

**FACULTAD DE INGENIERÍA**

Sistema de Gestión de Voluntarios para el Cuerpo de Bomberos de Lima Metropolitana

**CURSO:**

PROGRAMACIÓN ORIENTADA A OBJETOS II

**PROFESOR:**

ROBERTO RODRÍGUEZ URQUIAGA

**INTEGRANTES:**

CARDENAS TANTALEAN, Jhair Alexis - 100%

(Ingeniería de Software, 2311595)

SANCHEZ EQUIAPAZA, Axel Gael - 100%

(Ingeniería de Sistemas de Información, 2311409)

SONO ARCOS, Fabricio Jose - 100%

(Ingeniería Empresarial, 2313228)

**LIMA – PERÚ**

**2025**

TABLA DE CONTENIDOS

[**1. Introducción 3**](#_b4btpg5pewew)

[**2. Descripción del Caso 4**](#_sb91pjh20pnu)

[**3. Objetivos 5**](#_3u25hwkhlbmb)

[Objetivo General 5](#_pqykkfp882ll)

[Objetivos Específicos 5](#_x7d6qfll7gg0)

[**4. Metodología de Desarrollo 6**](#_r7jg3br72cie)

[Enfoque de Desarrollo 6](#_g5b9vbo54yny)

[Patrones de Diseño Implementados 6](#_i254xgx7cim)

[**Referencias Bibliográficas 7**](#_wq0z4w6tm8yw)

# 

# 1. Introducción

El Cuerpo General de Bomberos Voluntarios del Perú (CGBVP) cumple una labor esencial en la protección de vidas, bienes y el medio ambiente frente a emergencias y desastres. Sin embargo, gran parte de su gestión operativa y administrativa —en especial la relacionada con el registro, control y asignación de voluntarios— aún se realiza mediante procesos manuales, hojas de cálculo o archivos no centralizados. Esta situación limita la eficiencia institucional, genera duplicidad de información y dificulta la toma oportuna de decisiones ante emergencias.

En este contexto, la presente investigación propone el desarrollo de una Plataforma de Gestión y Monitoreo de Voluntarios, orientada a modernizar el registro y administración del personal operativo del CGBVP en Lima Metropolitana. El sistema permitirá gestionar información relevante como datos personales, especialidades, experiencia, disponibilidad y documentación de los bomberos, asegurando una administración más ordenada, transparente y accesible.

El desarrollo de la plataforma se llevó a cabo aplicando los principios de la Programación Orientada a Objetos (POO) en el lenguaje Java, utilizando el entorno de desarrollo Apache NetBeans IDE. Asimismo, se adoptó la arquitectura Modelo–Vista–Controlador (MVC) para garantizar una estructura modular y escalable, junto con la aplicación de los patrones de diseño Factory y Builder, que facilitaron la creación y organización de los objetos del sistema.

El proyecto tiene un enfoque académico y práctico, pues busca demostrar la capacidad de los estudiantes para aplicar los fundamentos de la ingeniería de software en la solución de problemas reales. Además, este trabajo se alinea con el propósito institucional de promover el desarrollo de soluciones tecnológicas innovadoras que fortalezcan la gestión pública y el servicio social. De este modo, la plataforma representa no solo un ejercicio técnico, sino también un aporte significativo hacia la digitalización de procesos en una institución clave para la sociedad peruana.

# 2. Descripción del Caso

El Cuerpo General de Bomberos Voluntarios del Perú (CGBVP) es una institución que presta servicios esenciales en la atención de incendios, rescates y emergencias médicas. Sin embargo, una parte significativa de su gestión interna, especialmente la relacionada con el registro y control de los voluntarios, continúa realizándose de forma manual, mediante planillas físicas o archivos dispersos en hojas de cálculo. Este método tradicional dificulta la actualización oportuna de datos, la identificación del personal disponible y la correcta asignación de recursos humanos durante las emergencias.

Ante esta problemática, se plantea el desarrollo de una Plataforma de Gestión y Monitoreo de Voluntarios, cuyo propósito es digitalizar y centralizar la información del personal operativo. A través del sistema, será posible registrar datos personales, habilidades, experiencia, disponibilidad, y documentación de los bomberos, además de facilitar la búsqueda, modificación y actualización de registros en tiempo real.

El sistema se implementa en Java, dentro del entorno de desarrollo Apache NetBeans IDE, aplicando los principios de la Programación Orientada a Objetos (POO) y la arquitectura Modelo–Vista–Controlador (MVC). Esta estructura permitió dividir el proyecto en componentes independientes y facilitar su mantenimiento y escalabilidad. Asimismo, se integraron los patrones de diseño Factory y Builder, que ayudaron a gestionar la creación de objetos de manera flexible y ordenada.

El proyecto responde a una necesidad real de optimización tecnológica dentro del CGBVP, ofreciendo una solución funcional que contribuye a mejorar la eficiencia y transparencia de los procesos administrativos. Si bien el sistema tiene carácter académico y se presenta como un prototipo, su estructura modular y su lógica de desarrollo permiten adaptarlo a futuras versiones institucionales con funcionalidades más avanzadas, como monitoreo en tiempo real o integración con bases de datos externas.

# 3. Objetivos

#### **Objetivo General:**

Desarrollar una plataforma para la gestión y monitoreo de voluntarios del Cuerpo General de Bomberos Voluntarios del Perú – Lima Metropolitana, aplicando los principios de la Programación Orientada a Objetos (POO), la arquitectura Modelo–Vista–Controlador (MVC) y los patrones de diseño Factory y Builder, con el fin de lograr un sistema modular, ordenado y funcional que optimice el registro y la organización del personal voluntario.

**Objetivos Específicos:**

1. Diseñar la estructura lógica y funcional del sistema aplicando la arquitectura MVC, separando las responsabilidades entre la gestión de datos, el control de operaciones y la interfaz de usuario.
2. Implementar un módulo de registro y administración de voluntarios, que permite agregar, editar, eliminar y consultar información básica del personal mediante una interfaz gráfica desarrollada en Java Swing.
3. Aplicar los patrones de diseño Factory y Builder para la creación y configuración flexible de los objetos del sistema, garantizando un código reutilizable y mantenible.
4. Integrar mecanismos de prueba y validación del código, empleando JUnit para verificar el funcionamiento de las clases principales y GitHub como herramienta de control de versiones y documentación del avance del proyecto.
5. Simular escenarios de gestión de voluntarios para comprobar la funcionalidad del sistema, priorizando la facilidad de uso, la claridad visual y la estabilidad del prototipo durante su ejecución en NetBeans.

# 4. Desarrollo

**4.1 Requerimientos**

#### **Requerimientos Funcionales:**

1. Registro de voluntarios:  
    Permitir el registro digital de los bomberos voluntarios con información básica como nombre, especialidad, turno y disponibilidad, almacenando los datos de forma ordenada y accesible.
2. Consulta y búsqueda de registros:  
    Facilitar la visualización y búsqueda de los voluntarios registrados, filtrando la información por nombre, especialidad o turno.
3. Actualización y eliminación de datos:  
    Brindar la posibilidad de modificar o eliminar registros existentes, asegurando que la información se mantenga actualizada y sin duplicados.
4. Gestión de estaciones o compañías:  
    Permitir registrar y consultar las compañías o estaciones donde se encuentra asignado cada voluntario.
5. Visualización general del personal:  
    Mostrar una lista ordenada con todos los voluntarios registrados, indicando su estado de disponibilidad, turno y especialidad.
6. Generación de reportes básicos:  
    Emitir un resumen general del personal activo o total registrado, mostrando el número de voluntarios por estación o especialidad.

**Requerimientos No Funcionales:**

1. **Rendimiento:** El sistema debe permitir realizar las operaciones principales (registro, búsqueda, actualización y eliminación) de manera rápida y fluida en un entorno local.
2. **Usabilidad:** La interfaz debe ser intuitiva, clara y sencilla, diseñada con componentes de Java Swing para facilitar el uso incluso a usuarios con poca experiencia en software.
3. **Mantenibilidad:** El código debe estar estructurado siguiendo los principios de la Programación Orientada a Objetos (POO) y buenas prácticas de diseño como MVC, Factory y Builder, lo que permitirá futuras mejoras.
4. **Compatibilidad:** El sistema debe ser ejecutable en diferentes sistemas operativos compatibles con la Máquina Virtual de Java (JVM), como Windows, Linux o macOS.
5. **Seguridad básica:** La aplicación debe validar la información ingresada por el usuario para evitar errores o datos inconsistentes en los formularios.
6. **Escalabilidad académica:** La arquitectura debe permitir la incorporación futura de nuevos módulos, como gestión de roles o control de asistencia, sin alterar la estructura principal.

**4.2 Análisis de Hardware y Software**

#### **Hardware:**

El sistema está diseñado para funcionar en equipos personales de escritorio o portátiles utilizados en entornos académicos o institucionales, sin requerir infraestructura de red avanzada.

**Requerimientos mínimos recomendados:**

**Equipo local (usuario):**

* Procesador: Intel Core i3 o equivalente
* Memoria RAM: 4 GB mínimo
* Almacenamiento: 256 GB (HDD o SSD)
* Monitor: Resolución 1366×768 o superior
* Periféricos: Teclado, mouse y conexión a internet opcional (para control de versiones)

**Software**

* **Sistema operativo:** Windows 10/11, Ubuntu Linux o macOS (cualquier sistema compatible con la JVM).
* **Lenguaje de programación:** Java SE 17 o superior, por su portabilidad, robustez y compatibilidad con herramientas educativas.
* **Entorno de desarrollo (IDE):** Apache NetBeans, debido a su integración con Java Swing, soporte para pruebas unitarias y facilidad en la gestión de proyectos MVC.
* **Frameworks y librerías:**
  + **Swing:** para la construcción de la interfaz gráfica del usuario.
  + **Java Collections Framework:** para la gestión eficiente de listas y estructuras de datos.
  + **JUnit:** para pruebas unitarias y verificación del funcionamiento de las clases principales.
  + **Git y GitHub:** para control de versiones y respaldo del código fuente.

**Justificación:**

El uso de Java SE y NetBeans garantiza una solución multiplataforma y didáctica, ideal para un entorno académico. La aplicación de la arquitectura MVC permite separar la lógica de negocio, los datos y la interfaz de usuario, facilitando la comprensión y mantenimiento del código. Asimismo, el uso de patrones de diseño como Factory y Builder refuerza la creación estructurada de objetos, demostrando el dominio de los principios de la Programación Orientada a Objetos II. El sistema, al ser un prototipo educativo, prioriza la claridad, la modularidad y la correcta aplicación de conceptos técnicos sobre la complejidad funcional, cumpliendo plenamente los objetivos del curso.

# Referencias Bibliográficas

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Oracle Corporation. (2023). *NetBeans IDE Documentation*. Recuperado de<https://netbeans.apache.org/docs/>

Martin, R. C. (2000). *Design Principles and Design Patterns*. Robert Martin Media.

Laudon, K. C., & Laudon, J. P. (2020). *Management Information Systems: Managing the Digital Firm* (16th ed.). Pearson.

Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.

Enlace GitHub:<https://github.com/daJackLy/CruzRojaSystem>