**Для разработки графического интерфейса приложения использована платформа WPF.**

Платформа Windows Presentation Foundation (WPF) позволяет создавать клиентские приложения для настольных систем Windows с привлекательным пользовательским интерфейсом.

В основе WPF лежит независимый от разрешения векторный модуль визуализации, использующий возможности современного графического оборудования. Возможности этого модуля расширяются с помощью комплексного набора функций разработки приложений, которые включают в себя язык XAML, элементы управления, привязку к данным, макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF является частью .NET, поэтому вы можете создавать приложения, включающие другие элементы .NET API.

WPF позволяет разрабатывать приложения, используя как разметку, так и код программной части, который пишется на языке программирования C#. Разметка XAML обычно используется для определения внешнего вида приложения, а управляемые языки программирования (код программной части) — для реализации его поведения. Такое разделение внешнего вида и поведения имеет ряд преимуществ.

Затраты на разработку и обслуживание снижаются, так как разметка, определяющая внешний вид, не связана тесно с кодом, обуславливающим поведение.

Также повышается эффективность разработки, так как дизайнеры, занимающиеся внешним видом приложения, могут работать параллельно с разработчиками, реализующими поведение приложения.

**Данные приложения будут храниться в базе данных. Для хранения данных использована СУБД SQLite.**

SQLite — это встраиваемая СУБД, когда система управления встраивается в саму программу. Это значит, что все запросы и команды идут в базу не через посредника, а напрямую из приложения.

А ещё SQLite — это проект с открытым исходным кодом. Это значит, что его может использовать кто угодно без опасения, что проект закроют и все базы сразу перестанут работать.

Все данные в SQLite хранятся в одном файле — таблицы, служебные поля, связи и всё остальное. Это упрощает работу с базой и позволяет легко переносить данные из одного места в другое.

Особенность разработки SQLite также в том, что тестами покрыто 100% исходного кода. Это значит, что в нём протестированы каждая функция, обработчик и класс, причём на всех уровнях — от юнита до всей системы. Словарь тестировщика: автотесты, юнит-тесты и другие важные слова

Тестов в разработке SQLite настолько много, что объём кода для тестов давно превысил объём самого SQLite. А всё для того, чтобы база данных работала даже в самых сложных условиях.

**Обеспечение связи базы данных SQLite с приложением WPF использована технология 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.

Центральной концепцией Entity Framework является понятие сущности или Entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

**Одноуровневая архитектура – архитектурный подход, который предполагает размещение всех трех слоев кода на одном сервере.**

Приложение реализовано на этом типе архитектуры. При этом обычно имеются следующие слои:

1. Слой представления. Он содержит пользовательский интерфейс и отвечает за обеспечение хорошего пользовательского опыта.
2. Слой бизнес-логики. Данный слой, как следует из названия, содержит бизнес-логику приложения. Он отделяет UI/UX от вычислений, связанных с бизнесом. Это позволяет с легкостью изменять логику в зависимости от постоянно меняющихся бизнес-требований, никак не влияя на другие слои.
3. Слой передачи данных. Этот слой отвечает за взаимодействие с постоянными хранилищами, такими как базы данных, и прочую обработку информации, которая не связана с бизнесом.

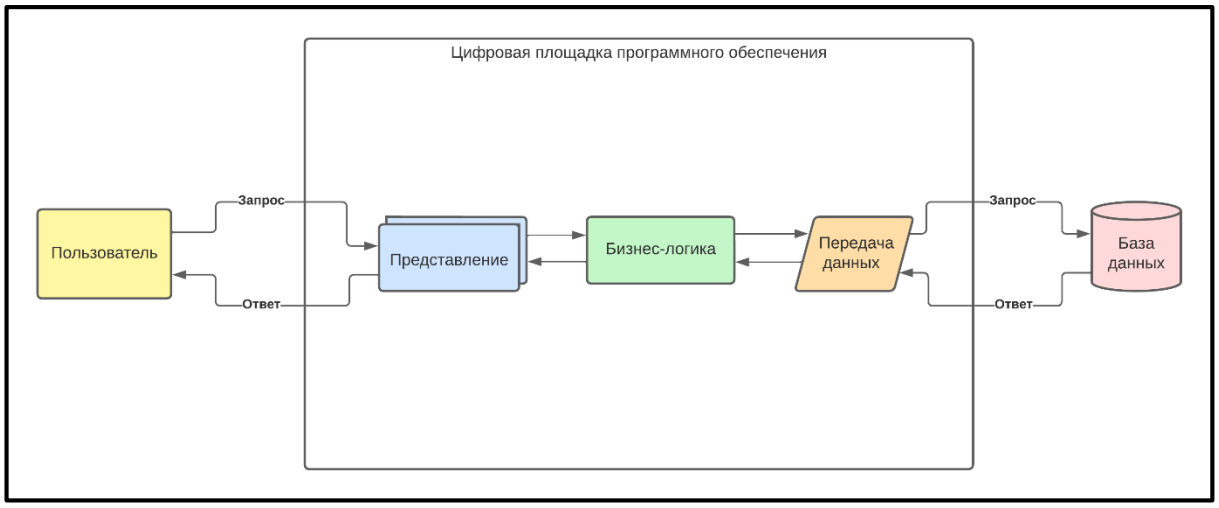


Рисунок 1 – Диаграмма архитектуры приложения