# Документация по структуре проекта

Проект организован по модульной структуре с разделением на различные папки и файлы, каждый из которых отвечает за определённую функциональность:

**Корневая директория**

• .gitignore — Список файлов и директорий, которые должны игнорироваться системой контроля версий Git.

• composer.json — Управляет зависимостями проекта и конфигурацией для Composer (менеджера зависимостей PHP).

• composer.lock — Фиксирует точные версии зависимостей для обеспечения стабильности и совместимости.

• normalize\_tables.sql — SQL-скрипт для нормализации таблиц в базе данных.

**Директория config**

• .env — Файл для хранения переменных окружения, таких как настройки базы данных, API ключи и другие конфиденциальные данные.

• config.php — Файл конфигурации, содержащий основные настройки проекта, которые могут быть использованы во всем приложении.

**Директория src**

В директории src находятся все основные модули и классы, разделённые по функциональным областям:

**Папка API**

• API.php — Основной класс для работы с API, реализующий основные методы для обработки запросов.

• InterfaceAPI.php — Интерфейс для класса API.php, определяющий необходимые методы и правила для взаимодействия с API.

**Папка DataBase**

• DataBase.php — Класс для работы с базой данных, включающий методы для выполнения запросов, подключения и управления данными.

• InterfaceDatabase.php — Интерфейс для класса DataBase.php, определяющий методы и структуру взаимодействия с базой данных.

**Папка Order**

• Barcode.php — Класс для генерации штрих-кодов для заказов.

• InterfaceBarcode.php — Интерфейс для класса Barcode.php, определяющий основные методы работы со штрих-кодами.

• InterfaceOrder.php — Интерфейс для класса Order.php, определяющий основные методы для работы с заказами.

• Order.php — Класс для управления заказами, включающий основные методы создания, обновления и получения информации о заказах.

**Директория vendor**

Содержит зависимости и библиотеки, установленные с помощью Composer.

# Обоснование решения

1. Уникальные баркоды для каждого билета: Таблица tickets позволяет создавать отдельные записи для каждого билета, привязывая их к конкретной записи в order\_tickets. Это обеспечивает уникальный баркод для каждого билета, позволяя проверять их индивидуально.

2. Масштабируемость: Если заказ включает несколько типов билетов, каждый тип будет иметь свои отдельные записи в order\_tickets, а для каждого билета можно создать отдельный баркод в tickets.

3. Нормализация: Данные разложены на отдельные таблицы, что уменьшает дублирование и обеспечивает целостность данных. Баркоды сохраняются на уровне отдельных билетов, что делает структуру более логичной и удобной для проверки билетов.