**UNIVERSIDAD AUTÓNOMA GABRIEL RENÉ MORENO**

**FACULTAD DE CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES**

**PATRONES DE DISEÑO**

**Estudiante:** Mauricio Velasco Torrico

**Registro:** 216095581

Diciembre 2022

Santa Cruz de la Sierra - Bolivia

Contenido

[1. Caso de Estudio 3](#_Toc122489893)

[2. Patrones Utilizados 4](#_Toc122489894)

[2.1. Patrón Strategy-Estrategia 4](#_Toc122489895)

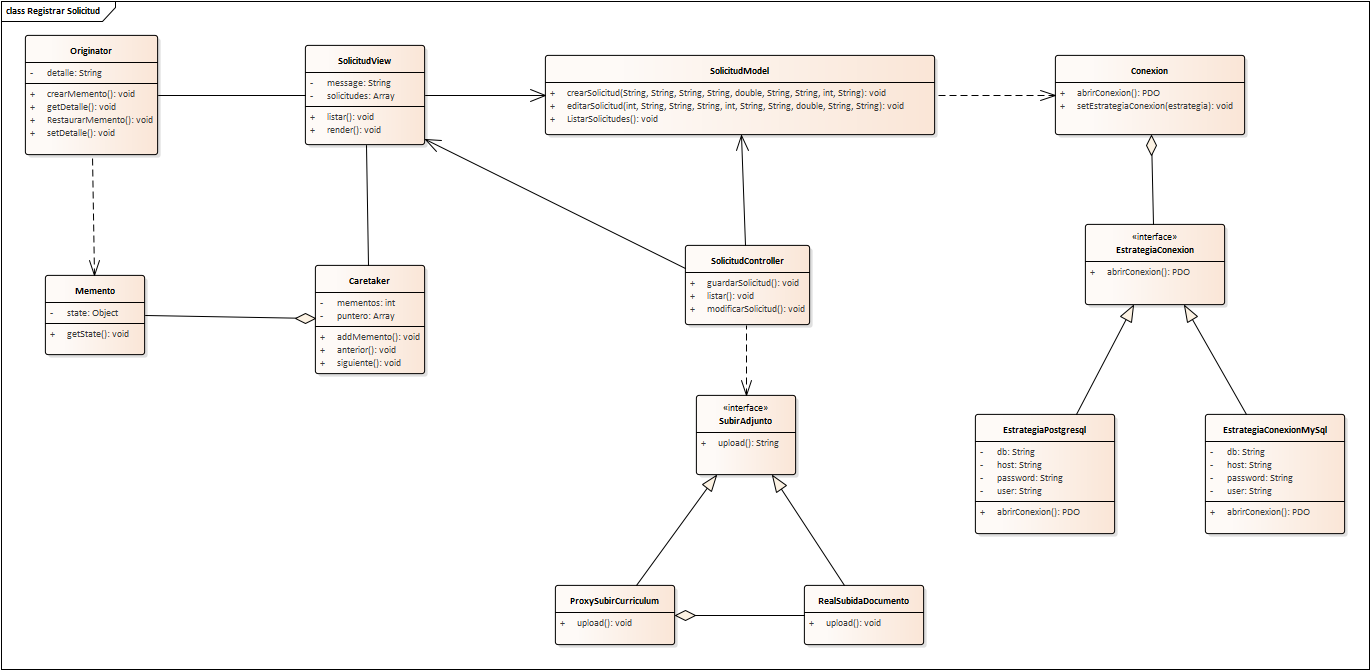
[2.2. Patrón Memento 5](#_Toc122489896)

[2.3. Patrón Proxy 6](#_Toc122489897)

[3. Bibliografía 7](#_Toc122489898)

# 1. Caso de Estudio

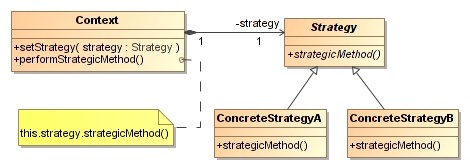
Se requiere crear un formulario para mandar solicitudes de trabajo, en donde se necesita conectar con diferentes Bases de Datos utilizando el Patrón Estrategia, además de poder guardar las frases que escriba el usuario en la pregunta que le realiza el formulario, para esto se debe utilizar el Patrón Memento y para subir los currículums utilizar el patrón proxy.

**Diagrama**

# 2. Patrones Utilizados

## 2.1. Patrón Strategy-Estrategia

Es un patrón de comportamiento, permite establecer en tiempo de ejecución el rol de comportamiento de una clase. Stretagy se basa en el polimorfismo para implementar una serie de comportamientos que podrán ser intercambiados durante la ejecución del programa, logrando con esto que un objeto se pueda comportar de forma distinta según la estrategia establecida.

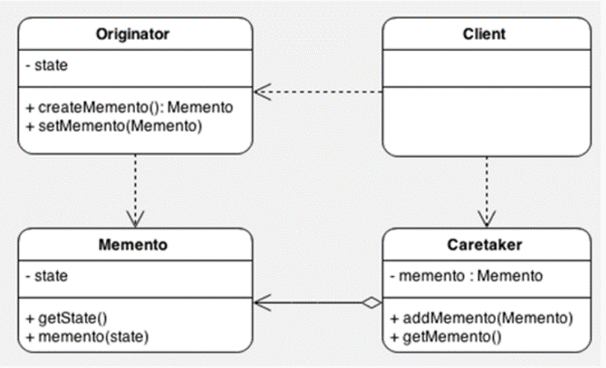
**Diagrama Genérico**

**Componentes**

* + **Context:** La clase Contextomantiene una referencia a una de las estrategias concretas y se comunica con este objeto únicamente a través de la interfaz estrategia.
  + **Estrategy:** La interfaz **Estrategia** es común a todas las estrategias concretas. Declara un método que la clase contexto utiliza para ejecutar una estrategia.
  + **Concrete Strategy:** Las Estrategias Concretasimplementan distintas variaciones de un algoritmo que la clase contexto utiliza.

## 2.2. Patrón Memento

Memento es un patrón de diseño de comportamiento, que guarda parte o todo el estado interno de un objeto, para que este estado pueda ser restaurado posteriormente. Esta operación debe ocurrir sin romper el principio del encapsulamiento.

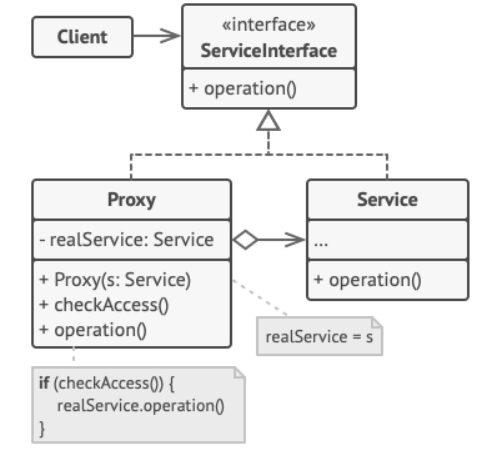
**Diagrama Genérico**

**Componentes**

* + **Originator:** Es el objeto para el que se va a guardar el estado. Crea el Memento y lo utiliza para recuperar su estado
  + **Memento:** Almacena el estado del Originator en un momento determinado
  + **Caretaker:** Registra los cambios del Originator. Nos permite viajar entre los distintos estados del Originator.
  + **Cliente:** Es el que afecta al Originator y registra al nuevo estado con el Caretaker. Es decir, realiza el cambio sobre el objeto y registra el estado.

## 2.3. Patrón Proxy

Proxy es un patrón de diseño estructural que proporciona un objeto que actúa como sustituto de un objeto de servicio real utilizado por un cliente. Un proxy recibe solicitudes del cliente, realiza parte del trabajo (control de acceso, almacenamiento en caché, etc.)



**Componentes**

* La **Interfaz de Servicio** declara la interfaz del Servicio. El proxy debe seguir esta interfaz para poder camuflarse como objeto de servicio.
* **Servicio** es una clase que proporciona una lógica de negocio útil.
* La clase **Proxy** tiene un campo de referencia que apunta a un objeto de servicio. Cuando el proxy finaliza su procesamiento (por ejemplo, inicialización diferida, registro, control de acceso, almacenamiento en caché, etc.), pasa la solicitud al objeto de servicio.
* El **Cliente** debe funcionar con servicios y proxies a través de la misma interfaz. De este modo puedes pasar un proxy a cualquier código que espere un objeto de servicio.

# 3. Bibliografía

Alexander S. (2021). Sumérgete en los Patrones de Diseño.

https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/strategy-pattern/