**Вопросы к лабораторной работе №11**

1. Что такое MVC, MVP, MVVM?

**1) MVC** (Model View Controller – Модель Представление Контроллер) – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер таким образом, что модификация каждого компонента может осуществляться независимо.

**Модель** предоставляет данные и реагирует на команды контроллера, изменяя свое состояние

**Представление** отвечает за отображение данных модель юзеру, реагируя на изменения модели

**Контроллер** интерпретирует действия юзера, оповещая модель о необходимости изменения

Основная цель – отделение бизнес-логики (модели) от её визуализации (представления, вида)

**2) MVP** (Minimum Viable Product ) – минимально жизнеспособный продукт, который позволяет получить осмысленную обратную связь от юзеров, понять что им нужно и не создавать то, что им неинтересно и за что они не готовы платить.

\*сэкономить деньги, не вкладывая в провальный проект

\*проверить, интересует ли ваш продукт потенциальных пользователей

\*с помощью итераций узнать, какое направление развития будет самым оптимальным

\*собрать базу потенциальных клиентов и найти ранних приверженцев своего продукта

**3) MVVM** (Model View ViewModel) – шаблон проектирования архитектуры приложения.

Используется для разделения модели и её представления, что необходимо для их изменения отдельно друг от друга. Например, разработчик задаёт логику работы с данными, а дизайнер работает с пользовательским интерфейсом.

2. Объясните из каких компонентов состоит паттерн MVVM?

\***Модель** представляет собой логику работы с данными и описание фундаментальных данных, необхоимых для работы приложения

**\*Представление** – графический интерфейс (окна, списки, кнопки и т.д.). Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью Представления (если в Модели Представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновленное значение свойства из Модели Представления)

**\*Модель представления** – с одной стороны, абстракция Представления, а с другой – обёртка данных из Модели, подлежащие связыванию. То есть, она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель

3. Каким образом паттерн можно реализовать в WPF?

\*Построение модели – EF

\*Создание репозитория

\*Реализация репозитория

4. Объясните назначение интерфейса INotifyPropertyChanged?

Для использования механизма привязки.

Этот интерфейс реализует систему уведомлений которая активируется, когда значение свойств изменяется. Это требуется в модели-представления, чтобы сделать механизм привязки пользовательского интерфейса XAML динамическмс.

5. Зачем нужен интерфейс ICommand и как паттерн MVVM использует команды в WPF-приложениях.

Команды могут привязываться к определенному XAML-элементу и определять поведение данного элемента при определенных действиях.

Для ряда визуальных элементов WPF (кнопок) определена поддержка команд. Сами команды определяются в ViewModel и затем механизм привязки устанавливаются для элементов управления.

6. Чем отличаются команды от событий?   
события могут туннелировать и всплывать, а комманды только всплывать

Команды – привязанные объекты, что позволяет отделить логику от польз.интерф. – исп.для взаимодействия юзера и приложения

В мввм команды использ.вместо событий

Проблема wpf – не все события могут вызвать команду. Обычно с командой связано только одно осн.событие некот.контролов (клик на кнопке)

7. Расскажите о назначении и об основных возможностях следующих библиотек и фреймворков: Light Toolkit, Catel и Prism.

**Light Toolkit** добавляет в VS новый тип проекта – таким образом не нужно создавать стр-ру проекта и все сделано за нас

Это кросс-платформенная библиотека с открытым исходным кодом для построения граф.интерф.юзера.

**PRISM** позволяет проектировать и писать приложения со слабо связанными компонентами, кот.могут независимо развиваться и потом объединяться в одно целое без особых усилий

**CATEL –** платформа для разработки приложений. Цель – предоставить полный набор функций для приложений, написанных на любой .NET технологии.