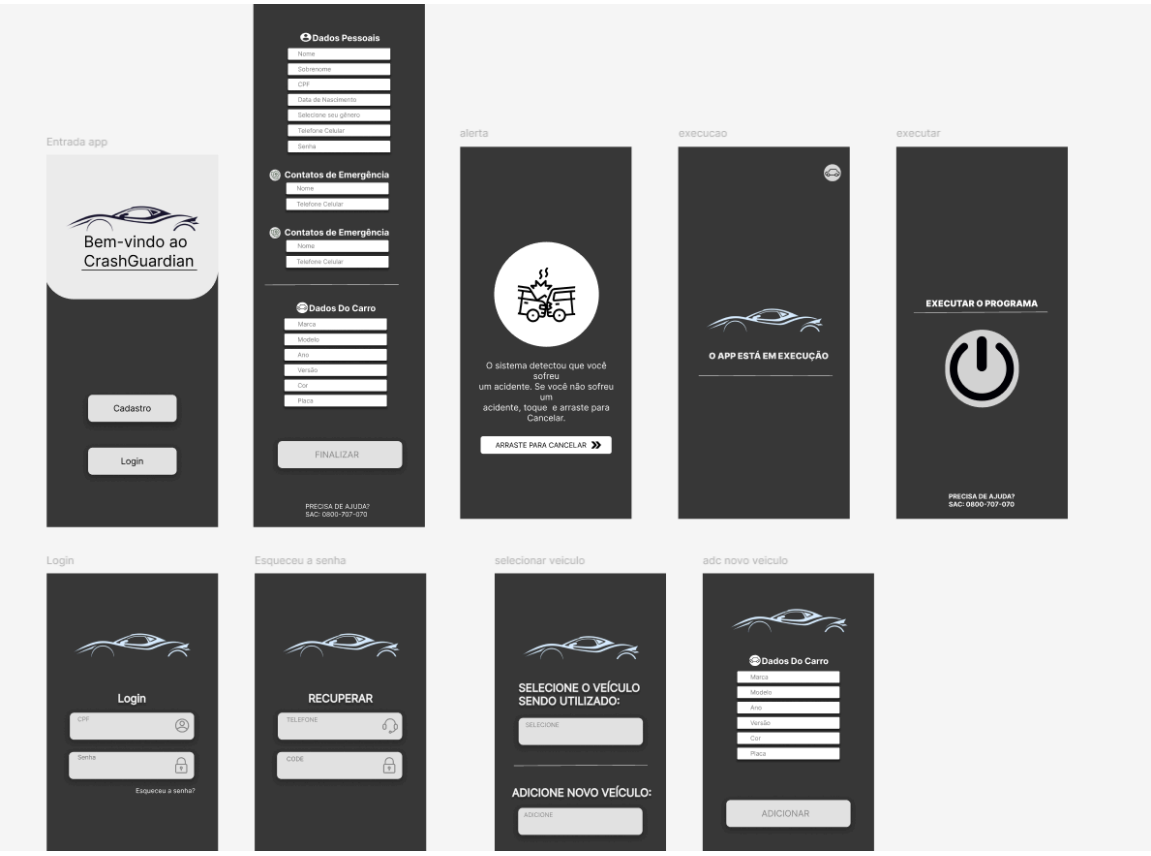
Nome	VARCHAR(100)	Nome do contato de emergencia do usuario
Telefone	INT	Numero de emergencia do contato usuário

7. Protótipos Visuais

Liste ferramentas ou links de protótipos visuais que mostram como será o layout da aplicação. Se ainda não tem, deixe esta parte em branco.

- Figma:
https://www.figma.com/design/up5TCalNNbUoSEMFWpfXwr/CrashGuardian?m=auto&t=bPuVOAAjUvc726hw-6

- Wireframes:



8. Cronograma de Desenvolvimento

Defina um cronograma com as fases do projeto e estimativas de tempo para cada etapa. Esta parte pode ser deixada em branco no momento.

Fase	Descrição	Tempo Estimado
Planejamento	Definição de requisitos e arquitetura	1 semana
Desenvolvimento do Backend	Implementação das APIs e banco de dados	3 semanas
Desenvolvimento do Frontend	Implementação do frontend e integração	3 semanas
Testes	Testes funcionais e ajustes	1 semana
Implantação	Deploy da aplicação em produção	1 semana

9. Equipe

Defina quem ficará responsável pelo que no projeto.

- Gestor de Projeto: Gabriel da Silva.

- Desenvolvedor Backend: Pedro e Heitor.

- Desenvolvedor Frontend: Brunna Paganini e Gabriel da Silva.

- Designer UX/UI: Brunna, Gabriel, Heitor e Pedro.

10. Documentação Técnica

Lembre-se de criar documentação para que os futuros desenvolvedores possam entender o projeto facilmente. Também pode ser deixada em branco no momento.

- Documentação das APIs.

- Explicação da estrutura do banco de dados.

- Guia de configuração e deploy.