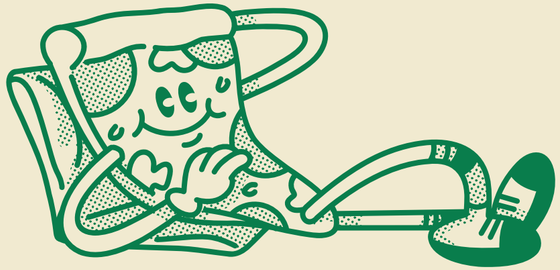
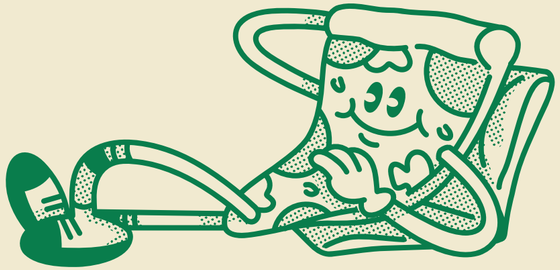
14/08/2025

APLICANDO LOS PRINCIPIOS SOLID

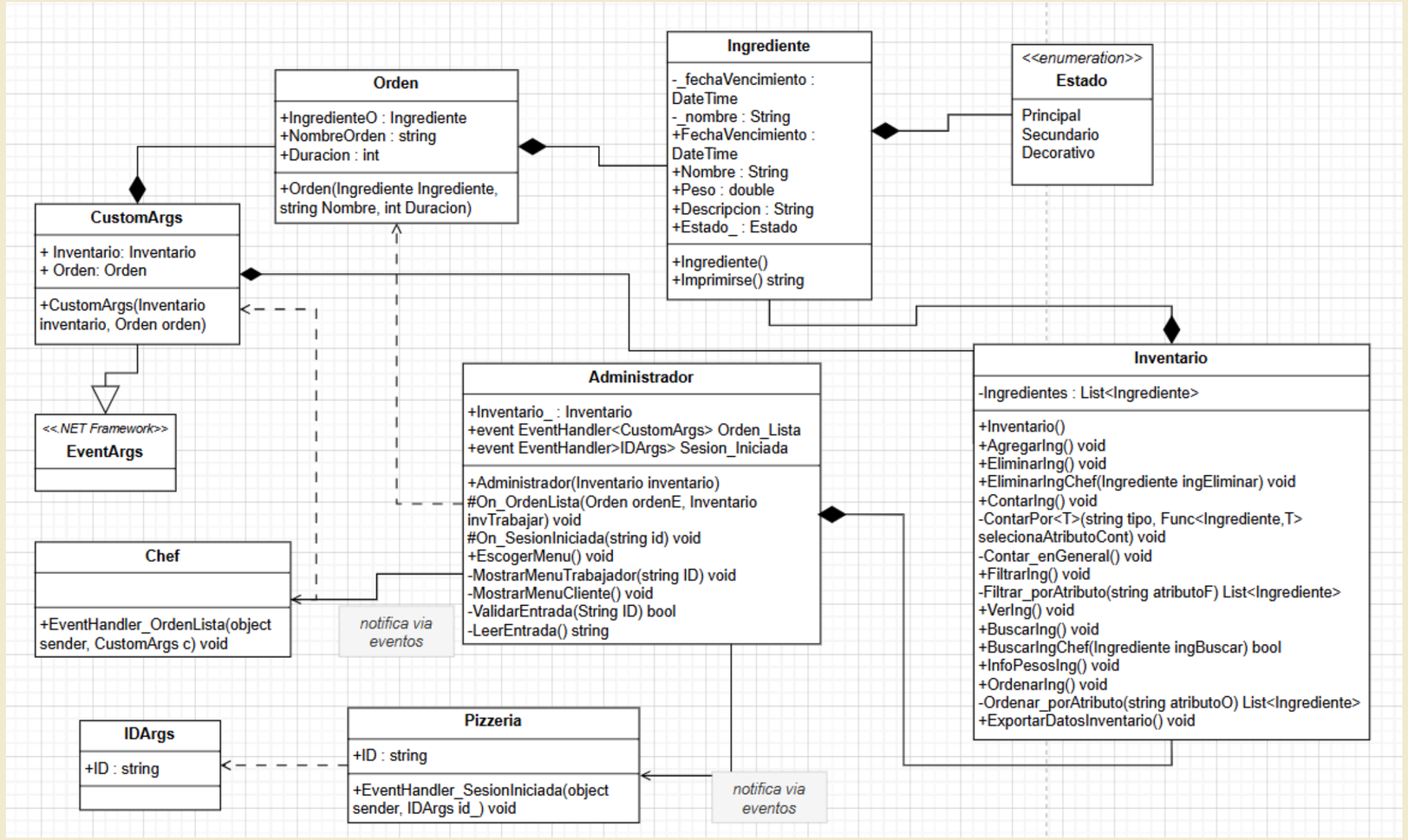
a una pizzeria



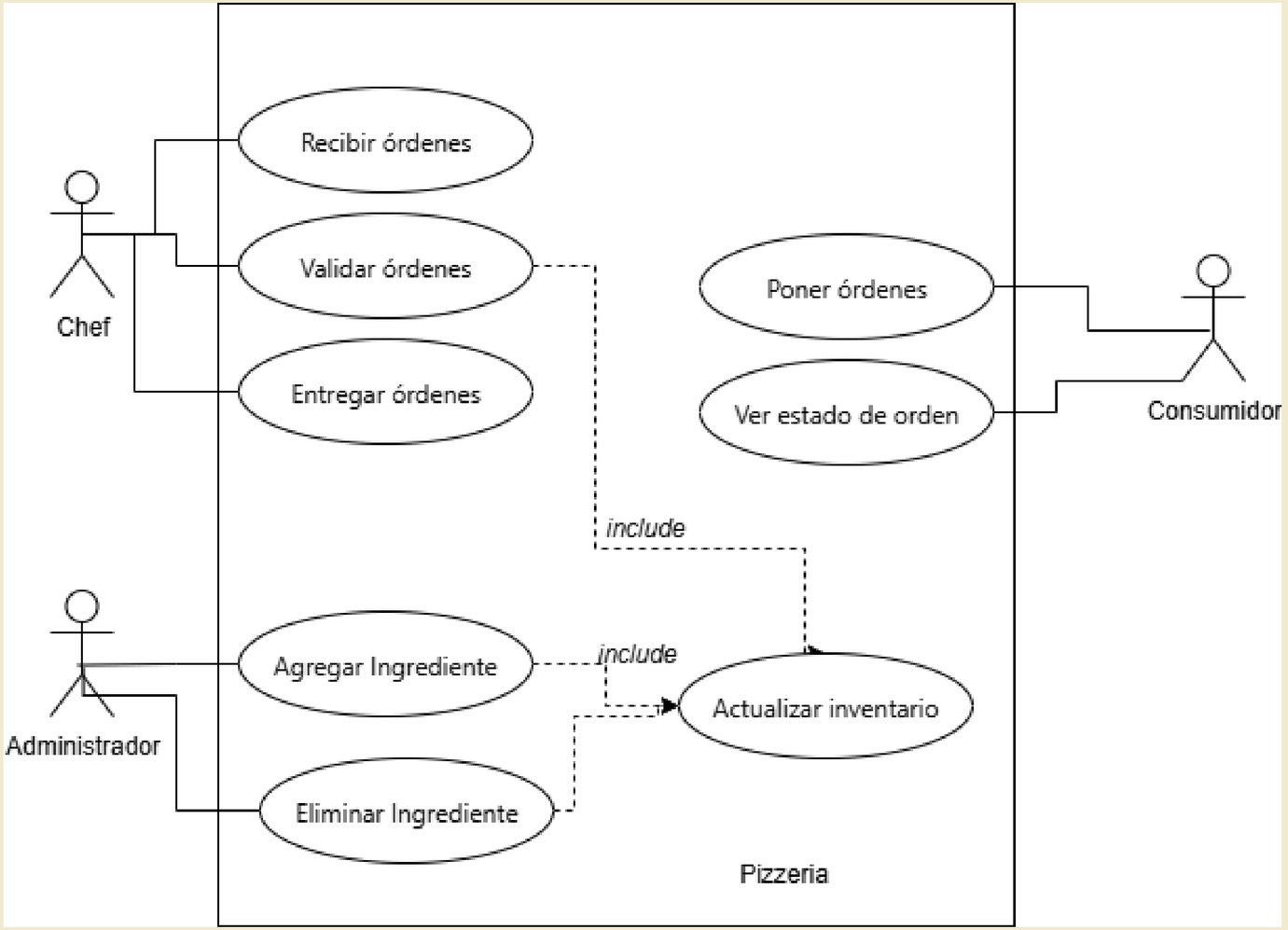


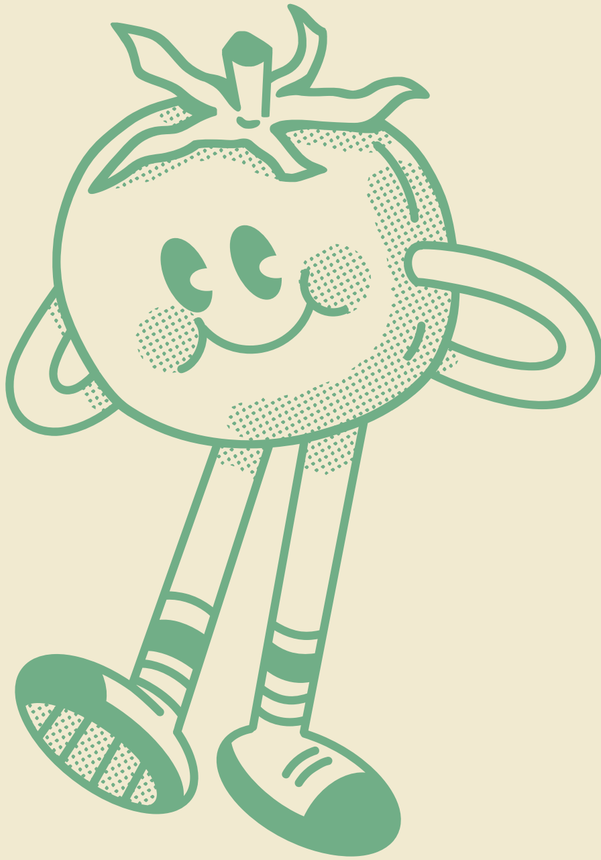
PROYECTO ANTES

- *Alto Acoplamiento*
- *Clases con muchas Responsabilidades*



ACTORES

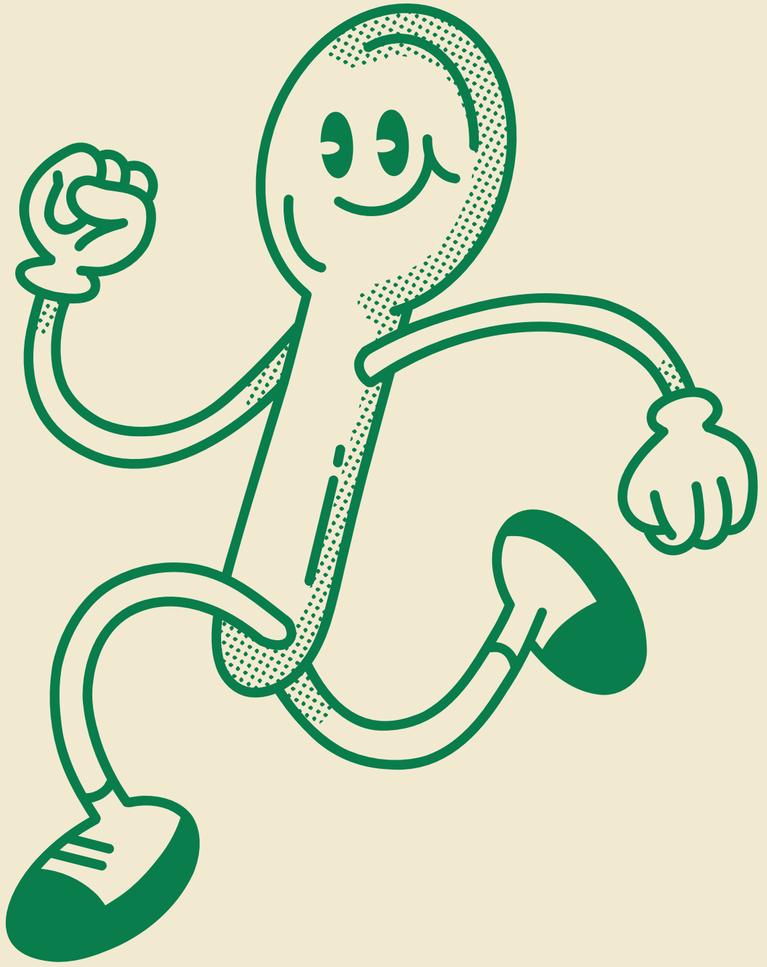

Recibir órdenes
Validar órdenes
Entregar órdenes
Chef
Poner órdenes
Ver estado de orden
Consumidor
include
Agregar Ingrediente
include
Actualizar inventario
Administrador
Eliminar Ingrediente
Pizzeria



14/08/2025

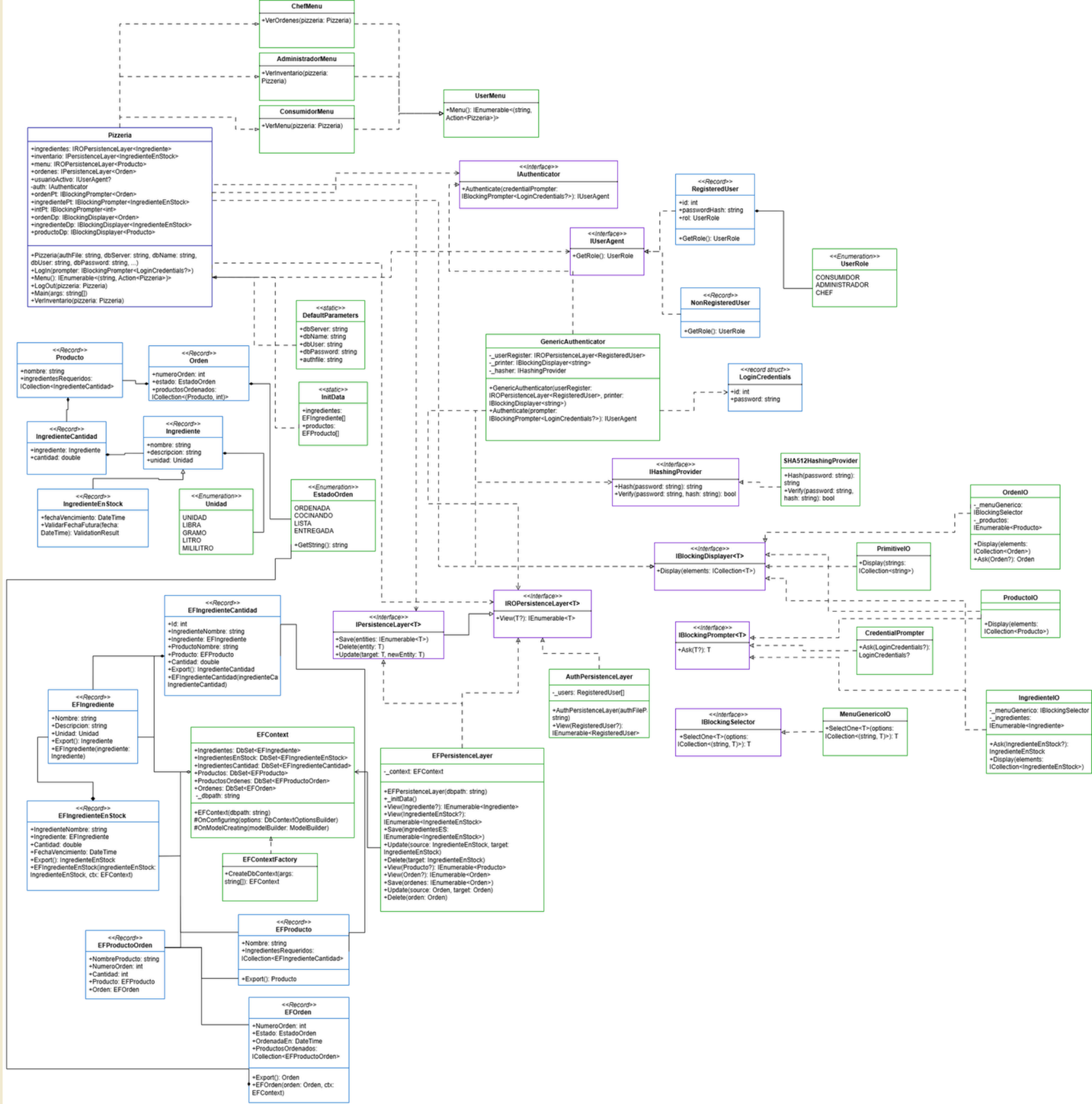
LUEGO DE SOLID...

y una ardua reconstrucción



NUEVO DIAGRAMA

- *Recurso*



S - Responsabilidad Simple

```
public interface IHashingProvider
{
    2 referencias
    public string Hash(string password);
    2 referencias
    public bool Verify(string password, string hash);
}

1 referencia
public class SHA512HashingProvider : IHashingProvider
{
    2 referencias
    public string Hash(string password)
        => BitConverter.ToString(SHA512.Create()
                    .ComputeHash(Encoding.UTF8.GetBytes(password)))
                    .Replace("-", "")
                    .ToLower();

    2 referencias
    public bool Verify(string password, string hash)
        => Hash(password).Equals(hash);
}
```

Única responsabilidad muy específica: hashear y verificar contraseñas

O - Principio Open/Closed

```
    3 referencias
    public class UserMenu
    {
        /// <summary>
        /// Generar opciones de menú a partir de los métodos de la clase actual.
        /// </summary>
        /// <returns>Lista de opciones para el menú actual.</returns>
#pragma warning disable CS8600
#pragma warning disable CS8602
        3 referencias
        public IEnumerable<(string, Action<Pizzeria>)> Menu()
            => this.GetType()
                .GetMethods(BindingFlags.Public | BindingFlags.Static)
                .Where(method => Attribute.IsDefined(
                    method, typeof(MenuOptionAttribute)
                ))
                .Select(method => (
                    ((MenuOptionAttribute)
                     method.GetCustomAttribute(typeof(MenuOptionAttribute))
                    ).label,
                    (Action<Pizzeria>)
                        method.CreateDelegate(typeof(Action<Pizzeria>))
                ));
#pragma warning restore CS8600
    }
```

Cerrada para modificación

pero abierta para extensión

L - Sustitución de Liskov

```
5 referencias
public interface IUserAgent
{
    /// <summary>
    /// Obtener el rol del usuario.
    /// </summary>
    /// returns rol del usuario.
    3 referencias
    public UserRole GetRole();
}

/// <summary>
/// Usuario registrado (con privilegios especiales) dentro del
/// sistema.
/// </summary>
9 referencias
public record class RegisteredUser(int id, string passwordHash, UserRole rol) : IUserAgent
{
    2 referencias
    public UserRole GetRole() => this.rol;
}

/// <summary>
/// Usuario no registrado (consumidor) dentro del sistema.
/// </summary>
1 referencia
public record class NonRegisteredUser: IUserAgent
{
    2 referencias
    public UserRole GetRole() => UserRole.CONSUMIDOR;
}
```

- *Ambas implementaciones pueden ser sustituidas por IUserAgent sin alterar el comportamiento del programa*
- *Ambas cumplen el contrato*

I - Segregación de Interfaces

```
9 referencias
public interface IROPersistenceLayer<T>
{
    11 referencias
    public IEnumerable<T> View(T? _);
}

/// <summary>
/// Representa cualquier implementación de una capa de persistencia
/// para el tipo de dato T
/// </summary>
4 referencias
public interface IPersistenceLayer<T> : IROPersistenceLayer<T>
{
    2 referencias
    public void Save(IEnumerable<T> entities);
    2 referencias
    public void Delete(T entity);
    2 referencias
    public void Update(T target, T newEntity);
}
```

- *Las interfaces están segregadas según las necesidades del cliente*
- *Evita que clientes de solo-lectura dependan de métodos que no necesita*

D - Inversión de Dependencia

```
public class GenericAuthenticator: IAuthenticator
{
    private IROPersistenceLayer<RegisteredUser> _userRegister;
    private IBlockingDisplayer<string> _printer;
    private IHashingProvider _hasher;

    1 referencia
    public GenericAuthenticator(
        IROPersistenceLayer<RegisteredUser> userRegister,
        IBlockingDisplayer<string> printer
    )
    {
        this._userRegister = userRegister;
        this._printer = printer;
```

La clase depende de abstracciones en lugar de implementaciones concretas

Bonus

```
public interface IBlockingSelector
{
    /// <summary> Preguntarle al usuario cuál de las opciones seleccionar
    6 referencias
    public T SelectOne<T>(ICollection<(string label, T option)> options);
}

/// <summary> Representa cualquier objeto capaz de mostrarle al usuario una cole .
14 referencias
public interface IBlockingDisplayer<T>
{
    /// <summary> Mostrar al usuario los elementos en elements.
    8 referencias
    public void Display(ICollection<T> elements);
}

/// <summary> Representa cualquier objeto capaz de preguntarle al usuario por un .
13 referencias
public interface IBlockingPrompter<T>
{
    /// <summary> Mostrar al usuario los elementos en elements.
    7 referencias
    public T Ask(T? _);
    //cambio de interfaz por problema de ambigüedad con el parámetro genérico T
}
```

```
public Pizzeria
(
    string authFile,
    string dbServer,
    string dbName,
    string dbUser,
    string dbPassword,
    IBlockingSelector selector,
    IBlockingDisplayer<Producto> productoDp,
    IBlockingDisplayer<IngredienteEnStock> ingredienteDp,
    IBlockingDisplayer<Orden> ordenDp,
    IBlockingDisplayer<string> stringDp,
    Func<IEnumerable<Ingrediente>, IBlockingPrompter<IngredienteEnStock>> ingredientePtGen,
    Func<IEnumerable<Producto>, IBlockingPrompter<Orden>> ordenPtGen
)
```

GRACIAS

- **Repositorio**
  - **Designing the persistence layer**