Universidad Tecnológica de Puebla

Portada de

PRODUCTO 2
ARQUITECTURA DE SOFTWARE

Presentado por: Dominguez Aldana Jonathan Jesus
Nombre del Profesor:JAVIER NOLASCO HERNANDEZ
Grupo:7:D Vespertino

ÍNDICE

ÍNDICE **2**
Introducción **3**
Producto 3:Implementación de Patrón Factory Method en Prototipo de Notificaciones **4**
Justificación de Patrones de Diseño Seleccionados **4**
Patrón Factory Method **4**
Diagrama UML de Clases - Patrón Factory Method **6**
Diagrama de Secuencia - Flujo con Factory Method **7**
Descripción General **9**
Arquitectura Factory Method Implementada **9**
Funcionalidades Principales **9**
Muestra Grafica del Funcionamiento **11**
El proyecto implementó con éxito una Calculadora de Porcentajes que utiliza el patrón Factory Method para gestionar tres tipos de visualizaciones: gráfico de barras, gráfico de pastel y tabla de resultados. Esta arquitectura permitió una clara separación entre la lógica de creación de objetos, los cálculos matemáticos y la presentación visual. **12**
La implementación del Factory Method facilitó la creación flexible de diferentes visualizaciones sin acoplar el código a clases concretas. Los diagramas UML desarrollados (clases, secuencia, componentes y flujo de vistas) demostraron la solidez de la arquitectura y la claridad en las interacciones entre los componentes. **12**
Las principales ventajas obtenidas incluyen una alta extensibilidad para agregar nuevas visualizaciones, un código mantenible gracias al desacoplamiento, y la capacidad de cambiar implementaciones sin afectar al sistema existente. La aplicación final ofrece tres vistas completamente funcionales que responden dinámicamente a los cambios en los datos de entrada. **12**
El patrón Factory Method resultó ser la elección adecuada para este proyecto, proporcionando una base sólida para futuras expansiones mientras se mantiene la coherencia arquitectónica y la facilidad de mantenimiento. **12**

Introducción

El presente proyecto desarrolla una calculadora interactiva de porcentajes implementada bajo el patrón arquitectónico Modelo–Vista–Controlador (MVC). La propuesta evidencia cómo la aplicación adecuada de principios de diseño permite transformar una solución aparentemente sencilla en un sistema estructurado, eficiente y escalable.

La aplicación web permite al usuario ingresar tres valores numéricos y visualizar sus porcentajes a través de tres modalidades de presentación: una vista tabular, un gráfico de barras y un gráfico circular. Una característica sobresaliente de esta implementación es la capacidad de alternar dinámicamente entre las distintas representaciones visuales sin comprometer la lógica central del sistema.

La arquitectura MVC adoptada asegura una separación clara de responsabilidades:

El Modelo se encarga del procesamiento matemático y la gestión de los datos.

Las Vistas administran la representación visual de la información mediante diferentes enfoques gráficos.

El Controlador actúa como enlace, coordinando la interacción entre el Modelo y las Vistas.

Esta división estructural facilita el mantenimiento, fomenta la escalabilidad del proyecto y favorece la reutilización del modelo de datos en diversas interfaces sin requerir modificaciones en la lógica interna.

Desarrollado en TypeScript empleando tecnologías como HTML5 y Canvas, este proyecto constituye un ejemplo práctico de cómo los patrones de diseño aplicados de manera correcta contribuyen a mejorar la calidad del código, su capacidad de prueba y la flexibilidad para futuras extensiones o integraciones.

Producto 3:Implementación de Patrón Factory Method en Prototipo de Notificaciones

Justificación de Patrones de Diseño Seleccionados

Patrón Factory Method

Justificación de la Selección:

El patrón Factory Method fue seleccionado por las siguientes razones:

Responsabilidades:

La fábrica centraliza la creación de notificaciones sin que el código principal conozca las clases concretas.

El cliente únicamente solicita el tipo de notificación; la fábrica decide qué clase devolver.

Se evita que la lógica de negocio dependa de implementaciones específicas.

Mantenibilidad y Escalabilidad:

Permite agregar nuevos tipos de notificaciones (Email, SMS, Push, WhatsApp) sin modificar la lógica ya existente.

Facilita la evolución del sistema al reducir el acoplamiento entre módulos.

Las clases concretas pueden crecer de forma independiente sin afectar al resto del sistema.

Código:

La lógica de creación se implementa una sola vez dentro de la fábrica.

El mismo método creador puede ser usado en múltiples partes del sistema.

Las notificaciones comparten una interfaz común, lo que permite intercambiarlas libremente.

Pruebas Unitarias:

Cada tipo de notificación puede ser probado de forma aislada.

Se pueden verificar las instancias generadas por la fábrica asegurando que correspondan al tipo solicitado.

El desacoplamiento facilita el uso de mocks o simulaciones en pruebas.

Arquitectura Flexible:

Permite manejar múltiples tipos de notificaciones desde un único punto de creación.

La aplicación puede cambiar dinámicamente el tipo de notificación según la necesidad del usuario.

El sistema puede ampliarse agregando nuevas clases concretas sin alterar la estructura original.

Justificación:

Aunque solo se aplica un patrón, Factory Method proporciona una arquitectura profesional al permitir:

Crear objetos especializados sin modificar el código del cliente.

Mantener un sistema modular, organizando mejor la lógica.

Diagrama UML de Clases - Patrón Factory Method

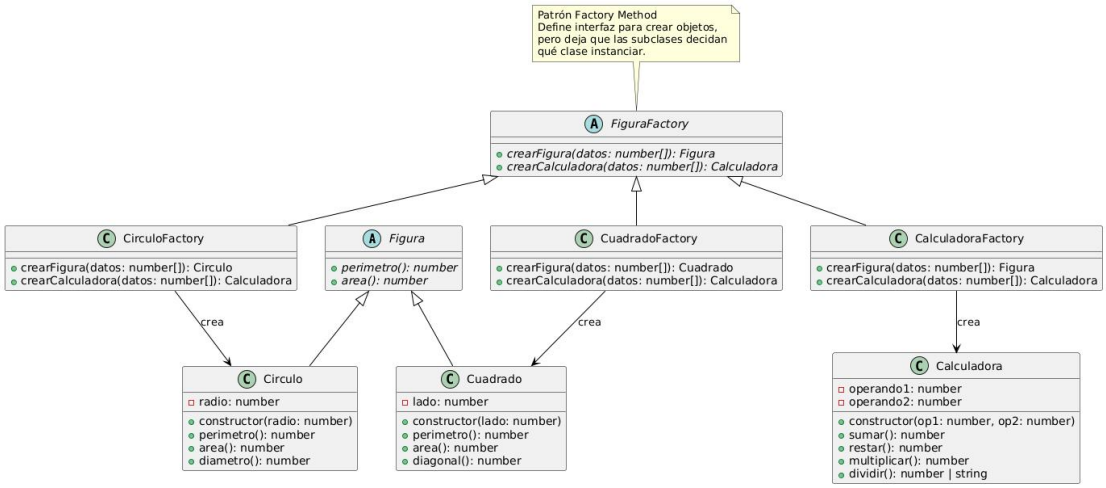


Diagrama de Secuencia - Flujo con Factory Method

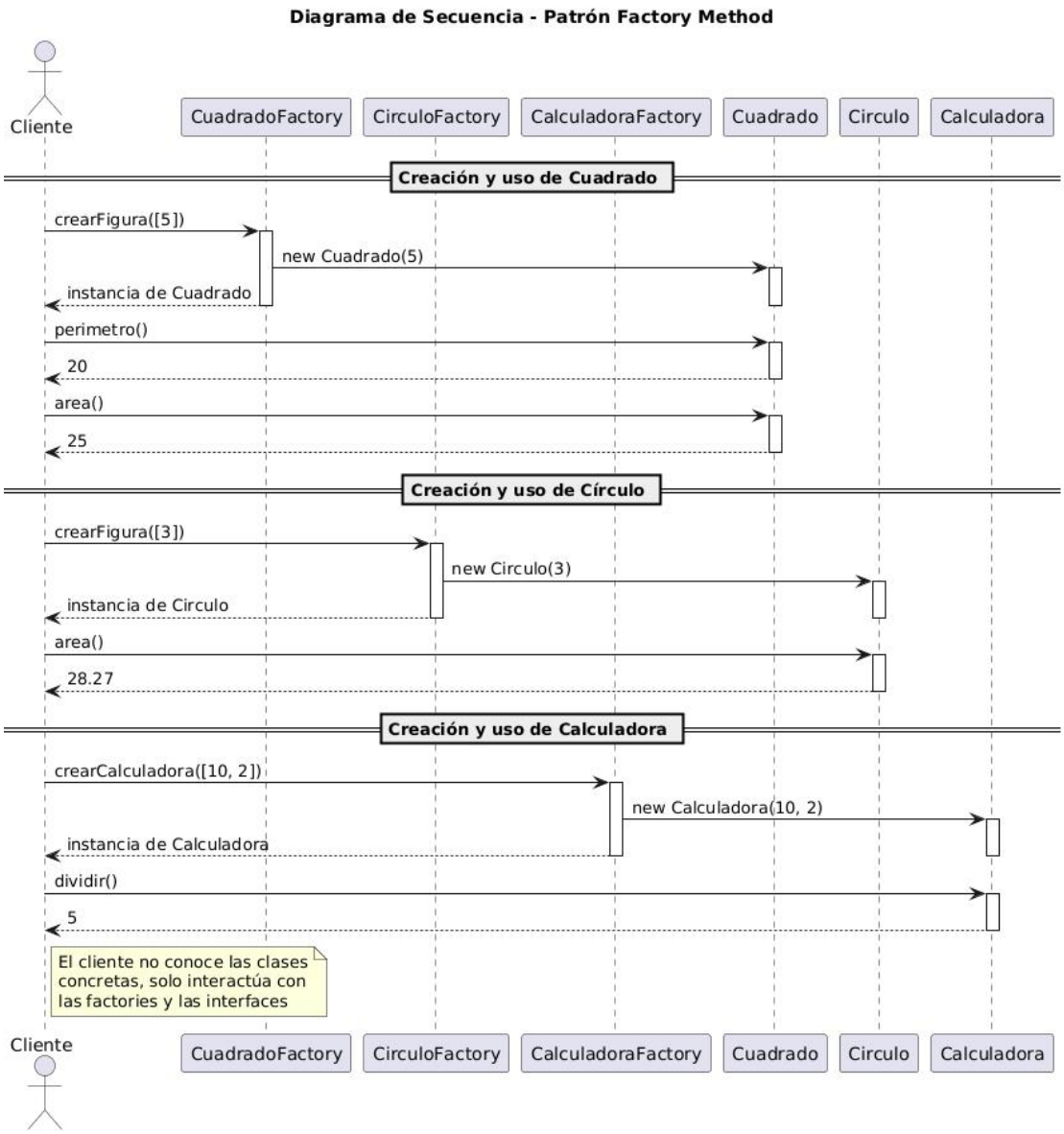


Diagrama de Componentes

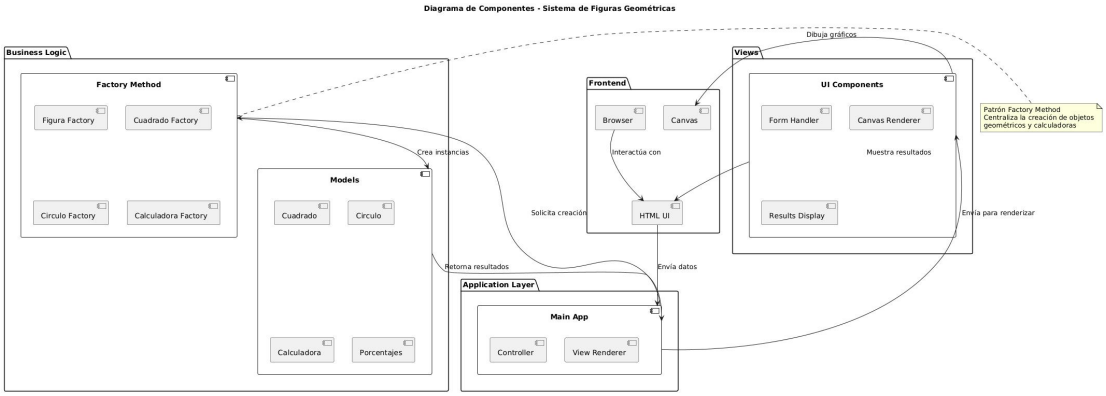


Diagrama de Flujo de Selección de Vistas

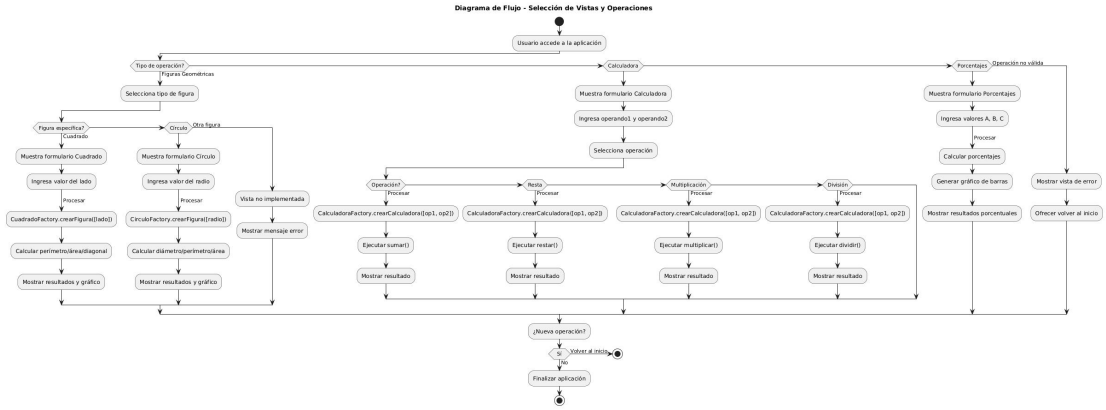


Diagrama de Flujo de Navegación entre Vistas

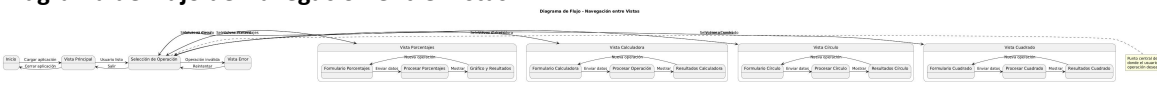
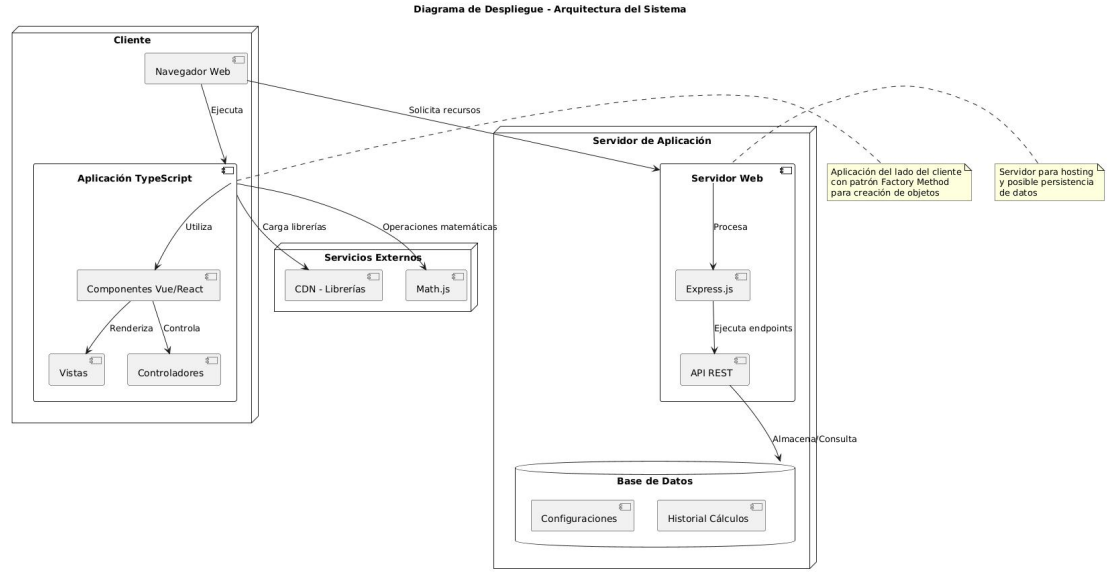


Diagrama de Despliegue



Descripción General

Este prototipo es una aplicación web que implementa el patrón de diseño Factory Method para crear diferentes figuras geométricas y realizar operaciones matemáticas. La aplicación permite a los usuarios seleccionar entre múltiples tipos de cálculos y visualizar los resultados de manera dinámica mediante diferentes representaciones gráficas y numéricas.

Arquitectura Factory Method Implementada

El sistema está estructurado bajo el patrón Factory Method, que centraliza y encapsula la creación de objetos en factories especializados. La arquitectura separa claramente la creación de objetos de su uso, permitiendo una extensibilidad máxima del sistema.

La capa de Abstract Factory define la interfaz FiguraFactory con métodos para crear figuras geométricas y calculadoras. Las Concrete Factories (CuadradoFactory, CirculoFactory, CalculadoraFactory) implementan estos métodos y deciden qué objetos concretos instanciar basándose en los parámetros proporcionados.

Los Productos consisten en tres categorías principales: Cuadrado y Circulo que representan figuras geométricas con métodos para calcular perímetro, área y propiedades específicas; Calculadora que realiza operaciones aritméticas básicas; y Porcentajes que calcula distribuciones porcentuales. Todos estos productos implementan interfaces cohesivas que garantizan un comportamiento predecible.

Funcionalidades Principales

Esta aplicación es una herramienta interactiva diseñada para que los usuarios puedan ingresar y luego visualizar un conjunto de tres números de diversas maneras.

Entrada de Datos: Tienes un área de formulario donde debes introducir tres valores numéricos que llamaremos A, B y C. La aplicación se encarga de revisar que estos datos sean válidos.

Selector de Visualización: Hay un menú desplegable (dropdown) que actúa como un interruptor de vistas. Esto te permite cambiar instantáneamente la forma en que ves los datos sin tener que recargar nada.

Opciones de Presentación: Los datos (A, B y C) y su proporción porcentual se pueden mostrar en tres formatos distintos:

Vista de Tabla (Estructurada): Muestra la información de forma ordenada en filas y columnas. Es ideal para ver los valores originales y sus porcentajes exactos de forma clara y textual.

Vista de Barras (Comparativa): Utiliza un gráfico de barras, donde cada número (A, B o C) tiene su propio color. Es excelente para una comparación rápida e intuitiva, ya que los porcentajes exactos se muestran directamente encima de cada barra.

Vista de Pastel (Proporcional): Presenta los datos como un círculo dividido en "rebanadas" o sectores. El tamaño de cada sector es proporcional al valor que representa. Incluye una leyenda para identificar cada segmento y su porcentaje.

En esencia, la aplicación toma tres números, los valida y te ofrece tres formas dinámicas y diferentes de analizarlos.

Tecnologías Utilizadas
TypeScript con Orientación a Objetos que aprovecha interfaces, clases abstractas y herencia para implementar el patrón Factory Method de manera type-safe. HTML5 Canvas y CSS3 para renderizado de gráficos vectoriales y representaciones visuales de datos geométricos y porcentuales.

Arquitectura de Componentes Modulares donde cada factory y producto es un módulo independiente que puede ser extendido sin afectar el sistema existente. Sistema de Eventos para la comunicación entre la interfaz de usuario y la lógica de factorie.

Muestra Grafica del Funcionamiento

Calculadora de Porcentajes

Valor A:

3

Valor B:

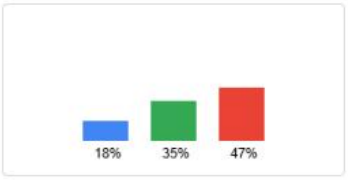
6

Valor C:

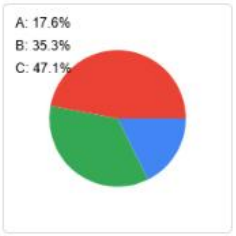
8

Calcular Porcentajes

Gráfica de Barras



Gráfica de Pastel



Resultados

3 + 6 + 8 = 17 → 100%

Variable	Valor	Porcentaje
A	3	17.6%
B	6	35.3%
C	8	47.1%

Conclusión

El proyecto implementó con éxito una Calculadora de Porcentajes que utiliza el patrón Factory Method para gestionar tres tipos de visualizaciones: gráfico de barras, gráfico de pastel y tabla de resultados. Esta arquitectura permitió una clara separación entre la lógica de creación de objetos, los cálculos matemáticos y la presentación visual.

La implementación del Factory Method facilitó la creación flexible de diferentes visualizaciones sin acoplar el código a clases concretas. Los diagramas UML desarrollados (clases, secuencia, componentes y flujo de vistas) demostraron la solidez de la arquitectura y la claridad en las interacciones entre los componentes.

Las principales ventajas obtenidas incluyen una alta extensibilidad para agregar nuevas visualizaciones, un código mantenible gracias al desacoplamiento, y la capacidad de cambiar implementaciones sin afectar al sistema existente. La aplicación final ofrece tres vistas completamente funcionales que responden dinámicamente a los cambios en los datos de entrada.

El patrón Factory Method resultó ser la elección adecuada para este proyecto, proporcionando una base sólida para futuras expansiones mientras se mantiene la coherencia arquitectónica y la facilidad de mantenimiento.