ARQUITECTURA DEL SOFTWARE

INGENIERIA EN SOFTWARE
ESTANCIA I
CATEDRATICO: ALICIA ORTIZ MONTES
PROYECTO MINDPEACE
INTEGRANTES: ALFREDO BERROCAL CHAVEZ
DAVID AZAEL HERNANDEZ ESCUDERO
ITZEL MANCILLA BETANZOS
ELIAS MAYOR GONZALEZ
MARCO ANTONIO SOLIS PARDO

Índice

Índice **2**
Introducción **3**
Arquitecturas **3**
De Tres Capas 3
Componentes del Sistema 3
Frontend (Capa de Presentación) 3
Capa de Lógica de Negocio (Capa Media) 3
Backend (Capa de Datos) 4
Cliente-Servidor 4
Objetivo 4
Componentes del sistema 5
Frontend (Cliente) 5
Interfaz del usuario 5
Backend (Servidor) 5
Gestión de datos 5
Autenticación y Seguridad: 5
Principios **6**
Principio de Segregación de Interfaces (ISP) 6
Aplicación en nuestra página 6
Ventajas 6
Principio de Responsabilidad Única (SRP) 6
Ventajas 6
Desventajas: 6
Principio abierto cerrado 7
Ventajas 7
Desventajas 7
Principio de Inversión de Dependencia (DIP) 7
Ventajas: 8
Desventajas: 8
Patrones **8**
Prototype 8
Pros y Contras 9
Imagen generada con graphviz - Prototype 10
Factory Method 10
Pros y Contras 11
Imagen generada con graphviz- Factory Method 12
Conclusión **12**
Bibliografía **13**

Introducción

El proyecto MindPeace se centra en desarrollar una plataforma web para la detección y manejo de la ansiedad, combinando evaluaciones interactivas y recursos educativos. Hemos adoptado una arquitectura cliente-servidor en tres capas para garantizar una clara separación de responsabilidades, mejorando la escalabilidad y mantenibilidad del sistema. Utilizamos patrones de diseño como Factory Method y Prototype para lograr flexibilidad y extensibilidad. Además, aplicamos principios SOLID, incluyendo la segregación de interfaces, responsabilidad única, abierto/cerrado e inversión de dependencia, para asegurar un código modular y limpio. En este documento, se presenta una propuesta detallada de la arquitectura del software, sentando las bases para el desarrollo y evolución continua de MindPeace.

Arquitecturas

De Tres Capas

Es una arquitectura que organiza las aplicaciones en tres niveles informáticos lógicos y físicos: el nivel de presentación o interfaz de usuario, el nivel de aplicación, donde se procesan los datos, y el nivel de datos, donde se almacenan y gestionan los datos asociados con la aplicación.

Componentes del Sistema

Frontend (Capa de Presentación)

Interfaz del usuario

- Página de inicio con información general y opciones de navegación.
- Formularios interactivos para encuestas de ansiedad.
- Panel de usuario donde pueden ver y actualizar su perfil.
- Sección de recursos con tutoriales, videos y ejercicios.
- Página de login y registro de usuarios.

Funciones clave

- Presentación de la interfaz de usuario de forma intuitiva.
- Comunicación con la capa de lógica de negocio para obtener y mostrar datos.
- Manejo de interacciones del usuario y respuesta en tiempo real.

Capa de Lógica de Negocio (Capa Media)

Funciones principales

- Procesamiento de encuestas para calcular y analizar resultados.
- Gestión de usuarios: crear, actualizar y eliminar perfiles.
- Recomendación de contenido según los resultados de las encuestas.
- Manejo de sesiones y autenticación de usuarios.
- Integración con APIs externas para funcionalidades adicionales.

Roles

- Implementación de la lógica central.
- Manejo de las reglas del negocio y procesamiento de transacciones.
- Coordinación entre la capa de presentación y la capa de datos.

Backend (Capa de Datos)

Gestión de datos

- Almacenar respuestas de encuestas en una base de datos segura.
- Gestión de perfiles de usuario y sus datos personales.
- Almacenamiento y entrega de contenido multimedia (tutoriales, videos, ejercicios).

Autenticación y Seguridad

- Sistema de registro y login seguro con cifrado de contraseñas.
- Autorización basada en roles para acceder a diferentes partes del sitio.

Funciones

- Almacenamiento y recuperación eficiente de datos.
- Gestión de conexiones a la base de datos y transacciones.
- Provisión de datos a la capa de lógica de negocio según sea necesario

Cliente-Servidor

Objetivo

Desarrollar una página web interactiva y accesible que permita a los usuarios realizar encuestas para evaluar su nivel de ansiedad, acceder a su perfil personal y explorar tutoriales, videos y ejercicios diseñados para ayudar a controlar la ansiedad. La página también contará con un sistema de autenticación de usuarios (login).

Componentes del sistema

Frontend (Cliente)

Interfaz del usuario

- Página de inicio con información general y opciones de navegación.
- Formularios interactivos para encuestas de ansiedad.
- Panel de usuario donde pueden ver y actualizar su perfil.
- Sección de recursos con tutoriales, videos y ejercicios.
- Página de login y registro de usuarios.

Backend (Servidor)

Gestión de datos

- Almacenar las respuestas de las encuestas en una base de datos segura.
- Gestión de perfiles de usuario y sus datos personales.
- Almacenamiento y entrega de contenido multimedia (tutoriales, videos, ejercicios).

Autenticación y Seguridad:

- Sistema de registro y login seguro con cifrado de contraseñas.
- Autorización basada en roles para acceder a diferentes partes del sitio.

Principios

Principio de Segregación de Interfaces (ISP)

Aplicación en nuestra página

En nuestra página web para la evaluación y gestión de la ansiedad, aplicaremos el principio de segregación de interfaces para mejorar la modularidad y la mantenibilidad del código. Esto significa que cada componente de nuestra página (como encuestas, perfiles de usuario y recursos de ayuda) tendrá interfaces específicas y enfocadas en sus funciones particulares.

Ventajas

1. Reducción de Dependencias innecesarias. Cada módulo solo dependerá de lo que necesita, evitando complejidad innecesaria.
2. Mejora de la Flexibilidad y la Extensibilidad: Facilita la modificación y extensión de funcionalidades sin afectar otros módulos.
3. Claridad y Legibilidad del Código: Interfaces más pequeñas y específicas hacen que el propósito de cada una sea más claro y el código más fácil de entender.

Principio de Responsabilidad Única (SRP)

Establece que cada módulo o clase debe tener una única razón para cambiar, es decir, una sola responsabilidad. En MindPeace, esto significa que cada componente como el manejo de usuarios o las encuestas, debe centrarse únicamente en su propia funcionalidad. Esto se aplica en cada capa de la arquitectura, asegurando que cada parte del sistema tenga una función específica, lo que facilita su mantenimiento y expansión.

Ventajas

1. **Mantenibilidad**: Facilita la comprensión y modificación del código.
2. **Pruebas**: Simplifica la creación de pruebas unitarias.
3. **Reutilización**: Permite reutilizar componentes en diferentes partes del sistema.
4. **Escalabilidad**: Facilita la adición de nuevas funcionalidades sin afectar otras áreas.
5. **Modularidad**: Mejora la organización del código al dividirlo en componentes específicos.

Desventajas:

1. **Complejidad inicial**: Requiere más planificación y diseño desde el inicio.
2. **Mayor cantidad de clases**: Puede resultar en un aumento del número de clases
3. **Sobrecarga en la gestión**: Más componentes para gestionar y coordinar.

Principio abierto cerrado

Este principio establece que "las entidades de software (clases, módulos, funciones, etc.) deben estar abiertas para su extensión, pero cerradas para su modificación".
En el proyecto MindPeace, se aplica el principio abierto/cerrado a través de la extensibilidad de funciones, uso de interfaces y herencia, y la incorporación de plugins o módulos adicionales. Se diseñan módulos que pueden ser extendidos sin modificar el código existente, se utilizan interfaces para que nuevas pruebas o contenidos se añadan como implementaciones independientes, y se estructura el proyecto para permitir la adición de características como plugins, facilitando la integración de nuevas funcionalidades sin alterar el núcleo del sistema.

Ventajas

1. **Facilidad de Mantenimiento**: Al mantener el código existente sin cambios y permitir extensiones, se reduce el riesgo de introducir errores en partes ya probadas y funcionales del sistema.
2. **Escalabilidad**: Permite que el proyecto crezca de manera sostenible. A medida que MindPeace evolucione y requiera más funcionalidades, se podrá añadir fácilmente sin necesidad de reescribir o modificar el código existente.
3. **Flexibilidad y Adaptabilidad**: Facilita la adaptación del software a las necesidades cambiantes sin involucrar grandes refactorizaciones. Esto es especialmente útil en el entorno educativo, donde las demandas y técnicas pueden evolucionar rápidamente.
4. **Fomento de la Colaboración**: Al tener un sistema modular donde nuevos módulos o funcionalidades pueden ser desarrollados independientemente por diferentes miembros o equipos, se fomenta la colaboración y se acelera el proceso de desarrollo.

Desventajas

1. **Complejidad Inicial Aumentada**: Diseñar sistemas que sean extensibles sin necesidad de modificar el código existente puede requerir una planificación y diseño más complejos desde el inicio.
2. **Sobrediseño**: Hay un riesgo de sobrediseño al intentar hacer cada parte del sistema extensible.
3. **Rendimiento**: A veces, el uso excesivo de abstracciones y la indirección necesaria para mantener los componentes cerrados a modificaciones puede afectar el rendimiento del sistema.
4. **Dificultades en la Prueba**: Los sistemas altamente modulares y extensibles pueden ser más difíciles de probar debido a las interacciones entre módulos o componentes.

Principio de Inversión de Dependencia (DIP)

El Principio de Inversión de Dependencia se centra en reducir las dependencias entre los módulos de software, lo que permite un diseño más robusto y mantenible. En el contexto de MindPeace, este principio se aplica asegurando que las capas de alta y baja nivel no dependan directamente unas de otras, sino que ambas interactúen a través de abstracciones.

Ventajas:

1. **Flexibilidad y Escalabilidad:** Al depender de abstracciones, se facilita la modificación y extensión de componentes individuales sin afectar el resto del sistema.
2. **Facilidad de Pruebas:** Permite realizar pruebas unitarias más efectivas y aisladas al poder reemplazar implementaciones concretas con objetos simulados (mocks) o de prueba.
3. **Desacoplamiento:** Promueve un menor acoplamiento entre módulos, lo cual reduce las dependencias directas y mejora la modularidad del sistema.

Desventajas:

1. **Complejidad Inicial:** Puede aumentar la complejidad del diseño inicial al requerir la definición de abstracciones e interfaces antes de desarrollar las implementaciones concretas.
2. **Sobrediseño:** Hay riesgo de sobrediseño si se implementan muchas abstracciones innecesarias, lo cual puede llevar a un código más difícil de entender y mantener.
3. **Rendimiento:** En algunos casos, la inyección de dependencias y la indirección a través de interfaces pueden introducir una sobrecarga que afecte el rendimiento del sistema.

Patrones

Prototype

El patrón de diseño Prototype es un enfoque creacional que permite la duplicación de objetos existentes sin que el código dependa directamente de las clases de estos objetos.

Aunque comúnmente asociado con la programación orientada a objetos (POO), en este contexto, el patrón Prototype se adaptará como una técnica de desarrollo de software. Esta aplicación implica la creación de versiones preliminares de un software con el objetivo de explorar su funcionalidad en situaciones reales. Este método facilita una comprensión más profunda de los requisitos y funcionalidades del sistema por parte de los desarrolladores, gracias a la utilización de modelos funcionales interactivos. Estos modelos son susceptibles de ser evaluados y refinados a través de iteraciones sucesivas, lo que permite ajustes continuos hasta alcanzar el resultado deseado.

Dado que ***MindPeace*** es una aplicación web diseñada para ayudar a los usuarios a detectar y gestionar la ansiedad, el uso del patrón de prototipo permite al equipo de desarrollo:

- Experimentar con diferentes interfaces y flujos de usuario para optimizar la experiencia de usuario (UX).

- Obtener retroalimentación temprana de los usuarios finales, lo que es crucial para un sistema centrado en la salud mental.
- Ajustar rápidamente las funcionalidades según las necesidades y preferencias de los usuarios.

Pros y Contras

Pros	***Contras***
Retroalimentación temprana: Permite identificar errores de diseño y requisitos no capturados antes de la implementación final.	**Limitaciones Técnicas:** Los prototipos no siempre pueden capturar la complejidad y el rendimiento del sistema final, lo que puede llevar a suposiciones incorrectas sobre la viabilidad técnica.
Comprensión Mejorada: Ayuda a todos los involucrados a entender mejor las funcionalidades propuestas y los requisitos del sistema.	**Costo y Tiempo:** Puede ser costoso y tiempo consumidor desarrollar múltiples iteraciones del prototipo.
Reducción de Riesgos: Disminuye el riesgo de fracaso del proyecto al validar ideas y supuestos en etapas tempranas.	

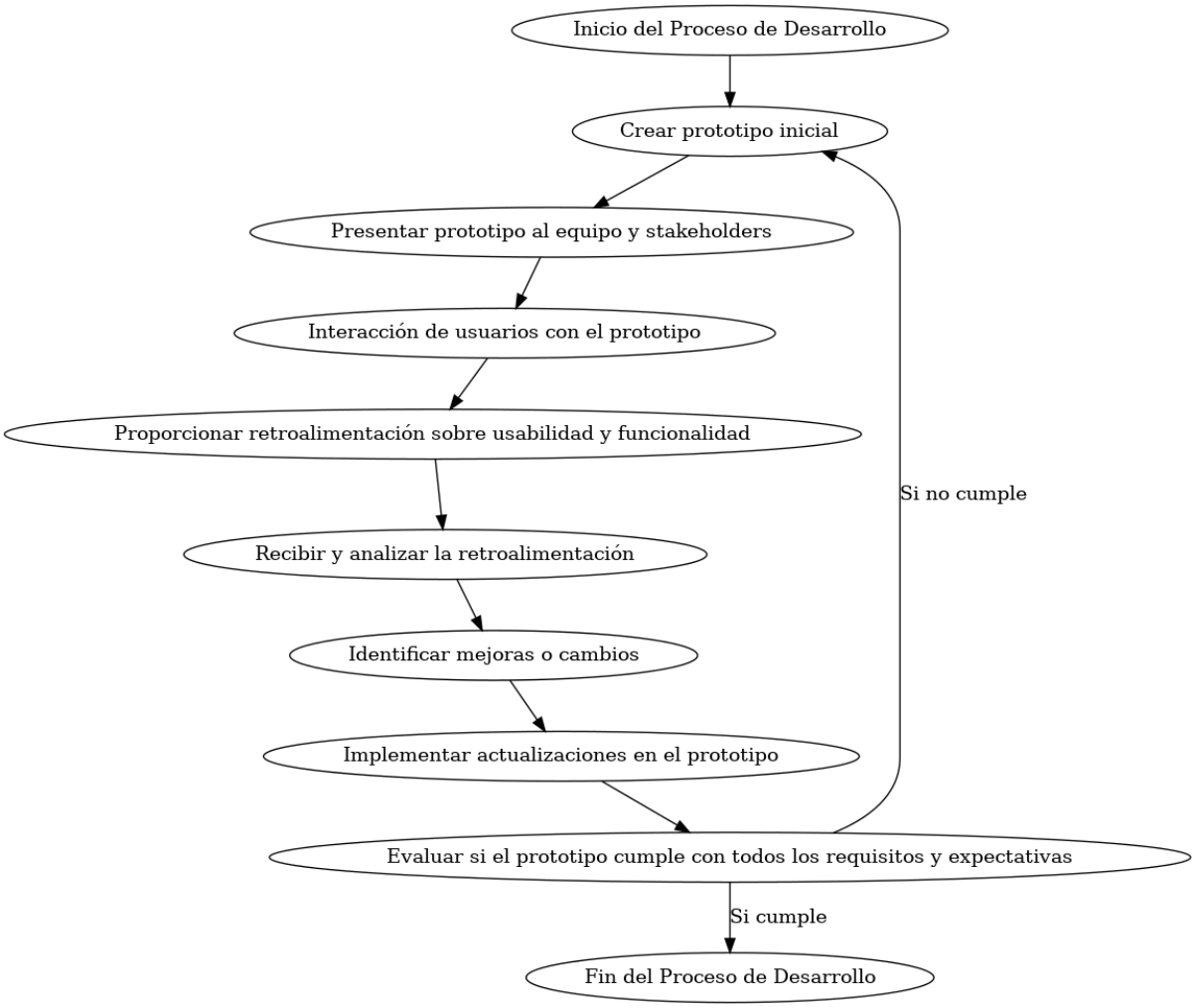

Imagen generada con graphviz - Prototype

Factory Method

El patrón de diseño Factory Method es un enfoque creacional que proporciona una interfaz para la creación de objetos en una superclase, pero permite que las subclases alteren el tipo de objetos que se crearán. Esto se logra definiendo un método fábrica, que las subclases pueden sobrescribir para crear objetos específicos sin alterar el código cliente que utiliza el objeto.

Aunque este patrón es ampliamente utilizado en la programación orientada a objetos (POO) para abstraer los detalles de la creación de objetos, su adaptación en el contexto de desarrollo de software es igualmente beneficiosa. El Factory Method facilita la gestión y extensión de sistemas complejos al encapsular la creación de objetos en métodos específicos. Esto permite desarrollar un sistema modular donde los cambios en la manera de crear objetos no afectan el código que los utiliza.

Por ejemplo, en el desarrollo de una aplicación como ***MindPeace***, que ofrece múltiples funcionalidades para gestionar la ansiedad, el patrón Factory Method podría utilizarse para:

- Crear diferentes tipos de tests o sesiones de manejo de ansiedad según las necesidades del usuario.
- Facilitar la integración de nuevas funcionalidades sin alterar el funcionamiento existente de la aplicación.
- Permitir que distintos desarrolladores trabajen en diferentes tipos de objetos o módulos de la aplicación sin interferir unos con otros (importante)

Pros y Contras

Encapsulación del código de creación: Configurabilidad y Reutilización, reduciendo duplicidades y errores, como en la creación de perfiles.	**Complejidad Incrementada:** La implementación de Factory Method puede hacer que el sistema de MindPeace sea inicialmente más complejo, especialmente si el número de componentes o variantes no es suficientemente grande como para justificar esta abstracción
Facilidad para Escalar y Expandir: Al utilizar el Factory Method, MindPeace puede fácilmente introducir nuevas funcionalidades y tipos de contenido para manejar la ansiedad, como videos, guías interactivas o nuevos tipos de evaluaciones, sin afectar el código existente.	**Potencial Ineficiencia en Tiempos de Respuesta:** La creación de objetos a través de métodos de fábrica podría introducir un retardo adicional en la carga de componentes de la aplicación, especialmente si la lógica de decisión es compleja o si se accede a recursos externos durante la creación.
Mantenimiento Simplificado: Centralizar la creación de objetos en fábricas específicas ayuda a mantener el código más organizado y simplifica el proceso de actualización y mantenimiento, ya que los cambios se realizan en un solo lugar.	

.

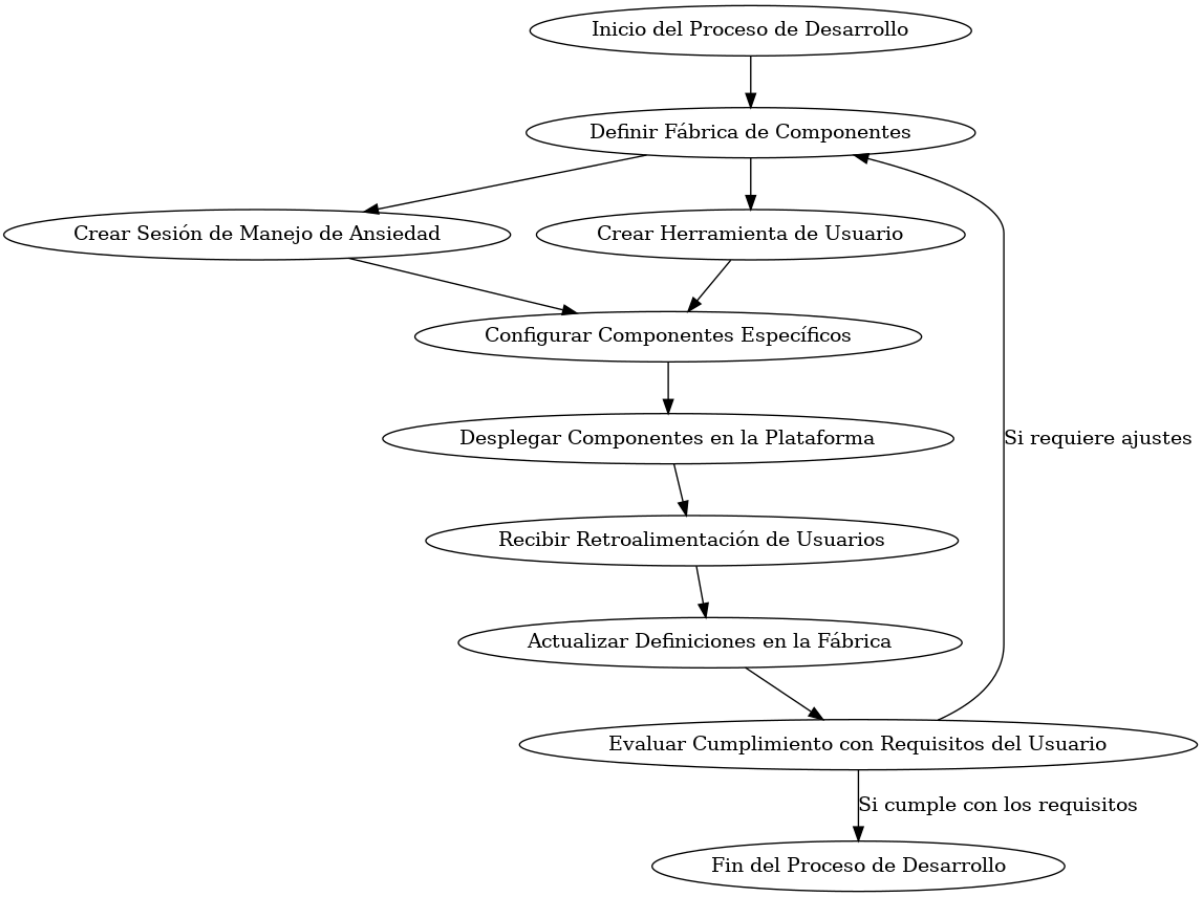

Imagen generada con graphviz- Factory Method

Conclusión

A lo largo del desarrollo de MindPeace, hemos adoptado diversas estrategias arquitectónicas y principios de diseño para crear una plataforma robusta y escalable que facilita la gestión y el manejo de la ansiedad. Mediante la aplicación de patrones de diseño como Factory Method y Prototype, junto con principios SOLID como Segregación de Interfaces, Responsabilidad Única, Abierto/Cerrado, y ahora Inversión de Dependencia, hemos asegurado que nuestro sistema sea mantenible y extensible. MindPeace no solo es un testimonio del impacto positivo de prácticas de desarrollo bien fundamentadas, sino también una herramienta vital en la lucha contra la ansiedad, proporcionando a los usuarios recursos accesibles y personalizados para mejorar su calidad de vida. Con cada iteración, MindPeace se acerca más a ser un recurso indispensable en el campo de la salud mental, demostrando la importancia de un buen diseño y arquitectura en el desarrollo de software.

Bibliografía

- *Prototype*. (2014). Refactoring.guru. https://refactoring.guru/es/design-patterns/prototype
- Programación.net. (25 Octubre 2016). Patrones de Diseño : Patrones de Creación — Prototipo. 12/10/2016, de Meet Magento Madrid Sitio web: http://programacion.net/articulo/patrones_de_diseno_y_patrones_de_creacion_prototipo_1005
- *¿Qué es la arquitectura de tres niveles?* | *IBM*. (s. f.). https://www.ibm.com/mx-es/topics/three-tier-architecture
- Colaboradores de Wikipedia. (2020, 3 noviembre). *Principio de responsabilidad única*. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/Principio_de_responsabilidad_%C3%BAnica
- *Factory Method*. (2014). Refactoring.guru. https://refactoring.guru/es/design-patterns/factory-method
-