TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

DEPARTAMENTO DE INFORMÁTICA

EZJAVACODE

MANUAL TÉCNICO

RUBÉN MATAMOROS TRIGO

2024/2025

Índice

1. Introducción

EZJavaCode es una aplicación de escritorio desarrollada con el propósito de asistir a estudiantes y programadores principiantes en la generación estructurada de código en lenguaje **Java**. La herramienta permite crear clases, atributos y métodos de manera visual e intuitiva, reduciendo la necesidad de escribir manualmente código repetitivo y minimizando los errores sintácticos. Gracias a su enfoque gráfico, la aplicación ofrece una experiencia didáctica accesible, facilitando el aprendizaje de los fundamentos de la programación orientada a objetos.

El proyecto ha sido concebido con una arquitectura modular que separa claramente la lógica de negocio (back-end) de la interfaz de usuario (front-end). Para el diseño visual se ha empleado **JavaFX**, junto con hojas de estilo CSS personalizadas, lo que ha permitido construir una interfaz moderna, clara y funcional. Además, se ha utilizado **draw.io** durante la fase de diseño para estructurar los wireframes de la interfaz y planificar la distribución de componentes.

La lógica interna de la aplicación ha sido desarrollada en **Java**, utilizando **JDK 21** y el entorno de desarrollo **IntelliJ IDEA**. El back-end gestiona la representación y manipulación de clases, métodos y atributos, permitiendo la generación de archivos `.java` listos para su uso. También se ha empleado **Visual Studio** de forma complementaria para la edición de recursos y archivos auxiliares.

Este manual técnico documenta detalladamente los aspectos clave del desarrollo de **EZJavaCode:** la arquitectura de la aplicación, los entornos de desarrollo utilizados, las tecnologías aplicadas, las pruebas realizadas, así como una propuesta de mejoras futuras. Su objetivo es servir como guía para la comprensión, mantenimiento y ampliación del proyecto, ofreciendo una visión completa del funcionamiento interno de la herramienta.

2. Arquitectura de la aplicación

2.1. Frontend

2.1.1. Tecnologías usadas

Para el desarrollo del Frontend de **EZJavaCode** se ha empleado **JavaFX**, un framework de Java especialmente diseñado para la creación de interfaces gráficas de usuario (GUI) modernas, interactivas y multiplataforma. JavaFX permite el desarrollo de aplicaciones con una experiencia visual avanzada, integrando una amplia variedad de controles y layouts como botones, etiquetas, tablas, cuadros de texto, menús y paneles de organización, facilitando la construcción de vistas complejas y personalizables.

El Frontend está estructurado en el paquete Visual, donde se definen todas las vistas y componentes gráficos que interactúan directamente con el usuario. Entre los componentes más destacados se encuentran formularios para la creación y edición de clases, métodos y atributos, así como menús y paneles de navegación que mejoran la usabilidad y la experiencia de usuario.

Para la personalización visual, se hace uso de hojas de estilos **CSS** específicas, aplicadas sobre los componentes **JavaFX**. Esto permite adaptar la apariencia de la aplicación a una estética moderna y profesional, ajustando colores, fuentes, tamaños y efectos visuales para una mejor presentación.

Las tecnologías y librerías empleadas en el **Frontend** son:

JavaFX: Incluye módulos como javafx.application, javafx.stage, javafx.scene, javafx.scene.control, javafx.scene.layout, javafx.scene.text, javafx.collections y javafx.geometry, que permiten la creación y gestión de todos los elementos gráficos y su disposición en pantalla.

CSS personalizado: Utilizado para definir estilos visuales avanzados y coherentes en toda la interfaz gráfica.

Las versiones empleadas en el Frontend son:

JavaFX: Compatible con Java 21 (la versión exacta depende del entorno de instalación).
CSS: Personalizado para la aplicación.

2.1.2. Entorno de desarrollo

El desarrollo del **front-end de EZJavaCode** se ha llevado a cabo utilizando herramientas enfocadas al diseño visual y a la implementación de interfaces gráficas de usuario en aplicaciones de escritorio. A continuación se detallan los recursos empleados para esta parte del proyecto:

JavaFX

La interfaz gráfica de **EZJavaCode** ha sido desarrollada utilizando **JavaFX,** un framework moderno para la construcción de interfaces de usuario en Java. JavaFX permite crear componentes visuales interactivos y personalizables, facilitando el desarrollo de un entorno intuitivo para el usuario.
Características de JavaFX utilizadas en el proyecto:

- Creación de ventanas, menús, botones, formularios y tarjetas visuales.
- Uso de layouts dinámicos como VBox, HBox y GridPane para organizar los elementos.
- Aplicación de estilos mediante CSS personalizado para mejorar la apariencia de la interfaz.
- Gestión de eventos y acciones asociadas a los elementos gráficos.

Draw.io

Para el diseño preliminar de la interfaz y la planificación del layout visual, se ha utilizado draw.io (también conocido como diagrams.net), una herramienta gratuita y online para la creación de diagramas y wireframes.
Usos de draw.io en el proyecto:

- Creación de wireframes que representen la estructura visual de la aplicación antes de su implementación.
- Planificación de la distribución de componentes y navegación entre pantallas.
- Apoyo en la toma de decisiones de diseño y usabilidad.

Gracias al uso combinado de estas herramientas, ha sido posible construir una interfaz coherente, funcional y alineada con los objetivos del proyecto, facilitando al usuario la generación de código Java de manera visual y estructurada.

2.2. Backend

2.2.1. Tecnologías usadas

El Backend de **EZJavaCode** está desarrollado íntegramente en Java, utilizando la versión 21 del **JDK,** lo que permite aprovechar las últimas características y mejoras del lenguaje. El Backend se encarga de la lógica de negocio de la aplicación, gestionando la creación, edición y almacenamiento de clases, métodos y atributos, así como la comunicación entre los distintos módulos internos.

La lógica Backend está organizada en el paquete Funcional, que contiene clases dedicadas a la representación y manipulación de estructuras de código Java, como clases, atributos y métodos. Estas clases permiten generar, modificar y exportar código fuente de manera dinámica, facilitando el trabajo del usuario en la creación de programas Java.

Para el procesamiento y almacenamiento de datos, así como para la gestión de operaciones internas, se utilizan las siguientes librerías estándar de Java:

java.util: Proporciona colecciones, listas, mapas y utilidades generales para la manipulación eficiente de datos.

java.io: Permite la lectura y escritura de archivos, fundamental para la exportación y almacenamiento de código generado.

java.lang: Incluye clases base del lenguaje, necesarias para el funcionamiento general del programa.

j**ava.text:** Facilita el formateo de textos y fechas, útil para la presentación y registro de información.

java.time: Gestiona fechas y horas, permitiendo el control de eventos y registros temporales.

java.math: Ofrece operaciones matemáticas avanzadas, en caso de ser necesarias para la lógica de negocio.

2.2.2. Entorno de desarrollo

El desarrollo del **back-end de EZJavaCode** se ha centrado en la lógica de generación de código Java a partir de estructuras definidas por el usuario. Para ello, se ha utilizado un entorno de desarrollo robusto y actual, junto con herramientas auxiliares para mejorar la productividad y la organización del proyecto.

IntelliJ IDEA

El entorno principal para el desarrollo del back-end ha sido IntelliJ IDEA, una de las herramientas más completas para el desarrollo en Java. Se ha trabajado con la versión compatible con **JDK 21**, lo cual ha permitido aprovechar las nuevas funcionalidades del lenguaje, así como una mayor seguridad y rendimiento.
Usos de IntelliJ IDEA en el proyecto:

- Implementación de la lógica del programa, incluyendo la creación dinámica de clases, atributos y métodos.
- Gestión del proyecto mediante módulos organizados y estructurados.
- Herramientas avanzadas de depuración, análisis estático de código y refactorización.
- Posibilidad de ejecutar pruebas locales y simular flujos funcionales.
  Java y JDK 21

Ventajas de usar JDK 21:

- Acceso a nuevas características del lenguaje como patrones de switch, mejoras en el rendimiento de colecciones, y mayor estabilidad.
  Compatibilidad total con JavaFX, utilizado en el front-end.
- Capacidad para generar archivos .java válidos desde estructuras internas del programa

Visual Studio (uso complementario)

- Edición y validación de archivos auxiliares del sistema (como configuraciones, recursos, archivos de texto…).
- Visualización rápida de estructuras de carpetas, especialmente en sistemas de archivos complejos.
- Revisión puntual del código fuente con extensiones para Java.

Gracias a este conjunto de herramientas, el desarrollo del back-end de EZJavaCode ha sido eficiente, mantenible y fácilmente escalable, permitiendo una integración fluida con el front-end y una generación automática de código Java clara y funcional

3. Documentación técnica

3.1. Análisis

Wireframe

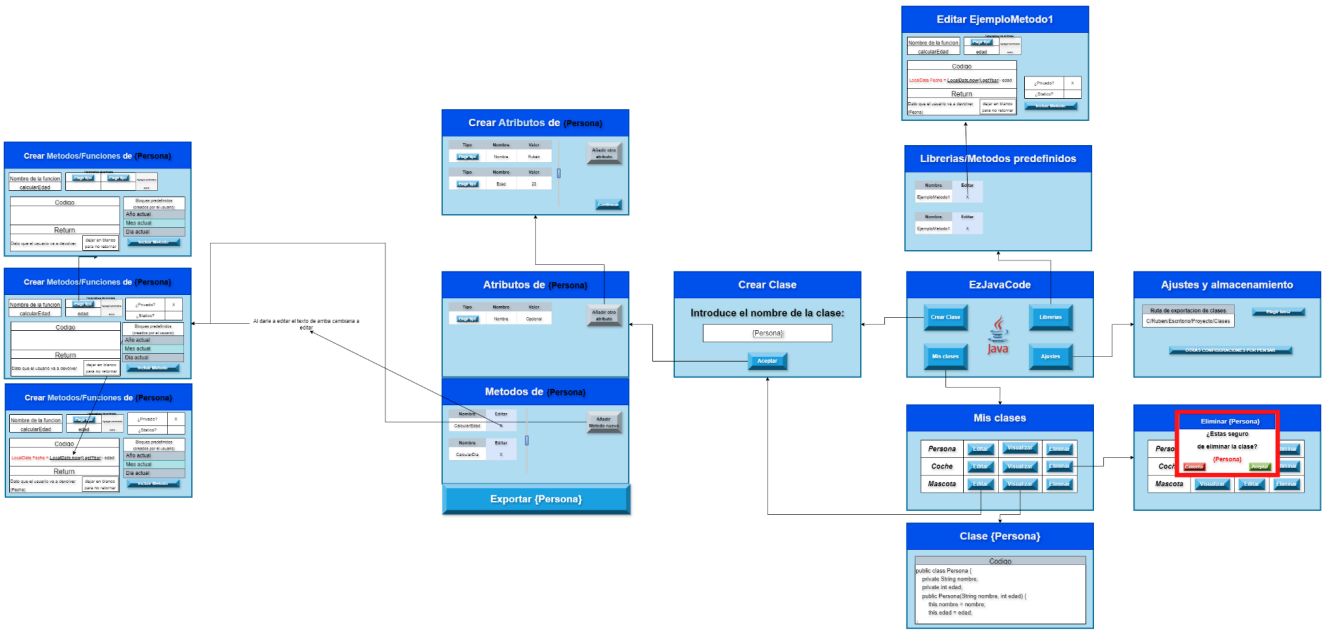


Diagrama de clases

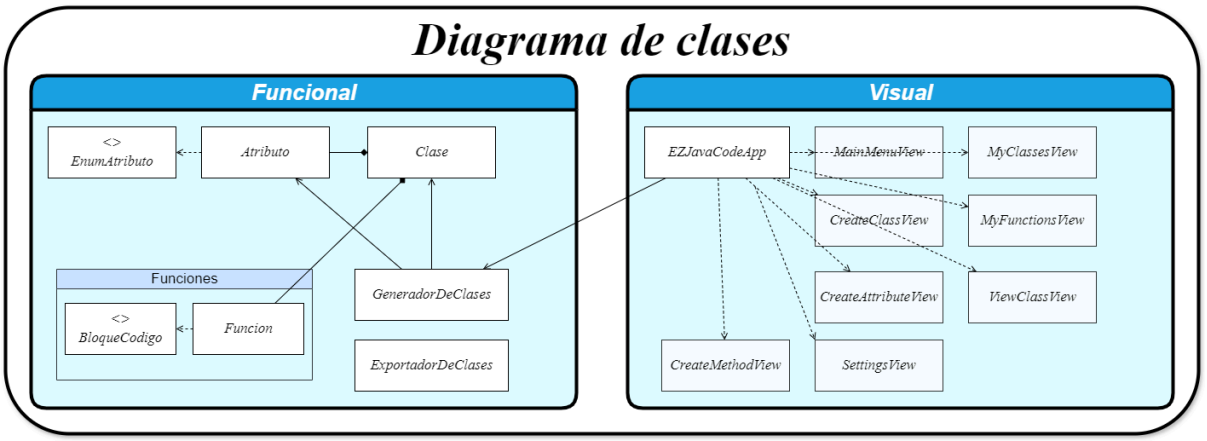
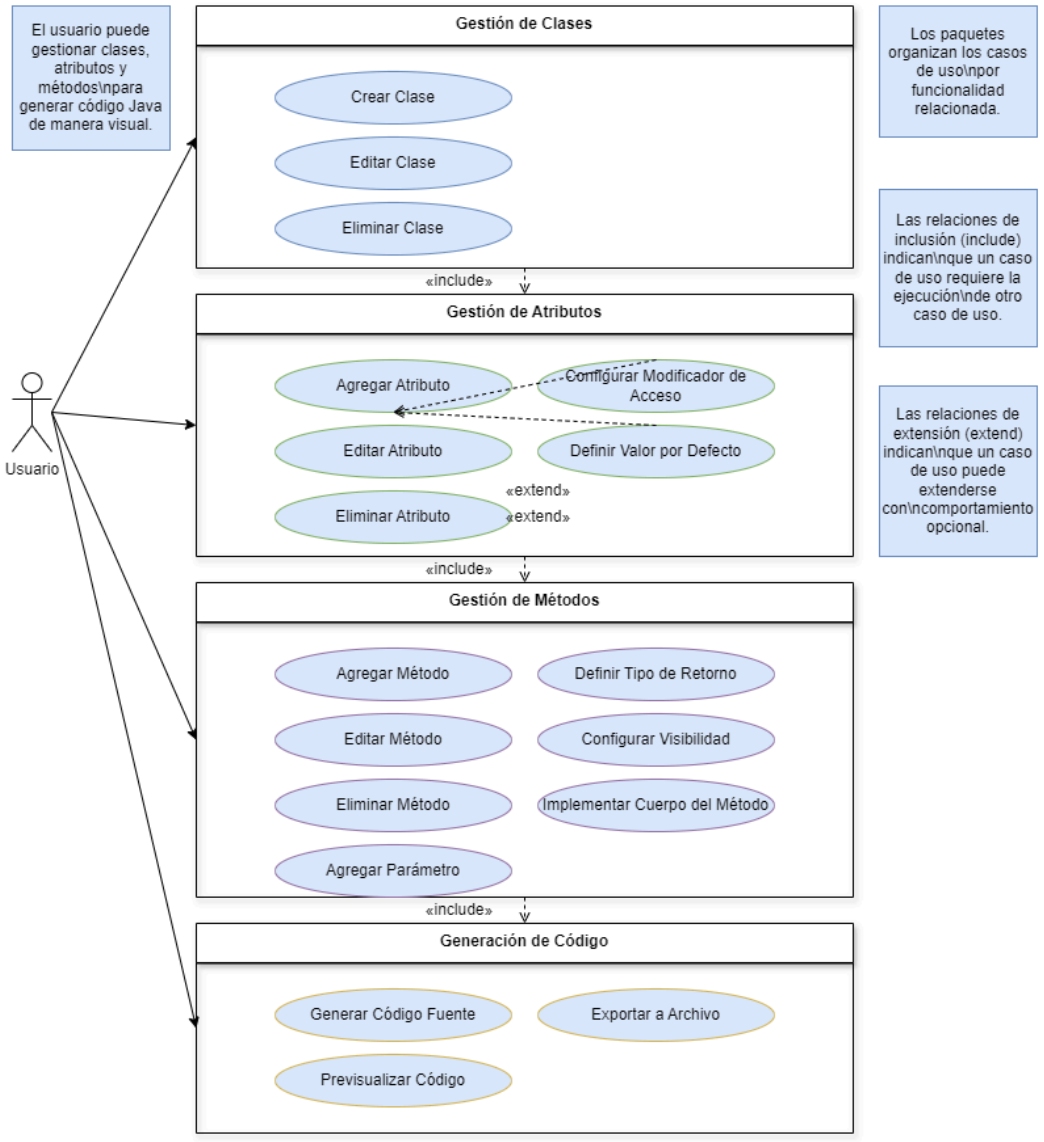


Diagrama de clases completo

3.2. Desarrollo

Diagrama de casos de uso



3.3. Pruebas realizadas

Durante el desarrollo de **EZJavaCode**, se han llevado a cabo distintas pruebas centradas en asegurar el correcto funcionamiento de cada funcionalidad y en evaluar la experiencia de usuario. Estas pruebas han sido principalmente de tipo funcional e informal, realizadas de manera continua a medida que se añadían nuevas características al sistema.

Pruebas funcionales incrementales

A medida que se desarrollaban las distintas funcionalidades del programa —como la creación de clases, atributos o métodos— se realizaron pruebas manuales inmediatas para comprobar que cada elemento funcionaba como se esperaba. Este enfoque permitió identificar y corregir errores de forma progresiva y eficiente.

Entre los aspectos verificados se encuentran:
- La correcta creación y visualización de bloques (clase, atributo, método).
- La interacción entre distintos componentes de la interfaz.
- La validación de datos introducidos por el usuario.

Pruebas con usuarios

Se invitó a varios usuarios a probar la aplicación de manera informal, con el objetivo de obtener una perspectiva externa sobre el funcionamiento del sistema. Estas pruebas permitieron identificar errores que no habían sido detectados durante las pruebas internas, validar la lógica de interacción de la interfaz desde el punto de vista del usuario final y recoger sugerencias sobre la disposición de elementos y la navegación.

Gracias a este testeo colaborativo, se ajustaron ciertos aspectos visuales y funcionales para mejorar la experiencia general.

Evaluación de usabilidad y accesibilidad

Durante las pruebas con usuarios también se prestó atención a aspectos de usabilidad y accesibilidad, con el fin de garantizar que la herramienta sea comprensible y utilizable por distintos perfiles. Se evaluaron elementos como:
- Claridad y legibilidad de los textos e iconos.
- Distribución intuitiva de los menús y botones.
- Facilidad para completar acciones sin instrucciones externas.
- Accesibilidad básica para personas con visión reducida (contraste, tamaño de fuente, disposición).

4. Proceso de despliegue

Instrucciones de instalación y ejecución

Forma 1:

1. Instala Java JDK 21
   Descarga e instala desde:
   https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe

2. Ejecuta EZJavaCode
   Haz doble clic en el archivo llamado `EZJavaCodeEjecutable`

- No es necesario instalar nada más, todo lo necesario está incluido en la carpeta del programa.

Forma 2:

Si surge algún problema al ejecutar el .jar con tu dispositivo recomiendo utilizar algún IDE como IntelliJ
y darle a ejecutar al archivo llamado

Instala Java JDK 21
Descarga e instala desde:
https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe

EZJavaCodeAppLauncher.java

Esta es la ruta donde esta ese archivo
ezjavacode\codigo\ezjavacode\src\Visual\EZJavaCodeAppLauncher.java

5. Propuesta de mejoras

1. Validación y compilación del código generado
Incorporar un sistema de validación que permita comprobar la sintaxis del código Java exportado directamente desde la aplicación. Además, podría integrarse una función para compilar y ejecutar pruebas básicas sin necesidad de un entorno externo.

2. Soporte para herencia y relaciones entre clases
Ampliar la lógica de creación de clases para permitir la definición de relaciones como herencia, composición o agregación, haciendo posible la generación de

estructuras de código más complejas y representativas de la programación orientada a objetos.

3. Personalización avanzada de métodos y atributos

Permitir al usuario definir modificadores de acceso, tipos de retorno personalizados, parámetros múltiples y anotaciones en los métodos. También se podría ofrecer un editor avanzado para insertar código adicional dentro del cuerpo de métodos.

4. Sistema de guardado y carga de proyectos

Incorporar una funcionalidad que permita guardar el estado actual del proyecto (clases, atributos y métodos definidos) y cargarlo posteriormente para continuar con la edición, facilitando el trabajo en sesiones prolongadas.

5. Mejora de la interfaz gráfica

Refinar el diseño visual de la interfaz con componentes más modernos, incorporación de temas (claro/oscuro), animaciones suaves y mejores indicadores visuales para errores o campos incompletos.

6. Internacionalización (i18n)

Agregar soporte multilingüe para que la aplicación sea accesible a usuarios de distintos idiomas, especialmente considerando su potencial uso educativo.

7. Integración con Git o entornos externos

Posibilitar la integración con repositorios Git o con entornos de desarrollo externos, permitiendo guardar directamente los archivos generados en un repositorio o abrirlos en otras herramientas desde la interfaz.

6. Bibliografía

Java Development Kit (JDK) 21

Oracle. *Java SE Development Kit 21 Documentation*. https://docs.oracle.com/en/java/javase/21/

Utilizado como base para el desarrollo del back-end, incluyendo las funcionalidades del lenguaje y el entorno de ejecución.

JavaFX

OpenJFX. *JavaFX Documentation*. https://openjfx.io/

Empleado para la construcción de la interfaz gráfica de usuario del proyecto.

draw.io / diagrams.net

JGraph Ltd. *Draw.io – Diagramming tool*. https://www.diagrams.net/

Utilizado para la creación de wireframes y el diseño visual previo a la implementación del front-end.

IntelliJ IDEA

JetBrains. *IntelliJ IDEA Documentation*. https://www.jetbrains.com/idea/

Entorno de desarrollo principal utilizado para la implementación de la lógica del sistema en Java.

Visual Studio

Microsoft. *Visual Studio IDE*. https://visualstudio.microsoft.com/

Utilizado de manera complementaria para la edición de archivos auxiliares y la organización de recursos del proyecto.