**Вопросы к лабораторной работе №10**

1. Для чего используют многоуровневые архитектуры?

Многоуровневые архитектуры используются для создания приложений, которые могут быть более гибкими, расширяемыми и легко поддерживаемыми. Они позволяют разделить приложение на несколько уровней или слоев, каждый из которых выполняет определенную функцию.

2. Опишите назначение слоев многоуровневой архитектуре. Business layer (уровень бизнес-логики), Data Access layer (уровень доступа к данным).

**Business layer** (уровень бизнес-логики): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

**Data Access layer** (уровень доступа к данным): хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

3. Поясните назначение и принцип использования паттерна Repository?

Одним из наиболее часто используемых паттернов при работе с данными является паттерн 'Репозиторий'. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

4. Поясните назначение и принцип использования паттерна Unit of Work?

Паттерн Unit of Work позволяет упростить работу с различными репозиториями и дает уверенность, что все репозитории будут использовать один и тот же контекст данных.

5. Опишите основные особенности библиотеки Entity Framework?

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является **Entity Data Model**. Эта модель сопоставляет классы сущностей с реальными таблицами в БД.

Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга).

На концептуальном уровне происходит определение классов сущностей, используемых в приложении.

Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных.

Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

6. Какие преимущества обеспечивает использование слабосвязанного кода?

Слабосвязанный код обеспечивает ряд преимуществ, включая:

Гибкость: Слабосвязанный код легко модифицировать и настраивать. Если два компонента имеют слабую связь, то изменение одного из них не приведет к нарушению работы другого компонента.

Повторное использование: Компоненты, которые связаны слабо, могут быть легко использованы повторно в других приложениях или в других частях того же приложения.

Упрощение тестирования: Компоненты, связанные слабо, могут быть легко тестированы независимо друг от друга, что облегчает процесс тестирования и ускоряет выявление ошибок.

Легкость поддержки: Код, который связан слабо, проще поддерживать и изменять, потому что любые изменения могут быть внесены локально без влияния на остальную часть приложения.

Улучшенная масштабируемость: Слабая связь позволяет легко масштабировать приложение, так как компоненты могут быть легко добавлены или удалены без нарушения работы остальной части приложения.

Повышение производительности: Слабосвязанный код обеспечивает более быструю загрузку и выполнение, поскольку он не требует большого количества связей между компонентами и не затрудняет выполнение операций.