Integrating a Frontend Application with a Backend Server

Introduction

Integrating a frontend application with a backend server is a critical aspect of modern web development. This process allows the frontend to interact with the backend to fetch, send, and manage data, enabling dynamic and interactive user experiences. This report covers the integration process, including an overview of RESTful APIs, making API calls from the frontend, and examples.

RESTful APIs

What is a RESTful API?

REST (Representational State Transfer) is an architectural style for designing networked applications. A RESTful API is an interface that adheres to REST principles, allowing communication between a client (frontend) and a server (backend) over HTTP.

Key Principles of RESTful APIs

1. **Statelessness:** Each request from the client to the server must contain all the information needed to understand and process the request. The server does not store client context between requests.
2. **Client-Server Architecture:** The client and server are separate entities, each with distinct responsibilities. The client handles the user interface, while the server manages data storage and business logic.
3. **Uniform Interface**: The API should have a consistent interface, simplifying and decoupling the architecture. This includes using standard HTTP methods (GET, POST, PUT, DELETE) and URIs (Uniform Resource Identifiers).
4. **Resource-Based:** Resources (data entities) are identified by URIs, and clients interact with these resources using HTTP methods.

HTTP Methods in RESTful APIs

- GET: Retrieve data from the server (e.g., fetching a list of users).
- POST: Send data to the server to create a new resource (e.g., creating a new user).
- PUT: Update an existing resource on the server (e.g., updating user information).
- DELETE: Remove a resource from the server (e.g., deleting a user).

Making API Calls from the Frontend

Tools and Libraries

To make API calls from the frontend, developers commonly use JavaScript along with libraries like fetch, axios, or frameworks like React, Angular, and Vue.

Using Fetch API

The fetch API is a built-in JavaScript function that allows making network requests similar to XMLHttpRequest but with a more powerful and flexible feature set.

Example: Fetching Data with GET Request

```
fetch('https://api.example.com/users')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Using Axios

axios is a popular HTTP client library that simplifies making requests and handling responses.

Example: Fetching Data with GET Request

```
import axios from 'axios';

axios.get('https://api.example.com/users')
  .then(response => console.log(response.data))
  .catch(error => console.error('Error:', error));
```

Handling POST Requests

Example: Sending Data with POST Request Using Fetch

```
const user = {
  name: 'John Doe',
  email: 'john.doe@example.com'
};
fetch('https://api.example.com/users', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(user)
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Example: Sending Data with POST Request Using Axios

```
import axios from 'axios';

const user = {
  name: 'John Doe',
  email: 'john.doe@example.com'
};
axios.post('https://api.example.com/users', user)
  .then(response => console.log(response.data))
  .catch(error => console.error('Error:', error));
```

Example: Integrating React Frontend with Express Backend

Backend Setup (Express)

1. Initialize the Project:

```
mkdir myapp
cd myapp
npm init -y
```

2. Install Dependencies:

```
npm install express body-parser cors
```

3. Create Server File (server.js):

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
const PORT = 5000;

app.use(cors());
app.use(bodyParser.json());

let users = [];
app.get('/users', (req, res) => {
  res.json(users);
});
app.post('/users', (req, res) => {
  const user = req.body;
  users.push(user);
  res.status(201).json(user);
});
app.listen(PORT, () => {
  console.log(Server running on
http://localhost:${PORT});
});
```

Frontend Setup (React)

1. Create React App:

```
npx create-react-app frontend
cd frontend
```

2. Install Axios:

```
npm install axios
```

3. Create UserList Component:

```
import React, { useEffect, useState } from 'react';
import axios from 'axios';

function UserList() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    axios.get('http://localhost:5000/users')
      .then(response => setUsers(response.data))
      .catch(error => console.error('Error:', error));
  }, []);

  return (
    <div>
      <h1>User List</h1>
      <ul>
```

```
        {users.map((user, index) => (
          <li key={index}>{user.name} - {user.email}</li>
        ))}
      </ul>
    </div>
  );
}
export default UserList;
```

4. Create AddUser Component:

```
import React, { useState } from 'react';
import axios from 'axios';
function AddUser() {
  const [name, setName] = useState(");
  const [email, setEmail] = useState(");

  const handleSubmit = (e) => {
    e.preventDefault();
    axios.post('http://localhost:5000/users', { name, email })
      .then(response => {
        console.log(response.data);
        setName(");
        setEmail(");
      })
      .catch(error => console.error('Error:', error));
  };
```

5. Integrate Components in App Component:

```
import React from 'react';
import UserList from './UserList';
import AddUser from './AddUser';

function App() {
  return (
    <div>
      <h1>My App</h1>
      <AddUser />
      <UserList />
    </div>
  );
}

export default App;
  return (
    <form onSubmit={handleSubmit}>
```

```
      <input
        type="text"
        placeholder="Name"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <input
        type="email"
        placeholder="Email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
      />
      <button type="submit">Add User</button>
    </form>
  );
}
export default AddUser;
```