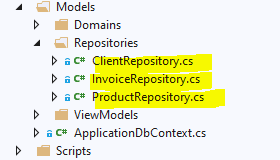
Operacje na Bazie danych



Wszelakich zmian związanych za operacjami na bazie danych musimy przeprowadzić tylko w plikach repozytorium. Dzięki temu, że mamy dobrze rozdzielone warstwy.

Zaczynamy od ClientRepository.

|  |
| --- |
| public class ClientRepository  {  public List<Client> GetClients(string userId)  {  using (var context = new ApplicationDbContext())  {  return context.Clients.Where(x => x.UserId == userId).ToList();  }  }  } |

|  |
| --- |
| public class ProductRepository  {  public List<Product> GetProducts()  {  using (var context = new ApplicationDbContext())  {  return context.Products.ToList();  }  }  public Product GetProduct(int productId)  {  using (var context = new ApplicationDbContext())  {  // może być First(), FirstOrDefault()  // Single jeżeli nie znajdzie lub będzie więcej produktów o danym Id  // To w tym miejscu zostanie rzucony wyjątek  return context.Products.Single(x => x.Id == productId);  }  }  } |

|  |
| --- |
| using System.Data.Entity;    public List<Invoice> GetInvoices(string userId)  {  using (var context = new ApplicationDbContext())  {  return context.Invoices  .Include(x => x.Client )  .Where(x => x.UserId == userId)  .ToList();  }  } |

Include() ponieważ chcemy załączyć informacje o kliencie dla każdej faktury.  
Aby Include zadziałało musimy dodać namespace **System.Data.Entity**

|  |
| --- |
| public Invoice GetInvoice(int id, string userId)  {  using (var context = new ApplicationDbContext())  {  // chcemy załączyć:  // informacje o pozycjach  // informacje o Produktach  return context.Invoices  .Include(x => x.InvoicePositions)  .Include(x => x.InvoicePositions.Select(y => y.Product))  .Include(x => x.MethodOfPayment)  .Include(x => x.User)  .Include(x => x.User.Address)  .Include(x => x.Client)  .Include(x => x.Client.Address)  .Single(x => x.Id == id && x.UserId == userId);  }  } |

Dodanie rekordu

|  |
| --- |
| public void Add(Invoice invoice)  {  using (var context = new ApplicationDbContext())  {  invoice.CreatedDate = DateTime.Now;  context.Invoices.Add(invoice);  context.SaveChanges();  }  } |

Aktualizacja Rekordu

|  |
| --- |
| public void Update(Invoice invoice)  {  using (var context = new ApplicationDbContext())  {  // pobieramy pojedynczy rekord z fakturą do aktualizacji  var invoiceToUpdate = context.Invoices  .Single(x => x.Id == invoice.Id && x.UserId == invoice.UserId);  // dokonujemu zmian  invoiceToUpdate.Client = invoice.Client;  invoiceToUpdate.Comments = invoice.Comments;  invoiceToUpdate.MethodOfPaymentId = invoice.MethodOfPaymentId;  invoiceToUpdate.PaymentDate = invoice.PaymentDate;  invoiceToUpdate.Title = invoice.Title;  // zapisujemy zmiany  context.SaveChanges();  }  } |

Dodanie Pozycji Faktury

|  |
| --- |
| public void AddPosition(InvoicePossition invoicePosition, string userId)  {  // na początku musimy się upewnić, czy użytkownik dodaje pozycje do swojej faktury  // czyli sprawdzamy czy istnieje faktura o id takim jak id dodawnej pozycji  // i czy ta faktura przynależy do użytkownika  using (var context = new ApplicationDbContext())  {  var invoiceToUpdate = context.Invoices  .Single(x =>  x.Id == invoicePosition.InvoiceId &&  x.UserId == userId );    // jeżeli nie ma takiego rekordu to zostanie rzucony wyjątek    context.InvoicePossitions.Add(invoicePosition);  context.SaveChanges();  }  } |

Aktualizacja Pozycji Faktury

|  |
| --- |
| public void UpdatePosition(InvoicePossition invoicePosition, string userId)  {  using (var context = new ApplicationDbContext())  {  // pobieramy pozycje z faktury  var invoicePositionToUpdate = context.InvoicePossitions  .Include(x => x.Product)  .Include(x => x.Invoice)  .Single(x =>  x.Id == invoicePosition.Id &&  x.Invoice.UserId == userId);  invoicePositionToUpdate.Lp = invoicePosition.Lp;  invoicePositionToUpdate.ProductId = invoicePosition.ProductId;  invoicePositionToUpdate.Quantity = invoicePosition.Quantity;  invoicePositionToUpdate.Value =  invoicePosition.Quantity \* invoicePosition.Product.Value;  context.SaveChanges();  }  } |

Aktualizacja Wartości Faktury

|  |
| --- |
| public decimal UpdateInvoiceValue(int invoiceId, string userId)  {  using (var context = new ApplicationDbContext())  {  var invoiceToUpdate = context.Invoices  .Include( x => x.InvoicePositions)  .Single(x =>  x.Id == invoiceId &&  x.UserId == userId);  invoiceToUpdate.Value = invoiceToUpdate.InvoicePositions.Sum(x => x.Value);  context.SaveChanges();  return invoiceToUpdate.Value;  }  } |

Usuwanie Pozycji Faktury

|  |
| --- |
| public void Delete(int invoiceId, string userId)  {  using (var context = new ApplicationDbContext())  {  var invoiceToDelete = context.Invoices  .Single(x =>  x.Id == invoiceId &&  x.UserId == userId);  context.Invoices.Remove(invoiceToDelete);  context.SaveChanges();  }  throw new NotImplementedException();  } |