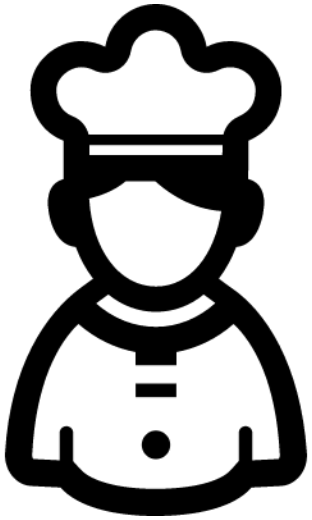
Arquitectura de Aplicaciones

- Repasando conceptos
- Acoplamiento y cohesión
- Sobre los Patrones de Diseño
- Arquitectura de software
- Modelo MVC
- Modelo MVC en Asp Net Core

Responsabilidades de cada parte

Ejemplo de un bar

Cocinero

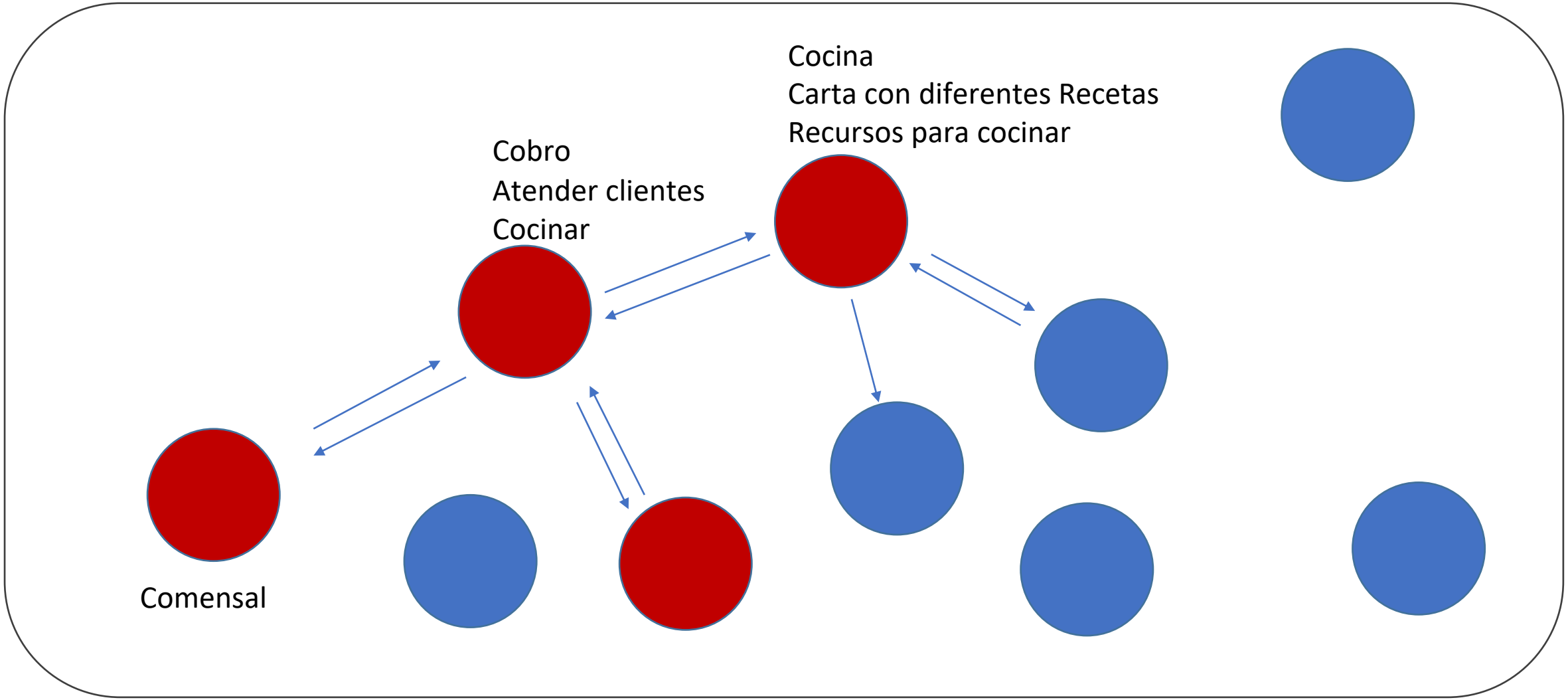


Mozo

Cliente

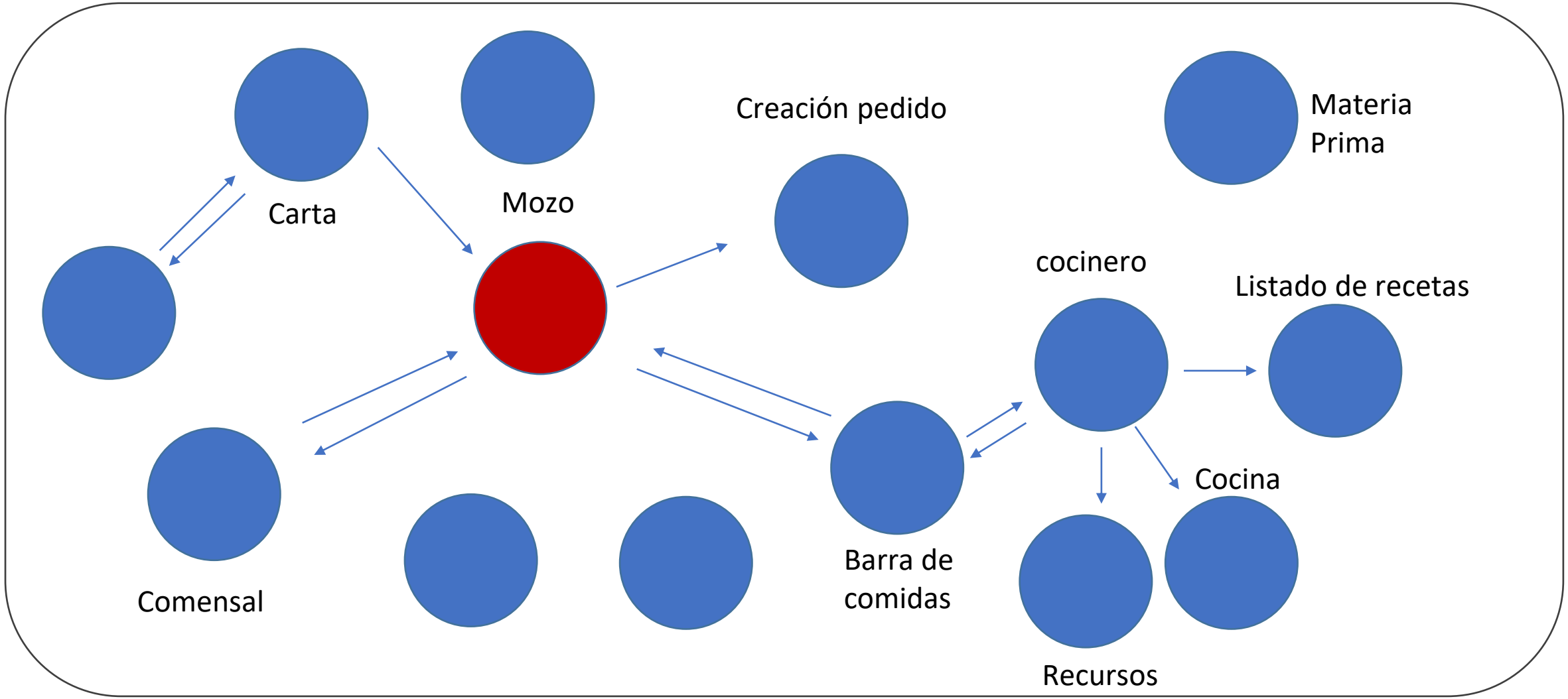
Aplicación: Bar

Todas las responsabilidades sobre El MOZO



Aplicación

Distribuyendo responsabilidades



Arquitectura de una aplicación

Acoplamiento

Según normas ISO:

> El **acoplamiento** es la forma y nivel de ***interdependencia*** entre módulos de software. Una medida de qué tan cercanamente conectados están dos rutinas o módulos de software.

ISO/IEC/IEEE 24765:2010 Systems and software engineering — Vocabulary

El acoplamiento se produce cada vez que una clase accede a un dato o función que pertenece a otra clase.

Arquitectura de una aplicación

Clase Altamente Cohesión

Una clase altamente cohesiva en programación es aquella que tiene una sola responsabilidad bien definida y realiza un conjunto específico de tareas relacionadas.

```
public class Calculadora {

    public int sumar(int numero1, int numero2) {
        return numero1 + numero2;
    }

    public int restar(int numero1, int numero2) {
        return numero1 - numero2;
    }

    public int multiplicar(int numero1, int numero2) {
        return numero1 * numero2;
    }

    public int dividir(int dividendo, int divisor) {
        if (divisor == 0) {
            throw new IllegalArgumentException("No se puede dividir por cero.");
        }
        return dividendo / divisor;
    }
}
```

Arquitectura de una aplicación

Clase altamente acoplada

Una clase altamente acoplada en programación es aquella que tiene muchas dependencias con otras clases o módulos, lo que significa que está fuertemente interconectada con otras partes del sistema. Esto puede hacer que la clase sea menos flexible y más difícil de mantener

```
public class Pedido {
    private Cliente cliente;
    private Producto producto;
    private Factura factura;

    public Pedido(Cliente cliente, Producto producto) {
        this.cliente = cliente;
        this.producto = producto;
    }

    public void procesarPedido() {
        // Realizar lógica para procesar el pedido
        // Esto puede involucrar interacciones con el cliente, producto y la
generación de una factura
    }

    public void generarFactura() {
        factura = new Factura(cliente, producto);
        factura.emitir();
    }
}
```

Un problema con esta alta dependencia es que si se realiza un cambio en una de las clases relacionadas (por ejemplo, en la forma en que se crea una factura), podría requerir modificaciones en la clase Pedido

Arquitectura de una aplicación

Clase altamente acoplada

Para reducir el acoplamiento, es una buena práctica separar las responsabilidades y reducir las dependencias innecesarias entre las clases.

```
public class Pedido {
    private Cliente cliente;
    private Producto producto;

    public Pedido(Cliente cliente, Producto producto) {
        this.cliente = cliente;
        this.producto = producto;
    }

    public void procesarPedido() {
        // Realizar lógica para procesar el pedido
    }
}

public class Facturador {
    public Factura EmitirFactura(Pedido pedido) {
                return new Factura(Pedido pedido);
     }
}
```

Arquitectura de una aplicación

Clase altamente acoplada

Una clase altamente acoplada en programación es aquella que tiene muchas dependencias con otras clases o módulos, lo que significa que está fuertemente interconectada con otras partes del sistema. Esto puede hacer que la clase sea menos flexible y más difícil de mantener

```
public class Pedido {
    private Cliente cliente;
    private Producto producto;
    private Factura factura;

    public Pedido(Cliente cliente, Producto producto) {
        this.cliente = cliente;
        this.producto = producto;
    }

    public void procesarPedido() {
        // Realizar lógica para procesar el pedido
        // Esto puede involucrar interacciones con el cliente, producto y la
generación de una factura
    }

    public void generarFactura() {
        factura = new Factura(cliente, producto);
        factura.emitir();
    }
}
```

Arquitectura de una aplicación

Acoplamiento - Ventajas

El bajo acoplamiento permite:

- Mejorar la mantenibilidad de las unidades de software.
- Aumentar la reutilización de las unidades de software.
- Minimiza el riesgo de tener que cambiar múltiples unidades de software cuando se debe alterar una.

Arquitectura de una aplicación

Acoplamiento

Se busca siempre bajo acoplamiento y alta cohesión

Cuanto menos dependiente sean las partes que constituyen un sistema informático, mejor será el resultado. Sin embargo, es imposible un desacoplamiento total de las unidades.

El bajo acoplamiento permite:

- Mejorar la mantenibilidad de las unidades de software.
- Aumentar la reutilización de las unidades de software.
- Minimiza el riesgo de tener que cambiar múltiples unidades de software cuando se debe alterar una.

Patrones de diseño

Los patrones de diseño son **soluciones para problemas típicos y recurrentes** que nos podemos encontrar a la hora de desarrollar una aplicación.

Patrones de diseño

Los patrones de diseño pretenden:

- **Proporcionar** catálogos **de elementos reusables** en el diseño de sistemas software.
- **Evitar** la reiteración en la **búsqueda de soluciones a problemas ya conocidos** y solucionados anteriormente.
- **Formalizar un vocabulario** común entre diseñadores.
- **Estandarizar** el modo en que se realiza el diseño.
- **Facilitar el aprendizaje de las nuevas generaciones** de diseñadores condensando conocimiento ya existente.

Patrones de diseño

Clasificación según la escala o nivel de abstracción:

- **Patrones de arquitectura**: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software.
- **Patrones de diseño**: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.
- **Dialectos**: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

Arquitectura de software

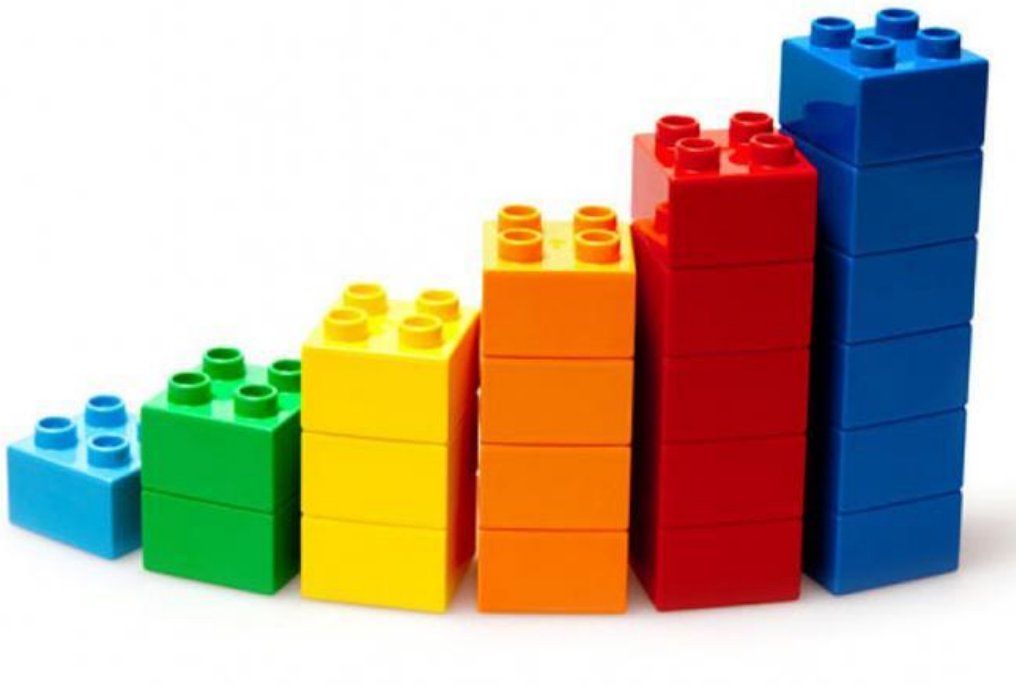
¿Qué es un patrón arquitectónico?

*Un **patrón arquitectónico** es una solución general y reutilizable a un problema común en la arquitectura de software dentro de un contexto dado.*

Arquitectura de software

¿Por que es necesaria?

¿Cómo se comporta nuestro software cuando escala o crece?



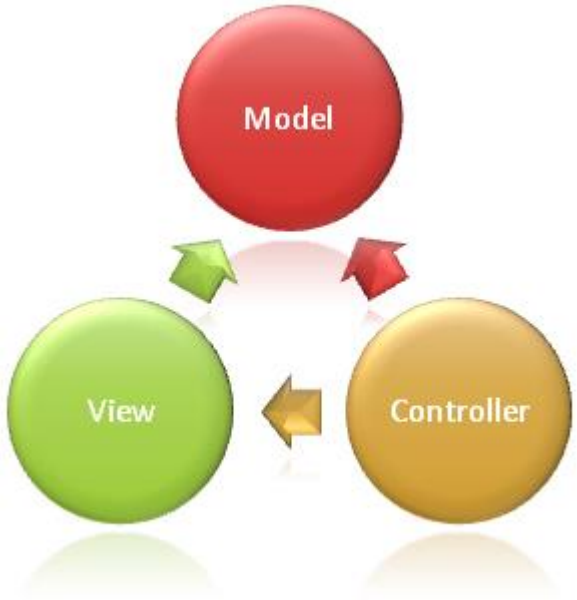
El Patrón Modelo-Vista-Controlador

Este patrón, también conocido como patrón MVC, divide una aplicación interactiva en 3 partes, como

1.modelo — contiene la funcionalidad y los datos básicos

2.vista : muestra la información al usuario (se puede definir más de una vista)

3.controlador : maneja la entrada del usuario



Partes de una web

Front End

- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation

Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup