



DelphiDay  
italian conference

# Dividi et Impera: domina Delphi con il pattern MVC

usare il pattern MVC per soluzioni pulite,  
multiplatforma e ben organizzate



# Carlo Barazzetta

**Ethea S.r.l.** - Socio fondatore e  
Responsabile progetti IT

carlo.barazzetta@ethea.it

x.com/CarloBarazzetta

[github.com/EtheaDev](https://github.com/EtheaDev)

[github.com/carloBarazzetta](https://github.com/carloBarazzetta)

linkedin.com/in/carlo-barazzetta



19-20 Giugno 2025  
Piacenza



## **OPEN-SOURCE PROJECTS** **COMPONENTS**

[github.com/EtheaDev](https://github.com/EtheaDev)

**SVG Icon ImageList** <sup>355\*</sup>

[github.com/EtheaDev/SVGIconImageList](https://github.com/EtheaDev/SVGIconImageList)

**Icon Fonts ImageList** <sup>222\*</sup>

[github.com/EtheaDev/IconFontsImageList](https://github.com/EtheaDev/IconFontsImageList)

**Styled Components** <sup>174\*</sup>

[github.com/EtheaDev/StyledComponents](https://github.com/EtheaDev/StyledComponents)

## **OPEN-SOURCE PROJECTS** **COMPONENTS**

[github.com/EtheaDev](https://github.com/EtheaDev)

**Delphi GoogleMap** <sup>82\*</sup>

[github.com/EtheaDev/DelphiGoogleMap](https://github.com/EtheaDev/DelphiGoogleMap)

**Markdown Help Viewer** <sup>74\*</sup>

[github.com/EtheaDev/MarkdownHelpViewer](https://github.com/EtheaDev/MarkdownHelpViewer)

**DBAware Labeled Components** <sup>30\*</sup>

[github.com/EtheaDev/DBAwareLabeledComponents](https://github.com/EtheaDev/DBAwareLabeledComponents)



## **OPEN-SOURCE PROJECTS**

### **SHELL EXTENSIONS**

[github.com/EtheaDev](https://github.com/EtheaDev)

**SVG Shell Extensions** <sup>163</sup>

[github.com/EtheaDev/SVGShellExtensions](https://github.com/EtheaDev/SVGShellExtensions)

**SKIA Shell Extensions** <sup>72\*</sup>

[github.com/EtheaDev/SKIAShellExtensions](https://github.com/EtheaDev/SKIAShellExtensions)

**Markdown Shell Extensions** <sup>84\*</sup>

[github.com/EtheaDev/MarkdownShellExtensions](https://github.com/EtheaDev/MarkdownShellExtensions)

**Fattura Elettronica Explorer** <sup>24\*</sup>

[github.com/EtheaDev/FExplorer](https://github.com/EtheaDev/FExplorer)

## **OPEN-SOURCE PROJECTS**

### **OTHERS**

[github.com/EtheaDev](https://github.com/EtheaDev)

**InstantObjects** <sup>103\*</sup>

[github.com/EtheaDev/InstantObjects](https://github.com/EtheaDev/InstantObjects)

**VCL Theme Selector** <sup>59\*</sup>

[github.com/EtheaDev/VCLThemeSelector](https://github.com/EtheaDev/VCLThemeSelector)

**Delphi MVC**

[github.com/carloBarazzetta/Delphi MVC](https://github.com/carloBarazzetta/Delphi_MVC)

# AGENDA

---

## 1. Il pattern M.V.C. Principi Generali

- a. Struttura del pattern M.V.C. e Schema visuale
- b. Perché usarlo in Delphi? E' applicabile in App esistenti?

## 2. Esempi di applicazione con il pattern M.V.C.

- a. Una Calcolatrice MVC (VCL e FireMonkey)
- b. Una semplice applicazione di gestione Ordini Multi View (VCL, FMX, Console)
- c. Una applicazione di gestione Ordini/Clienti Multi View (VCL, FMX)

## 3. Utilizzo di un ORM/OPF per la persistenza

- a. Un esempio con InstantObjects

## 4. Conclusioni

- a. Che fine hanno fatto i DataModule e i DataSet?
- b. Per me non è applicabile... o forse sì!

# Il pattern M.V.C. Principi generali

1

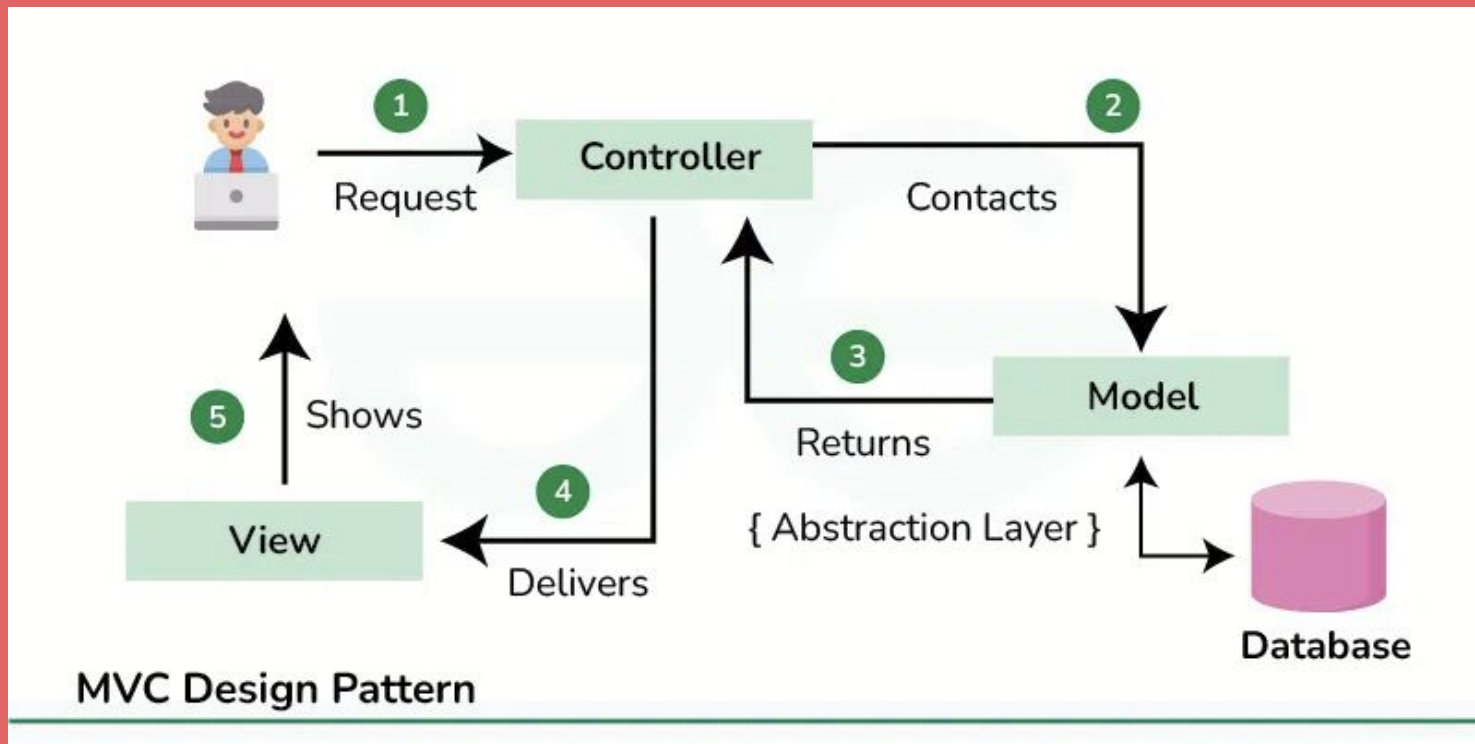
# Struttura del pattern M.V.C.

---

## → MODEL, VIEWS, CONTROLLER

- ◆ Model: Dati e controlli di validità
  - Generalmente Classi
  - Non contiene uno “stato”
- ◆ View(s): Presentazione e interazione con l’utente
  - Interagisce con l’utente (input) e mostra i dati (output)
  - Accede al model per recuperare i dati da mostrare
- ◆ Controller: interagisce tra la View e il Model
  - Riceve l’input dalle View
  - Logica di orchestrazione e manipolazione dei dati del Model
  - Aggiorna la(e) View per mostrare i nuovi dati

# Schema del pattern M.V.C.





# Perché usarlo in Delphi

→ Perché è un linguaggio perfetto per usarlo:

- Gestione dei dati come Classi (e generici) e supporta le Interfacce
- Le Form come “Views” sono già pronte (sono oggetti)
- Se usiamo i Datamodules possiamo trattarli come “Models”
- Ma il Controller??? questo probabilmente è il punto dolente...

→ Perché aiuta a non fare l'errore:

- Di mettere la logica sulla Form
- Di confondere azioni utente (Click) con operazioni da fare

→ Non è l'unico pattern e nemmeno il più complesso...

- Ma è sufficiente per capire i problemi
- Permette di creare Unit Test del funzionamento applicativo

# E' applicabile in App esistenti?

→ In genere no, ma dipende molto se:

- Si è mantenuta separata la logica di presentazione da quella dei dati
- Se si sono usati Datamodule che non fanno “uses” di Form

→ E' facile da applicare se:

- Si fa già uso di Classi/Oggetti per gestire i dati
- Le Form non hanno dipendenze tra loro (usando RegisterClass)

→ E' sicuramente fattibile se:

- Abbiamo sviluppato una applicazione server (es.REST) e serve una GUI

# Esempi di applicazione con il pattern M.V.C.

# 2

# Una Calcolatrice MVC (1)

---

- L'idea: una “challenge” interessante
  - <https://github.com/Embarcadero/ComparisonResearch>
- Partiamo da una versione VCL tutta su una form:
  - Dividiamo le responsabilità
    - Cosa deve fare la View
    - Cosa deve fare il Controller
    - Cosa deve fare il Model
- Versione MVC per VCL
  - Spostiamo le operazioni nel Controller
  - Spostiamo la logica di calcolo nel Model

# Una Calcolatrice MVC (2)

---

## → Versione per FMX

- Model e Controller non cambiano (ben isolati)
- La Form FMX mantiene le stesse logiche della VCL (è stupida!)

## → Domande:

- Ha senso il pattern MVC?
- Quale altro pattern/struttura avresti usato?
- Non c'è qualcosa di sbagliato nell'esempio?

## → Esercizio:

- Puliamo il Model
- Pensiamo ad una versione Multi-View (Standard / Scientifica)



# Semplice app gestione Ordini

## → Struttura Minimale MVC (MultiPiattaforma VCL, FMX, Console)

- Model semplice (Ordine e Lista Ordini)
- Una sola View (Elenco Ordini, Add, Delete)
- Un Controller semplice (AddOrder, DeleteOrder, RetrieveOrder)
- Una interfaccia IViewHandler per astrarsi da VCL, FMX, Console

## → Domande:

- Perché usare le Interfacce con le View?
- A cosa serve la function YesNo in IViewHandler?

## → Esercizio:

- Aggiungiamo la funzionalità di Edit current Order

# App gestione Ordini/Clienti

## → Struttura estesa MVC (MultiPiattaforma VCL, FMX)

- Model composito (Ordini, Clienti, Countries)
- 3 View: Main (container), Customers e Orders (filtrabile)
- Un Controller per le operazioni (AddXXX, DeleteXXX, RetrieveXXX)
- Una interfaccia IViewHandler avanzata (RefreshCurrentView)

## → Domande:

- Meglio un solo Controller o un Controller per ogni "View"?
- Meglio Forms o Frames per le View "Embedded"?

## → Esercizio:

- Aggiungiamo la View di visualizzazione dei Countries

# Utilizzo di un ORM/OPF per la persistenza

3

# Esempio con InstantObjects (1)

## → Un DataModel per l'accesso e la persistenza

- InstantConnector: Broker XML (Le classi devono ereditare da InstantObjects)
- InstantSelector per “presentare” gli oggetti (esempio manuale)
- Svantaggi:
  - ◆ Il Model deve essere strutturato secondo i “canoni” di InstantObjects
  - ◆ Dipendenza dalla VCL (superabile...)
- Vantaggi:
  - ◆ Flessibilità: si adatta a qualsiasi “storage” SQL o file
  - ◆ Performance, grazie a RefCounting e Lazy-Loading
  - ◆ Sviluppato in Italia

# Esempio con InstantObjects (2)

## → Un sguardo alla Demo Primer

- Brokers di vari tipi (XML, FireDAC/SQL, ADO)
- Form di View e di Editing
- Ereditarietà delle Classi e Visuale
- Utilizzabile anche con database “legacy”
- Forte Integrazione nell’IDE
- 

◆ Facilità nello sviluppo di server REST con WiRL (domani!)



# Conclusioni

4

# E per le vecchie applicazioni?

→ Che fine hanno fatto i DataModule e i DataSet?

◆ Salvaguardiamo i principi MVC:

- I Datamodule sono “isolati” (non vedono le View)?
  - Posso usare la logica di business creando un server REST
  - Posso renderli più simili ad un “Model”
  - Ma come fanno a rinfrescare le Form “View”
- Le Form (View) sono isolate tra di loro?
  - Creare un meccanismo di registrazione e visualizzazione
  - Creare un Controller che le orchestra
  - Creare un Datamodule (isolato) per ogni Form



THANK YOU

