grupoGHL

GHL Innovación & Tecnología

Modelos en el Mundo Real: Experiencias aplicando MDE en la industria

Agenda

1. Grupo GHL
2. Proyecto Colciencias
3. Estrategia
4. Lenguajes
5. Generador
6. Resultados
7. Conclusiones

grupoGHL

GHL Hoteles

GHL Investments

GHL Innovación & Tecnología

GHL Centros de Convenciones

GHL Catering

GHL Club Vacacional

GHL Restaurantes

GHL Iniciativa Social

- 11 Países
- 61 Operaciones
- 6824 Habitaciones

GHL Hoteles

Centro de Convenciones Cartagena de Indias

Expofuturo (2017) - Pereira

GHL Centros de Convenciones

- 10.000 socios activos en el sistema.
- 35.000 usuarios

GHL Club Vacacional

- Alianza con los Hermanos Rausch
- 6 Restaurantes

GHL Restaurantes

- Mas de 20 operaciones.
- 9.000 servicios diarios producidos en promedio.

GHL Catering

- Fundación Palmarito
- Parque Wisirare
- Fundación GHL

GHL Iniciativa Social

- Servicios de Tecnología a mas de 60 operaciones
- Marketing Digital
- Desarrollo de Software

GHL Innovación & Tecnología

GHL Innovación & Tecnología

Proyecto

GHL Innovación & Tecnología

COLCIENCIAS
Ciencia, Tecnología e Innovación

PROYECTO DE INSERCIÓN LABORAL

IMPLANTACION DE METODOLOGIAS DE DESARROLLO DE SOFTWARE DIRIGIDO POR MODELOS, EN UN DOMINIO ESPECIFICO, CON EL OBJETIVO DE INCREMENTAR LA PRODUCTIVIDAD EN EQUIPOS DE DESARROLLO DE SOFTWARE REDUCIDOS.

Un problema de comunicación

Un problema de comunicación

Operaciones

Un problema de comunicación

Operaciones

Tecnología

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

Finanzas

Cuenta

Activo

Caja

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

XML

Finanzas

Cuenta

Activo

Caja

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

XML

Finanzas

Cuenta

Activo

Caja

Mayorista

OTA

GDS

Mercadeo

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

XML

C#

Finanzas

Cuenta

Activo

Caja

Mayorista

OTA

GDS

Mercadeo

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

XML

SQL

C#

Finanzas

Cuenta

Activo

Caja

Dotación

Proyecto

Inversionista

Desarrollo

Mayorista

OTA

GDS

Mercadeo

Un problema de comunicación

Operaciones

Hotel

Huésped

Servicio

Tecnología

HTML

XML

SQL

C#

Finanzas

Cuenta

Activo

Caja

Proyecto

Dotación

Inversionista

Desarrollo

Mayorista

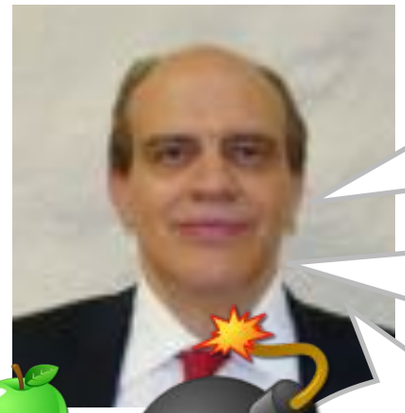
OTA

GDS

Mercadeo

Un problema de comunicación

Operaciones



Hotel

Huésped

Servicio

Tecnología

HTML

XML

SQL

C#

Finanzas

Cuenta

Activo

Caja

Proyecto

Dotación

Inversionista

Desarrollo

Mayorista

OTA

GDS

Mercadeo

MDE y Generación de Software

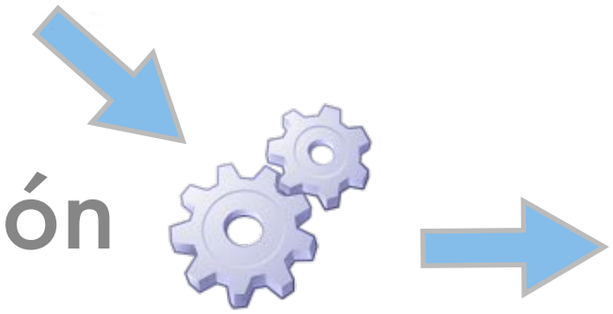
MDE y Generación de Software

Modelo especificado usando un
lenguaje de modelado de dominio especifico

MDE y Generación de Software

Modelo especificado usando un
lenguaje de modelado de dominio especifico

Transformación
Automatica



MDE y Generación de Software

Modelo especificado usando un
lenguaje de modelado de dominio especifico

Transformación
Automatica

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Tu
{
    public class Emailer : Tu.Base
    {
        Standard Constructor
        /// <summary>
        /// Send an email.
        /// We use 'SendAsync' instead of 'Send' so that we DON'T block (wait)
        /// for the sending process to complete.
        /// We handle all exceptions in this function, an EasyLanguage runtime
        /// error will not be thrown unless we encounter a major error.
        /// </summary>
        /// <param name="ReceiverEmailAddress">Email address of the intended
        /// receiver e.g. MyFriend@HisDomain.com</param>
        /// <param name="SenderEmailAddress">Your email address: e.g. Me@MyDomain.com</param>
        /// <param name="SenderMailServer">Your mail server address: e.g.
        /// mail.MyDomain.com ( Without the '@' )</param>
        /// <param name="Subject">Email subject</param>
        /// <param name="Body">Body Text</param>
        /// <returns>'0' if successfull, '-1' if unsuccessfull. You have to
        /// check this status to know if the parameters
        /// you used were accepted. However, a 'success' does not guarantee the
        /// email has been received by the recipient</returns>
        public int Send(Ev<string> ReceiverEmailAddress, Ev<string> SenderEmailAddress,
                        Ev<string> SenderMailServer, Ev<string> Subject, Ev<string> Body)
        {
            try
            {
                System.Net.Mail.SmtpClient sc = new System.Net.Mail.SmtpClient(
                                                SenderMailServer.Value.ToLower());
                System.Net.Mail.MailMessage mm = new System.Net.Mail.MailMessage(
                                                SenderEmailAddress.Value.ToLower(),
                                                ReceiverEmailAddress.Value.ToLower(),
                                                Subject.Value, Body.Value);
                sc.SendAsync(mm, null);
                return 0;
            }
            catch
            {
            }
            return -1;
        }
        Blocking 'Send'
    }
}
```

MDE y Generación de Software

Modelo especificado usando un
lenguaje de modelado de dominio especifico

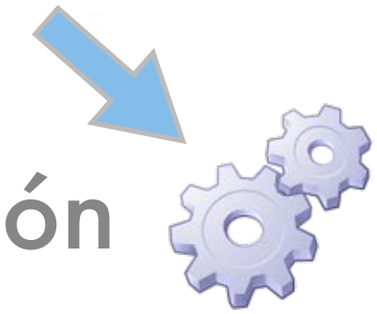
Transformación
Automatica

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Tu
{
    public class Emailer : Tu.Base
    {
        Standard Constructor
        /// <summary>
        /// Send an email.
        /// We use 'SendAsync' instead of 'Send' so that we DON'T block (wait)
        /// for the sending process to complete.
        /// We handle all exceptions in this function, an EasyLanguage runtime
        /// error will not be thrown unless we encounter a major error.
        /// </summary>
        /// <param name="ReceiverEmailAddress">Email address of the intended
        /// receiver e.g. MyFriend@HisDomain.com</param>
        /// <param name="SenderEmailAddress">Your email address: e.g. Me@MyDomain.com</param>
        /// <param name="SenderMailServer">Your mail server address: e.g.
        /// mail.MyDomain.com ( Without the '@' )</param>
        /// <param name="Subject">Email subject</param>
        /// <param name="Body">Body Text</param>
        /// <returns>'0' if successfull, '-1' if unsuccessfull. You have to
        /// check this status to know if the parameters
        /// you used were accepted. However, a 'success' does not guarantee the
        /// email has been received by the recipient</returns>
        public int Send(Ev<string> ReceiverEmailAddress, Ev<string> SenderEmailAddress,
                        Ev<string> SenderMailServer, Ev<string> Subject, Ev<string> Body)
        {
            try
            {
                System.Net.Mail.SmtpClient sc = new System.Net.Mail.SmtpClient(
                                                SenderMailServer.Value.ToLower());
                System.Net.Mail.MailMessage mm = new System.Net.Mail.MailMessage(
                                                SenderEmailAddress.Value.ToLower(),
                                                ReceiverEmailAddress.Value.ToLower(),
                                                Subject.Value, Body.Value);
                sc.SendAsync(mm, null);
                return 0;
            }
            catch
            {
            }
            return -1;
        }
        Blocking 'Send'
    }
}
```

MDE y Generación de Software

Modelo especificado usando un
lenguaje de modelado de dominio especifico

Transformación
Automatica



```
using System;
using System.Collections.Generic;
using System.Text;
namespace Tu
{
    public class Emailer : Tu.Base
    {
        Standard Constructor
        /// <summary>
        /// Send an email.
        /// We use 'SendAsync' instead of 'Send' so that we DON'T block (wait)
        /// for the sending process to complete.
        /// We handle all exceptions in this function, an EasyLanguage runtime
        /// error will not be thrown unless we encounter a major error.
        /// </summary>
        /// <param name="ReceiverEmailAddress">Email address of the intended
        /// receiver e.g. MyFriend@HisDomain.com</param>
        /// <param name="SenderEmailAddress">Your email address: e.g. Me@MyDomain.com</param>
        /// <param name="SenderMailServer">Your mail server address: e.g.
        /// mail.MyDomain.com ( Without the '@' )</param>
        /// <param name="Subject">Email subject</param>
        /// <param name="Body">Body Text</param>
        /// <returns>'0' if successfull, '-1' if unsuccessfull. You have to
        /// check this status to know if the parameters
        /// you used were accepted. However, a 'success' does not guarantee the
        /// email has been received by the recipient</returns>
        public int Send(Ev<string> ReceiverEmailAddress, Ev<string> SenderEmailAddress,
                        Ev<string> SenderMailServer, Ev<string> Subject, Ev<string> Body)
        {
            try
            {
                System.Net.Mail.SmtpClient sc = new System.Net.Mail.SmtpClient(
                                                SenderMailServer.Value.ToLower());
                System.Net.Mail.MailMessage mm = new System.Net.Mail.MailMessage(
                                                SenderEmailAddress.Value.ToLower(),
                                                ReceiverEmailAddress.Value.ToLower(),
                                                Subject.Value, Body.Value);
                sc.SendAsync(mm, null);
                return 0;
            }
            catch
            {
            }
            return -1;
        }
        Blocking 'Send'
    }
}
```

MDE y Generación de Software

Modelo especificado usando un
lenguaje de modelado de dominio especifico

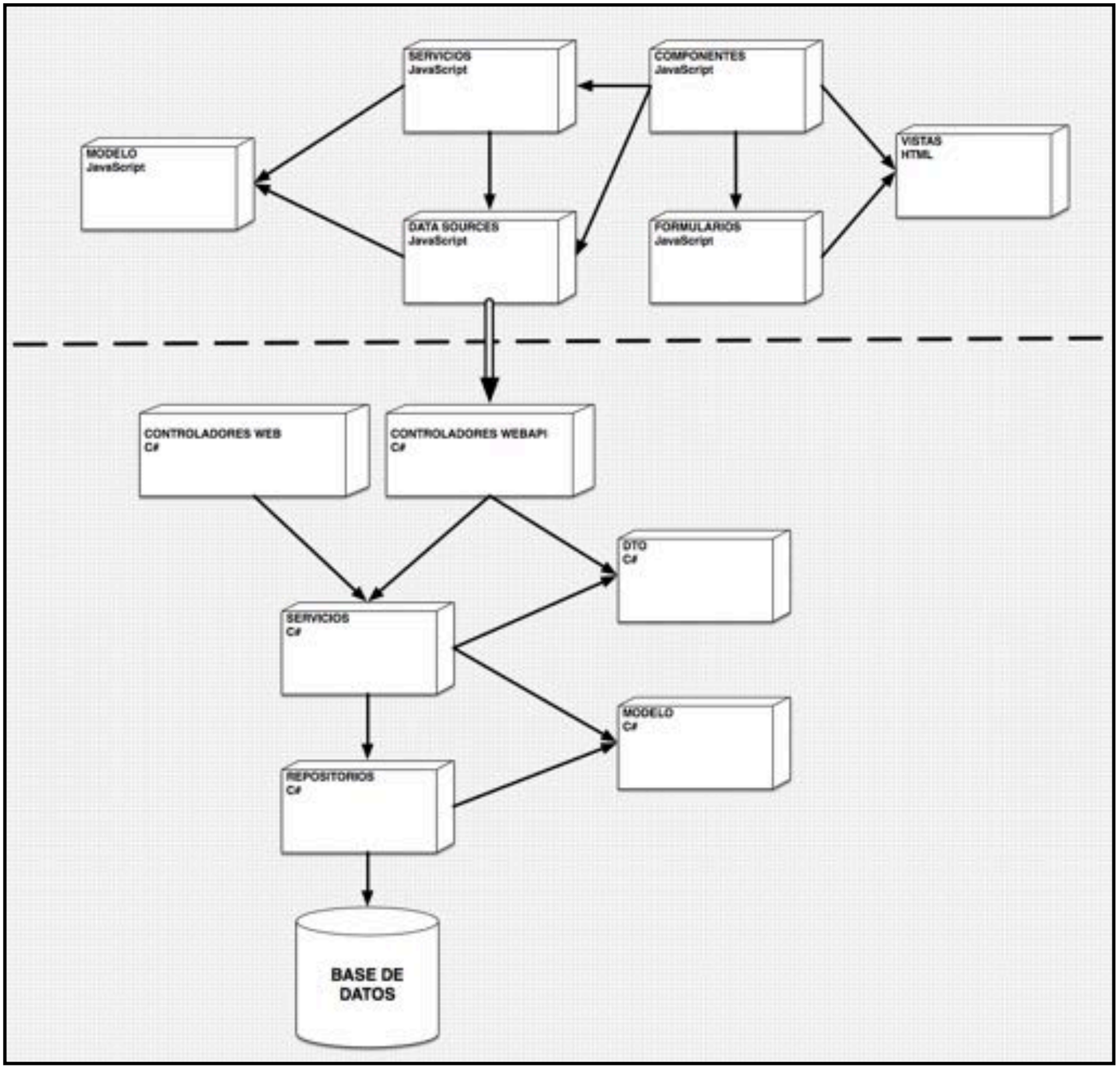
Transformación
Automatica


```
using System;
using System.Collections.Generic;
using System.Text;
namespace Tu
{
    public class Emailer : Tu.Base
    {
        Standard Constructor
        /// <summary>
        /// Send an email.
        /// We use 'SendAsync' instead of 'Send' so that we DON'T block (wait)
        /// for the sending process to complete.
        /// We handle all exceptions in this function, an EasyLanguage runtime
        /// error will not be thrown unless we encounter a major error.
        /// </summary>
        /// <param name="ReceiverEmailAddress">Email address of the intended
        /// receiver e.g. MyFriend@HisDomain.com</param>
        /// <param name="SenderEmailAddress">Your email address: e.g. Me@MyDomain.com</param>
        /// <param name="SenderMailServer">Your mail server address: e.g.
        /// mail.MyDomain.com ( Without the '@' )</param>
        /// <param name="Subject">Email subject</param>
        /// <param name="Body">Body Text</param>
        /// <returns>'0' if successfull, '-1' if unsuccessfull. You have to
        /// check this status to know if the parameters
        /// you used were accepted. However, a 'success' does not guarantee the
        /// email has been received by the recipient</returns>
        public int Send(Ev<string> ReceiverEmailAddress, Ev<string> SenderEmailAddress,
                        Ev<string> SenderMailServer, Ev<string> Subject, Ev<string> Body)
        {
            try
            {
                System.Net.Mail.SmtpClient sc = new System.Net.Mail.SmtpClient(
                                                SenderMailServer.Value.ToLower());
                System.Net.Mail.MailMessage mm = new System.Net.Mail.MailMessage(
                                                SenderEmailAddress.Value.ToLower(),
                                                ReceiverEmailAddress.Value.ToLower(),
                                                Subject.Value, Body.Value);
                sc.SendAsync(mm, null);
                return 0;
            }
            catch
            {
            }
            return -1;
        }
        Blocking 'Send'
    }
}
```

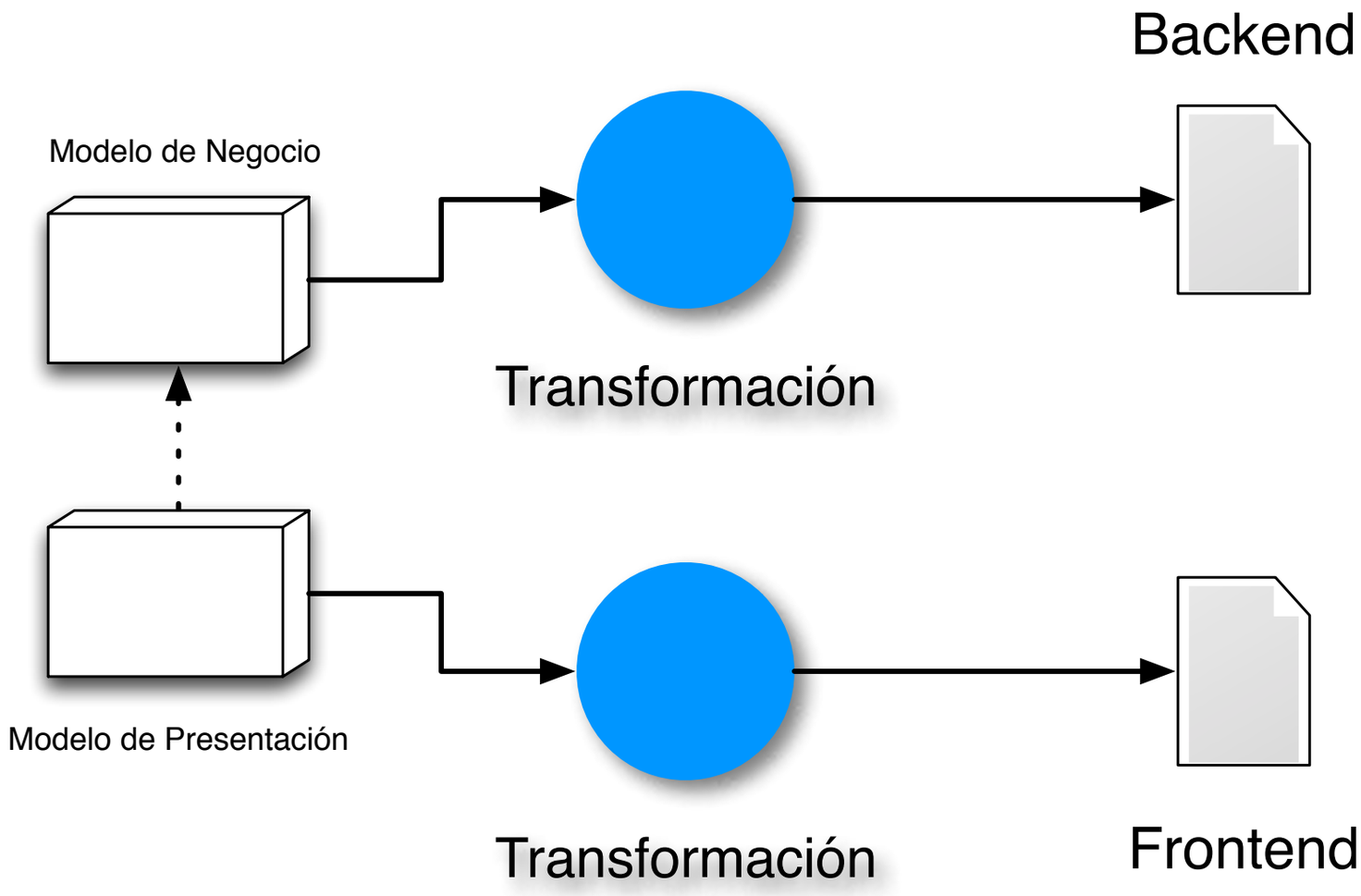
GHL Innovación & Tecnología

Estrategia

Arquitectura



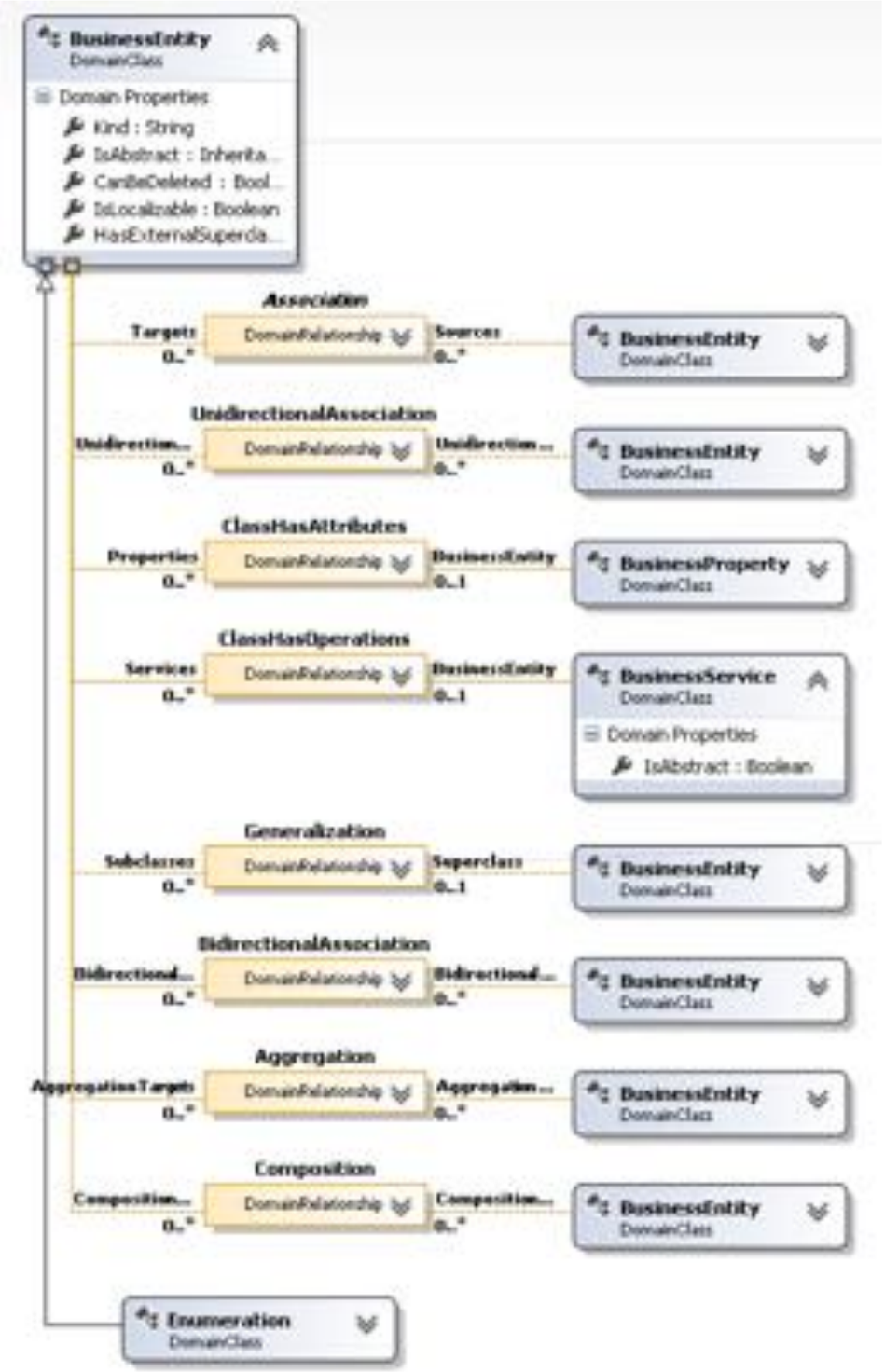
Estrategia


Backend
Modelo de Negocio
Transformación
Modelo de Presentación
Transformación
Frontend

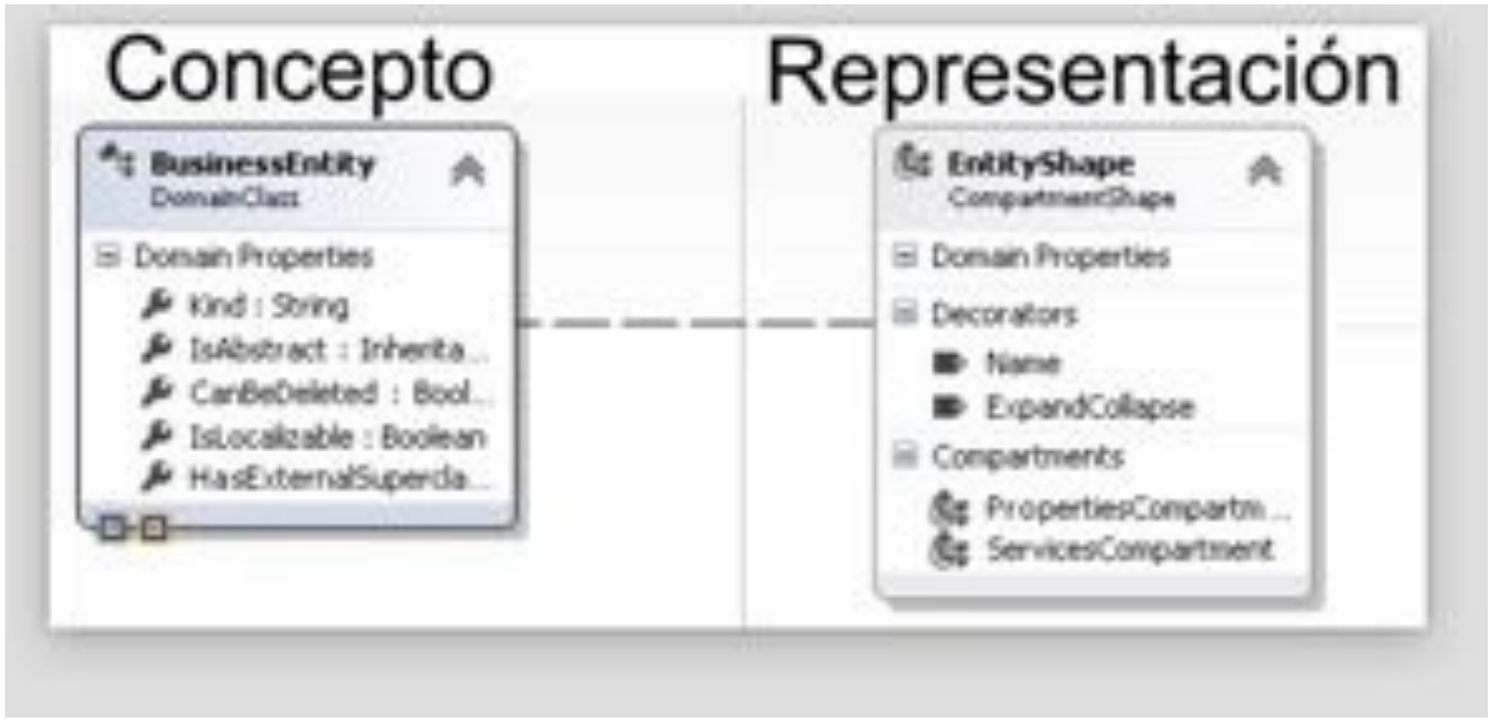
GHL Innovación & Tecnología

Lenguajes

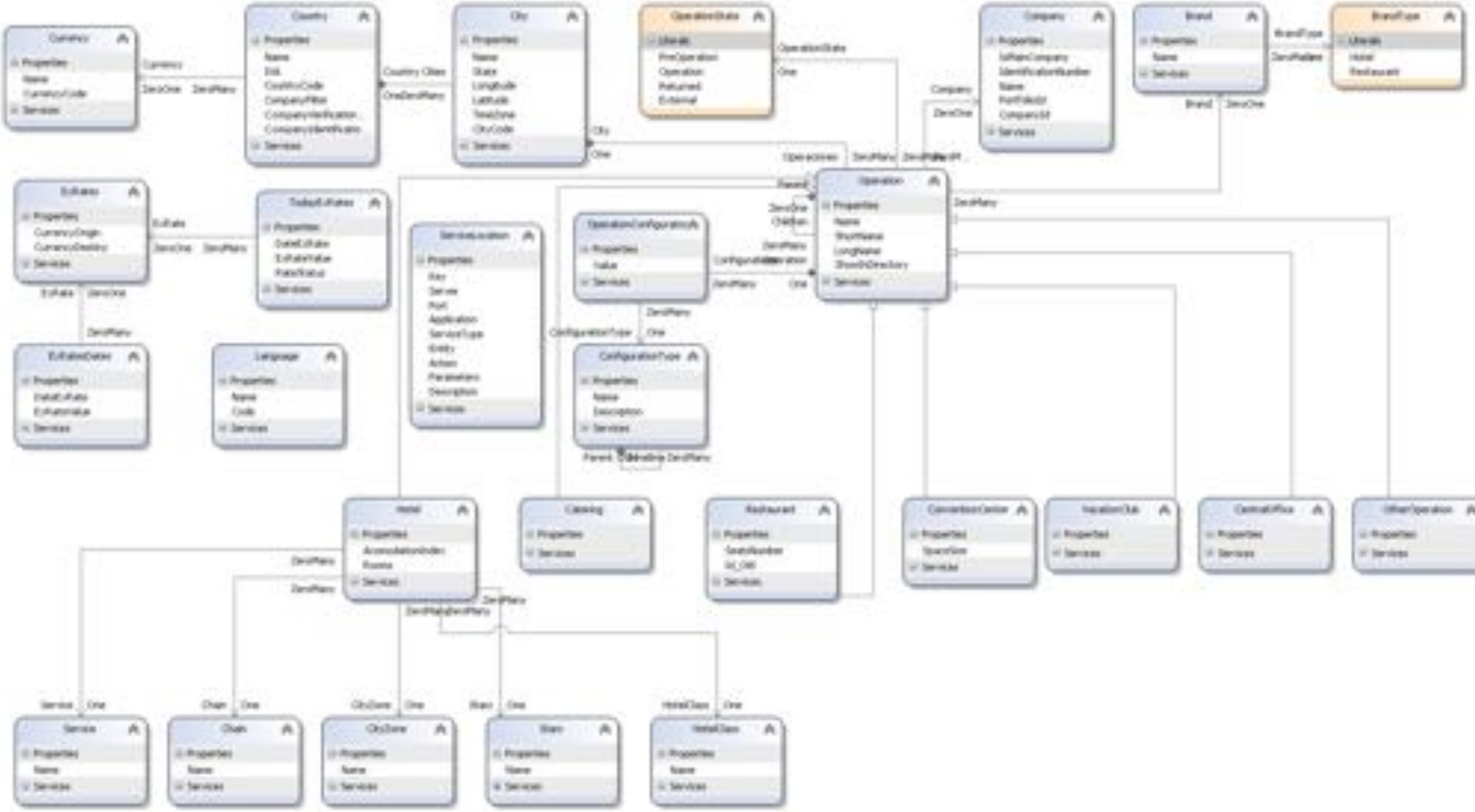
Lenguaje de Negocio



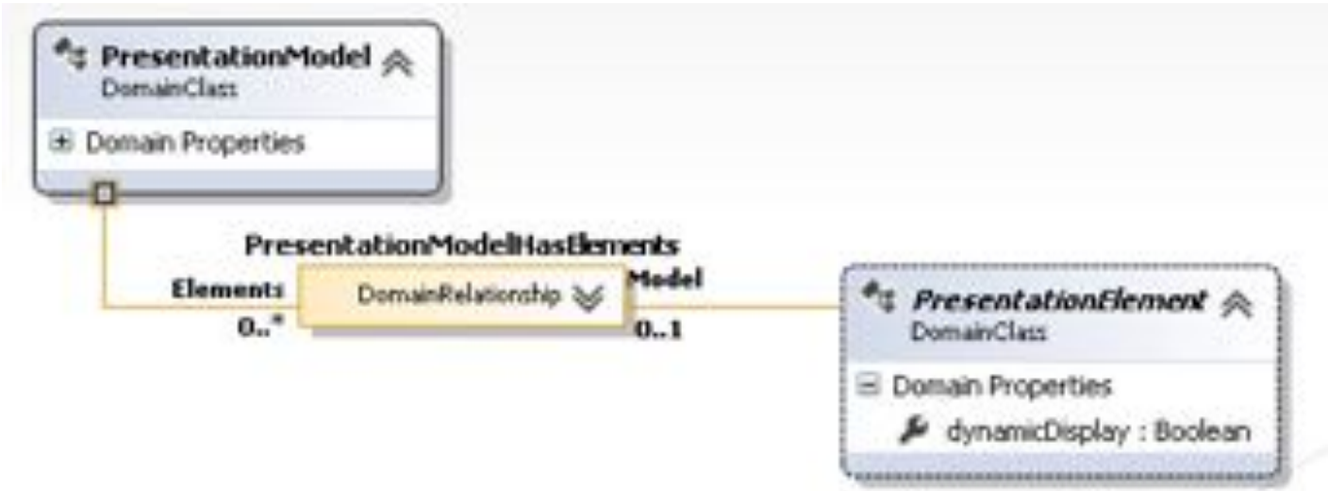
Representación Gráfica

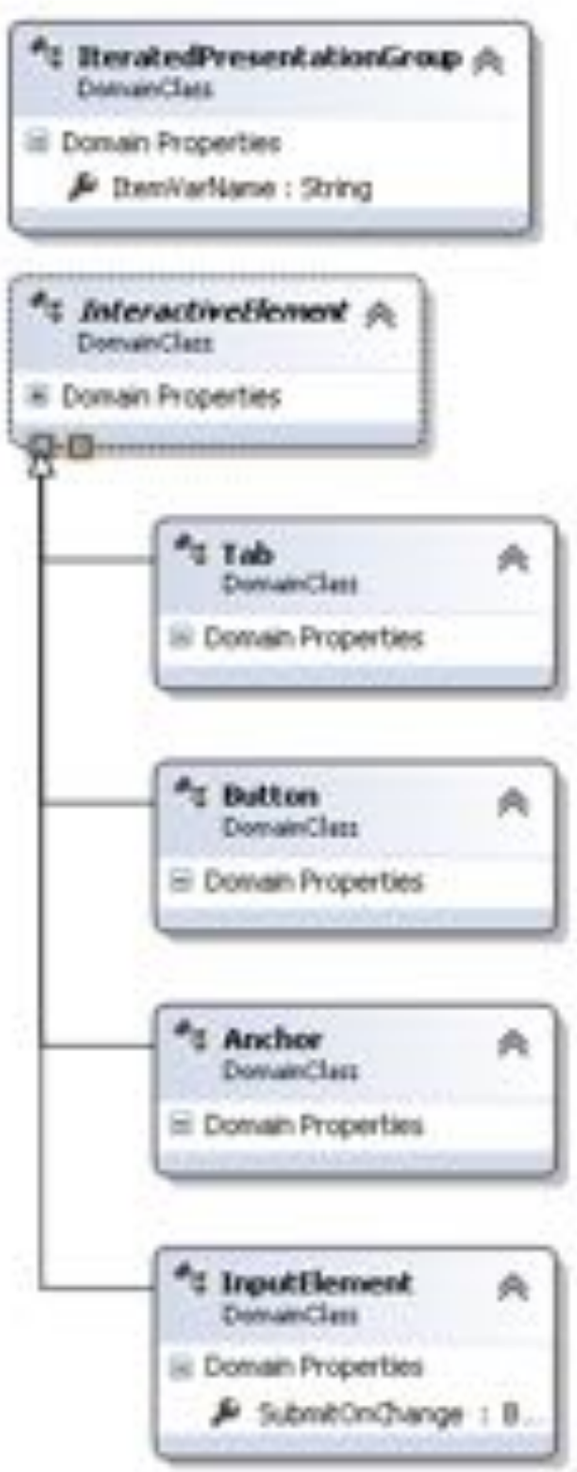


Modelo de Negocio

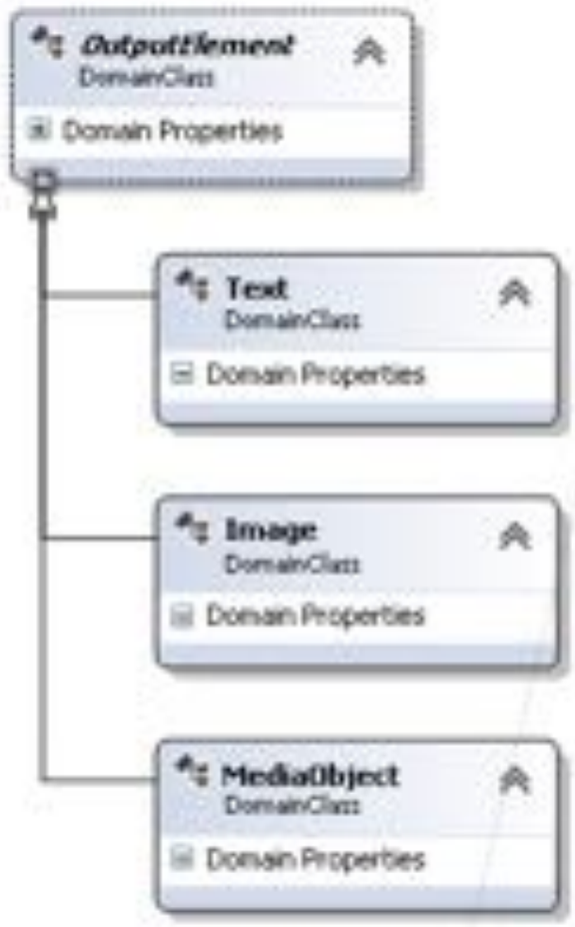


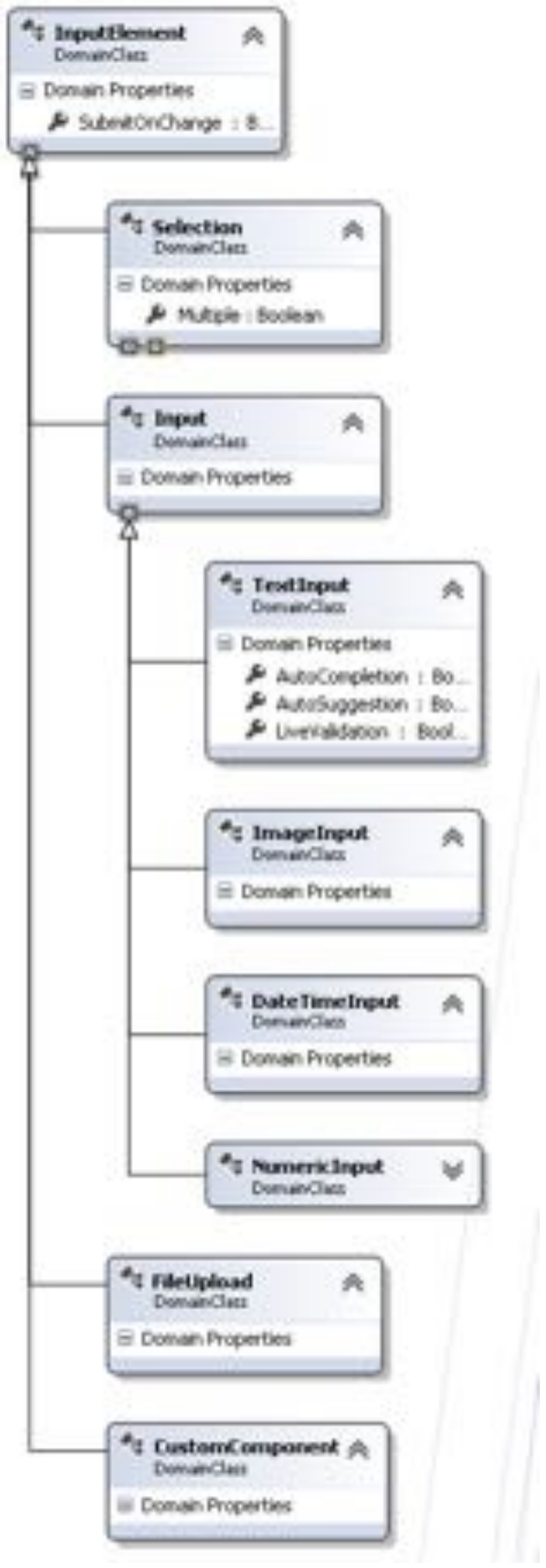
Lenguaje de Presentación



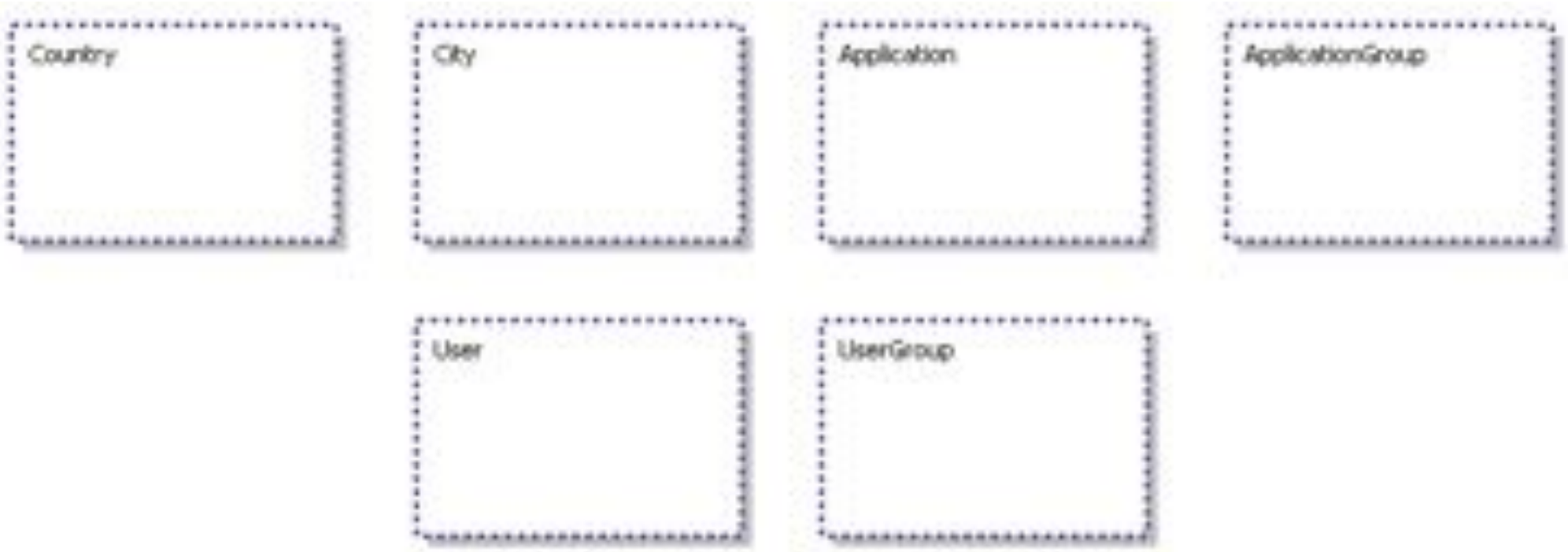


Lenguaje de Presentación

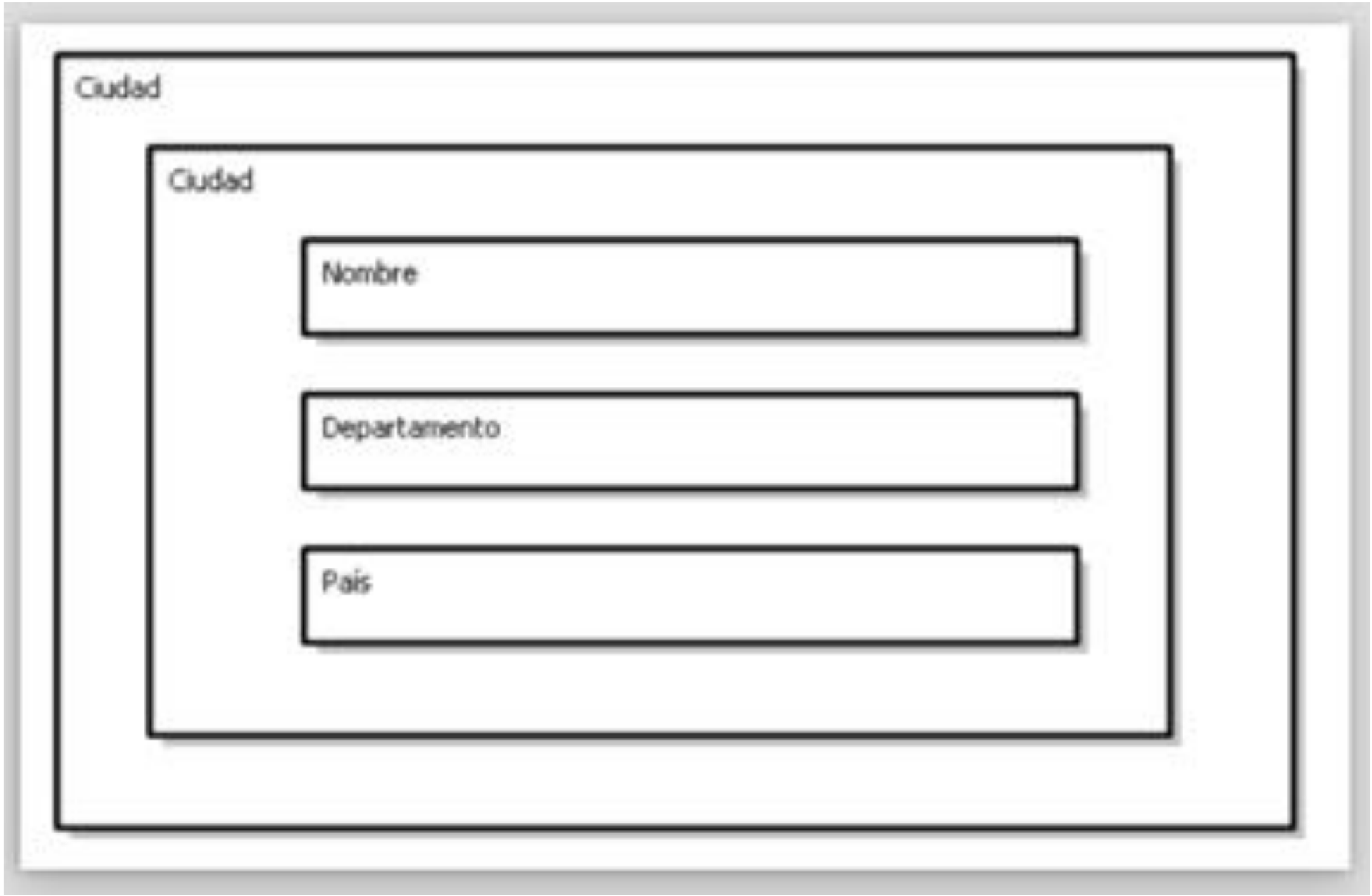




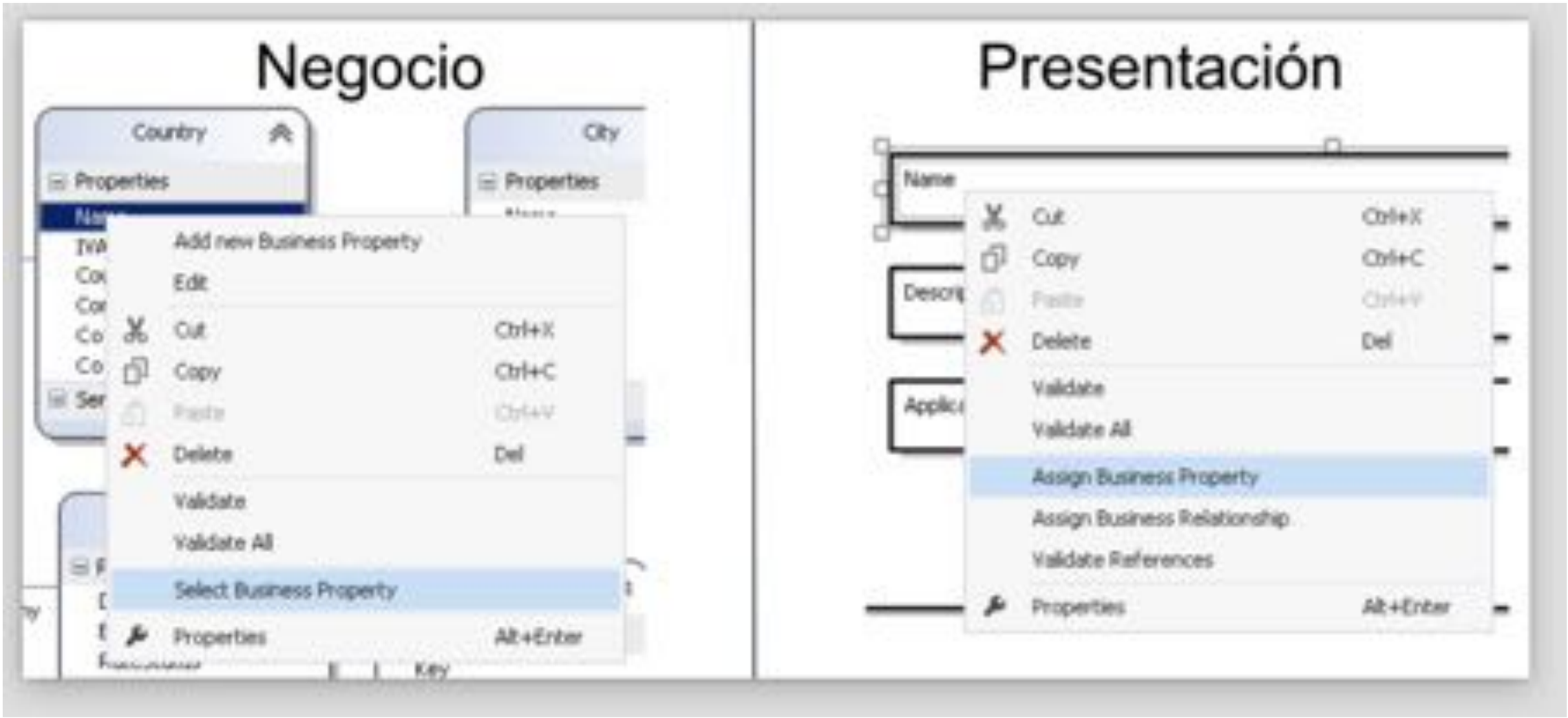
Modelo de Presentación



Modelo de Presentación



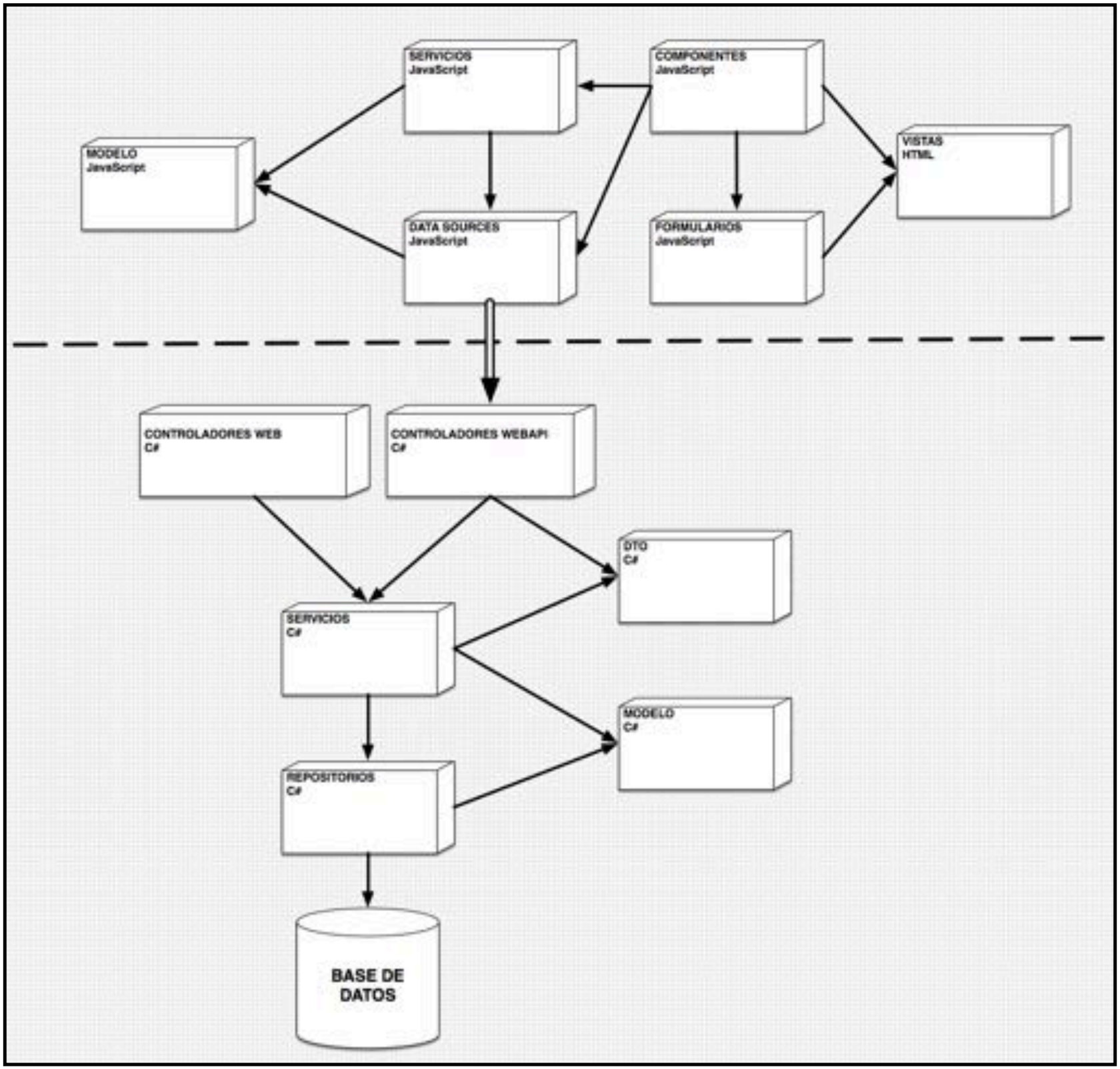
Relaciones Negocio - Presentación



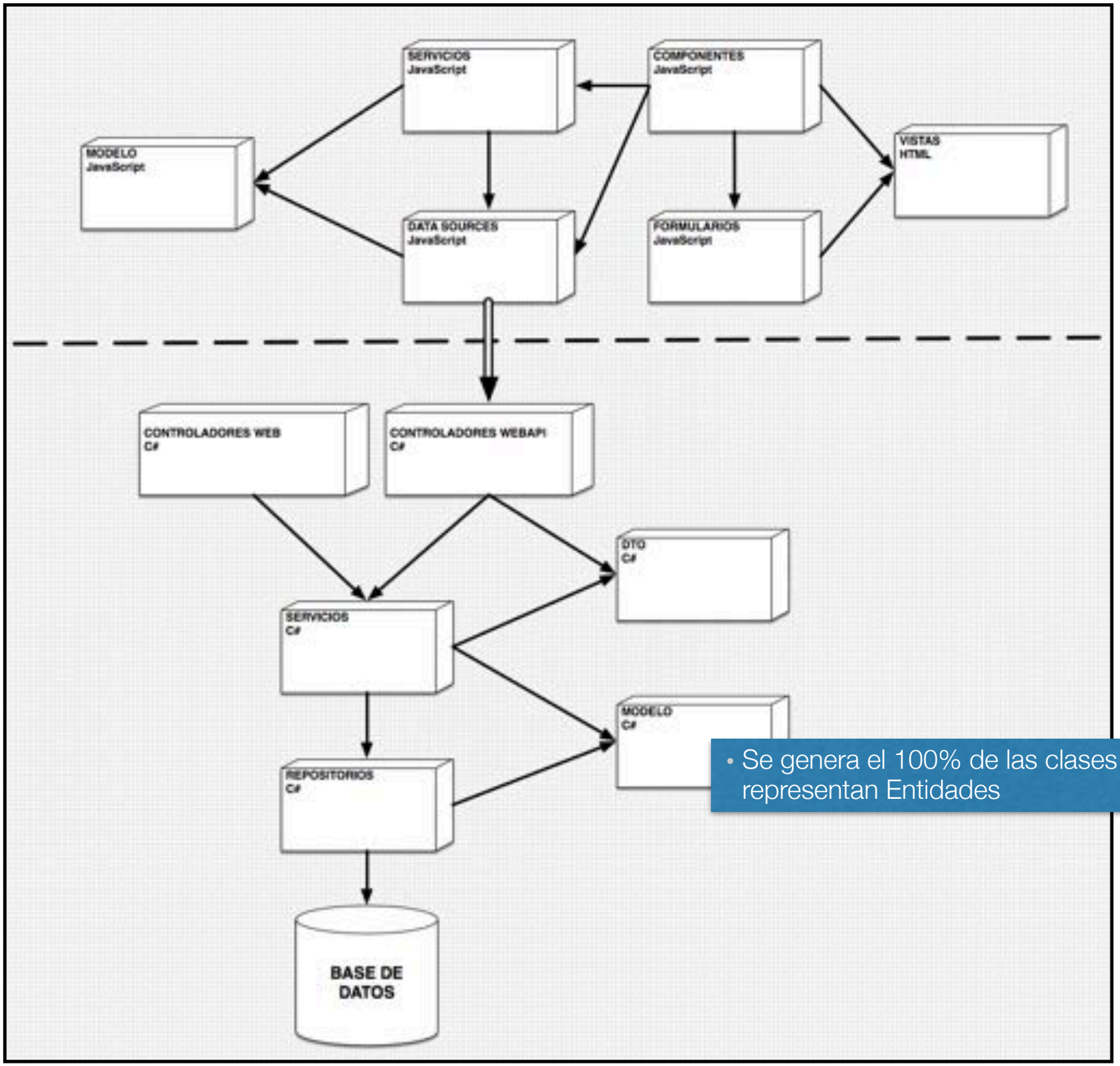
GHL Innovación & Tecnología

Generador

Generador de Código

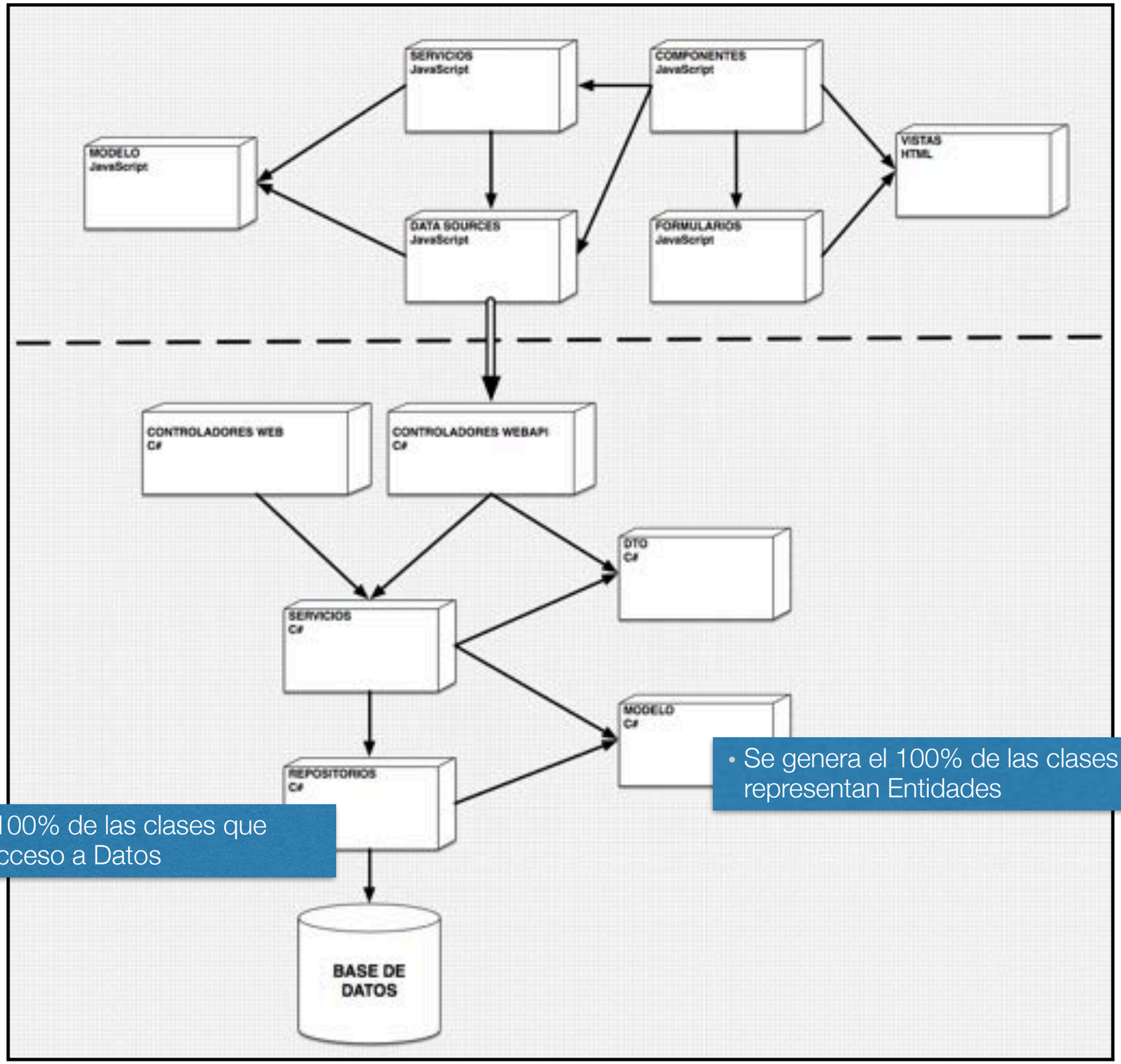


Generador de Código



- Se genera el 100% de las clases que representan Entidades

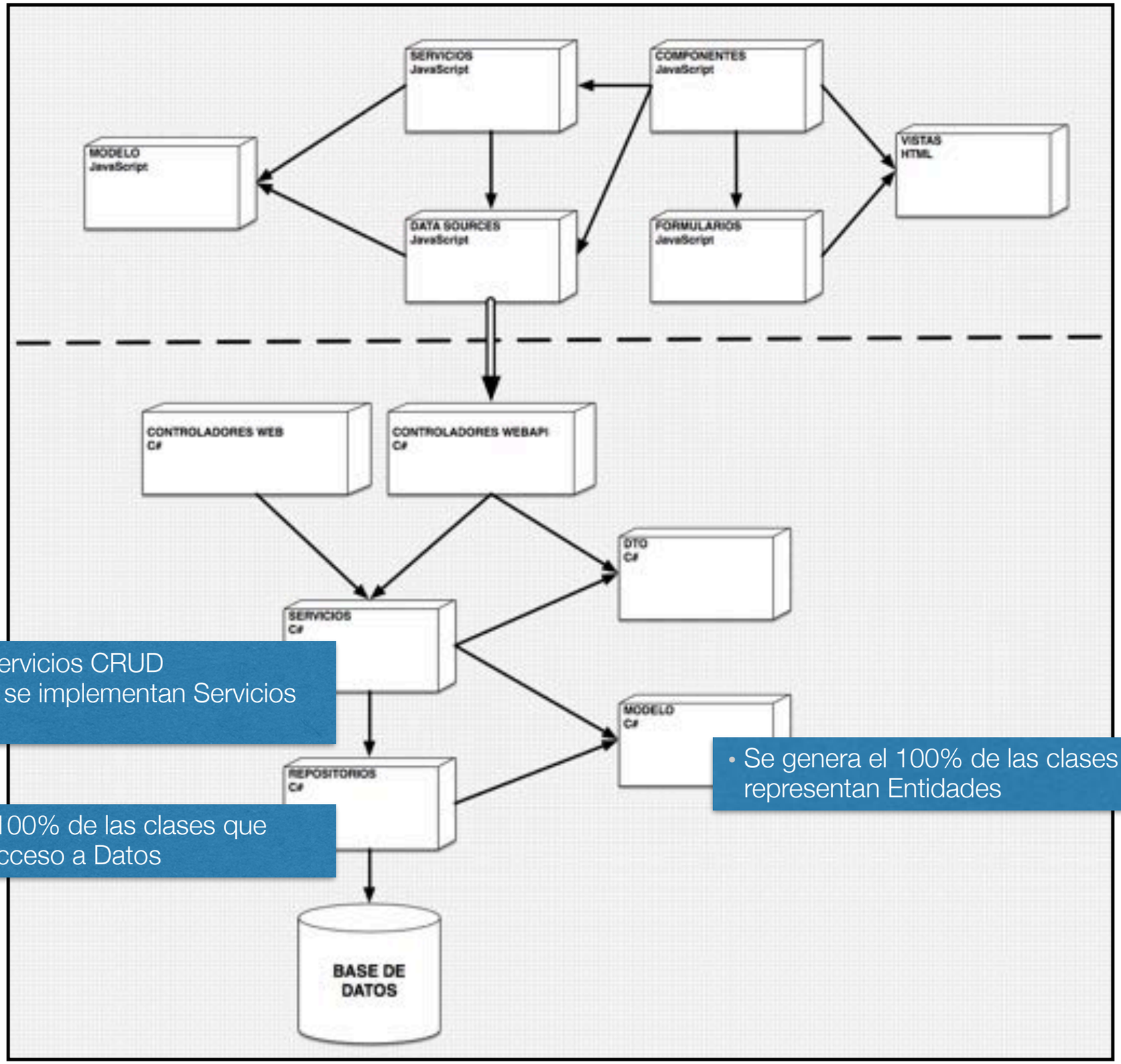
Generador de Código



- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

Generador de Código

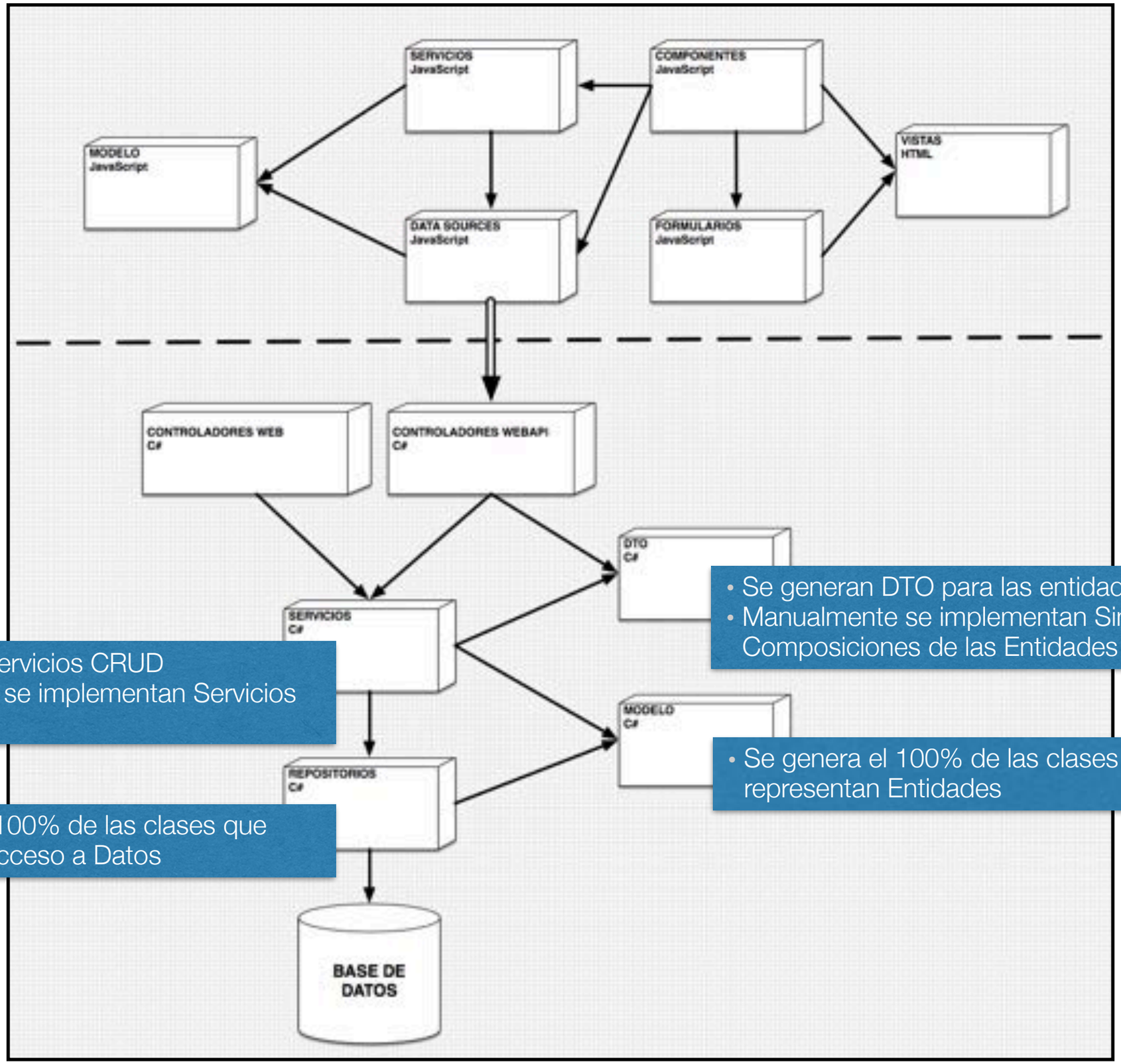


- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

Generador de Código



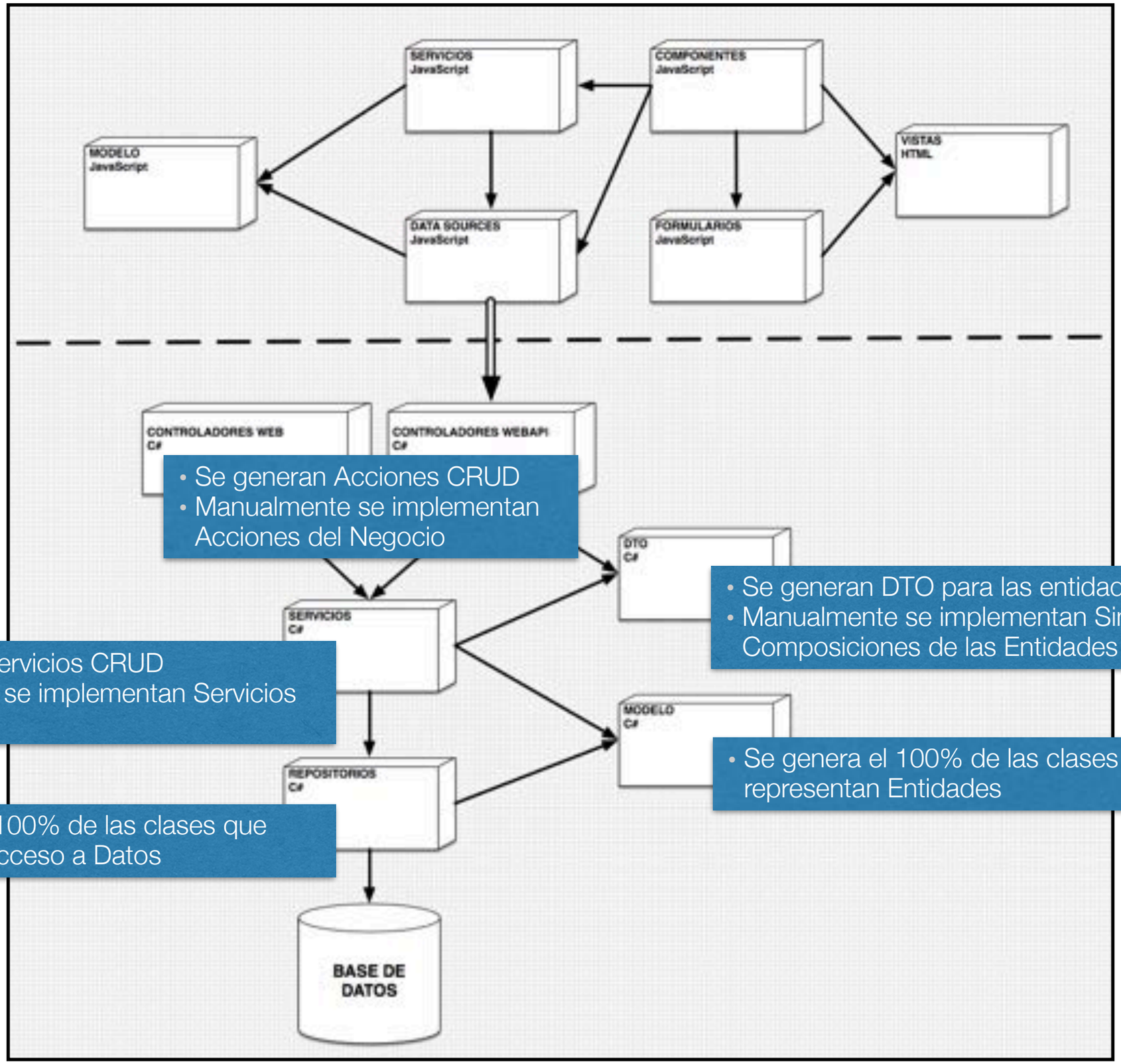
- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se generan DTO para las entidades de Negocio
- Manualmente se implementan Simplificaciones y Composiciones de las Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

- Se genera el 100% de las clases que representan Entidades

Generador de Código



- Se generan Acciones CRUD
- Manualmente se implementan Acciones del Negocio

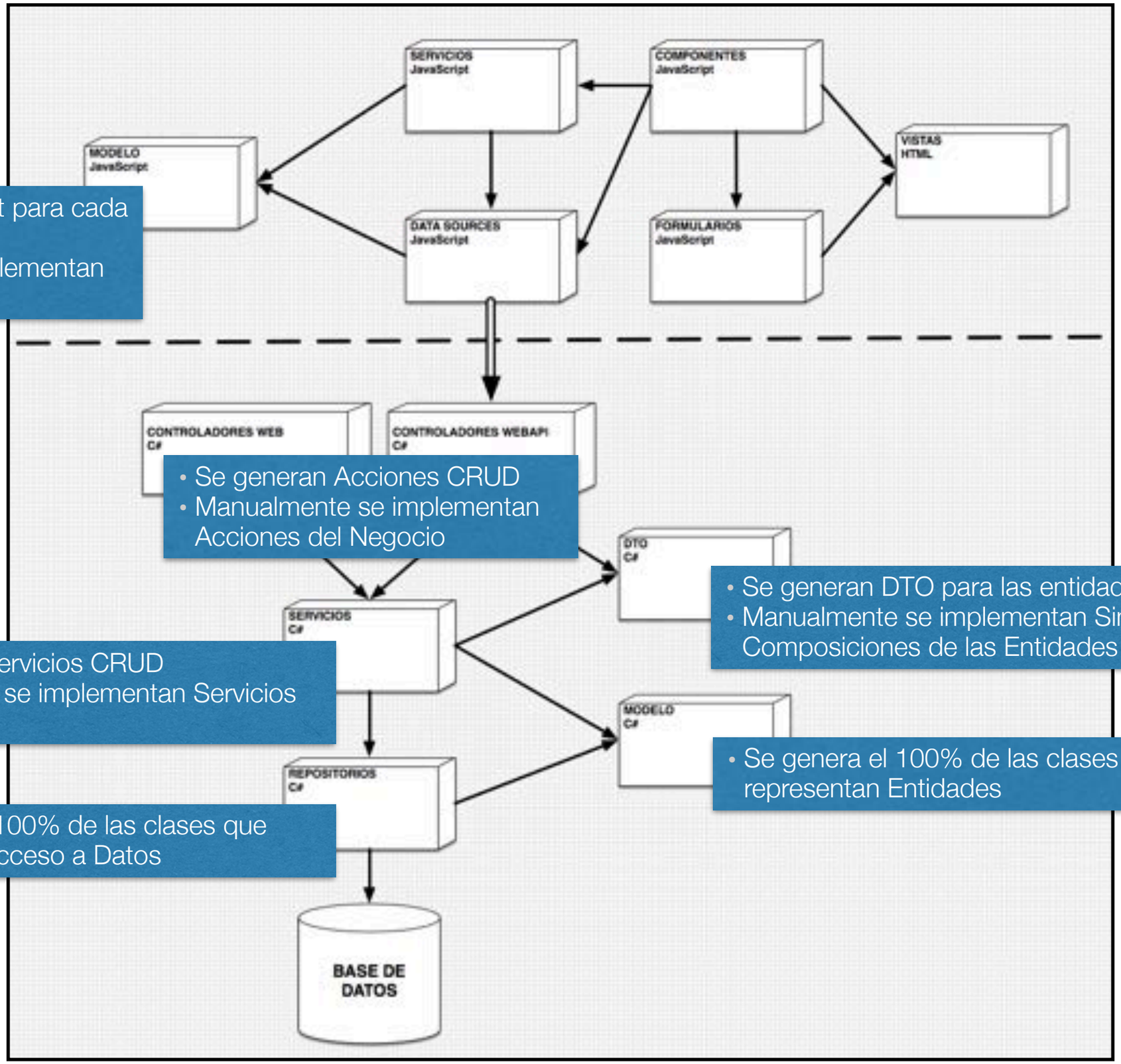
- Se generan DTO para las entidades de Negocio
- Manualmente se implementan Simplificaciones y Composiciones de las Entidades

- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

Generador de Código



- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Se generan Acciones CRUD
- Manualmente se implementan Acciones del Negocio

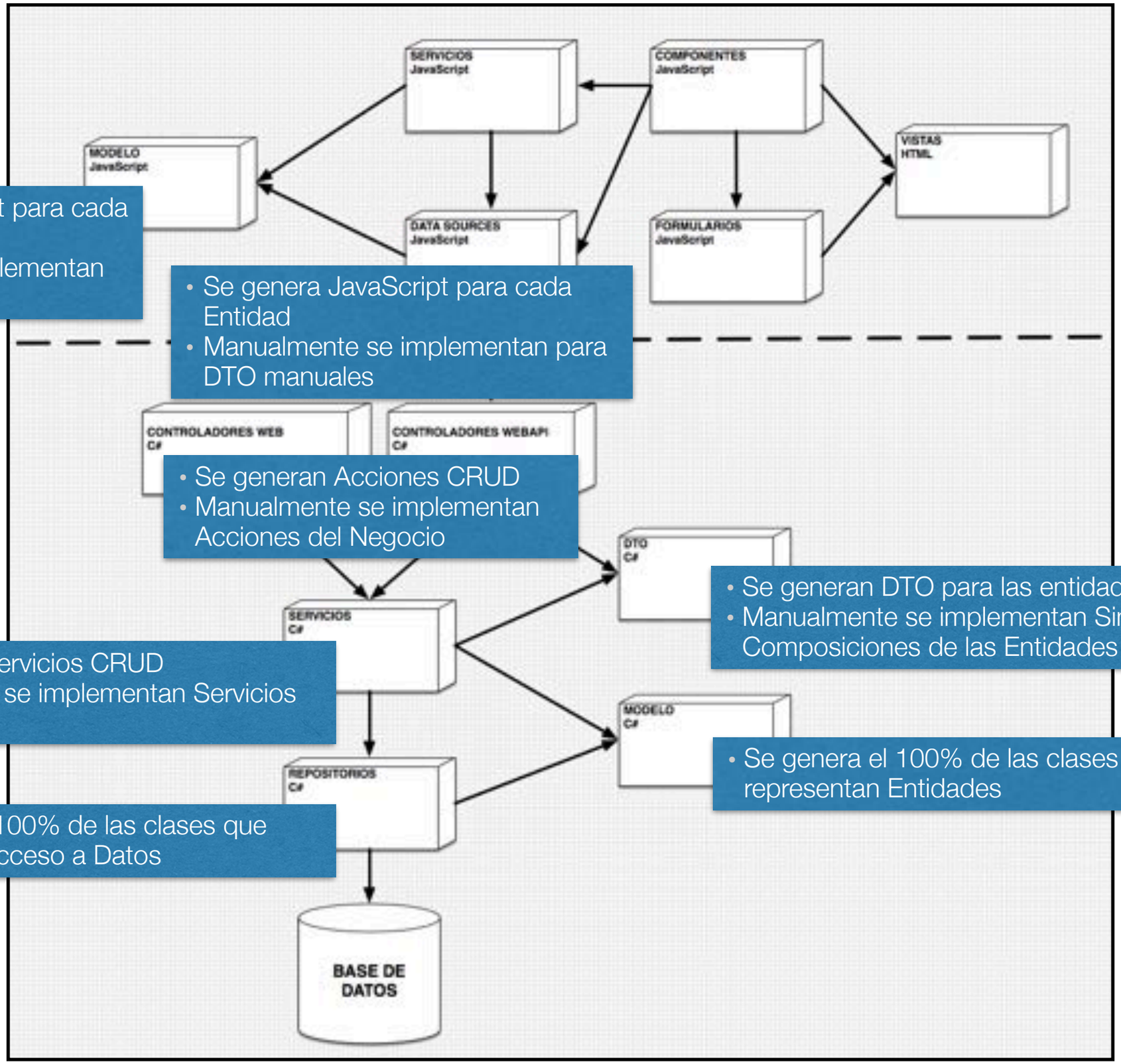
- Se generan DTO para las entidades de Negocio
- Manualmente se implementan Simplificaciones y Composiciones de las Entidades

- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

Generador de Código



- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Se generan Acciones CRUD
- Manualmente se implementan Acciones del Negocio

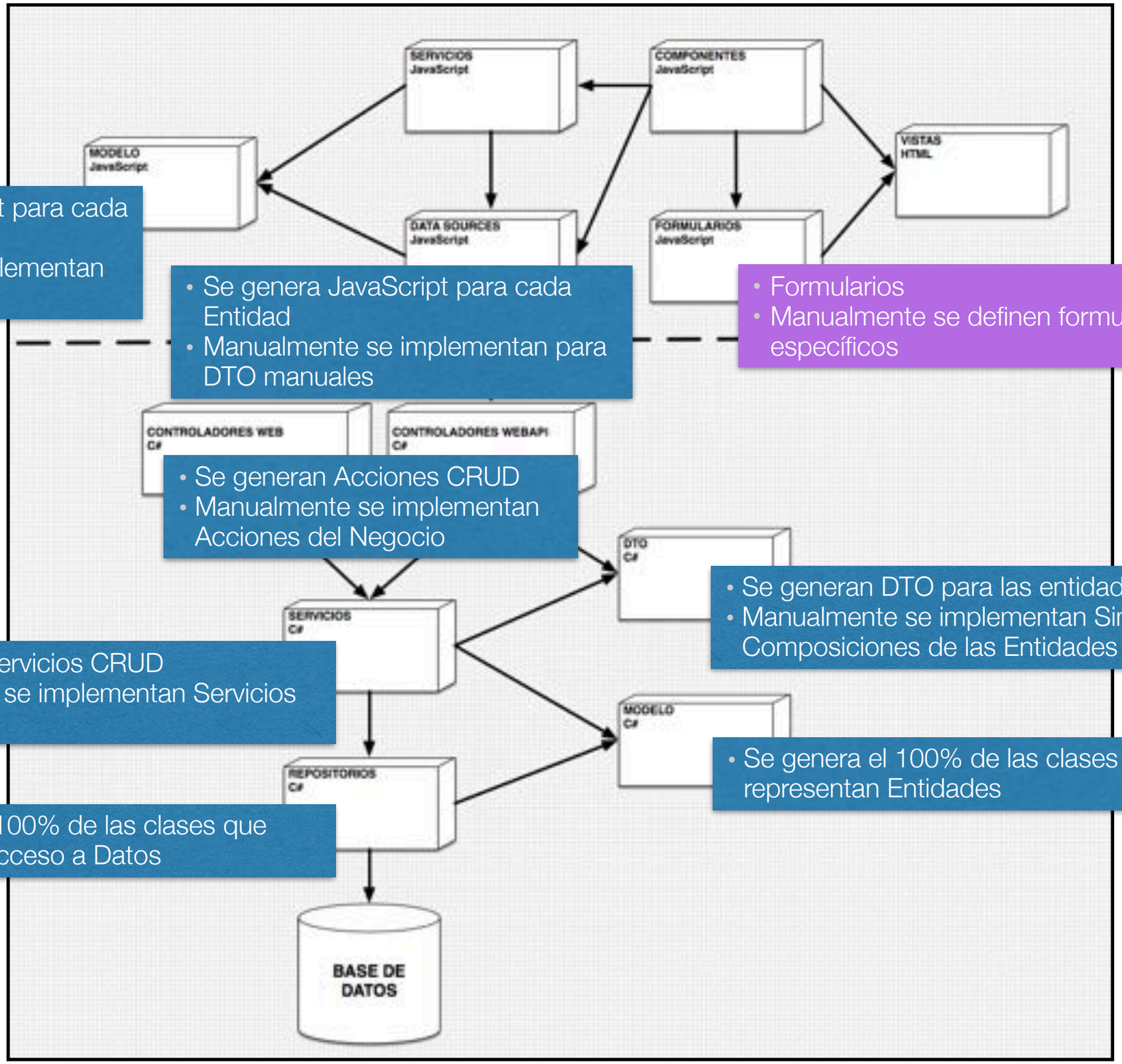
- Se generan DTO para las entidades de Negocio
- Manualmente se implementan Simplificaciones y Composiciones de las Entidades

- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

Generador de Código



- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Formularios
- Manualmente se definen formularios específicos

- Se generan Acciones CRUD
- Manualmente se implementan Acciones del Negocio

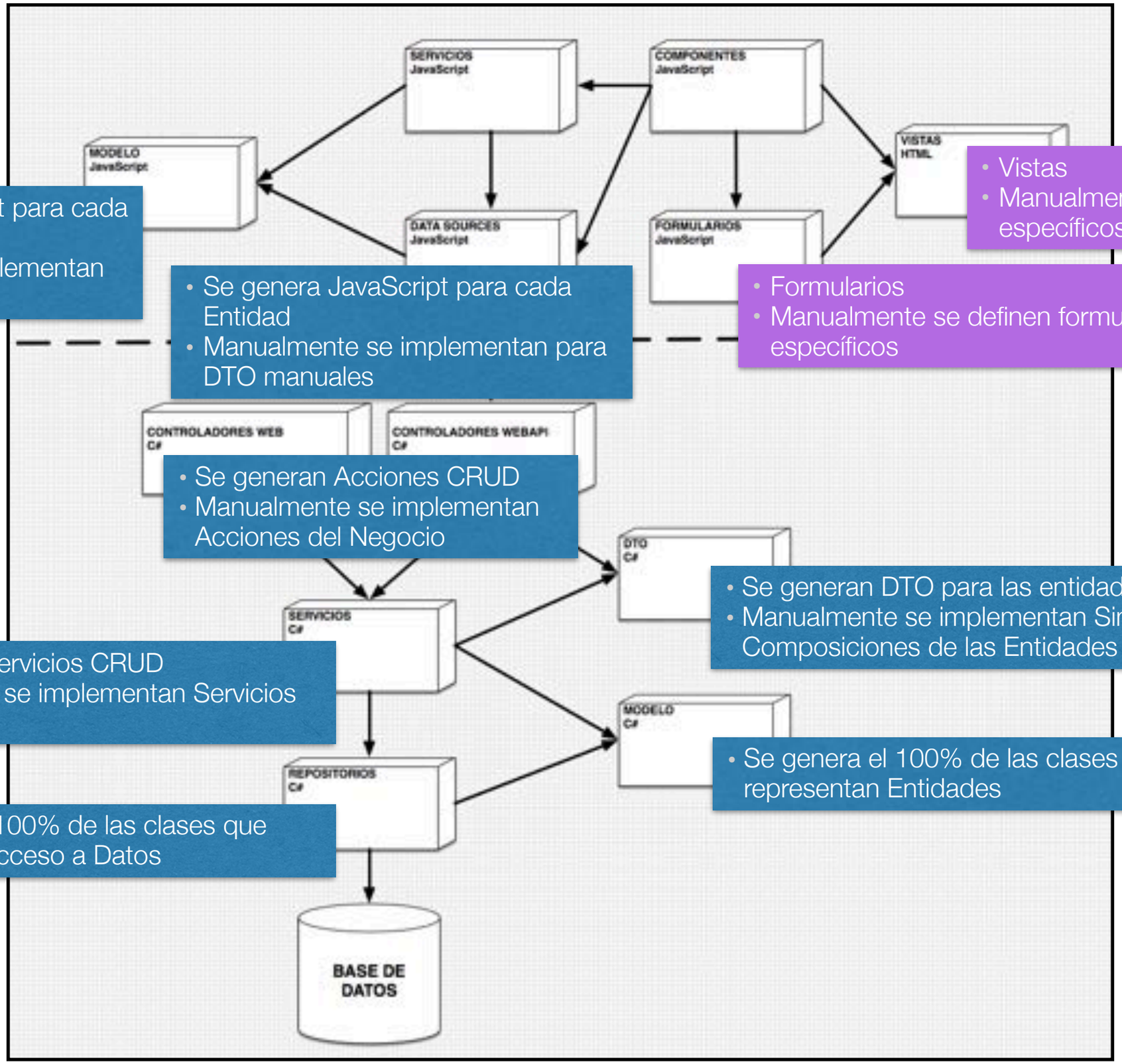
- Se generan DTO para las entidades de Negocio
- Manualmente se implementan Simplificaciones y Composiciones de las Entidades

- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

Generador de Código



- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Vistas
- Manualmente se definen vistas específicos

- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Formularios
- Manualmente se definen formularios específicos

- Se generan Acciones CRUD
- Manualmente se implementan Acciones del Negocio

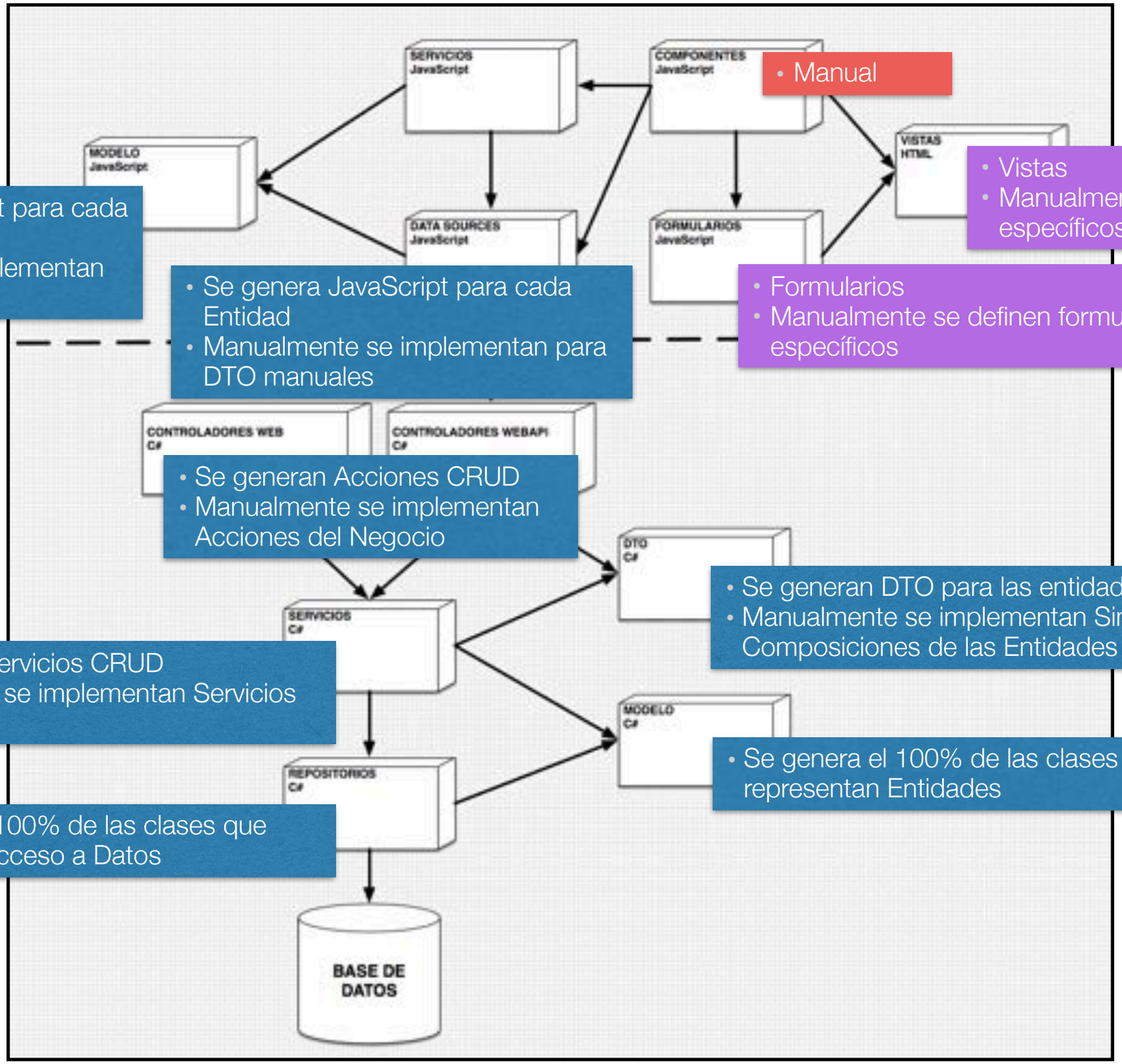
- Se generan DTO para las entidades de Negocio
- Manualmente se implementan Simplificaciones y Composiciones de las Entidades

- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

Generador de Código



- Manual

- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Vistas
- Manualmente se definen vistas específicos

- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Formularios
- Manualmente se definen formularios específicos

- Se generan Acciones CRUD
- Manualmente se implementan Acciones del Negocio

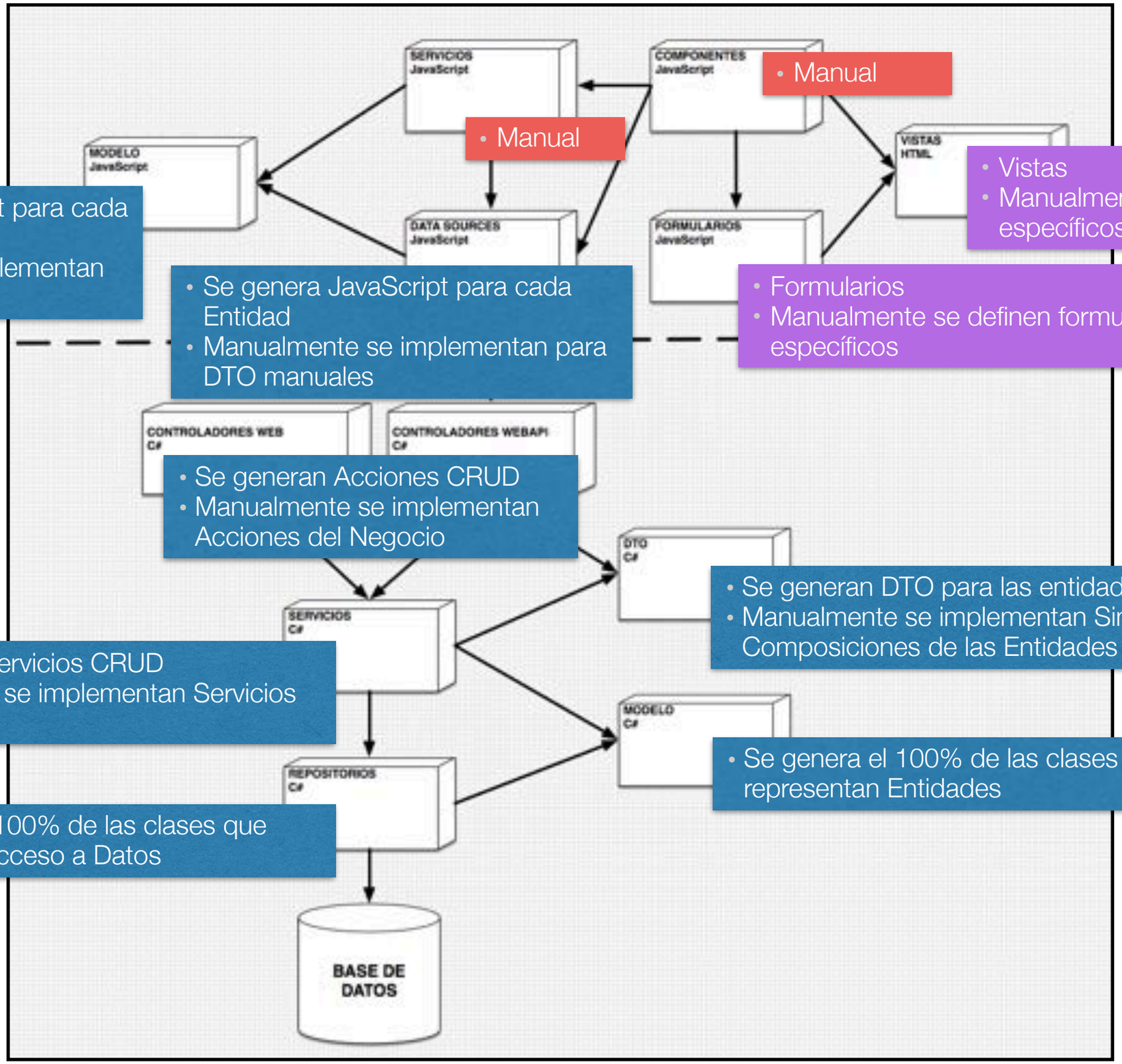
- Se generan DTO para las entidades de Negocio
- Manualmente se implementan Simplificaciones y Composiciones de las Entidades

- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

Generador de Código



- Manual

- Manual

- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Vistas
- Manualmente se definen vistas específicos

- Se genera JavaScript para cada Entidad
- Manualmente se implementan para DTO manuales

- Formularios
- Manualmente se definen formularios específicos

- Se generan Acciones CRUD
- Manualmente se implementan Acciones del Negocio

- Se generan DTO para las entidades de Negocio
- Manualmente se implementan Simplificaciones y Composiciones de las Entidades

- Se generan Servicios CRUD
- Manualmente se implementan Servicios del Negocio

- Se genera el 100% de las clases que representan Entidades

- Se genera el 100% de las clases que controlan el acceso a Datos

GHL Innovación & Tecnología

Resultados

Resultados

Proyectos	Manual	Generado	Total	Porcentaje Generado
GBook	59,103	92,167	151,270	**60.93%%**
Travelers Club	24,053	58,867	82,920	**70.99%**
SIM	41,313	36,389	77,702	**46.83%**
Master Services	9,547	28,442	37,989	**74.86%**
Navigation	13,963	17,812	31,775	**56.05%**
Indicators	11,694	10,731	22,425	**47.85%**
TOTAL	**159,673**	**244,408**	**404,081**	**60.48%**

Resultados

Lenguajes	Manual	Generado	Total	Porcentaje Generado
C#	55,970	178,759	234,729	**76.16%**
JavaScript	79,265	34,091	113,356	**30.07%**
Razor/ cshtml	24,438	31,558	55,996	**56.36%**
Total	**159,673**	**244,408**	**404,081**	**60.48%**

Conclusiones

- La generación de código aumenta la productividad de equipos de desarrollo pequeños.
- Excelente modelador (VS NET Visualization & Modeling SDK).
- Deficiente soporte para Transformaciones.
- C# Alto soporte para código manual/generado.

Conclusiones

- Lenguaje “Hotelero” no es posible en el estado del arte actual
- Arquitectura UI/UX traslada complejidad al cliente
- Crear modelos de presentación para UI/UX es muy complejo y requiere un gran esfuerzo.

Trabajo actual/futuro

- Replantear modelador presentación.
- Incrementar generación JavaScript/TypeScript con controles simples y reutilizables.
- Manejo de Código Manual/Generado en TypeScript/ JavaScript/CSHTML.
- Implementar Modelo de variabilidad para plataforma y patrones de diseño.
- Modelar y generar Autorización

LOGICA GHL

GHL Innovación & Tecnología