1-First of all install package: Microsoft.Extensions.cache.sqlserver

2-then we can create a Interface and a class like this and in constructor inject a IDistributedCache:

public interface IDistributedCachAdapter

{

T Get<T>(string Key);

void Set<T>(string key,T value);

}

public class DistributedCachAdapter : IDistributedCachAdapter

{

private readonly IDistributedCache distributedCache;

public DistributedCachAdapter(IDistributedCache distributedCache)

{

this.distributedCache = distributedCache;

}

public T Get<T>(string Key)

{

var result = distributedCache.GetString(Key);

return JsonConvert.DeserializeObject<T>(result);

}

public void Set<T>(string key, T value)

{

var result = JsonConvert.SerializeObject(value);

distributedCache.SetString(key, result);

}

}

3-then Add DistributedSqlServerCache in program.cs:

builder.Services.AddDistributedSqlServerCache(x =>

{

x.SchemaName = "dbo";

x.TableName = "ChacheTable";

x.ConnectionString = "Server=DESKTOP-FPKJ6AU\\SQL2022;Database=CacheDb;Trusted\_Connection=True;TrustServerCertificate=True";

});

4-in this phase we can use caching.here is a sample of using:

app.MapGet("/cacheTesting", async (HttpContext context, IDistributedCachAdapter cache) =>

{

int counter = 0;

string key = "myCounter";

counter = cache.Get<int>(key);

cache.Set(key, ++counter);

context.Response.WriteAsync(counter.ToString());

});