UNIVERSIDADE ESTADUAL DE MARINGÁ
INFORMÁTICA - GRADUAÇÃO

GUSTAVO BONASSOLI	125235
INGRID LOUISE PEREIRA LOHMANN	117698
JOSÉ VINÍCIUS PEREIRA POLETTO	115971
LARISSA EMY KASSUYA	102452
LUIZ CLAUDIO NAGAHAMA	64456

TECNOLOGIAS ESCOLHIDAS

MARINGÁ
2025

O sistema será uma aplicação web desenvolvida em React, implementando design responsivo para garantir usabilidade em múltiplos dispositivos, com ênfase na experiência mobile devido ao perfil majoritário de acessos via smartphones. A equipe está atualmente avaliando a viabilidade técnica e aderência ao padrão de design entre as bibliotecas Chakra UI e Material Design 3, visando selecionar a solução mais adequada para garantir consistência visual, acessibilidade e performance. O backend será desenvolvido em NestJS, fornecendo uma arquitetura modular, escalável e alinhada às melhores práticas para APIs RESTful.

Tecnologias do Frontend

- **Linguagem: TypeScript**
  TypeScript é um superset do JavaScript que adiciona tipagem estática ao código. Essa característica proporciona maior segurança, facilita a detecção de erros em tempo de desenvolvimento e melhora a manutenibilidade e escalabilidade do projeto. Sua integração com React e Vite é sólida e amplamente adotada no mercado.

- **Biblioteca: React**
  Biblioteca JavaScript declarativa e baseada em componentes reutilizáveis, que permite o desenvolvimento de interfaces de usuário interativas e eficientes. Possui grande comunidade, documentação rica e vasto ecossistema de ferramentas.

- **Ferramenta de build/empacotamento: Vite**
  Ferramenta moderna que oferece ambiente de desenvolvimento otimizado, com carregamento rápido (hot reload) e build mais eficiente em comparação a ferramentas tradicionais como Webpack.

- **Gerenciador de pacotes: Yarn**
  Gerenciador de pacotes rápido e confiável, com suporte a funcionalidades avançadas como cache offline e workspaces. Foi escolhido para manter consistência com o backend e facilitar o gerenciamento de dependências.

- **Cliente HTTP: Axios**
  Biblioteca leve e baseada em Promises, que simplifica o envio de requisições HTTP e o tratamento de respostas. Oferece recursos como interceptadores, cancelamento de requisições e tratamento centralizado de erros.

- **Framework de testes: Jest (possivelmente)**
  Framework de testes mantido pelo time do React. Permite a criação de testes unitários e de integração com facilidade, além de recursos como mocks automáticos e cobertura de código.

- **Biblioteca de rotas: React Router**
  Solução padrão para roteamento em aplicações React. Suporta rotas aninhadas, navegação programática e estrutura declarativa, permitindo uma navegação dinâmica e eficiente.

- **Controle de versão: Git**
  Ferramenta essencial para versionamento de código, controle de mudanças, colaboração em equipe e integração com fluxos de CI/CD.

- **Linting: ESLint**
  Ferramenta de análise estática que ajuda a manter o código limpo e consistente, detectando erros comuns e incentivando boas práticas de programação.

- **Formatação de código: Prettier**
  Formata automaticamente o código-fonte, garantindo padronização entre desenvolvedores e facilitando a leitura e revisão do código.

- **Ganchos de pré-commit: Husky**
  Permite a execução de scripts antes dos commits, como verificação de lint e testes, prevenindo a inclusão de código com falhas no repositório.

- **Hospedagem: Vercel**
  Plataforma especializada para deploy de aplicações frontend modernas, como projetos com React e Vite. Oferece integração contínua com GitHub, deploy automático e preview de branches.

- **Integração Contínua (CI): GitHub Actions**
  Ferramenta de CI integrada ao GitHub, utilizada para automatizar testes, builds e deploys. Garante maior confiabilidade no processo de desenvolvimento e facilita a detecção de erros.

- **Biblioteca de estilos: Chakra UI / Material Design 3**
  A equipe está avaliando qual das bibliotecas será mais adequada ao projeto.

  ○ *Chakra UI:* Biblioteca React acessível e modular, com suporte nativo a temas, dark mode e sistema de design responsivo baseado em props, facilitando a criação de interfaces consistentes e customizáveis.

  ○ *Material Design 3:* Implementação oficial do padrão de design do Google, com foco em acessibilidade, responsividade e experiência do usuário em dispositivos móveis, por meio de componentes prontos e bem estruturados.

Tecnologias do Backend

- **Linguagem: TypeScript**
  Superset do JavaScript com tipagem estática opcional, que aumenta a segurança do código e reduz erros em tempo de execução. Proporciona melhor escalabilidade e manutenção do projeto.

- **Framework: NestJS**
  Framework Node.js robusto baseado em arquitetura modular, inspirado no Angular. Ideal para o desenvolvimento de APIs escaláveis, organizadas e com alta manutenibilidade.

- **Gerenciador de pacotes: Yarn**
  Utilizado também no backend para manter consistência com o frontend e aproveitar os mesmos benefícios de performance e organização de dependências.

- **Framework de testes: Jest**
  Já integrado ao NestJS, é utilizado para testes unitários, de integração e mocks. Permite testes automatizados e suporte a TDD (Test-Driven Development).

- **Validação de dados:** `class-validator, class-transformer`
  Bibliotecas padrão no NestJS para validação e transformação de dados recebidos via DTOs (Data Transfer Objects), garantindo integridade e consistência na comunicação com o backend.

- **Documentação automática da API:** `@nestjs/swagger`
  Gera automaticamente documentação interativa da API utilizando Swagger UI, facilitando testes e integração entre frontend e backend.

- **Controle de versão: Git**
  Permite versionamento seguro, controle colaborativo de código e integração com pipelines de CI/CD, assim como no frontend.

- **Linting: ESLint**
  Utilizado para manter a qualidade e padronização do código, evitando más práticas e erros comuns.

- **Formatação de código: Prettier**
  Garante a formatação consistente do código, facilitando revisões e legibilidade em equipe.

- **Ganchos de pré-commit: Husky**
  Executa scripts como lint e testes antes de cada commit, ajudando a preservar a qualidade do código entregue ao repositório.

- **Hospedagem: Railway**
  Plataforma moderna para hospedagem de aplicações backend com Node.js e NestJS. Oferece suporte a deploy contínuo, monitoramento em tempo real e integração nativa com bancos de dados como PostgreSQL.

- **ORM: TypeORM (ou Prisma)**
  Ferramentas para mapeamento objeto-relacional, facilitando o acesso e manipulação de dados.
  ○ **TypeORM**: Integra-se de forma mais nativa ao NestJS.
  ○ **Prisma**: Possui tipagem mais rigorosa e moderna. A escolha será feita com base na preferência e nas necessidades da equipe.

- **Banco de dados**
  ○ **SQLite (desenvolvimento)**: Banco de dados leve, embutido e sem necessidade de servidor, ideal para testes locais e desenvolvimento ágil.
  ○ **PostgreSQL (produção)**: Banco de dados relacional robusto, seguro e escalável, amplamente adotado em ambientes produtivos.

- **Integração Contínua (CI): GitHub Actions**
  Automatiza os processos de build, teste e deploy do backend. Auxilia na detecção precoce de falhas e assegura que o código entregue esteja funcional.

- **Variáveis de ambiente:** `@nestjs/config`
  Permite o gerenciamento seguro de configurações e segredos da aplicação, com suporte a arquivos `.env` e validação de variáveis em tempo de execução.