**Тема: «Работа с базами данных. Технологии доступа к данным»**

**Практическая часть:**

Для каждого варианта необходимо выполнить три пункта на уровне сложности 3:

1. **Реализация CRUD-операций**
2. **Привязка данных из базы к интерфейсу**
3. **Сохранение данных**

В конце документа представлены некоторые шаблоны. Их можно использовать.

**1. АРМ работника склада**

• **CRUD-операции:** создать ProductRepository с методами GetAllAsync(), AddAsync(Product), UpdateAsync(Product), DeleteAsync(Product)  
• **Привязка:** во WarehouseViewModel свойство public ObservableCollection<Product> Products { get; } и в XAML <DataGrid ItemsSource="{Binding Products}" />  
• **Сохранение:** в командах AddProductCommand и DeleteProductCommand после вызова репозитория вызывать await \_context.SaveChangesAsync()

**2. CRM для небольшого бизнеса**

• **CRUD-операции:** ClientRepository с GetAllClientsAsync(), AddClientAsync(Client), UpdateClientAsync(Client), DeleteClientAsync(Client)  
• **Привязка:** Clients в CRMViewModel как ObservableCollection<Client>, в XAML <ListView ItemsSource="{Binding Clients}" />  
• **Сохранение:** в SaveClientsCommand выполнять await \_context.SaveChangesAsync()

**3. Электронный дневник студента**

• **CRUD-операции:** GradeRepository с GetGradesAsync(), AddGradeAsync(Grade), UpdateGradeAsync(Grade), DeleteGradeAsync(Grade)  
• **Привязка:** Grades в DiaryViewModel, XAML <DataGrid ItemsSource="{Binding Grades}" />  
• **Сохранение:** команды AddGradeCommand и DeleteGradeCommand вызывают await \_context.SaveChangesAsync()

**4. Управление задачами (Task Manager)**

• **CRUD-операции:** TaskRepository с GetTasksAsync(), AddTaskAsync(TaskItem), UpdateTaskAsync(TaskItem), DeleteTaskAsync(TaskItem)  
• **Привязка:** Tasks в TaskViewModel, XAML <ListBox ItemsSource="{Binding Tasks}" />  
• **Сохранение:** в CompleteTaskCommand после изменения статуса – await \_context.SaveChangesAsync()

**5. Система бронирования отеля**

• **CRUD-операции:** RoomRepository и BookingRepository с методами GetAllAsync(), AddAsync(), UpdateAsync(), DeleteAsync()  
• **Привязка:** Rooms к <ListView ItemsSource="{Binding Rooms}" />, Bookings к <DataGrid ItemsSource="{Binding Bookings}" />  
• **Сохранение:** в BookRoomCommand и CancelBookingCommand вызывать await \_context.SaveChangesAsync()

**6. Управление заказами в интернет-магазине**

• **CRUD-операции:** OrderRepository и OrderItemRepository с методами для получения, добавления, обновления, удаления  
• **Привязка:** Orders в ShopViewModel к <DataGrid ItemsSource="{Binding Orders}" />  
• **Сохранение:** команды AddOrderCommand, DeleteOrderCommand выполняют await \_context.SaveChangesAsync()

**7. Автоматизация учёта сотрудников**

• **CRUD-операции:** EmployeeRepository с GetEmployeesAsync(), AddEmployeeAsync(Employee), UpdateEmployeeAsync(Employee), DeleteEmployeeAsync(Employee)  
• **Привязка:** Employees к <ListView ItemsSource="{Binding Employees}" />  
• **Сохранение:** в FireEmployeeCommand вызывать await \_context.SaveChangesAsync()

**8. Приложение для учёта финансов**

• **CRUD-операции:** TransactionRepository с методами GetTransactionsAsync(), AddTransactionAsync(Transaction), UpdateTransactionAsync(Transaction), DeleteTransactionAsync(Transaction)  
• **Привязка:** Transactions к <DataGrid ItemsSource="{Binding Transactions}" />  
• **Сохранение:** в AddTransactionCommand и DeleteTransactionCommand вызывать await \_context.SaveChangesAsync()

**9. Бронирование билетов в кинотеатр**

• **CRUD-операции:** SessionRepository и TicketRepository для сеансов и билетов  
• **Привязка:** Sessions к <ComboBox ItemsSource="{Binding Sessions}" />, Tickets к <ListBox ItemsSource="{Binding Tickets}" />  
• **Сохранение:** в BookTicketCommand и CancelTicketCommand выполнять await \_context.SaveChangesAsync()

**10. Система контроля версий документов**

• **CRUD-операции:** DocumentRepository и VersionRepository с методами GetAllAsync(), AddAsync(), UpdateAsync(), DeleteAsync()  
• **Привязка:** Versions к <TreeView ItemsSource="{Binding Versions}" />  
• **Сохранение:** команды AddVersionCommand и DeleteVersionCommand вызывают await \_context.SaveChangesAsync()

**11. Программа для учёта медицинских записей**

• **CRUD-операции:** PatientRepository и RecordRepository с методами получения, добавления, изменения, удаления  
• **Привязка:** Patients к <ListView ItemsSource="{Binding Patients}" />, Records к <DataGrid ItemsSource="{Binding Records}" />  
• **Сохранение:** в RegisterPatientCommand и AddRecordCommand вызывать await \_context.SaveChangesAsync()

**12. Управление учебным процессом (электронный журнал)**

• **CRUD-операции:** CourseRepository, StudentRepository, EnrollmentRepository  
• **Привязка:** Enrollments к <DataGrid ItemsSource="{Binding Enrollments}" />  
• **Сохранение:** команды EnrollStudentCommand, AddGradeCommand выполняют await \_context.SaveChangesAsync()

**13. Приложение для аренды автомобилей**

• **CRUD-операции:** CarRepository и RentalRepository  
• **Привязка:** Cars к <ListView ItemsSource="{Binding Cars}" />  
• **Сохранение:** в RentCarCommand и ReturnCarCommand – await \_context.SaveChangesAsync()

**14. Приложение для организации мероприятий**

• **CRUD-операции:** EventRepository и ParticipantRepository  
• **Привязка:** Events к <ListView ItemsSource="{Binding Events}" />, Participants к <ListBox ItemsSource="{Binding Participants}" />  
• **Сохранение:** команды CreateEventCommand, RegisterParticipantCommand выполняют await \_context.SaveChangesAsync()

**15. Электронная библиотека**

• **CRUD-операции:** BookRepository и LoanRepository  
• **Привязка:** Books к <ListBox ItemsSource="{Binding Books}" />  
• **Сохранение:** команды BorrowBookCommand, ReturnBookCommand выполняют await \_context.SaveChangesAsync()

**Шаблон ViewModel для всех проектов:**

public class BaseViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged([CallerMemberName]string name=null)

=> PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

}

public class ProjectViewModel<T> : BaseViewModel

{

private readonly IRepository<T> \_repo;

public ObservableCollection<T> Items { get; } = new();

public ICommand LoadCommand { get; }

public ICommand AddCommand { get; }

public ICommand DeleteCommand { get; }

public ProjectViewModel(IRepository<T> repo)

{

\_repo = repo;

LoadCommand = new AsyncRelayCommand(async () => {

var list = await \_repo.GetAllAsync();

Items.Clear();

foreach(var i in list) Items.Add(i);

});

AddCommand = new AsyncRelayCommand(async () => {

// реализация добавления

await \_repo.AddAsync(new T());

await \_repo.SaveAsync();

await ((AsyncRelayCommand)LoadCommand).ExecuteAsync(null);

});

DeleteCommand = new AsyncRelayCommand(async () => {

// реализация удаления

await \_repo.DeleteAsync(SelectedItem);

await \_repo.SaveAsync();

await ((AsyncRelayCommand)LoadCommand).ExecuteAsync(null);

});

}

public T SelectedItem { get; set; }

}

**View XAML:**

xml

КопироватьРедактировать

<DataGrid ItemsSource="{Binding Items}" SelectedItem="{Binding SelectedItem}" AutoGenerateColumns="True"/>

<Button Content="Load" Command="{Binding LoadCommand}"/>

<Button Content="Add" Command="{Binding AddCommand}"/>

<Button Content="Delete" Command="{Binding DeleteCommand}"/>