دليل ربط الواجهة الأمامية (React) بالواجهة الخلفية (Go API)

نظرة عامة

سنقوم بربط مشروع React الخاص بك (صيدلية النهدي) بالواجهة الخلفية المبنية بـ Go. هذا الربط سيتم عبر طلبات HTTP إلى API endpoints.

1. بنية الربط

Plain Text

```
[React App على جهازك] ⟷ [Go API Server] ⟷ [PostgreSQL Database]
     (Frontend)              (Backend)           (Database)
```

2. خطوات الربط

الخطوة 1: إعداد API Service في React

أولاً، سنقوم بإنشاء ملف خدمة API في مشروع React الخاص بك:

JavaScript

```
// src/services/api.js
const API_BASE_URL = 'http://localhost:8080/api'; // عنوان الواجهة الخلفية

class ApiService {
  constructor() {
    this.baseURL = API_BASE_URL;
    this.token = localStorage.getItem('authToken');
  }

  // إعداد headers للطلبات
  getHeaders() {
    const headers = {
      'Content-Type': 'application/json',
    };
```

```
    if (this.token) {
      headers['Authorization'] = `Bearer ${this.token}`;
    }

    return headers;
  }

  // طلب GET عام
  async get(endpoint) {
    try {
      const response = await fetch(`${this.baseURL}${endpoint}`, {
        method: 'GET',
        headers: this.getHeaders(),
      });

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      return await response.json();
    } catch (error) {
      console.error('GET request failed:', error);
      throw error;
    }
  }

  // طلب POST عام
  async post(endpoint, data) {
    try {
      const response = await fetch(`${this.baseURL}${endpoint}`, {
        method: 'POST',
        headers: this.getHeaders(),
        body: JSON.stringify(data),
      });

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      return await response.json();
    } catch (error) {
      console.error('POST request failed:', error);
      throw error;
    }
  }

  // طلب PUT عام
  async put(endpoint, data) {
```

```
    try {
      const response = await fetch(`${this.baseURL}${endpoint}`, {
        method: 'PUT',
        headers: this.getHeaders(),
        body: JSON.stringify(data),
      });

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      return await response.json();
    } catch (error) {
      console.error('PUT request failed:', error);
      throw error;
    }
  }

  // طلب DELETE عام
  async delete(endpoint) {
    try {
      const response = await fetch(`${this.baseURL}${endpoint}`, {
        method: 'DELETE',
        headers: this.getHeaders(),
      });

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      return await response.json();
    } catch (error) {
      console.error('DELETE request failed:', error);
      throw error;
    }
  }

  // تحديث الرمز المميز
  setToken(token) {
    this.token = token;
    localStorage.setItem('authToken', token);
  }

  // إزالة الرمز المميز
  removeToken() {
    this.token = null;
    localStorage.removeItem('authToken');
  }
```

```
}

export default new ApiService();
```

الخطوة 2: إنشاء خدمات محددة للمنتجات

JavaScript

```
// src/services/productService.js
import ApiService from './api';

export const productService = {
  // الحصول على جميع المنتجات
  async getAllProducts(page = 1, limit = 20, category = '', search = '') {
    const params = new URLSearchParams({
      page: page.toString(),
      limit: limit.toString(),
      ...(category && { category }),
      ...(search && { search }),
    });

    return await ApiService.get(`/products?${params}`);
  },

  // الحصول على منتج محدد
  async getProduct(id) {
    return await ApiService.get(`/products/${id}`);
  },

  // الحصول على المنتجات المميزة
  async getFeaturedProducts() {
    return await ApiService.get('/products/featured');
  },

  // البحث في المنتجات
  async searchProducts(query) {
    return await ApiService.get(`/products/search?
q=${encodeURIComponent(query)}`);
  },

  // الحصول على منتجات فئة معينة
  async getProductsByCategory(categoryId) {
    return await ApiService.get(`/products/category/${categoryId}`);
  }
};
```

الخطوة 3: إنشاء خدمات للفئات

JavaScript

```
// src/services/categoryService.js
import ApiService from './api';

export const categoryService = {
  // الحصول على جميع الفئات
  async getAllCategories() {
    return await ApiService.get('/categories');
  },

  // الحصول على فئة محددة
  async getCategory(id) {
    return await ApiService.get(`/categories/${id}`);
  }
};
```

الخطوة 4: إنشاء خدمات المصادقة

JavaScript

```
// src/services/authService.js
import ApiService from './api';

export const authService = {
  // تسجيل الدخول
  async login(email, password) {
    const response = await ApiService.post('/auth/login', {
      email,
      password
    });

    if (response.token) {
      ApiService.setToken(response.token);
    }

    return response;
  },

  // تسجيل مستخدم جديد
  async register(userData) {
    const response = await ApiService.post('/auth/register', userData);
```

```
    if (response.token) {
      ApiService.setToken(response.token);
    }

    return response;
  },

  // تسجيل الخروج
  async logout() {
    await ApiService.post('/auth/logout', {});
    ApiService.removeToken();
  },

  // الحصول على ملف المستخدم
  async getProfile() {
    return await ApiService.get('/users/profile');
  },

  // تحديث ملف المستخدم
  async updateProfile(userData) {
    return await ApiService.put('/users/profile', userData);
  }
};
```

الخطوة 5: إنشاء خدمات سلة التسوق

JavaScript

```
// src/services/cartService.js
import ApiService from './api';

export const cartService = {
  // الحصول على سلة التسوق
  async getCart() {
    return await ApiService.get('/cart');
  },

  // إضافة منتج للسلة
  async addToCart(productId, quantity = 1) {
    return await ApiService.post('/cart/items', {
      product_id: productId,
      quantity
    });
  },

  // تحديث كمية منتج في السلة
```

```
  async updateCartItem(itemId, quantity) {
    return await ApiService.put(`/cart/items/${itemId}`, {
      quantity
    });
  },

  // حذف منتج من السلة
  async removeFromCart(itemId) {
    return await ApiService.delete(`/cart/items/${itemId}`);
  },

  // إفراغ السلة
  async clearCart() {
    return await ApiService.delete('/cart');
  }
};
```

الخطوة 6: إنشاء خدمات الطلبات

JavaScript

```
// src/services/orderService.js
import ApiService from './api';

export const orderService = {
  // إنشاء طلب جديد
  async createOrder(orderData) {
    return await ApiService.post('/orders', orderData);
  },

  // الحصول على طلبات المستخدم
  async getUserOrders() {
    return await ApiService.get('/orders');
  },

  // الحصول على طلب محدد
  async getOrder(orderId) {
    return await ApiService.get(`/orders/${orderId}`);
  },

  // تتبع الطلب
  async trackOrder(orderId) {
    return await ApiService.get(`/orders/${orderId}/tracking`);
  },

  // إلغاء الطلب
```

```
  async cancelOrder(orderId) {
    return await ApiService.post(`/orders/${orderId}/cancel`);
  }
};
```

3. تحديث Context للاستخدام مع API

تحديث ShopContext لاستخدام API

JavaScript

```
// src/context/ShopContext.jsx
import React, { createContext, useContext, useReducer, useEffect } from
'react';
import { productService } from '../services/productService';
import { cartService } from '../services/cartService';
import { authService } from '../services/authService';

const ShopContext = createContext();

const initialState = {
  products: [],
  categories: [],
  cartItems: [],
  favoriteItems: [],
  user: null,
  isLoading: false,
  error: null
};

function shopReducer(state, action) {
  switch (action.type) {
    case 'SET_LOADING':
      return { ...state, isLoading: action.payload };

    case 'SET_ERROR':
      return { ...state, error: action.payload, isLoading: false };

    case 'SET_PRODUCTS':
      return { ...state, products: action.payload, isLoading: false };

    case 'SET_CATEGORIES':
      return { ...state, categories: action.payload };

    case 'SET_CART_ITEMS':
```

```
      return { ...state, cartItems: action.payload };

    case 'SET_USER':
      return { ...state, user: action.payload };

    case 'ADD_TO_CART':
      return { ...state, cartItems: [...state.cartItems, action.payload] };

    case 'UPDATE_CART_ITEM':
      return {
        ...state,
        cartItems: state.cartItems.map(item =>
          item.id === action.payload.id ? action.payload : item
        )
      };

    case 'REMOVE_FROM_CART':
      return {
        ...state,
        cartItems: state.cartItems.filter(item => item.id !== action.payload)
      };

    case 'CLEAR_CART':
      return { ...state, cartItems: [] };

    default:
      return state;
  }
}

export function ShopProvider({ children }) {
  const [state, dispatch] = useReducer(shopReducer, initialState);

  // تحميل البيانات الأولية
  useEffect(() => {
    loadInitialData();
  }, []);

  const loadInitialData = async () => {
    try {
      dispatch({ type: 'SET_LOADING', payload: true });

      // تحميل المنتجات والفئات
      const [productsResponse, categoriesResponse] = await Promise.all([
        productService.getAllProducts(),
        categoryService.getAllCategories()
      ]);
```

```
      dispatch({ type: 'SET_PRODUCTS', payload: productsResponse.data });
      dispatch({ type: 'SET_CATEGORIES', payload: categoriesResponse.data });

      // تحميل سلة التسوق إذا كان المستخدم مسجل الدخول
      const token = localStorage.getItem('authToken');
      if (token) {
        const cartResponse = await cartService.getCart();
        dispatch({ type: 'SET_CART_ITEMS', payload: cartResponse.data });

        const userResponse = await authService.getProfile();
        dispatch({ type: 'SET_USER', payload: userResponse.data });
      }

    } catch (error) {
      dispatch({ type: 'SET_ERROR', payload: error.message });
    }
  };

  const addToCart = async (product, quantity = 1) => {
    try {
      const response = await cartService.addToCart(product.id, quantity);
      dispatch({ type: 'ADD_TO_CART', payload: response.data });
    } catch (error) {
      dispatch({ type: 'SET_ERROR', payload: error.message });
    }
  };

  const updateCartItem = async (itemId, quantity) => {
    try {
      const response = await cartService.updateCartItem(itemId, quantity);
      dispatch({ type: 'UPDATE_CART_ITEM', payload: response.data });
    } catch (error) {
      dispatch({ type: 'SET_ERROR', payload: error.message });
    }
  };

  const removeFromCart = async (itemId) => {
    try {
      await cartService.removeFromCart(itemId);
      dispatch({ type: 'REMOVE_FROM_CART', payload: itemId });
    } catch (error) {
      dispatch({ type: 'SET_ERROR', payload: error.message });
    }
  };

  const clearCart = async () => {
    try {
      await cartService.clearCart();
```

```
      dispatch({ type: 'CLEAR_CART' });
    } catch (error) {
      dispatch({ type: 'SET_ERROR', payload: error.message });
    }
  };

  const login = async (email, password) => {
    try {
      const response = await authService.login(email, password);
      dispatch({ type: 'SET_USER', payload: response.user });

      // تحميل سلة التسوق بعد تسجيل الدخول
      const cartResponse = await cartService.getCart();
      dispatch({ type: 'SET_CART_ITEMS', payload: cartResponse.data });

      return response;
    } catch (error) {
      dispatch({ type: 'SET_ERROR', payload: error.message });
      throw error;
    }
  };

  const logout = async () => {
    try {
      await authService.logout();
      dispatch({ type: 'SET_USER', payload: null });
      dispatch({ type: 'CLEAR_CART' });
    } catch (error) {
      dispatch({ type: 'SET_ERROR', payload: error.message });
    }
  };

  const value = {
    ...state,
    addToCart,
    updateCartItem,
    removeFromCart,
    clearCart,
    login,
    logout,
    loadInitialData
  };

  return (
    <ShopContext.Provider value={value}>
      {children}
    </ShopContext.Provider>
  );
```

```
}

export function useShop() {
  const context = useContext(ShopContext);
  if (!context) {
    throw new Error('useShop must be used within a ShopProvider');
  }
  return context;
}
```

4. تحديث المكونات لاستخدام API

مثال: تحديث HomePage لاستخدام البيانات من API

JavaScript

```
// src/components/HomePage.jsx
import React, { useEffect, useState } from 'react';
import { useShop } from '../context/ShopContext';
import { productService } from '../services/productService';

const HomePage = () => {
  const { categories, addToCart, isLoading } = useShop();
  const [featuredProducts, setFeaturedProducts] = useState([]);

  useEffect(() => {
    loadFeaturedProducts();
  }, []);

  const loadFeaturedProducts = async () => {
    try {
      const response = await productService.getFeaturedProducts();
      setFeaturedProducts(response.data);
    } catch (error) {
      console.error('Error loading featured products:', error);
    }
  };

  if (isLoading) {
    return <div className="text-center py-8">جاري التحميل...</div>;
  }

  return (
    <div>
      {/* عرض الفئات */}
```

```
      <section className="categories-section">
        <h2>الفئات</h2>
        <div className="categories-grid">
          {categories.map(category => (
            <div key={category.id} className="category-card">
              <img src={category.image_url} alt={category.name} />
              <h3>{category.name}</h3>
            </div>
          ))}
        </div>
      </section>

      {/* عرض المنتجات المميزة */}
      <section className="featured-products">
        <h2>المنتجات المميزة</h2>
        <div className="products-grid">
          {featuredProducts.map(product => (
            <div key={product.id} className="product-card">
              <img src={product.image_url} alt={product.name} />
              <h3>{product.name}</h3>
              <p>{product.price} ريال</p>
              <button onClick={() => addToCart(product)}>
                إضافة للسلة
              </button>
            </div>
          ))}
        </div>
      </section>
    </div>
  );
};

export default HomePage;
```

5. إعداد متغيرات البيئة

إنشاء ملف .env في مشروع React

Plain Text

```
# .env
REACT_APP_API_BASE_URL=http://localhost:8080/api
REACT_APP_API_TIMEOUT=10000
```

تحديث ملف api.js لاستخدام متغيرات البيئة

JavaScript

```
// src/services/api.js
const API_BASE_URL = process.env.REACT_APP_API_BASE_URL ||
'http://localhost:8080/api';
const API_TIMEOUT = process.env.REACT_APP_API_TIMEOUT || 10000;

// باقي الكود...
```

6. التعامل مع الأخطاء

إنشاء مكون لعرض الأخطاء

JavaScript

```
// src/components/ErrorBoundary.jsx
import React from 'react';

class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true, error };
  }

  componentDidCatch(error, errorInfo) {
    console.error('Error caught by boundary:', error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      return (
        <div className="error-boundary">
          <h2>حدث خطأ غير متوقع</h2>
          <p>نعتذر عن هذا الخطأ. يرجى إعادة تحميل الصفحة.</p>
          <button onClick={() => window.location.reload()}>
            إعادة تحميل
          </button>
        </div>
```

```
      );
    }

    return this.props.children;
  }
}

export default ErrorBoundary;
```

7. اختبار الربط

خطوات الاختبار

1. **تشغيل الواجهة الخلفية (Go API)**:

2. **تشغيل الواجهة الأمامية (React)**:

3. **التحقق من الاتصال**:

- افتح Developer Tools في المتصفح
- تحقق من Network tab لرؤية طلبات API
- تأكد من عدم وجود أخطاء CORS

8. نشر التطبيق

للنشر على الإنترنت:

1. **نشر الواجهة الخلفية**: سنستخدم خدمة النشر المتاحة
2. **تحديث عنوان API**: تغيير `API_BASE_URL` إلى عنوان الخادم المنشور
3. **نشر الواجهة الأمامية**: يمكن نشرها على Netlify أو Vercel

الخلاصة

هذا الدليل يوضح كيفية ربط مشروع React الخاص بك بالواجهة الخلفية المبنية بـ Go. الربط يتم عبر:

1. **طلبات HTTP** من React إلى Go API
2. **خدمات API** منظمة في React
3. **إدارة الحالة** المحدثة لاستخدام البيانات من الخادم
4. **معالجة الأخطاء** والتحميل
5. **المصادقة والتفويض** عبر JWT tokens

بهذا الشكل، ستحصل على تطبيق متكامل يعمل بكفاءة عالية!