GA4-220501095-AA2-EV05 - DESARROLLAR LA ARQUITECTURA DE SOFTWARE DE ACUERDO AL PATRÓN DE DISEÑO SELECCIONADO.

Sor Junny Londoño Rivera

Donaldo Andrés Beltrán Prieto

Instructor

Servicio Nacional de Aprendizaje-SENA

ANALISIS Y DESARROLLO DE SOFTWARE (2627038)

Regional Quindío.

2023

INTRODUCCIÓN.

En el ámbito del software cada vez es más común escuchar el término “arquitectura de software”, y encontrar oportunidades de empleo para “arquitectos de software”. Aun así, este concepto tiende a ser malentendido y la falta de comprensión al respecto de sus principios frecuentemente repercute de manera negativa en la construcción de sistemas de software.

El concepto de arquitectura de software se refiere a la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, modificabilidad), y servir como guía en el desarrollo. Al igual que en la ingeniería civil, las decisiones críticas relativas al diseño general de un sistema de software complejo deben de hacerse desde un principio. El no crear este diseño desde etapas tempranas del desarrollo puede limitar severamente el que el producto final satisfaga las necesidades de los clientes. Además, el costo de las correcciones relacionadas con problemas en la arquitectura es muy elevado. Es así que la arquitectura de software juega un papel fundamental dentro del desarrollo.

La arquitectura de software es de especial importancia ya que la manera en que se estructura un sistema tiene un impacto directo sobre la capacidad de este para satisfacer lo que se conoce como los atributos de calidad del sistema. Ejemplos de atributos de calidad son el desempeño, que tiene que ver con el tiempo de respuesta del sistema a las peticiones que se le hacen, la usabilidad, que tiene que ver con qué tan sencillo les resulta a los usuarios realizar operaciones con el sistema, o bien la modificabilidad, que tiene que ver con qué tan simple resulta introducir cambios en el sistema. Los atributos de calidad son parte de los requerimientos (no funcionales) del sistema y son características que deben expresarse de forma cuantitativa. No tiene sentido, por ejemplo, decir que el sistema debe devolver una petición “de manera rápida”, o presentar una página “ligera”, ya que no es posible evaluar objetivamente si el sistema cubre o no esos requerimientos.

La manera en que se estructura un sistema permitirá o impedirá que se satisfagan los atributos de calidad. Por ejemplo, un sistema estructurado de tal manera que una petición deba transitar por muchos componentes antes de que se devuelva una respuesta podría tener un desempeño pobre. Por otro lado, un sistema estructurado de tal manera que los componentes estén altamente acoplados entre ellos limitará severamente la modificabilidad. Curiosamente, la estructuración tiene un impacto mucho menor respecto a los requerimientos funcionales del sistema. Por ejemplo, un sistema difícil de modificar puede satisfacer plenamente los requerimientos funcionales que se le imponen.

Además de los atributos de calidad, la arquitectura de software juega un papel fundamental para guiar el desarrollo. Una de las múltiples estructuras que la componen se enfoca en partir el sistema en componentes que serán desarrollados por individuos o grupos de individuos.

La identificación de esta estructura de asignación de trabajo es esencial para apoyar las tareas de planeación del proyecto.

A continuación, se describen dichas etapas.

Requerimientos: La etapa de requerimientos se enfoca en la captura, documentación y priorización de requerimientos que influencian la arquitectura. Como se mencionó anteriormente, los atributos de calidad juegan un papel preponderante dentro de estos requerimientos, así que esta etapa hace énfasis en ellos. Otros requerimientos, sin embargo, son también relevantes para la arquitectura, estos son los requerimientos funcionales primarios y las restricciones.

Diseño: La etapa de diseño es la etapa central en relación con la arquitectura y probablemente la más compleja. Durante esta etapa se definen las estructuras que componen la arquitectura. La creación de estas estructuras se hace en base a patrones de diseño, tácticas de diseño y elecciones tecnológicas.

Documentación: Una vez creado el diseño de la arquitectura, es necesario poder comunicarlo a otros involucrados dentro del desarrollo. La comunicación exitosa del diseño muchas veces depende de que dicho diseño sea documentado de forma apropiada.

Evaluación: Dado que la arquitectura de software juega un papel crítico en el desarrollo, es conveniente evaluar el diseño una vez que este ha sido documentado con el fin de identificar posibles problemas y riesgos. La ventaja de evaluar el diseño es que es una actividad que se puede realizar de manera temprana (aún antes de codificar), y que el costo de corrección de los defectos identificados a través de la evaluación es mucho menor al costo que tendría el corregir estos defectos una vez que el sistema ha sido construido.

SELECCIONAR DE CADA UNA DE LAS CATEGORÍAS, UN PATRÓN DE DISEÑO Y APLICARLO AL PROYECTO.

NOMBRE DEL PATRÓN: PATRÓN CREACIONAL.

CATEGORIA: PROTOTYPE, CLON O CLONE.

Tiene como finalidad crear nuevos objetos clonando una instancia creada previamente. Este patrón especifica la clase de objetos a crear mediante la clonación de un prototipo que es una instancia ya creada.

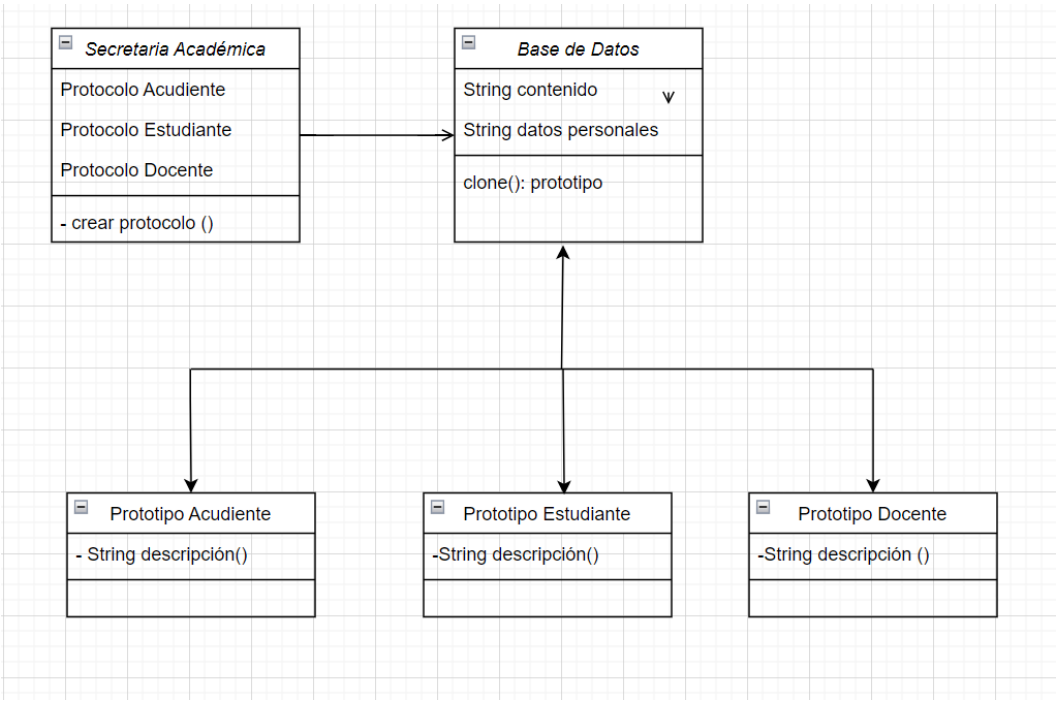
PROBLEMA

Se desea generar un clonar toda la información posible sobre el registro de notas académicas en las instituciones educativas establecidas.

SOLUCIÓN

De la mano de esta categoría, buscaré adaptarla a la ejecución de mi Software, con el fin de crear y fortalecer una base de datos muy sólida y real.

DIAGRAMA



NOMBRE DEL PATRÓN: PATRÓN ESTRUCTURAL

CATEGORIA: DECORATOR

Permite a los desarrolladores añadir y eliminar las dependencias de un objeto de manera dinámica y, cuando sea necesario, durante el tiempo de ejecución.

PROBLEMA

Sin duda alguna a lo largo de la ejecución del Software para el registro de notas académicas, se tendrán que realizar diferentes modificaciones ya sea, por petición del cliente, por errores encontrados, fallas identificadas y demás, no obstante, lo que deba permanecer se conservara sin ser sujeto a cambios.

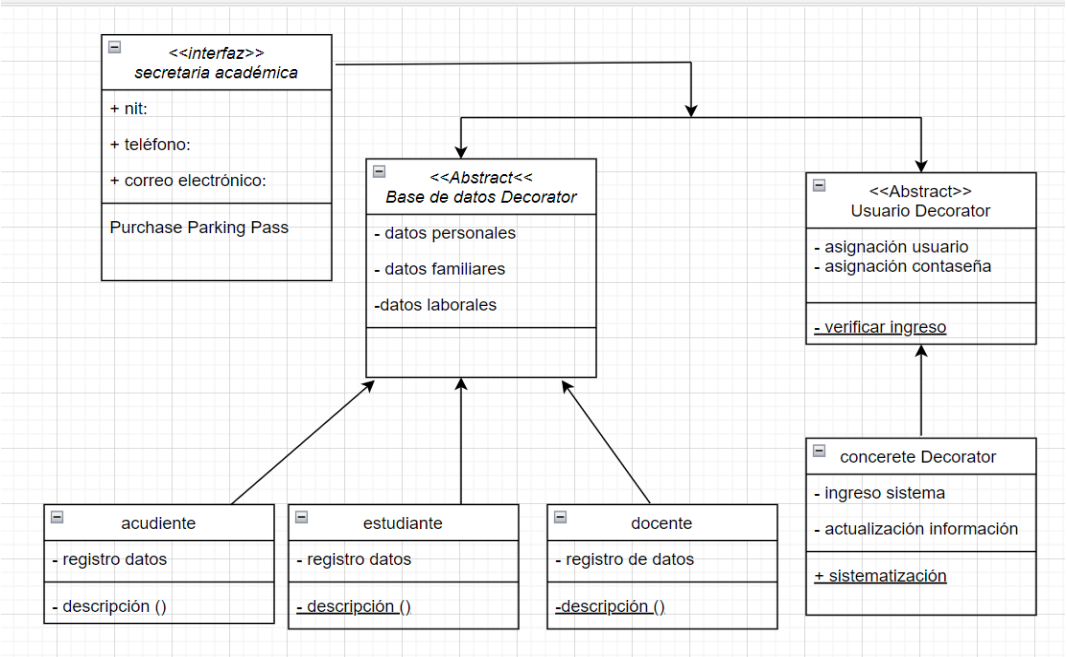
SOLUCIÓN

Se establecerán algunas clases o distinciones con el fin de reconocer lo que se debe modificar y no tocar lo que se debe conservar intacto, para ellos se acudirá a las palabras: Líder y jefe, lo cual significará:

Líder: Lo que se conserva.

Jefe: Lo que esta susceptible a modificaciones.

DIAGRAMA.



NOMBRE DEL PATRÓN: PATRÓN DE COMPORTAMIENTO

CATEGORIA: MEDIATOR

Define un objeto que encapsula cómo un conjunto de objetos interactúa. Este patrón de diseño está considerado como un patrón de comportamiento debido al hecho de que puede alterar el comportamiento del programa en ejecución.

PROBLEMA

Cuando interfieren varios partes, es inevitable que, por diversos interés y modos de actuar, se puedan presentar problemas de comunicación.

SOLUCIÓN

Se establece que el líder será quien realice el proceso de la comunicación entre las partes que conformarán el Software de registro de notas, con el fin de brindar un principio fundamental como lo suele ser, asertividad entre lo que se dice y se hace.

DIAGRAMA

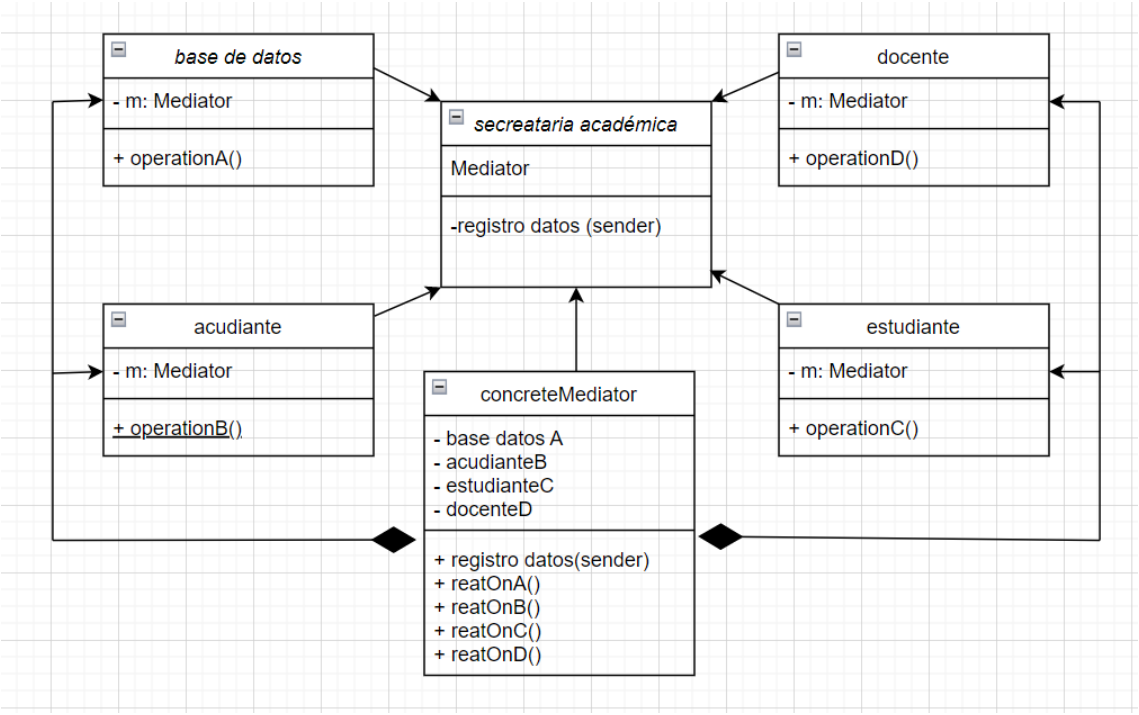


DIAGRAMA DE COMPONENTES

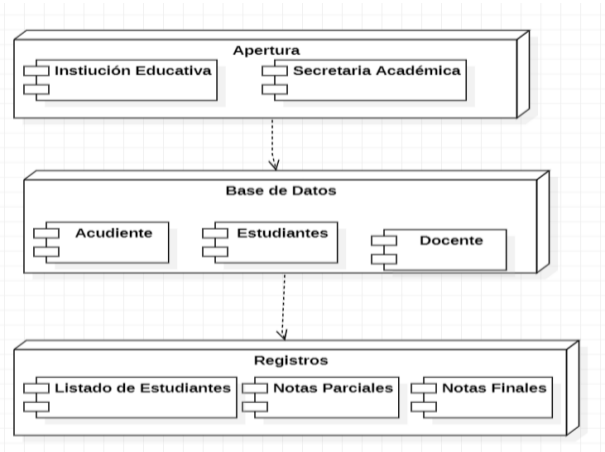
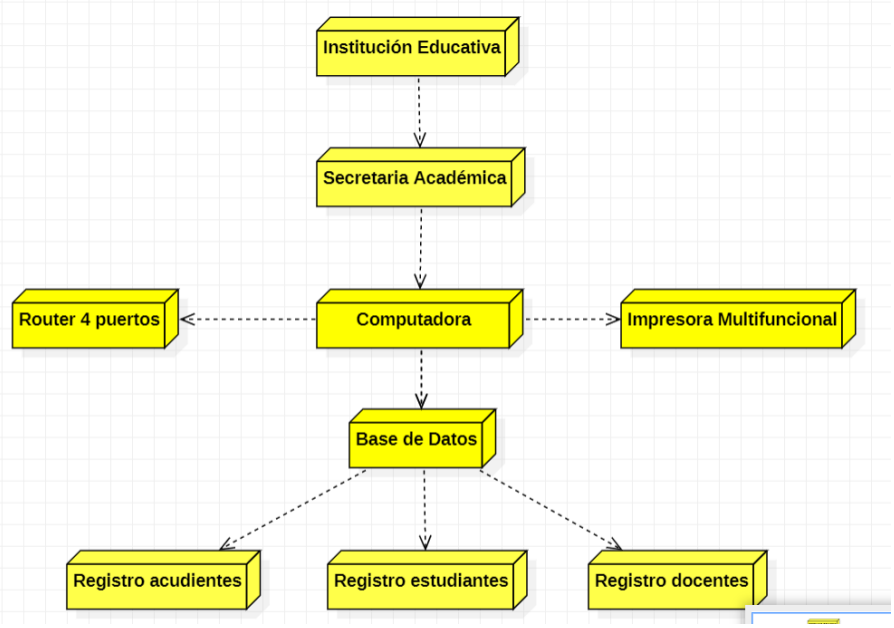


DIAGRAMA DE DESPLIEGUE



CONCLUIONES.

La Importancia de la Arquitectura de Software radica principalmente en la búsqueda temprana de problemas para que se tome decisiones tempranas de arquitectura ayudando con el diseño y la solución de algunos problemas que encuentra, además de el alto nivel de abstracción que nos brinda al hacerlo junto al diseño, vinculada con requerimientos no funcionales fuerte impulso en la academia y la industria ya que da un marco de entendimiento y mejor desarrollo del software además de brindarnos herramientas arquitectónicas aún en proceso de definición y desarrollo que nos permiten desarrollar mejor nuestras aplicaciones de software, resta elaborar: tácticas arquitectónicas, métodos basados en arquitectura, vínculo entre conceptos de arquitectura, DSLS, factorías, building blocks.

Implementar una arquitectura nos ayuda a entender mejor de qué trata nuestro software, a centrarnos en el dominio de nuestra aplicación, que es el valor real y que al fin y al cabo es la razón que nos lleva a escribir software. Implementar un modelo rico basado en nuestro dominio hace que todos los miembros del equipo tengan el mismo vocabulario y fuerza a ponerle nombres a conceptos por lo que facilita el entendimiento. Nos facilita también que nuestro código sea más mantenible, testable y en consecuencia nos ayuda a cumplir con los principios SOLID.

Finalmente, los diseños arquitectónicos que se crean en una organización pueden ser reutilizados para crear sistemas distintos. Esto permite reducir costos y aumentar la calidad, sobre todo si dichos diseños han resultado previamente en sistemas exitosos.

BIBLIOGRAFÍA

https://refactoring.guru/es/design-patterns/mediator

SENA (2023). Diseño de patrones de *software*. *https://sena.territorio.la/perfil.php?id=25323023*

https://www.youtube.com/watch?v=I8Tws3MKNnY