**Arquitetura do Sistema - Aplicativo Sinalize**

A arquitetura do **Sinalize** será baseada em uma arquitetura de **3 camadas (3-tier)**, que separa o sistema em diferentes responsabilidades e facilita a escalabilidade, manutenção e modularização do software. Essas camadas são:

* **Camada de Apresentação (Frontend)**
* **Camada de Lógica de Negócio (Backend)**
* **Camada de Dados (Banco de Dados)**

**1. Camada de Apresentação (Frontend)**

A camada de apresentação será responsável por toda a interface do usuário (UI) e interação direta com os usuários. A aplicação terá suporte para dispositivos móveis (iOS e Android).

**Tecnologias Utilizadas:**

* **React Native**: Framework para desenvolvimento de aplicativos móveis com uma base de código única para iOS e Android.
* **Redux**: Biblioteca para gerenciamento de estado global da aplicação.
* **Expo**: Ferramenta que facilita o desenvolvimento e a implantação de apps React Native.
* **React Navigation**: Para gerenciamento de rotas e navegação dentro do app.

**Responsabilidades:**

* Interface com o usuário, renderizando as telas das lições, quizzes, dicionário de sinais e notificações.
* Enviar e receber dados através de chamadas para a API do backend.
* Gerenciamento do estado do usuário (progresso das lições, dados do perfil).

**Componentes Importantes:**

* **Telas de Login e Cadastro**: Para autenticação de usuários.
* **Tela Principal (Dashboard)**: Onde os usuários acessam as lições e funcionalidades principais.
* **Módulo de Lições**: Apresenta vídeos e textos explicativos sobre sinais.
* **Módulo de Quizzes**: Testes interativos que avaliam o conhecimento adquirido.
* **Dicionário de Libras**: Interface que permite ao usuário pesquisar e visualizar sinais específicos.

**2. Camada de Lógica de Negócio (Backend)**

O backend será responsável pelo processamento das regras de negócio, tratamento de dados e integração com o banco de dados. Ele fornecerá uma API para o frontend através de endpoints RESTful, que serão consumidos pela aplicação móvel.

**Tecnologias Utilizadas:**

* **Node.js com Express**: Plataforma de desenvolvimento do lado do servidor para construção de APIs RESTful.
* **JWT (JSON Web Token)**: Para autenticação e controle de sessão.
* **Socket.io** (para futuras interações em tempo real): Caso o aplicativo adicione funcionalidades como chat ou colaboração em tempo real.
* **Cloud Storage (Firebase/Google Cloud/AWS S3)**: Para armazenar os vídeos e conteúdos multimídia das lições de Libras.

**Responsabilidades:**

* Implementação de APIs para lidar com registro, login e autenticação.
* Gerenciamento do progresso dos usuários (lições completadas, desempenho nos quizzes, etc.).
* Lidar com a lógica de gamificação (pontuação, recompensas, notificações).
* Processamento das consultas ao dicionário de sinais.
* Enviar notificações push para lembrar os usuários de lições ou metas.

**Estrutura de Endpoints RESTful:**

* **/api/users**: Endpoints para registro, login e gerenciamento de usuários.
* **/api/lessons**: Endpoints para listar, criar e atualizar lições.
* **/api/progress**: Endpoints para acompanhar o progresso dos usuários nas lições.
* **/api/quizzes**: Endpoints para gerenciamento e envio de quizzes.
* **/api/dictionary**: Endpoints para consulta de sinais no dicionário de Libras.

**3. Camada de Dados (Banco de Dados)**

A camada de dados será responsável por armazenar as informações do usuário, progresso, lições e quizzes. Será utilizado um banco de dados NoSQL devido à flexibilidade e escalabilidade necessária para gerenciar grandes volumes de dados dinâmicos.

**Tecnologias Utilizadas:**

* **MongoDB**: Banco de dados NoSQL para armazenamento de dados de usuários, progresso e conteúdo de lições.
* **Firebase Cloud Storage**: Para armazenamento de vídeos e outros conteúdos multimídia, permitindo fácil integração com o backend.
* **Redis** (opcional): Para caching de informações frequentes como resultados de quizzes e dados do dicionário de sinais, melhorando a performance.

**Estrutura do Banco de Dados:**

* **Usuários**:
  + *ID do usuário*
  + *Nome*
  + *Email*
  + *Senha (hash)*
  + *Progresso das lições*
  + *Pontuação e conquistas*
* **Lições**:
  + *ID da lição*
  + *Título*
  + *Conteúdo (texto, vídeos, imagens)*
  + *Categoria (básico, intermediário, avançado)*
* **Quizzes**:
  + *ID do quiz*
  + *Perguntas*
  + *Respostas corretas*
  + *Dificuldade*
* **Dicionário de Sinais**:
  + *ID do sinal*
  + *Descrição*
  + *Vídeo do sinal*
  + *Palavras-chave associadas*

**4. Integrações Externas**

O **Sinalize** poderá integrar algumas APIs e serviços externos para melhorar a experiência do usuário:

* **Firebase Cloud Messaging (FCM)**: Para envio de notificações push, garantindo que os usuários sejam lembrados de praticar regularmente.
* **Google Analytics ou Firebase Analytics**: Para monitorar o uso do aplicativo e obter insights sobre o comportamento dos usuários.
* **API de Tradução (opcional)**: Para tradução de conteúdos ou suporte a outros idiomas no futuro.

**5. Fluxo de Dados**

1. **Usuário** acessa o app pelo frontend (React Native).
2. O frontend envia uma **requisição para a API** no backend via chamadas HTTP REST.
3. O backend, ao receber a requisição, realiza o processamento necessário, consulta ou atualiza o **banco de dados** (MongoDB), e responde ao frontend.
4. O frontend atualiza a **interface do usuário** com os dados recebidos (progresso, novas lições, resultado dos quizzes, etc.).
5. Quando o usuário conclui uma lição, o progresso é enviado ao backend, que **atualiza os registros** no banco de dados.
6. **Notificações push** são enviadas periodicamente via Firebase, conforme os lembretes e conquistas do usuário.

**6. Segurança e Controle de Acesso**

* **Autenticação e Autorização**: Será utilizado JWT (JSON Web Tokens) para controlar o acesso dos usuários. Somente usuários autenticados poderão acessar recursos protegidos, como progresso de lições e conquistas.
* **Criptografia de Dados**: As senhas serão armazenadas de forma segura usando algoritmos de hash (como bcrypt).
* **Conexão Segura**: Todas as comunicações entre o aplicativo e o backend serão feitas através de HTTPS para garantir a integridade e confidencialidade dos dados