**SQL Server temporal tables in EF Core 6.0 and above version (Authenticity & Transparency)**

Temporal tables (also known as system-versioned temporal tables) are a database feature that brings built-in support for providing information about data stored in the table at any point in time, rather than only the data that is correct at the current moment in time.

**Temporal table usage scenarios & benefits:**

1. Data audit
2. Performing data analysis
3. Point-in-time analysis (time travel)
4. OLTP with auto-generated data history
5. Anomaly detection
6. Slowly changing dimensions
7. Repairing row-level data corruption

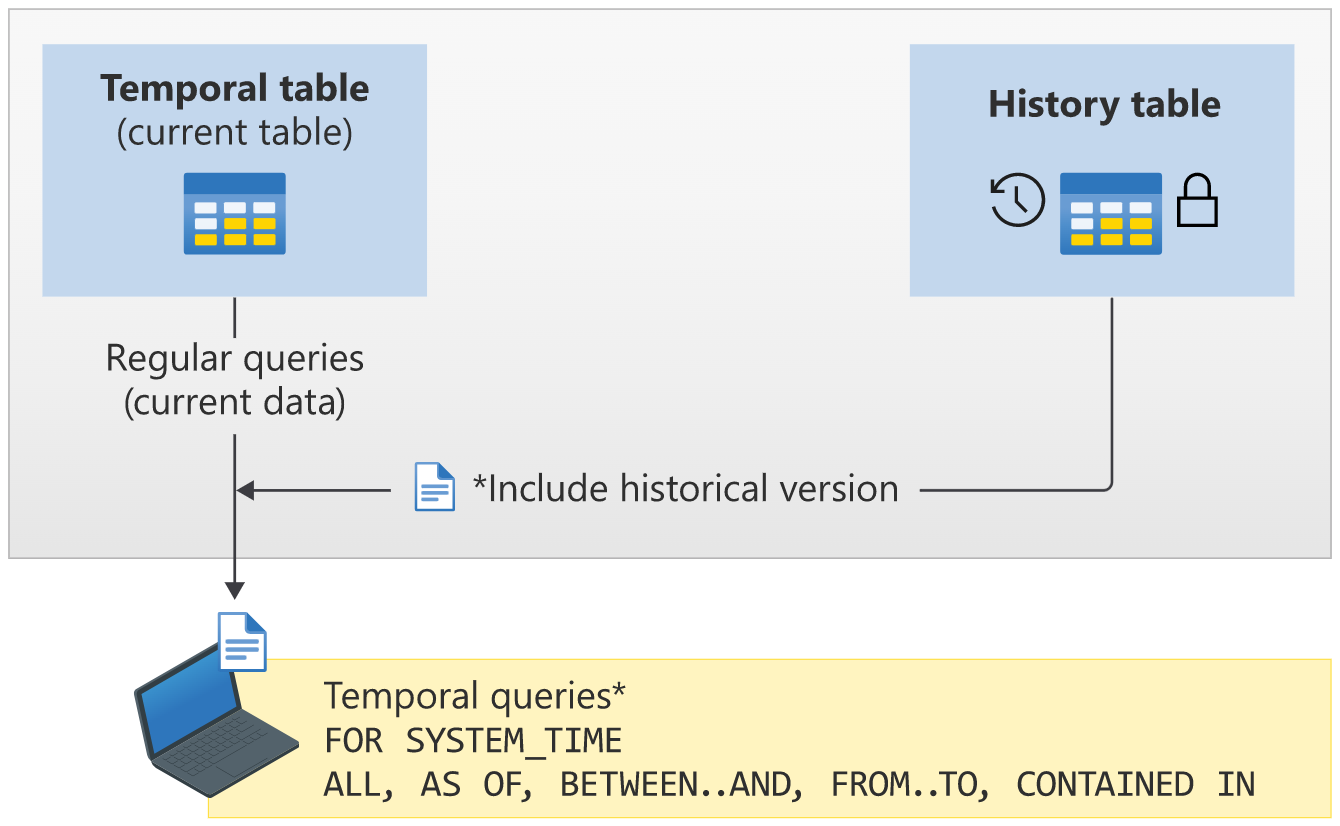
**Temporal table considerations and limitations:**

There are some considerations and limitations to be aware of when working with temporal tables, due to the nature of system-versioning:

1. A temporal table must have a primary key defined in order to correlate records between the current table and the history table, and the history table can't have a primary key defined.
2. The SYSTEM\_TIME period columns used to record the PeriodStart and PeriodEnd values must be defined with a datatype of datetime2.
3. By default, the history table is PAGE compressed.
4. A node or edge table can't be created as or altered to a temporal table.
5. History table must be created in the same database as the current table. Temporal querying over linked servers isn't supported.
6. History table can't have constraints (primary key, foreign key, table or column constraints).
7. TRUNCATE TABLE isn't supported while SYSTEM\_VERSIONING is ON.
8. Direct modification of the data in a history table isn't permitted.

**How do I query temporal data?**

The SELECT statement FROM <table> clause has a new clause FOR SYSTEM\_TIME with five temporal-specific sub-clauses to query data across the current and history tables.



SELECT \* FROM [Students]

FOR SYSTEM\_TIME

BETWEEN '2022-12-23 13:30:31.4538669' AND '2022-12-23 13:42:52.5924951'

WHERE Id='6B5A9538-97F6-4202-C71F-08DAE4E9DD07'

More details about querying a temporal table- <https://learn.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-table?view=sql-server-ver16>

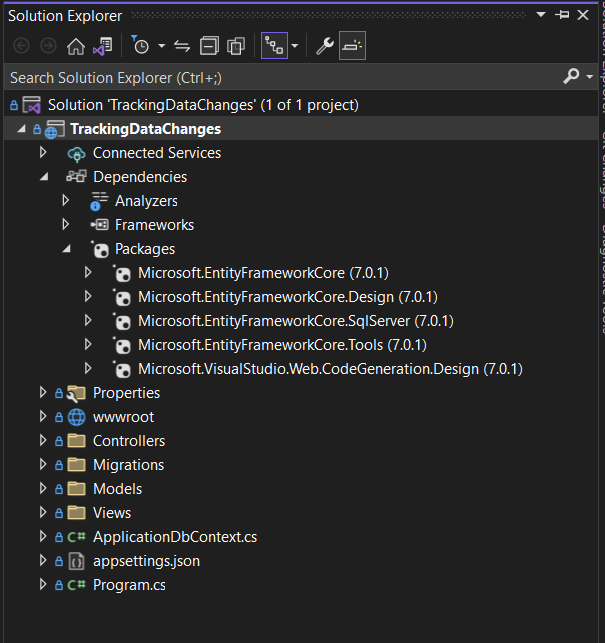
[**ASP.NET Core Web Application**](https://github.com/eshohag/TrackingDataChanges) **to learn how to build a temporal application using temporal tables.**

**Software Prerequisites:**

1. SQL Server 2016 (or higher)
2. ASP.NET Core 1.0 SDK (or higher). Optional: Visual Studio 2015 Update 3 (or higher) or Visual Studio Code Editor.

**1st Steps (ASP.NET Project Creation)**

Create an ASP.NET project of your convenience with Entity-Framework Core 6 or higher.



NuGet Package Manager Console:

Install-Package Microsoft.EntityFrameworkCore -Version 7.0.1

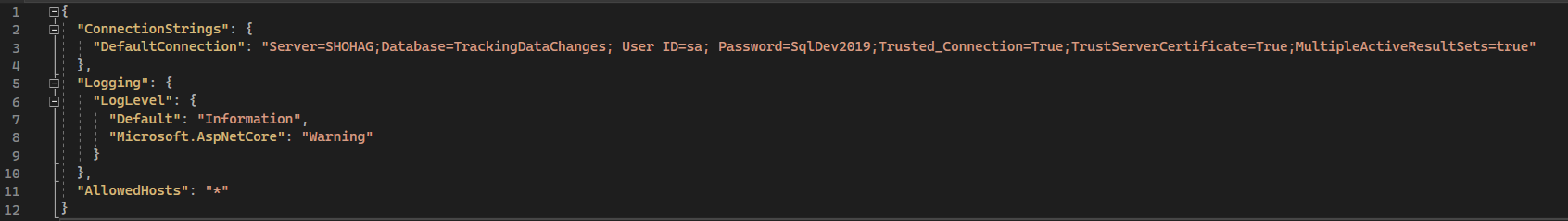
Install-Package Microsoft.EntityFrameworkCore.Design -Version 7.0.1

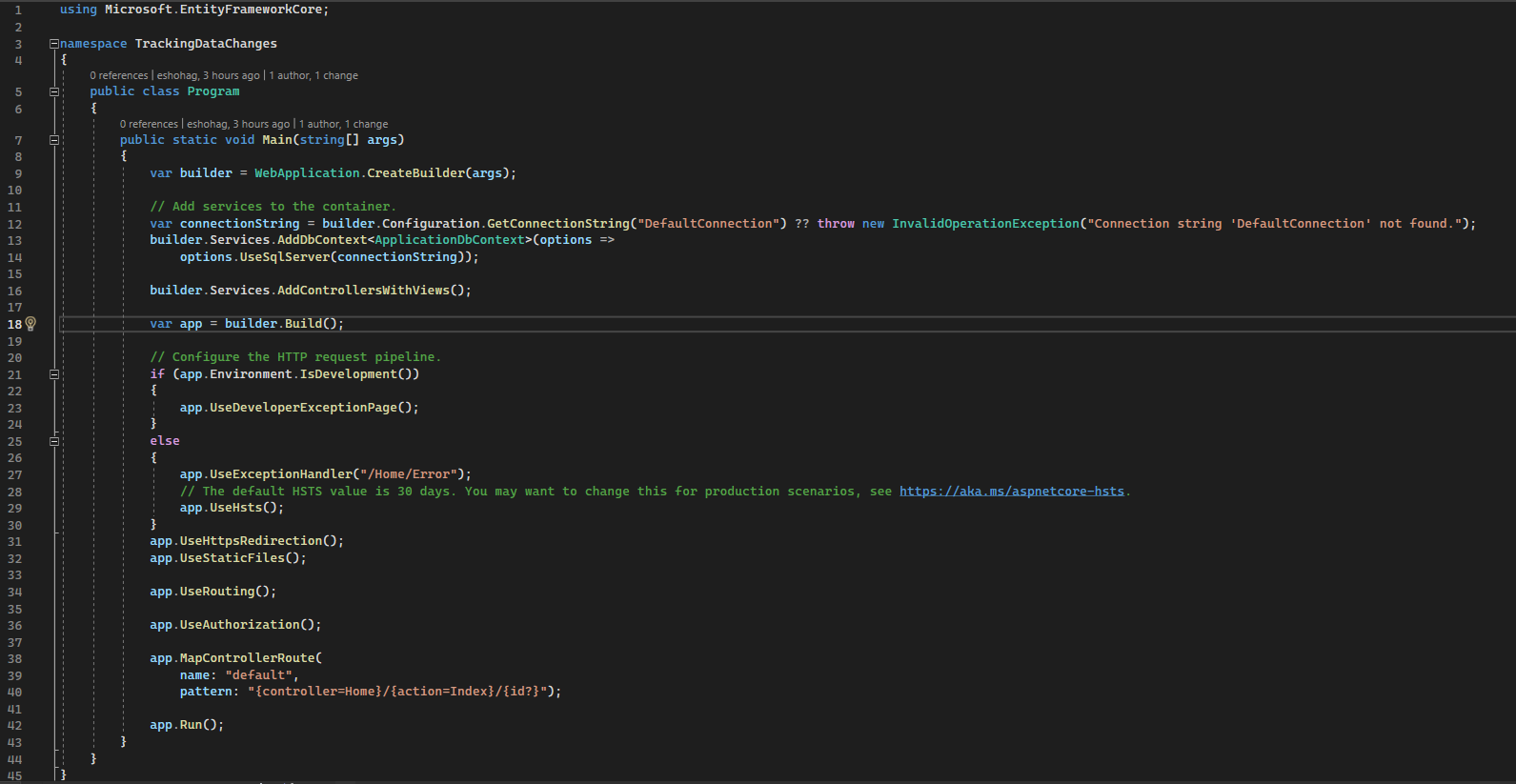
Install-Package Microsoft.EntityFrameworkCore.SqlServer -Version 7.0.1

Install-Package Microsoft.EntityFrameworkCore.Tools -Version 7.0.1

**2nd Steps (Connection String)**

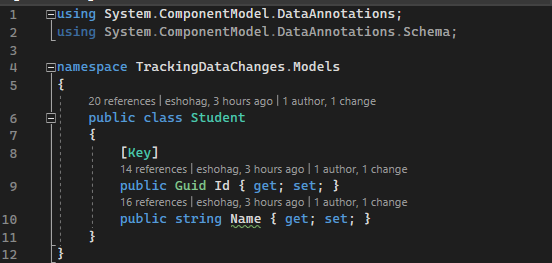
Configure your connections string & connect with your MSSQL Server DB

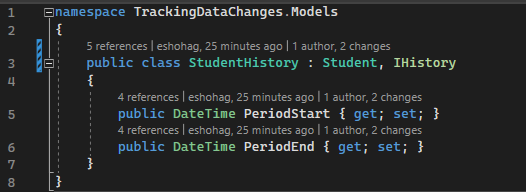




**3rd Steps (Models Creations)**

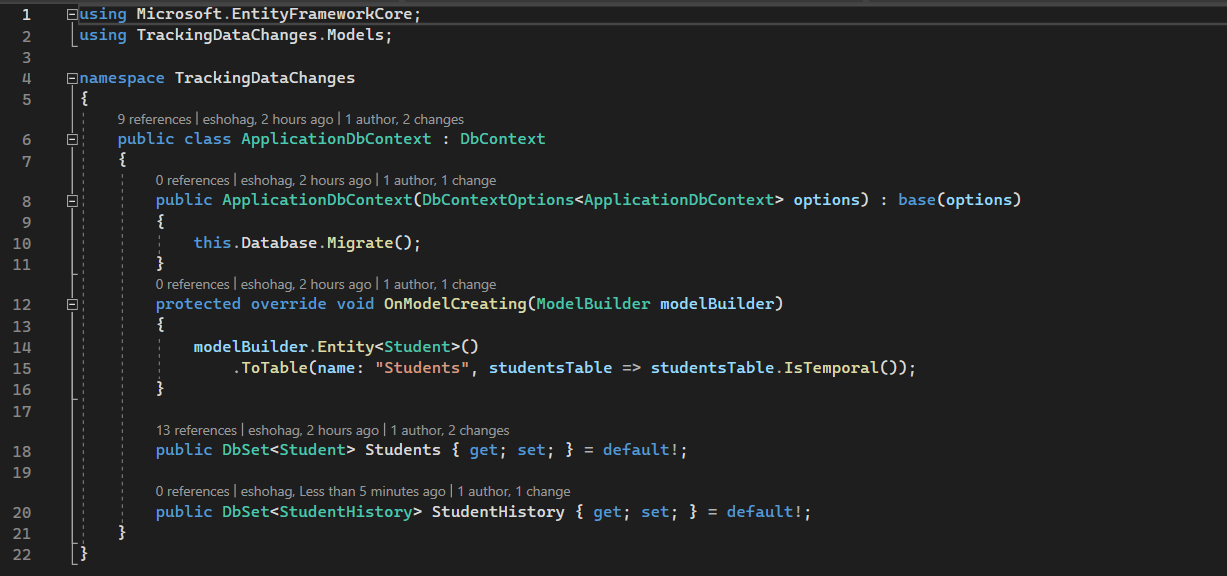
Define a model according to your business requirements as a temporal table.





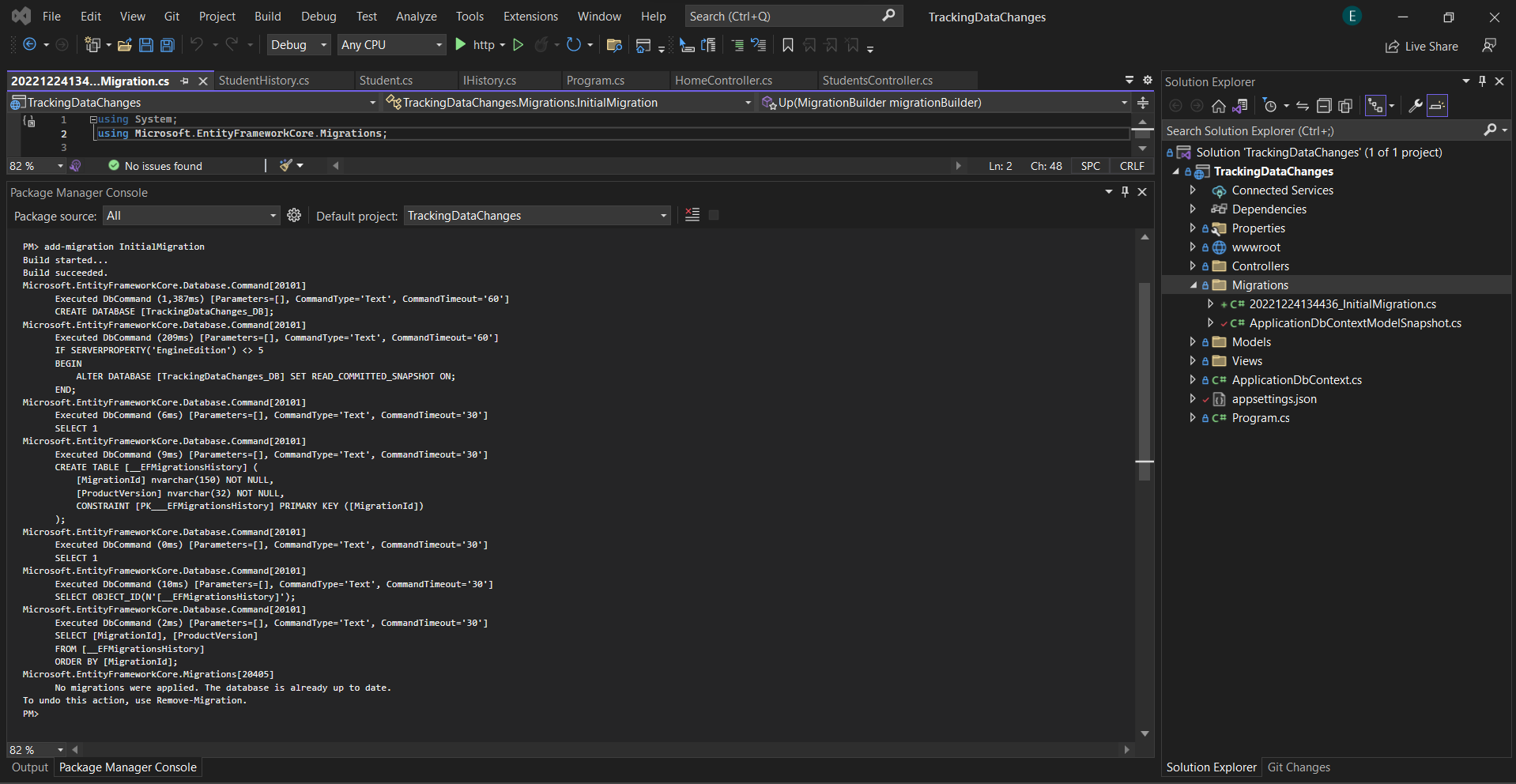
**4th Steps (Configuration temporal tables)**

Configuration with temporal table specified in OnModelCreating



**5th Steps (Database Creation)**

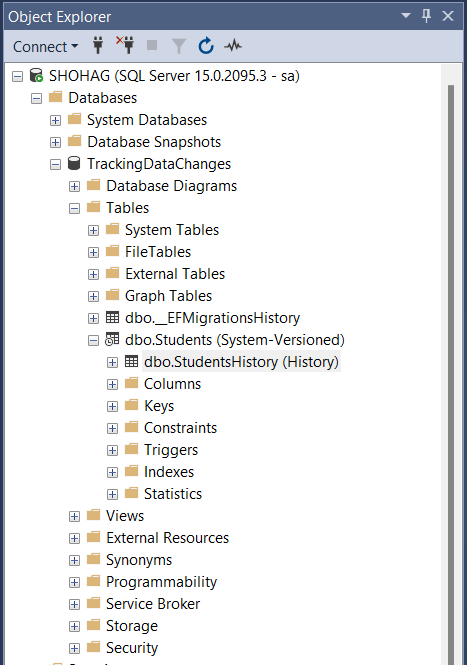
Add a migration to create a database.

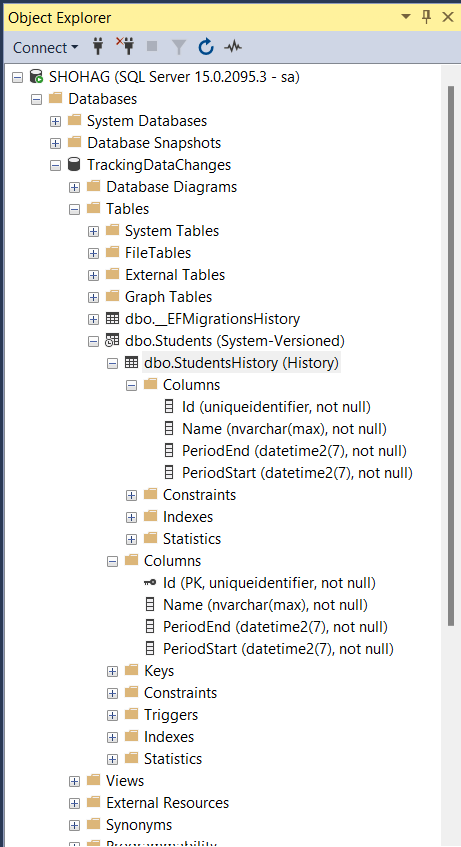


A computer screen capture

Description automatically generated with medium confidence

A system versioned table is automatically created inside the history table, as you see below:





PeriodStart and PeriodEnd are always the current UTC time, *not your server time*. Both columns are automatically created in the temporal table column.

**6th Steps (Fetching temporal table data)**

EF Core supports several temporal table query operators:

* TemporalAll
* TemporalAsOf
* TemporalFromTo
* TemporalBetween
* TemporalContainedIn

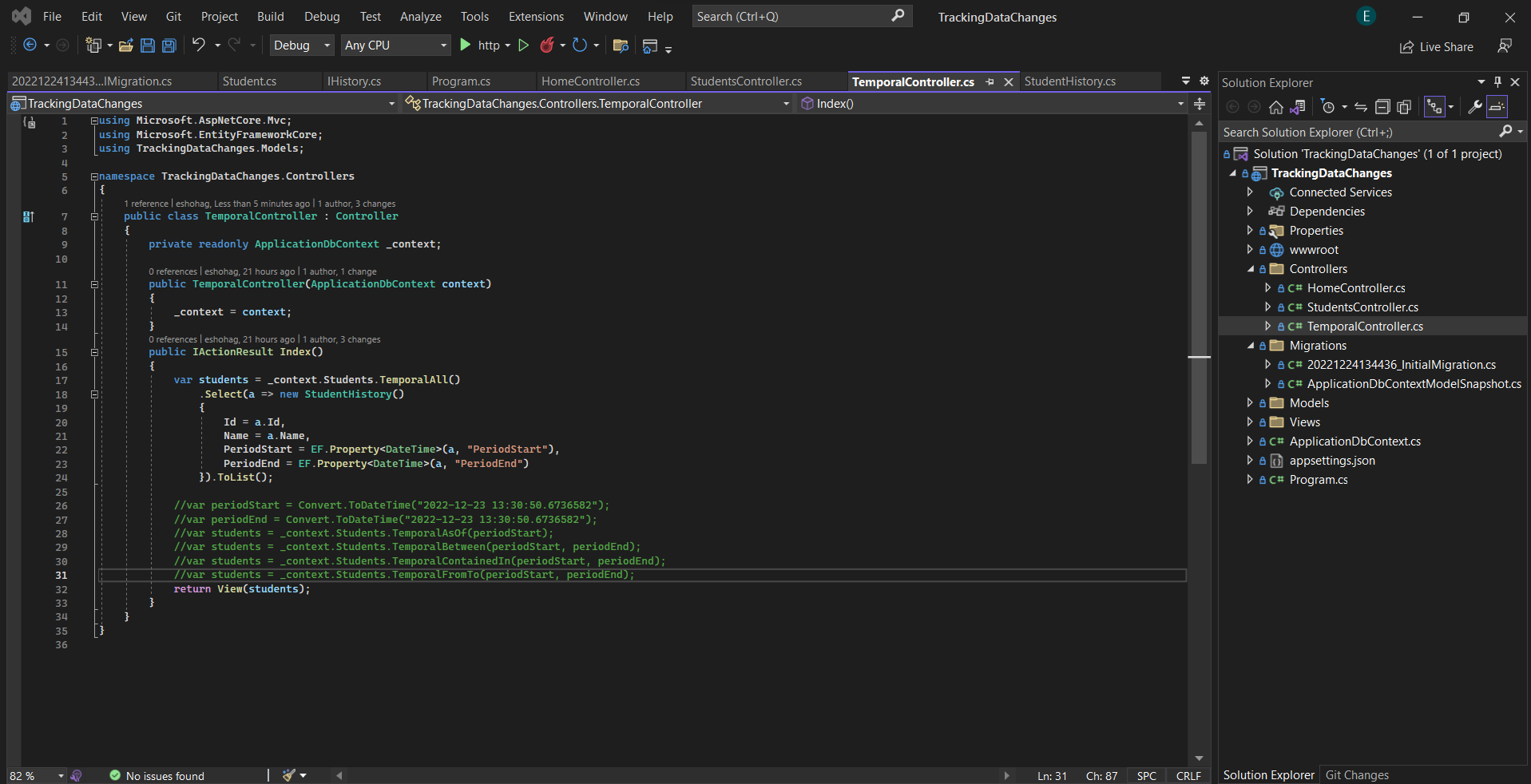
TemporalAll: Returns all rows in the historical data. This is typically many rows from the history table for a given primary key.

TemporalAsOf: Returns rows that were active (current) at the given UTC time. This is a single row from the history table for a given primary key.

TemporalFromTo: Returns all rows that were active between two given UTC times. This may be many rows from the history table for a given primary key.

TemporalBetween: The same as TemporalFromTo, except that rows are included that became active on the upper boundary.

TemporalContainedIn: Returns all rows that started being active and ended being active between two given UTC times. This may be many rows from the history table for a given primary key.



Source Code Link- [eshohag/TrackingDataChanges](https://github.com/eshohag/TrackingDataChanges)

The code included in this sample is not intended to demonstrate some general guidelines and architectural patterns for web development. You can easily modify this code to fit your application architecture.

More Details- <https://devblogs.microsoft.com/dotnet/prime-your-flux-capacitor-sql-server-temporal-tables-in-ef-core-6-0/>

<https://learn.microsoft.com/en-us/sql/relational-databases/tables/temporal-tables?view=sql-server-ver16>

Contact [eshohag@outlook.com](mailto:eshohag@outlook.com) with any additional questions or comments.