Migración UI a .Net C#

Generación automática de apis rest

INTERSYSTEMS MEETUP 2023

inalsa anesa alcar zamecal

01

HISTORIA

inalsa anesa alcar zamecal

inalsa anesa alcar zamecal

60 años

Más de 60 años de experiencia

73 %

73% exportaciones de aluminio

117 sectores

Trabajamos 117 sectores diferentes

Delegaciones en Alemania

1. **Dusseldorf**
   - alemania.norte@inalsa.net
2. **Munich**
   - alemania.sur@inalsa.net
3. **Hamburgo**
   - alemania.noreste@inalsa.net

Delegaciones en Francia

1. **Lyon**
   - commercial@inalsa.net
2. **Paris**
   - ventes@inalsa.net
3. **Toulouse**
   - ventes@inalsa.net

Delegaciones en España

1. **Central**
   - ventas@inalsa.net
2. **Cataluña**
   - mrg@inalsa.net
3. **Norte**
   - csc@inalsa.net
4. **Levante**
   - perfiles@inalsa.net
5. **Centro**
   - zonacentro@inalsa.net

Delegaciones en Reino Unido

1. **Birmingham**
   - sales@inalsa.net
2. **Londres**
   - ve1@inalsa.net

HISTORIA

1957

- Nacimiento de Inalsa.
- Incorporación de prensas de extrusión de aluminio.
- Nacimiento de Anesa (anodizadores) y Alcar (distribuidores carpinteria)

1980s

- Informatización de determinadas secciones de la empresa basado en PDP11 con DSM (Digital Standard Mumps)

1990s

- Migración a ISM Intersystem
- Comienzo desarrollo de aplicaciones gráficas con VB6

2000s

- Migración a Cache Intersystem e interconexión con VB6.
- Informatización completa del ERP, incluida la digitalización de planta.

2010s

- Migración de VB6 a .NET C#
- Migración de CACHE a IRIS
- Apertura delegación en Colombia
- Nacimiento de Zamecal (centro de mecanizado)

ACTUALMENTE

- Implementación de PowerBI con conexión de cubos de Analytics
- Desarrollo de la interoperabilidad para diversos aspectos (automatización de recepciones de proveedores, lecturas de autómatas,..)
- Integración de repositorio de control de código fuente de IRIS
- Actualizar a IRIS 2023 para el uso de Machine Learning

02

MIGRACIÓN NET

inalsa anesa alcar zamecal

REQUISITOS

Aplicación de escritorio

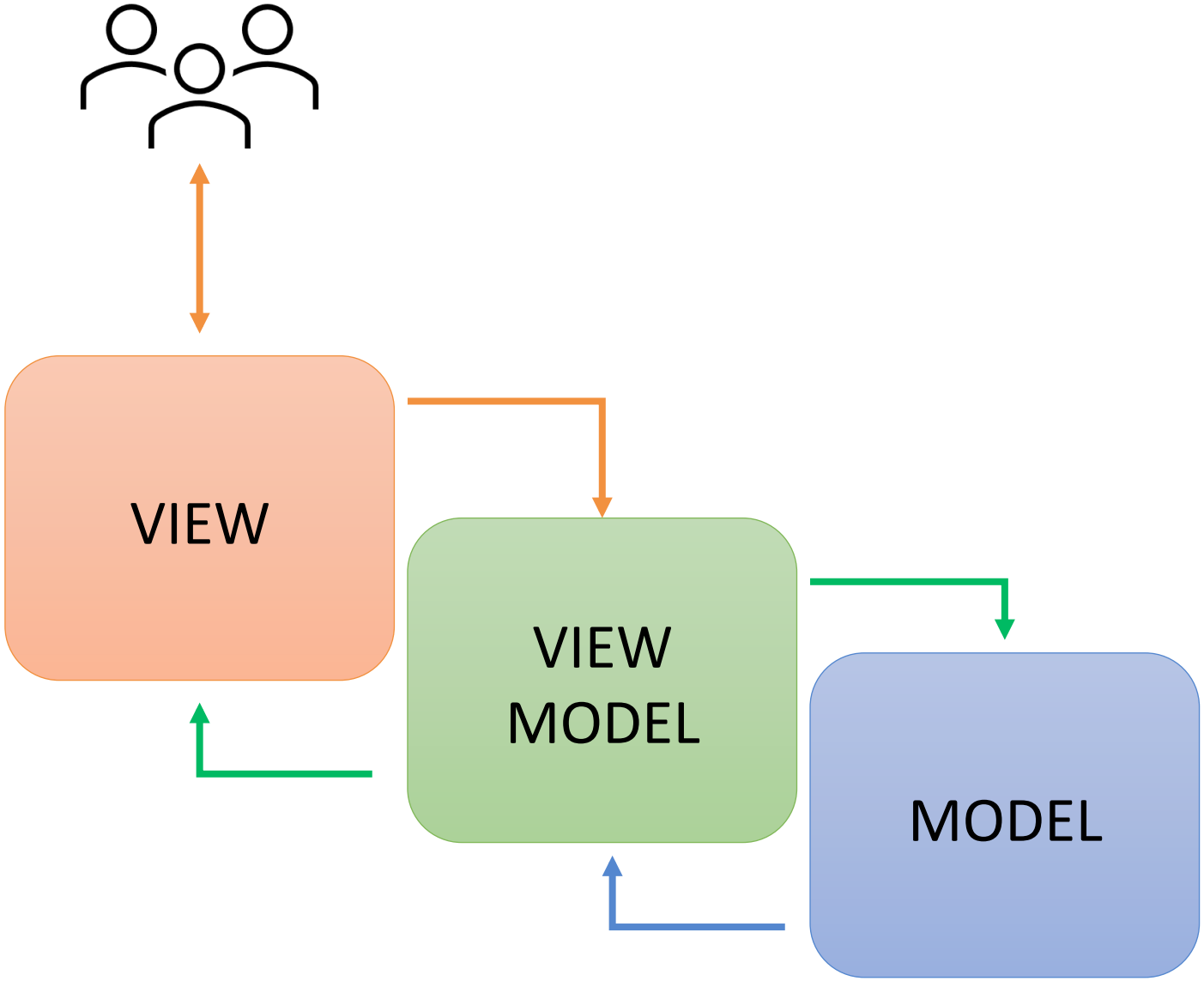
Diseño responsivo

C#

- Abandono de Visual Basic por C#
- Lenguaje más utilizado
- Mayor documentación

WPF

- Enlace a datos con binding
- Mayor flexibilidad a la hora de crear controles.
- XAML mismo lenguaje para crear aplicaciones móviles.

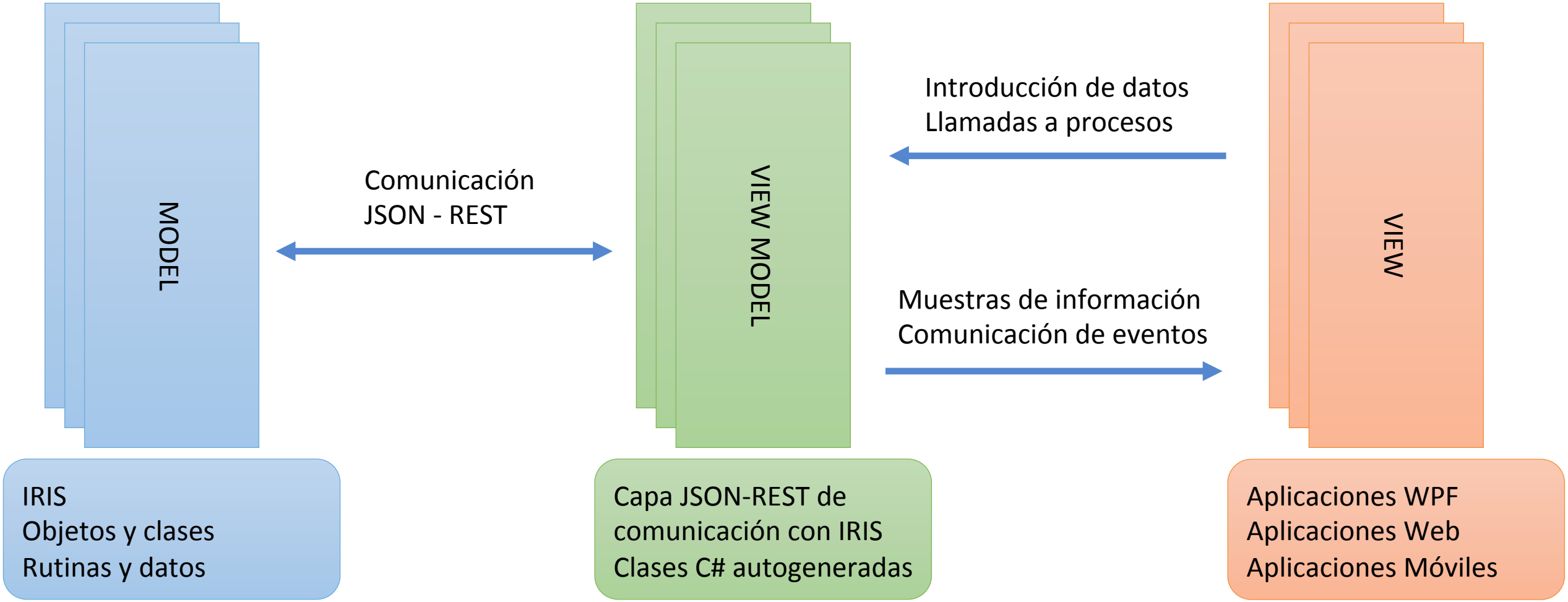


MVVM

Patrón de diseño MVVM (en inglés, model–view–viewmodel)

Se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación

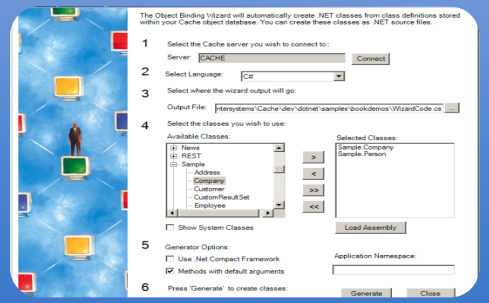
MVVM: MODEL -> VIEWMODEL -> VIEW



03

ACCESO A DATOS

ACCESO A DATOS

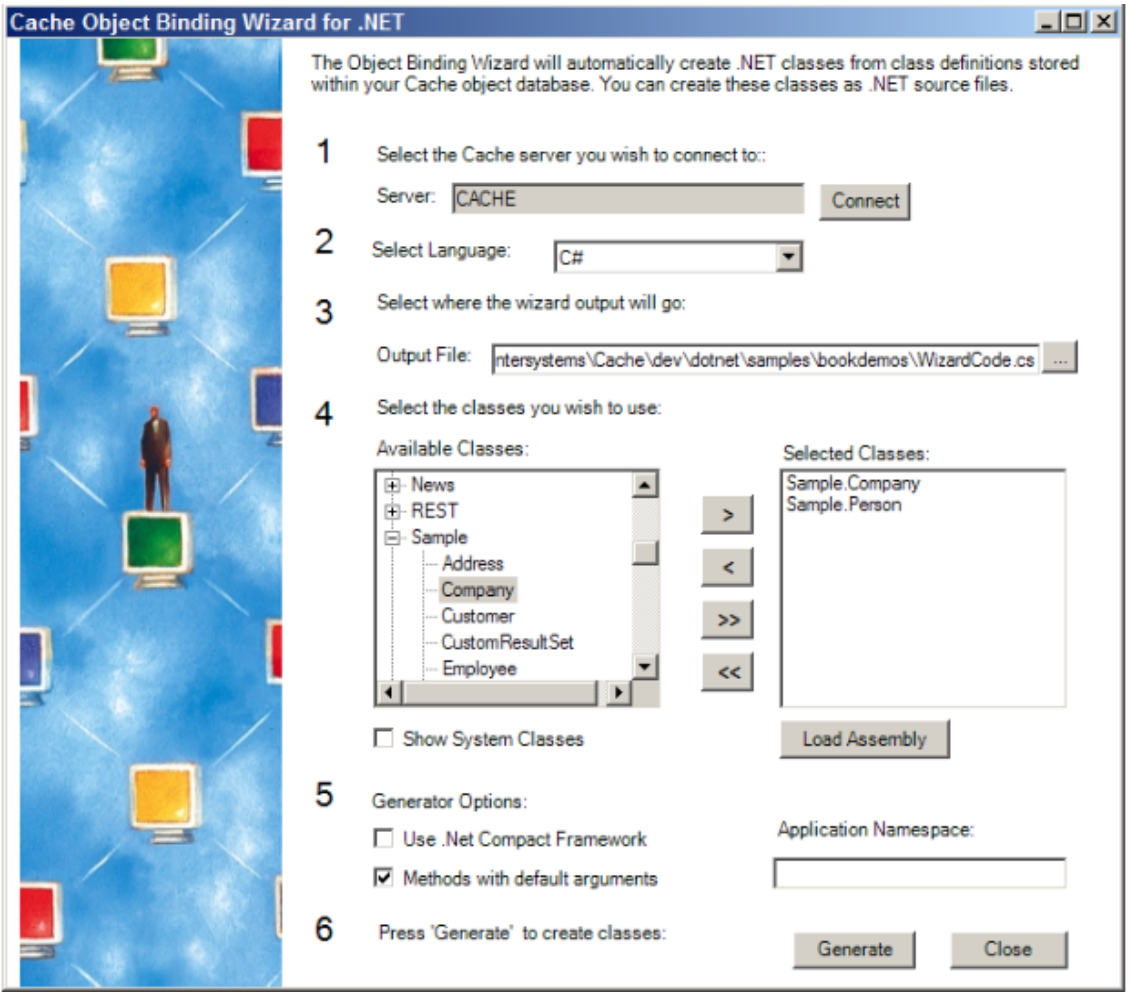


CACHE OBJECT BINDING WIZARD .NET

ENTITY FRAMEWORK

SERVICIOS REST

CACHE OBJECT BINDING WIZARD .NET



SE DESCARTO DEBIDO A:

Discontinuidad en IRIS

Operativa complicada cada vez que se hacía un cambio en la base de datos

Ficheros de gran tamaño

SE DESCARTO DEBIDO A:

Problemas con los tipos de datos

Errores con las relaciones de tablas

SERVICIOS REST

VENTAJAS:

Multiplataforma

Posibilidad de trasladar la inteligencia del negocio a la base de datos

Escalabilidad: Separación cliente – servidor

Seguridad: permite mecanismos de autenticación en cada llamada.

04

CAMINO A REST

CAMINO a REST

En base a las experiencias anteriormente explicadas y tras unas primeras pruebas iniciales en rest se observaron dos elementos que había que realizar cada vez

Modificar un archivo .cs que tuviera las llamadas de los métodos de IRIS

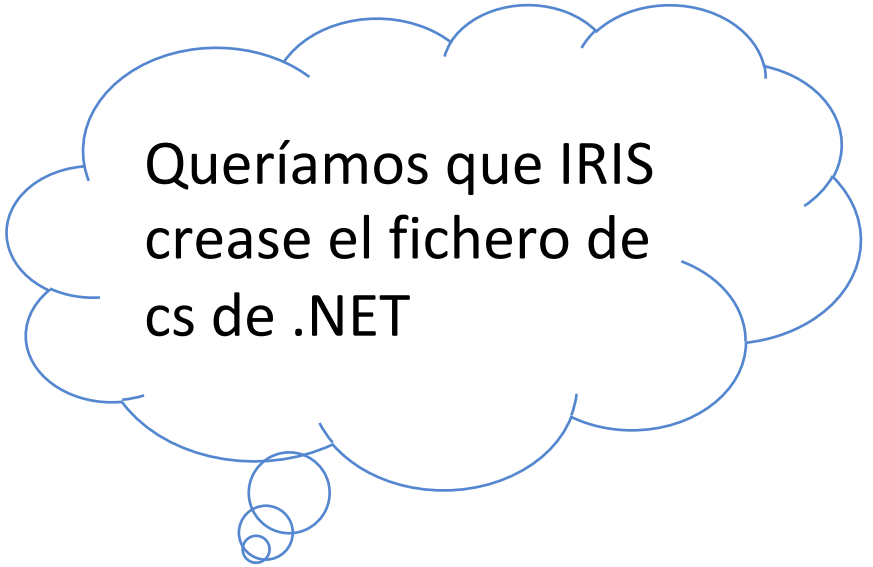
Modificar el fichero de enrutamiento

¿Por qué no automatizarlo?

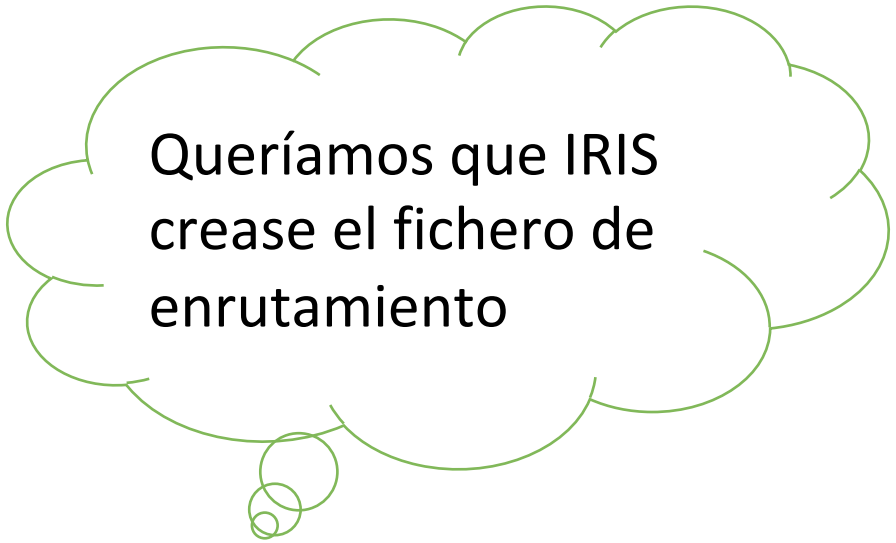
Objetivo

Tener una clase de .cs de .NET que contenga las propiedades y métodos relacionados con una tabla.

CAMINO a REST



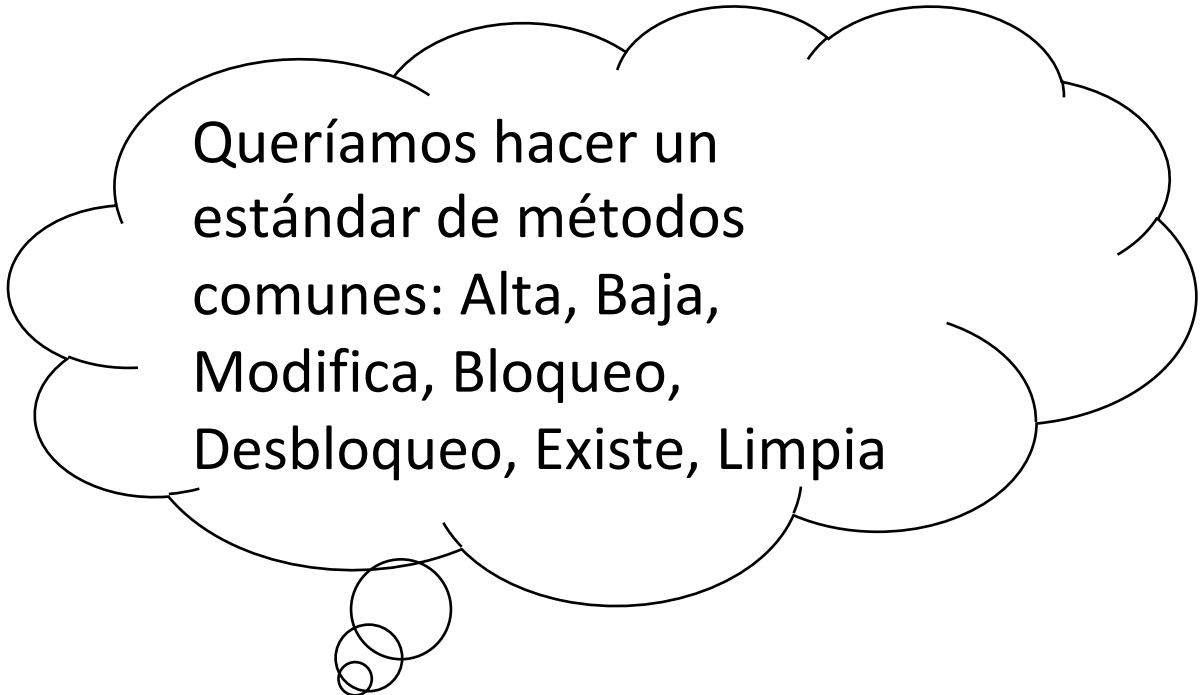
CAMINO a REST



CAMINO a REST

Utilizando la tecnología de objetos de IRIS queríamos una clase proyectada que tuvieran los métodos y propiedades de la lógica de negocio, así como lo necesario para la persistencia de los datos

CAMINO a REST



CAMINO a REST

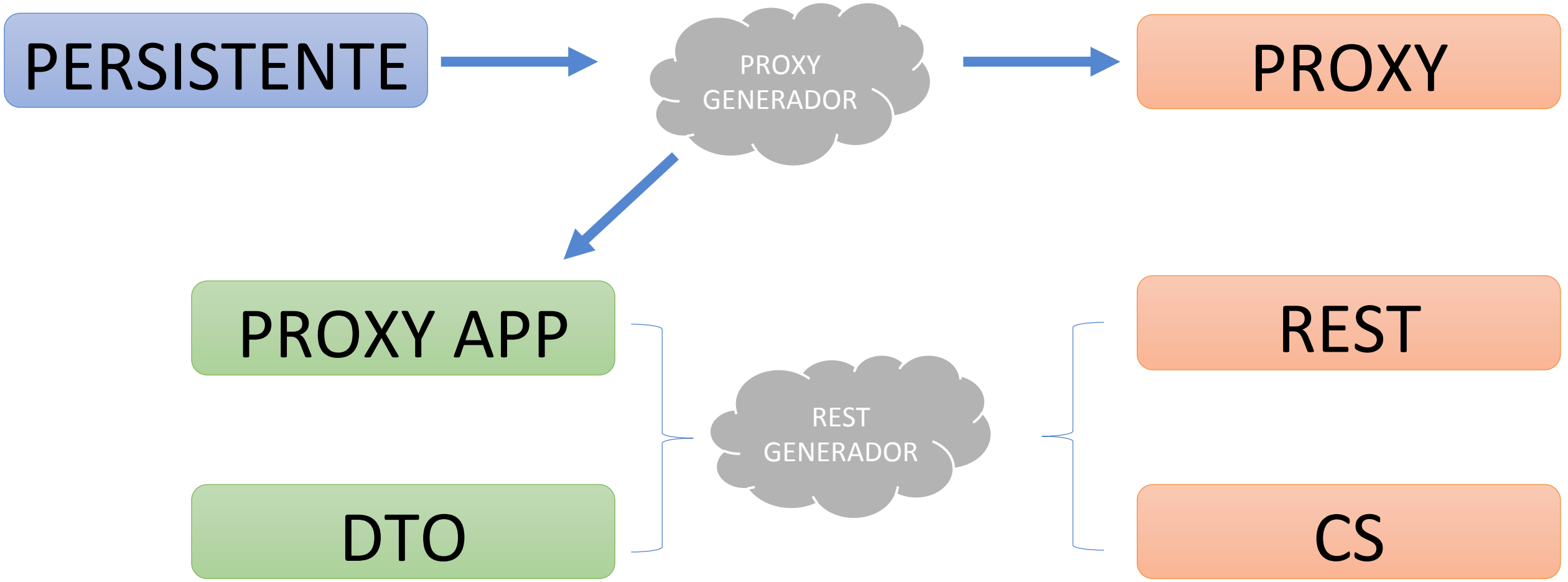
Utilizando la tecnología de objetos de IRIS queríamos una clase proyectada que tuvieran los métodos y propiedades de la lógica de negocio, así como lo necesario para la persistencia de los datos

Queríamos que IRIS crease el fichero de cs de .NET

Queríamos que IRIS crease el fichero de enrutamiento

Queríamos hacer un estándar de métodos comunes: Alta, Baja, Modifica, Bloqueo, Desbloqueo, Existe, Limpia

CAMINO a REST



%Persistant
Lógica de Negocio
Generación Automática

CAMINO a REST

PROXY

Método comunes:

- Alta
- Baja
- Eliminar
- Existe,
- …

Generada por la persistente.

Se crea cada vez que se compila la persistente.

Clase Abstracta

Contiene métodos ‘get’ de cada propiedad: con esto se consigue de manera puntual obtener el valor de un campo indicándole el RowId. Más utilizado para la obtención de campos calculados.

CAMINO a REST

PROXY APP

Clase Registrada que contiene:

- Validaciones antes de guardar.
- SQLs para llenar grids de datos.
- Obtener listas para llenar combos.

Hereda de la Proxy.

Solo se crea cuando no existe y cuando se compila la persistente.

Es la que tiene la lógica de negocio.

Es la encargada de generar el archivo cs que se importa en NET.

Es la encargada de generar el archivo de enrutamiento.

CAMINO a REST

DTO

Clase Registrada que contiene:

- Propiedades básicas.
- Puede contener otras clases ProxyApp.
- Métodos propios de la clase.

Utilizada básicamente para el desarrollo de formularios.

Es la encargada de generar el archivo cs que se importa en NET.

Es la encargada de generar el archivo de enrutamiento.

NO TODO FUE UN CAMINO DE ROSAS…

05

INCONVENIENTES

inalsa anesa alcar zamecal

INCONVENIENTES

BLOQUEOS DE CACHE

NO SE PUEDEN UTILIZAR (lock), YA QUE SON PETICIONES AISLADAS CON JOBs DIFERENTES

CARACTERES RESERVADOS JSON

PETICIONES GET NO DEBEN DE CONTENER CARACTERES RESERVADOS DE LA SINTAXIS REST, EJ: '/'

CONVERSIONES DE TIPOS DE DATOS IRIS - NET

- $LIST DE IRIS LO PROYECTAMOS COMO UNA LISTA
- NUMERIC, INTEGER DE IRIS COMO SE TRANSFORMAN EN .NET (FLOAT, DOUBLE, DECIMAL,…)
- FECHA LAS DEVUELVE EN FORMATO INTERNO
- STREAM PARA CATALOGAR FICHEROS O DEVOLVER UN FICHERO ALMACENADO EN IRIS

INCONVENIENTES

EJECUCION DE TRABAJOS DE LARGA DURACION:

En determinadas circunstancias había que ejecutar trabajos costos, como por ejemplo un recalculo de la producción de un mes.
Tuvimos que ejecutar un job aislado en IRIS e ir informando el estado del job a través de una tabla y en .NET ir preguntando por el estado

ENRUTAMIENTO

Cada vez que compilábamos una clase se generaba todo el enrutamiento y podía dar errores en otras clases.
Tuvimos que crear un semáforo para ver en qué casos generábamos todo el enrutamiento o únicamente el de mi clase

SABER EL ESTADO DE UNA PETICION

Queríamos saber si se había producido algún error interno o si una validación era incorrecta y su motivo
Implementamos CodigoError Y DescripcionError como propiedades de cada clase.

INCONVENIENTES

...PERO
TAMBIÉN
HUBO COSAS
BUENAS...

06

VENTAJAS

inalsa anesa alcar zamecal

VENTAJAS

TIEMPO DE DESARROLLO

COSTE DE DESARROLLO POR FORMULARIO SE HA BAJADO EN CASI UN 50%

ESTANDAR DE TRABAJO

Definición y unificación de una metodología desarrollo y buenas practicas

LOGICA DE NEGOCIO

Conseguimos trasladar la lógica de negocio a IRIS y tener únicamente frontales de visualización

VENTAJAS

CALIDAD DEL DESARROLLO

Al tener métodos estándar de Alta, Baja, Eliminar se consiguieron evitar los errores “básicos”

TEST

Unificación de pruebas entre IRIS Terminal y NET

RESOLUCIÓN DE ERRORES

Más rápida y centralizada al estar la inteligencia en IRIS. En el 90% de los casos no se necesita publicar una nueva versión de las aplicaciones.

¡MUCHAS GRACIAS!!

inalsa anesa alcar zamecal