МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных »

Тема: ИС для кафе быстрого питания

Студентка гр. 9304 _____ Шуняев А.В.

Студент гр. 9303 _____ Алексеенко Б.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

Студенты

Шуняев. А.В.
Группа 9304

Алексеенко Б.
Группа 9303

Тема работы: Разработка веб-приложения для управления внутренними процессами кафе быстрого питания.

Исходные данные:

Необходимо реализовать приложение для СУБД(MongoDB).

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к работе»

«Сценарии использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 01.09.2022

Дата защиты реферата: 22.12.2022

Студентка гр. 9304		Шуняев А.В.
Студент гр. 9303		Алексеенко Б.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

Разработана информационная система для управления процессами в кафе быстрого питания. Были использованы следующие технологии:

1. Фреймворк ASP.NET Core 6 для построения серверной части;
2. СУБД MongoDB;
3. Vite для создания окружения frontend части приложения;
4. Фреймворк React для построения клиентской части;
5. Язык программирования C#;
6. Язык программирования TypeScript.

Разработанная информационная система в полной мере отражает состояние базы данных, позволяет его мутировать и производить экспорт в машинно-читаемые форматы.

SUMMARY

An information system for managing processes in a fast food cafe has been developed. The following technologies were used:

1. ASP.NET Core 6 framework for building the server side;
2. DBMS MongoDB;
3. Vite to create an environment for the frontend part of the application;
4. React framework for building the client side;
5. C# programming language;
6. TypeScript programming language.

The developed information system fully reflects the state of the database, allowing it to be mutated and exported to machine-readable formats.

СОДЕРЖАНИЕ

Введение

Целью работы является создание информационной системы, которая позволит управлять процессами работы кафе быстрого питания.

1. Качественные требования к решению

Необходимо провести разработку web-приложения, используя СУБД MongoDB.

2. Сценарии использования

Схема макетов пользовательского интерфейса представлена на рис. 1 – n.

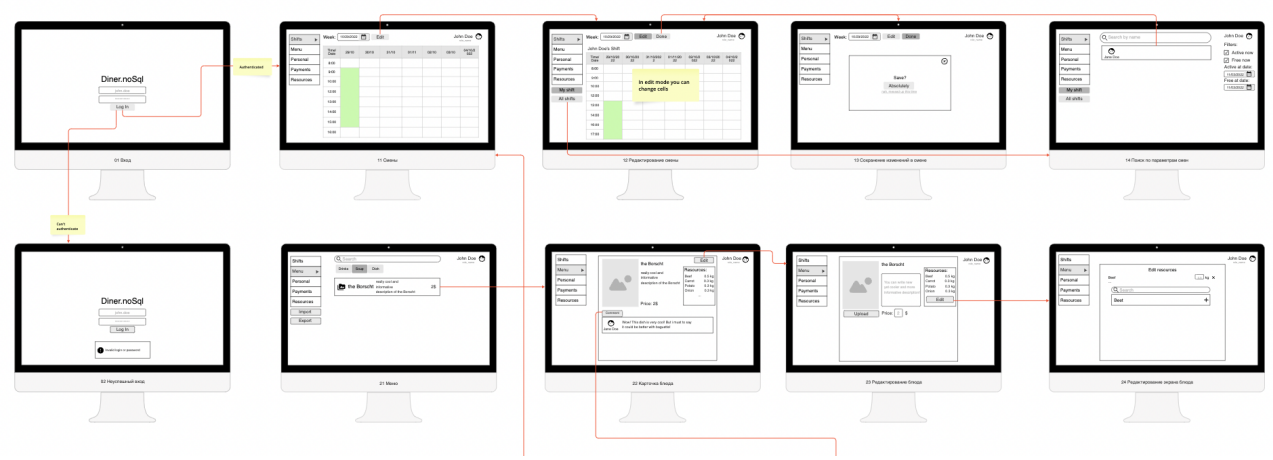

Рисунок 1 – Пользовательский интерфейс

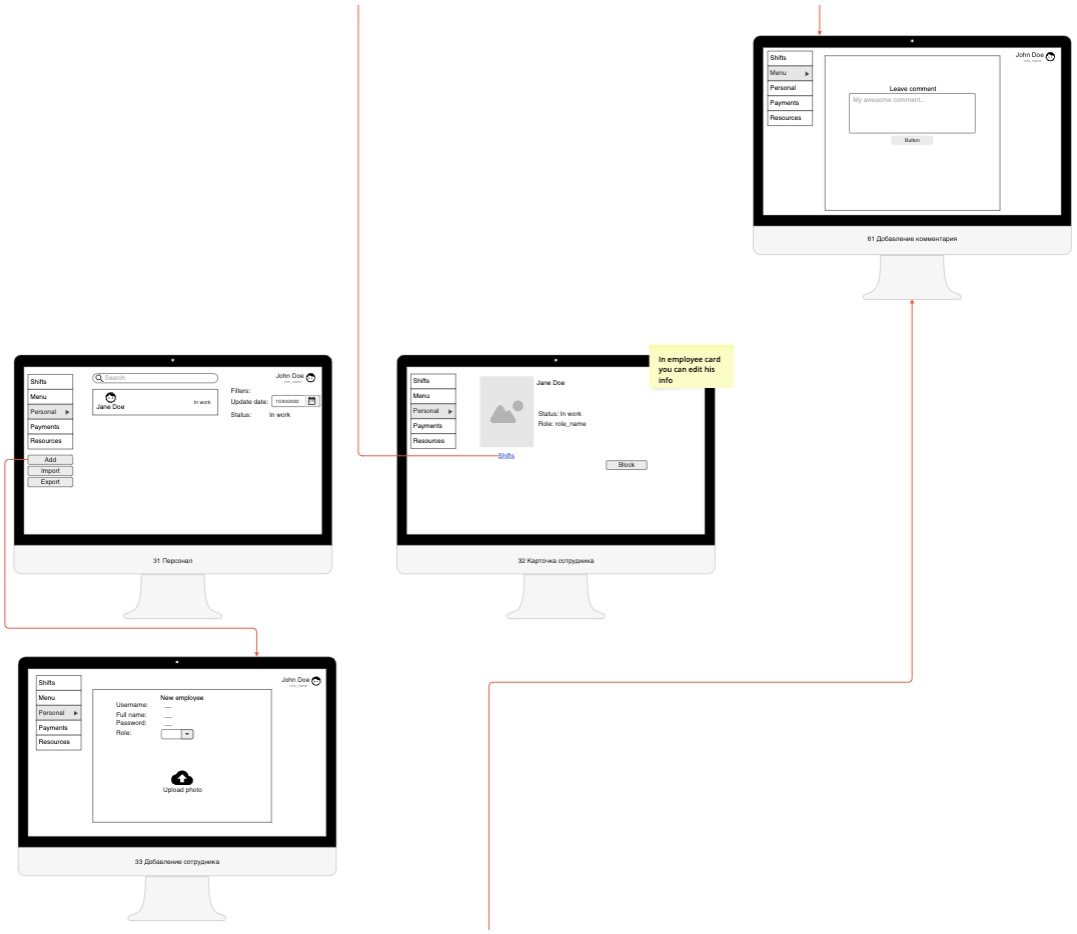

Рисунок 2 – Пользовательский интерфейс

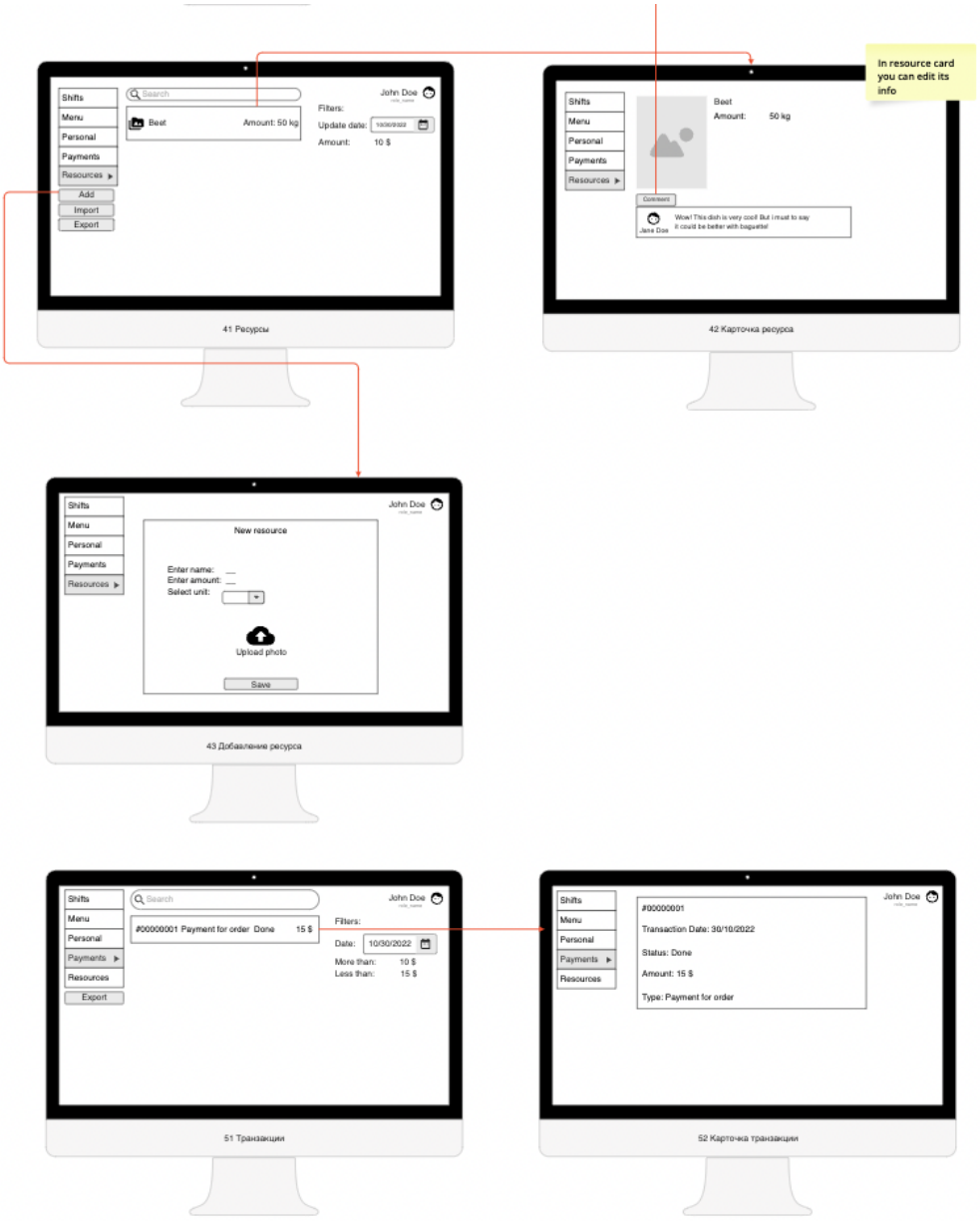

Рисунок 3 – Пользовательский интерфейс

Сценарии использования системы представлены в табл.1.

Таблица 1 – Сценарии использования

Название сценария	Сценарий	Действующее лицо
0.1/ Успешный вход в систему	1/ Пользователь получает логин и пароль для входа от администратора; 2/ Пользователь	Администратор; Официант;

	открывает страницу для входа в систему; 3/ Пользователь вводить логин и пароль в соответствующие поля; 4/ Пользователь нажимает на кнопку для входа в систему; 5/ Происходит аутентификация; 6/ Вход в систему выполнен	Завхоз; Повар; Менеджер;
0.2/ Неуспешный вход в систему	4/ Пользователь нажимает на кнопку для входа в систему; 5/ Происходит аутентификация; 6/ Вход в систему не выполнен; 7/ Пользователю выводится тост о неуспешной аутентификации	Администратор; Официант; Завхоз; Повар; Менеджер;
1/ Просмотр меню	1/ Пользователь аутентифицирован в системе; 2/ Пользователь попадает в элемент системы Меню; 3/ Пользователь выбирает тип меню (Барное/Еда/Чайно-кофейное?); 4/ Пользователь видит список блюд; 5/ Пользователь может выбрать блюдо; 6/ Есть возможность экспортировать меню в excel-таблицу; 7/ Есть возможность импортировать меню из excel-таблицы; 8/ Есть возможность поиска по названию блюда	Администратор; Официант; Завхоз; Повар; Менеджер;
2/ Просмотр карточки блюда	1/ Пользователь аутентифицирован в системе; 2/ Пользователь попадает на карточку блюда; 3/ Пользователь просматривает информацию о блюде (дата добавления, дата редактирования, название, ресурсы необходимые для приготовления, список комментариев, цена); 4/ Пользователь имеет возможность редактировать блюдо; 5/ При редактировании, пользователь имеет возможность удалить блюдо; 6/ Пользователь имеет возможность оставить комментарий; 7/ Пользователь может перейти на карточку ресурса, необходимого при готовке;	Официант; Завхоз; Повар; Менеджер;
3/ Просмотр списка ресурсов	1. 1/ Пользователь аутентифицирован в системе; 2/ Пользователь попадает в элемент системы Ресурсы; 3/ Пользователь видит список всех	Официант; Завхоз; Повар; Менеджер;

	ресурсов, где отражены название и количество; 4/ Пользователь имеет возможность экспортировать Ресурсы в excel-таблицу; 5/ Пользователь имеет возможность импортировать Ресурсы из excel-таблицы (пример – обновление состояния после поставки); 6/ Пользователь имеет возможность поиска по названию ресурса	
4/ Просмотр карточки ресурса.	/ Пользователь аутентифицирован в системе; 2/ Пользователь попадает на карточку ресурса (ингредиента); 3/ Пользователь просматривает информацию о ресурсе (дата добавления, дата редактирования, название, количество на складе (метрика зависит от типа продукта));	Официант; Завхоз; Повар; Менеджер;
5/ Просмотр списка персонала	1/ Пользователь аутентифицирован в системе; 2/ Пользователь попадает в элемент системы Персонал; 3/ Пользователь просматривает список сотрудников; 4/ Пользователь имеет возможность перейти в карточку определенного сотрудника; 5/ Пользователь имеет возможность импортировать список сотрудников в excel-таблицу; 6/ Пользователь имеет возможность добавить нового сотрудника; 7/ Пользователь имеет возможность поиска по имени сотрудника	Официант; Завхоз; Повар; Менеджер;
6/ Просмотр карточки сотрудника	1/ Пользователь аутентифицирован в системе; 2/ Пользователь попадает на карточку сотрудника; 3/ Пользователь видит информацию о сотруднике (дата добавления, дата редактирования, статус, роль, ФИО, фото, описание); 4/ Пользователь имеет возможность редактировать карточку сотрудника; 5/ Пользователь имеет возможность перейти в расписание смен сотрудника;	Официант; Завхоз; Повар; Менеджер;
7/ Просмотр смены сотрудника	1/ Пользователь аутентифицирован в системе; 2/ Пользователь попадает на карточку расписания смен сотрудника; 3/ Пользователь	Администратор; Официант;

	видит календарь на неделю, на которой отображено расписание смен сотрудника; 4/ Пользователь имеет возможность редактировать расписание смен сотрудника;	Завхоз; Повар; Менеджер;
8/ Просмотр списка транзакций	1/ Пользователь аутентифицирован в системе; 2/ Пользователь попадает в элемент системы транзакций; 3/ Пользователь видит список всех транзакций; 4/ Пользователь может перейти на карточку определенной транзакции; 5/ Пользователь имеет возможность фильтровать транзакции; 6/ Пользователь имеет возможность экспортировать транзакции в excel-таблицу;	Администратор; Официант; Завхоз; Повар; Менеджер;

3. Модель данных

3.1. Нереляционная модель данных

Модель хранения данных представлена на рис.4.

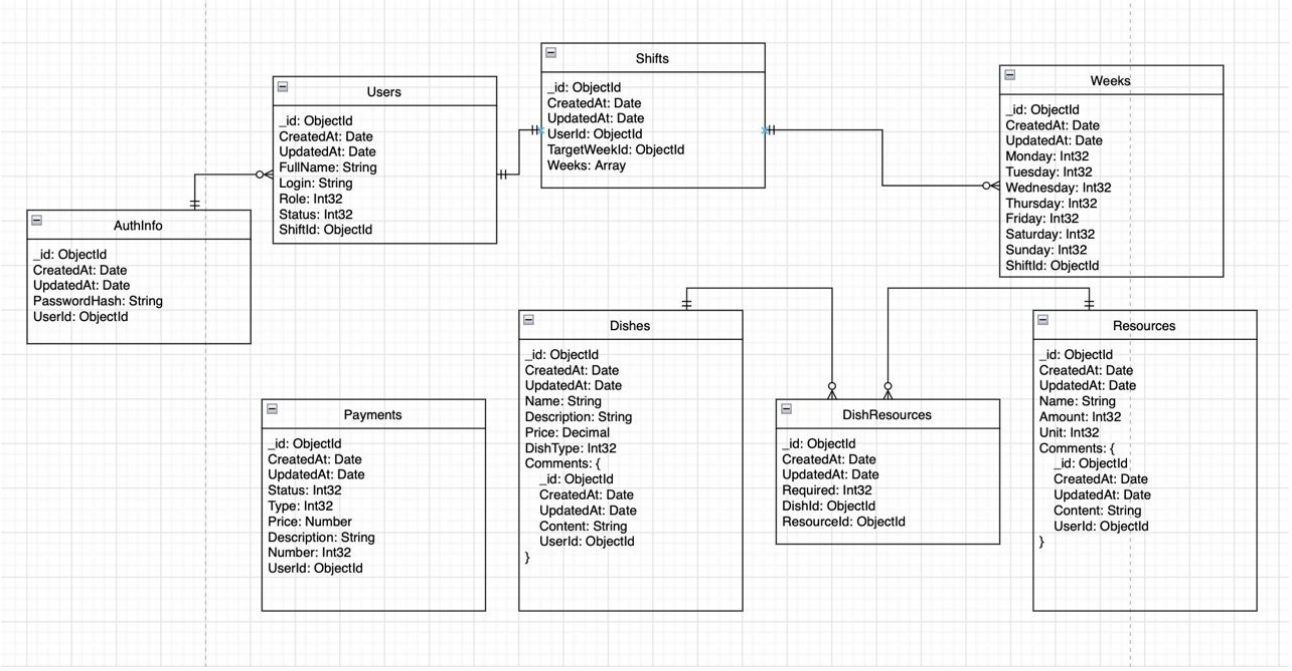

Рисунок 4 – Модель хранения данных

Оценка объема информации

Пользователь (119 байт)

```
User: {                     // Общий размер - 119 байт
    _id: ObjectId,          // ID - 12 байт
    CreatedAt: Date,        // Дата создания объекта - 8 байт
    UpdatedAt: Date,        // Дата обновления объекта - 8 байт
    FullName: String,       // ФИО пользователя (60 символов) - 60 байт
    Login: String,          // Логин пользователя (11) - 11 байт
    Role: Enum,             // Роль пользователя - 4 байта
    Status: Enum,           // Статус пользователя - 4 байта
    ShiftId: ObjectId       // ID смены пользователя - 12 байт
}
```

Тогда, чистый объем равен $N_{user} = 107$

Рисунок 5 - Описание коллекции “Users”

Данные авторизации (64 байта)

```
AuthInfo: {                 // Общий размер - 64 байта
    _id: ObjectId,          // ID - 12 байт
    CreatedAt: Date,        // Дата создания объекта - 8 байт
    UpdatedAt: Date,        // Дата обновления объекта - 8 байт
    PasswordHash: String,   // Хэш пароля - 44 байт
    UserId: ObjectId        // ID - 12 байт
}
```

Тогда, чистый объем равен $N_{authInfo} = 52$

Рисунок 6 - Описание коллекции “AuthInfos”

Смена (724 байта)

```
Shift: {                    // Общий размер - 724 байт
    _id: ObjectId,          // ID - 12 байт
    CreatedAt: Date,        // Дата создания объекта - 8 байт
    UpdatedAt: Date,        // Дата обновление объекта - 8 байт
    UserId: ObjectId,       // ID пользователя - 12 байт
    TargetWeekId: ObjectId, // ID текущей недели - 12 байт
    Weeks: Array<ObjectId>  // список ID прошедших недель - 672 байт (12 байт *
}
```

Тогда, чистый объем равен $N_{shift} = 712$

Рисунок 7 - Описание коллекции “Shifts”

Неделя (68 байт)

```
Week: {                 // Общий размер - 68 байт
    _id: ObjectId,      // ID - 12 байт
    CreatedAt: Date,    // Дата создания объекта - 8 байт
    UpdatedAt: Date,    // Дата обновления объекта - 8 байт
    Monday: Number,     // 32 битное число - 4 байта
    Tuesday: Number,    // 32 битное число - 4 байта
    Wednesday: Number,  // 32 битное число - 4 байта
    Thursday: Number,   // 32 битное число - 4 байта
    Friday: Number,     // 32 битное число - 4 байта
    Saturday: number,   // 32 битное число - 4 байта
    Sunday: Number,     // 32 битное число - 4 байта
    ShiftId: ObjectId   // ID - 12 байт
}
```

Тогда, чистый объем равен $N_{week} = 56$

Рисунок 8 - Описание коллекции “Weeks”

Блюдо (3273 байт)

```
Dish: {                 // Общий размер - 3273 байт
    _id: ObjectId,      // ID - 12 байт
    CreatedAt: Date,    // Дата создания объекта - 8 байт
    UpdatedAt: Date,    // Дата обновления объекта - 8 байт
    Name: String,       // Название блюда (32 символа) - 32 байт
    Description: String,// Описание (255 символов) - 255 байт
    Price: Number,      // Цена - 4 байта
    DishType: Enum,     // Тип блюда - 4 байта
    Comments: [{        // Общий объем - 295 байт (для 10 комментариев 2950)
        _id: ObjectId,      // ID - 12 байт
        CreatedAt: Date,    // Дата создания объекта - 8 байт
        UpdatedAt: Date,    // Дата обновления объекта - 8 байт
        Content: String,    // Текст комментария (255 символов) - 255 байт
        UserId: ObjectId,   // ID пользователя - 12 байт
    }]
}
```

Тогда, чистый объем равен $N_{dish} = 3141$

Рисунок 9 - Описание коллекции “Dishes”

Ингредиент (3018 байт)

```
Resource: {              // Общий размер - 3018 байт
    _id: ObjectId,       // ID - 12 байт
    CreatedAt: Date,     // Дата создания объекта - 8 байт
    UpdatedAt: Date,     // Дата обновления объекта - 8 байт
    Name: String,        // Максимальная длина названия - 32 байта
    Amount: Number,      // 32 битовое число - 4 байта
    Unit: Enum,          // 32 битовое число - 4 байта
    Comments: [{         // Общий объем - 295 байт (для 10 комментариев 2950)
        _id: ObjectId,         // ID - 12 байт
        CreatedAt: Date,       // Дата создания объекта - 8 байт
        UpdatedAt: Date,       // Дата обновления объекта - 8 байт
        Content: String,       // Текст комментария (255 символов) - 255 байт
        UserId: ObjectId,      // ID пользователя - 12 байт
    }]
}
```

Тогда, чистый объем равен $N_{resource} = 2886$

Рисунок 10 - Описание коллекции “Resources”

Связь Блюда и Ингредиента (56 байт)

```
DishResource: {          // Общий размер - 56 байт
    _id: ObjectId,       // ID - 12 байт
    CreatedAt: Date,     // Дата создания объекта - 8 байт
    UpdatedAt: Date,     // Дата обновления объекта - 8 байт
    Required: Number,    // Требуемое количество ингредиентов - 4 байта
    DishId: ObjectId,    // ID блюда - 12 байт
    ResourceId: ObjectId // ID ингредиента - 12 байт
}
```

Тогда, чистый объем равен $N_{dishResource} = 44$

Рисунок 11 - Описание коллекции “DishResources”

Оплата (311 байт)

```
Payment: {                  // Общий размер - 311
    _id: ObjectId,          // ID объекта - 12 байт
    CreatedAt: Date,        // Дата создания объекта - 8 байт
    UpdatedAt: Date,        // Дата обновления объекта - 8 байт
    Status: Enum,           // 32 битовое число - 4 байта
    Type: Enum,             // 32 битовое число - 4 байта
    Price: Number,          // 32 битовое число - 4 байта
    Description: String,    // Строка - 255 байт (максимальная длина описания - 255
    Number: Number,         // 32 битовое число - 4 байта
    UserId: ObjectId        // ID пользователя - 12 байт
}
```

Тогда, чистый объем равен $N_{payment} = 299$

Рисунок 12 - Описание коллекции “Payments”

Оценка удельного объема информации, хранимой в модели

Пусть количество блюд – N, тогда количество ресурсов N * 10 (10 максимально количество связей dishResource для одного блюда); Количество пользователей -- 10 (т.к. максимальное количество персонала для кафе – 10 человек); Количество смен – 10 * 56 = 560, что равно количеству недель, Количество документов информации о входе равно количеству пользователей -- 10. Пусть в системе совершенно 20 операций, тогда удельный объем равен = N * 20 + 437650

3.2. Реляционная модель данных

Модель хранения данных представлена на рис.13.

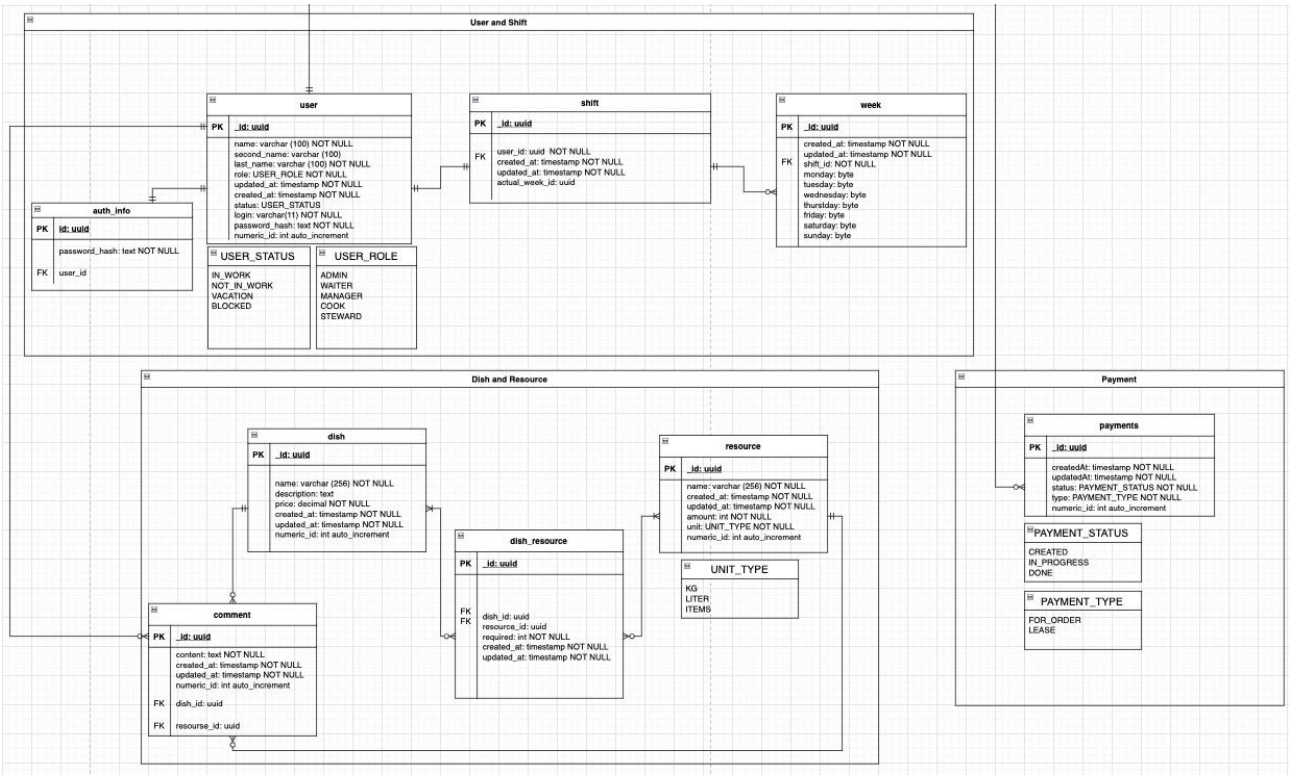

Рисунок 13 – Модель хранения данных

Расчет объема памяти для реляционной СУБД

При условии:

Пусть в системе будет 10 пользователей, для каждого из них расписано 56 недель расписания. В меню представлено 5 блюд и 10 ингредиентов. Каждое блюдо состоит из 5 ингредиентов. Для каждого блюда и ингредиента по 10 комментариев. Также совершено 20 операций оплаты.

Фактические объемы равны:

- Таблица Users - 1190 байт
- Таблица AuthInfos- 640 байт
- Таблица Shifts- 7240 байт
- Таблица Weeks- 38080 байт
- Таблица Dishes- 2215 байт
- Таблица Resources - 1880 байт

- Таблица DishResources - 1400 байт
- Таблица Comments - 3070 байт
- Таблица Payments- 6220 байт

Общий фактический объем равен 59720 байт.

Сравнение моделей

Размеры SQL и NoSQL моделей зависят де факто одинаково. И там, и там объем базы данных линейно зависит от количества сущностей. Реляционная БД занимает меньше места и медленнее растёт с увеличением числа классов. У реляционной базы данных избыточность выше. В среднем, реляционная и нереляционная базы данных задействуют одинаковое количество коллекция, для выполнения запросов.

В среднем, по количеству запросов, реляционная и нереляционные модели данных дают +/- одинаковые результаты, однако реляционная база данных представляет возможность реализовывать ту часть логики запросов своим инструментарием, в то время как для нереляционной модели эту часть приходиться реализовывать на стороне сервера (яркий пример - запросы для авторизации пользователя).

Сделать вывод о том, какая модель хранения данных будет однозначно лучше - тяжело. По полученным результатам можно немного прослеживается, что документо-ориентированные базы данных занимают больше места, но в них быстрее работает поиск, что является обычным поведением. Исходя из рабочего опыта авторов, можно предположить, что для хранения данных этой предметной области лучше подойдет реляционная модель хранения, т.к. она предлагает более универсальный подход, более чистую архитектуру и хранение по нереляционной модели не даст каких-то явных преимуществ.

4. Разработанное приложение

Было реализовано приложение, отвечающее требованиям. Экраны приложения представлены ны рис.6 - рис.12.

Рисунок 14 – Главный экран

Рисунок 15 – Экран со списком оплат

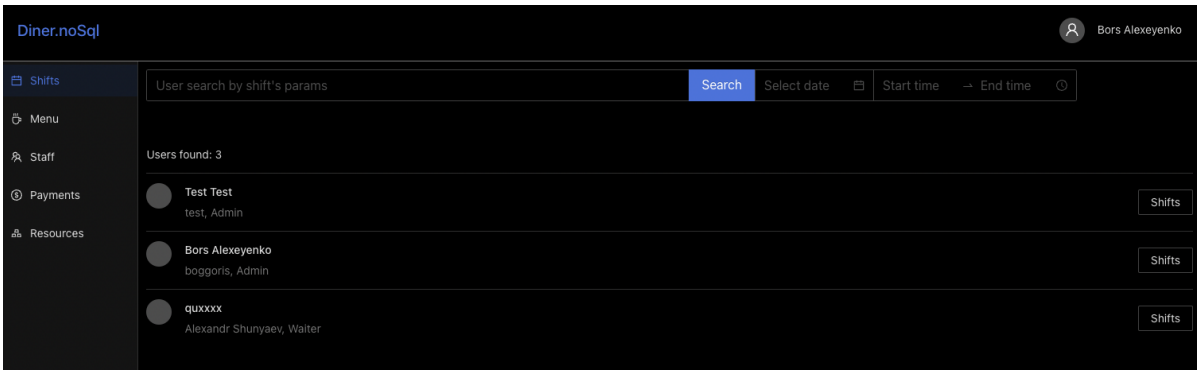

Рисунок 16 – Экран со списком сотрудников с поиском по параметрам смен

Рисунок 17 – Экран со сменой сотрудника

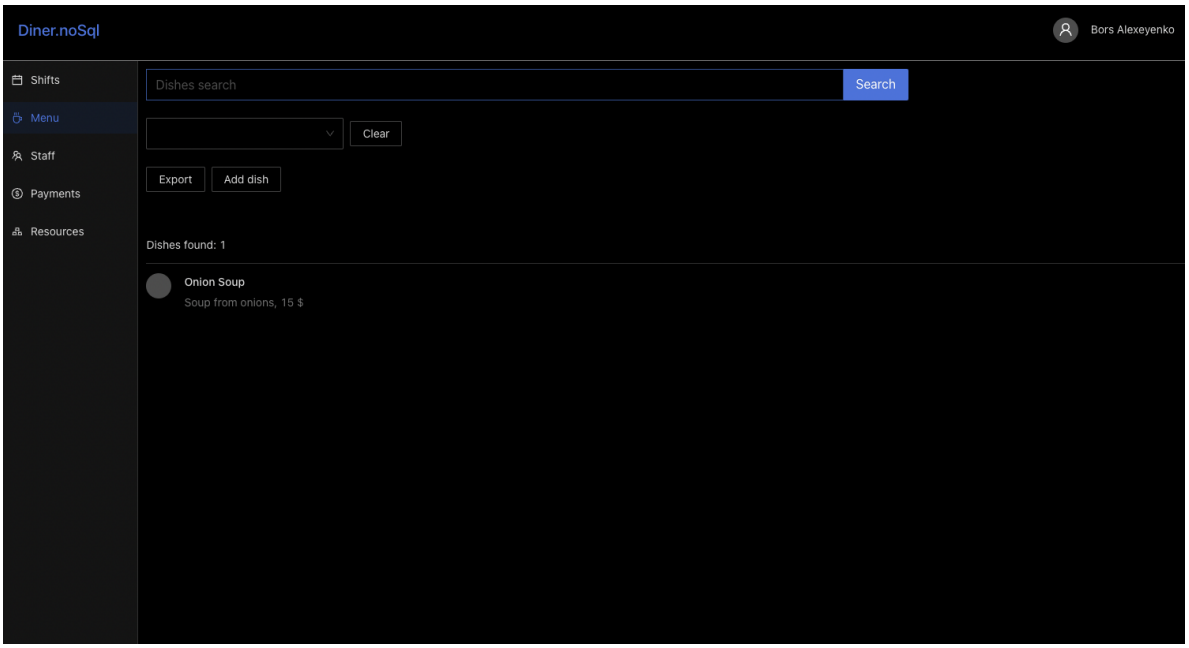

Рисунок 18 – Экран со списком блюд

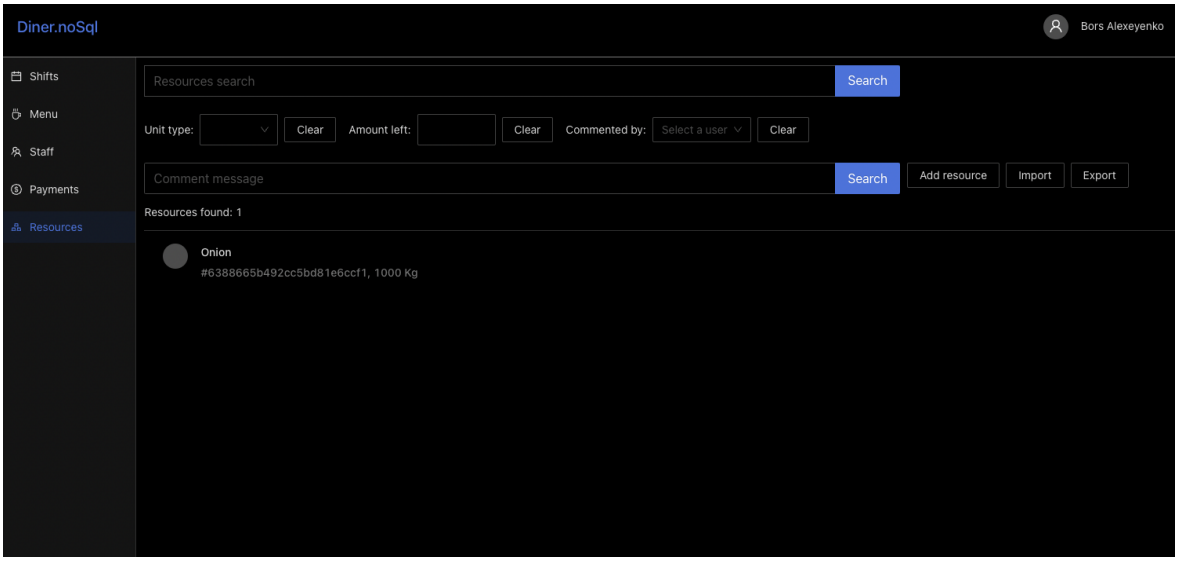

Рисунок 19 – Экран со списком ресурсов

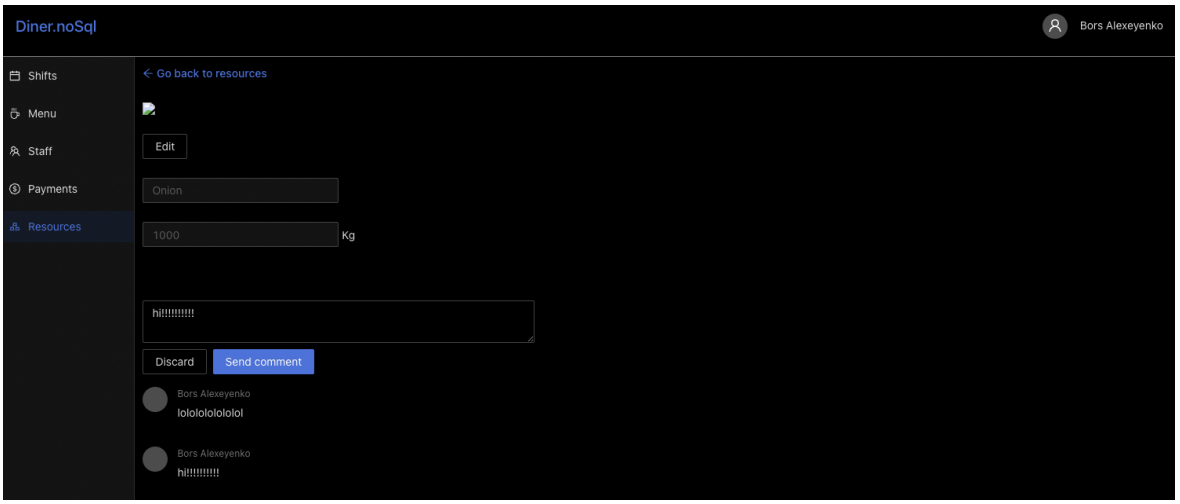

Рисунок 20 – Экран карточки ресурса с комментариями

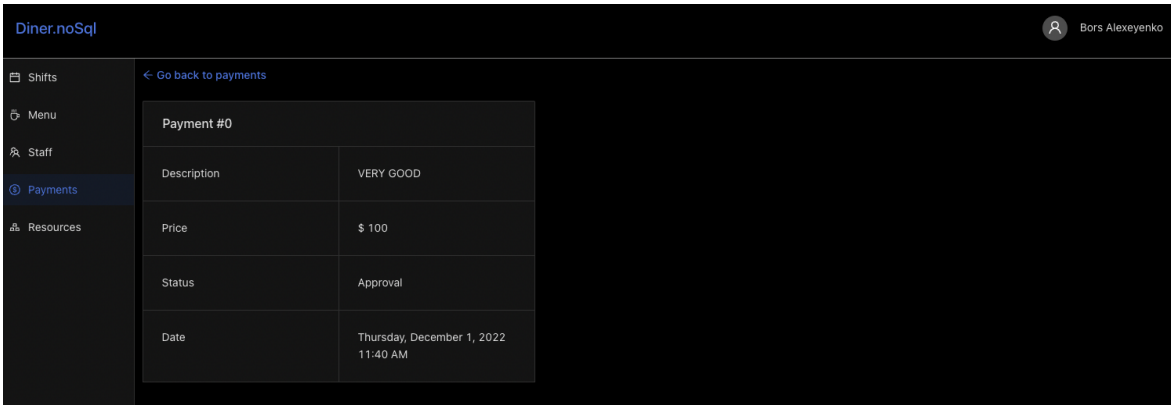

Рисунок 21 – Экран с карточкой оплаты

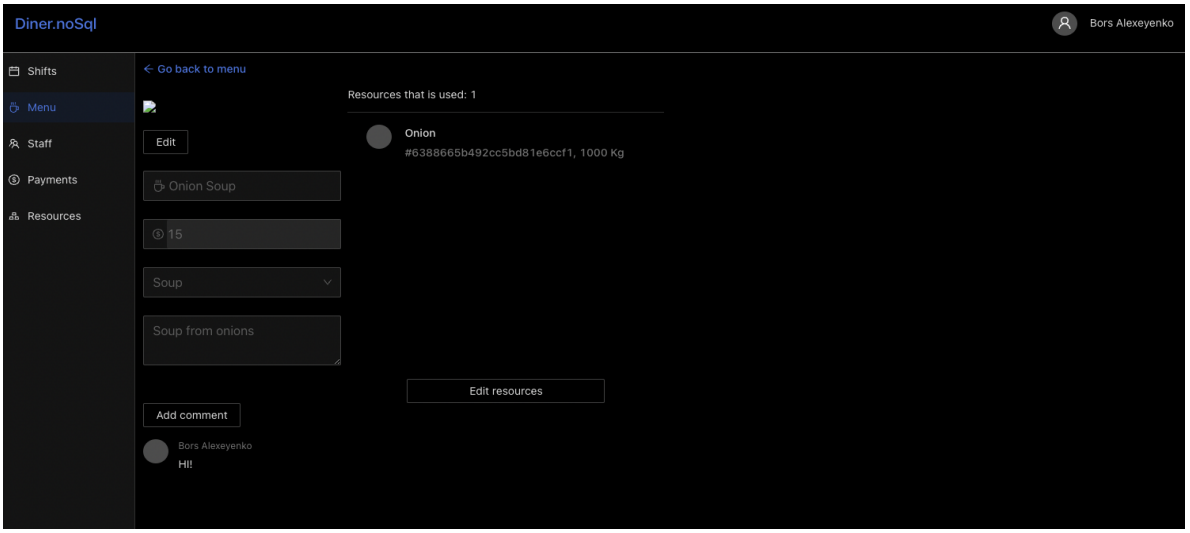

Рисунок 22 – Экран с карточкой блюда

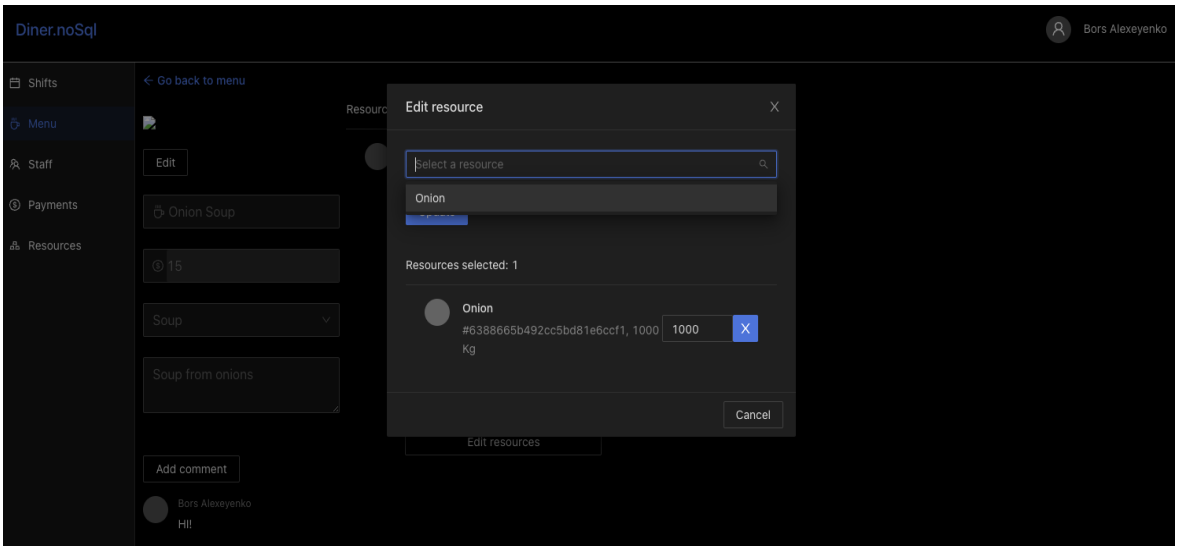

Рисунок 23 – Экран с редактированием ресурсов блюда

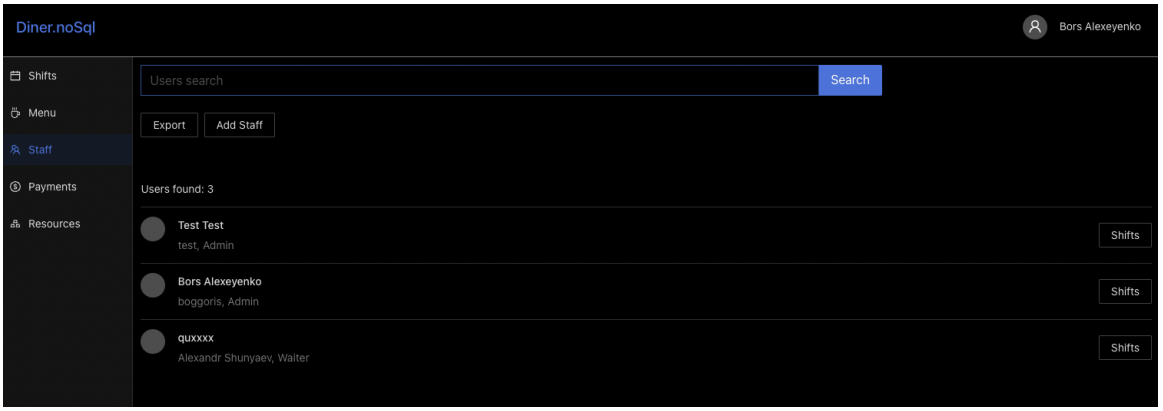

Рисунок 24 – Экран со списком сотрудников и поискам по параметрам сотрудника.

Рисунок 25 – Экран карточки блюда

Использованные технологии

СУБД: MongoDB

Backend: ASP.NET Core 6, MongoDB Driver

Frontend: Vite, React

Ссылка на приложение

Ссылка на репозиторий проекта:

https://github.com/moevm/nosql2h22-diner

ЗАКЛЮЧЕНИЕ

В ходе работы было разработано веб-приложение, позволяющее автоматизировать управление внутренними процессами кафе быстрого питания. Приложение позволяет взаимодействовать с базой данных: просмотр содержимого, добавление элементов и экспорт/импорт данных, а также была реализована авторизация.

На данном этапе разработке приложение соответствует макету.

ПРИЛОЖЕНИЕ

Документация по сборке и развертыванию приложения

1. Скачать проект из репозитория;
2. docker compose build для сборки приложения;
3. docker compose up для запуска приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Документация ASP.NET [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/aspnet/core/release-notes/aspnetcore-6.0?view=aspnetcore-6.0

[2] Документация MongoDB [Электронный ресурс] URL: https://www.mongodb.com/docs/

[3] Документация Vite [Электронный ресурс] URL: https://vitejs.dev/

[4] Документация React [Электронный ресурс] URL: https://ru.reactjs.org/docs/getting-started.html