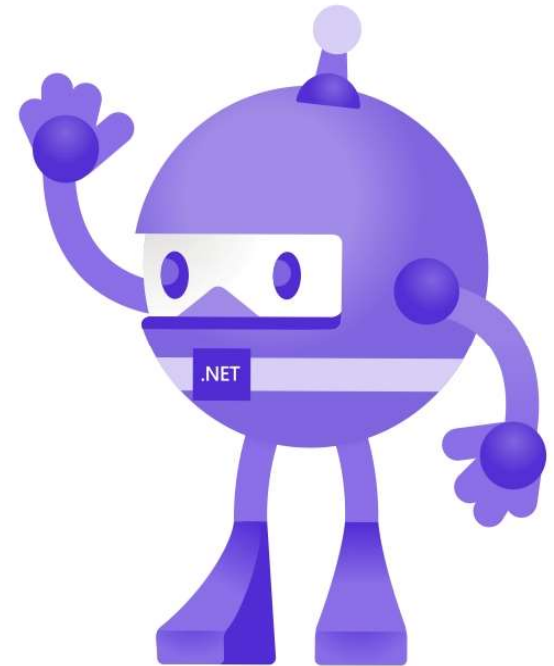


# MAUI & Base Dati disconessa



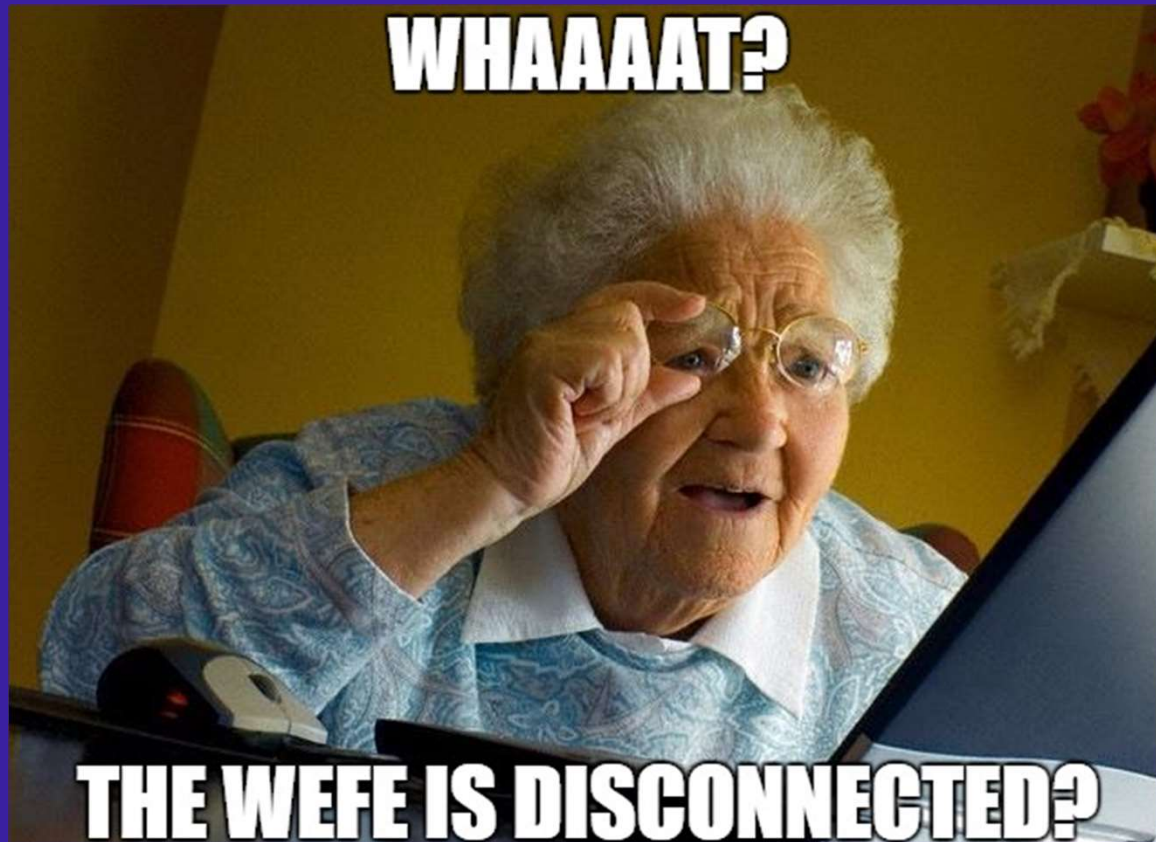
Giampaolo Tucci



# SPONSOR



# Offline-first ?!



- Copertura rete mobile non totale
- Situazioni geografiche sfavorevoli (palazzi, muri, luoghi senza connettività, etc)
- Contratto mobile con limitazioni
- Resilienza e performance

# Challenge per off-line first



- Due database tra loro sincronizzati che sono normalmente disconnessi
  - Database client e backend difformi
  - Dimensione set dati
- Sicurezza
- Gestione del cambiamento dello schema dei dati
- Gestione della concorrenza
- Partizionamento verticale/orizzontale dei dati

# DB Disconnessi: possibili soluzioni



- Cache chiamate Http/Get
- DB Incremental Updates
  - Database Tracking
  - UpdatedAt & Soft-Deleted



# Cache chiamate Http/Get: Akavache

```
//Inizializzazione da fare in fase di startup
Akavache.Registrations.Start(«<Application Name>");

//Salvare un oggetto
var listaSpese = new List<Expense>();
await BlobCache.LocalMachine.InsertObject(«lstSpese", listaSpese);

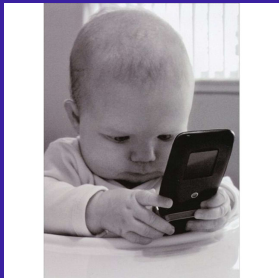
// Recuperare l'oggetto
var listaSpeseFromCache = await
BlobCache.LocalMachine.GetObject<List<Expense>>("lstSpese");

//Attenzione: chiusura corretta (da fare in fase di chiusura applicativo)
BlobCache.Shutdown().Wait();

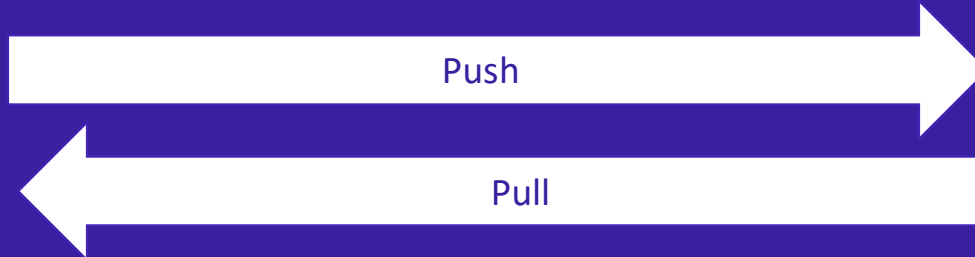
//Chiamata http con supporto Akavacache
await BlobCache.LocalMachine.GetOrFetchObject<List<T>>("lstSpese", async () =>
await Get_ItemsFromWebAsync<T>(), DateTimeOffset.Now.AddHours(8));

//Invalidare Cache
await BlobCache.LocalMachine.Invalidate("lstSpese");
```

# DB Incremental Updates: Definizioni



Client Mobile



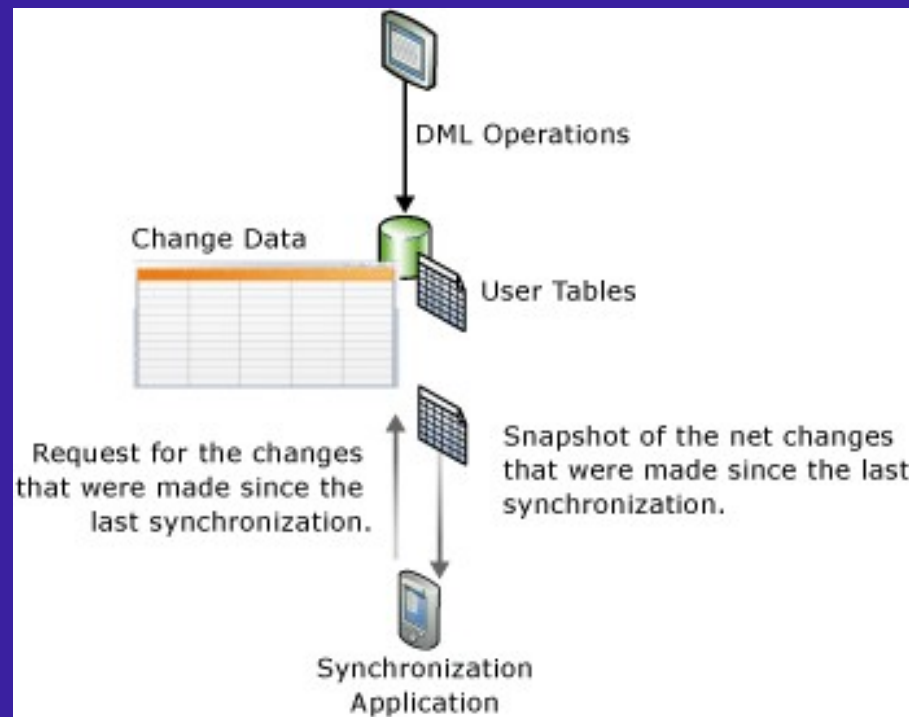
Backend

La sincronizzazione incrementale permette al client di lavorare in offline su una base dati locale e di sincronizzarsi con il database principale ospitato nel backend.

- **Push:** Inviare verso il backend solo i dati inseriti/modificati successivamente all'ultima medesima operazione
- **Pull:** Ricevere dal backend i dati con le medesime modalità.

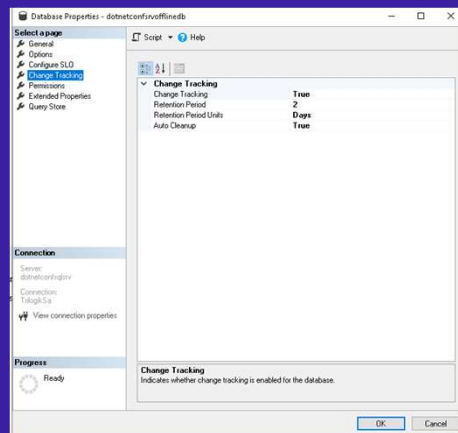
Problema: Come rintracciare i dati che sono stati modificati ?

# Sql Server Change Tracking



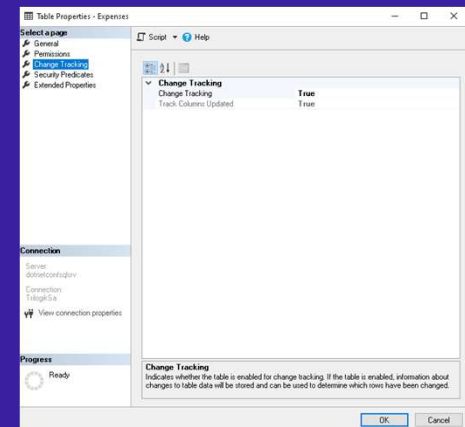


# Sql Server Change Tracking: Attivazione (Database e Tabelle)



```
ALTER DATABASE <Nome Database>  
SET CHANGE_TRACKING = ON  
(CHANGE_RETENTION = 15 DAYS, AUTO_CLEANUP = ON)
```

```
ALTER TABLE <Nome Tabella>  
ENABLE CHANGE_TRACKING  
WITH (TRACK_COLUMNS_UPDATED = OFF)
```



# Sql Server Change Tracker: Comandi notevoli

## Versione Attuale del database

```
select CHANGE_TRACKING_CURRENT_VERSION()
```

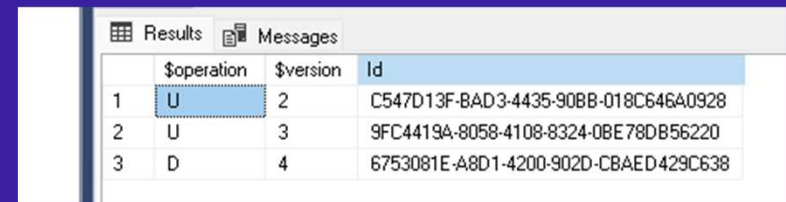
## Modifiche avvenute a partire dalla versione

```
SELECT
```

```
    ct.SYS_CHANGE_OPERATION as '$operation',  
    ct.SYS_CHANGE_VERSION as '$version',  
    ct.Id
```

```
FROM
```

```
    changetable(changes <Nome Tabella>, @fromversion)
```



The screenshot shows the SQL Server Enterprise Manager interface. The 'Results' pane displays a table with three columns: '\$operation', '\$version', and 'Id'. There are three rows of data. The first row shows 'U' for update, version 2, and a GUID. The second row shows 'U' for update, version 3, and a GUID. The third row shows 'D' for delete, version 4, and a GUID.

	\$operation	\$version	Id
1	U	2	C547D13F-BAD3-4435-90BB-018C646A0928
2	U	3	9FC4419A-8058-4108-8324-0BE78DB56220
3	D	4	6753081E-A8D1-4200-902D-CBAED429C638

..... e per i database NoSql ?



### **Azure Cosmos DB**

- Change Feed (Pull Model)

### **Mongo DB**

- Mongo DB Atlas

### **FireBase**

- L'unico pacchetto disponibile in MAUI è *FirestoreDatabase.net*, che però NON supporta offline

# DB Incremental Updates: UpdatedAt & Soft-Delete

## Soluzione *Self-Made*

### Backend

```
public abstract class BaseEntity
{
    [Key]
    public Guid Id { get; set; }

    public DateTimeOffset UpdatedAt { get; set; }

    public bool Deleted { get; set; }
}
```

### Client

```
public abstract class BaseEntityForMobile
{
    [PrimaryKey]
    public Guid Id { get; set; }

    public DateTimeOffset UpdatedAt { get; set; }

    public bool Deleted { get; set; }
}

public class LatestUpdateAt
{
    public string EntityName { get; set; }

    public DateTimeOffset MaxUpdateAt { get; set; }

    public bool Rx { get; set; }
}
```

# DB Incremental Updates: UpdatedAt & Soft-Delete

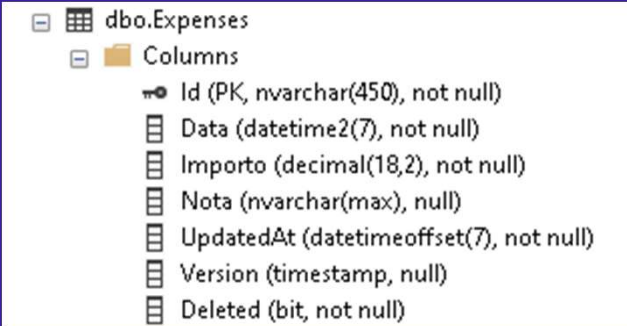
## Soluzione *Microsoft Datasync Framework*

```
public abstract class DatasyncClientData
{
    public string Id { get; set; }

    public bool Deleted { get; set; }

    public DateTimeOffset UpdatedAt { get; set; }

    public byte[] Version { get; set; }
}
```



The screenshot shows the 'Columns' folder expanded for the 'dbo.Expenses' table. The columns listed are:

Column Name	Data Type	Nullable
Id (PK)	nvarchar(450)	not null
Data	datetime2(7)	not null
Importo	decimal(18,2)	not null
Nota	nvarchar(max)	null
UpdatedAt	datetimeoffset(7)	not null
Version	timestamp	null
Deleted	bit	not null

## ...And the winner is.... Conflict resolution client-side



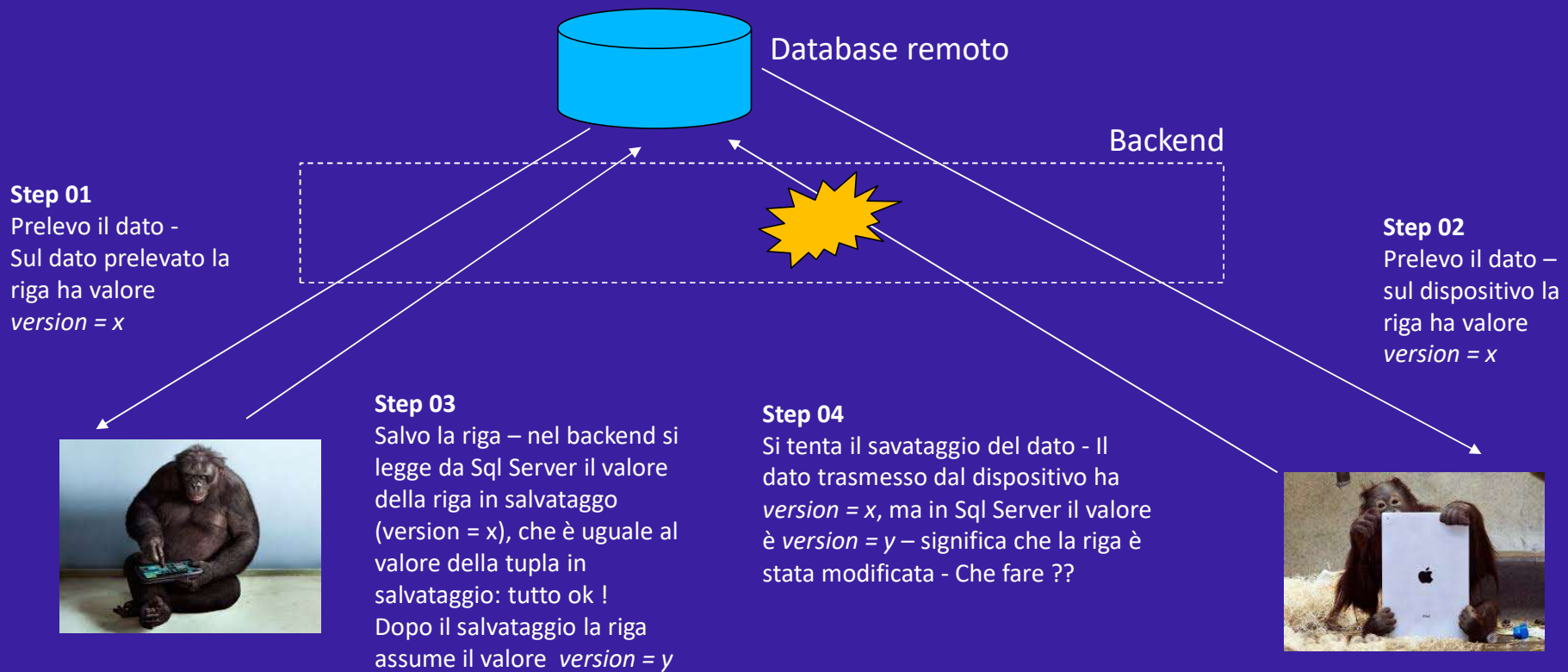
La gestione della concorrenza può essere facilmente implementata lato server, ma spesso è necessario presentare all'utilizzatore la scelta di quale versione debba "vincere".

L'utilizzo dell'SDK *Microsoft Datasync* permette di scegliere, lato client, cosa fare in caso di una modifica che concorrente che insiste su una tupla già modificata.

Lato client il framework rende disponibile facilmente la versione salvata sul server e la versione che si sta spendendo al backend, per poter presentare una scelta all'utilizzatore evidenziando le differenze.



# Gestione della concorrenza (Optimistic Concurrency)



# Perchè non possiamo essere amici ? Come risolvere lato client con *Datasync Framework*



## Eccezione PushFailedException

Eccezione che restituisce le tuple che hanno causato il problema di concorrenza.

## Comandi per risolvere la concorrenza

### *CancelAndUpdateItemAsync*

Cancella l'aggiornamento in corso (client->backend) e aggiorna la tupla locale con quanto passato come argomento.

### *CancelAndDiscardItemAsync*

Elimina l'aggiornamento in corso da parte del client.

### *UpdateOperationAsync*

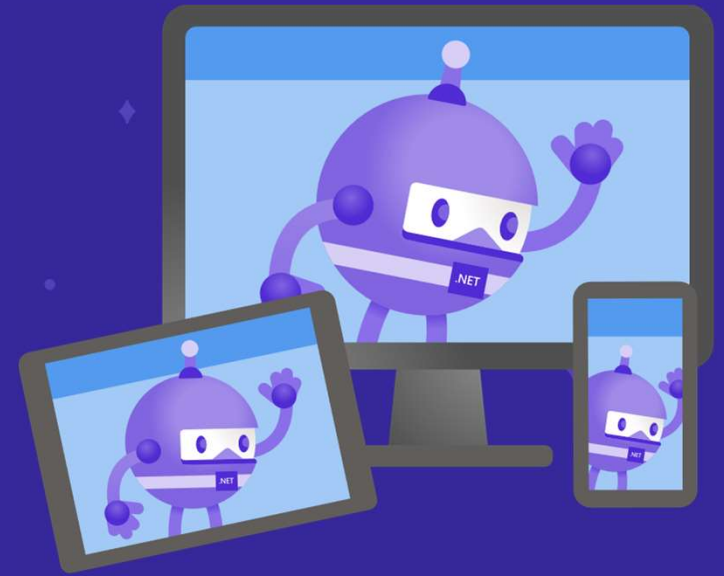
Aggiorna il record locale con un nuovo valore passato come argomento e riesegue il push verso il backend.

# .NET Conference 2024



# Thank you

Giampaolo TUCCI



# .NET Conference 2024



## VOTA LA SESSIONE

Facci sapere se questa sessione  
ti piace.  
Inquadra il QR code e esprimi  
una tua opinione.  
Ci aiuterai a migliorare.

