Prototipo per la Visualizzazione dei Dati Interferometrici su Mappa Google utilizzando Oracle Spatial

* Autore: Fabio Privoli
* Data: 5 agosto 2025

[Introduzione 2](#_Toc205302737)

[Architettura Generale 2](#_Toc205302738)

[Componenti Principali 2](#_Toc205302739)

[GeoDataOverlayWindow (VB.NET WPF) 2](#_Toc205302740)

[Gestione Database Oracle 3](#_Toc205302741)

[Gestione Overlay WebGL e Raster 3](#_Toc205302742)

[Comunicazione con WebView2 e Google Maps 3](#_Toc205302743)

[Gestione Grafici (OxyPlot) 3](#_Toc205302744)

[Gestione Colori e Visualizzazione 3](#_Toc205302745)

[Campionamento dei Dati: Logiche, Soglie e Dettaglio Tecnico 4](#_Toc205302746)

[**Perché il campionamento?** 4](#_Toc205302747)

[**Specifiche tecniche di campionamento implementate** 4](#_Toc205302748)

[Gestione Overlay Raster: Scenari e Motivazioni 5](#_Toc205302749)

[**Quando viene usata la visualizzazione raster (immagine base64)?** 5](#_Toc205302750)

[**Perché questa scelta?** 5](#_Toc205302751)

[Motivazioni Tecnologiche: Scelta delle Librerie 5](#_Toc205302752)

[**Deck.gl** 5](#_Toc205302753)

[**WebGL** 6](#_Toc205302754)

[**Altre librerie:** 6](#_Toc205302755)

[Esempi di Codice e Logica Implementativa 6](#_Toc205302756)

[Caricamento e campionamento punti da Oracle 6](#_Toc205302757)

[Overlay Raster 7](#_Toc205302758)

[Considerazioni sulle Performance 7](#_Toc205302759)

[Glossario 7](#_Toc205302760)

[Indice 8](#_Toc205302761)

## Introduzione

Questa documentazione descrive in profondità la logica e le scelte tecnologiche adottate nel prototipo per la visualizzazione di dati interferometrici su mappa Google, sfruttando Oracle Spatial come backend e tecnologie di rendering moderne (WebGL, rasterizzazione dinamica).

## Architettura Generale

L’applicazione si basa su una finestra overlay in ambiente WPF (VB.NET) che consente di caricare e visualizzare dati spaziali in modo interattivo e performante. L’architettura è suddivisa in:

* **Frontend desktop**: interfaccia WPF per la gestione della mappa e dei dati;
* **Backend dati**: Oracle Spatial per interrogazioni spaziali, campionamento e gestione attributi;
* **Rendering avanzato**: WebView2 con Google Maps e Deck.gl per la visualizzazione di grandi dataset;
* **Analisi dettagliata**: popup informativi e grafici storici tramite OxyPlot.

## Componenti Principali

### GeoDataOverlayWindow (VB.NET WPF)

Questa classe gestisce:

* Caricamento dei bounds globali dal database;
* Richiesta e visualizzazione dei punti spaziali tramite overlay WebGL o raster;
* Debug e log tramite pannello dedicato;
* Gestione eventi utente (click, zoom, aggiornamento automatico);
* Popup informativi sui punti, inclusi attributi e grafico storico.

#### Strutture dati principali

' Oggetto punto geografico con attributi  
Public Class GeoPoint  
 Public Property Id As String  
 Public Property Latitude As Double  
 Public Property Longitude As Double  
 Public Property Value As Double  
 Public Property Attributes As New Dictionary(Of String, Object)  
End Class  
  
' Bounding box geografico  
Public Class BoundingBox  
 Public Property North As Double  
 Public Property South As Double  
 Public Property East As Double  
 Public Property West As Double  
End Class

### Gestione Database Oracle

La classe OracleConnectionHelper incapsula la connessione e l’esecuzione di query su database Oracle.  
Permette sia query di lettura (con ritorno DataTable) che operazioni di scrittura (ExecuteNonQuery), con gestione di parametri e timeout.

**Esempio di utilizzo**

' Esecuzione di una query SQL parametrica  
Dim dt = \_oracleHelper.ExecuteQuery(query)

### Gestione Overlay WebGL e Raster

* **Overlay WebGL**: utilizza Deck.gl tramite interfaccia JavaScript in WebView2, permettendo interazione e selezione dei punti.
* **Overlay Raster**: genera un’immagine PNG in memoria sulla base dei punti, la codifica in base64 e la invia al client JS per visualizzazione tramite GroundOverlay di Google Maps.

### Comunicazione con WebView2 e Google Maps

La comunicazione tra backend .NET e frontend JS avviene tramite messaggi JSON.  
Gli eventi principali gestiti sono:

* Caricamento, bounds cambiati, zoom cambiato;
* Click su marker o mappa;
* Richiesta di overlay/rimozione;
* Messaggi di debug ed errori.

### Gestione Grafici (OxyPlot)

Apertura di una finestra popup che mostra la serie storica degli spostamenti per il punto selezionato, con:

* Asse X temporale (date degli acquisiti);
* Asse Y spostamento (mm);
* Colori allineati alla visualizzazione su mappa;
* Calcolo e disegno di una regressione lineare sui dati.

### Gestione Colori e Visualizzazione

Gestione avanzata della colorazione tramite:

* Scala fissa coerente tra mappa e grafici;
* Scala rainbow dinamica normalizzata tra minimo e massimo dei valori.

## Campionamento dei Dati: Logiche, Soglie e Dettaglio Tecnico

### **Perché il campionamento?**

Il campionamento dinamico serve a ridurre il numero di punti caricati e visualizzati in funzione del livello di zoom sulla mappa.  
Questo permette di:

* Evitare il sovraccarico del browser e della rete;
* Mantenere l’interattività dell’applicazione anche con milioni di record;
* Fornire una rappresentazione significativa dei dati sia su zoom ampio che su dettaglio locale.

### **Specifiche tecniche di campionamento implementate**

Il sistema adotta una strategia a soglie ben definite, che variano in base al livello di zoom della mappa.  
Le soglie e il campionamento sono configurabili, ma la logica tipica adottata è la seguente:

| Livello di Zoom | Percentuale SAMPLE Oracle | Numero Massimo Punti (ROWNUM) |
| --- | --- | --- |
| < 12 | 5% | 100.000 |
| 12 – 15 | 20% | 150.000 |
| ≥ 15 | Nessun SAMPLE | 300.000 |

**Esempio di query SQL Oracle con campionamento**

SELECT CODE, VEL,  
 SDO\_GEOM.SDO\_CENTROID(GEOM\_SDO, 0.001).SDO\_POINT.X AS LONGITUDE,  
 SDO\_GEOM.SDO\_CENTROID(GEOM\_SDO, 0.001).SDO\_POINT.Y AS LATITUDE  
FROM GEO\_DATI\_INTERFEROMETRICI\_SPATIAL SAMPLE(20)  
WHERE SDO\_FILTER(  
 GEOM\_SDO,  
 SDO\_GEOMETRY(2003, 4326, NULL,  
 SDO\_ELEM\_INFO\_ARRAY(1,1003,3),  
 SDO\_ORDINATE\_ARRAY(:west, :south, :east, :north)  
 )  
) = 'TRUE'  
AND ROWNUM <= 150000

#### **Logica di selezione del campionamento**

* **Zoom < 12**:
  + Si applica un campionamento molto forte (SAMPLE(5)), per rappresentare solo una minima parte del dataset ma comunque sufficiente per visualizzazioni di insieme.
  + Si limita a 100.000 punti massimi (ROWNUM).
* **Zoom tra 12 e 15**:
  + Campionamento intermedio (SAMPLE(20)), per aumentare il dettaglio man mano che si zooma.
  + Si limita a 150.000 punti massimi.
* **Zoom ≥ 15**:
  + Nessun campionamento (tutti i punti dentro i bounds), ma comunque si pone un tetto massimo a 300.000 record per evitare saturazione.
* **Zoom > 17**:
  + Si può anche aumentare il tetto massimo, ma nella pratica si mantiene la soglia per sicurezza.

**Questa logica permette di scalare da visualizzazioni globali a dettagli puntuali senza mai saturare il client.**

## Gestione Overlay Raster: Scenari e Motivazioni

### **Quando viene usata la visualizzazione raster (immagine base64)?**

La scelta tra overlay raster o overlay WebGL avviene dinamicamente, in base al numero di punti da visualizzare:

* **Se il numero di punti è superiore a 100.000** (soglia configurabile),  
  viene creata un’immagine raster (PNG) in memoria, dove ogni punto è rappresentato da un pixel colorato nella posizione geografica corrispondente.
  + L’immagine viene convertita in una stringa base64 e inviata al frontend per essere visualizzata come GroundOverlay su Google Maps.
  + Ciò consente di rappresentare visivamente la densità e la distribuzione dei punti anche quando sono troppi per un rendering interattivo puntuale.
* **Se il numero di punti è inferiore o uguale a 100.000**,  
  si utilizza l’overlay WebGL, che permette l’interazione con i singoli punti (click, popup, evidenziazione).

### **Perché questa scelta?**

* **Motivazione tecnica**:  
  I browser e le librerie WebGL hanno un limite pratico di rendering interattivo (sia per performance che per limiti di memoria grafica). Oltre i 100.000 punti, anche con accelerazione hardware, la fluidità e la reattività della mappa possono degradare notevolmente.
* **Efficienza**:  
  La rasterizzazione consente di rappresentare milioni di punti in modo estremamente veloce e con un carico minimo lato client (si invia solo un’immagine PNG).
* **Esperienza utente**:  
  L’utente percepisce subito la distribuzione spaziale, e può zoomare per ottenere più dettaglio e interazione.

## Motivazioni Tecnologiche: Scelta delle Librerie

### **Deck.gl**

* **Deck.gl** è una libreria JavaScript open source, ottimizzata per la visualizzazione di grandi dataset geospaziali tramite WebGL.
* **Motivazioni della scelta:**
  + Supporta rendering accelerato hardware, fondamentale per prestazioni elevate con decine di migliaia di punti.
  + Offre layer specializzati per scatterplot, heatmap, poligoni e altro, permettendo personalizzazione avanzata.
  + Integrabile direttamente su Google Maps tramite il pacchetto @deck.gl/google-maps, mantenendo la base cartografica e l’interattività.
  + Gestisce nativamente eventi di selezione, highlight e tooltip, semplificando lo sviluppo di overlay interattivi.
  + Scalabilità: in test reali, Deck.gl gestisce senza problemi >50.000 marker con interazioni fluide.

### **WebGL**

* **WebGL** è il motore di rendering grafico alla base di Deck.gl e di molte librerie di visualizzazione avanzate.
* **Motivazioni della scelta:**
  + Permette di scaricare il carico computazionale sulla GPU del client, liberando la CPU per altre operazioni (ad es. gestione eventi, logica dell’applicazione).
  + È l’unica soluzione realmente scalabile per dataset di grandi dimensioni da visualizzare in modo interattivo lato browser.

### **Altre librerie:**

* **OxyPlot** (per i grafici): scelto per la sua integrazione nativa in .NET/WPF, la qualità grafica e la flessibilità nella customizzazione degli assi e delle serie.
* **WebView2**: permette l’integrazione di un browser Chromium direttamente in applicazioni WPF, garantendo compatibilità e prestazioni con tecnologie web moderne.
* **Google Maps JS API**: standard de facto per la base cartografica, offre funzioni avanzate di gestione viewport, eventi e overlay.

## Esempi di Codice e Logica Implementativa

### Caricamento e campionamento punti da Oracle

' Query con campionamento dinamico in base allo zoom  
If Not samplePercentNullable.HasValue OrElse zoomLevel >= 17 Then  
 ' Nessun campionamento, prendi tutti i punti (fino a maxPunti)  
 query = $"  
 SELECT ... FROM GEO\_DATI\_INTERFEROMETRICI\_SPATIAL  
 WHERE SDO\_FILTER(...) = 'TRUE'  
 AND ROWNUM <= {maxPunti}  
 "  
Else  
 ' Applica SAMPLE  
 query = $"  
 SELECT ... FROM GEO\_DATI\_INTERFEROMETRICI\_SPATIAL SAMPLE({samplePercent})  
 WHERE SDO\_FILTER(...) = 'TRUE'  
 AND ROWNUM <= {maxPunti}  
 "  
End If

**Commento**:  
Questa logica assicura che il numero di punti caricati sia proporzionato al livello di zoom e alle capacità del client.

### Overlay Raster

' Per ogni punto, calcola posizione nel raster e colora il pixel  
Dim x = CInt((punto.Longitude - bounds.West) / (bounds.East - bounds.West) \* width)  
Dim y = CInt((bounds.North - punto.Latitude) / (bounds.North - bounds.South) \* height)  
bmp.SetPixel(x, y, colore)

**Commento**:  
Rappresentazione efficiente e “statica” dei dati, utile per visualizzazioni ad alta densità.

Overlay WebGL

// JS - Creazione layer Deck.gl  
new deck.ScatterplotLayer({  
 id: 'scatter-layer',  
 data: punti,  
 getPosition: d => [d[0], d[1]],  
 getFillColor: d => d[3],  
 radiusMinPixels: 4,  
 pickable: true  
})

**Commento**:  
Ogni punto è interattivo e può essere selezionato o evidenziato. La funzione getFillColor applica la colorazione calcolata lato backend.

## Considerazioni sulle Performance

* **Campionamento dinamico**: riduce il carico su database e frontend in funzione del livello di zoom.
* **Overlay raster**: usato per grandissimi dataset (>100.000 punti), minimizza traffico e carico di rendering.
* **Overlay WebGL**: garantisce interazione fluida fino a decine di migliaia di marker.
* **Throttling/debounce eventi**: evita overload di richieste e aggiornamenti.
* **Query parallele Oracle**: hint /\*+ PARALLEL(4) \*/ per sfruttare il parallelismo e accelerare le interrogazioni.

## Glossario

* **Oracle Spatial**: Estensione Oracle per la gestione di dati e query geospaziali.
* **SDO\_GEOMETRY**: Tipo dati spaziale utilizzato in Oracle.
* **WebView2**: Componente browser basato su Chromium per WPF.
* **Deck.gl**: Framework JS per visualizzazione WebGL.
* **OxyPlot**: Libreria .NET per grafici scientifici.
* **Base64**: Codifica binaria-to-text per immagini e dati.
* **BoundingBox**: Area geografica definita da nord, sud, est, ovest.
* **GeoPoint**: Punto geografico con attributi associati.
* **Raster**: Immagine georeferenziata per overlay su mappa.
* **WebGL**: Rendering grafico accelerato via GPU.

## Indice

1. [Introduzione](#introduzione)
2. [Architettura Generale](#architettura-generale)
3. [Componenti Principali](#componenti-principali)
   * [GeoDataOverlayWindow (VB.NET WPF)](#geodataoverlaywindow-vbnet-wpf)
   * [Gestione Database Oracle](#gestione-database-oracle)
   * [Gestione Overlay WebGL e Raster](#gestione-overlay-webgl-e-raster)
   * [Comunicazione con WebView2 e Google Maps](#comunicazione-con-webview2-e-google-maps)
   * [Gestione Grafici (OxyPlot)](#gestione-grafici-oxyplot)
   * [Gestione Colori e Visualizzazione](#gestione-colori-e-visualizzazione)
4. [Campionamento dei Dati: Logiche, Soglie e Dettaglio Tecnico](#X140f7e77f4a81d33f2c7a5359c3b374c012fb9f)
5. [Gestione Overlay Raster: Scenari e Motivazioni](#X1ac5a9566e65b50346ed28ba2367c6a7c5aea1e)
6. [Motivazioni Tecnologiche: Scelta delle Librerie](#X759d354a8b5c8592dea61483ab23f257cce876b)
7. [Esempi di Codice e Logica Implementativa](#esempi-di-codice-e-logica-implementativa)
8. [Considerazioni sulle Performance](#considerazioni-sulle-performance)
9. [Glossario](#glossario)