1. **Выбор и обоснование средств проектирования и реализации** 
   1. **Средства проектирования**

Для проектирования системы использовались следующие инструменты:

* **DBDesigner** выбран как инструмент для проектирования баз данных благодаря своему интуитивному интерфейсу, широкому набору функций, включая генерацию скриптов для различных СУБД, и статусу свободного программного обеспечения с открытым исходным кодом, что обеспечивает гибкость и доступность для разработчиков.
* **Figma** был выбран из-за его высокой гибкости и универсальности. Это интуитивно понятное средство, которое обеспечивает не только дизайн интерфейсов, но и создание детальных прототипов. А также возможность работать в браузере без необходимости установки дополнительного программного обеспечения повлияла на выбор в пользу этого инструмента.
* **StarUML** был выбран в качестве средства проектирования из-за его многофункциональности и удобства использования. Этот инструмент предоставляет широкий спектр возможностей для моделирования, включая диаграммы классов, вариантов использования и многие другие. Интерфейс StarUML интуитивно понятен, что упрощает процесс создания и редактирования диаграмм.
  1. **Средства реализации**

Выбор использования технологий React, Node.js, Express.js и PostgreSQL для реализации проекта был обоснован рядом причин.

Во-первых, React обеспечивает создание приложения с использованием компонентов, которые могут быть многократно использованы. В случае автоматизированной системы для художественной школы, это может включать компоненты для управления расписанием занятий, регистрацией студентов, отслеживания прогресса и других функций. Компоненты облегчают поддержку и модификацию системы в будущем.

Во-вторых, выбор Node.js и Express.js для серверной части приложения обусловлен их преимуществами в области разработки веб-приложений. Node.js является средой выполнения JavaScript на стороне сервера, что позволяет использовать единый язык программирования как на клиентской, так и на серверной стороне. Express.js, в свою очередь, является легковесным и гибким фреймворком для создания веб-приложений на основе Node.js. Он предоставляет удобные инструменты для маршрутизации, обработки запросов и других задач, необходимых для разработки серверной части приложения.

PostgreSQL была выбрана в качестве базы данных для хранения информации и выполнения SQL-запросов. PostgreSQL является одной из самых популярных и широко используемых реляционных баз данных. Она обладает простым в использовании интерфейсом, хорошей производительностью и поддерживает полный набор функций SQL. В контексте проекта позволяет эффективно хранить и организовывать данные.

Таким образом, выбор технологий React, Node.js, Express.js и PostgreSQL для реализации данного проекта обусловлен их преимуществами и соответствием требованиям проекта, включая гибкость, эффективность и удобство использования в контексте создания визуальной новеллы для обучения SQL-запросам.

1. **Проектирование архитектуры приложения**

Для приложения была выбрана клиент-серверная архитектура, которая представлена на рисунке 1.

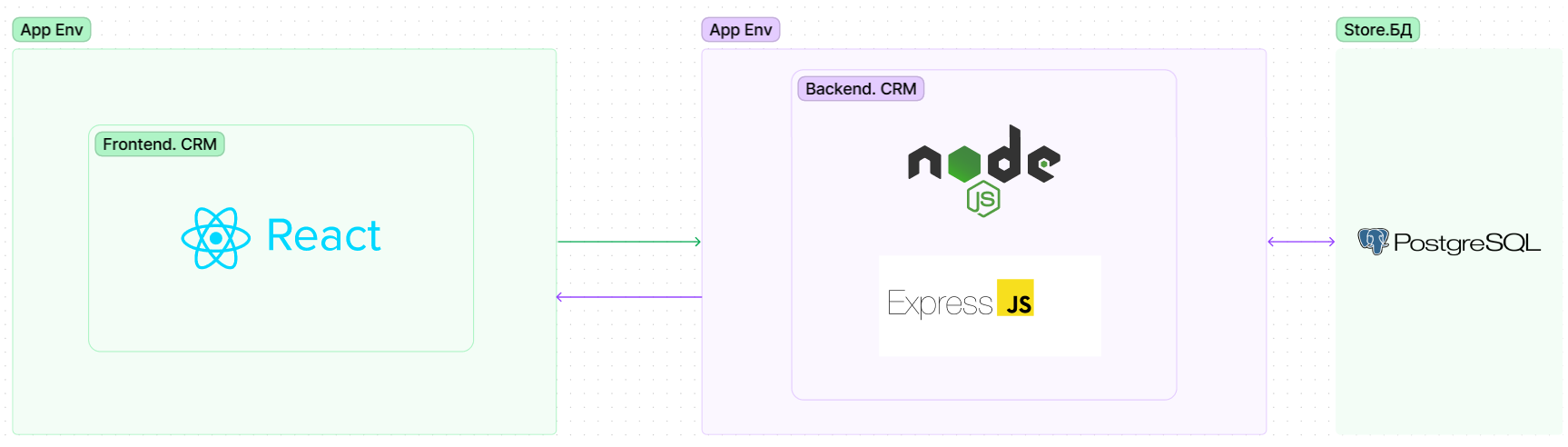
****

Рисунок 1 - Архитектура приложения

Клиент-серверная архитектура представляет собой структурный подход к организации и взаимодействию программных систем. В этой архитектуре функции приложения распределены между клиентской (пользовательской) и серверной (центральной) частями. На клиентской стороне располагается интерфейс пользователя, обеспечивающий визуальное взаимодействие. Кроме того, часть логики приложения может быть реализована на клиенте для повышения отзывчивости.

Серверная часть, в свою очередь, содержит основную бизнес-логику, обработку данных и принятие решений. Здесь также осуществляется управление данными, и их хранение может быть реализовано через работу с базой данных или другими хранилищами данных. Сервер принимает запросы от клиента, обрабатывает их и возвращает результаты выполнения запросов.

Взаимодействие между клиентом и сервером осуществляется через клиент-серверное взаимодействие по сети. Это может быть реализовано с использованием различных сетевых технологий, таких как HTTP, WebSocket и другие. Ключевое в этой архитектуре — асинхронность взаимодействия, что способствует эффективности приложения.

Данная архитектура была выбрана, потому что она обеспечивает модульность приложения, позволяя разделять бизнес-логику и пользовательский интерфейс. Это содействует масштабируемости и облегчает управление доступом к данным.

1. **Проектирование хранилища базы данных**
   1. **Основные сущности**

Система имеет следующие основные сущности (табл. 1).

Таблица 1 – Описание сущностей

|  |  |
| --- | --- |
| **Сущность** | **Атрибуты** |
| Преподаватель | ФИО преподавателя, пол, паспорт, телефон, дата рождения |
| Группа | Название группы, ID преподавателя, ID учебной программы |
| Ученик | ФИО ученика, дата рождения, телефон, почта, ID группы |
| Расписание | ID преподавателя, ID предмета, ID времени, ID дня, ID группы, ID кабинета |
| Посещаемость | ID расписания, ID ученика, присутствие, причина |
| Успеваемость | ID расписания, ID ученика, оценка |

* 1. **Логическая модель данных**

На рисунке 2 представлена логическая модель данных.

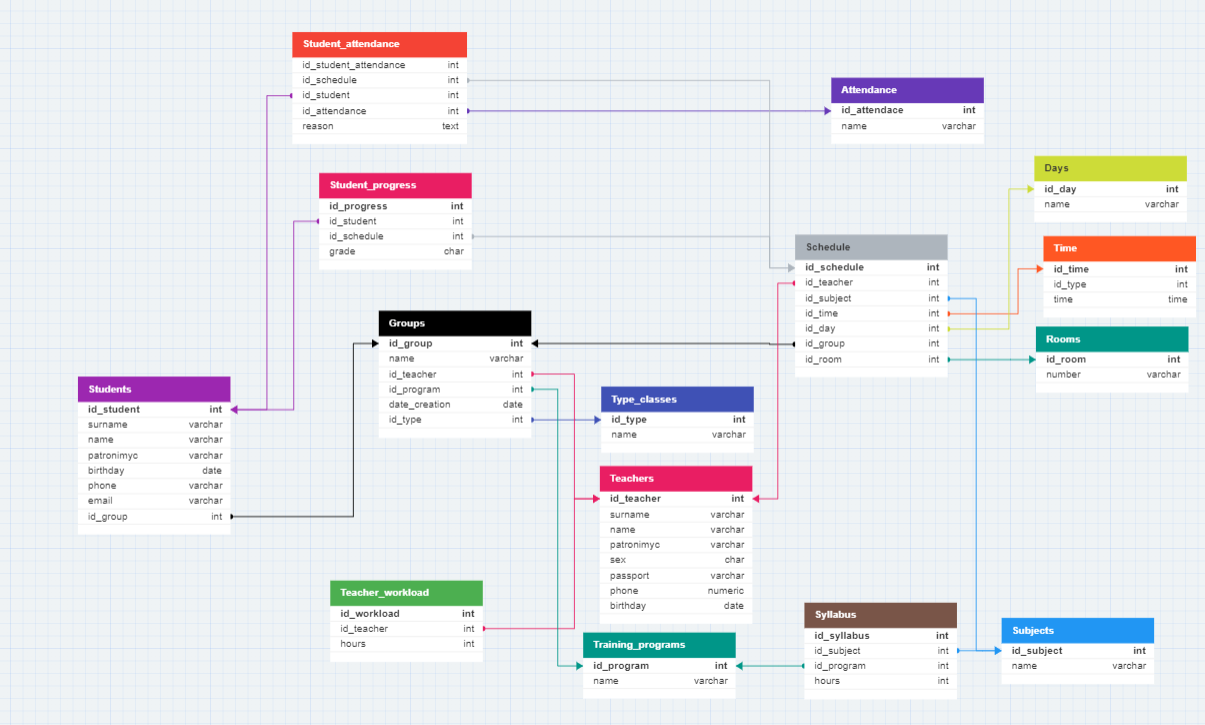


Рисунок 2 - Логическая модель данных

1. **Проектирование пользовательского интерфейса**