МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ

ГБОУ школа №444

8«И» класс

Командный кейс «Управление столовой»

Профиль «Информационные технологии»

https://autogreatfood.ru

Выполнили ученики:	Макеев Илья Константинович Михайлов Евгений Вячеславович
Куратор:	Колтунов Роман Павлович

Москва 2026

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Сейчас в большинстве школьных столовых всё ещё используется ручной учёт заказов и бумажные талоны. Это создаёт много проблем: часто возникают ошибки при подсчёте, образуются очереди, персонал тратит много времени на рутинную работу. Кроме того, ученикам неудобно планировать своё питание, а администрация не может быстро получить статистику по работе столовой. Наш проект решает эти проблемы с помощью современных информационных технологий. Мы разработали систему, которая автоматизирует весь процесс — от заказа еды до её выдачи и формирования отчётов.

Цель проекта – создать удобную информационную систему управления школьным питанием, которая работает на современном технологическом стеке с использованием базы данных PostgreSQL.

Для достижения цели мы поставили следующие задачи:

1. Выбрать подходящие языки программирования, фреймворки и базу данных, объяснить свой выбор.
2. Разработать схемы, показывающие структуру и работу системы.
3. Спроектировать базу данных.
4. Написать код системы (клиентскую и серверную части).
5. Протестировать готовое программное обеспечение.

2. АНАЛИТИЧЕСКАЯ ЧАСТЬ

2.1. Описание предметной области

Наша система автоматизирует организацию питания в школе и работает с тремя категориями пользователей: учениками, персоналом столовой (поварами) и администраторами. Каждая категория имеет свои функции и возможности.

Возможности для учеников

Ученики могут просматривать меню на текущий день и на месяц вперёд. Это позволяет им заранее планировать своё питание. Система позволяет формировать заказы на отдельные дни или покупать абонементы сразу на несколько дней. При покупке абонемента ученик сам выбирает даты, на которые он хочет питание.

Каждый ученик имеет личный баланс счёта, с которого списываются деньги за заказы. Баланс можно пополнять, и в любой момент можно посмотреть историю всех операций. При регистрации в системе каждому ученику автоматически начисляется начальный баланс для оплаты первых заказов.

Ученики могут оставлять отзывы о блюдах после получения заказа. Это помогает администрации оценивать качество питания и корректировать меню с учётом предпочтений учащихся. Отзывы включают оценку по пятибалльной шкале и возможность оставить комментарий.

Система учитывает пищевые аллергии и предпочтения учащихся. При регистрации ученик может указать аллергены и продукты, которые ему нельзя употреблять. В меню автоматически отображаются предупреждения о блюдах, содержащих указанные аллергены, что помогает избежать проблем со здоровьем.

Возможности для персонала столовой

Повара работают с оптимизированным интерфейсом, который позволяет быстро выдавать заказы. Они видят список всех заказов с возможностью фильтрации по типу питания (завтрак, обед, полдник). Выдача происходит по номеру заказа. Важная особенность — система автоматически фиксирует каждую выдачу и не позволяет выдать один и тот же заказ дважды. Это исключает злоупотребления и ошибки.

Повара могут отслеживать остатки продуктов на складе и формировать заявки на закупку недостающих ингредиентов. Система автоматически рассчитывает необходимое количество продуктов на основе запланированного меню и текущих остатков. Заявки отправляются администратору для согласования.

Возможности для администраторов

Администраторы имеют доступ к аналитическим инструментам и панели управления. Они могут отслеживать ключевые показатели: количество активных пользователей, общий объём заказов, выручку и её динамику по дням. Также администраторы управляют учётными записями пользователей, могут корректировать меню и изменять цены на блюда. Система формирует детализированные отчёты, которые помогают анализировать эффективность работы столовой и планировать закупки.

Модуль уведомлений

Система включает модуль уведомлений для всех категорий пользователей. Ученики получают уведомления о: новых блюдах в меню, изменениях в их заказах, успешном пополнении баланса, необходимости пополнения при недостаточных средствах. Повара получают уведомления о: новых заказах, требующих подготовки, критически низких остатках продуктов на складе. Администраторы получают уведомления о: новых заявках на закупку, требующих согласования, аномалиях в статистике заказов, отзывах с низкими оценками, требующих внимания. Все уведомления отображаются в интерфейсе системы и опционально могут быть отправлены на электронную почту.

Администраторы согласовывают заявки на закупку продуктов, поступающие от поваров. Они могут просматривать детализацию заявок, корректировать количество позиций и отклонять или утверждать заявки.

После утверждения заявка передаётся в отдел закупок. Система формирует отчёты по питанию и затратам, включая анализ популярности блюд, расход продуктов по периодам и динамику затрат.

2.2. Выбор технологий

При выборе технологий мы учитывали несколько факторов: производительность, возможность масштабирования решения в будущем, наличие специалистов на рынке, богатство доступных библиотек и простоту развёртывания готовой системы.

Клиентская часть

React — мы выбрали эту библиотеку для создания пользовательского интерфейса по нескольким причинам. Во-первых, она обеспечивает высокую производительность благодаря виртуальному DOM — специальной технологии, которая обновляет только изменившиеся части страницы. Во-вторых, React использует компонентный подход: мы создаём отдельные компоненты (кнопки, формы, карточки) и используем их многократно. Это упрощает разработку и поддержку кода. В-третьих, у React очень большая экосистема — есть готовые решения практически для любой задачи.

Vite — современный инструмент для сборки проекта. Он значительно ускоряет разработку, потому что использует нативные ES-модули и очень быстро запускает dev-сервер. При сборке финальной версии Vite создаёт оптимизированные файлы, которые быстро загружаются в браузере.

Серверная часть

Node.js — платформа, которая позволяет запускать JavaScript на сервере. Главное преимущество для нас — использование одного языка программирования и на клиенте, и на сервере. Это упрощает разработку, потому что не нужно переключаться между разными языками. Кроме того, Node.js хорошо подходит для веб-приложений благодаря асинхронной модели

работы — сервер может эффективно обрабатывать множество одновременных запросов.

Express — минималистичный и гибкий веб-фреймворк для Node.js. Он предоставляет базовый функционал для обработки HTTP-запросов и большой выбор готовых модулей (middleware) для различных задач: парсинга запросов, работы с CORS, управления сессиями, логирования.

Безопасность

bcryptjs — библиотека для безопасного хеширования паролей. Она добавляет к каждому паролю случайную строку (соль) и многократно шифрует результат. Это делает практически невозможным подбор пароля, даже если злоумышленник получит доступ к базе данных.

express-session — модуль для управления сессиями пользователей. Он использует httpOnly cookies, которые нельзя прочитать из JavaScript. Это защищает от XSS-атак — попыток украсть идентификатор сессии через вредоносный код на странице.

2.3. Выбор базы данных PostgreSQL

Для хранения данных мы выбрали систему управления базами данных PostgreSQL. Это оптимальное решение с точки зрения надёжности, производительности и возможностей масштабирования.

Надёжность и функциональность

PostgreSQL — это объектно-реляционная СУБД с открытым исходным кодом, которая зарекомендовала себя за десятилетия использования в крупных проектах. Она полностью поддерживает стандарт SQL и предлагает много дополнительных возможностей: работу с JSON, полнотекстовый поиск, сложные типы данных.

Производительность

PostgreSQL использует технологию MVCC (Multi-Version Concurrency Control). Это означает, что система создаёт несколько версий данных, благодаря чему читающие и пишущие операции не блокируют друг друга. Это очень важно для нашей системы: в часы пик (например, во время большой перемены) множество учеников одновременно делают заказы, а повара в это же время выдают еду. PostgreSQL справляется с такой нагрузкой без замедления.

Целостность данных

PostgreSQL предоставляет продвинутые механизмы для обеспечения целостности данных. Можно создавать различные типы ключей и проверочные ограничения. Транзакционная модель полностью соответствует требованиям ACID, что гарантирует надёжность всех финансовых операций. Например, когда ученик делает заказ, деньги спишутся со счёта только если заказ действительно создан — никаких потерь денег или «подвисших» операций.

Расширяемость

PostgreSQL позволяет создавать пользовательские функции прямо в базе данных. Поддерживается несколько языков программирования, включая PL/pgSQL для хранимых процедур. Это даёт возможность реализовать сложную бизнес-логику на уровне базы данных, что обеспечивает дополнительный уровень валидации данных.

Отказоустойчивость

Система репликации PostgreSQL поддерживает как асинхронную, так и синхронную репликацию. Это позволяет создавать копии базы данных на случай сбоев. Логическая репликация даёт возможность реплицировать

отдельные таблицы, что можно использовать для создания аналитических копий без нагрузки на основной сервер.

Важным фактором является наличие большого активного сообщества разработчиков. PostgreSQL регулярно обновляется, уязвимости оперативно исправляются. Есть обширная документация на русском языке и много специалистов, что упрощает поддержку системы в будущем.

3. ПРОЕКТНАЯ ЧАСТЬ

3.1. Структурная схема системы

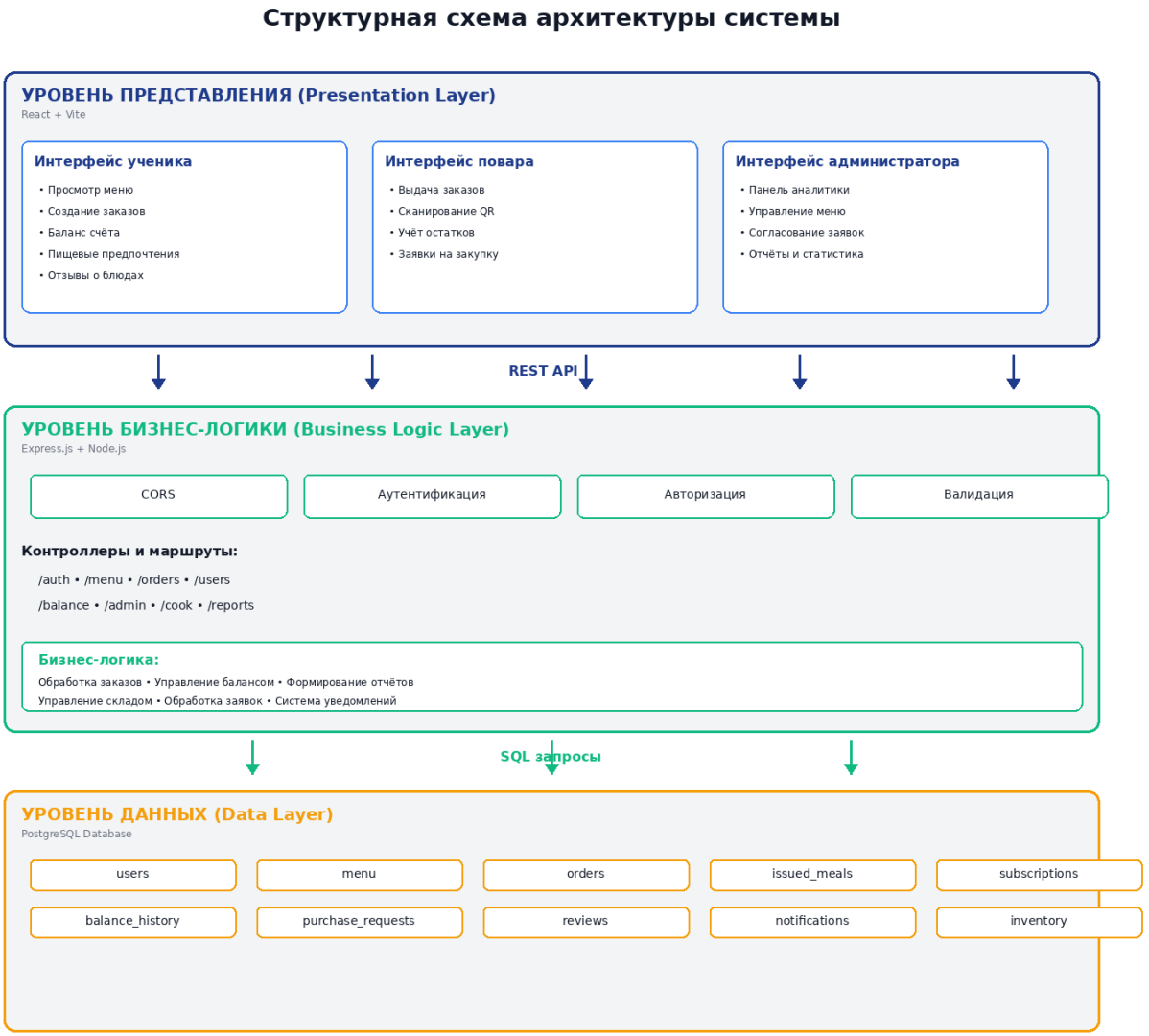

Рис. 1. Структурная схема проекта

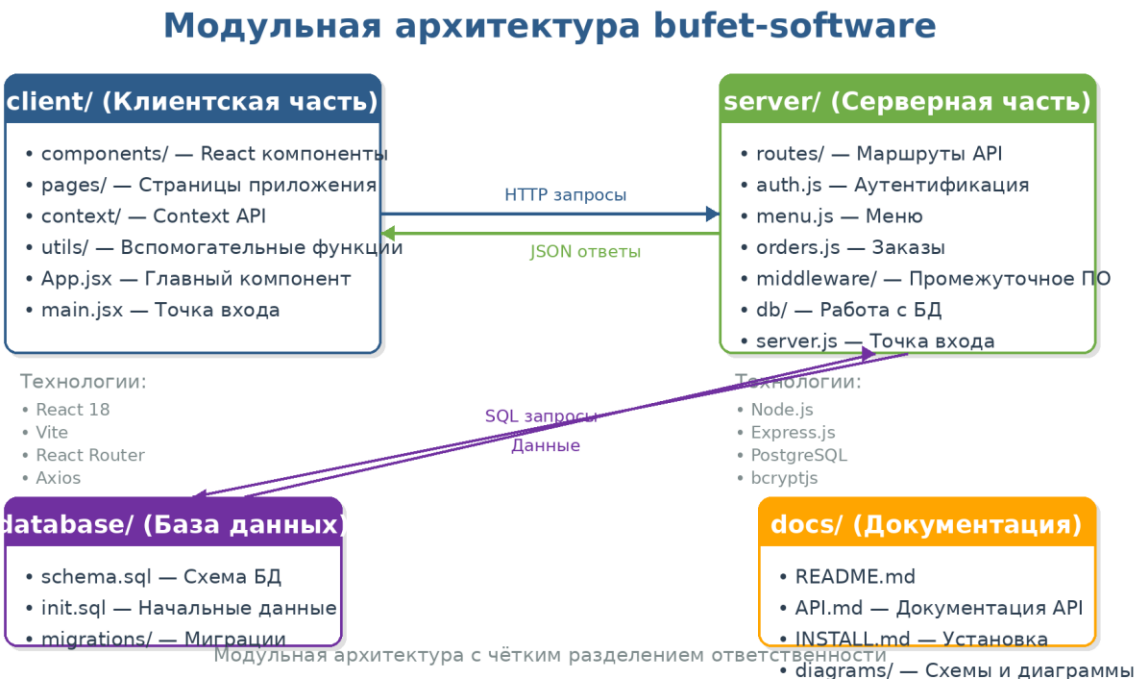

Рис. 2. Модульная архитектура проекта

Структурная схема показывает, как устроена наша система и как взаимодействуют её компоненты. Мы использовали классическую трёхуровневую архитектуру с чётким разделением: уровень представления, уровень бизнес-логики и уровень данных.

На уровне представления находится клиентское приложение, которое разделено на три интерфейса в соответствии с ролями пользователей. Все компоненты написаны на React и общаются с сервером только через REST API. Это обеспечивает слабую связанность — клиент и сервер можно разрабатывать и обновлять независимо друг от друга.

Уровень бизнес-логики представлен Express-сервером. Он обрабатывает входящие HTTP-запросы через систему middleware. Каждый запрос последовательно проходит через обработчики: сначала проверяется CORS (разрешён ли доступ с данного домена), затем middleware управления сессиями проверяет аутентификацию, потом body-parser разбирает тело запроса. Маршруты организованы по функциональным модулям, что

упрощает поддержку кода. Уровень данных — это база данных PostgreSQL, которая хранит всю информацию системы.

3.2. Функциональная схема программы

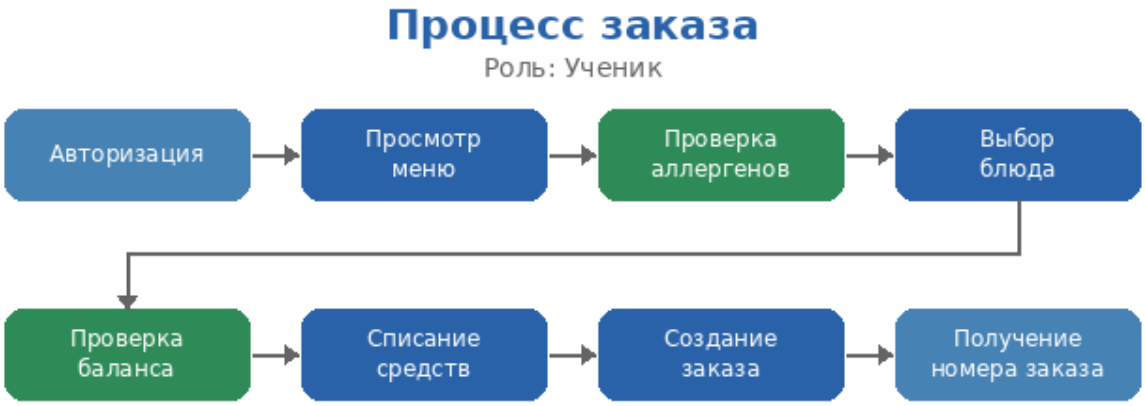

Рис. 3. Роль «Ученик». Процесс заказа

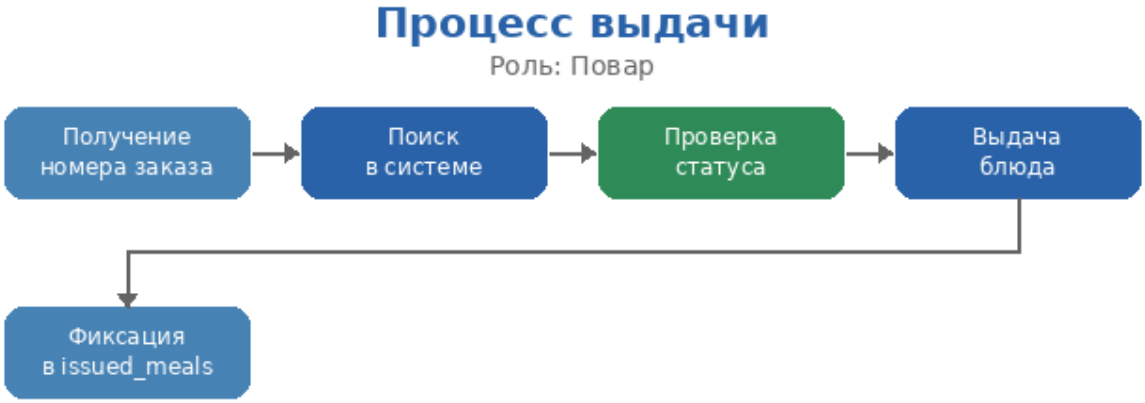

Рис. 4. Роль «Повар». Процесс выдачи заказа

Рис. 5. Роли «Повар» и «Администратор». Процесс закупки продуктов.

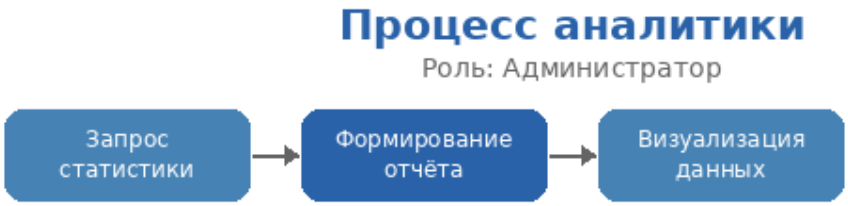

Рис. 6. Роль «Администратор». Процесс аналитики.

Функциональная схема показывает основные процессы системы и последовательность операций при выполнении типовых действий.

Схема отражает два ключевых процесса. При создании заказа система выполняет несколько проверок: сначала проверяет, достаточно ли денег на балансе ученика. Затем атомарно (то есть как одну неделимую операцию) списывает сумму с баланса и создаёт запись заказа для последующей идентификации при выдаче.

Процесс выдачи включает множественные проверки. Система проверяет статус заказа, чтобы предотвратить дублирование выдачи одного и того же

блюда. Это защищает от ситуации, когда ученик может получить одно блюдо несколько раз.

3.3. Блок-схема основного алгоритма

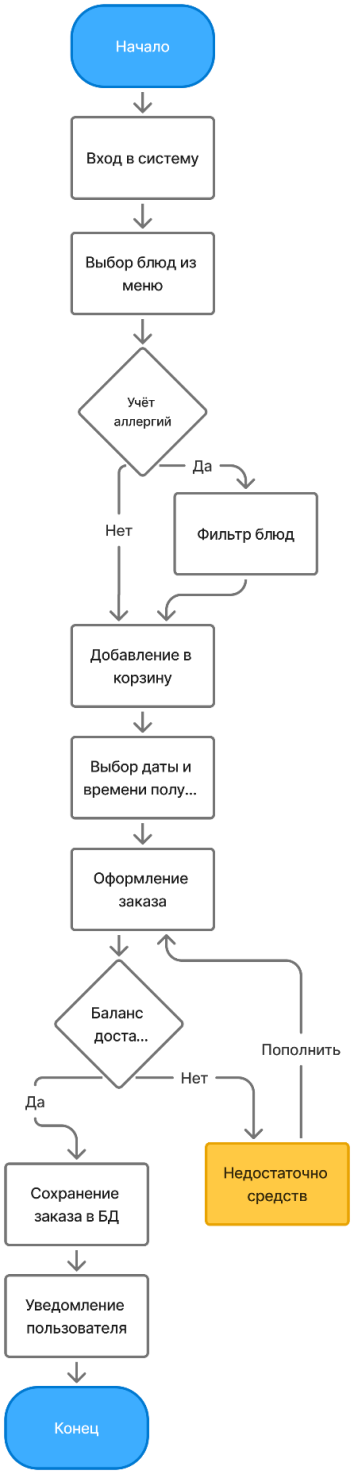

Рис. 7. Блок-схема основного алгоритма «Создание заказа»

Блок-схема детально показывает алгоритм обработки запроса на создание заказа. Это центральная операция в системе, которая требует особого внимания к обеспечению целостности данных.

Критический момент этого алгоритма — использование транзакций базы данных. Транзакция объединяет несколько операций в одну: проверка баланса, списание средств и создание заказа выполняются как единое целое. Если что-то пойдёт не так на любом этапе (например, недостаточно средств или произошёл сбой), вся транзакция откатывается — база данных возвращается в исходное состояние.

Это также защищает от проблем при одновременных операциях. Например, если ученик одновременно с двух устройств попытается сделать заказ, транзакционная модель гарантирует, что деньги спишутся корректно и не возникнет несогласованного состояния данных.

3.4. Схема базы данных

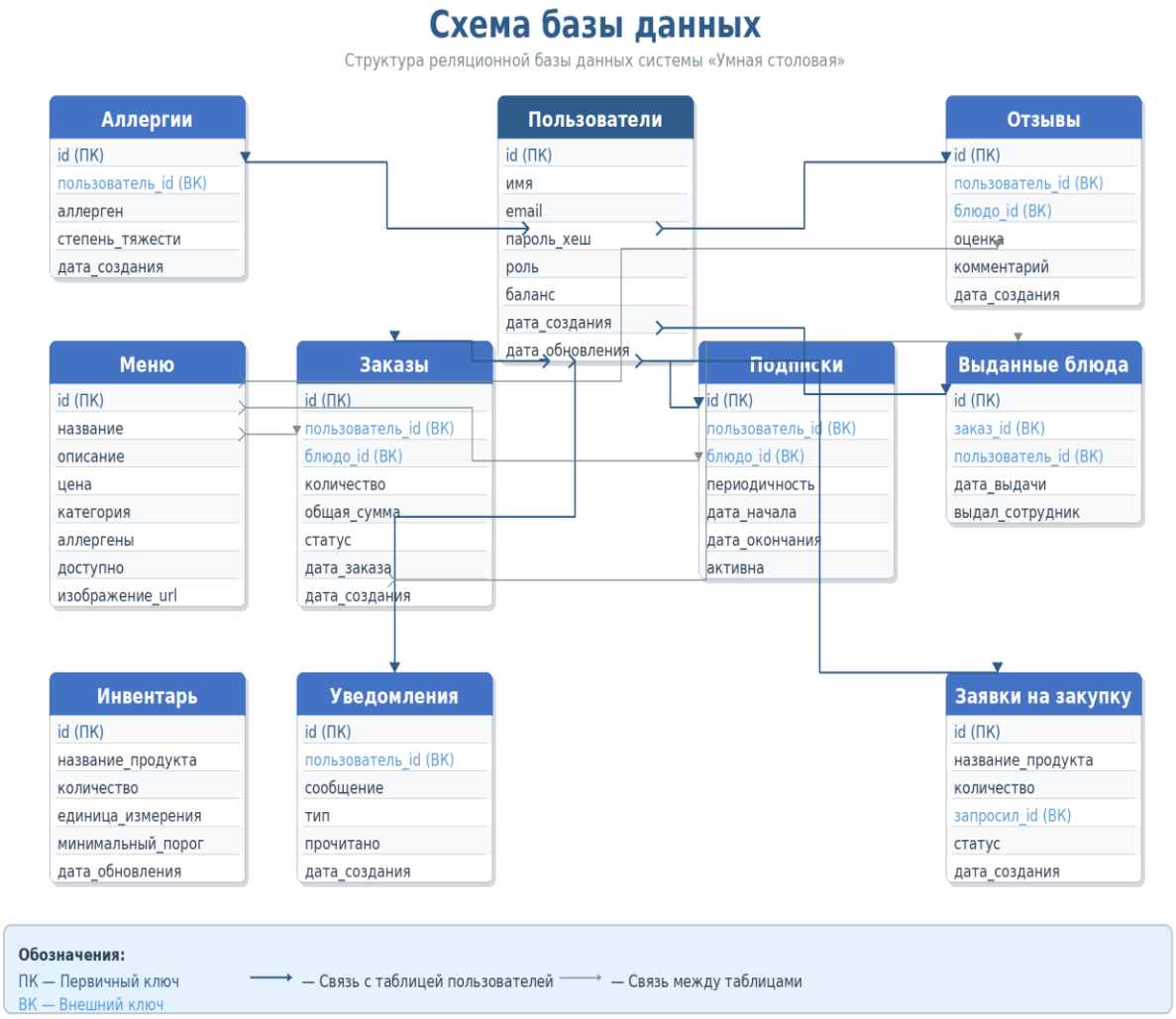

Рис. 8. Схема Базы Данных управления столовой

Схема базы данных спроектирована в соответствии с требованиями третьей нормальной формы. Это означает, что данные организованы так, чтобы исключить избыточность и обеспечить целостность. В базе данных пять основных таблиц.

Таблица users — центральная таблица, которая содержит учётные данные всех пользователей независимо от роли. Поле role определяет права доступа и функциональность, доступную пользователю. Пароли хранятся в виде bcrypt-хешей с автоматически сгенерированной солью — в базе данных

невозможно увидеть настоящий пароль. Для учеников дополнительно хранится номер класса и баланс счёта.

Таблица menu содержит информацию о блюдах, доступных для заказа. Уникальное ограничение на комбинацию даты, типа питания и названия блюда предотвращает дублирование позиций в меню. Поле available позволяет временно скрыть блюдо из доступных без удаления записи — это важно для сохранения истории заказов.

Таблица orders фиксирует все заказы учеников. Важная особенность: цена копируется из таблицы menu в момент создания заказа. Это обеспечивает неизменность стоимости заказа, даже если потом цены в меню изменятся. Каждый заказ имеет уникальный идентификатор для быстрой идентификации при выдаче.

Таблица subscriptions хранит информацию о купленных абонементах — питании на несколько дней вперёд.

Таблица issued_meals фиксирует факты выдачи блюд с указанием, какой повар выдал и в какое точное время. Эта информация используется для аналитики работы столовой и, главное, предотвращает повторную выдачу одного заказа.

3.5. Тестирование системы

Система прошла комплексное тестирование с проверкой основных сценариев использования и граничных случаев. В таблице 1 представлены результаты ключевых тестов.

Таблица 1. Тестирование основных сценариев использования

№	Сценарий испытания	Ожидаемый результат	Статус
1	Попытка повторной отметки питания одним учеником	Система блокирует повторную выдачу, отображает сообщение об ошибке	Пройден
2	Создание заказа при недостаточном балансе	Заказ не создаётся, отображается предупреждение о необходимости пополнения	Пройден
3	Недостаток продуктов на складе при формировании меню	Система создаёт автоматическую заявку на закупку и уведомляет повара	Пройден
4	Одновременное создание заказов с одного аккаунта	Транзакционная система обеспечивает целостность данных, только один заказ создаётся	Пройден
5	Проверка идентификации заказа при выдачезаказа	Отображается сообщение об отсутствии заказа	Пройден
6	Доступ к функциям администратора без соответствующих прав	Доступ запрещён, пользователь перенаправляется на страницу авторизации	Пройден
7	Оставление отзыва на блюдо до его получения	Функция недоступна, отзывы можно оставлять только после получения заказа	Пройден

Все критические сценарии успешно протестированы. Система корректно обрабатывает граничные случаи и защищена от типичных уязвимостей.

4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

4.1. Структура проекта

Рис. 9. Структура проекта: клиент, сервер, база данных и документация

Проект организован в модульную структуру с чётким разделением клиентского и серверного кода. Исходный код клиентской части находится в

директории src, серверная часть — в директории server. Такое разделение упрощает навигацию и позволяет разрабатывать обе части независимо.

Клиентская часть организована по принципу разделения компонентов и страниц. Компоненты — это переиспользуемые элементы интерфейса (кнопки, формы, карточки), а страницы объединяют компоненты для формирования полноценных экранов приложения. Переключение между страницами происходит через React Router без перезагрузки — это обеспечивает быструю и плавную навигацию.

Серверная часть структурирована по функциональным модулям. Каждый модуль в директории routes отвечает за определённую область функциональности: работа с пользователями, меню, заказами. Каждый модуль экспортирует Express router с набором endpoints. Это обеспечивает модульность — можно добавлять новую функциональность без изменения существующего кода.

4.2. Описание кода

Программный код разработан с соблюдением принципов чистого кода и современных практик веб-разработки.

Клиентская часть (React)

Построена на функциональных компонентах с использованием хуков для управления состоянием. Это современный подход в React, который делает код более читаемым и простым в поддержке.

Реализованы три специализированных интерфейса: для учащихся (просмотр меню, создание заказов, управление балансом), для поваров (отображение очереди заказов на выдачу, поиск по номеру заказа) и для администраторов (аналитика, статистика, управление пользователями и меню).

Серверная часть (Node.js/Express)

Архитектура основана на middleware — промежуточных обработчиках. Каждый входящий запрос последовательно проходит через несколько middleware: обработчик CORS проверяет, разрешён ли доступ с данного домена, парсер тела запроса преобразует JSON в объекты JavaScript, middleware проверки сессии определяет, авторизован ли пользователь.

Система аутентификации использует bcryptjs для хеширования паролей и защищённые сессии. Когда пользователь входит в систему, создаётся сессия и её идентификатор сохраняется в httpOnly cookie — такой cookie нельзя прочитать из JavaScript, что защищает от атак.

Взаимодействие с базой данных

Вся логика работы с PostgreSQL инкапсулирована в отдельных модулях. Используется пул соединений для эффективности — вместо создания нового соединения для каждого запроса, сервер переиспользует существующие.

Все запросы формируются с помощью параметризованных выражений. Это означает, что данные от пользователя никогда не вставляются напрямую в текст SQL-запроса, что предотвращает SQL-инъекции — один из самых распространённых типов атак на веб-приложения.

Критические операции выполняются в рамках транзакций. Например, создание заказа с одновременным списанием средств: если хотя бы одна операция не удастся, вся транзакция откатится и данные останутся в согласованном состоянии.

Валидация данных

Валдация данных реализована на двух уровнях. Клиентская валидация обеспечивает быструю обратную связь пользователю — сразу показывает ошибки в форме. Серверная валидация — обязательная проверка всех данных,

потому что клиентскую валидацию можно обойти. Это защищает от любых модифицированных или некорректных запросов.

4.3. Ссылка на репозиторий

Весь исходный код проекта размещён в открытом репозитории на платформе GitHub:
https://github.com/IlyaMakeev0/bufet-software

В репозитории содержится:

6. Полный исходный код системы (клиентская и серверная части)
7. Файлы конфигурации для развёртывания
8. Подробная документация по установке и настройке
9. История всех изменений (коммитов)

Проект распространяется под лицензией MIT. Это означает, что любой может свободно использовать код, модифицировать его и распространять свои версии.

Инструкция по развёртыванию (README)

Краткое описание: Система управления школьным питанием — веб-приложение для автоматизации процесса организации питания в образовательных учреждениях. Включает функционал для учеников (просмотр меню, заказы, баланс), поваров (выдача заказов, управление складом) и администраторов (аналитика, управление).

Инструкция по установке и запуску:

1. Клонировать репозиторий: git clone https://github.com/IlyaMakeev0/bufet-software.git
2. Перейти в директорию проекта: cd bufet-software
3. Установить зависимости для сервера: cd server && npm install
4. Установить зависимости для клиента: cd ../client && npm install

5. Настроить PostgreSQL: создать базу данных и выполнить SQL-скрипты из папки /database

6. Создать файл .env в папке server со следующими переменными:

- DATABASE_URL=postgresql://user:password@localhost:5432/canteen_db
- SESSION_SECRET=ваш_секретный_ключ
- PORT=3001

7. Запустить сервер: cd server && npm start

8. Запустить клиент (в новом терминале): cd client && npm run dev

9. Открыть браузер по адресу http://localhost:5173

Ссылка на видеопрезентацию с демонстрацией работы системы: https://rutube.ru/video/private/82a9eb25f80867b93af7d89a84c6dbe2/?p=ChJ4hmeJdLje16WF5JdLDw

4.4. Демонстрация работы

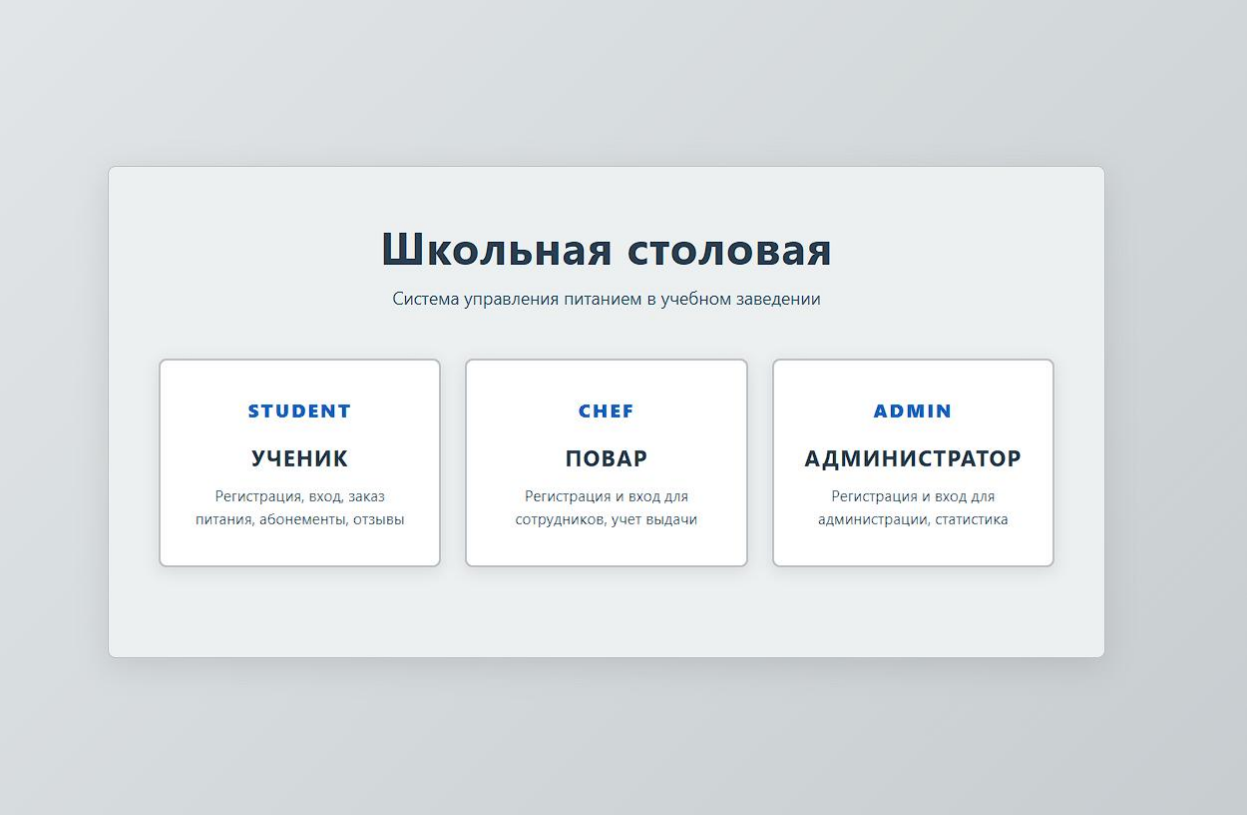

Рис. 10. Выбор типа пользователя: «Ученик», «Повар» и «Администратор»

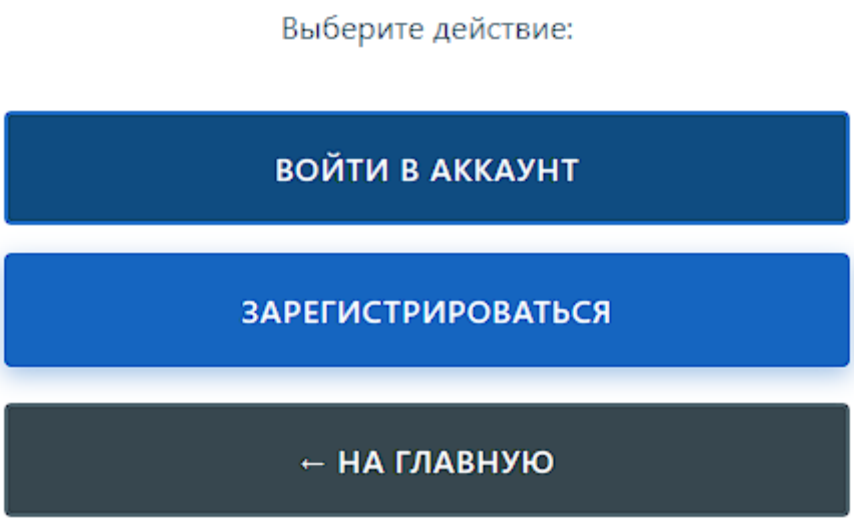

Рис. 11. Вход и регистрация

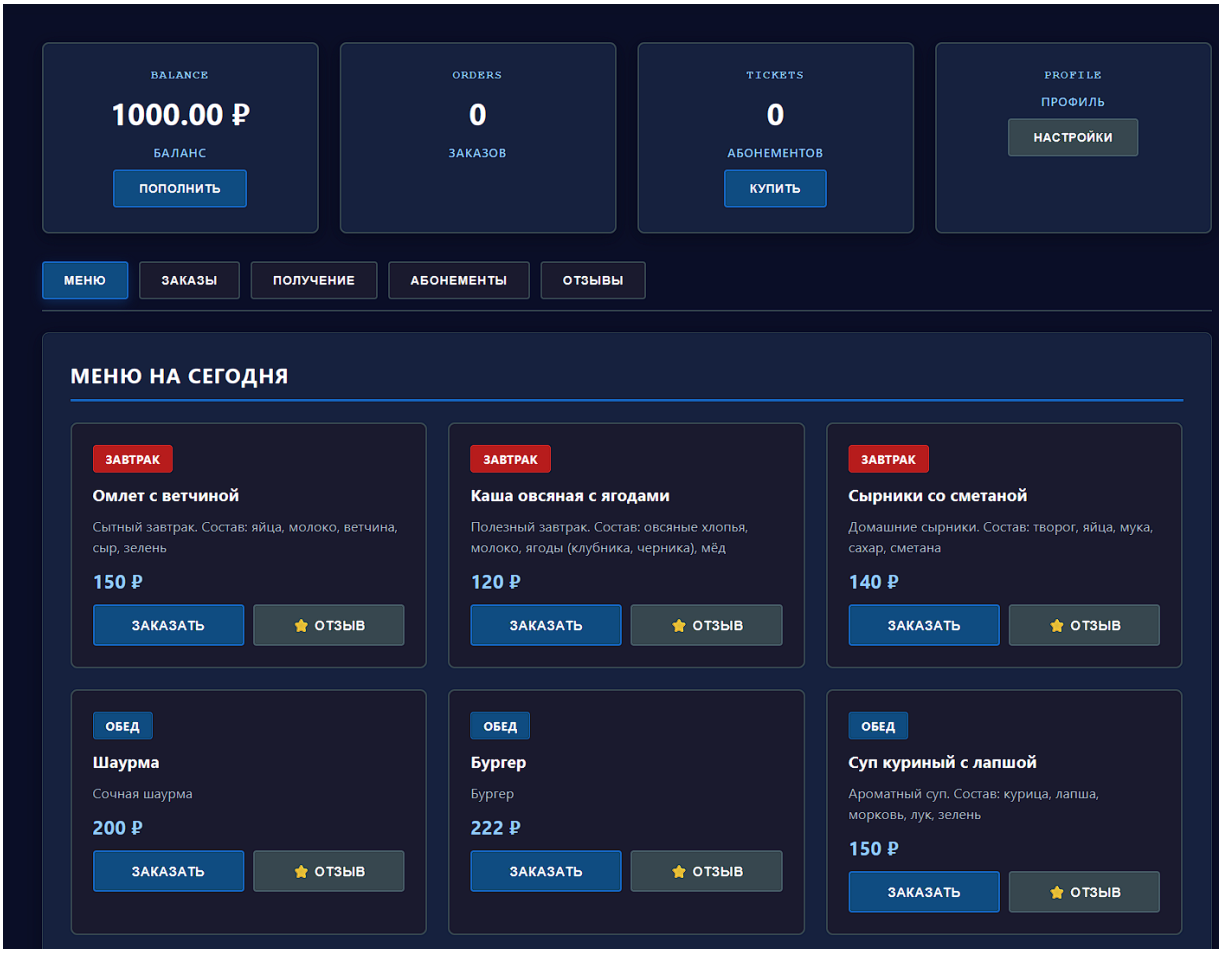

Рис. 12. Панель меню для роли «Ученик»

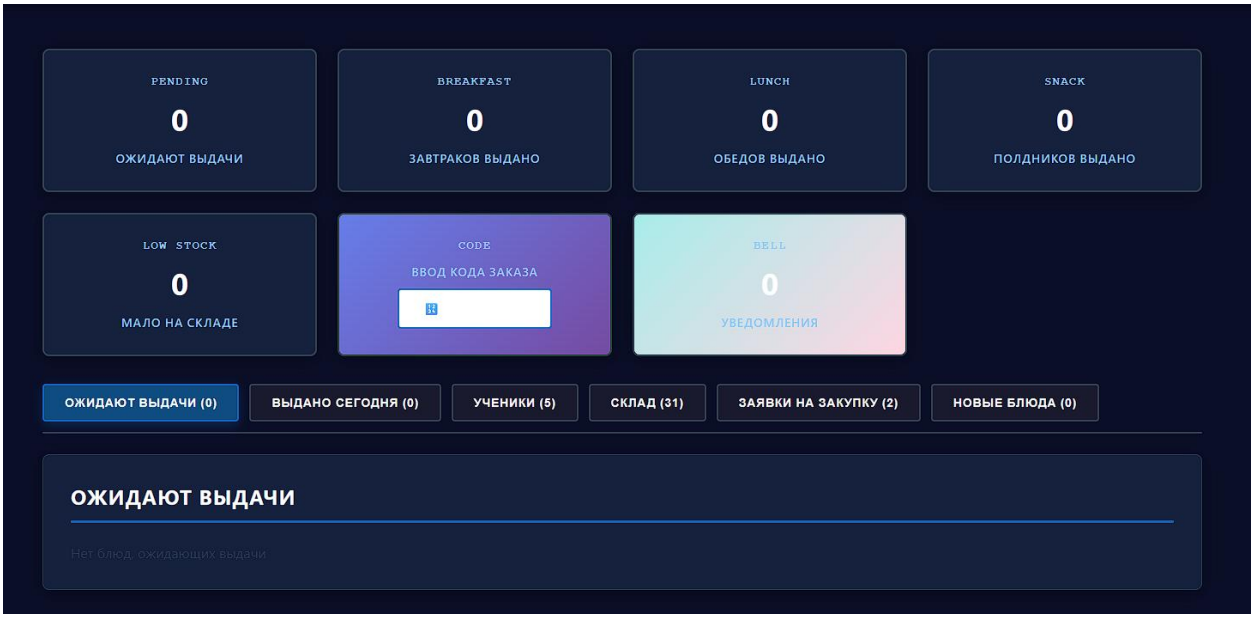

Рис. 13 Панель заказов для роли «Повар»

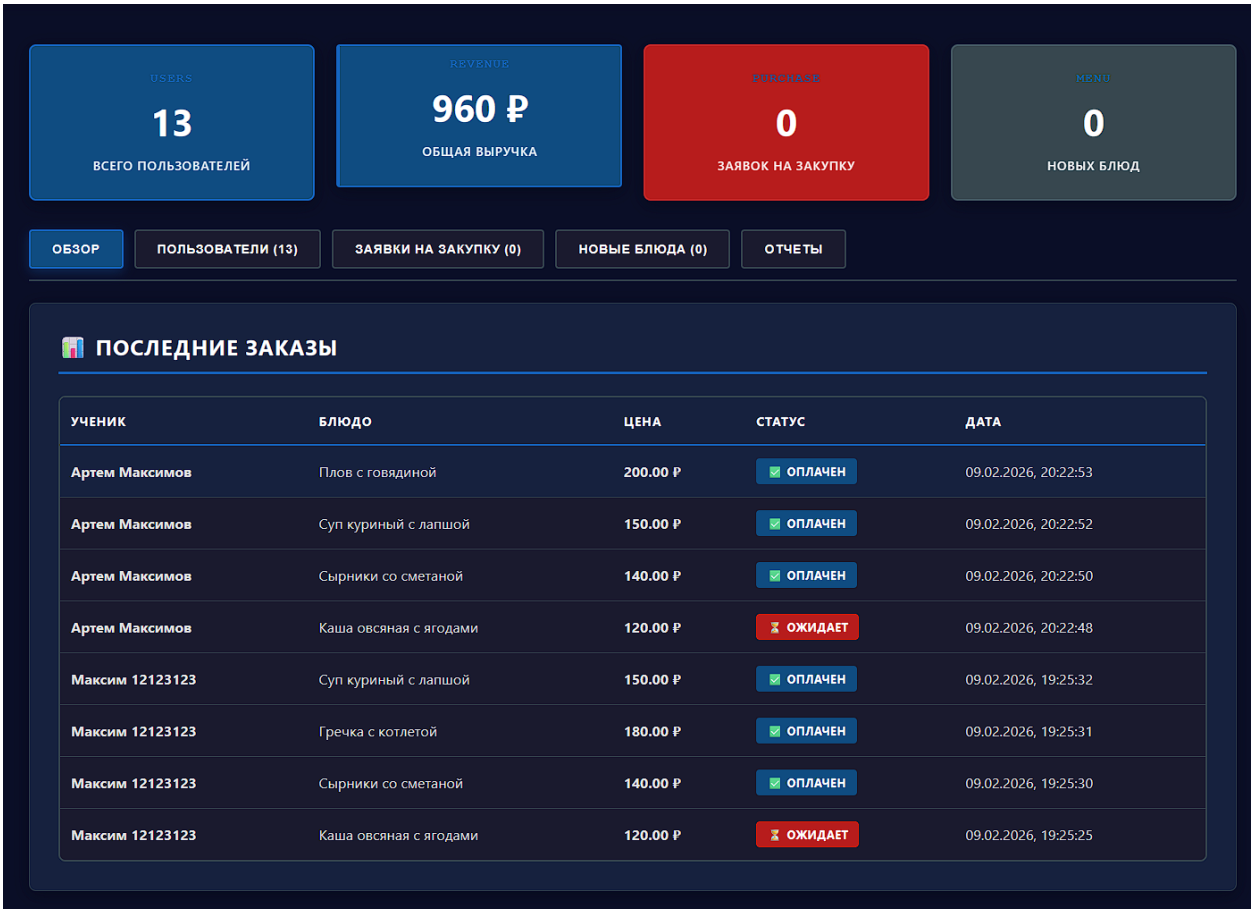

Рис. 14. Панель статистики заказов для роли «Администратор»

Готовая рабочая версия системы управления столовой развёрнута на сервере и доступна для тестирования:
https://autogreatfood.ru

Видеопрезентация с подробной демонстрацией всех возможностей системы размещена на платформе RuTube:
https://rutube.ru/video/private/82a9eb25f80867b93af7d89a84c6dbe2/?p=ChJ4hmeJdLje16WF5JdLDw

В видео показаны следующие сценарии использования:

1. Процесс регистрации пользователей разных ролей (ученик, повар, администратор)
2. Навигация по интерфейсу и использование основных функций
3. Создание заказов учениками, проверка баланса и оплата
4. Покупка абонементов на несколько дней
5. Работа поваров с системой выдачи (фильтрация заказов, поиск по номеру)
6. Административные функции: управление пользователями, редактирование меню, просмотр аналитики и статистики

ЗАКЛЮЧЕНИЕ

В результате работы над проектом мы создали полнофункциональную информационную систему управления школьным питанием. Система автоматизирует весь процесс — от формирования заказа до выдачи еды и формирования отчётов для администрации.

Система реализована на современном технологическом стеке: React для клиентской части, Node.js с Express для серверной части и PostgreSQL для хранения данных. Такой выбор технологий обеспечивает высокую производительность, безопасность и возможность развития системы в будущем.

Все поставленные задачи выполнены:

1. Проведён анализ предметной области и обоснован выбор технологий
2. Разработаны структурная и функциональная схемы системы, блок-схема основного алгоритма
3. Спроектирована нормализованная схема базы данных
4. Реализован программный код с модульной структурой и чётким разделением ответственности
5. Использованы современные подходы: функциональные компоненты React, middleware архитектура на сервере, транзакции для критических операций
6. Реализованы механизмы безопасности: хеширование паролей, защищённые сессии, защита от SQL-инъекций
7. Проведено тестирование, система развёрнута и доступна для использования

Разработанная система показывает, как современные веб-технологии могут эффективно решать реальные задачи в образовательных учреждениях. Открытый исходный код позволяет другим школам адаптировать систему под свои нужды, а также служит учебным материалом по современной веб-разработке и проектированию баз данных.