

MANUAL 1: GUÍA DE INSTALACIÓN Y CONFIGURACIÓN - SISTEMA CONA

INFORMACIÓN DEL DOCUMENTO

Fecha de Creación: 21 de Julio de 2025

Proyecto: Sistema CONA (Gestión CONAVEG)

Audiencia: Desarrolladores, DevOps, Administradores de Sistema

Nivel: Intermedio - Avanzado

Tiempo Estimado: 2-4 horas (instalación completa)

Última Actualización: 21 de Julio de 2025

OBJETIVOS DE APRENDIZAJE

Al finalizar este manual, serás capaz de:

- ☒ Instalar y configurar completamente el Sistema CONA
 - ☒ Configurar la base de datos MariaDB para desarrollo y producción
 - ☒ Establecer variables de entorno y perfiles de configuración
 - ☒ Configurar el sistema de seguridad (BCrypt, JWT)
 - ☒ Configurar servicios externos (email, SMTP)
 - ☒ Verificar la instalación y resolver problemas comunes
 - ☒ Optimizar la configuración para diferentes entornos
-

REQUISITOS PREVIOS

Conocimientos Necesarios:

- Conocimientos básicos de Java y Spring Boot
- Experiencia con bases de datos relacionales
- Familiaridad con Maven y línea de comandos
- Conceptos básicos de desarrollo web y APIs REST

Acceso Requerido:

- Permisos de administrador en el sistema operativo
 - Acceso a internet para descargar dependencias
 - Acceso al repositorio del proyecto (si aplica)
-

REQUISITOS DEL SISTEMA

Software Obligatorio:

Java Development Kit (JDK)

- ☒ Versión: Java 21 LTS (recomendado)
- ☒ Alternativas: Java 17 LTS (mínimo)
- ☒ Distribución: OpenJDK, Oracle JDK, o Amazon Corretto

Apache Maven




- ☒ Versión: 3.8.0 o superior
- ☒ Propósito: Gestión de dependencias y construcción del proyecto

Base de Datos MariaDB




- ☒ Versión: 10.5 o superior (recomendado 10.11 LTS)
- ☒ Alternativa: MySQL 8.0+ (compatible)

Software Recomendado:




IDE de Desarrollo

-  IntelliJ IDEA (Ultimate o Community)
-  Eclipse IDE for Enterprise Java Developers
-  Visual Studio Code (con extensiones Java)

Herramientas de Base de Datos

-  DBeaver (multiplataforma, gratuito)
-  HeidiSQL (Windows)
-  phpMyAdmin (web-based)



Herramientas de Testing de APIs

-  Postman (recomendado)
-  Insomnia
-  cURL (línea de comandos)





Requisitos de Hardware:

Desarrollo:

-  RAM: 8GB mínimo, 16GB recomendado

-  Almacenamiento: 10GB libres mínimo
-  CPU: 4 cores recomendado

Producción:

-  RAM: 16GB mínimo, 32GB recomendado
-  Almacenamiento: 100GB+ (depende del volumen de datos)
-  CPU: 8 cores recomendado
-  Red: Conexión estable a internet

INSTALACIÓN DE DEPENDENCIAS

PASO 1: Instalación de Java JDK 21

Windows:

```
# Opción 1: Usar Chocolatey (recomendado)
choco install openjdk21

# Opción 2: Descarga manual desde https://adoptium.net/
# Seguir wizard de instalación

# Verificar instalación
java -version
javac -version
```

Linux (Ubuntu/Debian):

```
# Actualizar repositorios
sudo apt update

# Instalar OpenJDK 21
sudo apt install openjdk-21-jdk

# Configurar JAVA_HOME
echo 'export JAVA_HOME=/usr/lib/jvm/java-21-openjdk-amd64' >> ~/.bashrc
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> ~/.bashrc
source ~/.bashrc

# Verificar instalación
java -version
javac -version
```

macOS:

```
# Usar Homebrew
brew install openjdk@21

# Agregar al PATH
echo 'export PATH="/opt/homebrew/opt/openjdk@21/bin:$PATH"' >> ~/.zshrc
```

```
source ~/.zshrc
```

```
# Verificar instalación  
java -version  
javac -version
```

PASO 2: Instalación de Maven

Windows:

```
# Usar Chocolatey  
choco install maven
```

```
# Verificar instalación  
mvn -version
```

Linux:

```
# Ubuntu/Debian  
sudo apt install maven
```

```
# CentOS/RHEL  
sudo yum install maven
```

```
# Verificar instalación  
mvn -version
```

macOS:

```
# Usar Homebrew  
brew install maven
```

```
# Verificar instalación  
mvn -version
```

PASO 3: Instalación de MariaDB

Windows:

```
# Descargar desde https://mariadb.org/download/  
# Ejecutar instalador MSI  
# Durante la instalación:  
# - Configurar contraseña para root  
# - Habilitar UTF8 como charset por defecto  
# - Habilitar networking (puerto 3306)  
  
# Verificar instalación  
mysql -u root -p
```

Linux (Ubuntu/Debian):

```
# Instalar MariaDB Server  
sudo apt update  
sudo apt install mariadb-server mariadb-client  
  
# Configurar seguridad básica
```

```
sudo mysql_secure_installation

# Crear usuario para la aplicación
sudo mysql -u root -p

# En el prompt de MySQL:
CREATE USER 'cona_user'@'localhost' IDENTIFIED BY 'cona_password_secure';
GRANT ALL PRIVILEGES ON *.* TO 'cona_user'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

macOS:

```
# Usar Homebrew
brew install mariadb

# Iniciar servicio
brew services start mariadb

# Configurar seguridad
mysql_secure_installation
```

PASO 4: Configuración de MariaDB

Crear Base de Datos del Proyecto:

```
-- Conectar como root
mysql -u root -p

-- Crear base de datos
CREATE DATABASE conaveg_db DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

-- Crear usuario específico para la aplicación
CREATE USER 'cona_user'@'localhost' IDENTIFIED BY 'TU_PASSWORD_SEGURO_AQUI';

-- Otorgar permisos
GRANT ALL PRIVILEGES ON conaveg_db.* TO 'cona_user'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER ON conaveg_db.* TO 'cona_user'@'localhost';

-- Aplicar cambios
FLUSH PRIVILEGES;

-- Verificar usuario
SELECT User, Host FROM mysql.user WHERE User = 'cona_user';

-- Salir
EXIT;
```

Verificar Conectividad:

```
# Probar conexión con usuario de la aplicación
mysql -u cona_user -p conaveg_db
```

```
# Verificar charset de la base de datos
SHOW CREATE DATABASE conaveg_db;
```

OBTENCIÓN DEL CÓDIGO FUENTE

Método 1: Clonación desde Git (Recomendado)

```
# Clonar repositorio
git clone https://github.com/LucianolejandroUtp/conaveg-backend.git

# Entrar al directorio
cd conaveg-backend

# Verificar estructura del proyecto
ls -la
```

Método 2: Descarga de Archivo ZIP

```
# Si no tienes acceso a Git
# Descargar ZIP desde GitHub y extraer

# Entrar al directorio extraído
cd cona-main
```

Verificación de la Estructura del Proyecto:

```
cona/
├── docs/                # Documentación del proyecto
├── files/              # Archivos de la aplicación
├── logs/              # Logs de la aplicación
├── src/               # Código fuente
│   ├── main/
│   │   ├── java/com/conaveg/cona/ # Código Java
│   │   └── resources/             # Recursos y configuración
│   └── test/                    # Tests
├── target/                    # Archivos compilados (se genera)
├── pom.xml                   # Configuración de Maven
├── README.md                 # Documentación principal
├── start-dev.bat             # Script de inicio (Windows)
└── start-dev.sh              # Script de inicio (Linux/Mac)
```

CONFIGURACIÓN DE LA APLICACIÓN

PASO 1: Configuración Principal (application.properties)

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

```
# =====
# CONFIGURACIÓN BÁSICA DE LA APLICACIÓN
# =====
spring.application.name=cona
server.port=8080
```

```
server.servlet.context-path=/conaveg

# =====
# CONFIGURACIÓN DE BASE DE DATOS - DESARROLLO
# =====
spring.datasource.url=jdbc:mariadb://localhost:3306/conaveg_db?useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC
spring.datasource.username=cona_user
spring.datasource.password=TU_PASSWORD_AQUI
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver

# =====
# CONFIGURACIÓN DE JPA/HIBERNATE
# =====
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=false
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
spring.jpa.properties.hibernate.jdbc.time_zone=UTC

# =====
# CONFIGURACIÓN DE POOL DE CONEXIONES
# =====
spring.datasource.hikari.maximum-pool-size=20
spring.datasource.hikari.minimum-idle=5
spring.datasource.hikari.idle-timeout=300000
spring.datasource.hikari.max-lifetime=1200000
spring.datasource.hikari.connection-timeout=20000

# =====
# CONFIGURACIÓN DE SEGURIDAD JWT
# =====
app.jwt.secret=cona_secret_key_change_in_production_must_be_very_long_and_secure_2025
app.jwt.expiration=86400000
app.jwt.refresh-window=900000

# =====
# CONFIGURACIÓN DE BCrypt
# =====
app.security.bcrypt.strength=12

# =====
# CONFIGURACIÓN DE RATE LIMITING
# =====
app.security.rate-limit.enabled=true
app.security.rate-limit.max-attempts-per-ip=10
app.security.rate-limit.max-attempts-per-email=20
app.security.rate-limit.window-size-hours=1
app.security.rate-limit.block-duration-minutes=15

# =====
# CONFIGURACIÓN DE ARCHIVOS
# =====
app.file.upload-dir=files
spring.servlet.multipart.max-file-size=10MB
```

```

spring.servlet.multipart.max-request-size=10MB

# =====
# CONFIGURACIÓN DE EMAIL (SMTP)
# =====
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=tu_email@gmail.com
spring.mail.password=tu_app_password
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
app.mail.from=noreply@conaveg.com

# =====
# CONFIGURACIÓN DE LOGGING
# =====
logging.level.com.conaveg.cona=INFO
logging.level.org.springframework.security=WARN
logging.level.org.hibernate.SQL=WARN
logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n
logging.file.name=logs/spring.log

# =====
# CONFIGURACIÓN DE ACTUATOR (MONITOREO)
# =====
management.endpoints.web.exposure.include=health,metrics,info
management.endpoint.health.show-details=when-authorized
management.metrics.enable.jvm=true

# =====
# CONFIGURACIÓN DE SWAGGER/OPENAPI
# =====
springdoc.api-docs.path=/api-docs
springdoc.swagger-ui.path=/swagger-ui.html
springdoc.swagger-ui.operationsSorter=method

```

PASO 2: Configuración para Desarrollo (application-dev.properties)

Archivo: `src/main/resources/application-dev.properties`

```

# =====
# CONFIGURACIÓN ESPECÍFICA DE DESARROLLO
# =====

# Modo desarrollo sin autenticación (SOLO DESARROLLO)
app.dev.skip-authentication=true

# Logging más detallado en desarrollo
logging.level.com.conaveg.cona=DEBUG
logging.level.org.springframework.security=DEBUG
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

# Base de datos - mostrar SQL en desarrollo
spring.jpa.show-sql=true

```



```

spring.jpa.properties.hibernate.format_sql=true

# Pool de conexiones reducido para desarrollo
spring.datasource.hikari.maximum-pool-size=5
spring.datasource.hikari.minimum-idle=2

# Configuración de cache deshabilitada
spring.jpa.properties.hibernate.cache.use_second_level_cache=false
spring.jpa.properties.hibernate.cache.use_query_cache=false

# Hot reload habilitado
spring.devtools.restart.enabled=true
spring.devtools.livereload.enabled=true

# Rate limiting más permisivo en desarrollo
app.security.rate-limit.max-attempts-per-ip=100
app.security.rate-limit.max-attempts-per-email=100
app.security.rate-limit.block-duration-minutes=1

# Actuator expuesto completamente en desarrollo
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always

```

PASO 3: Configuración para Producción (application-prod.properties)

Archivo: `src/main/resources/application-prod.properties`

```

# =====
# CONFIGURACIÓN ESPECÍFICA DE PRODUCCIÓN
# =====

# Seguridad estricta
app.dev.skip-authentication=false

# Logging optimizado para producción
logging.level.com.conaveg.cona=WARN
logging.level.org.springframework.security=WARN
logging.level.org.hibernate=WARN
logging.level.org.hibernate.SQL=ERROR

# Base de datos - sin mostrar SQL
spring.jpa.show-sql=false
spring.jpa.properties.hibernate.format_sql=false

# DDL auto deshabilitado en producción
spring.jpa.hibernate.ddl-auto=validate

# Pool de conexiones optimizado para producción
spring.datasource.hikari.maximum-pool-size=50
spring.datasource.hikari.minimum-idle=10
spring.datasource.hikari.idle-timeout=600000
spring.datasource.hikari.max-lifetime=1800000

# Cache habilitado
spring.jpa.properties.hibernate.cache.use_second_level_cache=true

```

```
spring.jpa.properties.hibernate.cache.use_query_cache=true
```

```
# Rate limiting estricto
```

```
app.security.rate-limit.enabled=true
```

```
app.security.rate-limit.max-attempts-per-ip=5
```

```
app.security.rate-limit.max-attempts-per-email=10
```

```
app.security.rate-limit.block-duration-minutes=30
```

```
# Actuator mínimo en producción
```

```
management.endpoints.web.exposure.include=health,metrics
```

```
management.endpoint.health.show-details=never
```

```
# Configuración de seguridad adicional
```

```
server.error.include-stacktrace=never
```

```
server.error.include-message=never
```

```
server.error.include-binding-errors=never
```

PASO 4: Configuración para Testing (application-test.properties)

Archivo: `src/test/resources/application-test.properties`

```
# =====
```

```
# CONFIGURACIÓN ESPECÍFICA DE TESTING
```

```
# =====
```

```
# Base de datos en memoria para tests
```

```
spring.datasource.url=jdbc:h2:mem:testdb;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
```

```
spring.datasource.driverClassName=org.h2.Driver
```

```
spring.datasource.username=sa
```

```
spring.datasource.password=
```

```
# JPA para H2
```

```
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
```

```
spring.jpa.hibernate.ddl-auto=create-drop
```

```
spring.jpa.show-sql=false
```

```
# Logging mínimo en tests
```

```
logging.level.com.conaveg.cona=WARN
```

```
logging.level.org.springframework=WARN
```

```
logging.level.org.hibernate=WARN
```

```
# BCrypt con menor costo para tests más rápidos
```

```
app.security.bcrypt.strength=4
```

```
# JWT con expiración corta para tests
```

```
app.jwt.expiration=3600000
```

```
app.jwt.secret=test_secret_key_for_testing_only
```

```
# Rate limiting deshabilitado en tests
```

```
app.security.rate-limit.enabled=false
```

```
# Email mock en tests
```

```
spring.mail.host=localhost
```

```
spring.mail.port=2525
```

CONFIGURACIÓN DE SEGURIDAD

Variables de Entorno Críticas

Crear archivo `.env` (NO versionar):

```
# Archivo: .env (en la raíz del proyecto)

# Base de datos
DB_URL=jdbc:mariadb://localhost:3306/conaveg_db
DB_USERNAME=cona_user
DB_PASSWORD=TU_PASSWORD_MUY_SEGURO_AQUI

# JWT
JWT_SECRET=cona_jwt_secret_muy_largo_y_seguro_para_produccion_2025_cambiar_obligatorio
JWT_EXPIRATION=86400000

# Email
MAIL_USERNAME=tu_email@gmail.com
MAIL_PASSWORD=tu_app_password_de_gmail

# Entorno
SPRING_PROFILES_ACTIVE=dev
```

Configurar variables en el sistema:

Windows:

```
# Configurar variables de entorno de sistema
setx SPRING_PROFILES_ACTIVE "dev"
setx DB_PASSWORD "tu_password_seguro"
setx JWT_SECRET "tu_jwt_secret_muy_largo_y_seguro"

# Para sesión actual
set SPRING_PROFILES_ACTIVE=dev
```

Linux/macOS:

```
# Agregar al ~/.bashrc o ~/.zshrc
export SPRING_PROFILES_ACTIVE=dev
export DB_PASSWORD="tu_password_seguro"
export JWT_SECRET="tu_jwt_secret_muy_largo_y_seguro"

# Aplicar cambios
source ~/.bashrc
```

Configuración de JWT Seguro

Generar clave JWT segura:

```
# Método 1: Usar OpenSSL
openssl rand -base64 64
```

```
# Método 2: Usar Java
java -cp ".: *" -c
"System.out.println(Java.util.Base64.getEncoder().encodeToString(Java.security.SecureRandom.getInstanceStrong().generateSeed(64)))"
```

```
# Método 3: Online (usar solo para desarrollo)
# https://generate-random.org/api-key-generator
```

IMPORTANTE:

- La clave JWT debe tener mínimo 256 bits (32 caracteres)
- Cambiar OBLIGATORIAMENTE en producción
- No versionar nunca en Git

CONFIGURACIÓN DE EMAIL/SMTP

Configuración para Gmail:

```
# En application.properties
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=tu_cuenta@gmail.com
spring.mail.password=tu_app_password
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.ssl.trust=smtp.gmail.com
```

Configurar App Password en Gmail:

1. Ir a Cuenta de Google → Seguridad
2. Habilitar Verificación en 2 pasos
3. Generar Contraseña de aplicación
4. Usar esa contraseña en la configuración

Configuración para Outlook/Hotmail:

```
spring.mail.host=smtp-mail.outlook.com
spring.mail.port=587
spring.mail.username=tu_cuenta@outlook.com
spring.mail.password=tu_password
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

Configuración para Servidor SMTP Personalizado:

```
spring.mail.host=mail.tu-empresa.com
spring.mail.port=587
spring.mail.username=noreply@tu-empresa.com
spring.mail.password=password_del_servidor
```

```
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.ssl.enable=false
```

CONSTRUCCIÓN Y EJECUCIÓN

PASO 1: Compilación del Proyecto

Compilación Básica:

```
# Entrar al directorio del proyecto
cd cona

# Limpiar y compilar
mvn clean compile

# Verificar que no hay errores de compilación
echo $? # Debe retornar 0 en Linux/Mac
echo %ERRORLEVEL% # Debe retornar 0 en Windows
```

Ejecutar Tests:

```
# Ejecutar todos los tests
mvn test

# Ejecutar tests específicos
mvn test -Dtest=UserServiceTest
mvn test -Dtest=*IntegrationTest

# Ejecutar tests con perfil específico
mvn test -Dspring.profiles.active=test
```

Empaquetar la Aplicación:

```
# Crear JAR ejecutable
mvn clean package

# Saltar tests durante empaquetado (si es necesario)
mvn clean package -DskipTests

# Verificar JAR creado
ls -la target/*.jar
```

PASO 2: Ejecución de la Aplicación

Método 1: Maven (Desarrollo):

```
# Ejecutar con perfil de desarrollo
mvn spring-boot:run -Dspring.profiles.active=dev

# Ejecutar con variables de entorno
SPRING_PROFILES_ACTIVE=dev mvn spring-boot:run
```

```
# Ejecutar con memoria personalizada
mvn spring-boot:run -Dspring.profiles.active=dev -Xmx2g
```

Método 2: JAR Ejecutable:

```
# Ejecutar JAR directamente
java -jar target/cona-1.0.0.jar --spring.profiles.active=dev

# Con configuración de memoria
java -Xmx2g -Xms512m -jar target/cona-1.0.0.jar --spring.profiles.active=prod

# Con variables de entorno
SPRING_PROFILES_ACTIVE=prod java -jar target/cona-1.0.0.jar
```

Método 3: Scripts de Inicio:

Windows (**start-dev.bat**):

```
@echo off
echo Iniciando CONA en modo desarrollo...
set SPRING_PROFILES_ACTIVE=dev
mvn spring-boot:run
pause
```

Linux/Mac (**start-dev.sh**):

```
#!/bin/bash
echo "Iniciando CONA en modo desarrollo..."
export SPRING_PROFILES_ACTIVE=dev
mvn spring-boot:run

# Hacer ejecutable
chmod +x start-dev.sh

# Ejecutar
./start-dev.sh
```

✓ VERIFICACIÓN DE LA INSTALACIÓN

PASO 1: Verificación Básica de Servicios

Verificar que la aplicación inició correctamente:

```
# Verificar que el puerto 8080 está en uso
netstat -tulpn | grep :8080 # Linux
netstat -an | findstr :8080 # Windows

# Verificar logs de inicio
tail -f logs/spring.log

# Buscar mensaje de inicio exitoso
grep "Started ConaApplication" logs/spring.log
```

Health Check de la Aplicación:

```
# Verificar endpoint de salud
curl -X GET http://localhost:8080/conaveg/actuator/health

# Respuesta esperada:
# {"status": "UP", "components": {"db": {"status": "UP"}, "diskSpace": {"status": "UP"}}
```

PASO 2: Verificación de Base de Datos

Conectividad con la Base de Datos:

```
# Verificar conexión directa
mysql -u cona_user -p conaveg_db -e "SELECT 1 as test;"

# Verificar tablas creadas por la aplicación
mysql -u cona_user -p conaveg_db -e "SHOW TABLES;"
```

Verificar Datos de Prueba (si existen):

```
-- Conectar a la base de datos
mysql -u cona_user -p conaveg_db

-- Verificar tablas principales
DESCRIBE users;
DESCRIBE roles;
DESCRIBE empleados;

-- Verificar datos iniciales
SELECT COUNT(*) FROM roles;
SELECT COUNT(*) FROM users;
```

PASO 3: Verificación de APIs

Documentación Swagger:

```
# Abrir en navegador
open http://localhost:8080/conaveg/swagger-ui/index.html
# o
firefox http://localhost:8080/conaveg/swagger-ui/index.html
```

Endpoints Básicos:

```
# Verificar endpoint de información
curl -X GET http://localhost:8080/conaveg/actuator/info

# Si está en modo desarrollo, verificar endpoint de desarrollo
curl -X GET http://localhost:8080/conaveg/api/dev/status

# Respuesta esperada en modo dev:
# {"message": "Modo desarrollo activo", "skipAuthentication": true}
```

Test de Endpoints Públicos:

```
# Verificar endpoint de roles (si está en modo dev)
curl -X GET http://localhost:8080/conaveg/api/roles
```

```
# Verificar endpoint de usuarios (si está en modo dev)
curl -X GET http://localhost:8080/conaveg/api/users
```

PASO 4: Verificación de Seguridad

Verificar JWT y Autenticación (en modo producción):

```
# Intentar acceso sin token (debe fallar con 401/403)
curl -X GET http://localhost:8080/conaveg/api/users

# Login de prueba (si hay usuarios de prueba)
curl -X POST http://localhost:8080/conaveg/api/auth/login \
-H "Content-Type: application/json" \
-d '{"email": "admin@test.com", "password": "password123"}'
```

Verificar BCrypt:

```
# En los logs, buscar líneas relacionadas con BCrypt
grep -i "bcrypt\|password" logs/spring.log
```

CONFIGURACIÓN DEL IDE

IntelliJ IDEA

Importar Proyecto:

1. File → Open → Seleccionar carpeta del proyecto
2. Esperar a que Maven sincronice dependencias
3. File → Project Structure → Verificar Project SDK: Java 21

Configurar Run Configuration:

1. Run → Edit Configurations
2. Add New → Spring Boot
3. Main class: `com.conaveg.cona.ConApplication`
4. Environment variables: `SPRING_PROFILES_ACTIVE=dev`
5. VM options: `-Xmx2g -Xms512m`

Plugins Recomendados:

- Spring Boot (ya incluido en Ultimate)
- Database Tools and SQL (para conectar a MariaDB)
- HTTP Client (para testing de APIs)
- GitToolBox (para Git avanzado)

Visual Studio Code

Extensiones Requeridas:

```
# Instalar extensiones desde línea de comandos
code --install-extension vscjava.vscode-java-pack
code --install-extension pivotal.vscode-spring-boot
code --install-extension vscjava.vscode-spring-initializr
code --install-extension vscjava.vscode-spring-boot-dashboard
```

Configuración de Workspace ([.vscode/settings.json](#)):

```
{
  "java.configuration.updateBuildConfiguration": "automatic",
  "java.compile.nullAnalysis.mode": "automatic",
  "java.format.settings.url":
    "https://raw.githubusercontent.com/google/styleguide/gh-pages/eclipse-java-google-style.xml",
  "spring-boot.ls.problem.application-properties.unknown-property": "warning",
  "files.exclude": {
    "**/target": true,
    "**/.mvn": true
  }
}
```

Configuración de Launch ([.vscode/launch.json](#)):

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "java",
      "name": "CONA Development",
      "request": "launch",
      "mainClass": "com.conaveg.cona.ConApplication",
      "projectName": "cona",
      "env": {
        "SPRING_PROFILES_ACTIVE": "dev"
      },
      "vmArgs": "-Xmx2g -Xms512m"
    }
  ]
}
```

Eclipse IDE

Importar Proyecto Maven:

1. File → Import → Existing Maven Projects
2. Seleccionar carpeta del proyecto
3. Esperar sincronización de Maven

Configurar Run Configuration:

1. Run → Run Configurations
 2. Java Application → New
 3. Main class: `com.conaveg.cona.ConapApplication`
 4. Environment → Add: `SPRING_PROFILES_ACTIVE=dev`
-



TROUBLESHOOTING COMÚN

Problema 1: Error de Conexión a Base de Datos

Síntomas:

```
SQLException: Access denied for user 'cona_user'@'localhost'
```

Soluciones:

```
# 1. Verificar que MariaDB esté ejecutándose
sudo systemctl status mariadb # Linux
brew services list | grep mariadb # macOS

# 2. Verificar credenciales
mysql -u cona_user -p conaveg_db

# 3. Recrear usuario si es necesario
mysql -u root -p
DROP USER IF EXISTS 'cona_user'@'localhost';
CREATE USER 'cona_user'@'localhost' IDENTIFIED BY 'nueva_password';
GRANT ALL PRIVILEGES ON conaveg_db.* TO 'cona_user'@'localhost';
FLUSH PRIVILEGES;
```

Problema 2: Error de Puerto en Uso

Síntomas:

```
Port 8080 was already in use
```

Soluciones:

```
# 1. Encontrar proceso usando puerto 8080
lsof -i :8080 # Linux/macOS
netstat -ano | findstr :8080 # Windows

# 2. Matar proceso
kill -9 PID # Linux/macOS
taskkill /PID PID /F # Windows

# 3. O cambiar puerto en application.properties
server.port=8081
```

Problema 3: OutOfMemoryError

Síntomas:

```
java.lang.OutOfMemoryError: Java heap space
```

Soluciones:

```
# 1. Aumentar memoria heap
export MAVEN_OPTS="-Xmx4g -Xms1g"

# 2. O ejecutar con más memoria
java -Xmx4g -Xms1g -jar target/cona-1.0.0.jar

# 3. Verificar memoria disponible
free -h # Linux
```

Problema 4: Error de Dependencias Maven

Síntomas:

```
Could not resolve dependencies
```

Soluciones:

```
# 1. Limpiar cache de Maven
mvn dependency:purge-local-repository

# 2. Forzar actualización
mvn clean install -U

# 3. Verificar conectividad
ping repo1.maven.org
```

Problema 5: Error de JWT Secret

Síntomas:

```
JWT secret key is too short
```

Soluciones:

```
# 1. Generar nueva clave de 256+ bits
openssl rand -base64 64

# 2. Actualizar en application.properties
app.jwt.secret=NUEVA_CLAVE_MUY_LARGA_AQUI

# 3. Verificar longitud
echo "tu_clave_aqui" | wc -c # Debe ser > 32
```

Problema 6: Error de Encoding/Charset

Síntomas:

Characters showing as ??? or incorrect encoding

Soluciones:

```
-- 1. Verificar charset de la base de datos
SHOW CREATE DATABASE conaveg_db;

-- 2. Recrear con UTF8 si es necesario
DROP DATABASE conaveg_db;
CREATE DATABASE conaveg_db DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

-- 3. Verificar URL de conexión
# Debe incluir: ?useUnicode=true&characterEncoding=UTF-8
```

VERIFICACIÓN FINAL Y CHECKLIST

Checklist de Instalación Completa:

Infraestructura Base:

- ☐ Java JDK 21 instalado y configurado
- ☐ Maven 3.8+ instalado y funcionando
- ☐ MariaDB 10.5+ instalado y ejecutándose
- ☐ IDE configurado con extensiones necesarias

Base de Datos:

- ☐ Base de datos **conaveg_db** creada
- ☐ Usuario **cona_user** creado con permisos correctos
- ☐ Conectividad verificada desde aplicación
- ☐ Charset UTF8MB4 configurado

Configuración de la Aplicación:

- ☐ **application.properties** configurado correctamente
- ☐ Perfiles de desarrollo y producción configurados
- ☐ Variables de entorno configuradas
- ☐ JWT secret configurado (256+ bits)
- ☐ SMTP configurado (si se requiere email)

Compilación y Ejecución:

- ☐ Proyecto compila sin errores (`mvn compile`)
- ☐ Tests pasan correctamente (`mvn test`)
- ☐ JAR se genera correctamente (`mvn package`)
- ☐ Aplicación inicia sin errores

Verificación de Servicios:

- ☐ Health check responde OK
- ☐ Swagger UI accesible
- ☐ Base de datos conectada
- ☐ Endpoints básicos responden
- ☐ Logs se generan correctamente

Seguridad:

- ☐ BCrypt configurado con strength 12
- ☐ JWT funciona correctamente
- ☐ Rate limiting habilitado
- ☐ Variables sensibles no versionadas
- ☐ Modo desarrollo funciona (si aplica)

Comandos de Verificación Rápida:

```
# Test completo en una línea
curl -f http://localhost:8080/conaveg/actuator/health && echo "✅ CONA está funcionando correctamente"

# Verificar todos los servicios críticos
echo "🔍 Verificando servicios..."
java -version && echo "✅ Java OK"
mvn -version && echo "✅ Maven OK"
mysql --version && echo "✅ MariaDB OK"
curl -s http://localhost:8080/conaveg/actuator/health > /dev/null && echo "✅ CONA OK"
```

OPTIMIZACIONES DE PERFORMANCE

Configuración JVM Optimizada:

```
# Para desarrollo (8GB RAM disponible)
export MAVEN_OPTS="-Xmx2g -Xms512m -XX:+UseG1GC -XX:MaxGCPauseMillis=200"

# Para producción (16GB+ RAM disponible)
```

```
java -Xmx8g -Xms2g -XX: +UseG1GC -XX: MaxGCPauseMi l l i s=100 -XX: +UseStri ngDedupl i cati on  
-j ar target/cona-1.0.0.jar
```

Configuración de Base de Datos Optimizada:


```
-- my.cnf optimizado para CONA  
[mysqld]  
innodb_buffer_pool_size = 2G  
innodb_log_file_size = 256M  
innodb_flush_log_at_trx_commit = 2  
query_cache_size = 128M  
max_connections = 200  
thread_cache_size = 8  
table_open_cache = 4000
```

Configuración de Hikari Optimizada:

```
# Para alta concurrencia  
spring.datasource.hikari.maximum-pool-size=50  
spring.datasource.hikari.minimum-idle=10  
spring.datasource.hikari.connection-timeout=20000  
spring.datasource.hikari.idle-timeout=300000  
spring.datasource.hikari.max-lifetime=1200000  
spring.datasource.hikari.leak-detection-threshold=60000
```

SOPORTE Y RECURSOS ADICIONALES




Documentación del Proyecto:






-  [Manual de Desarrollo y API](#)
-  [Manual de Testing](#)
-  [Manual de Seguridad](#)
-  [Manual de Monitoreo](#)

Recursos Externos:

- [Spring Boot Documentation](#)
- [MariaDB Documentation](#)
- [Maven Documentation](#)



Canales de Soporte:

-  Email: soporte-tecnico@conaveg.com
 -  Slack: #cona-soporte
 -  Teams: Canal CONA Desarrollo
-

 Fecha de Creación: 21 de Julio de 2025
  Responsable: Equipo de Documentación CONA
 Estado: Manual Completo y Validado
 Próxima Revisión: 21 de Agosto de 2025

NOTAS FINALES

Este manual ha sido diseñado para proporcionar una guía completa y detallada para la instalación y configuración del Sistema CONA. Siguiendo todos los pasos descritos, tendrás un entorno completamente funcional para desarrollo o producción.

  Recordatorios Importantes:

- Cambiar TODAS las contraseñas por defecto en producción
- Configurar backups automáticos de la base de datos
- Implementar monitoreo de la aplicación
- Mantener actualizadas las dependencias de seguridad

¡El Sistema CONA está listo para ser utilizado! 