

Министерство просвещения Российской Федерации
ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ
ГБОУ ШКОЛА №2009

Предпрофессиональная олимпиада школьников
КЕЙС 2
СИСТЕМА УПРАВЛЕНИЯ СТОЛОВОЙ

Авторы проекта,
учащиеся 11 класса,
Тюрин Платон Сергеевич
Губанов Святослав Алексеевич

Москва 2026

РЕФЕРАТ

Пояснительная записка **27 с., 4 рис., 2 табл., 6 ист., 4 приложения.**

АВТОМАТИЗАЦИЯ, ВЕБ-РАЗРАБОТКА, FULLSTACK, ШКОЛЬНОЕ ПИТАНИЕ, БАЗЫ ДАННЫХ, ИНТЕРФЕЙС, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, OPTIMIZATION, TYPESCRIPT, NEXT.JS, POSTGRESQL.

Цель работы — Разработка и внедрение автоматизированной информационной системы управления школьной столовой для повышения прозрачности процессов учета питания и оптимизации взаимодействия между учащимися и сотрудниками пищеблока.

В процессе работы был проведен анализ предметной области организации школьного питания, выявлены недостатки существующих систем учета. Спроектирована архитектура Fullstack-приложения, разработана схема реляционной базы данных, реализованы клиентская и серверная части программного продукта.

В результате работы было создано кроссплатформенное веб-приложение, обеспечивающее автоматизацию заказов, учет продуктов и формирование отчетности в соответствии с техническим заданием.

Предполагаемые направления использования устройства: внедрение в общеобразовательные учреждения (школы, лицеи, колледжи) для модернизации системы безналичного расчета и складского учета в столовых.

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете о НИР применяют следующие термины с соответствующими определениями:

Backend (Бэкенд) — программно-аппаратная часть сервиса, отвечающая за функционирование его внутренней логики, обработку данных, взаимодействие с базой данных и отправку ответов клиентской части приложения.

Express.js — минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для разработки мобильных и веб-приложений, используемый в проекте для организации API.

Frontend (Фронтенд) — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса, с которой непосредственно взаимодействует пользователь через браузер.

Fullstack (Фулстек) — комплексная разработка, включающая в себя одновременное проектирование и реализацию как клиентской (внешней), так и серверной (внутренней) частей веб-приложения.

Git — распределенная система управления версиями, позволяющая отслеживать изменения в файлах и вести совместную разработку программного обеспечения.

JavaScript (JS) — мультипарадигменный язык программирования, поддерживающий объектно-ориентированный, императивный и функциональный стили, являющийся базовым для выбранного стека технологий.

Next.js — открытый фреймворк на JavaScript, созданный поверх React, который расширяет его функциональность, позволяя создавать веб-приложения с использованием рендеринга на стороне сервера (SSR) и генерации статических сайтов.

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JS из

узкоспециализированного языка, работающего в браузере, в язык общего назначения.

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД), основанная на языке SQL, характеризующаяся высокой надежностью и поддержкой сложных запросов.

Tailwind CSS — utility-first CSS-фреймворк для создания пользовательских интерфейсов, предоставляющий низкоуровневые служебные классы для быстрой стилизации компонентов.

TypeScript (TS) — язык программирования, представленный Microsoft, позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript за счет добавления строгой статической типизации.

Авторизация — предоставление определенному лицу или группе лиц прав на выполнение определенных действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий.

Веб-приложение — клиент-серверное приложение, в котором клиентом выступает браузер, а сервером — веб-сервер, причем логика приложения распределена между сервером и клиентом.

Интерфейс — совокупность средств и методов, при помощи которых пользователь взаимодействует с программой или устройством.

Репозиторий — место, где хранятся и поддерживаются какие-либо данные, в контексте разработки — хранилище исходного кода проекта с историей изменений.

Сервер — специализированный компьютер или программное обеспечение, выполняющее сервисные функции по запросу клиента, предоставляя доступ к определенным ресурсам или услугам.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете о НИР применяют следующие сокращения и обозначения:

API — Application Programming Interface (Программный интерфейс приложения)

CSS — Cascading Style Sheets (Каскадные таблицы стилей)

DOM — Document Object Model (Объектная модель документа)

HTML — HyperText Markup Language (Язык гипертекстовой разметки)

HTTP — HyperText Transfer Protocol (Протокол передачи гипертекста)

IT — Information Technology (Информационные технологии)

JSON — JavaScript Object Notation (Текстовый формат обмена данными)

MVC — Model-View-Controller (Модель-Представление-Контроллер)

PC (ПК) — Personal Computer (Персональный компьютер)

REST — Representational State Transfer (Передача состояния представления)

SQL — Structured Query Language (Язык структурированных запросов)

SSR — Server-Side Rendering (Рендеринг на стороне сервера)

URL — Uniform Resource Locator (Единый указатель ресурсов)

UI — User Interface (Пользовательский интерфейс)

UX — User Experience (Пользовательский опыт)

БД — База Данных

ГОСТ — Государственный стандарт

ОС — Операционная система

ПО — Программное обеспечение

СУБД — Система управления базами данных

ТЗ — Техническое задание

ВВЕДЕНИЕ.....	7
Проблема.....	7
Цель проекта.....	7
Задачи проекта.....	7
Актуальность.....	8
Ожидаемые результаты проекта.....	8
1 ПОИСКОВО-ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП.....	9
1.1 Анализ предметной области.....	9
1.2 Анализ аналогов.....	9
1.3 Выбор технологий.....	10
1.4 Описание выбранных средств разработки.....	10
1.5 Формулировка технического задания (ТЗ).....	11
2 КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКИЙ ЭТАП.....	13
2.1 Проектирование архитектуры системы.....	13
2.2 Проектирование базы данных.....	13
2.3 Моделирование бизнес-процессов (Use Case).....	14
2.4 Реализация серверной части.....	14
2.5 Реализация клиентской части.....	14
3 ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП.....	16
3.1 Тестирование.....	16
3.2 Результаты работы.....	16
4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ.....	17
4.1 Установка и запуск Backend.....	17
4.2 Установка и запуск Frontend.....	17
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	18
ПРИЛОЖЕНИЕ А.....	19
Диаграмма вариантов использования (UML Use Case).....	19
ПРИЛОЖЕНИЕ Б.....	20
Схема базы данных (ER-диаграмма).....	20
ПРИЛОЖЕНИЕ В.....	21
API эндпоинты.....	21
ПРИЛОЖЕНИЕ Г.....	22
Скриншоты интерфейса.....	22
ПРИЛОЖЕНИЕ Г.....	26
Ссылка на видео обзор сайта.....	26

ВВЕДЕНИЕ

Проблема

В современных школах организация питания учащихся часто сопровождается рядом проблем:

- Длинные очереди в столовой из-за ручного расчёта наличными деньгами.
- Отсутствие инструмента контроля над финансами учащихся со стороны родителей.
- Невозможность заранее узнать меню и спланировать питание.
- Сложности с учётом аллергий и диетических ограничений учеников.
- Отсутствие механизма обратной связи о качестве блюд для администрации.
- Высокая трудоемкость администрирования процессов закупок и финансовой отчетности.

Цель проекта

Цель проекта — создание автономной информационной системы для автоматизации процессов школьного питания, обеспечивающей электронный заказ, систему безналичной оплаты, учет предпочтений учащихся и формирование автоматической отчетности для администрации школы и столовой.

Задачи проекта

1. Провести детальный анализ предметной области организации школьного питания.
2. Изучить существующие аналоги систем автоматизации столовых.
3. Осуществить выбор оптимального стека технологий для разработки веб-приложения.
4. Разработать архитектуру системы и спроектировать базу данных.
5. Создать программный код серверной части (Backend).
6. Разработать пользовательский интерфейс для трёх ролей пользователей (Frontend).
7. Реализовать систему электронных талонов (QR-коды).
8. Внедрить модуль отзывов и рейтингов.
9. Провести функциональное тестирование системы.

10. Подготовить техническую документацию.

Актуальность

Внедрение цифровых сервисов в образовательный процесс является государственным приоритетом. Информационная система "Школьная столовая" решает насущные проблемы организации питания, повышая скорость обслуживания, прозрачность финансовых потоков и удовлетворённость учащихся качеством еды. Система также способствует формированию культуры здорового питания благодаря открытости меню и составов блюд.

Ожидаемые результаты проекта

В результате работы будет создано веб-приложение "Школьная столовая", удовлетворяющее следующим требованиям:

- Наличие личных кабинетов для Ученика, Повара и Администратора.
- Функционал электронного меню с фото и ценами.
- Возможность покупки блюд с баланса ученика.
- Система подтверждения получения заказа через электронный талон.
- Модуль отзывов и рейтингов блюд.
- Инструменты для планирования закупок продуктов.
- Автоматическая генерация отчетов по продажам и расходам.

1 ПОИСКОВО-ИССЛЕДОВАТЕЛЬСКИЙ ЭТАП

1.1 Анализ предметной области

Процесс школьного питания включает в себя взаимодействие нескольких сторон:

1. **Столовая (Повар)**: планирует меню, закупает продукты, готовит блюда.
2. **Потребители (Ученики)**: выбирают блюда, оплачивают, потребляют, оценивают.
3. **Управление (Администрация)**: контролирует финансы, качество, утверждает закупки.

Ключевым процессом является цикл: Планирование меню -> Закупка -> Приготовление -> Продажа -> Отзыв.

1.2 Анализ аналогов

Были проанализированы существующие на рынке решения.

Таблица 1 — Сравнительный анализ аналогов

Система	Достоинства	Недостатки
"Аксиома"	Полная интеграция с банковскими системами, терминалы оплаты.	Высокая стоимость внедрения и обслуживания оборудования. Закрытая экосистема.
"Ладошки" (Сбербанк)	Оплата по биометрии (ладони), удобный кабинет для родителей.	Требует установки дорогостоящего сканирующего оборудования. Проблемы с персональными данными.
"Глолайм"	Безналичные карты, турникеты, буфет.	Сложный интерфейс, зависимость от физических карт.

По результатам анализа принято решение разрабатывать собственную веб-систему, не требующую специального оборудования (работает на смартфонах/планшетах/ПК).

1.3 Выбор технологий

Для реализации проекта был выбран современный стек веб-технологий.

Таблица 2 — Выбор средств реализации

Компонент	Выбранная технология	Обоснование выбора
Язык Backend	TypeScript (Node.js)	Строгая типизация, высокая производительность V8, единый язык с фронтендом.
Фреймворк Backend	Express.js	Минималистичный, гибкий, огромная экосистема middleware.
База данных	SQLite	Легковесная, не требует сервера БД, идеально для прототипа и малых внедрений.
Язык Frontend	TypeScript (React)	Лидер индустрии, компонентный подход, виртуальный DOM.
Фреймворк Frontend	Next.js 14	Серверный рендеринг (SSR), встроенный роутинг, оптимизация изображений.
UI Библиотека	shadcn/ui + Tailwind	Быстрая вёрстка, современные доступные компоненты, легкая кастомизация.

1.4 Описание выбранных средств разработки

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. **Express** — веб-фреймворк для приложений Node.js, предоставляющий обширный

набор функций для мобильных и веб-приложений. **SQLite** — компактная встраиваемая СУБД. Представляет собой библиотеку, линкуемую с приложением. **React** — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. **Next.js** — фреймворк на поверх React, дающий функционал SSR (Server Side Rendering) и SSG (Static Site Generation).

1.5 Формулировка технического задания (ТЗ)

1. Общие требования Система должна представлять собой веб-приложение, доступное через интернет-браузер. Интерфейс должен быть адаптирован под мобильные устройства.

2. Требования к функционалу

Для роли "Ученик":

- Регистрация и вход по логину/паролю.
- Просмотр меню (фильтрация: завтрак, обед).
- Просмотр баланса и его пополнение.
- Добавление блюд в корзину и покупка.
- Просмотр активного заказа (QR-код/номер).
- Подтверждение получения заказа.
- Оценка блюда (рейтинг 1-5 звезд) и текстовый отзыв.
- Настройка профиля (указание аллергий).

Для роли "Повар":

- Просмотр и редактирование меню.
- Указание количества доступных порций.
- Создание заявок на закупку продуктов.
- Просмотр склада.

Для роли "Администратор":

- Просмотр статистики продаж (выручка, популярные блюда).
- Утверждение или отклонение заявок на закупку.
- Генерация отчётов (HTML/PDF).

3. Требования к дизайну Дизайн должен быть выполнен в светлой цветовой гамме, быть интуитивно понятным и минималистичным. Использовать современные принципы UX/UI.

2 КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКИЙ ЭТАП

2.1 Проектирование архитектуры системы

Архитектура приложения построена по принципу RESTful API.

Клиентская часть (Frontend) взаимодействует с сервером (Backend) посредством HTTP-запросов, обмениваясь данными в формате JSON.

Структурная схема взаимодействия:

[КЛИЕНТ (Браузер)]



HTTP Запросы (GET, POST, PATCH)



v

[СЕРВЕР (Node.js + Express)]



[Auth] [Student] [Admin] <-- Контроллеры API



v

[ORM / Query Builder]



v

[БАЗА ДАННЫХ (SQLite)]

2.2 Проектирование базы данных

База данных спроектирована в третьей нормальной форме для исключения избыточности.

Описание таблиц:

- users: Хранит учетные данные всех пользователей, их роли, баланс и информацию об аллергиях.
- menu: Содержит список всех блюд, их цены, описания и типы.
- inventory: Складской учет ингредиентов.
- orders: История покупок учеников. Связывает пользователя и блюдо.
- reviews: Хранит оценки и отзывы пользователей о блюдах.

- procurement_requests: Заявки повара на пополнение склада.

(Подробная схема приведена в Приложении Б)

2.3 Моделирование бизнес-процессов (Use Case)

Сценарий "Покупка обеда":

1. Ученик авторизуется в системе.
2. Система проверяет баланс.
3. Ученик выбирает блюдо в меню и нажимает "Купить".
4. Система проверяет доступное количество порций.
5. Если порции есть и баланс достаточен:
 - списываются средства с баланса.
 - Уменьшается счетчик порций.
 - Создается запись в таблице orders.
 - Пользователю выдается "Электронный талон".
6. Ученик показывает талон повару.
7. Ученик нажимает "Подтвердить получение".
8. Заказ переходит в статус "completed".

2.4 Реализация серверной части

Сервер написан на TypeScript. В качестве менеджера пакетов используется npm. Для обработки запросов используется паттерн "Маршрут-Контроллер-Сервис".

Пример маршрута (student.ts):

```
router.get('/menu', async (req, res) => {
  const menu = await db.all('SELECT * FROM menu WHERE available_qty > 0');
  res.json(menu);
});
```

Реализована Middleware аутентификации, которая проверяет JWT-токен в заголовках запроса и защищает приватные маршруты.

2.5 Реализация клиентской части

Использована файловая маршрутизация Next.js (App Router).

- /app/page.tsx - Страница входа.
- /app/student/page.tsx - Дашборд студента (SSR + Client Components).
- /app/admin/page.tsx - Дашборд администратора.

Для состояния используется React Hooks (useState, useEffect).

Взаимодействие с API вынесено в отдельный слой services/api.ts. Дизайн реализован с помощью утилитарных классов Tailwind CSS, что позволяет быстро адаптировать верстку под разные экраны.

3 ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП

3.1 Тестирование

Было проведено ручное тестирование системы по следующим кейсам:

1. **Тест регистрации:** Создание пользователя с существующим именем -> Ошибка (Успешно).
2. **Тест покупки:** Покупка при нулевом балансе -> Ошибка "Недостаточно средств" (Успешно).
3. **Тест покупки:** Покупка доступного блюда -> Списание средств, появление талона (Успешно).
4. **Тест отзывов:** Отправка отзыва -> Появление его в списке отзывов у блюда (Успешно).
5. **Тест админки:** Генерация отчета -> Корректный подсчет итоговой суммы (Успешно).

3.2 Результаты работы

Все поставленные цели и задачи выполнены в полном объеме. Разработана и протестирована информационная система "Школьная столовая".

Достигнутые показатели:

- Время обработки заказа сократилось (в теории) до 10-15 секунд (показ QR-кода).
- Реализована полная прозрачность меню и отзывов.
- Создан удобный инструмент администрирования.

Система готова к опытной эксплуатации и демонстрации.

4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ

Для запуска проекта локально вам понадобятся установленные **Node.js** (версии 18+) и **npm**.

4.1 Установка и запуск Backend

1. Откройте терминал и перейдите в папку backend:

```
cd backend
```

2. Установите зависимости:

```
npm install
```

3. Запустите сервер:

```
npm run dev
```

Сервер запустится по адресу: <http://localhost:5000>

4.2 Установка и запуск Frontend

1. Откройте новый терминал и перейдите в папку **frontend**:

```
cd frontend
```

2. Установите зависимости:

```
npm install
```

3. Запустите клиентское приложение:

```
npm run dev
```

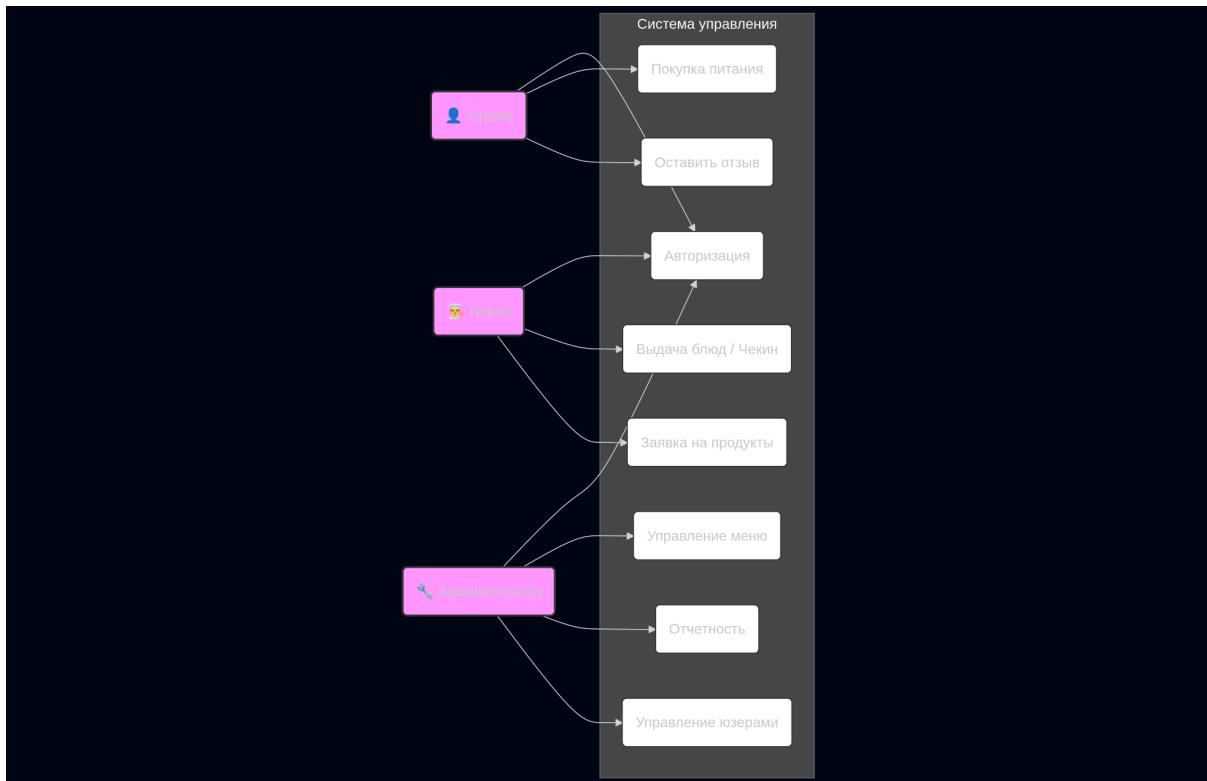
Приложение будет доступно по адресу: <http://localhost:3000>

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация Node.js [Электронный ресурс]. URL: <https://nodejs.org/docs/> (дата обращения: 02.02.2026).
2. Руководство по Express.js [Электронный ресурс]. URL: <https://expressjs.com/> (дата обращения: 02.02.2026).
3. Документация Next.js [Электронный ресурс]. URL: <https://nextjs.org/docs> (дата обращения: 02.02.2026).
4. React Documentation [Электронный ресурс]. URL: <https://react.dev/> (дата обращения: 02.02.2026).
5. Официальный сайт SQLite [Электронный ресурс]. URL: <https://www.sqlite.org/docs.html> (дата обращения: 02.02.2026).
6. Введение в JWT [Электронный ресурс]. URL: <https://jwt.io/introduction> (дата обращения: 02.02.2026).

ПРИЛОЖЕНИЕ А

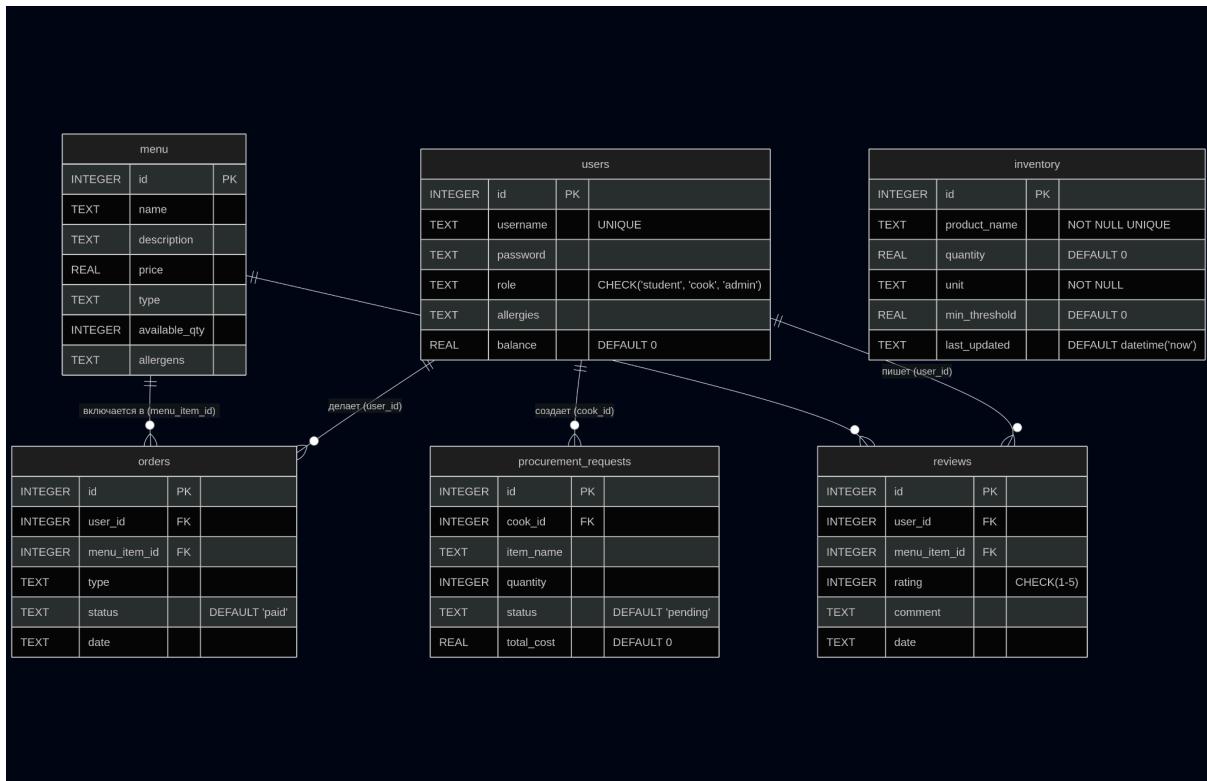
Диаграмма вариантов использования (UML Use Case)



Диаг. 1

ПРИЛОЖЕНИЕ Б

Схема базы данных (ER-диаграмма)



Диаг. 2

ПРИЛОЖЕНИЕ В

API эндпоинты

1. Auth

- POST /auth/register - Регистрация
- POST /auth/login - Логин

2. Student

- GET /student/menu - Меню
- POST /student/buy - Покупка
- POST /student/reviews - Отзыв

3. Cook

- POST /cook/menu - Добавить блюдо
- POST /cook/request - Запрос товара

4. Admin

- GET /admin/stats - Статистика
- PUT /admin/requests/:id/approve - Одобрить запрос

ПРИЛОЖЕНИЕ Г

Скриншоты интерфейса

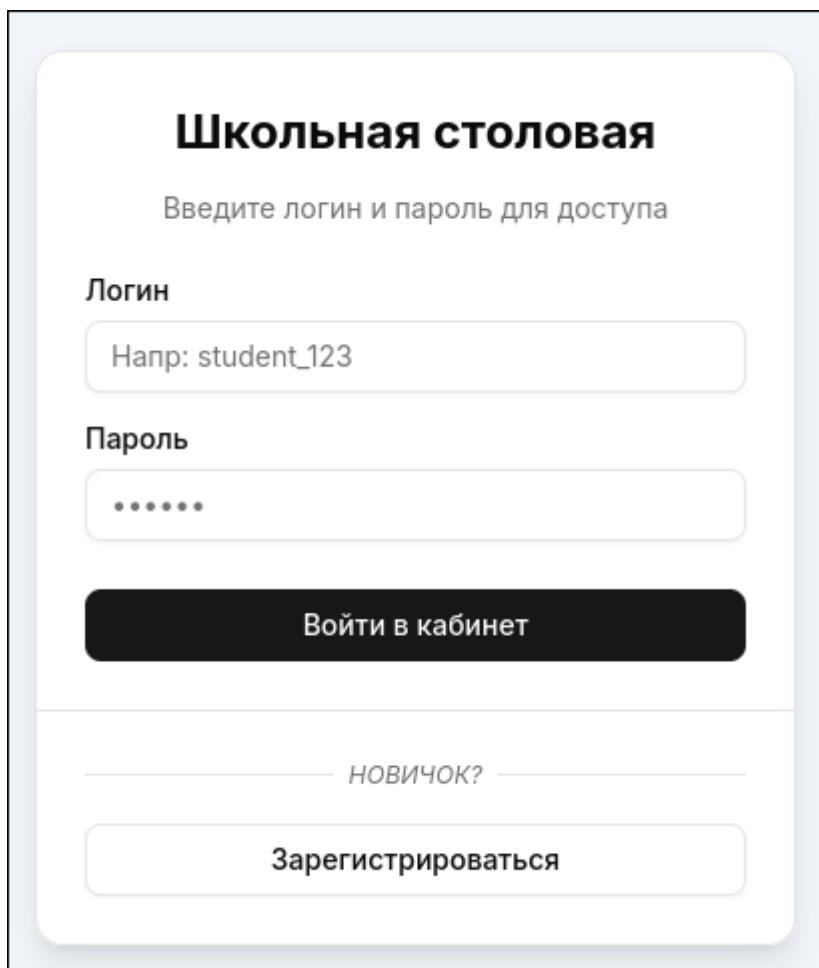


Рис. 1

Личный кабинет

Школьное питание

БАЛАНС
0 ₽

МОИ АЛЛЕРГИИ
Нет аллергий

Добавить (напр. Орехи)

Выход

Завтраки Обеды

Борщ с мясом 120 ₽
Традиционный борщ со свининой и сметаной

Оценить
☆☆☆☆☆
Комментарий...
Отправить

Котлета с пюре 150 ₽
Куриная котлета с картофельным пюре и подливой

Оценить
☆☆☆☆☆
Комментарий...
Отправить

Рыба с рисом 140 ₽
Филе минтая запечённое с рисом и овощами

Оценить
☆☆☆☆☆
Комментарий...
Отправить

Макароны с сыром 100 ₽
Макароны с сырным соусом и зеленью

Оценить
☆☆☆☆☆
Комментарий...
Отправить

Куриный суп с лапшой 110 ₽
Лёгкий суп с курицей и домашней лапшой

Оценить
☆☆☆☆☆
Комментарий...
Отправить

Суп с макаронами 1234 ₽
Без описания

Оценить
☆☆☆☆☆
Комментарий...
Отправить

Купить Абонемент

Посмотреть отзывы

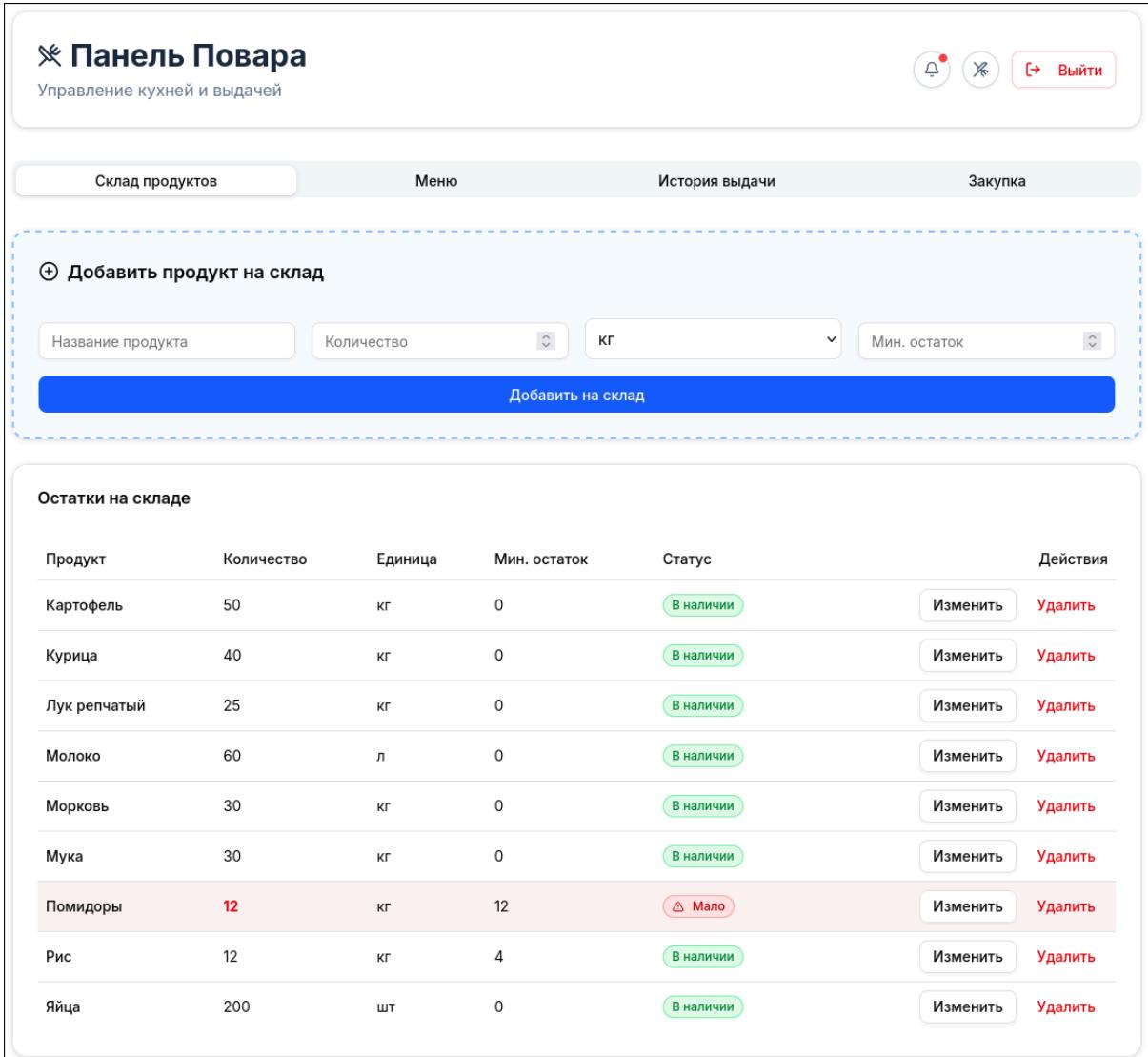
Купить Абонемент

Посмотреть отзывы

Купить Абонемент

Посмотреть отзывы

Рис. 2



Панель Повара

Управление кухней и выдачей

Склад продуктов Меню История выдачи Закупка

+ Добавить продукт на склад

Название продукта Количество Единица Мин. остаток Добавить на склад

Остатки на складе

Продукт	Количество	Единица	Мин. остаток	Статус	Действия
Картофель	50	кг	0	В наличии	Изменить Удалить
Курица	40	кг	0	В наличии	Изменить Удалить
Лук репчатый	25	кг	0	В наличии	Изменить Удалить
Молоко	60	л	0	В наличии	Изменить Удалить
Морковь	30	кг	0	В наличии	Изменить Удалить
Мука	30	кг	0	В наличии	Изменить Удалить
Помидоры	12	кг	12	⚠ Мало	Изменить Удалить
Рис	12	кг	4	В наличии	Изменить Удалить
Яйца	200	шт	0	В наличии	Изменить Удалить

Рис. 3

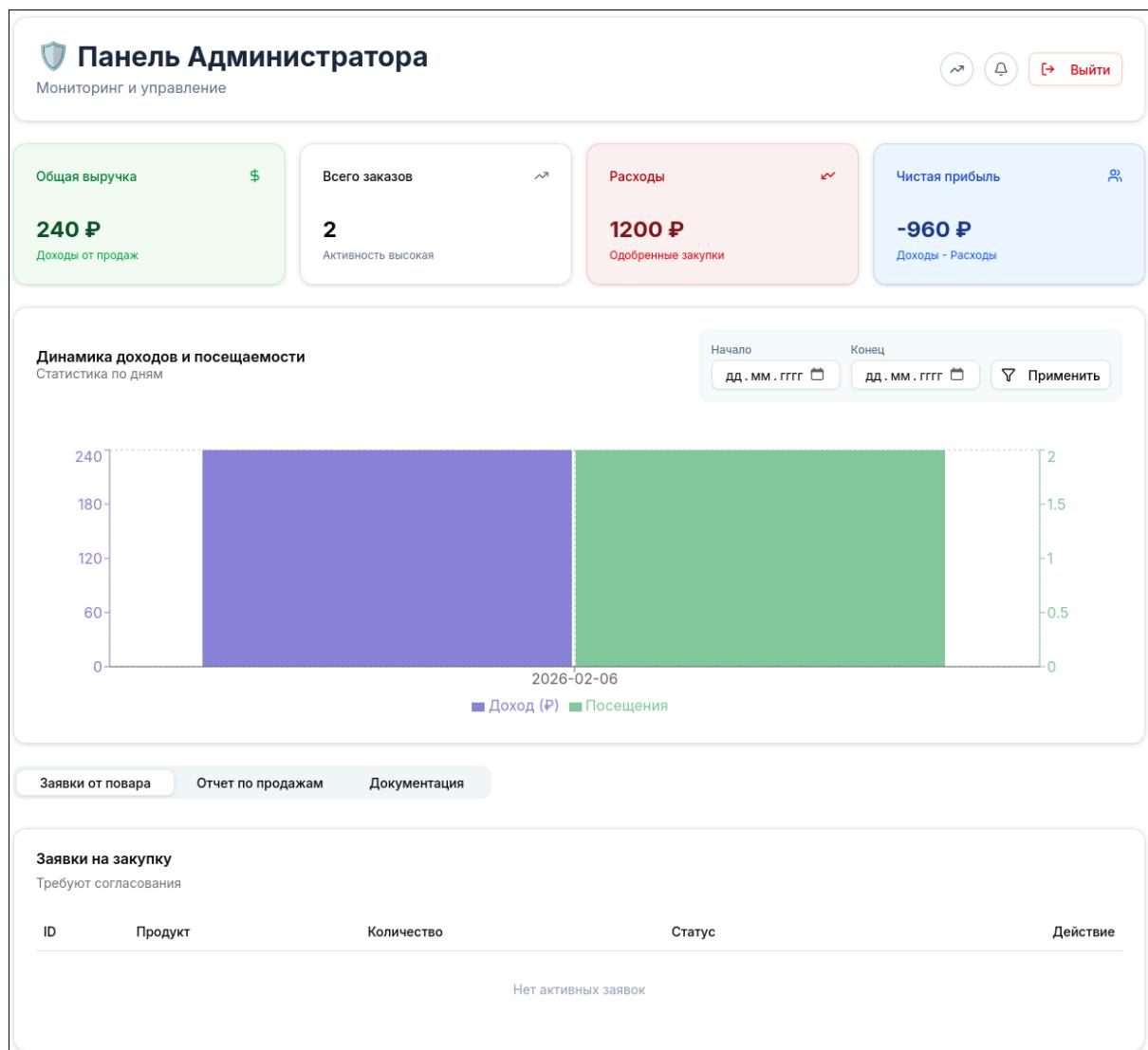


Рис. 4

ПРИЛОЖЕНИЕ Г

Ссылка на видео обзор сайта

<https://rutube.ru/video/5421bc8cac902243ec22d34d85c75852/?r=a/>