GUÍA:
Integración RESTful con React y Azure

Sigla	Asignatura	Experiencia de Aprendizaje
DSY1104	Desarrollo Full Stack II	EA Integración RESTful con Google Cloud Platform
Tiempo	**Modalidad de Trabajo**	**Indicadores de logro**
2 h	Individual	IL 4.3

Antecedentes generales

Esta guía tiene como objetivo enumerar las acciones necesarias para dar solución a los problemas planteados.

Requerimientos para esta actividad

Para el desarrollo de esta actividad deberás disponer de:

- Computador
- Azure

Actividad

Esta actividad consiste en enumerar las acciones necesarias para dar solución a los casos que se verán a continuación, para ello los estudiantes deberán realizar la actividad de forma individual.

Sigue las Instrucciones

1. Configuración del proyecto Azure

1. Crea una cuenta en Azure si aún no tienes una.
2. Accede al Portal de Azure.
3. Crea un nuevo grupo de recursos para tu proyecto.
4. Instala Azure CLI en tu máquina local y configúrala.

2. Desarrollo del Frontend (React)

1. Crea una nueva aplicación React:

```
```

```
npx create-react-app mi-app-react
cd mi-app-react
```

2. Instala dependencias adicionales:

```
npm install axios react-router-dom @azure/msal-browser @azure/msal-react
```

3. Crea componentes para tu aplicación (por ejemplo, `src/components/ProductList.js`, `src/components/ProductDetail.js`).

4. Implementa llamadas a la API utilizando Axios:

```javascript
import axios from 'axios';

const API_URL = 'https://tu-app-name.azurewebsites.net/api';

export const getProducts = async () => {
  const response = await axios.get(`${API_URL}/products`);
  return response.data;
};
```

5. Utiliza los datos en tus componentes React.

3. Desarrollo del Backend

1. Crea un nuevo proyecto (ejemplo con Node.js/Express):

```
mkdir mi-backend
cd mi-backend
npm init -y
npm install express cors dotenv
```

2. Crea el archivo principal `app.js`:

```javascript
const express = require('express');
const cors = require('cors');
require('dotenv').config();
```

```
const app = express();
app.use(cors());
app.use(express.json());

app.get('/api/products', (req, res) => {
  // Lógica para obtener productos
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

module.exports = app;
```

3. Implementa las rutas y la lógica de negocio necesaria.

4. Configuración de la base de datos (Azure SQL Database)

1. Crea una instancia de Azure SQL Database desde el Portal de Azure.
2. Configura las credenciales y la base de datos.
3. Conecta tu backend con Azure SQL Database:

```javascript
const sql = require('mssql');

const config = {
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  server: process.env.DB_SERVER,
  database: process.env.DB_NAME,
  options: {
    encrypt: true
  }
};

sql.connect(config).then(pool => {
  // Puedes usar el pool para hacer consultas
});
```

5. Configuración de Azure Blob Storage (opcional)

1. Crea una cuenta de almacenamiento en Azure.
2. Crea un contenedor en Azure Blob Storage.
3. Utiliza el SDK de Azure para Node.js para manejar archivos:

```javascript
const { BlobServiceClient } = require('@azure/storage-blob');

const blobServiceClient = BlobServiceClient.fromConnectionString(process.env.AZURE_STORAGE_CONNECTION_STRING);
```

6. Despliegue

Frontend:

1. Construye tu aplicación React:

```
npm run build
```

2. Despliega en Azure Static Web Apps:
   - Crea un nuevo recurso de Azure Static Web Apps en el Portal de Azure.
   - Conecta tu repositorio de GitHub (o similar) y configura el flujo de trabajo de CI/CD.

Backend:

1. Crea un nuevo App Service en Azure:

```
az webapp up --sku F1 --name tu-app-name --resource-group tu-grupo-de-recursos
```

2. Configura las variables de entorno en la sección "Configuración" del App Service.

3. Despliega tu código:

```
git init
git add .
git commit -m "Initial commit"
git remote add azure https://tu-app-name.scm.azurewebsites.net:443/tu-app-name.git
git push azure master
```

```

7. Configuración final

1. Actualiza la URL del backend en tu frontend para que apunte a la URL de tu App Service.
2. Configura CORS en tu backend y en la configuración del App Service.
3. Implementa autenticación y autorización (por ejemplo, con Azure Active Directory).
4. Configura un dominio personalizado para tu frontend (Static Web App) y backend (App Service) si es necesario.

8. Pruebas y monitoreo

1. Realiza pruebas exhaustivas de tu aplicación.
2. Configura Azure Monitor y Application Insights para monitorear tu App Service y SQL Database.
3. Utiliza Azure Application Insights para analíticas del frontend y backend.
```

Recursos de apoyo

Servicios de informática en la nube. (s/f). Microsoft.com. Recuperado el 11 de agosto de 2024, de https://azure.microsoft.com/es-mx