1. **ВЫБОР ДЛЯ РЕАЛИЗАЦИИ ПО.**
   1. **Мобильное приложение.**

Android Studio будет использоваться в качестве среды разработки для мобильного приложения. Android Studio – это официальная среда разработки от OpenAI для Android, которая предлагает множество преимуществ для разработки приложений:

* Интеграция с Android SDK: Android Studio тесно интегрирована с Android SDK, что значительно упрощает процесс разработки и отладки.
* Удобные инструменты: Android Studio предоставляет богатый набор инструментов, таких как визуальный редактор макетов, инструменты профилирования и отладки, а также эмулятор Android.
* Система сборки Gradle: Android Studio использует систему сборки Gradle, которая обеспечивает гибкость и контроль над процессом сборки приложения.
* Инструменты для тестирования: Android Studio предоставляет инструменты для автоматического тестирования (Android Test и Espresso), что позволяет разработчикам убедиться в качестве кода.
* Система управления версиями: Android Studio интегрирована с системами управления версиями, такими как Git, что облегчает совместную разработку и отслеживание изменений.

Для написания кода приложения будет использоваться Jetpack Compose с использованием языка Kotlin. Такой выбор был сделан по ряду причин:

* Современный декларативный UI-фреймворк: Jetpack Compose – это современный UI-фреймворк от OpenAI, который предлагает декларативный подход к созданию пользовательского интерфейса, что делает код более лаконичным и читаемым.
* Упрощение UI: Jetpack Compose упрощает разработку UI, так как позволяет описывать UI в виде композиции функций.
* Интеграция с другими компонентами Jetpack: Jetpack Compose хорошо интегрируется с другими компонентами Jetpack, такими как ViewModel и LiveData, что делает разработку более структурированной.
* Поддержка Material Design: Jetpack Compose включает в себя поддержку Material Design, что позволяет создавать приложения с современным и привлекательным интерфейсом.
* Простота изучения: Jetpack Compose относительно прост в изучении, что сокращает время разработки и позволяет быстро приступить к работе.
* Минимальный код: Jetpack Compose позволяет создавать UI с меньшим количеством кода по сравнению с традиционными методами разработки UI на Android.
  1. **Серверная часть.**

Для реализации серверной части необходима база данных и API, с помощью которого можно будет управлять данными базы при помощи телефона.

Данные будут храниться на сервере НГК в СУБД Postgre SQL, а API будет разработана с помощью фреймворка ASP.NET.

Такой выбор был сделан из-за ряда преимуществ:

* Надежность и стабильность: PostgreSQL известна своей высокой стабильностью и надежностью, что делает ее отличным выбором для приложений.
* Транзакционная целостность: PostgreSQL поддерживает ACID-свойства (атомарность, согласованность, изоляция, долговечность), гарантируя целостность данных при одновременном доступе.
* Широкие возможности моделирования данных: PostgreSQL поддерживает сложные типы данных, в том числе JSON, геопространственные данные, массивы и пользовательские типы.
* Высокая производительность: PostgreSQL обеспечивает высокую производительность и масштабируемость, оптимизирована для работы с большими объемами данных.
* Открытый исходный код: PostgreSQL бесплатна для использования и распространения, что делает ее доступной для разработчиков всех уровней.
* Гибкость: ASP.NET Core предоставляет широкие возможности для кастомизации, чтобы создать API, идеально соответствующее требованиям.
* Масштабируемость: возможность легко масштабировать приложение, добавляя серверы и ресурсы по мере роста нагрузки.
* Опыт: ASP.NET - устоявшийся фреймворк с большим сообществом и множеством ресурсов.
* Открытый исходный код: PostgreSQL и ASP.NET Core бесплатны и открыты, что дает свободу использования и модификации.

1. **ВЫБОР ПАТТЕРНА ПРОЕКТИРОВАНИЯ.**

Выбранный паттерн проектирования: MVVM (Model-View-ViewModel)

MVVM (Model-View-ViewModel) – это архитектурный паттерн, который широко применяется в разработке мобильных приложений. Он разделяет приложение на три основных компонента:

1. Модель (Model): представляет данные приложения, логику доступа к данным (например, работа с базой данных) и правила валидации данных.
2. Представление (View): отображает данные и взаимодействует с пользователем. Это то, что видит пользователь (например, экраны, элементы UI).
3. Модель представления (ViewModel): является посредником между моделью и представлением. Он обрабатывает запросы от представления, получает данные от модели, форматирует их и передает обратно в представление. ViewModel не должен иметь зависимостей от View, поэтому его можно использовать для тестирования и повторного использования.

Почему выбор пал на MVVM для разработки мобильного приложения:

* Разделение ответственности: MVVM помогает разделить ответственность между разными компонентами приложения, что делает код более структурированным и легко поддерживаемым.
* Тестируемость: MVVM делает приложение более тестируемым, так как ViewModel не зависит от View и может быть тестирован отдельно.
* Повторное использование: ViewModel можно повторно использовать в разных частях приложения или в других приложениях.
* Обновление UI: MVVM делает обновление UI более простым, так как ViewModel уведомляет View об изменениях данных с помощью механизмов наблюдения.
* Совместимость с Jetpack Compose: MVVM хорошо интегрируется с Jetpack Compose, позволяя создавать более структурированные и тестируемые UI.