Conozca todos los pasos del desarrollo de aplicaciones bien diseñadas (con código)

Publicado el 27 de julio de 2021

Esta publicación ayuda a comprender el proceso de desarrollo de aplicaciones bien diseñadas y la arquitectura de un proyecto de software desde los requisitos hasta una arquitectura en capas desacoplada completamente funcional.

No se centrará en la parte teórica del análisis comercial (como historias de usuarios o casos de uso) y tampoco desviará la atención a la creación de una aplicación web para evitar tener que lidiar con componentes de la interfaz de usuario (como HTML, JavaScript, CSS, etcétera). Esta publicación trabajará en los conceptos básicos de la programación orientada a objetos, cómo implementar varios patrones de diseño y usar varios estilos y principios arquitectónicos de un proyecto que resulte en código desacoplado y más fácil de entender.

El presente proyecto resultante tendrá como objetivo crear una plantilla bien formateada con arquitectura desacoplada que se pueda implementar en cualquier proyecto nuevo que pueda ponerse en marcha rápidamente.

# Requisitos – ABC Pet Shop

“ABC Pet Shop” es una cadena de tiendas de mascotas ubicada en Sydney y Melbourne. La tienda de mascotas requiere un sistema de gestión de clientes simple para sus puntos de venta minoristas con las siguientes características (en una primera fase).

## Fase 1 – requisitos

1. La aplicación requerirá 5 campos: nombre completo, número de teléfono, monto de la factura, fecha de la factura y dirección del cliente.
2. Hay dos tipos de clientes que vienen a la tienda de mascotas: 1) cliente; 2) visitante.

* Un cliente es una persona que viene y compra cosas en una tienda y está involucrada en una transacción financiera. Todos los campos mencionados en 1 son obligatorios para los clientes.
* Un visitante es una persona que viene a la tienda de mascotas para hacer una consulta, pero no compra nada en ese momento. Solo un nombre completo y un número de teléfono son obligatorios para un visitante.
* El sistema debería poder agregar tipos de clientes si es necesario en el futuro

1. El sistema debe ser capaz de:

* Agregar nuevas reglas de validación sin problemas y estas reglas de validación deberán ser flexibles y reutilizables por el sistema.
* El sistema debe tener la capacidad de mostrar, agregar, actualizar y eliminar datos de clientes.
* Por lo pronto, el sistema deberá usar el servidor SQL y ADO.net como capa de datos y en los próximos meses deberá migrar a Entity Framework. La migración debe ser perfecta sin muchos cambios a lo largo del sistema.
* El sistema debe tener la capacidad de cancelar cualquier modificación realizada en la pantalla, lo que significa que, si estamos editando un registro y cambiando valores, deberíamos poder volver a los valores anteriores.

Estos son los requisitos iniciales para la Fase 1 del proyecto. Ahora, para continuar con estos requisitos, es esencial comprender algunos conceptos básicos de los principios orientados a objetos (OOP) para identificar las clases.

## ¿Qué es una Entidad?

“Una entidad es una cosa con existencia distinta e independiente”. Las entidades son cosas que vemos en el mundo real como sustantivos (por ejemplo, persona, lugar o cosas). En una perspectiva orientada a objetos, un objeto de entidad representa un objeto en el dominio del problema del mundo real. Entonces, en esta publicación, el objeto o las entidades se utilizarán como sinónimos.

# Pasos de desarrollo de aplicaciones bien diseñadas

## Paso 1 🡪 Identificar Entidades u Objetos de su sistema

El software se desarrolla principalmente para resolver problemas del mundo real y proporcionar automatización a escenarios y personas del mundo real. Un código de software debe replicar entidades del mundo real en sus objetos. Por ejemplo: un proveedor que suministra bienes en el mundo real. Entonces el código de software debe tener una entidad llamada Proveedor. Para identificar entidades/objetos necesitamos identificar sustantivos a partir de una especificación funcional o una historia de usuario. Los sustantivos se convierten en entidades, un verbo se convierte en acciones para las entidades y los pronombres se convierten en las propiedades y el comportamiento de esas entidades.

Entre los requisitos existen los siguientes sustantivos: Sydney, Melbourne, cliente, visitante, tienda de mascotas.

Existe una práctica común utilizada por los arquitectos de seguir sustantivos, verbos y pronombres. Pero un arquitecto debe tener cuidado de no elegir sustantivos y verbos no deseados que no se relacionen con el sistema. En la lista anterior de sustantivos, Sydney y Melbourne son ciudades y no tienen conexión directa con otros requisitos del sistema. Los únicos sustantivos útiles son cliente y visitante, que es donde el requisito solicita realizar acciones como validar, agregar, actualizar y eliminar.

### ¿Qué es una clase?

En OOP, una clase es un modelo (o plantilla) para crear objetos. La clase actúa como una plantilla que define la naturaleza de un objeto futuro (que debe contener el nombre, las acciones y las propiedades del objeto).

Diagrama

Descripción generada automáticamente

### Procesos trifásicos de POO

La POO es un proceso de tres fases, a saber:

1. **Creación de plantillas:** Creación de clases y lógica de escritura en esas clases.
2. **Instanciar:** Cree entidades/objetos de estas clases y tráigalos a vivir en la RAM/computadora.
3. **Ejecutar:** Interactuar con estos objetos para lograr la funcionalidad del software.

Un arquitecto de software/desarrollador de software encontrará problemas de diseño comunes/recurrentes en estas tres fases. Los patrones de diseño tienen soluciones para esos problemas.

Los patrones de diseño se dividen en tres categorías que cubren estas fases:

|  |  |
| --- | --- |
| **Fase OOP** | **Categoría de Patrón de Diseño** |
| Creación de plantillas/clases | Estructural |
| Problemas de instanciación | Creacional |
| Problemas de tiempo de ejecución | De comportamiento |

Diagrama

Descripción generada automáticamente

Cree una clase de cliente con propiedades y acciones (métodos) en Visual Studio agregando una biblioteca de clases llamada "CustomerLibrary" en la solución "ABCPetShop":

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente