2.1 Выбор языка программирования и базы данных

**Python** - это универсальный язык программирования общего назначения, который широко используется в различных областях, таких как веб-разработка, машинное обучение, научные вычисления, автоматизация и многое другое.

**Плюсы Python:**

* **Простота изучения и использования:** Python имеет простой и читаемый синтаксис, схожий с естественным языком, что делает его доступным для начинающих программистов.
* **Большая библиотека и фреймворки:** Python обладает обширной экосистемой библиотек и фреймворков для различных задач, таких как веб-разработка (Django, Flask), машинное обучение (Scikit-learn, TensorFlow), анализ данных (NumPy, Pandas) и многое другое.
* **Кроссплатформенность:** Python работает на различных операционных системах (Windows, macOS, Linux), что делает его удобным для разработки на разных устройствах.
* **Продуктивность:** Python позволяет писать лаконичный и читаемый код, что делает разработку более быстрой и эффективной.
* **Поддержка сообщества:** Python имеет большое и активное сообщество разработчиков, что обеспечивает доступ к обширным ресурсам, документации и поддержке.

**Минусы Python:**

* **Производительность:** Python может быть медленнее, чем некоторые другие компилируемые языки, такие как C++ или Java, особенно для ресурсоемких задач.
* **Интерпретируемый язык:** Python интерпретируется во время выполнения, что может привести к незначительному снижению производительности по сравнению с компилируемыми языками.
* **Гибкость:** Из-за своей динамической типизации Python может быть более подвержен ошибкам во время выполнения, чем статически типизированные языки

2.1.1 Почему Python удобен для работы с SQLite:

* **Простота интеграции:** Python и SQLite имеют простые интерфейсы API, что делает их удобными для взаимодействия друг с другом.
* **Объектно-ориентированность:** Python поддерживает объектно-ориентированное программирование, что хорошо подходит для работы с базами данных, таких как SQLite.
* **Библиотеки и фреймворки:** Python имеет множество библиотек и фреймворков, таких как SQLAlchemy, которые упрощают работу с SQLite.
* **Производительность:** SQLite достаточно быстрая для большинства задач, и Python не является узким местом при работе с ней.

**Выбор Python для работы с SQLite:**

Если вы ищете простой, универсальный и удобный язык программирования для работы с базами данных SQLite, Python - отличный выбор.

Его простой синтаксис, обширная библиотека и фреймворки, а также кроссплатформенность делают его идеальным для начинающих и опытных разработчиков.

SQLite, как легкая и высокопроизводительная база данных, прекрасно дополняет Python, позволяя создавать эффективные и надежные приложения.

2.1.2 Выбор среды разработки программы   
 **Visual Studio Code (VS Code)** - это бесплатный редактор кода с открытым исходным кодом, разработанный Microsoft. Он быстро стал одной из самых популярных сред разработки (IDE) благодаря своему широкому набору функций, расширяемости и кроссплатформенности.

**Плюсы VS Code:**

* **Бесплатный и с открытым исходным кодом:** VS Code доступен бесплатно для всех, а его исходный код открыт для сообщества разработчиков.
* **Расширяемость:** VS Code имеет обширный магазин расширений, где можно найти тысячи бесплатных и платных расширений для добавления различных функций, таких как отладка, подсветка синтаксиса, автодополнение кода, интеграция с Git и многое другое.
* **Кроссплатформенность:** VS Code работает на Windows, macOS и Linux, что делает его доступным для разработчиков на любой платформе.
* **Легкий и быстрый:** VS Code потребляет меньше ресурсов компьютера, чем другие IDE, что делает его идеальным для работы на слабых устройствах.
* **Интегрированная поддержка Git:** VS Code имеет встроенную поддержку Git, что позволяет легко управлять версиями кода.
* **Подсветка синтаксиса и автодополнение кода:** VS Code поддерживает подсветку синтаксиса для множества языков программирования и предлагает функции автодополнения кода, которые помогают писать код быстрее и с меньшим количеством ошибок.
* **Отладка:** VS Code позволяет отлаживать код непосредственно в редакторе, что упрощает поиск и исправление ошибок.
* **Настраиваемый интерфейс:** VS Code позволяет настраивать интерфейс под свои нужды, меняя темы оформления, раскладку окон и другие параметры.

**Минусы VS Code:**

* **Ограниченные функции для некоторых языков:** VS Code не имеет таких же функций IDE, как Visual Studio для C# или C++, для некоторых языков программирования.
* **Может быть перегружен расширениями:** Установка слишком большого количества расширений может замедлить работу VS Code и привести к конфликтам.
* **Не подходит для крупных проектов:** VS Code не так хорошо подходит для управления крупными проектами, как некоторые другие IDE, такие как IntelliJ IDEA или Eclipse.

**2.**1.3 Выбор базы данных

**Плюсы:**

* **Простота:** SQLite имеет простой синтаксис SQL, схожий с естественным языком, что делает его доступным для начинающих пользователей.
* **Легкий вес:** SQLite - это легкая библиотека, что делает ее идеальной для использования в мобильных приложениях и на устройствах с ограниченными ресурсами.
* **Самодостаточность:** SQLite не требует сервера или отдельного процесса, что упрощает его развертывание и использование.
* **Производительность:** SQLite достаточно быстрая для большинства задач, особенно для простых запросов и операций с данными.
* **Встраиваемость:** SQLite можно легко встроить в другие приложения, что делает его удобным для создания автономных баз данных.
* **Бесплатность и открытость:** SQLite - это бесплатный и открытый проект, что позволяет использовать его без ограничений и вносить изменения в код.
* **Широкая поддержка:** SQLite поддерживается на большинстве операционных систем, включая Windows, macOS, Linux, iOS, Android и многие другие.
* **Надежность:** SQLite - это надежная база данных, которая обеспечивает целостность данных и устойчивость к сбоям.
* **Расширяемость:** SQLite поддерживает расширения, позволяющие добавлять новые функции и возможности.

**Минусы:**

* **Ограниченная масштабируемость:** SQLite не подходит для хранения и обработки очень больших объемов данных.
* **Отсутствие некоторых функций:** SQLite не имеет некоторых функций, которые есть в других СУБД, таких как хранимые процедуры, триггеры и многопользовательский режим.
* **Ограниченная безопасность:** SQLite по умолчанию не обеспечивает шифрование данных и аутентификацию пользователей.
* **Не подходит для сложных запросов:** SQLite может быть менее эффективным при выполнении сложных запросов, особенно с соединениями таблиц.
* **Не подходит для распределенных систем:** SQLite не предназначен для использования в распределенных системах, где данные должны быть доступны из разных мест.

В целом, SQLite - это отличный выбор для создания небольших, автономных баз данных, где простота, легковесность и производительность являются приоритетными.

2.2 Разработка Програмного обеспечения   
2.2.1 UML-диаграмма: Обращение клиента на сайт и взаимодействие с базой данных

**Цель:**

Представить UML-диаграмму, иллюстрирующую взаимодействие между клиентом, веб-сайтом и базой данных в рамках вашего скрипта для стоматологии. Скрипт будет проверять наличие заболеваний у клиента.

**UML-диаграмма:**

**Участники:**

* **Клиент:** Пользователь, который посещает веб-сайт и взаимодействует со скриптом.
* **Веб-сайт:** Стоматологический веб-сайт, на котором размещен скрипт.
* **База данных:** Хранилище информации о пациентах, включая их заболевания.

**Шаги:**

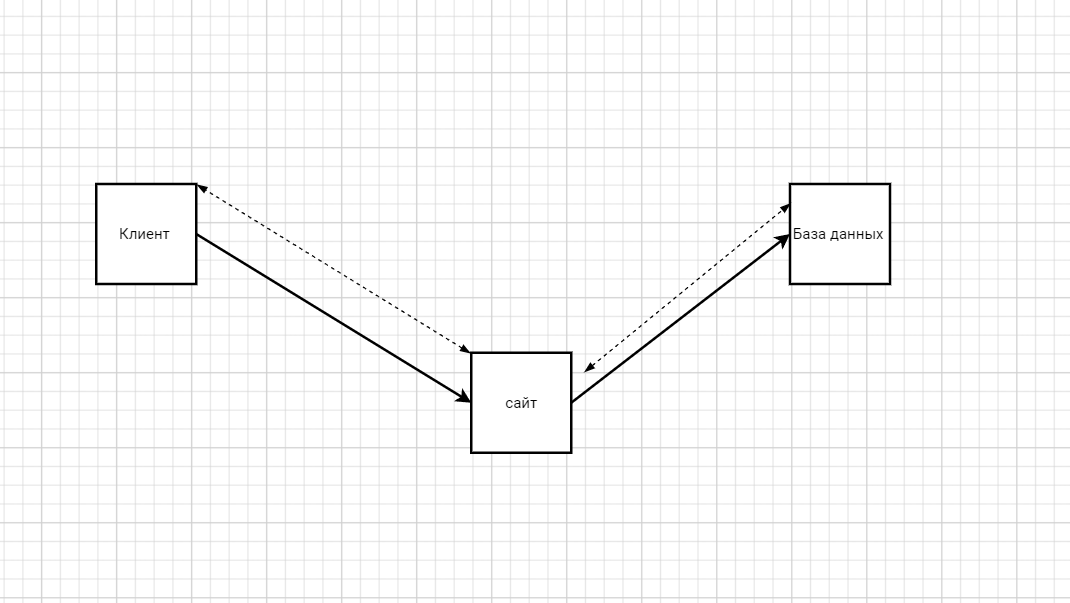
1. **Клиент отправляет запрос:**
   * Клиент вводит свои данные (например, ФИО, дату рождения) на веб-сайте.
   * Веб-сайт получает запрос клиента.
2. **Веб-сайт обращается к базе данных:**
   * Веб-сайт отправляет запрос в базу данных, используя введенные клиентом данные.
   * База данных выполняет поиск информации о пациенте.
3. **База данных возвращает результат:**
   * База данных находит информацию о пациенте и возвращает ее веб-сайту.
   * Результат может включать информацию о наличии или отсутствии заболеваний.
4. **Веб-сайт отображает результат:**
   * Веб-сайт получает информацию от базы данных и обрабатывает ее.
   * Скрипт проверяет наличие заболеваний у клиента.
   * Веб-сайт отображает результат проверки клиенту.

**Варианты:**

* **Заболевание обнаружено:**
  + Скрипт сообщает клиенту о наличии заболевания.
  + Скрипт может предоставить клиенту дополнительную информацию о заболевании,
    - например, описание, симптомы, методы лечения.
* **Заболевание не обнаружено:**
  + Скрипт сообщает клиенту, что у него нет заболеваний.
  + Скрипт может порекомендовать клиенту пройти профилактический осмотр.

**Преимущества:**

* **Наглядность:** UML-диаграмма позволяет визуально представить процесс взаимодействия между клиентом, веб-сайтом и базой данных.
* **Понятность:** Диаграмма понятна как техническим специалистам, так и людям без технических знаний.
* **Эффективность:** Диаграмма помогает выявить и устранить потенциальные проблемы в процессе взаимодействия.



Рсунок № 1

2.2.2 Написание програмного когда

* **Клиент:** Пользователь, инициирующий процесс проверки наличия заболеваний.
* **Сервер:** Центральный компонент системы, реализованный на Python,
  + отвечающий за обработку запросов клиентов
  + и взаимодействие с базой данных SQLite.
* **База данных:** Хранилище информации о пациентах,
  + реализованное на SQLite,
  + содержит ФИО, дату рождения
  + и данные о наличии или отсутствии заболеваний.

**Действия:**

**1. Прием запроса от клиента:**

* Сервер получает запрос от клиента (например, через HTTP POST).
* Запрос может содержать информацию о пациенте (ФИО, дата рождения)
  + и другие релевантные сведения.

**2. Валидация запроса:**

* Серверная часть, написанная на Python,
  + проверяет корректность и полноту полученных данных.
  + Некорректный запрос может привести к возврату сообщения об ошибке клиенту.

**3. Обработка запроса:**

* Сервер формирует SQL-запрос к базе данных SQLite,
  + используя информацию из полученного запроса.
  + Запрос отправляется в базу данных.

**4. Взаимодействие с базой данных:**

* База данных SQLite получает SQL-запрос от сервера.
* Она выполняет поиск информации о пациенте,
  + используя предоставленные ФИО и дату рождения.
  + Если информация о пациенте найдена,
    - база данных проверяет наличие или отсутствие заболеваний.
  + Результат поиска и проверки формируется в ответный пакет.

**5. Обработка ответа базы данных:**

* Сервер получает ответный пакет от базы данных SQLite.
* Серверная часть, написанная на Python,
  + анализирует полученную информацию.
  + В зависимости от результата:
    - **Заболевание обнаружено:**
      * Сервер формирует ответ клиенту,
        + уведомляя о наличии заболевания.
        + Могут быть добавлены дополнительные сведения о заболевании.
    - **Заболевание не обнаружено:**
      * Сервер сообщает клиенту об отсутствии заболеваний.
      * Может быть рекомендовано профилактическое обследование.

**6. Отправка ответа клиенту:**

* Сервер отправляет подготовленный ответ клиенту.
* Ответ может быть в формате JSON, XML или другом подходящем формате.

**Варианты обработки ошибок:**

* **Некорректный запрос:**
  + Сервер возвращает клиенту сообщение об ошибке
    - с указанием причины ошибки.
* **Ошибка базы данных:**
  + Сервер регистрирует ошибку
    - и предпринимает попытку повторить запрос.
  + Если повторный запрос не удается,
    - сервер сообщает клиенту о технической проблеме.
* **Не найдена информация о пациенте:**
  + Сервер сообщает клиенту, что информация о пациенте не найдена.

**Дополнительные соображения:**

* **Обеспечение безопасности:**
  + Защита данных пользователей
    - от несанкционированного доступа
    - и утечек информации.
* **Масштабируемость:**
  + Способность системы
    - обрабатывать растущий объем запросов
    - без потери производительности.
* **Мониторинг и логирование:**
  + Отслеживание работы системы
    - для выявления и устранения проблем.
* **Документация:**
  + Подробное описание системы
    - для облегчения ее поддержки и развития.