```
================================================================================
                    LINQ IMPLEMENTATION REFERENCE DOCUMENT
                    Utility Management System - .NET Backend
================================================================================

This document provides a comprehensive overview of all LINQ (Language Integrated
Query) implementations used throughout the Utility Management API project.

================================================================================
TABLE OF CONTENTS
================================================================================
1. BillService.cs
2. ConsumerService.cs
3. ConnectionService.cs
4. ConnectionRequestService.cs
5. MeterReadingService.cs
6. PaymentService.cs
7. TariffPlanService.cs
8. UtilityTypeService.cs
9. BillingCycleService.cs
10. NotificationService.cs
11. ReportService.cs

================================================================================
1. BILL SERVICE (BillService.cs)
================================================================================
Location: Services/Implementations/BillService.cs

LINQ Methods Used:
- Include() / ThenInclude() - Eager loading related entities
- FirstOrDefaultAsync() - Finding single records
- Where() - Filtering data
- OrderByDescending() / ThenByDescending() - Sorting
- Skip() / Take() - Pagination
- Select() - Projection to DTOs
- ToListAsync() - Materializing queries
- AnyAsync() - Existence checks
- CountAsync() - Counting records
- Distinct() - Removing duplicates
- SumAsync() - Aggregation

EXAMPLES:

a) Eager Loading with Include/ThenInclude (Lines 41-50):
   ------------------------------------------------------
   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.Connection)
           .ThenInclude(c => c.TariffPlan)
       .Include(b => b.Payment)
       .FirstOrDefaultAsync(b => b.BillNumber == billNumber);

b) Filtering with Where (Lines 71-72):
   ------------------------------------
   .Where(b => b.ConnectionId == connectionId)
```

```
   .OrderByDescending(b => b.BillingYear)

c) Multiple Filters with Conditional Where (Lines 117-134):
   ------------------------------------------------------
   if (!string.IsNullOrEmpty(status))
       query = query.Where(b => b.Status == statusEnum);

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

   if (!string.IsNullOrEmpty(paginationParams.SearchTerm))
       query = query.Where(b =>
           b.BillNumber.ToLower().Contains(searchTerm) ||
           b.Connection.Consumer.ConsumerNumber.ToLower().Contains(searchTerm) ||
           b.Connection.Consumer.User.FirstName.ToLower().Contains(searchTerm));

d) Pagination with Skip/Take (Lines 150-155):
   ----------------------------------------
   var bills = await query
       .OrderByDescending(b => b.BillingYear)
       .ThenByDescending(b => b.BillingMonth)
       .Skip((paginationParams.PageNumber - 1) * paginationParams.PageSize)
       .Take(paginationParams.PageSize)
       .Select(b => new BillListDto { ... })
       .ToListAsync();

e) Existence Check with AnyAsync (Lines 207-212):
   ---------------------------------------------
   if (await _context.Bills.AnyAsync(b =>
       b.ConnectionId == dto.ConnectionId &&
       b.BillingMonth == dto.BillingMonth &&
       b.BillingYear == dto.BillingYear))

f) Bulk Bill Generation with Distinct (Lines 318-328):
   -------------------------------------------------
   connectionIds = await _context.MeterReadings
       .Where(m => m.BillingMonth == dto.BillingMonth &&
                   m.BillingYear == dto.BillingYear &&
                   !billedMeterReadingIds.Contains(m.Id))
       .Select(m => m.ConnectionId)
       .Distinct()
       .ToListAsync();

================================================================================
2. CONSUMER SERVICE (ConsumerService.cs)
================================================================================
Location: Services/Implementations/ConsumerService.cs

LINQ Methods Used:
- Include() / ThenInclude() - Eager loading
- FirstOrDefaultAsync() - Single record retrieval
- Where() - Filtering
- OrderBy() / OrderByDescending() / ThenBy() - Sorting
- Skip() / Take() - Pagination
- Select() - Projection
- CountAsync() - Counting

EXAMPLES:

a) Complex Include Chain (Lines 24-30):
```

```
   ------------------------------------
   var consumer = await _context.Consumers
       .Include(c => c.User)
       .Include(c => c.Connections)
           .ThenInclude(conn => conn.UtilityType)
       .Include(c => c.Connections)
           .ThenInclude(conn => conn.TariffPlan)
       .FirstOrDefaultAsync(c => c.Id == id);

b) Dynamic Sorting with Switch Expression (Lines 99-109):
   ------------------------------------------------------
   query = paginationParams.SortBy?.ToLower() switch
   {
       "name" => paginationParams.SortDescending
           ? query.OrderByDescending(c => c.User.LastName)
                  .ThenByDescending(c => c.User.FirstName)
           : query.OrderBy(c => c.User.LastName)
                  .ThenBy(c => c.User.FirstName),
       "consumernumber" => paginationParams.SortDescending
           ? query.OrderByDescending(c => c.ConsumerNumber)
           : query.OrderBy(c => c.ConsumerNumber),
       _ => query.OrderByDescending(c => c.CreatedAt),
   };

c) Projection to DTO with Nested Count (Lines 116-128):
   ------------------------------------------------
   var consumers = await query
       .Skip((paginationParams.PageNumber - 1) * paginationParams.PageSize)
       .Take(paginationParams.PageSize)
       .Select(c => new ConsumerListDto
       {
           Id = c.Id,
           ConsumerNumber = c.ConsumerNumber,
           FullName = $"{c.User.FirstName} {c.User.LastName}",
           TotalConnections = c.Connections.Count,  // Nested Count
           IsActive = c.IsActive,
       })
       .ToListAsync();

================================================================================
3. CONNECTION SERVICE (ConnectionService.cs)
================================================================================
Location: Services/Implementations/ConnectionService.cs

LINQ Methods Used:
- Include() / ThenInclude() - Eager loading
- OrderByDescending() - Sorting within Include
- Take() - Limiting results within Include
- FirstOrDefaultAsync() - Single record
- Where() - Filtering
- Select() - Projection

EXAMPLES:

a) Filtered Include with OrderBy and Take (Lines 21-27):
   ------------------------------------------------
   var connection = await _context.Connections
       .Include(c => c.Consumer)
           .ThenInclude(co => co.User)
```

```
        .Include(c => c.UtilityType)
        .Include(c => c.TariffPlan)
        .Include(c => c.MeterReadings.OrderByDescending(m => m.ReadingDate).Take(1))
        .FirstOrDefaultAsync(c => c.Id == id);

b) Multiple Condition Filter (Lines 92-99):
   ----------------------------------------
   if (!string.IsNullOrEmpty(paginationParams.SearchTerm))
   {
       query = query.Where(c =>
           c.ConnectionNumber.ToLower().Contains(searchTerm) ||
           c.MeterNumber.ToLower().Contains(searchTerm) ||
           c.Consumer.ConsumerNumber.ToLower().Contains(searchTerm) ||
           c.Consumer.User.FirstName.ToLower().Contains(searchTerm));
   }

================================================================================
4. CONNECTION REQUEST SERVICE (ConnectionRequestService.cs)
================================================================================
Location: Services/Implementations/ConnectionRequestService.cs

LINQ Methods Used:
- Where() - Filtering
- Select() - Projection
- Distinct() - Removing duplicates
- OrderByDescending() / OrderBy() - Sorting
- Skip() / Take() - Pagination
- Contains() - Collection membership check
- Concat() - Combining collections

EXAMPLES:

a) Getting Available Utilities (Exclude already connected) (Lines 210-231):
   ------------------------------------------------------------------------
   var existingConnectionUtilityIds = await _context.Connections
       .Where(c => c.ConsumerId == consumer.Id &&
                   (c.Status == ConnectionStatus.Active || c.Status == ConnectionStatus.Suspended
       .Select(c => c.UtilityTypeId)
       .ToListAsync();

   var pendingRequestUtilityIds = await _context.ConnectionRequests
       .Where(r => r.ConsumerId == consumer.Id &&
                   r.Status == ConnectionRequestStatus.Pending)
       .Select(r => r.UtilityTypeId)
       .ToListAsync();

   var excludedUtilityIds = existingConnectionUtilityIds
       .Concat(pendingRequestUtilityIds)
       .Distinct()
       .ToList();

   var availableUtilities = await _context.UtilityTypes
       .Where(u => u.IsActive && !excludedUtilityIds.Contains(u.Id))
       .Include(u => u.TariffPlans.Where(t => t.IsActive))
       .Select(u => new AvailableUtilityDto { ... })
       .ToListAsync();

b) Complex Projection with Nested Select (Lines 232-248):
   -------------------------------------------------------
   .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,
           FixedCharges = t.FixedCharges,
       }).ToList(),
   })

c) Generating Sequential Numbers (Lines 459-460):
   ---------------------------------------------
   var lastRequest = await _context.ConnectionRequests
       .Where(r => r.RequestNumber.StartsWith(prefix))
       .OrderByDescending(r => r.RequestNumber)
       .FirstOrDefaultAsync();

================================================================================
5. METER READING SERVICE (MeterReadingService.cs)
================================================================================
Location: Services/Implementations/MeterReadingService.cs

LINQ Methods Used:
- Include() / ThenInclude() - Deep eager loading
- Where() - Filtering
- Select() - Projection
- ToListAsync() - Materialization

EXAMPLES:

a) Deep Include Chain (Lines 28-34):
   ---------------------------------
   await _context.MeterReadings
       .Include(m => m.Connection)
           .ThenInclude(c => c.Consumer)
               .ThenInclude(co => co.User)
       .Include(m => m.Connection)
           .ThenInclude(c => c.UtilityType)
       .Include(m => m.ReadByUser)
       .Include(m => m.Bill)

b) Filter by Connection (Lines 55-61):
   -----------------------------------
   .Where(m => m.ConnectionId == connectionId)
   ...
   readings.Select(MapToDto).ToList()

================================================================================
6. PAYMENT SERVICE (PaymentService.cs)
================================================================================
Location: Services/Implementations/PaymentService.cs

LINQ Methods Used:
- Where() - Filtering
- OrderByDescending() - Sorting
- Sum() - Aggregation
- GroupBy() - Grouping
- ToDictionary() - Converting to dictionary
```

EXAMPLES:

a) Multiple Date Range Filters (Lines 113-123):
```
   ---------------------------------------------
   if (!string.IsNullOrEmpty(paymentMethod))
       query = query.Where(p => p.PaymentMethod == method);

   if (fromDate.HasValue)
       query = query.Where(p => p.PaymentDate >= fromDate.Value);

   if (toDate.HasValue)
       query = query.Where(p => p.PaymentDate <= toDate.Value);
```

b) Payment Summary with GroupBy and Sum (Lines 315-322):
```
   -----------------------------------------------------
   var summary = new PaymentSummaryDto
   {
       TotalAmount = payments.Sum(p => p.Amount),
       PaymentMethodBreakdown = payments
           .GroupBy(p => p.PaymentMethod.ToString())
           .ToDictionary(g => g.Key, g => g.Sum(p => p.Amount)),
       TodayCollection = payments
           .Where(p => p.PaymentDate.Date == today)
           .Sum(p => p.Amount),
       MonthCollection = payments
           .Where(p => p.PaymentDate >= startOfMonth)
           .Sum(p => p.Amount),
   };
```

==
7. TARIFF PLAN SERVICE (TariffPlanService.cs)
==
Location: Services/Implementations/TariffPlanService.cs

LINQ Methods Used:
- Where() - Filtering
- OrderBy() / ThenBy() - Sorting
- Select() - Projection
- Skip() / Take() - Pagination
- Count() - Counting with predicate

EXAMPLES:

a) Chained Filtering and Sorting (Lines 46-57):
```
   ---------------------------------------------
   if (isActive.HasValue)
       query = query.Where(t => t.IsActive == isActive.Value);

   if (utilityTypeId.HasValue)
       query = query.Where(t => t.UtilityTypeId == utilityTypeId.Value);

   var tariffPlans = await query
       .OrderBy(t => t.UtilityType.Name)
       .ThenBy(t => t.Name)
       .Select(t => MapToDto(t))
       .ToListAsync();
```

b) Count with Predicate (Lines 236, 292):
```
   ---------------------------------------
```

```
   var activeConnectionsCount = tariffPlan.Connections.Count(c =>
       c.Status == ConnectionStatus.Active);

   tariffPlan.Connections?.Count(c => c.Status == ConnectionStatus.Active) ?? 0
```

```
================================================================================
8. UTILITY TYPE SERVICE (UtilityTypeService.cs)
================================================================================
Location: Services/Implementations/UtilityTypeService.cs

LINQ Methods Used:
- Where() - Conditional filtering
- OrderBy() - Sorting
- Select() - Projection
- Any() - Existence check on collections

EXAMPLES:

a) Conditional Query Building (Lines 43-46):
```

```
   ----------------------------------------
   if (isActive.HasValue)
       query = query.Where(u => u.IsActive == isActive.Value);

   var utilityTypes = await query
       .OrderBy(u => u.Name)
       .Select(u => MapToDto(u))
       .ToListAsync();
```

```
b) Checking Related Data Before Delete (Lines 167-174):
```

```
   ----------------------------------------------------
   if (utilityType.Connections.Any())
       return ApiResponse<bool>.ErrorResponse("Cannot delete: active connections exist");

   if (utilityType.TariffPlans.Any())
       return ApiResponse<bool>.ErrorResponse("Cannot delete: tariff plans exist");
```

```
================================================================================
9. BILLING CYCLE SERVICE (BillingCycleService.cs)
================================================================================
Location: Services/Implementations/BillingCycleService.cs

LINQ Methods Used:
- Include() - Eager loading
- FirstOrDefaultAsync() - Single record
- Where() - Filtering
- OrderByDescending() / ThenByDescending() - Sorting
- AnyAsync() - Existence check

EXAMPLES:

a) Get Current Billing Cycle (Lines 42-46):
```

```
   ----------------------------------------
   var cycle = await _context.BillingCycles
       .Include(bc => bc.CreatedByUser)
       .FirstOrDefaultAsync(bc => bc.Month == currentMonth && bc.Year == currentYear);
```

```
b) Sorted List Query (Lines 64-68):
```

```
   ---------------------------------
   var cycles = await query
       .OrderByDescending(bc => bc.Year)
```

```
        .ThenByDescending(bc => bc.Month)
        .ToListAsync();

================================================================================
10. NOTIFICATION SERVICE (NotificationService.cs)
================================================================================
Location: Services/Implementations/NotificationService.cs

LINQ Methods Used:
- Where() - Filtering
- OrderByDescending() - Sorting
- Take() - Limiting results
- Select() - Projection
- CountAsync() - Counting

EXAMPLES:

a) Get User Notifications with Filters (Lines 33-45):
   ------------------------------------------------
   var query = _context.Notifications
       .Where(n => n.UserId == userId)
       .AsQueryable();

   if (isRead.HasValue)
       query = query.Where(n => n.IsRead == isRead.Value);

   var notifications = await query
       .OrderByDescending(n => n.CreatedAt)
       .Take(100)
       .Select(n => MapToDto(n))
       .ToListAsync();

b) Get Unread Notifications for Batch Update (Lines 109-113):
   ------------------------------------------------------------
   var notifications = await _context.Notifications
       .Where(n => n.UserId == userId && !n.IsRead)
       .ToListAsync();

================================================================================
11. REPORT SERVICE (ReportService.cs) - MOST COMPREHENSIVE LINQ USAGE
================================================================================
Location: Services/Implementations/ReportService.cs

LINQ Methods Used:
- Where() - Filtering
- Select() - Projection
- GroupBy() - Grouping data
- Sum() / Average() / Min() / Max() - Aggregations
- Count() - Counting
- Distinct() - Unique values
- OrderBy() / OrderByDescending() - Sorting
- Take() - Limiting results
- Concat() - Combining sequences
- ToDictionary() - Converting to dictionary
- ContainsKey() - Dictionary lookup

EXAMPLES:

a) Dashboard Summary - Multiple Aggregations (Lines 39-49):
   ------------------------------------------------------
```

```
    var revenueThisMonth = await _context.Payments
        .Where(p => p.PaymentDate.Month == currentMonth &&
                    p.PaymentDate.Year == currentYear &&
                    p.Status == PaymentStatus.Completed)
        .SumAsync(p => p.Amount);

    var totalOutstanding = await _context.Bills
        .Where(b => b.OutstandingBalance > 0)
        .SumAsync(b => b.OutstandingBalance);
```

b) Complex GroupBy with Aggregations (Lines 126-135):

```
    ------------------------------------------------------
    var consumptionGroups = meterReadingsWithUtility
        .GroupBy(m => new { m.UtilityTypeName, m.Unit })
        .ToDictionary(
            g => g.Key.UtilityTypeName,
            g => new
            {
                TotalConsumption = g.Sum(m => m.UnitsConsumed),
                ConnectionCount = g.Select(m => m.ConnectionId).Distinct().Count(),
                Unit = g.Key.Unit,
            });
```

c) Combining Activities with Concat and Sorting (Lines 99-103):

```
    ------------------------------------------------------------
    var recentActivities = recentBills
        .Concat(recentPayments)
        .OrderByDescending(a => a.Timestamp)
        .Take(10)
        .ToList();
```

d) Monthly Revenue Report with GroupBy (Lines 236-248):

```
    ------------------------------------------------------
    var byUtilityType = bills
        .GroupBy(b => b.Connection.UtilityType.Name)
        .Select(g => new UtilityRevenueDto
        {
            UtilityType = g.Key,
            BilledAmount = g.Sum(b => b.TotalAmount),
            Collected = payments
                .Where(p => p.Bill.Connection.UtilityType.Name == g.Key)
                .Sum(p => p.Amount),
            BillCount = g.Count(),
        })
        .ToList();
```

e) Outstanding Dues - Age Buckets with Where and Sum (Lines 316-348):

```
    ---------------------------------------------------------------------
    var ageBuckets = new List<OutstandingByAgeDto>
    {
        new OutstandingByAgeDto
        {
            AgeBucket = "Current (0-30 days)",
            Amount = outstandingBills
                .Where(b => today.DayNumber - b.DueDate.DayNumber <= 30)
                .Sum(b => b.OutstandingBalance),
            Count = outstandingBills
                .Count(b => today.DayNumber - b.DueDate.DayNumber <= 30),
        },
```

```
        // ... more buckets for 31-60, 61-90, Over 90 days
    };
```

f) Top Defaulters with GroupBy and Complex Projection (Lines 358-378):

```
    ---------------------------------------------------------------------
    var topDefaulters = outstandingBills
        .GroupBy(b => new
        {
            b.Connection.ConsumerId,
            b.Connection.Consumer.ConsumerNumber,
            ConsumerName = $"{b.Connection.Consumer.User.FirstName} {b.Connection.Consumer.User.La
        })
        .Select(g => new ConsumerOutstandingDto
        {
            ConsumerId = g.Key.ConsumerId,
            ConsumerNumber = g.Key.ConsumerNumber,
            PenaltyAmount = g.Sum(b => b.PenaltyAmount),
            DueAmount = g.Sum(b => b.OutstandingBalance - b.PenaltyAmount),
            OutstandingAmount = g.Sum(b => b.OutstandingBalance),
            OverdueBills = g.Count(b => b.DueDate < today),
            OldestDueDate = g.Min(b => b.DueDate),
        })
        .OrderByDescending(d => d.OutstandingAmount)
        .Take(10)
        .ToList();
```

g) Consumption Report with All Aggregations (Lines 424-440):

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

h) Consumer Billing Summary with Nested LINQ (Lines 497-510):

```
    ------------------------------------------------------------
    var connectionSummaries = consumer.Connections
        .Select(conn => new ConnectionBillingSummaryDto
        {
            ConnectionId = conn.Id,
            BillCount = bills.Count(b => b.ConnectionId == conn.Id),
            TotalBilled = bills.Where(b => b.ConnectionId == conn.Id).Sum(b => b.TotalAmount),
            TotalPaid = bills.Where(b => b.ConnectionId == conn.Id).Sum(b => b.AmountPaid),
            Outstanding = bills.Where(b => b.ConnectionId == conn.Id).Sum(b => b.OutstandingBalanc
            TotalConsumption = readings.Where(r => r.ConnectionId == conn.Id).Sum(r => r.UnitsCons
        })
        .ToList();
```

i) Collection Report - Daily Grouping (Lines 542-553):

```
    ---------------------------------------------------
    var dailyCollections = payments
```

```
        .GroupBy(p => p.PaymentDate.Date)
        .Select(g => new DailyCollectionDto
        {
            Date = g.Key,
            Amount = g.Sum(p => p.Amount),
            PaymentCount = g.Count(),
        })
        .OrderBy(d => d.Date)
        .ToList();
```

j) Payment Method Breakdown with Percentage (Lines 558-567):

```
    ------------------------------------------------------------
    var byPaymentMethod = payments
        .GroupBy(p => p.PaymentMethod.ToString())
        .Select(g => new PaymentMethodCollectionDto
        {
            PaymentMethod = g.Key,
            Amount = g.Sum(p => p.Amount),
            Count = g.Count(),
            Percentage = totalCollected > 0 ? (g.Sum(p => p.Amount) / totalCollected) * 100 : 0,
        })
        .ToList();
```

```
================================================================================
SUMMARY OF LINQ METHODS USED
================================================================================

| Method                | Description                                    | Files Used
|-----------------------|------------------------------------------------|-----------------------
| Include()             | Eager loading related entities                 | All Service Files
| ThenInclude()         | Nested eager loading                           | All Service Files
| Where()               | Filtering records                              | All Service Files
| Select()              | Projection to DTOs/anonymous types             | All Service Files
| FirstOrDefaultAsync()| Get single record or null                      | All Service Files
| ToListAsync()         | Materialize query to list                      | All Service Files
| OrderBy()             | Ascending sort                                 | Most Service Files
| OrderByDescending()   | Descending sort                                | Most Service Files
| ThenBy()              | Secondary ascending sort                       | Consumer, Tariff Servic
| ThenByDescending()    | Secondary descending sort                      | Bill, BillingCycle Serv
| Skip()                | Skip records for pagination                    | Paginated queries
| Take()                | Take limited records                           | Paginated queries
| CountAsync()          | Count records asynchronously                   | Paginated, Dashboard qu
| Count()               | Count with predicate                           | TariffPlan, Report Serv
| AnyAsync()            | Check existence asynchronously                 | Bill, BillingCycle Serv
| Any()                 | Check if collection has elements               | UtilityType Service
| Sum() / SumAsync()    | Sum aggregation                                | Payment, Report Service
| Average()             | Average aggregation                            | Report Service
| Min()                 | Minimum value                                  | Report Service
| Max()                 | Maximum value                                  | Report Service
| GroupBy()             | Group records                                  | Payment, Report Service
| ToDictionary()        | Convert to dictionary                          | Payment, Report Service
| Distinct()            | Remove duplicates                              | ConnectionRequest, Repo
| Concat()              | Combine sequences                              | Report Service
| Contains()            | Check membership                               | ConnectionRequest Servi

================================================================================
END OF DOCUMENT
================================================================================
```