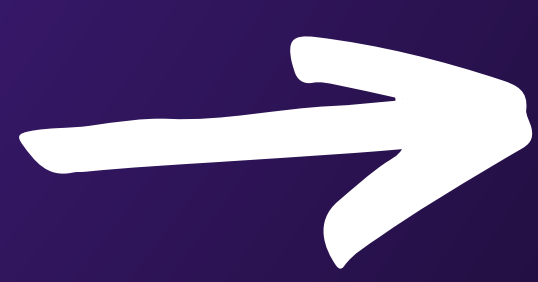
Uxman124

TYPES OF APIS



Uxman124

RESTFUL API

Representational State Transfer

(REST) is an architectural style for designing networked applications. RESTful APIs use standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources. For example, the Twitter API allows you to fetch tweets, post new tweets, and delete tweets using RESTful endpoints.

Uxman124

RESTFUL API

```
const axios = require('axios');

// GET request to fetch data
axios.get('https://api.example.com/users')
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error(error);
  });

// POST request to create data
axios.post('https://api.example.com/users', { name: 'John Doe', email:
'johndoe@example.com' })
  .then(response => {
    console.log(response.data);
})
.catch(error => {
  console.error(error);
});
```

Uxman124

SOAP API

Simple Object Access Protocol

(SOAP) is a protocol for exchanging structured information in web services using XML. SOAP APIs define a set of rules for communication between client and server. An example is the Amazon Web Services (AWS) API, which provides SOAP-based APIs for various services like Amazon S3 and Amazon EC2.

Uxman124

SOAP API

```javascript
const axios = require('axios');
const xml2js = require('xml2js');

const soapMessage = `
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
      <GetWeather xmlns="http://www.example.com/weather">
        <City>New York</City>
      </GetWeather>
    </soap:Body>
  </soap:Envelope>
`;

axios.post('https://api.example.com/soap', soapMessage, {
  headers: {
    'Content-Type': 'text/xml'
  }
})
  .then(response => {
    // Parse the XML response
    xml2js.parseString(response.data, (error, result) => {
      if (error) {
        console.error(error);
      } else {
        console.log(result);
      }
    });
  })
  .catch(error => {
    console.error(error);
  });
```

Uxman124

GRAPHQL API:

GraphQL is a query language for APIs that allows clients to request specific data requirements and get exactly what they need. It provides a single endpoint where clients can specify the fields and relationships they want in the response. GitHub's GraphQL API is an example, where you can query for specific information about repositories, users, and more.

GRAPHQL API:

```
import { ApolloClient, InMemoryCache, gql } from '@apollo/client';

const client = new ApolloClient({
  uri: 'https://api.example.com/graphql',
  cache: new InMemoryCache(),
});

// GraphQL query to fetch data
client.query({
  query: gql`
    query {
      users {
        id
        name
        email
      }
    }
  `
})
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error(error);
  });

// GraphQL mutation to create data
client.mutate({
  mutation: gql`
    mutation {
      createUser(name: "John Doe", email: "johndoe@example.com") {
        id
        name
        email
      }
    }
  `
})
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error(error);
  });
```

WEBSOCKET API:

WebSocket is a communication protocol that provides full-duplex communication channels over a single TCP connection. It allows real-time, bidirectional communication between clients and servers. A popular example is the WebSocket API used in chat applications, multiplayer games, and real-time data streaming.

Uxman124

WEBSOCKET API:

```
const io = require('socket.io-client');

const socket = io('https://api.example.com');

// Listen for 'message' event
socket.on('message', data => {
  console.log(data);
});

// Emit 'sendMessage' event
socket.emit('sendMessage', 'Hello, server!');
```

JSON-RPC AND XML-RPC:

JSON-RPC and XML-RPC are remote procedure call (RPC) protocols that enable communication between a client and a server using JSON or XML. These APIs allow clients to invoke methods on the server and receive the results. Bitcoin's JSON-RPC API is an example, where you can interact with a Bitcoin node programmatically.

JSON-RPC AND XML-RPC:

```
const axios = require('axios');

const rpcRequest = {
  jsonrpc: '2.0',
  method: 'getUser',
  params: {
    id: 123
  },
  id: 1
};

axios.post('https://api.example.com/json-rpc', rpcRequest, {
  headers: {
    'Content-Type': 'application/json'
  }
})
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error(error);
  });
```

Uxman124

OAUTH API:

OAuth (Open Authorization) is an authentication and authorization protocol used to grant third-party applications limited access to protected resources on behalf of a user. OAuth APIs allow users to securely authorize applications to access their data. The Google OAuth API is widely used for authentication and authorization with various Google services.

Uxman124

OAUTH API:

```
const passport = require('passport');
const GoogleStrategy = require('passport-google-oauth20').Strategy;

passport.use(new GoogleStrategy({
  clientID: 'your-client-id',
  clientSecret: 'your-client-secret',
  callbackURL: 'https://your-app.com/auth/google/callback'
}, (accessToken, refreshToken, profile, done) => {
  // Handle authentication and authorization logic here
  // ...
}));

// Authenticate using Google OAuth
app.get('/auth/google', passport.authenticate('google', { scope: ['profile', 'email'] }));

// Callback URL after successful authentication
app.get('/auth/google/callback', passport.authenticate('google', { successRedirect:
'/dashboard', failureRedirect: '/login' }));
```

Uxman124

THANK YOU

FOLLOW FOR MORE