|  |
| --- |
| 1. Для чего используют многоуровневые архитектуры? |
|  |  |
|  | Многоуровневые архитектуры в WPF используются для организации приложения в виде различных слоев или уровней, каждый из которых отвечает за определенную функциональность. Обычно такие архитектуры состоят из трех основных уровней:  Представление (View) - уровень, который отвечает за отображение данных и взаимодействие с пользователем. В WPF это может быть реализовано с помощью XAML-файлов и кода бэкэнда (code-behind), который управляет поведением пользовательского интерфейса.  Бизнес-логика (Business Logic) - уровень, который содержит логику приложения и управляет обработкой данных. Этот уровень может содержать классы, которые отвечают за получение и обработку данных, а также бизнес-правила, которые определяют, как эти данные должны быть обработаны.  Доступ к данным (Data Access) - уровень, который отвечает за получение данных из источника данных и их сохранение. В WPF это может быть реализовано с помощью классов, которые работают с базами данных или другими источниками данных.  Использование многоуровневых архитектур позволяет лучше структурировать приложение и облегчает его разработку и тестирование. Кроме того, это позволяет лучше разделить ответственность между различными частями приложения, что упрощает сопровождение и расширение приложения в будущем. |
|  |  |
|  | 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? |
|  |  |
|  | «Репозиторий» — один из наиболее часто используемых паттернов проектирования при работе с базами данных (БД). |
|  | Он позволяет отделить программную логику, работающую непосредственно с БД, |
|  | от всей остальной программы выступая посредником меду ними с помощью интерфейса во многом схожего с коллекциями. |
|  |  |
|  | В чем преимущество использования паттерна «Репозиторий»? |
|  | Отделение программной логики, работающей непосредственно с БД от всей остальной программы. |
|  | Если требуется изменить логику работы с БД или даже тип БД (например, перенос БД с MS SQL Server на MySQL), |
|  | то внесение изменений осуществляется локально и централизованно. |
|  | Нет необходимости вносить многочисленные правки по всей программе с вероятностью что-либо пропустить и тем самым спровоцировать ошибку в работе программы. |
|  | С помощью «Репозитория» можно значительно упростить и алгоритмы по работе с БД в остальной программе. |
|  | Чаще всего для выполнения какой-либо операции с БД достаточно вызвать один из методов класса, реализующего этот паттерн. |
|  |  |
|  |  |
|  | 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? |
|  |  |
|  | Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. |
|  | Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, |
|  | то Entity Framework представляет собой более высокий уровень абстракции, |
|  | который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. |
|  | Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, |
|  | который нам предлагает Entity Framework, мы уже работает с объектами. |
|  |  |
|  | 6.Какие преимущества обеспечивает использование слабосвязанного |
|  | кода? |
|  |  |
|  | Самое большое преимущество заключается в том, что введение изменений в один модуль не нарушает другие модули непредсказуемым образом. |
|  | Другим преимуществом является возможность замены компонентов более легко, когда у вас более одной реализации. |