Frontend Developer Guide

Multi-Tenant SaaS Platform with React + Vite + Tailwind

📋 Table of Contents

🎯 Overview

This platform allows users to:

- **Preview** multiple React templates
- **Select** a template for their business
- **Deploy** to their own subdomain/domain
- **Manage content** through Wagtail CMS

Key Principles

✅ **One template = unlimited users**
✅ **No code duplication per user**
✅ **Content comes from Wagtail API**
✅ **Dynamic branding per tenant**

🏗️ Architecture

```
┌─────────────────────────────────────────────────────┐
│          GitHub Repository                          │
│  saas-frontend-templates/                           │
│    ├── templates-frontend/                          │
│    │    ├── school-template/                        │
│    │    ├── hospital-template/                      │
│    │    └── restaurant-template/                    │
│    └── .github/workflows/deploy.yml                 │
└─────────────────────────────────────────────────────┘
                    ↓
┌─────────────────────────────────────────────────────┐
│        GitHub Actions (Auto Build)                  │
│  - npm install                                      │
│  - npm run build                                    │
│  - Upload dist/ to VPS                              │
└─────────────────────────────────────────────────────┘
                    ↓
┌─────────────────────────────────────────────────────┐
│           VPS (Hostinger)                           │
│  /var/www/                                          │
│    ├── user123/dist/  ← Apache serves this          │
│    ├── user456/dist/                                │
│    └── user789/dist/                                │
└─────────────────────────────────────────────────────┘
                    ↓
┌─────────────────────────────────────────────────────┐
│        Wagtail Backend API                          │
│  Provides tenant-specific content:                  │
│  - Logo, colors, branding                           │
│  - Pages, images, text                              │
│  - Menu items, footer                               │
└─────────────────────────────────────────────────────┘
```

📁 Project Structure

Complete Template Structure

```
school-template/
├── public/
```

```
│   ├── favicon.ico
│   └── logo.svg
│
├── src/
│   ├── components/
│   │   ├── Header.tsx
│   │   ├── Footer.tsx
│   │   ├── Hero.tsx
│   │   └── ContactForm.tsx
│   │
│   ├── pages/
│   │   ├── Home.tsx
│   │   ├── About.tsx
│   │   └── Contact.tsx
│   │
│   ├── services/
│   │   ├── tenantLoader.ts     ← Key file
│   │   └── api.ts
│   │
│   ├── types/
│   │   └── tenant.ts
│   │
│   ├── App.tsx
│   ├── App.css
│   └── main.tsx
│
├── .env.example
├── package.json
├── tailwind.config.js
├── tsconfig.json
└── vite.config.ts
```

🛠️ Template Development

Step 1: Create New Template

bash

```
# Navigate to templates folder
cd templates-frontend/

# Create new Vite + React + TypeScript project
npm create vite@latest my-template -- --template react-ts

# Navigate into template
cd my-template

# Install dependencies
npm install

# Install Tailwind CSS
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p

# Install additional dependencies
npm install axios
```

Step 2: Configure Tailwind

tailwind.config.js

```javascript
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {
      colors: {
        primary: 'var(--primary-color)',
        secondary: 'var(--secondary-color)',
        accent: 'var(--accent-color)',
      },
    },
  },
  plugins: [],
}
```

Step 3: Setup Environment Variables

.env.example

```
VITE_API_URL=http://localhost:8000
VITE_TENANT_DOMAIN=localhost
```

.env

```
VITE_API_URL=https://api.yoursaas.com
VITE_TENANT_DOMAIN=demo.yoursaas.com
```

🔄 Tenant Loading System

Core Concept

Every React template detects which tenant (user) it belongs to and fetches their specific content from Wagtail.

tenantLoader.ts

```
typescript
```

```
// src/services/tenantLoader.ts
import axios from 'axios';

export interface TenantData {
  id: string;
  name: string;
  domain: string;
  logo: string;
  primary_color: string;
  secondary_color: string;
  accent_color: string;
  font_family: string;
  pages: Page[];
}

export interface Page {
  id: string;
  slug: string;
  title: string;
  content: any;
}

export async function loadTenantData(): Promise<TenantData> {
  try {
    // Get current domain
    const domain = window.location.hostname;

    // Fetch tenant configuration from Wagtail
    const response = await axios.get(
      `${import.meta.env.VITE_API_URL}/api/tenant/?domain=${domain}`
    );

    return response.data;
  } catch (error) {
    console.error('Failed to load tenant data:', error);

    // Return default configuration
    return {
      id: 'default',
      name: 'Default Company',
      domain: domain,
      logo: '/default-logo.svg',
      primary_color: '#3b82f6',
      secondary_color: '#1e40af',
      accent_color: '#f59e0b',
      font_family: 'Inter, sans-serif',
      pages: []
    };
  }
}
```

App.tsx Implementation

```
typescript
```

```
// src/App.tsx
import { useState, useEffect } from 'react';
import { loadTenantData, TenantData } from './services/tenantLoader';
import Header from './components/Header';
import Footer from './components/Footer';
import Home from './pages/Home';
import './App.css';

function App() {
  const [tenant, setTenant] = useState<TenantData | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    loadTenantData()
      .then(data => {
        setTenant(data);
        applyTenantStyles(data);
        setLoading(false);
      })
      .catch(err => {
        setError('Failed to load tenant configuration');
        setLoading(false);
      });
  }, []);

  const applyTenantStyles = (data: TenantData) => {
    const root = document.documentElement;
    root.style.setProperty('--primary-color', data.primary_color);
    root.style.setProperty('--secondary-color', data.secondary_color);
    root.style.setProperty('--accent-color', data.accent_color);
    root.style.setProperty('--font-family', data.font_family);

    // Update page title
    document.title = data.name;
  };

  if (loading) {
    return (
      <div className="min-h-screen flex items-center justify-center">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-primary"></div>
      </div>
    );
  }

  if (error || !tenant) {
    return (
      <div className="min-h-screen flex items-center justify-center">
        <div className="text-center">
          <h1 className="text-2xl font-bold text-red-600">Error</h1>
          <p className="text-gray-600">{error || 'Failed to load'}</p>
        </div>
      </div>
    );
  }

  return (
    <div className="min-h-screen flex flex-col">
      <Header tenant={tenant} />
      <main className="flex-grow">
        <Home tenant={tenant} />
      </main>
      <Footer tenant={tenant} />
    </div>
  );
}

export default App;
```

🎨 Dynamic Styling

CSS Variables Approach

App.css

```
css
```

```
:root {
  /* Default colors - will be overridden by tenant config */
  --primary-color: #3b82f6;
  --secondary-color: #1e40af;
  --accent-color: #f59e0b;
  --font-family: system-ui, -apple-system, sans-serif;
}

body {
  font-family: var(--font-family);
  margin: 0;
  padding: 0;
}

.btn-primary {
  background-color: var(--primary-color);
  color: white;
  padding: 0.75rem 1.5rem;
  border-radius: 0.5rem;
  transition: all 0.3s;
}

.btn-primary:hover {
  background-color: var(--secondary-color);
}

.text-primary {
  color: var(--primary-color);
}

.bg-primary {
  background-color: var(--primary-color);
}
```

Example Component with Dynamic Styling

```
typescript
```

```
// src/components/Hero.tsx
import { TenantData } from '../services/tenantLoader';

interface HeroProps {
  tenant: TenantData;
}

export default function Hero({ tenant }: HeroProps) {
  return (
    <section
      className="min-h-screen flex items-center justify-center"
      style={{
        background: `linear-gradient(135deg, ${tenant.primary_color}, ${tenant.secondary_color})`
      }}
    >
      <div className="text-center text-white px-4">
        <img
          src={tenant.logo}
          alt={tenant.name}
          className="h-24 mx-auto mb-8"
        />
        <h1 className="text-5xl font-bold mb-4">
          {tenant.name}
        </h1>
        <p className="text-xl mb-8 max-w-2xl mx-auto">
          Welcome to our platform
        </p>
        <button className="btn-primary">
          Get Started
        </button>
      </div>
    </section>
  );
}
```

🔌 API Integration

API Service Layer

```typescript
```

```
// src/services/api.ts
import axios from 'axios';

const api = axios.create({
  baseURL: import.meta.env.VITE_API_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

export const tenantAPI = {
  // Get tenant by domain
  getByDomain: async (domain: string) => {
    const response = await api.get(`/api/tenant/?domain=${domain}`);
    return response.data;
  },

  // Get tenant pages
  getPages: async (tenantId: string) => {
    const response = await api.get(`/api/tenant/${tenantId}/pages/`);
    return response.data;
  },

  // Get specific page
  getPage: async (tenantId: string, slug: string) => {
    const response = await api.get(`/api/tenant/${tenantId}/pages/${slug}/`);
    return response.data;
  },
};

export default api;
```

🚀 Deployment Process

GitHub Actions Workflow

The deployment is **fully automated** via GitHub Actions.

What happens when a user selects a template:

1. **Backend triggers GitHub Actions** via API
2. **GitHub Actions runs:**

  - Checks out code
  - Installs dependencies
  - Builds the template (`npm run build`)
  - Uploads `dist/` to VPS
  - Creates Apache config
  - Reloads Apache
3. **User's site goes live** at their domain

Manual Build (for testing)

```bash
# Navigate to template
cd templates-frontend/school-template

# Install dependencies
npm install

# Build for production
npm run build

# Output will be in dist/
```

🧪 Testing

Local Development

```bash
# Run development server
npm run dev

# Access at http://localhost:5173
```

Testing with Different Tenants

Create multiple `.env` files:

.env.tenant1

```
VITE_API_URL=https://api.yoursaas.com
VITE_TENANT_DOMAIN=tenant1.yoursaas.com
```

.env.tenant2

```
VITE_API_URL=https://api.yoursaas.com
VITE_TENANT_DOMAIN=tenant2.yoursaas.com
```

Run with specific env:

```bash
cp .env.tenant1 .env
npm run dev
```

✅ Best Practices

1. Always Use CSS Variables for Colors

```css
/* ✅ Good */
.button {
  background-color: var(--primary-color);
}

/* ❌ Bad */
.button {
  background-color: #3b82f6;
}
```

2. Make All Content Dynamic

```typescript
// ✅ Good - fetch from API
const hero = tenant.pages.find(p => p.slug === 'hero');

// ❌ Bad - hardcoded
const heroTitle = "Welcome to Our School";
```

3. Handle Loading States

```typescript
if (loading) return <LoadingSpinner />;
if (error) return <ErrorMessage />;
if (!tenant) return <NotFound />;
```

4. Optimize Images

```typescript
<img
  src={tenant.logo}
  alt={tenant.name}
  loading="lazy"
  className="h-16 w-auto"
/>
```

5. Keep Templates Simple

- Focus on layout and structure
- Content should come from Wagtail
- Avoid complex business logic in templates

6. Use TypeScript Strictly

```typescript
// Always define types
interface Props {
  tenant: TenantData;
  page?: Page;
}
```

📚 Common Issues & Solutions

Issue 1: Tenant not loading

Solution: Check if domain matches exactly in Wagtail database

Issue 2: Colors not applying

Solution: Ensure CSS variables are set in root element

Issue 3: Build fails

Solution: Check Node version (18+), clear node_modules and reinstall

Issue 4: API CORS errors

Solution: Backend must allow your domain in CORS settings

🎯 Next Steps

1. ✅ Create your first template
2. ✅ Test with local Wagtail backend
3. ✅ Push to GitHub
4. ✅ Configure GitHub secrets
5. ✅ Test deployment
6. ✅ Create more templates

📞 Support

For questions or issues:

- Check GitHub Issues
- Review backend documentation
- Contact backend team for API changes

Version: 1.0
Last Updated: December 2024
Author: SaaS Platform Team