Manual de Implantación del Sistema OkBranding (Frontend y Backend)

1. Introducción

El presente documento describe el proceso de implantación del sistema web de OkBranding, compuesto por:

- Una aplicación **frontend** desarrollada en **Angular**, con módulos diferenciados para clientes y administradores.
- Un **backend** desarrollado en **Spring Boot (Java 21)**, que expone una API REST y se conecta a una base de datos **MySQL** (actualmente en Azure Database for MySQL Flexible Server).

El documento incluye requerimientos de software y hardware, pasos de instalación y configuración, consideraciones de migración, respaldo de datos y políticas de backup, plan de capacitación y plan de liberación del sistema de información, con el fin de garantizar una puesta en marcha controlada y sostenible en el tiempo, abarcando tanto frontend como backend.

---

2. Objetivos

2.1 Objetivo general

Implantar el sistema web de OkBranding (frontend + backend) en un entorno de producción estable, asegurando su operatividad, mantenibilidad, seguridad y escalabilidad, de forma que los usuarios finales (clientes y administradores) puedan utilizar todas las funcionalidades previstas.

2.2 Objetivos específicos

- Disponer de una aplicación web responsiva para la gestión de productos, categorías, colores, carrusel del home y proceso de cotizaciones.
- Facilitar la administración del contenido (productos, categorías, colores, carrusel, usuarios) a través del módulo de administración protegido.
- Exponer una API REST segura en el backend para la gestión de entidades de negocio (productos, categorías, usuarios, cotizaciones, etc.), con autenticación basada en **JWT**.
- Definir un proceso estándar de instalación, actualización y liberación de nuevas versiones del sistema (frontend y backend).
- Establecer lineamientos de respaldo y recuperación de la información almacenada en la base de datos de producción.
- Proveer lineamientos de soporte y mantenimiento correctivo y evolutivo tanto para el frontend como para el backend.

---

3. Requerimientos de software

3.1 Entorno de desarrollo

3.1.1 Frontend

- Sistema operativo: Windows 10/11, macOS o distribución Linux.
- Node.js: versión LTS (>= 18.x).
- Angular CLI: versión compatible con el proyecto (según `package.json`).
- NPM: versión incluida con Node LTS.
- Editor recomendado: VS Code o equivalente.
- Navegadores para pruebas: Chrome, Edge, Firefox.

3.1.2 Backend

- Sistema operativo: Windows 10/11, macOS o distribución Linux.
- **Java Development Kit (JDK) 21** (por ejemplo, Eclipse Temurin 21).
- **Maven 3.9+** (o uso de `mvnw` incluido en el proyecto).
- Servidor de base de datos **MySQL 8.x** (local o remoto) para desarrollo.
- Herramienta para gestionar la base de datos (MySQL Workbench, DBeaver, etc.).
- Opcional: Docker (para construir y ejecutar la imagen del backend).

3.2 Entorno de producción

Dependiendo de la estrategia de despliegue:

3.2.1 Frontend

- **Servidor web estático**:
  - Actualmente desplegado en **Azure Static Web Apps**.
  - URL pública de producción: `https://blue-sky-09b61240f.3.azurestaticapps.net/`.
- Configuración de red y DNS:
  - Dominio o subdominio público para el frontend (puede ser el proporcionado por Azure Static Web Apps o un dominio propio).
  - Certificados SSL/TLS válidos.

3.2.2 Backend

- **Servidor de aplicaciones Java**:
  - Actualmente desplegado en **Azure App Service**.
  - URL pública de producción: `https://okbranding-ava4htfqc2ajefhh.chilecentral-01.azurewebsites.net`.
- **Base de datos MySQL**:
  - Servicio gestionado **Azure Database for MySQL Flexible Server**.
  - Acceso desde el servidor backend a la base de datos (puerto, firewall, red virtual, etc.).
- Variables de entorno recomendadas:
  - `JDBC_DATABASE_URL` (URL de conexión a MySQL).
  - `JDBC_DATABASE_USERNAME` (usuario de BD).
  - `JDBC_DATABASE_PASSWORD` (contraseña de BD).
  - `PORT` (puerto HTTP del backend, en cloud suele venir dado).
  - En despliegues más avanzados, clave JWT y otros secretos deberían parametrizarse vía variables de entorno (ver sección 15).

3.3 Dependencias del proyecto

Frontend

Según `package.json`, el proyecto utiliza al menos:

- Angular Framework.
- Angular Material (componentes de interfaz).
- RxJS.
- Dependencias propias de Angular CLI.

Backend

Según `pom.xml`, el proyecto utiliza:

- `spring-boot-starter-web` (API REST).
- `spring-boot-starter-data-jpa` (acceso a datos con JPA/Hibernate).
- `spring-boot-starter-security` + `spring-security-crypto` (seguridad / autenticación).
- `spring-boot-starter-validation` (validación).
- `spring-boot-starter-actuator` (endpoints de monitorización).
- `mysql-connector-j` (conector MySQL).
- `jjwt-*` (manejo de tokens JWT).
- `spring-boot-starter-mail` (envío de correo, si se habilita).
- `lombok`, `mapstruct` y `mapstruct-processor` (DTOs y mapeos).
- Plugins de Maven para compilación y empaquetado con Spring Boot.

3.4 Repositorios y ramas

- Frontend:
  - Repositorio Git: `https://github.com/JANA-SENA-EDU/OkbrandingFrontend.git`
  - Rama principal: `main`.
- Backend:
  - Repositorio Git: `https://github.com/JANA-SENA-EDU/OkbrandingBackend.git`
  - Rama principal: `main`.

---

4. Requerimientos de hardware

4.1 Servidor de producción (frontend)

- CPU: 2 vCPU o superior.
- RAM: 4 GB o superior.
- Almacenamiento: 20 GB (mínimo) para sistema, logs y artefactos estáticos.
- Conectividad: acceso a internet o red corporativa según el alcance.

4.2 Servidor de producción (backend y base de datos)

Los valores pueden ajustarse según carga esperada:

- CPU: 2–4 vCPU para el backend; 2–4 vCPU para la base de datos (en caso de servidor separado).
- RAM:
  - Backend: 4–8 GB.

    - Base de datos: 8 GB o superior según volumen de datos.
- Almacenamiento:
    - Backend: 20–40 GB para sistema, logs y JARs/imágenes Docker.
    - BD: almacenamiento dimensionado según el crecimiento esperado (mínimo 50–100 GB), con discos SSD recomendados.
- Conectividad:
    - Acceso a la base de datos desde el backend (red interna segura o reglas de firewall adecuadas).
    - Acceso controlado desde internet o red corporativa hacia el backend (firewall / WAF).

4.3 Equipos de desarrollo

- CPU equivalente a Intel i5 o superior.
- RAM: 8 GB o superior.
- Disco SSD recomendado para mejorar tiempos de compilación y ejecución.

---

5. Dispositivos adicionales

No se requiere hardware adicional específico.

El sistema se utiliza desde navegadores web estándar (Chrome, Edge, Firefox, Safari) en equipos de escritorio o dispositivos móviles.
El backend expone una API HTTP/HTTPS consumida únicamente por el frontend y por herramientas de administración o integración si se definen.

---

6. Migración

6.1 Migración de frontend

En el contexto del frontend, la migración se centra en:

- Adaptar la nueva versión del frontend a la API existente del backend.
- Ajustar parámetros de endpoints, URLs y configuraciones de entorno (`apiUrl`, rutas, etc.).

Si existe una versión anterior del frontend:

- Validar compatibilidad con la API actual.
- Actualizar rutas, componentes y servicios según cambios funcionales.
- Coordinar una ventana de migración con el backend para minimizar impactos.

6.2 Migración de backend y datos

La migración de datos (productos, categorías, usuarios, cotizaciones, etc.) se realiza en la base de datos del backend.

- Revisar la configuración de `spring.jpa.hibernate.ddl-auto` (actualmente `update`):
    - Para entornos productivos puede ser recomendable usar migraciones controladas (Flyway/Liquibase) en lugar de cambios automáticos.
- Antes de cambios de esquema:
    - Realizar backup completo de la base de datos.

    - Probar los cambios en un entorno de staging con una copia de datos.
- Migración de lógica de negocio:
    - Revisar cambios en controladores (`CategoriaController`, `ProductoController`, `UsuarioController`, etc.) y DTOs.
    - Garantizar compatibilidad hacia atrás de la API o planificar cambios de versión (por ejemplo `/api/v1/...`).

---

7. Instalación del aplicativo

7.1 Instalación del frontend en entorno de desarrollo

1. Clonar el repositorio:

```
git clone https://github.com/JANA-SENA-EDU/OkbrandingFrontend.git
cd OkbrandingFrontend
```

2. Instalar dependencias:

```
npm install
```

3. Ejecutar el servidor de desarrollo:

```
ng serve
```

4. Acceder a la aplicación:
    - Navegador: `http://localhost:4200`

Asegurarse de que `environment.ts` apunte al backend de desarrollo (por ejemplo `https://okbranding-ava4htfqc2ajefhh.chilecentral-01.azurewebsites.net/okBranding/...`).

7.2 Instalación del backend en entorno de desarrollo

1. Clonar el repositorio:

```
git clone https://github.com/JANA-SENA-EDU/OkbrandingBackend.git
cd OkbrandingBackend
```

2. Configurar acceso a la base de datos MySQL de desarrollo:
    - Crear base de datos `okbrandingdb` (nombre de ejemplo o el especificado en `spring.datasource.url`).
    - Configurar usuario y contraseña con permisos sobre la base.
3. Configurar `application.properties` o variables de entorno:

- Opción A – usar variables de entorno:
  - `JDBC_DATABASE_URL=jdbc:mysql://<host>:3306/okbrandingdb?sslMode=REQUIRED|DISABLED`
  - `JDBC_DATABASE_USERNAME=<usuario>`
  - `JDBC_DATABASE_PASSWORD=<contraseña>`
- Opción B – ajustar directamente `application.properties` (no recomendado para producción, solo desarrollo).

4. Ejecutar el backend:

```
./mvnw spring-boot:run
```

o bien:

```
./mvnw clean package
java -jar target/back-0.0.1-SNAPSHOT.jar
```

5. Verificar:
   - Acceso a `https://okbranding-ava4htfqc2ajefhh.chilecentral-01.azurewebsites.net/actuator/health` (según configuración).
   - Endpoints de negocio, por ejemplo:
     - `GET https://okbranding-ava4htfqc2ajefhh.chilecentral-01.azurewebsites.net/okBranding/productos/listar`
     - `POST https://okbranding-ava4htfqc2ajefhh.chilecentral-01.azurewebsites.net/okBranding/auth/login`

7.3 Construcción y despliegue del frontend en producción

1. Generar build de producción:

```
ng build --configuration production
```

Los artefactos se generan en `dist/<nombre-proyecto>/` (según `angular.json`).

2. Copiar los archivos generados al servidor web (Azure Static Web Apps o Nginx/Apache si se usa otra infraestructura).
3. Configurar el servidor para:
   - Servir `index.html` como documento inicial.
   - Redirigir todas las rutas de la SPA a `index.html` (manejo de rutas en Angular).
   - Forzar HTTPS (redirecciones y certificados configurados).

En el entorno actual, la URL de producción del frontend es:

- `https://blue-sky-09b61240f.3.azurestaticapps.net/`

7.4 Construcción y despliegue del backend en producción

7.4.1 Despliegue sin contenedores

1. Construir el JAR:

```
./mvnw clean package -DskipTests
```

2. Copiar `target/back-0.0.1-SNAPSHOT.jar` al servidor de producción (si no se usa Docker).
3. Configurar variables de entorno para producción:
   - `JDBC_DATABASE_URL`, `JDBC_DATABASE_USERNAME`, `JDBC_DATABASE_PASSWORD`.
   - `PORT` (si se requiere un puerto distinto a 8080, aunque en Azure App Service suele gestionarse automáticamente).
4. Ejecutar el backend:

```
java -jar back-0.0.1-SNAPSHOT.jar
```

5. Configurar servicio (systemd o equivalente) para que se ejecute como servicio de sistema, con restart automático (en caso de servidor propio).

7.4.2 Despliegue con Docker

1. Construir la imagen Docker desde `OkbrandingBackend` (si se opta por desplegar en contenedores):

```
docker build -t okbranding-backend .
```

2. Ejecutar el contenedor:

```
docker run -d \
  -e JDBC_DATABASE_URL="jdbc:mysql://<host>:3306/okbrandingdb?
sslMode=REQUIRED" \
  -e JDBC_DATABASE_USERNAME="<usuario>" \
  -e JDBC_DATABASE_PASSWORD="<contraseña>" \
  -e PORT=8080 \
  -p 8080:8080 \
  --name okbranding-backend \
  okbranding-backend
```

3. Integrar con el balanceador o gateway de la infraestructura (si aplica).

En el entorno actual, el backend está desplegado en Azure App Service con la URL:

- `https://okbranding-ava4htfqc2ajefhh.chilecentral-01.azurewebsites.net`

---

8. Configuración y puesta en marcha

8.1 Configuración de entornos (frontend)

Archivos:

- `src/environments/environment.ts` (desarrollo).
- `src/environments/environment.prod.ts` (producción).

Parámetros clave:

- `apiUrl`: URL base del backend (REST API), por ejemplo:
  - Desarrollo: `https://okbranding-ava4htfqc2ajefhh.chilecentral-01.azurewebsites.net/okBranding`
  - Producción: `https://okbranding-ava4htfqc2ajefhh.chilecentral-01.azurewebsites.net/okBranding`
- Otros parámetros según necesidades (por ejemplo, base URL para recursos estáticos).

8.2 Configuración del backend

Archivo principal: `src/main/resources/application.properties`.

Parámetros clave:

- **Datasource**:
  - `spring.datasource.url=${JDBC_DATABASE_URL:jdbc:mysql://.../okbrandingdb?sslMode=REQUIRED}`
  - `spring.datasource.username=${JDBC_DATABASE_USERNAME:...}`
  - `spring.datasource.password=${JDBC_DATABASE_PASSWORD:...}`
- **JPA/Hibernate**:
  - `spring.jpa.hibernate.ddl-auto=${JPA_DDL_AUTO:update}`
  - `spring.jpa.show-sql=${JPA_SHOW_SQL:true}`
  - `spring.jpa.properties.hibernate.format_sql=${JPA_FORMAT_SQL:true}`
- **Servidor**:
  - `server.port=${PORT:8080}`
  - `server.address=0.0.0.0`
- **Logging**:
  - `logging.file.name=<ruta-logs>` (en cloud actual: `/home/LogFiles/application.log`).

Recomendaciones:

- En producción, parametrizar también la clave JWT y otros secretos en variables de entorno y no en código fuente.
- Configurar CORS adecuadamente en los controladores (`@CrossOrigin`) o a nivel global, permitiendo el dominio del frontend (por ejemplo, `https://blue-sky-09b61240f.3.azurestaticapps.net/`).

8.3 Puesta en marcha integrada

1. Tener backend en ejecución (local o producción).
2. Tener frontend desplegado y con `apiUrl` apuntando al backend correcto.
3. Verificar funcionamiento end-to-end:

    - Carga del index del frontend (en producción: `https://blue-sky-09b61240f.3.azurestaticapps.net/`).
    - Navegación por categorías y productos.
    - Apertura del detalle de producto.
    - Registro e inicio de sesión de usuarios vía endpoints `/okBranding/auth/register` y `/okBranding/auth/login`.
    - Acceso al módulo admin según roles y token JWT.
4. Verificar que las peticiones al backend se realizan correctamente:
    - Sin errores de CORS.
    - Sin rutas 404 en la API.
    - Autenticación y autorización funcionando (tokens válidos, expiración).

---

9. Entorno operativo

- **Frontend**:
    - Navegadores soportados:
        - Google Chrome (últimas versiones).
        - Microsoft Edge (Chromium).
        - Mozilla Firefox.
        - Safari (macOS/iOS).
    - Tipo de aplicación:
        - SPA (Single Page Application) en Angular.
        - Servido actualmente desde Azure Static Web Apps.
- **Backend**:
    - Aplicación Spring Boot auto-contenida (JAR) o empaquetada en contenedor Docker.
    - Desplegada actualmente en Azure App Service.
    - API REST accesible bajo una ruta base (`/okBranding/...`).
    - Autenticación y autorización mediante JWT.
    - Uso de Azure Database for MySQL Flexible Server como base de datos transaccional.

---

10. Planes de mantenimiento y soporte del software

10.1 Mantenimiento correctivo

Frontend:

- Detección de errores mediante:
    - Reportes de usuarios.
    - Logs de consola del navegador.
    - Herramientas de monitoreo de errores (si se integran).
- Correcciones en componentes, servicios y estilos, y despliegue de nuevas versiones.

Backend:

- Monitorización de logs de aplicación (`logging.file.name`) y métricas de Actuator.

- Corrección de bugs en controladores, servicios, repositorios y entidades.
- Aplicación de parches de seguridad en dependencias (actualización de `pom.xml`).

10.2 Mantenimiento evolutivo

- Incorporación de nuevas funcionalidades:
  - Nuevos módulos en el admin (gestión de nuevas entidades).
  - Nuevas vistas o filtros para el cliente.
  - Nuevos endpoints en el backend para soportar nuevas operaciones.
- Refactor de código:
  - En frontend: componentes, servicios, optimización de carga.
  - En backend: servicios, mappers, repositorios, optimización de consultas.

10.3 Soporte

- Canal de soporte:
  - Correo, sistema de tickets (Jira, GitHub Issues, etc.) o similar.
- Definición de SLA:
  - Tiempos objetivo de respuesta y resolución según criticidad de la incidencia.
- Escalamiento:
  - Procedimiento para escalar incidencias críticas a equipo de desarrollo o equipo de infraestructura.

---

11. Documentación del respaldo de datos

El frontend no almacena datos de negocio permanente; estos se guardan en la base de datos del backend.

La documentación de respaldo de datos aplica principalmente a:

- Base de datos del backend (catálogo de productos, categorías, usuarios, cotizaciones, etc.).
- Configuración del backend:
  - `application.properties`.
  - Variables de entorno en servidores o plataformas de despliegue.
- Scripts o configuraciones de infraestructura (Dockerfiles, pipelines CI/CD, etc.).

Desde el punto de vista del frontend:

- Respaldar el código fuente (repositorio Git).
- Respaldar configuraciones de entorno (`environment.*`) y cualquier archivo de configuración adicional.

---

12. Políticas de backup

12.1 Backups de base de datos (backend)

- Tipo y frecuencia: respaldo completo diario de la base de datos MySQL de producción (Azure Database for MySQL Flexible Server) ejecutado a las 02:00 UTC.
- Retención: 30 días en almacenamiento secundario; 7 días en almacenamiento primario.
- Almacenamiento de copias:

    - Primario: snapshots automáticas en Azure Database for MySQL (región Chile Central).
    - Secundario: exportación diaria a Azure Blob Storage (cuenta de almacenamiento redundante RA-GRS), contenedor □ackups-mysql/diarios/.
- Procedimiento de restauración:
    - Restauración puntual (point-in-time restore) desde Azure Portal o CLI hacia un servidor MySQL de recuperación.
    - Validación de integridad ejecutando pruebas de conexión y conteo de tablas críticas (productos, categorías, usuarios, cotizaciones).
- Pruebas periódicas:
    - Restauración mensual de prueba en entorno de staging.
    - Registro de resultados en bitácora de operaciones (SharePoint interno / Jira).

12.2 Backups del backend y frontend

- Código fuente:
    - Repositorios Git en GitHub (OkbrandingFrontend, OkbrandingBackend) con rama main protegida y PR obligatorio.
    - Tags por release (□MAJOR.MINOR.PATCH) y copia espejo semanal en repositorio privado (Azure DevOps).
- Artefactos de build:
    - Backend: JAR publicado en Azure Artifacts con retención de 6 meses.
    - Frontend: build dist/ empaquetado y almacenado en Azure Blob Storage (□rtefactos-frontend/) con retención de 6 meses.
- Configuración:
    - Variables sensibles en App Settings (Azure App Service y Azure Static Web Apps).
    - Respaldo semanal de plantillas □pplication.properties y environment.* en SharePoint interno con control de versiones.

13. Plan de capacitaciónón

13.1 Usuarios administradores

Contenidos:

- Acceso al módulo admin.
- Gestión de:
    - Productos.
    - Categorías.
    - Colores.
    - Imágenes del carrusel de inicio (banners del home).
    - Usuarios (según las funcionalidades expuestas en el módulo).
- Uso de formularios de creación/edición y visibilidad de cambios en el frontend cliente.

Formato:

- Sesión práctica (presencial o remota).
- Manual breve o guía paso a paso.

13.2 Usuarios clientes

Contenidos:

- Navegación por categorías.
- Consulta de detalle de producto.
- Uso del botón “Cotizar este producto”.
- Proceso de registro e inicio de sesión.

Formato:

- Manual de usuario (Drive): https://drive.google.com/file/d/15x6qnv3j073ofSaLfleB-MAx5FXBi8rt/view?usp=drive_link

13.3 Equipo técnico

Contenidos:

- Arquitectura general (Angular + Spring Boot + MySQL en Azure).
- Flujo de autenticación JWT (login, obtención y envío de token).
- Proceso de despliegue (build, configuración, despliegue en Azure Static Web Apps y Azure App Service).
- Procedimientos de backup y restauración.

Formato:

- Documentación técnica + sesiones técnicas con el equipo de desarrollo.

---

14. Migración de archivos principales

14.1 Frontend

Archivos principales del frontend:

- Imágenes:
  - Logo.
  - Banners del carrusel.
  - Imágenes de productos.
- Archivos en `src/assets/`.
- Archivos de configuración de entornos `environment.*`.

Para una nueva versión:

- Verificar que las rutas de imágenes (`assets/img/...`) se mantienen o se actualizan correctamente.
- Si se migran imágenes a un servidor externo/CDN, actualizar las URLs utilizadas por el frontend.

14.2 Backend

Archivos principales del backend:

- Código fuente en `src/main/java/com/okBranding/back/...` (controladores, servicios, entidades, etc.).
- Recursos en `src/main/resources/`:

    - `application.properties`.
    - Otros recursos de configuración.
- Dockerfile (si se usa contenedorización).
- Scripts de despliegue (si existen).

Al migrar a una nueva versión:

- Mantener versiones anteriores del JAR o imagen Docker como fallback.
- Versionar cambios en configuración (`application.properties`, variables de entorno) y documentarlos.

---

15. Parametrización

15.1 Frontend

La parametrización del frontend se realiza principalmente a través de:

- `src/environments/environment.ts`
- `src/environments/environment.prod.ts`

Parámetros típicos:

- `apiUrl`: URL base del backend.
- `production: true/false`.
- Otros flags o URLs de recursos según necesidades.

Opcionalmente se pueden agregar:

- Archivos JSON de configuración en `assets/` (por ejemplo `assets/config.json`) para textos, colores u otros parámetros que se quieran modificar sin recompilar, cargándolos dinámicamente desde el código.

15.2 Backend

La parametrización del backend se realiza mediante:

- `application.properties`.
- Variables de entorno leídas desde `application.properties` (por ejemplo `JDBC_DATABASE_URL`, `PORT`).

Parámetros típicos:

- Conexión a base de datos (`spring.datasource.*`).
- Configuración de JPA/Hibernate.
- Puerto de escucha (`server.port`).
- Configuración de logging.
- En una siguiente mejora, también:
    - Clave secreta JWT.
    - Parámetros de correo (`spring.mail.*`), si se utiliza.

---

16. Plan de liberación del sistema de información

16.1 Flujo de liberación

1. Desarrollo y pruebas en entorno local (frontend y backend).
2. Deploy en entorno de pruebas (staging):
   - Backend apuntando a base de datos de pruebas.
   - Frontend apuntando al backend de staging.
3. Pruebas funcionales e integración (frontend + backend):
   - Casos de uso principales (navegación, cotización, login, administración).
4. Aprobación del responsable funcional.
5. Build de producción:
   - Frontend:

     ```
     ng build --configuration production
     ```

   - Backend:

     ```
     ./mvnw clean package -DskipTests
     ```

6. Despliegue de artefactos en producción:
   - Frontend en Azure Static Web Apps (o infraestructura equivalente).
   - Backend en Azure App Service (o infraestructura equivalente como contenedores).
7. Pruebas rápidas de humo (smoke tests) en producción:
   - Acceso al home.
   - Navegación a categorías y productos.
   - Proceso de login y acceso a módulo admin.
   - Operaciones básicas de administración.

16.2 Versionado

- Usar versionado semántico:
  - `MAJOR.MINOR.PATCH` (por ejemplo, `1.3.0`).
- Etiquetar releases en los repositorios:

  ```
  git tag v1.3.0
  git push origin v1.3.0
  ```

- Mantener alineadas las versiones de frontend y backend (por ejemplo `frontend-1.3.0` y `backend-1.3.0` o un número de versión común en la documentación de entrega).

---

17. Gestión de entrega

Para cada entrega (release) se debe:

- Registrar:
  - Versión del frontend.
  - Versión del backend.
  - Fecha y hora de despliegue.
  - Entorno de despliegue (staging/producción).
  - Responsable técnico del despliegue.
- Adjuntar:
  - Notas de versión (cambios incluidos).
  - Estado de pruebas (superadas/no superadas, incidencias abiertas).
- Soportar rollback:
  - Indicar qué versión anterior se usará en caso de revertir.

El registro puede llevarse en:

- Un documento compartido.
- Un sistema de tickets/proyectos (Jira, GitHub Projects, etc.).

---

18. Requerimientos (resumen)

18.1 Funcionales (cliente)

- Visualizar carrusel de la página de inicio.
- Visualizar categorías y productos por categoría.
- Visualizar detalle de producto y generar cotización.
- Registrarse e iniciar sesión.
- Interactuar con la API de backend para obtener información de productos, categorías, etc.

18.2 Funcionales (administrador)

- Gestionar productos, categorías, colores.
- Gestionar imágenes del carrusel del home.
- Gestionar usuarios (creación, actualización, activación/desactivación según funcionalidad disponible).
- Cerrar sesión y volver al index público.
- Consumir endpoints protegidos del backend mediante token JWT.

18.3 No funcionales

- Rendimiento aceptable en navegadores modernos.
- Diseño responsivo (adaptado a escritorio y dispositivos móviles).
- Seguridad básica:
  - Rutas admin protegidas por `authGuard` en el frontend.
  - Autenticación y autorización en el backend mediante Spring Security + JWT.
  - Manejo de token de autenticación en `sessionStorage` o similar en el frontend.
- Disponibilidad y resiliencia:
  - Uso de backups regulares de la base de datos.
  - Supervisión y logs en backend.

19. Instalador

Este proyecto no cuenta con un instalador tradicional (EXE/MSI).

La "instalación" se realiza mediante:

- Frontend:
  - Comandos de build:

```
npm install
ng build --configuration production
```

  - Copia de los artefactos generados en `dist/` al servidor web.
- Backend:
  - Comandos de build:

```
./mvnw clean package -DskipTests
```

  - Ejecución del JAR o despliegue de la imagen Docker.

Opcionalmente se puede automatizar con:

- Scripts de despliegue (bash, PowerShell).
- Pipelines CI/CD (GitHub Actions, GitLab CI, Azure DevOps, etc.).

---

20. Archivo de configuración

Los principales archivos de configuración del sistema son:

- Frontend:
  - `src/environments/environment.ts`
  - `src/environments/environment.prod.ts`
- Backend:
  - `src/main/resources/application.properties`
  - Variables de entorno en el entorno de ejecución del backend.

En ellos se define, entre otros:

- URL base del backend (`apiUrl` en el frontend).
- Parámetros de conexión a la base de datos.
- Parámetros de entorno específicos para desarrollo/producción (puertos, logging, etc.).

Adicionalmente:

- Se pueden crear archivos de configuración en `assets/` del frontend (por ejemplo `assets/config.json`) que se lean en tiempo de ejecución para parámetros no críticos.

- En el backend se pueden añadir más propiedades para JWT, correo, límites de carga, etc., siempre parametrizadas y no embebidas como literales sensibles en el código.

Fin del documento