Manual Técnico

Sistema de Control de Acceso con QR

Proyecto: Sistema de control de acceso mediante validación de QR

Versión: 1.0

Fecha: 27 de enero de 2026

Equipo de Desarrollo:

Bohorquez Nicolás
Caiza Daniela
Cando Alexander
Cartuche Zoé
Castro Leandro

Repositorio:
https://github.com/Leo-Caz/Grupo1-control-acceso-qr-sqlite

Índice

Resumen Ejecutivo

Este manual técnico documenta el sistema de control de acceso desarrollado en Java utilizando arquitectura MVC. El sistema permite la validación de empleados mediante lectura de códigos QR con una cámara web, registro de accesos en base de datos SQLite y visualización en tiempo real de información. El proyecto está diseñado para entornos empresariales que requieren un control de acceso automatizado y eficiente.

1 Introducción y Visión General

1.1 Propósito del Sistema

El propósito principal de este sistema es automatizar el proceso de control de acceso en una organización empresarial mediante la implementación de un sistema basado en códigos QR. El sistema reemplaza los métodos tradicionales de registro manual, proporcionando:

- Validación instantánea de identidad mediante QR
- Registro automático y timestamp de cada acceso
- Interfaz visual con video en tiempo real
- Base de datos histórica para auditorías

1.2 Alcance del Proyecto

El sistema cubre las siguientes funcionalidades principales:

- Lectura y decodificación de códigos QR mediante cámara web
- Verificación de empleados en base de datos SQLite
- Registro de eventos de entrada
- Visualización de datos del empleado en tiempo real
- Gestión básica de usuarios (CRUD) en base de datos

Límites del sistema:

- No incluye control de acceso multi-nivel por áreas
- No implementa sistemas biométricos adicionales
- No genera reportes estadísticos avanzados
- No incluye integración con sistemas de nómina externos

Componente	Tecnología/Herramienta	Versión
Lenguaje Principal	Java	8+
Arquitectura	Modelo-Vista-Controlador (MVC)	–
Interfaz Gráfica	Java Swing	Incluido en JDK
Base de Datos	SQLite	3.x
Control de Versiones	Git / GitHub	–
Gestión de Dependencias	Maven	3.6+
IDE Principal	Visual Studio Code	–
Procesamiento de QR	ZXing ("Zebra Crossing")	3.5.0 (supuesto)

Cuadro 1: Tecnologías utilizadas en el proyecto

1.3 Tecnologías y Dependencias Principales

2 Arquitectura del Sistema

2.1 Estructura de Carpetas del Proyecto

La organización del proyecto sigue convenciones estándar de Java y Maven:

```
Grupo1-control-acceso-qr-sqlite/
          .gitignore
          pom.xml                          # Configuracion de Maven
          README.md                        # Documentacion basica
          src/
                main/java/
                      controlador/              # Clases controlador
                            AccessController.java
                            CameraController.java
                            QRController.java
                      modelo/                   # Logica de negocio y datos
                            entities/
                                  Empleado.java
                                  RegistroAcceso.java
                            persistence/
                                  DatabaseConnection.java
                                  EmpleadoDAO.java
                                  RegistroAccesoDAO.java
                            services/
                                QRService.java
                                ValidationService.java
                                CameraService.java
                      vista/                    # Interfaz de usuario
                          MainFrame.java
                          CameraPanel.java
                          InfoPanel.java
                          LogPanel.java
                main/resources/
                    config.properties        # Configuraciones
          lib/                             # Dependencias (.jar)
```

```
Database/
      schema.sql                    # Esquema de base de datos
      test_data.sql                 # Datos de prueba
fotos_usuarios/                   # Imagenes de perfil
bin/                              # Archivos compilados
```

Listing 1: Estructura de directorios del proyecto

2.2 Diagrama de Arquitectura MVC

Figura 1: Arquitectura Modelo-Vista-Controlador del sistema

2.3 Flujo de Datos del Sistema

1. El usuario presenta un código QR a la cámara
2. La Vista (CameraPanel) captura el frame y lo envía al Controlador
3. El Controlador (QRController) procesa la imagen con ZXing
4. El ID obtenido se envía al Modelo (ValidationService)
5. El Modelo consulta la base de datos mediante EmpleadoDAO
6. Si es válido, se crea un registro en RegistroAccesoDAO
7. El Controlador actualiza la Vista con los datos del empleado
8. Se muestra la información en tiempo real en la interfaz

3 Configuración del Entorno de Desarrollo

3.1 Requisitos Previos

- **Java Development Kit (JDK):** Versión 8 o superior
- **Visual Studio Code:** Con extensión (Extension Pack for Java)
- **Maven:** Versión 3.6 o superior
- **Git:** Para clonar el repositorio
- **Cámara web:** Funcional y accesible
- **SQLite:** Para manipulación manual de base de datos (opcional)

3.2 Instalación y Configuración Paso a Paso

3.2.1 1. Clonar el Repositorio

```
git clone https://github.com/Leo-Caz/Grupo1-control-acceso-qr-sqlite.git
cd Grupo1-control-acceso-qr-sqlite
```

Listing 2: Clonar repositorio desde GitHub

3.2.2 2. Configurar Dependencias con Maven

```
# Desde la raiz del proyecto (donde esta pom.xml)
mvn clean install
```

Listing 3: Instalar dependencias del proyecto

3.2.3 3. Configurar Base de Datos SQLite

1. Ejecutar el script SQL en `Database/schema.sql`:

```
-- Ejemplo de esquema basico
CREATE TABLE IF NOT EXISTS empleados (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    identificacion VARCHAR(20) UNIQUE NOT NULL,
    departamento VARCHAR(50),
    foto_perfil VARCHAR(255),
    activo BOOLEAN DEFAULT 1,
    fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS registros_acceso (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    empleado_id INTEGER NOT NULL,
    fecha_hora TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    tipo_evento VARCHAR(10) CHECK(tipo_evento IN ('ENTRADA', 'SALIDA'
        )),
    resultado VARCHAR(20) DEFAULT 'EXITOSO',
    FOREIGN KEY (empleado_id) REFERENCES empleados(id)
);
```

Listing 4: Script de creacion de base de datos

2. Configurar la ruta de la base de datos en `DatabaseConnection.java`:

```
public class DatabaseConnection {
    private static final String URL = "jdbc:sqlite:database/
        control_acceso.db";
    private static final String DRIVER = "org.sqlite.JDBC";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL);
    }
}
```

Listing 5: Configuracion de conexion a SQLite

3.2.4 4. Configurar Archivo de Propiedades

Crear `src/main/resources/config.properties`:

```
# Configuracion de la aplicacion
app.name=Control de Acceso QR
app.version=1.0

# Configuracion de base de datos
db.url=jdbc:sqlite:database/control_acceso.db
db.driver=org.sqlite.JDBC

# Configuracion de camara
camera.id=0
camera.width=640
camera.height=480
camera.fps=30

# Configuracion de QR
qr.encoding=UTF-8
qr.charset=ISO-8859-1

# Rutas de archivos
path.images=fotos_usuarios/
path.logs=logs/
```

Listing 6: Archivo de configuracion de propiedades

3.2.5 5. Ejecutar la Aplicación

```
# Opcion 1: Desde Maven
mvn compile
mvn exec:java -Dexec.mainClass="com.controlacceso.MainApp"

# Opcion 2: Desde VS Code
# 1. Abrir la carpeta del proyecto en VS Code
# 2. Asegurarse que la extension Java esta instalada
# 3. Buscar la clase MainApp.java
# 4. Hacer clic en "Run" (triangulo verde)
```

Listing 7: Comandos para ejecutar la aplicacion

4 Guía de Uso para Desarrolladores

4.1 Flujo de Trabajo del Sistema

1. **Inicialización:** La aplicación carga la configuración, inicializa la cámara y conecta a la base de datos.
2. **Espera de QR:** El sistema permanece en estado de espera, mostrando video en tiempo real.
3. **Detección de QR:** Cuando se detecta un código QR, se captura la imagen y se decodifica.
4. **Validación:** El ID decodificado se busca en la tabla `empleados`.
5. **Procesamiento:**
   - Si existe y está activo: Se registra acceso y muestra información
   - Si no existe o está inactivo: Muestra mensaje de acceso denegado
6. **Registro:** El evento se guarda en `registros_acceso`.
7. **Retroalimentación:** La interfaz se actualiza con el resultado.

Figura 2: Flujo de trabajo del sistema de control de acceso

4.2 Esquema Completo de Base de Datos

```
-- TABLA EMPLEADOS
CREATE TABLE empleados (
```

```
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    codigo_qr VARCHAR(100) UNIQUE NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    identificacion VARCHAR(20) UNIQUE NOT NULL,
    departamento VARCHAR(50),
    cargo VARCHAR(50),
    email VARCHAR(100),
    telefono VARCHAR(15),
    foto_perfil VARCHAR(255),
    fecha_ingreso DATE,
    activo BOOLEAN DEFAULT 1,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    actualizado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- TABLA REGISTROS_ACCESO
CREATE TABLE registros_acceso (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    empleado_id INTEGER NOT NULL,
    fecha_hora TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    tipo_evento VARCHAR(10) CHECK(tipo_evento IN ('ENTRADA', 'SALIDA')),
    resultado VARCHAR(20) DEFAULT 'EXITOSO',
    metodo_acceso VARCHAR(20) DEFAULT 'QR',
    observaciones TEXT,
    FOREIGN KEY (empleado_id) REFERENCES empleados(id)
);

-- INDICES PARA MEJOR PERFORMANCE
CREATE INDEX idx_empleados_activo ON empleados(activo);
CREATE INDEX idx_empleados_qr ON empleados(codigo_qr);
CREATE INDEX idx_registros_fecha ON registros_acceso(fecha_hora);
CREATE INDEX idx_registros_empleado ON registros_acceso(empleado_id);
```

Listing 8: Esquema completo de la base de datos

4.3 Ejemplos de Código Clave

4.3.1 Clase Principal de la Aplicación

```
package com.controlacceso;

import com.controlacceso.vista.MainFrame;
import javax.swing.SwingUtilities;

public class MainApp {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            try {
                MainFrame frame = new MainFrame();
                frame.setVisible(true);
                System.out.println("Aplicacion iniciada correctamente");
```

```
            } catch (Exception e) {
                System.err.println("Error al iniciar la aplicacion: " + e.
                    getMessage());
                e.printStackTrace();
            }
        });
    }
}
```

Listing 9: Clase MainApp.java - Punto de entrada

4.3.2 Controlador de Códigos QR

```
package com.controlacceso.controlador;

import com.google.zxing.*;
import com.google.zxing.client.j2se.BufferedImageLuminanceSource;
import com.google.zxing.common.HybridBinarizer;
import java.awt.image.BufferedImage;
import java.util.Map;
import java.util.HashMap;

public class QRController {
    private final Map<DecodeHintType, Object> hints;

    public QRController() {
        hints = new HashMap<>();
        hints.put(DecodeHintType.TRY_HARDER, Boolean.TRUE);
        hints.put(DecodeHintType.CHARACTER_SET, "UTF-8");
    }

    public String decodeQRCode(BufferedImage image) throws
        NotFoundException {
        LuminanceSource source = new BufferedImageLuminanceSource(image);
        BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source)
            );

        MultiFormatReader reader = new MultiFormatReader();
        Result result = reader.decode(bitmap, hints);

        return result.getText();
    }

    public boolean isValidQRFormat(String qrData) {
        // Validar formato esperado: ID-EMP-0001
        return qrData != null && qrData.matches("ID-EMP-\\d{4}");
    }
}
```

Listing 10: Controlador para procesamiento de QR

5 Mantenimiento y Solución de Problemas

5.1 Preguntas Frecuentes (FAQ)

Problema	Solución
La cámara no se enciende	Verificar permisos del sistema operativo. Comprobar que no esté siendo usada por otra aplicación.
Error de conexión a BD	Verificar que el archivo SQLite exista y tenga permisos de escritura. Revisar la ruta en config.properties.
QR no es reconocido	Asegurar buena iluminación. Verificar que el QR tenga el formato correcto (ID-EMP-XXXX).
La aplicación no compila	Verificar que todas las dependencias en pom.xml estén correctas. Ejecutar `mvn clean install`.
No se muestran fotos de perfil	Verificar que las imágenes existan en `fotos_usuarios/` y tengan el nombre correcto.
Lentitud en la detección	Reducir la resolución de la cámara en config.properties. Cerrar otras aplicaciones que usen cámara.

Cuadro 2: Problemas comunes y sus soluciones

5.2 Pruebas Recomendadas

5.2.1 Pruebas Unitarias

```
package com.controlacceso.test;

import com.controlacceso.controlador.QRController;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class QRControllerTest {

    @Test
    public void testValidQRFormat() {
        QRController controller = new QRController();

        assertTrue(controller.isValidQRFormat("ID-EMP-0001"));
        assertTrue(controller.isValidQRFormat("ID-EMP-9999"));
        assertFalse(controller.isValidQRFormat("EMP-0001"));
        assertFalse(controller.isValidQRFormat("ID-EMP-ABCD"));
        assertFalse(controller.isValidQRFormat(null));
        assertFalse(controller.isValidQRFormat(""));
    }
}
```

Listing 11: Ejemplo de prueba unitaria para validacion de QR

5.2.2 Pruebas de Integración

- **Prueba de flujo completo:** Escanear QR válido y verificar registro en BD
- **Prueba de cámara:** Conectar y desconectar cámara durante ejecución
- **Prueba de base de datos:** Simular fallos de conexión y recuperación
- **Prueba de carga:** Múltiples accesos en corto tiempo

5.3 Procedimientos de Mantenimiento

5.3.1 Backup de Base de Datos

```
#!/bin/bash
# backup_database.sh
BACKUP_DIR="backups/"
DATE=$(date +"%Y%m%d_%H%M%S")
DB_FILE="database/control_acceso.db"
BACKUP_FILE="${BACKUP_DIR}backup_${DATE}.db"

mkdir -p $BACKUP_DIR
cp $DB_FILE $BACKUP_FILE
echo "Backup creado: $BACKUP_FILE"

# Mantener solo ultimos 7 dias de backups
find $BACKUP_DIR -name "*.db" -mtime +7 -delete
```

Listing 12: Script para backup automatico de SQLite

5.3.2 Actualización de Dependencias

1. Revisar nuevas versiones en `pom.xml`
2. Actualizar una dependencia a la vez
3. Ejecutar pruebas después de cada actualización
4. Documentar cambios en `CHANGELOG.md`

6 Extensión y Escalabilidad

6.1 Puntos de Extensión Identificados

1. **Integración con APIs REST:** Conectar con sistemas de RRHH
2. **Notificaciones en tiempo real:** Enviar emails o mensajes
3. **Panel de administración web:** Para gestión remota

4. **Reportes avanzados:** Gráficos y estadísticas
5. **Soporte multi-cámara:** Para varias entradas/salidas
6. **Reconocimiento facial:** Como método alternativo de autenticación

6.2 Contribución al Proyecto

- **Reportar issues:** Usar el sistema de GitHub Issues
- **Enviar pull requests:** Siguir las guías de contribución
- **Documentación:** Mejorar o traducir documentación
- **Pruebas:** Reportar bugs y sugerir casos de prueba

Conclusión

Este sistema de control de acceso representa una solución robusta, modular y escalable para la gestión de accesos en entornos empresariales. La arquitectura MVC bien definida, junto con el uso de tecnologías estables como Java y SQLite, asegura un mantenimiento sencillo y una alta disponibilidad.

El manual técnico presentado proporciona toda la información necesaria para entender, instalar, configurar, usar y extender el sistema. Se recomienda mantener esta documentación actualizada con cada nueva versión del software.

Fin del Manual Técnico

A Glosario de Términos

QR (Quick Response): Código de barras bidimensional que almacena información.

SQLite: Sistema de gestión de bases de datos relacional embebido.

MVC (Modelo-Vista-Controlador): Patrón de arquitectura de software.

ZXing: Biblioteca open-source para procesamiento de códigos de barras.

JDBC (Java Database Connectivity): API para conectividad con bases de datos.

DAO (Data Access Object): Patrón para abstraer el acceso a datos.

B Recursos Adicionales

- **Documentación oficial Java:** https://docs.oracle.com/javase/
- **SQLite Documentation:** https://www.sqlite.org/docs.html
- **ZXing GitHub:** https://github.com/zxing/zxing
- **Repositorio del proyecto:** https://github.com/Leo-Caz/Grupo1-control-acceso-qr-sqlite
- **Plantilla de código QR generador:** https://qr-code-generator.com/