Implementacja DDD .NET - Zadanie 4

Bottega IT Minds

1. Zadanie 4 - CQRS

1.1. Wprowadzenie

W module *Scoring* została dodana infrastruktura, aby móc:

- Persystować obiekty przy użyciu ORM *EntityFramework*
- Odczytywać dane z bazy przy pomocy biblioteki *Dapper*

Ponadto, zostały zaimplementowane przypadki użycia w warstwie aplikacyjnej oraz napisane odpowiednie testy integracyjne.

Przykład zapisu:

```
public class CreateCustomerScoringCommandHandler :
ICommandHandler<CreateCustomerScoringCommand>
{
    private readonly ICustomerScoringRepository _customerScoringRepository;

    public CreateCustomerScoringCommandHandler(ICustomerScoringRepository
customerScoringRepository)
    {
        _customerScoringRepository = customerScoringRepository;
    }

    public async Task<Unit> Handle(CreateCustomerScoringCommand command,
CancellationToken cancellationToken)
    {
        var customerScoring = CustomerScoring.Create(command.CustomerId);

        await _customerScoringRepository.Add(customerScoring);

        return Unit.Value;
    }
}
```

Przykład odczytu:

```
public class GetCustomerScoringQueryHandler : IQueryHandler<GetCustomerScoringQuery,
CustomerScoringDto>
{
    private readonly IDbConnectionFactory _connectionFactory;

    public GetCustomerScoringQueryHandler(IDbConnectionFactory connectionFactory)
    {
        _connectionFactory = connectionFactory;
    }
```

```
    public async Task<CustomerScoringDto> Handle(GetCustomerScoringQuery query,
CancellationToken cancellationToken)
    {
        var connection = _connectionFactory.GetOpenConnection();

        const string sql = $@"SELECT
                  customer_scoring.customer_id
{nameof(CustomerScoringDto.CustomerId)},
                  customer_scoring.loan_limit_value
{nameof(CustomerScoringDto.LoanLimitValue)},
                  customer_scoring.loan_limit_currency
{nameof(CustomerScoringDto.LoanLimitCurrencyCode)}
                  FROM scoring.customer_scorings customer_scoring
                  WHERE customer_scoring.customer_id = :customerId";

        return await connection.QuerySingleAsync<CustomerScoringDto>(sql, new
        {
            query.CustomerId
        });
    }
}
```

Przykład testu integracyjnego:

```
    [Test]
    public async Task
GivenLoanLimitConfigurationSetToPercentage_AndTwoCustomerOrders_WhenChangeScoring_Then
LoanLimitIsSet()
    {
        // Given
        var customerId = Guid.NewGuid();

        await ScoringModule.ExecuteCommand(new CreateLoanLimitConfigurationCommand(
            0.1m,
            CurrentLoanLimitConfiguration.Percentage.Code));

        await ScoringModule.ExecuteCommand(new
CreateCustomerScoringDocumentCommand(customerId));

        await ScoringModule.ExecuteCommand(new RegisterOrderCommand(
            Guid.NewGuid(),
            customerId,
            100,
            "PLN"));

        await ScoringModule.ExecuteCommand(new RegisterOrderCommand(
            Guid.NewGuid(),
            customerId,
            400,
```

```
            "PLN"));

        // When
        await ScoringModule.ExecuteCommand(new
ChangeCustomerScoringDocumentCommand(customerId));

        // Then
        var scoring = await ScoringModule.ExecuteQuery(new
GetCustomerScoringDocumentQuery(customerId));
        scoring.CustomerId.Should().Be(customerId);
        scoring.LoanLimitValue.Should().Be(50);
        scoring.LoanLimitCurrencyCode.Should().Be("PLN");
    }
```

1.2. Treść Zadania

Na bazie implementacji warstwy infrastruktury i aplikacji w module *Scoring* zaimplementuj 3 przypadki użycia w module *Availability*:

1. Dodanie Zasobu
2. Pobranie Zasobu
3. Zablokowanie Zasobu Tymczasowe

Zaimplementuj testy integracyjne dla przypadków użycia *Dodanie Zasobu* i *Zablokowanie Zasobu Tymczasowe*.

Informacje pomocnicze:

1. Dla zapisu należy dodać mapę EntityFramework, implementację repozytorium oraz odpowiedniego Command Handlera
2. Dla odczytu należy użyć biblioteki Dapper pisząc odpowiedniego SQL
3. Wszystkie kwestie związane z konfiguracją połączenia do bazy danych, unit of work itp. są już zaimplementowane
4. Obiekty bazodanowe znajdują się w projekcie `Bottega.PhotoStock.Availability.Database` (schemat i tabela)
5. Podczas testów należy użyć do symulowania czasu klasy `SystemClock`