Отчет по заданию 1: База данных

1. Аргументация выбранного решения

1.1. Выбор СУБД: SQLite

Выбранное решение: SQLite - встраиваемая реляционная база данных

Аргументация выбора:

1. **Простота развертывания**
   - SQLite не требует установки отдельного сервера БД
   - База данных хранится в одном файле (`production_db.sqlite`)
   - Легко переносится между системами простым копированием файла
   - Не требует настройки пользователей, прав доступа и сетевых подключений
2. **Идеальность для учебных проектов**
   - Минимальные требования к ресурсам системы
   - Не требует административных прав для установки
   - Быстрый старт разработки без дополнительной настройки инфраструктуры
   - Достаточная функциональность для демонстрации работы с БД
3. **Совместимость и поддержка**
   - Встроенная поддержка в Python (библиотека `sqlite3`)
   - Широкая поддержка в различных языках программирования
   - Кроссплатформенность (Windows, Linux, macOS)
   - Активная поддержка и развитие проекта

4. **Соответствие требованиям проекта**
    - Проект не требует высокой нагрузки (учебная практика)
    - Не требуется одновременная работа множества пользователей
    - Данные статичны и не требуют сложных транзакций
    - Размер данных небольшой (172 записи)
5. **Интеграция с Python**
    - Нативная поддержка через стандартную библиотеку
    - Отличная интеграция с pandas для работы с Excel
    - Простота использования в скриптах импорта данных
    - Не требует дополнительных драйверов

2. Чем данное решение лучше других

2.1. Сравнение с альтернативными решениями

2.1.1. SQLite vs PostgreSQL

Критерий	**SQLite**	**PostgreSQL**
Установка	Не требуется (встроена)	Требует установки сервера
Настройка	Минимальная	Требует настройки пользователей, БД
Размер	Один файл (~100 КБ)	Отдельный сервер (~200 МБ)
Производительность	Отлично для малых объемов	Отлично для больших объемов
Сложность	Простая	Средняя/Высокая

Для учебного проекта	✅ Идеально	❌ Избыточно

Вывод: SQLite превосходит PostgreSQL для учебного проекта по простоте использования и отсутствию необходимости в настройке инфраструктуры.

2.1.2. SQLite vs MySQL

Критерий	**SQLite**	**MySQL**
Установка	Не требуется	Требует установки сервера
Конфигурация	Автоматическая	Требует ручной настройки
Порты и сеть	Не требуется	Требует настройки портов
Резервное копирование	Копирование файла	Требует специальных утилит
Для разработки	✅ Мгновенный старт	❌ Требует времени на настройку

> **Вывод:** SQLite обеспечивает мгновенный старт разработки без необходимости настройки сервера БД.

2.1.3. SQLite vs NoSQL (MongoDB)

Критерий	**SQLite**	**MongoDB**
Структура данных	Реляционная (SQL)	Документная (NoSQL)
Схема	Строгая типизация	Гибкая схема
Запросы	SQL (стандартизированный)	Собственный язык запросов
Для реляционных данных	✅ Идеально	❌ Не подходит
Сложность	Простая	Средняя

> **Вывод:** SQLite лучше подходит для реляционных данных с четкой структурой, что соответствует требованиям проекта.

2.2. Преимущества выбранного решения

2.2.1. Технические преимущества

1. **Нулевая конфигурация**
    - База данных создается автоматически при первом подключении
    - Не требуется настройка пользователей, паролей, прав доступа
    - Не требуется управление процессами и сервисами
2. **Простота резервного копирования**
    - Резервная копия = копирование одного файла
    - Можно использовать стандартные инструменты ОС
    - Легко версионировать через Git (для небольших БД)
3. **Производительность для малых объемов**
    - Отличная производительность для небольших БД (< 1 ГБ)
    - Нет сетевых задержек (локальный доступ)
    - Минимальные накладные расходы
4. **Стандартизация SQL**
    - Поддержка стандартного SQL синтаксиса
    - Легко мигрировать на другие СУБД при необходимости
    - Знания SQL применимы к другим системам

2.2.2. Преимущества для разработки

1. **Быстрый старт**
    - Можно начать работу сразу, без установки дополнительного ПО

- Не требуется знание администрирования БД
- Фокус на разработке, а не на настройке инфраструктуры

2. **Простота отладки**
   - Можно открыть БД в любом SQLite браузере
   - Легко просматривать и редактировать данные
   - Простое логирование и трассировка запросов
3. **Портативность**
   - Проект можно запустить на любой системе
   - Не требуется установка дополнительных зависимостей
   - Легко передать проект другим разработчикам
4. **Интеграция с инструментами**
   - Отличная поддержка в IDE (например, DB Browser for SQLite)
   - Интеграция с pandas для анализа данных
   - Поддержка в различных языках программирования

2.3. Реализованные возможности

2.3.1. Структура базы данных

Реализовано **5 взаимосвязанных таблиц**:

1. **material_type** - Типы материалов (4 записи)
2. **product_type** - Типы продукции (6 записей)
3. **workshops** - Цеха производства (12 записей)
4. **products** - Продукция (20 записей)
5. **product_workshops** - Связь продукции и цехов (130 записей)

Всего записей: 172

2.3.2. Реализованные связи

- **Внешние ключи (Foreign Keys)** для обеспечения целостности данных
- **Уникальные ограничения (UNIQUE)** для предотвращения дублирования
- **Первичные ключи (Primary Keys)** с автоинкрементом
- **Связь многие-ко-многим** через промежуточную таблицу

2.3.3. Автоматизация

- **Скрипт создания БД** (`create_database.py`)

• Автоматический импорт данных из Excel файлов **• Обработка ошибок** при импорте **• Валидация данных** перед вставкой

- **Статистика импорта** для контроля процесса