Módulo 1: Componentes para Conexão em Rede

Este módulo trata das ferramentas disponíveis no React Native para realizar requisições e consumir recursos externos (APIs ou WebServices).

|  |  |  |
| --- | --- | --- |
| Ponto Importante | Detalhe | Citação |
| **Finalidade da Conexão** | Aplicativos usam conexão remota para **Login**, **Consumo de dados** de APIs e **Persistência de dados**. O fluxo envolve realizar uma requisição, receber, tratar e exibir seu retorno. |  |
| **Fetch API (Nativo)** | É um componente **disponível nativamente** no React Native (e JavaScript em ambiente Web) para consumir e enviar dados usando métodos HTTP (GET, POST, etc.). |  |
| **Requisição GET com Fetch** | Usa o método .then() para receber a resposta e .json() para especificar o formato de dado a ser transferido (ex: JSON). |  |
| **Requisição POST com Fetch** | Exige a definição do método (method: 'POST'), o cabeçalho (headers) e o corpo (body) dos dados, que deve ser formatado como **string JSON** usando JSON.stringify. |  |
| **Axios (Biblioteca)** | É uma **biblioteca externa** amplamente utilizada para conexão remota. Deve ser instalada via npm install axios. O objeto de resposta (response) do Axios contém, entre outros, a data (o payload), status (código HTTP) e headers. |  |
| **Gerenciamento de Instâncias Axios** | Para múltiplos acessos ao mesmo recurso, é recomendado criar uma **única instância** de Axios (axios.create), definindo baseURL, timeout e headers, e importá-la nos locais necessários. |  |
| **Tratamento de Erros** | É fundamental usar o bloco .catch() para tratar possíveis falhas na requisição (ex: falha de conexão, erro HTTP 401) e fornecer feedback ao usuário. |  |

--------------------------------------------------------------------------------

Módulo 2: Persistência Remota com Controle de Acesso (REST e OAuth2)

Este módulo descreve a arquitetura de comunicação e os mecanismos de segurança para acesso a recursos remotos.

|  |  |  |
| --- | --- | --- |
| Ponto Importante | Detalhe | Citação |
| **Arquitetura REST** | É uma arquitetura de *software* (Transferência de Estado Representacional) utilizada para criar **serviços web (WebServices)** que manipulam recursos via URL. Faz uso do modelo Cliente x Servidor. |  |
| **Métodos HTTP em REST** | Utilizam-se verbos HTTP para operações, como **GET** (consulta de dados), **POST** (persistência/criação de novo dado), e **PUT** (atualização de dado existente). |  |
| **Formato de Dados** | Na arquitetura REST, utiliza-se, por padrão, o formato **JSON** (JavaScript Object Notation) para transmissão de dados entre cliente e servidor. |  |
| **Autenticação (Bearer Token)** | É um mecanismo essencial de controle de acesso para proteger APIs. O *token* é enviado no cabeçalho (headers) da requisição, geralmente sob a chave Authorization no formato Bearer <token>. |  |
| **OAuth2** | É um **Framework de Autorização** (RFC6749) que permite o acesso a recursos externos através de **tokens** de acesso, geralmente com tempo de vida útil determinado. |  |
| **Papéis do OAuth2** | Os quatro papéis são: Proprietário do Recurso, Cliente (o aplicativo), Servidor de Recurso e Servidor de Autorização. |  |
| **Ferramentas de Autenticação** | Existem diversas plataformas e bibliotecas que facilitam a implementação do controle de acesso, como Google Firebase Authentication, Auth0 e react-native-app-auth. |  |

--------------------------------------------------------------------------------

Módulo 3: Modelo Offline First

Este módulo explica a arquitetura que permite a funcionalidade do aplicativo mesmo sem conexão com a internet.

|  |  |  |
| --- | --- | --- |
| Ponto Importante | Detalhe | Citação |
| **Conceito** | O modelo *Offline First* permite que o aplicativo funcione de modo semelhante **com ou sem conexão à internet**. É uma recomendação que envolve preocupações e código extra. |  |
| **Fluxo Essencial** | O aplicativo deve **verificar o acesso à internet**, usar um **banco de dados embarcado** localmente e realizar a **sincronização** de dados (local -> remoto e remoto -> local) quando a conexão for restabelecida. |  |
| **Bancos de Dados Embarcados** | Opções para persistência local incluem AsyncStorage, SQLite, Realm e Watermelon DB. O Realm e o Watermelon DB já possuem mecanismos de sincronização. |  |
| **Componentes Offline/Online** | Bibliotecas como **Redux-offline** (que utiliza Redux) facilitam o controle das funcionalidades, definindo as ações a serem executadas quando online (effect e commit) e em caso de falha (rollback). |  |
| **Interface Otimista** | Técnica que melhora a experiência do usuário ao fazer parecer que o aplicativo é mais rápido do que é de fato. Mostra o sucesso da ação na interface imediatamente, antes da confirmação da persistência remota, que ocorre em segundo plano. |  |
| **Consistência de Dados** | Em aplicativos multiusuários, é crucial **identificar o usuário** que manipula a informação ao persistir dados localmente, garantindo a consistência durante a sincronização. |  |
| **Dependência Externa** | A implementação do *Offline First* exige uma estratégia bem definida e a instalação de **bibliotecas adicionais**, não sendo possível apenas com componentes nativos do React Native. |  |

Exercícios:

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Word

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.