STYLECLINIC

Manual Tecnico del Sistema Web

Versión 1.0

BY JUAN DIEGO BERNAL DELGADO, JUAN PABLO GUTIERREZ SIERRA

Tabla de Contenido

1. Introducción

1.1 Acerca de la marca

StyleClinic es una marca enfocada en ofrecer una experiencia moderna y personalizada en el sector de la moda urbana. Su propuesta se centra en combinar estilo, innovación y personalización, permitiendo que el usuario acceda a productos cuidadosamente seleccionados y, al mismo tiempo, diseñe prendas personalizadas mediante herramientas digitales.

El concepto de la marca se basa en la idea de que la moda es una forma de expresión individual. Por eso, StyleClinic busca ofrecer no solo productos, sino una experiencia flexible en la que cada persona pueda construir su propio estilo mediante una plataforma accesible, intuitiva y contemporánea.

La plataforma StyleClinic se diseñó para reflejar los valores de la marca: simplicidad, autenticidad y funcionalidad. Cada sección del sistema está pensada para guiar al usuario con claridad, facilitando la compra, la personalización y la interacción con la tienda.

1.2 Propósito del manual

Este manual tiene como fin documentar a nivel técnico todo el funcionamiento del sistema StyleClinic, abarcando tanto su arquitectura frontend y backend como los procesos de instalación, mantenimiento y despliegue.

Está dirigido a:

- Desarrolladores
- Integradores
- Administradores de sistemas
- Equipos de soporte técnico

1.3 Alcance

El manual cubre:

- Arquitectura general
- Implementación técnica del frontend en React
- Implementación técnica del backend en Spring Boot
- Comunicación entre capas
- API REST
- Instalación y despliegue
- Mantenimiento

Este documento no cubre la operación del sistema desde el punto de vista del usuario final, ya que ese contenido corresponde al Manual de Usuario.

---

2. Arquitectura General del Sistema

2.1 Visión general

StyleClinic está compuesto por dos módulos:

1. Frontend: Aplicación SPA construida con React.
2. Backend: API REST construida con Spring Boot.

Ambos se comunican mediante peticiones HTTP sobre JSON.

2.2 Tecnologías principales

Frontend

- React
- React Router
- Context API
- Axios

Backend

- Java
- Spring Boot
- Spring Web
- Spring Data JPA
- Spring Security (si se habilita)
- Base de datos (según configuración)

2.3 Comunicación Frontend–Backend

La comunicación ocurre mediante Axios en el frontend, consumiendo endpoints del backend.
Formato de intercambio: JSON.

---

3. Estructura del Proyecto

3.1 Frontend

Estructura típica encontrada:

```
src/
  components/
  pages/
  context/
  hooks/
  assets/
  router/
  App.js
  index.js
```

3.2 Backend

Estructura basada en Spring Boot:

```
src/main/java/com/styleclinic/
  controller/
  service/
  repository/
  model/
  config/
src/main/resources/
  application.properties
```

4. Frontend (React)

4.1 Instalación

```
npm install
npm start
```

4.2 Dependencias principales

- react-router-dom
- axios

4.3 Descripción de carpetas

components/
Componentes reutilizables: ProductCard, Navbar, Footer, etc.

pages/
Páginas renderizadas según las rutas: Home, Catalogo, Custom, Cart, Login, Register, Contact.

context/

- AuthContext: maneja autenticación
- CartContext: administra el carrito global

router/

- Enrutador.js: define rutas principales

4.4 Enrutamiento

El sistema usa React Router y define las rutas:

- /
- /catalogo
- /custom
- /cart
- /login
- /register

4.5 Contextos globales

AuthContext
Control del usuario, login, logout, persistencia básica.

CartContext
Gestión global del carrito: añadir, eliminar, actualizar cantidades.

4.6 Componentes principales

- ProductCard
- ProductDetail
- ProductCarousel
- Navbar
- Footer
- CustomForm

4.7 Ciclo de construcción y despliegue

```
npm run build
```

Genera carpeta `/build` lista para distribución.

---

5. Backend (Spring Boot)

5.1 Instalación

Importar en IDE, luego ejecutar:

```
mvn spring-boot:run
```

5.2 Dependencias principales

- spring-boot-starter-web
- spring-boot-starter-data-jpa
- spring-boot-starter-security (si se usa)
- driver de base de datos

5.3 Estructura interna

controller/
Define endpoints REST.

service/
Lógica de negocio.

repository/
Persistencia mediante JPA.

model/
Entidades mapeadas.

5.4 Controladores

Se encargan de entregar productos, gestionar usuarios, personalización y carrito.

5.5 Modelos y entidades

Representan productos, usuarios, pedidos, personalizaciones, etc.

5.6 Servicios

Implementación de reglas de negocio.

5.7 Seguridad

Opcional según desarrollo. Maneja autenticación y autorización.

5.8 Endpoints expuestos

Dependen del diseño, incluir:

- /productos
- /usuarios
- /auth
- /carrito

---

6. Base de Datos

6.1 Diseño propuesto

- Tabla usuarios
- Tabla productos
- Tabla carrito
- Tabla pedidos
- Tabla personalizaciones

6.2 Entidades

Corresponden a las tablas previamente mencionadas.

6.3 Relaciones

Típicas relaciones 1:N y N:N según caso.

---

7. API REST

7.1 Consideraciones

- Stateless
- JSON como formato
- Código de estados HTTP estándar

7.2 Endpoints detallados

Puede documentarse con Swagger o Postman.

---

8. Seguridad

8.1 Autenticación

Puede ser manejada mediante JWT o sesiones simples.

8.2 Autorización

Control por roles (si se implementa).

8.3 Protección de datos

Cifrado de contraseñas, validaciones, sanitización.

---

9. Instalación Completa del Sistema

9.1 Requisitos

- Node.js
- Java 17+
- Maven
- Base de datos configurada

9.2 Instalación frontend

```
npm install
npm start
```

9.3 Instalación backend

```
mvn clean install
mvn spring-boot:run
```

9.4 Configuración del entorno

Configurar conexión BD en application.properties.

---

10. Despliegue

10.1 Frontend

Subir carpeta build a servidor web o CDN.

10.2 Backend

Desplegar como JAR o mediante Tomcat/Cloud.

10.3 Arquitecturas recomendadas

- Docker
- Nginx como proxy reverso
- Integración con CI/CD

---

11. Mantenimiento

11.1 Actualizaciones

Actualizar dependencias periódicamente.

11.2 Limpieza

Eliminar logs antiguos y builds no usados.

11.3 Monitoreo

Uso de herramientas como Grafana, Spring Actuator.

12. Anexos

12.1 Comandos útiles

```
npm run build
mvn clean install
```

12.2 Buenas prácticas

- Código modular
- Documentación continua
- Versionamiento con Git

12.3 Convenciones

React: componentes en PascalCase.
Java: camelCase para métodos.