****

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**ANEXO – 01**

**“PRINCIPIOS DE DISEÑO”**

Curso: Patrones de software

Docente: Mag. Patrick Cuadros Quiroga

**Integrantes:**

**Romero Roque, Angelica Beatriz (2019063327)**

**Chambilla Maquera, Aroceli Noemi (2018060897)**

**Arhuata Turpo, Alex (2019063634)**

**Lima Ccosi, Jeackory Geilly (2019063326)**

**Rojas Bedregal Brian Erik (2018060904)**

**Vilca Condori Erlang Fernando (2019064024)**

**Tacna – Perú**

***2023***

*Principios de Diseño*

Versión *1.0*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CONTROL DE VERSIONES | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | ACHM BRB | ACHM | PCQ | 07/06/2023 | Versión Original |

INDICE

[1. INTRODUCCION 5](#_Toc139935494)

[2. Principio Inversión del Control (IoC) 5](#_Toc139935495)

[3. Principio Separación de Responsabilidades 6](#_Toc139935496)

[4. Principio Abierto/Cerrado 6](#_Toc139935497)

[5. Principio de Inversión de Dependencias 6](#_Toc139935498)

[6. Conclusión 6](#_Toc139935499)

[7. Referencia Bibliográfica: 6](#_Toc139935500)

# INTRODUCCION

SOLID es un acrónimo que brinda la orientación ideal para escribir correctamente nuestros proyectos, nos ayuda muchísimo cuando trabajamos con programación orientada a objetos. La intención es seguir estás técnicas y principios para programar con buenos diseños y alcanzar un código eficiente y fácil de mantener.

Los principios SOLID son guías que pueden ser aplicadas en el desarrollo de software para eliminar malos diseños provocando que el programador tenga que refactorizar el código fuente hasta que sea legible y extensible. Puede ser utilizado con el desarrollo guiado por pruebas, y forma parte de la estrategia global del desarrollo ágil de software y desarrollo adaptativo de software. El costo de escribir un mal código es altísimo porque interrumpe el desarrollo y complica el mantenimiento haciendo incluso que en algunos casos se abandone el proyecto.

La idea es minimizar este problema y de hecho tratar de evitarlo al máximo. Nuestro objetivo debe ser capaz de poder escribir un código principalmente testeable, preparado para ser mantenido, tolerante a cambios y más legible.

# Principio Inversión del Control (IoC)

Spring Boot utiliza el patrón de diseño de Inversión de Control (IoC) o Inyección de Dependencias (DI). Esto permite que los objetos dependientes sean creados, configurados y gestionados por el contenedor de Spring (ApplicationContext), en lugar de que los objetos los creen y gestionen ellos mismos. Esto promueve un acoplamiento flexible y facilita la reutilización de componentes.

# Principio Separación de Responsabilidades

Spring Boot fomenta la separación clara de responsabilidades en una aplicación. La capa de controladores se encarga de gestionar las solicitudes HTTP, los servicios se encargan de la lógica de negocio y los repositorios se encargan del acceso a la base de datos. Esta separación facilita la escalabilidad y el mantenimiento del código.

# Principio Abierto/Cerrado

Spring Boot se basa en el principio de Abierto/Cerrado al permitir la extensión de funcionalidades sin modificar el código fuente existente. Esto se logra a través de la configuración y personalización mediante anotaciones, sin necesidad de modificar el código subyacente. Por ejemplo, puedes agregar nuevas características a tus controladores mediante el uso de anotaciones como @RequestMapping sin cambiar el código fuente del controlador existente.

# Principio de Inversión de Dependencias

Spring Boot promueve el uso de interfaces y abstracciones en lugar de depender de implementaciones concretas. Esto permite una mayor flexibilidad y modularidad en el diseño, ya que los componentes dependen de contratos bien definidos en lugar de depender directamente de implementaciones concretas. Además, Spring Boot proporciona el mecanismo de Inyección de Dependencias (DI) para gestionar las dependencias y facilitar la inversión de dependencias.

# Conclusión

En conclusión, los principios es una guía valiosa para el diseño de software orientado a objetos, ya que promueve la creación de sistemas flexibles, mantenibles y escalables. Al aplicar estos principios, se obtiene un código más limpio, modular y reutilizable, lo que facilita la evolución del software a lo largo del tiempo y mejora la calidad general del sistema.

# Referencia Bibliográfica:

* Vista de Creación de una guía para la utilización de la arquitectura “clean architecture” en aplicaciones Android con lenguaje de programación Java, utilizando como modelo la implementación de un prototipo de monitoreo deportivo para un equipo de ultimate. (s/f). Utp.ac.pa. Recuperado de https://revistas.utp.ac.pa/index.php/ric/article/view/1750/html
* de Aplicações Web Utilizando Os Princípios Solid, M. V. R. D. (2022). UNIVERSIDADE DO SUL DE SANTA CATARINA. Com.br. Recuperado de https://repositorio.animaeducacao.com.br/bitstream/ANIMA/22977/1/2022-tcc\_Miguel-Ramos.pdf
* Benito, M., & Felipe, J. (2021). Metodologías ágiles de desarrollo aplicadas a la enseñanza de la programación. Recuperado de https://rua.ua.es/dspace/handle/10045/116132#vpreview
* DESARROLLO DEL MÓDULO DE CONTABILIDAD EN LENGUAJE NET CORE CUMPLIENDO CON LOS PRINCIPIOS DE DESARROLLO SOLID Y DE ARQUITECTURA LIMPIA USANDO LOS FRAMEWORK ENTITY, First, C., & DEVEXPRESS Y BOOTSTRAP 4 EN UN REPOSITORIO SQL SERVER CON CONTROL DE VERSIONES GIT BAJO EL MARCO DE TRABAJO SCRUM EN MICROSHIF. (s/f). PROGRAMA DE INGENIERÍA DE SISTEMAS. Edu.co. Recuperado de http://repositoriodspace.unipamplona.edu.co/jspui/bitstream/20.500.12744/5636/1/CUADRADO\_2019\_TG.pdf