

=====

LINQ IMPLEMENTATION DOCUMENTATION
Utility Billing System - ASP.NET Core Web API

=====

=====

1. LINQ METHODS USED IN THIS PROJECT

=====

Method	Purpose
.Where()	Filtering data based on conditions
.Select()	Projecting/transforming data
.Include()	Eager loading related entities (EF Core)
.ThenInclude()	Nested eager loading
.FirstOrDefault()	Get first matching element or null
.OrderBy()	Sort ascending
.OrderByDescending()	Sort descending
.ThenBy()	Secondary sorting
.GroupBy()	Group data by key
.Sum()	Calculate total
.Count()	Count elements
.Average()	Calculate average
.Min() / .Max()	Find minimum/maximum
.Any()	Check if elements exist
.Skip() / .Take()	Pagination
.Distinct()	Remove duplicates
.ToDictionary()	Convert to Dictionary

=====

2. KEY LINQ EXAMPLES BY CATEGORY

=====

A. EAGER LOADING WITH INCLUDE & THENINCLUDE

File: BillService.cs (Lines 43-51)

```
var bill = await _context.Bills
    .Include(b => b.Connection)
        .ThenInclude(c => c.Consumer)
            .ThenInclude(co => co.User)
    .Include(b => b.Connection)
        .ThenInclude(c => c.UtilityType)
    .Include(b => b.Payment)
    .FirstOrDefaultAsync(b => b.Id == id);
```

B. FILTERED INCLUDE (Getting only latest meter reading)

File: ConnectionService.cs (Lines 22-27)

```
var connection = await _context.Connections
    .Include(c => c.Consumer).ThenInclude(co => co.User)
    .Include(c => c.UtilityType)
    .Include(c => c.MeterReadings.OrderByDescending(m => m.ReadingDate).Take(1))
    .FirstOrDefaultAsync(c => c.Id == id);
```

C. DYNAMIC FILTERING WITH WHERE

File: BillService.cs (Lines 111-139)

```
var query = _context.Bills.AsQueryable();

if (!string.IsNullOrEmpty(status))
    query = query.Where(b => b.Status == statusEnum);

if (billingMonth.HasValue)
    query = query.Where(b => b.BillingMonth == billingMonth.Value);

if (billingYear.HasValue)
    query = query.Where(b => b.BillingYear == billingYear.Value);

if (!string.IsNullOrEmpty(searchTerm))
    query = query.Where(b => b.BillNumber.Contains(searchTerm)
        || b.Connection.Consumer.ConsumerNumber.Contains(searchTerm));
```

D. PAGINATION WITH SKIP & TAKE

File: ConsumerService.cs (Lines 124-130)

```
var consumers = await query
    .OrderByDescending(c => c.CreatedAt)
    .Skip((paginationParams.PageNumber - 1) * paginationParams.PageSize)
    .Take(paginationParams.PageSize)
    .Select(c => new ConsumerListDto { ... })
    .ToListAsync();
```

E. PROJECTION WITH SELECT

File: ConnectionRequestService.cs (Lines 232-248)

```
var availableUtilities = await _context.UtilityTypes
    .Where(u => u.IsActive && !excludedUtilityIds.Contains(u.Id))
    .Select(u => new AvailableUtilityDto
    {
        Id = u.Id,
        Name = u.Name,
        TariffPlans = u.TariffPlans
            .Select(t => new AvailableTariffPlanDto
            {
                Id = t.Id,
                Name = t.Name,
                RatePerUnit = t.RatePerUnit
            })
            .ToList()
    })
    .ToListAsync();
```

F. GROUPBY WITH AGGREGATIONS

File: ReportService.cs (Lines 424-440)

```
var byUtilityType = readings
    .GroupBy(m => new {
        m.Connection.UtilityType.Name,
```

```
        m.Connection.UtilityType.UnitOfMeasurement  
    })  
.Select(g => new UtilityConsumptionDetailDto  
{  
    UtilityType = g.Key.Name,  
    Unit = g.Key.UnitOfMeasurement,  
    TotalConsumption = g.Sum(m => m.UnitsConsumed),  
    AverageConsumption = g.Average(m => m.UnitsConsumed),  
    MinConsumption = g.Min(m => m.UnitsConsumed),  
    MaxConsumption = g.Max(m => m.UnitsConsumed),  
    ConnectionCount = g.Count()  
})  
.ToList();
```

G. GROUPBY WITH TODICTIONARY

File: PaymentService.cs (Lines 315-318)

```
var summary = new PaymentSummaryDto  
{  
    TotalAmount = payments.Sum(p => p.Amount),  
    ByMethod = payments  
        .GroupBy(p => p.PaymentMethod.ToString())  
        .ToDictionary(g => g.Key, g => g.Sum(p => p.Amount))  
};
```

H. AGGREGATION FUNCTIONS (SUM, COUNT)

File: BillService.cs (Lines 455-466)

```
var summary = new BillSummaryDto  
{  
    TotalBilledAmount = bills.Sum(b => b.TotalAmount),  
    TotalCollected = bills.Sum(b => b.AmountPaid),  
    TotalOutstanding = bills.Sum(b => b.OutstandingBalance),  
    TotalBills = bills.Count,  
    PaidBills = bills.Count(b => b.Status == BillStatus.Paid),  
    OverdueBills = bills.Count(b => b.Status != BillStatus.Paid && b.DueDate < today)  
};
```

I. ANY() FOR EXISTENCE CHECK

File: ConnectionService.cs (Lines 275-285)

```
var pendingBills = connection.Bills.Where(b => b.OutstandingBalance > 0).ToList();  
  
if (pendingBills.Any())  
{  
    var totalOutstanding = pendingBills.Sum(b => b.OutstandingBalance);  
    return Error($"Cannot deactivate. Outstanding: {totalOutstanding}");  
}  
  
if (connection.Bills.Any() || connection.MeterReadings.Any())  
    // Soft delete instead of hard delete
```

J. DISTINCT FOR UNIQUE VALUES

File: ConnectionRequestService.cs (Lines 210-226)

```
var existingUtilities = await _context.Connections
    .Where(c => c.ConsumerId == consumerId)
    .Select(c => c.UtilityTypeId)
    .ToListAsync();

var pendingUtilities = await _context.ConnectionRequests
    .Where(r => r.ConsumerId == consumerId && r.Status == Status.Pending)
    .Select(r => r.UtilityTypeId)
    .ToListAsync();

var excludedIds = existingUtilities
    .Concat(pendingUtilities)
    .Distinct()
    .ToList();
```

K. ORDERING WITH ORDERBY & THENBY

File: AuthService.cs (Lines 263-265)

```
var users = await _context.Users
    .Include(u => u.Consumer)
    .OrderBy(u => u.LastName)
    .ThenBy(u => u.FirstName)
    .ToListAsync();
```

L. TOP N RECORDS WITH TAKE

File: ReportService.cs (Lines 443-455)

```
var topConsumers = readings
    .OrderByDescending(m => m.UnitsConsumed)
    .Take(10)
    .Select(m => new TopConsumerDto
    {
        ConsumerName = $"{m.Connection.Consumer.User.FirstName} {m.Connection.Consumer.User.LastName}",
        UtilityType = m.Connection.UtilityType.Name,
        Consumption = m.UnitsConsumed
    })
    .ToList();
```

=====
3. FILES WHERE LINQ IS IMPLEMENTED
=====

Service File	Key LINQ Operations Used
AuthService.cs	Include, Where, Select, FirstOrDefault
BillService.cs	Include, Where, Sum, Count, GroupBy
BillingCycleService.cs	Include, Where, OrderBy
ConnectionService.cs	Filtered Include, Where, Any, Sum
ConnectionRequestService.cs	Distinct, Concat, Select, Where
ConsumerService.cs	Skip, Take, OrderBy, ThenBy
MeterReadingService.cs	Where, OrderByDescending, ThenBy
NotificationService.cs	Where, Take, OrderByDescending
PaymentService.cs	GroupBy, ToDictionary, Sum

ReportService.cs	GroupBy, Average, Min, Max, Take	
TariffPlanService.cs	Include, Where, OrderBy, Any	
UtilityTypeService.cs	Include, Where, Any, OrderBy	

=====

END OF DOCUMENTATION

=====