МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Казанский национальный исследовательский технический
университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)
Институт компьютерных технологий и защиты информации
(наименование института (факультета), филиала)

Кафедра Прикладной Математики и Информатики
(наименование кафедры)

09.03.01 «Информатика и вычислительная техника»
(шифр и наименование направления подготовки (специальности))

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

по дисциплине:
«Технологии Веб-программирования»

на тему: «Клиент-серверное приложение “Летнее кафе”»

Обучающийся 4317 _____ Идиятуллина А.А.
(номер группы) (подпись, дата) (Ф.И.О.)

Руководитель доцент каф. ПМИ Тутубалин П.И.
(должность) (Ф.И.О.)

Курсовая работа зачтена с оценкой _____

_____
(подпись, дата)

Казань 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Меняются времена и поколения, но одна вещь будет неизменной для человека всегда – это еда. Все любят вкусно покушать, а если в придачу к вкусной еде предложен удобный и хороший сервис, то клиент будет сыт не только физически, но и духовно. В системе общепита существует большое количество разнообразных сервисов по предоставлению клиентам информации о заведении, в том числе о блюдах, которые в нем подаются. Зачастую клиент хочет знать не только название блюда и его цену, но и то, из чего сделано это блюда. Также зачастую у человека возникает желание попробовать блюдо с определенным ингредиентом в составе или, наоборот, удостовериться, что в продукте нет его нелюбимого ингредиента или, еще хуже, аллергена. Все эти нюансы учтены в представленном сервисе.

Тема данной курсовой работы – разработка и реализация клиент-серверного приложения "Летнее кафе". В работе предложен один из вариантов подобного сервиса.

1. ПОСТАНОВКА ЗАДАЧИ

Целью курсовой работы является разработка и программная реализация клиент-серверного приложения, а также закрепления знаний, полученных при изучении дисциплины, и получение практических навыков программирования информационных систем с использованием современных технологий и инструментальных средств и оформления сопровождающей документации программного обеспечения.

Задача курсовой работы состоит в разработке клиент-серверного приложения «Летнее кафе» (web-клиента и web-сервера). Для выполнения этой задачи, были определены следующие требования:

- произвольная среда разработки – удобный современный текстовый редактор;
- применение системы контроля версий – git;
- опора на представление «Model-View-Controller»;
- разработка web-клиента с применением html, css, JavaScript, jQuery;
- разработка web-сервера с применением JSON, AJAX, Node.js, Express и NPM;
- использование базы данных (БД) на стороне web-сервера с применением СУБД – MongoDB;
- разворачивание ПО web-сервера либо на отдельном хосте, либо в среде виртуальной машины VirtualBox.

Актуальность работы заключается в том, что на технологии клиент-серверного взаимодействия строится практически вся сеть Интернет. Технология позволяет передавать информацию от серверного устройства к устройствам-клиентам на расстоянии.

Объектом моей курсовой работы является создание клиент-серверного приложения «Летнее кафе».

Предметом работы являются языки программирования для разработки приложений web-сервера и web-клиента.

2. МОДЕЛЬ ДАННЫХ СИСТЕМЫ

В данной работе используется уже ставшая классической модель Web-приложений на базе AJAX (см. рис. 2.1). Основная идея AJAX – возможность для приложения получать информацию и отдавать ее другим компьютерам без перезагрузки веб-страницы.

Рис. 2.1. Модель Web-приложений на базе AJAX

Страница посредством JavaScript в браузере пользователя, по какому-либо событию с помощью объекта XMLHttpRequest подает запрос серверу, обращаясь к некоему серверному обработчику, передавая ему некоторые параметры, например, значения, введенные пользователем в поле «Имя пользователя» на форме регистрации. Обработчик принимает этот запрос и обрабатывает его. По завершению обработки данных, обработчик выдает эти данные обратно браузеру пользователя.

3. МАКЕТЫ HTML-СТРАНИЦ СИСТЕМЫ

Макет главной страницы, а также страницы гостя изображен на рис. 3.1:

Рис. 3.1. Макет главной страницы

Макет страницы сотрудника и администратора изображен на рис. 3.2:

Рис. 3.2. Макет страницы сотрудника и администратора

4. СТРУКТУРНАЯ СХЕМА МОДУЛЕЙ ПО, СОСТАВЛЯЮЩИХ СИСТЕМУ

4.1 Структурная модель приложения. Модель MVC

Данная система использует шаблон «модель – представление – контроллер» (Model – View – Controller (MVC)) (см. рис. 4.1).

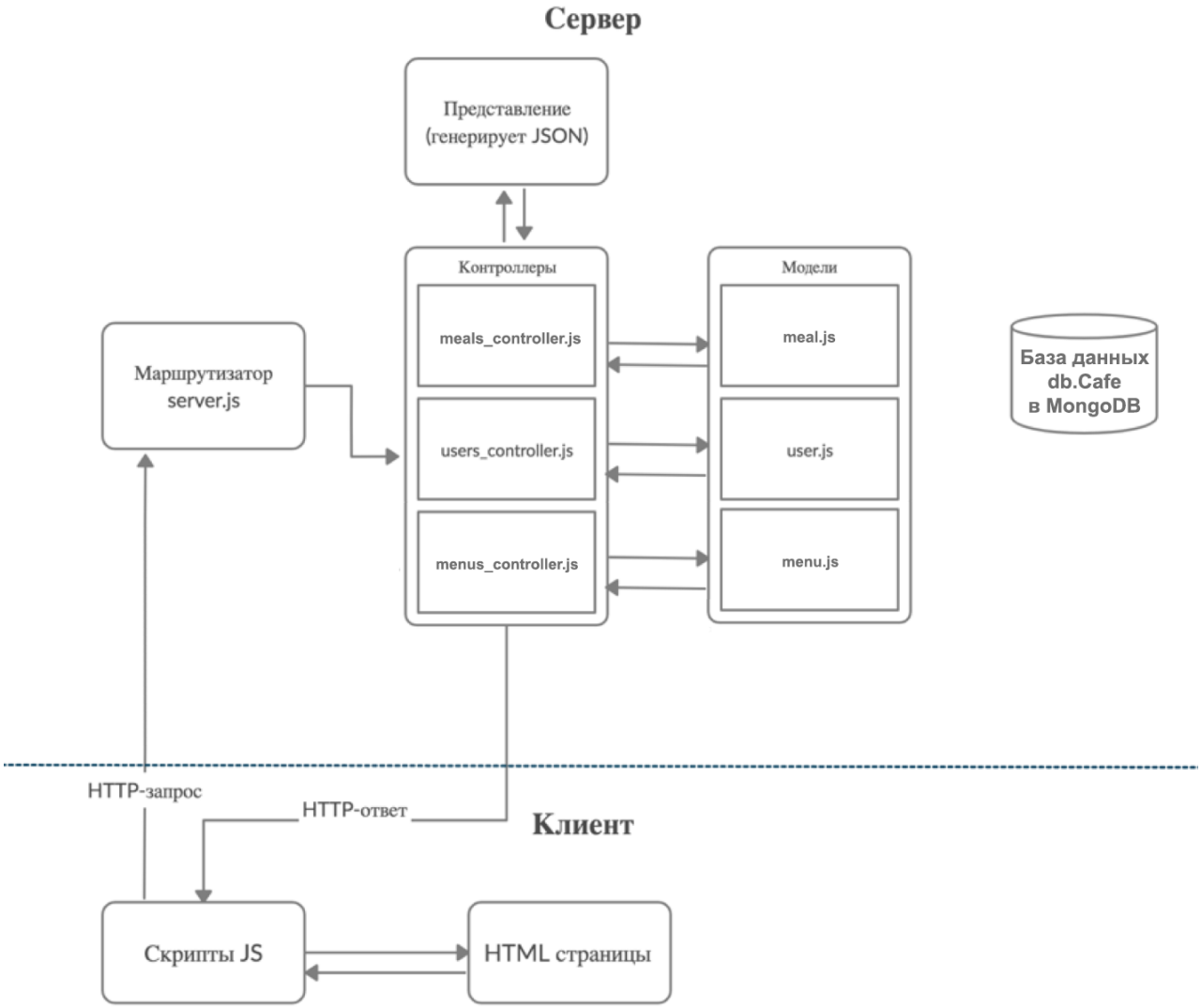

Рис. 4.1. Структурная схема приложения

Скрипты JS клиента (app.js, users.js) в зависимости от действий пользователя генерируют HTTP запросы.

Маршрутизатор – обрабатывает HTTP запрос с соответствующим действием контроллера.

Контроллер – преобразует запрос к серверу в действие. Как правило, соотносится с действием базы данных через модель. После этого отправляет ответ через представление.

Модель – это объектная абстракция элементов в базе данных, с ней взаимодействует контроллер.

Представление – в нашем случае это код HTML и CSS на клиентской стороне. Контроллер отправляет данные в виде JSON клиенту, а клиент решает, как их представить.

4.2 Структурная схема HTML страниц на стороне клиента

Структурная схема клиента, представленная HTML страницами, представлена на рис. 4.2:

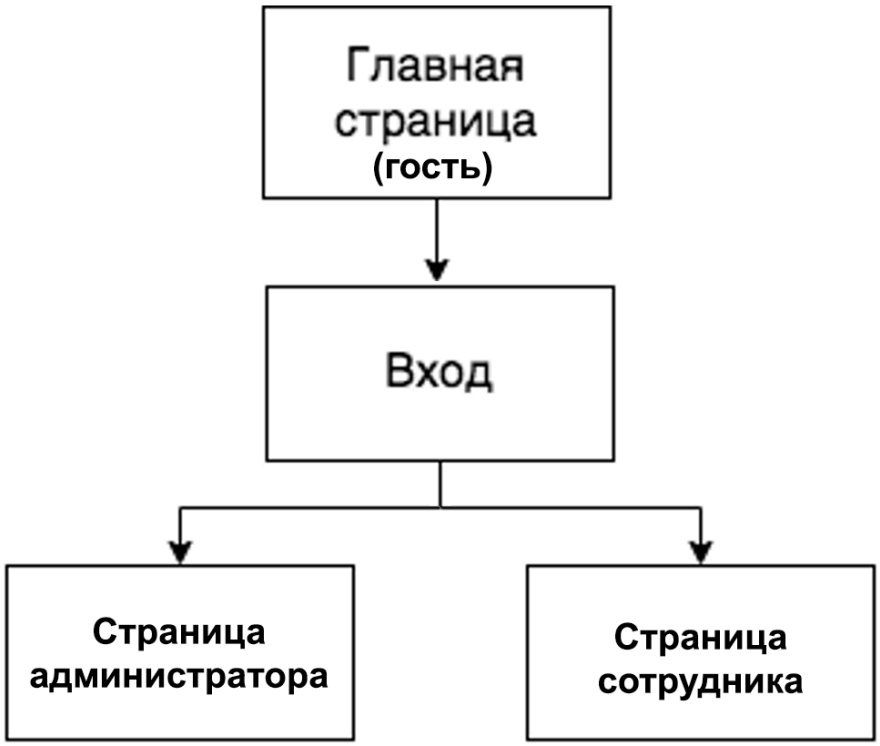

Рис. 4.2. Структурная схема клиента

5. ДИАГРАММА КЛАССОВ РАЗРАБОТАННОЙ СИСТЕМЫ

Диаграмма классов разработанной системы изображена на рис. 5.1:

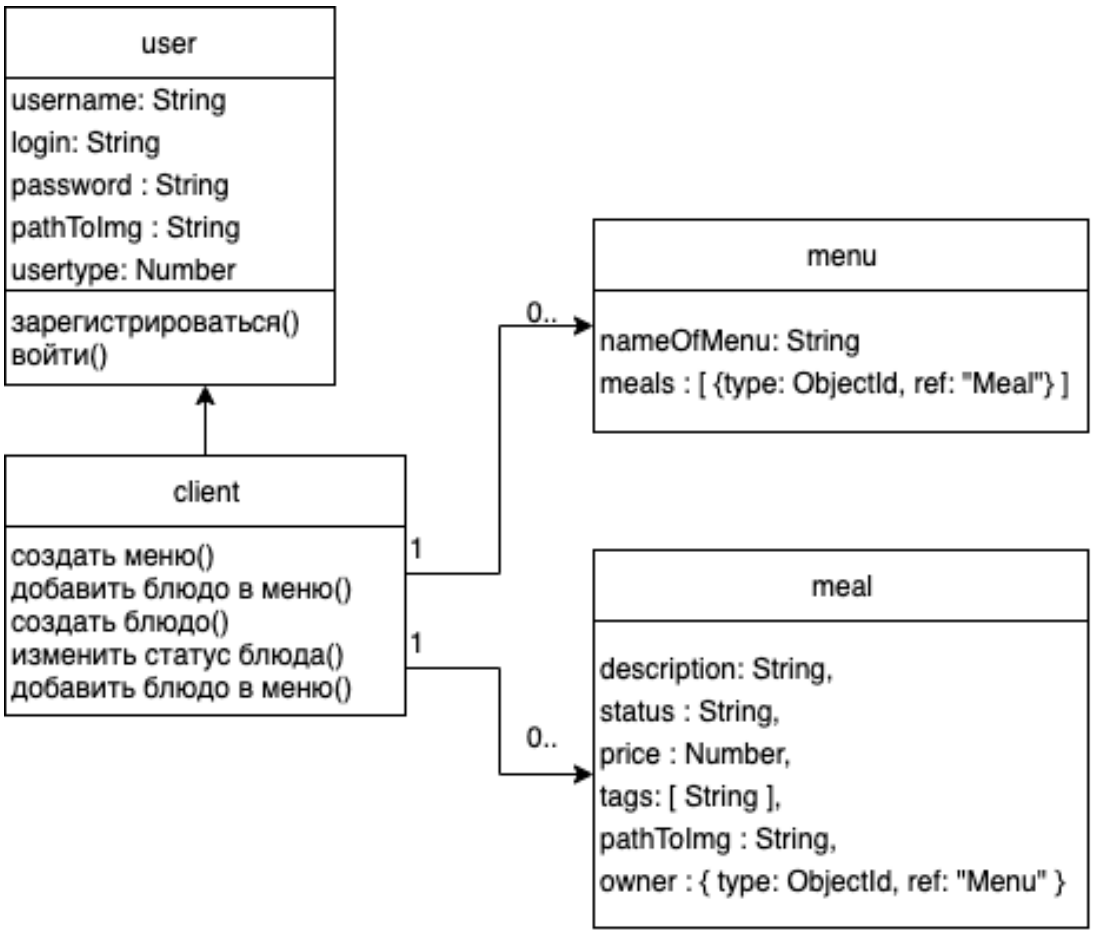

Рис. 5.1. Диаграмма классов разработанной системы

6. ОБЗОР ПО, ВЫБРАННОГО ДЛЯ РЕАЛИЗАЦИИ СИСТЕМЫ

6.1 Среда разработки – Sublime Text

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Некоторые возможности:

- Быстрая навигация (Goto Anything)
- Одновременное редактирование (Split Editing)
- Высокая степень настраиваемости (Customize Anything)
- Поддержка языков

Visual Studio Code поддерживает большое количество языков программирования и имеет возможность подсветки синтаксиса для C/C++/C#, CSS, HTML, Java, JavaScript, PHP, Python, SQL, XML и многих других.

В дополнение к тем языкам программирования, которые включены по умолчанию, пользователи имеют возможность загружать плагины для поддержки других языков.

6.2 Веб-браузер Yandex Browser и инструменты разработчика

Яндекс.Браузер — браузер, созданный компанией «Яндекс» на основе движка Blink, используемого в открытом браузере Chromium.

Так как Яндекс.Браузер является родственным Chromium'у, ему по определению присуща значительная часть преимуществ и недостатков последнего. Из-за этого браузер слабо выделяется основной функцией на фоне других многочисленных браузеров на базе WebKit и Blink.

Инструменты разработчика позволяют быстро анализировать содержание и ресурсы веб-страницы. Они полезны для проверки тегов Менеджера кампаний.

6.3 Программная платформа Node.js

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

В состав Node.js входит собственный установщик пакетов npm.

6.4 СУБД MongoDB

MongoDB — документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++. Используется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MEAN.

Поддерживается JavaScript в запросах, функциях агрегации.

7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

7.1. Главная страница сайта (страница гостя)

Для запуска программы необходимо в адресной строке веб-браузера необходимо ввести адрес сайта «Летнее кафе» и нажать клавишу Enter. Будет запущена главная страница сайта (см. рис. 7).

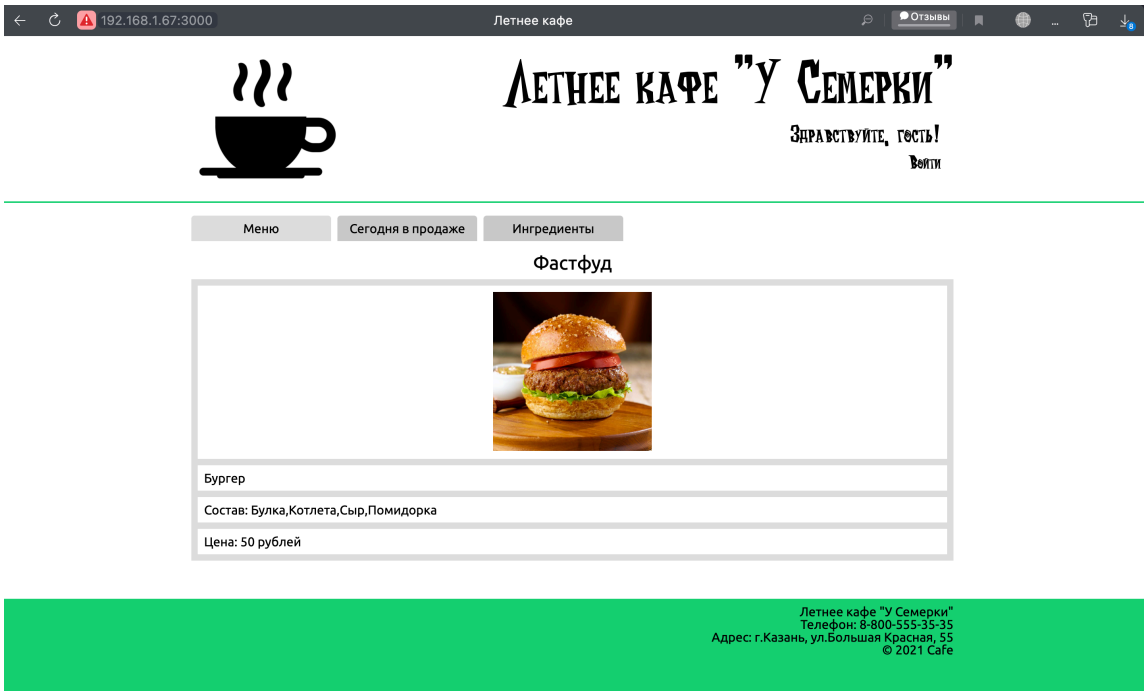

Рис. 7.1. Главная страница сайта

В «шапке» страницы изображена эмблема кафе и его название, а также приветственная надпись и кнопка входа в систему.

В центральной части страницы находятся вкладки «Меню», «Сегодня в продаже» и «Ингредиенты. На первой вкладке представлена информация обо всех блюдах кафе, блюда распределены между меню.

По нажатию на вторую вкладку будут выведены те блюда, которые есть в продаже сегодня.

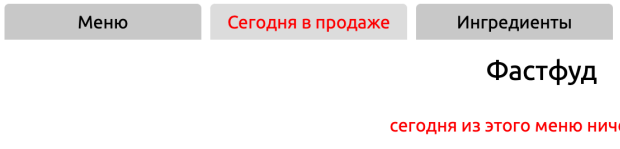

Рис. 7.2. Вкладка «Сегодня в продаже»

По нажатию на третью вкладку будет выведен список всех возможных ингредиентов и соответствующие для них блюда:

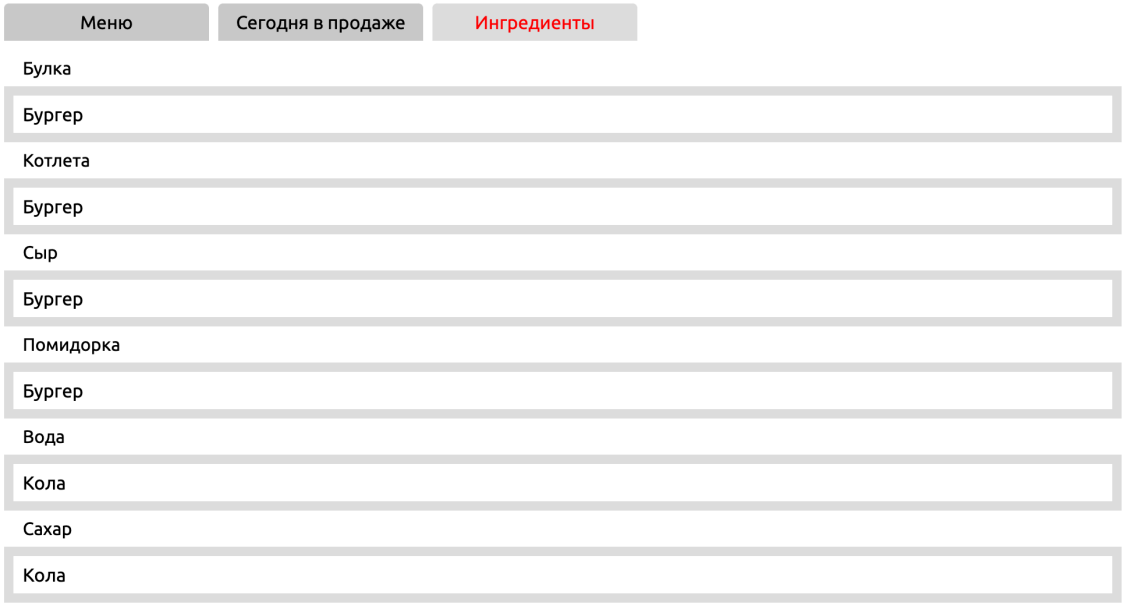

Рис. 7.3. Вкладка «Ингредиенты»

По нажатию на кнопку «Войти» в правом верхнем углу сайта будет открыта страница авторизации. Пользователю необходимо ввести логин и пароль для входа в систему:

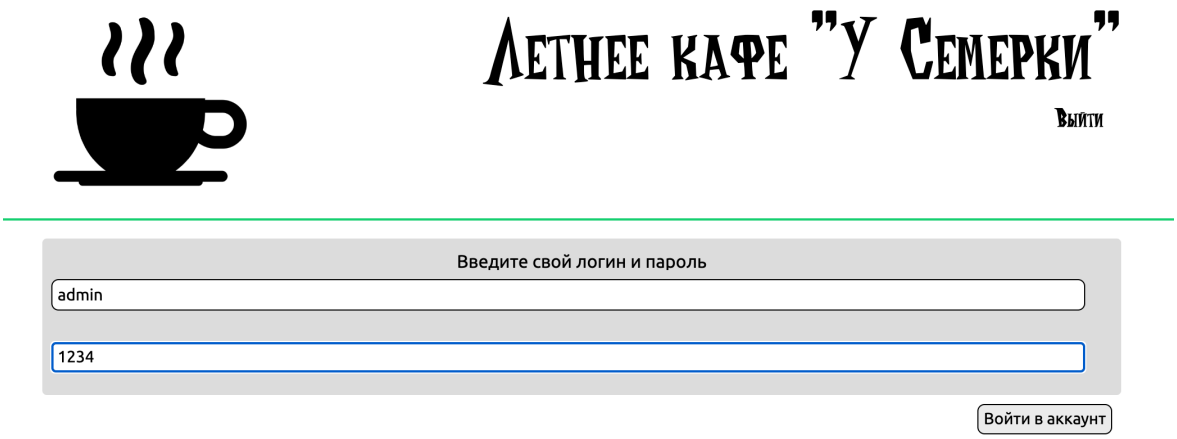

Рис. 7.4. Авторизация сайта

7.2. Страница администратора

При входе на страницу администратора будет открыта страница, на которой находится 6 вкладок: «Пользователи», «Новый сотрудник», «Все меню», «Создать меню», «Все блюда» и «Добавить блюдо». На первой вкладке находится информация обо всех пользователях (ФИО, изображение, путь к изображению, логин, пароль):

Рис. 7.5. Страница администратора

При нажатии на вкладку «Новый сотрудник» в центральной части страницы появится форма для заполнения данных о новом сотруднике:

Рис. 7.6. Вкладка «Новый сотрудник»

При нажатии на вкладку «Все меню» будет показан список всех созданных меню с возможностью их редактирования и удаления:

Рис. 7.7. Вкладка «Все меню»

При нажатии на вкладку «Создать меню» в центральной части страницы появится форма для заполнения данных о новом меню:

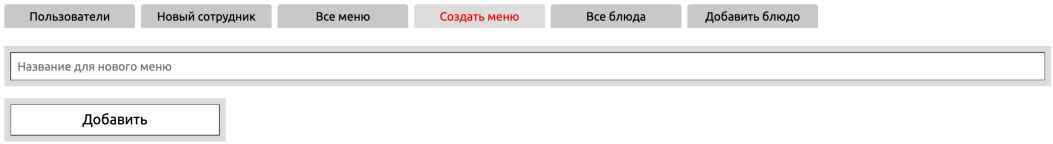

Рис. 7.8. Вкладка «Создать меню»

При нажатии на вкладку «Все блюда» будет показан список всех созданных блюд с возможностью добавления блюда в конкретное меню или удаления из него, с возможностью удаления блюда и изменения его статуса:

Рис. 7.9. Вкладка «Все блюда»

При нажатии на вкладку «Добавить блюдо» в центральной части страницы появится форма для заполнения данных о новом блюде:

Рис. 7.10. Вкладка «Добавить блюдо»

7.3. Страница сотрудника

При входе в аккаунт сотрудника будет открыта страница с тремя вкладками: «Все наши меню», «Сегодня в продаже» и «Нет в продаже». На первой вкладке показаны все блюда, рассортированные по соответствующим им меню. У сотрудника есть возможность изменять статус меню – сотрудник может указывать, в продаже блюдо сегодня или нет.

Рис. 7.11. Страница сотрудника

При нажатии на вторую и третью вкладки будет выдан список блюд со статусом «Сегодня в продаже» и «Нет в продаже» соответственно. На этих вкладках сотрудник также может изменять статус блюда.

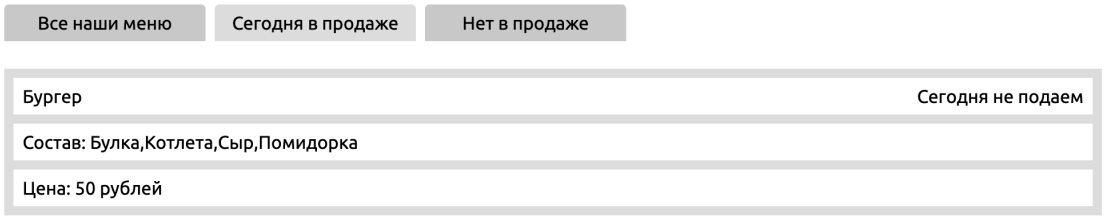

Рис. 7.12. Вкладка " Сегодня в продаже "

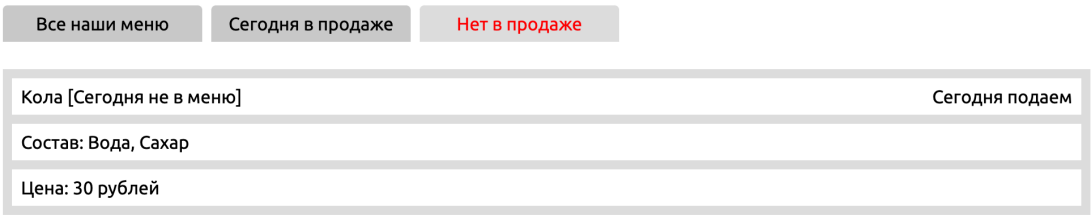

Рис. 7.13. Вкладка " Нет в продаже "

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Справочник по HTML. // HTMLBOOK [Электронный ресурс]. – Режим доступа: http://htmlbook.ru. – (Дата обращения: 5.06.2020).

2. Флэнаган Д. JavaScript. Подробное руководство. – Пер. с англ. – СПб: Символ-Плюс, 2008. – 992 с., ил.

3. Онлайн-руководство по MongoDB. // METANIT – сайт о программировании [Электронный ресурс]. – Режим доступа: https://metanit.com/nosql/mongodb/. – (Дата обращения: 6.06.2020)

4. JQuery // Википедия [Электронный ресурс]. Дата обновления: 29.01.2020. – Режим доступа: https://ru.wikipedia.org/?oldid=104831452 (дата обращения: 07.06.2020).

ПРИЛОЖЕНИЕ 1.Файл server.js

```
const { request } = require('express');

var express = require('express'),
    http = require("http"),
        // импорт представлений
        MealsController = require("./controllers/meals_controller.js"),
        UsersController = require("./controllers/users_controller.js"),
        MenuController = require("./controllers/menu_controller.js"),
    // импортируем библиотеку mongoose
    mongoose = require("mongoose"),
        database = 'Cafe'; //название хранилища в Mongo
    app = express();

// начинаем слушать запросы
http.createServer(app).listen(3000);

app.use('/',express.static(__dirname + "/client"));
app.use('/user/:username',express.static(__dirname + "/client"));

// командуем Express принять поступающие объекты JSON
app.use(express.urlencoded({ extended: true }));

// подключаемся к хранилищу данных в базе данных Mongo
mongoose.connect('mongodb://localhost/' + database, {
                useNewUrlParser: true,
                useCreateIndex: true,
                useUnifiedTopology: true
}).then(res => {
        console.log("DB Connected!")
}).catch(err => {
        console.log(Error, err.message);
});
```

```
// запросы для ВСЕХ блюд
app.get("/meals.json", MealsController.index);
app.get("/meals/:id", MealsController.show);
app.post("/meals", MealsController.create);
app.put("/meals/:id", MealsController.update);
app.delete("/meals/:id", MealsController.destroy);

// запросы для всех видов МЕНЮ
app.get("/menus.json", MenuController.index);
app.post("/menus", MenuController.create);
app.get("/menus/:id", MenuController.show);
app.put("/menus/:id", MenuController.update);
app.delete("/menus/:id", MenuController.destroy);

// запросы для пользователей
app.get("/users.json", UsersController.index);
app.post("/users", UsersController.create);
app.get("/users/:login", UsersController.show);
app.put("/users/:login", UsersController.update);
app.delete("/users/:login", UsersController.destroy);
```

ПРИЛОЖЕНИЕ 6. Файл users.js

```
var checkPassword = function (userObjects, login, password) {
    console.log(userObjects);
    var check = 0
    for (var i = userObjects.length-1; i>=0; i--) {
        if (userObjects[i].login == login) {
            console.log("пользователь найден");
            if (userObjects[i].password == password) {
                console.log("пароли совпадают");
                check = userObjects[i].pathToImg;

                if (userObjects[i].pathToImg == "") {
                    check = -1;
                }
                console.log(userObjects[i].pathToImg);
            }
            else { console.log("пароли не совпадают"); }
        }
    }
    return check
}

var initialize = function (password, callback) {
    console.log("Первый запуск приложения!");
    // добавляем пользователей
    $.get("/users.json", function (userObjects) {
        // создаем админа
        var admin = {"login": "admin", "password" : password,
"username" : "admin", "pathToImg" : null};
```

```
                $.post("/users", admin, function(result) {
                        console.log(result);
                        userObjects.push(admin); // отправляем на клиент
                }).done(function(responde) {
                        console.log(responde);
                }).fail(function(jqXHR, textStatus, error) {
                        console.log(error);
                        alert("Произошла  ошибка!\n"+jqXHR.status  +  "  "  +
jqXHR.textStatus);
                });

                var user = {"login": "user-1", "password" : "1111", "username" :
"Иванов Иван Иванович", "pathToImg" : "/images/users/user1.jpeg"};
                $.post("/users", user, function(result) {
                        console.log(result);
                        userObjects.push(user); // отправляем на клиент
                }).done(function(responde) {
                        console.log(responde);
                }).fail(function(jqXHR, textStatus, error) {
                        console.log(error);
                        alert("Произошла  ошибка!\n"+jqXHR.status  +  "  "  +
jqXHR.textStatus);
                });
        });

        // добавляем меню
        $.get("/menus.json", function (userObjects) {
                var menu = {"nameOfMenu" : "Фастфуд"};
                $.post("/menus", menu, function(result) {
```

```
                    console.log(result);
                    userObjects.push(menu); // отправляем на клиент
                }).done(function(responde) {
                    console.log(responde);
                }).fail(function(jqXHR, textStatus, error) {
                    console.log(error);
                    alert("Произошла ошибка!\n"+jqXHR.status + " " +
jqXHR.textStatus);
                });
            });

            // добавляем блюда
            $.get("/meals.json", function (userObjects) {
                var meal = {"description": "Бургер", "tags":
["Булка","Котлета","Сыр","Помидорка"], "status": 'Есть в меню', "price": 50,
"pathToImg" : "/images/meals/burger.jpeg"};
                $.post("/meals", meal, function(result) {
                    console.log(result);
                    userObjects.push(meal); // отправляем на клиент
                }).done(function(responde) {
                    console.log(responde);
                }).fail(function(jqXHR, textStatus, error) {
                    console.log(error);
                    alert("Произошла ошибка!\n"+jqXHR.status + " " +
jqXHR.textStatus);
                });
            })
```

```
    }

    var main = function (UsersObjects) {
        "use strict";
        var $info = $("<p align=center>").text("Введите свой логин и пароль"),
            $loginInput = $("<input width = 10px>").addClass("login"),
            $passwordInput = $("<input>").addClass("password"),
            $butLogin = $("<button>").text("Войти в аккаунт");

        $loginInput.attr('placeholder','Логин');
        $passwordInput.attr('placeholder','Пароль');

        $butLogin.on("click", function() {
            var login = $loginInput.val();
            var password = $passwordInput.val();

            if (login == "admin" && password !== null && password.trim() !== "") {
                $.ajax({
                    'url': '/users/'+login,
                    'type': 'GET'
                }).done(function(responde) {
                    console.log("все норм, заходим под админом");
                    let pathToImg = checkPassword(UsersObjects, login, password);
                    if (pathToImg != 0) {
                        window.location.replace('users/' + login + '/');
                    } else {
```

```
                                alert("Пароль администратора введен
неверно!!!");
                            }


                        }).fail(function(jqXHR, error) {
                            console.log("все не норм, ошибка");
                            if (jqXHR.status) {
                                initialize(password, function() {
                                    $butLogin.trigger("click");
                                });
                                alert(" Аккаунт администратора создан! \n
Добро пожаловать на сайт кафе 'У семерки'!");
                                window.location.reload();
                            } else {
                                alert("Произошла
ошибка!\n"+jqXHR.status + " " + jqXHR.textStatus);
                            }
                        });
                    }

                    else if (login !== null && login.trim() !== "" && password !==
null && password.trim() !== "") {
                        let pathToImg = checkPassword(UsersObjects, login,
password);
                        if (pathToImg != 0) {
                            Cookies.set('CurrentUser', login);
                            Cookies.set('PathToImgOfUser', pathToImg);
                            $.ajax({
```

```
                    'url': '/users/'+login,
                    'type': 'GET'
                }).done(function(responde) {
                    window.location.replace('users/' + login + '/');
                }).fail(function(jqXHR, error) {
                    console.log(error);
                    alert("Произошла
ошибка!\n"+jqXHR.status + " " + jqXHR.textStatus);
                });
            }
            else {
                alert("Пароль введен неверно или пользователя
не существует!");
            };
        } else {
            alert("Одно из полей пустое!");
        }
    });

    $("main .authorization").append($info);
    $("main .authorization").append($loginInput);
    $("main .authorization").append('<p>');
    $("main .authorization").append($passwordInput);
    $("main .authorization").append('<p>');
    $("main .authorization").append($butLogin);
}

$(document).ready(function() {
    $.getJSON("users.json", function (UsersObjects) {
```

```
                main(UsersObjects);
        });
});
```

ПРИЛОЖЕНИЕ 7. Файл worker.js

```
var organizeByTags = function (mealObjects) {
    // создание пустого массива для тегов
    var tags = [];
    // перебираем все задачи meals
    mealObjects.forEach(function (Receipt) {
        // перебираем все теги для каждой задачи
        Receipt.tags.forEach(function (tag) {
            // убеждаемся, что этого тега еще нет в массиве
            if (tags.indexOf(tag) === -1) {
                tags.push(tag);
            }
        });
    });
    var tagObjects = tags.map(function (tag) {
        // здесь мы находим все задачи,
        // содержащие этот тег
        var mealsWithTag = [];
        mealObjects.forEach(function (Receipt) {
            // проверка, что результат
            // indexOf is *не* равен -1
            if (Receipt.tags.indexOf(tag) !== -1) {
                mealsWithTag.push(Receipt.description);
            }
        });
        // мы связываем каждый тег с объектом, который содержит
название тега и массив
        return { "name": tag, "meals": mealsWithTag };
    });
    return tagObjects;
};

var addMealToMenu = function (Receipt, callback) {
    var $mealListItem = $("<li>").text(Receipt.description),
        $mealEditLink = $("<a>").attr("href", "/meals/" + Receipt._id),
```

```
                $mealReturnLink = $("<a>").attr("href", "/meals/" +
Receipt._id);

            $mealEditLink.addClass("linkEdit");
            $mealReturnLink.addClass("linkRemove");

            if (Receipt.status === 'Есть в меню') {
                $mealEditLink.text("Сегодня не подаем");
                $mealEditLink.on("click", function() {
                    var newDescription = Receipt.description + " [Сегодня не
в меню]";
                    if (newDescription !== null && newDescription.trim()
!== "") {
                        $.ajax({
                            "url": "/meals/" + Receipt._id,
                            "type": "PUT",
                            "data": { "description": newDescription,
"status": 'Сегодня не в меню',"price" : Receipt.price, "owner" : Receipt.owner }
                        }).done(function (responde) {
                            Receipt.status = 'Сегодня не в меню';
                            // location.reload();
                            callback();
                        }).fail(function (err) {
                            console.log("Произошла ошибка: " + err);
                        });
                    }
                    return false;
                });
                $mealListItem.append($mealEditLink);
            }
            else {
                $mealReturnLink.text("Сегодня подаем");
                $mealReturnLink.on("click", function () {
                    var oldDes = Receipt.description
```

```
                        var newDescription = oldDes.split(' [Сегодня не в меню]').join(' ');
                        var newDescription = newDescription.substring(0, newDescription.length - 1);
                        $.ajax({
                                "url": "/meals/" + Receipt._id,
                                "type": "PUT",
                                "data": { "description": newDescription, "status": 'Есть в меню',"price" : Receipt.price, "owner" : Receipt.owner }
                        }).done(function (responde) {
                                Receipt.status = 'Есть в меню';
                                // location.reload();
                                callback();
                        }).fail(function (err) {
                                console.log("Произошла ошибка: " + err);
                        });
                        return false;
                });
                $mealListItem.append($mealReturnLink);
        }

        return $mealListItem;
}

var main = function (mealObjects) {
        "use strict";
        // создание пустого массива с вкладками
        var tabs = [];

        // добавляем вкладку Меню
        tabs.push({
                "name": "Все наши меню",
                "content": function(callback) {
                        var arr = [];
                        var $content = $("<ul>");
```

```
                $.getJSON("/menus.json", function (menusObjects) {
                    console.log("вызов 1");
                    for (var i = menusObjects.length-1; i>=0; i--) {
                        let size = menusObjects[i].meals.length;
                        // console.log(size);
                        if (size != 0) { arr.push(menusObjects[i]);
}
                    }
                    // console.log(arr);
                }).fail(function (jqXHR, textStatus, error) {
                    callback(error, null);
                });

                $.getJSON("/meals.json", function (mealsObjects) {
                    console.log("вызов 2");
                    for (var i = arr.length-1; i>=0; i--) {
                        var $textInput =
$("<h3>").text(arr[i].nameOfMenu);
                        $textInput.addClass("nameOfMenu");
                        let id = arr[i]._id
                        $content.append($textInput);
                        for (var j = 0; j<=mealsObjects.length-1; j++)
{
                            if (mealsObjects[j].owner == id) {
                                console.log("вызов 3");
                                var $mealListItem =
addMealToMenu(mealsObjects[j], function() {
                                    console.log("вызов 4");
                                    $(".tabs a:first-child
span").trigger("click");
                                    console.log("есть
нажатие");
                                });
                                var $text = $("<p>").text("");
```

```
                                                var $ingredients =
$("<li>").text("Состав: " + mealsObjects[j].tags);
                                                var $price =
$("<li>").text("Цена: " + mealsObjects[j].price + " рублей");

        $content.append($mealListItem);
                                                $content.append($ingredients);
                                                $content.append($price);
                                                $content.append($text);
                                        }
                                }

                                }
                                callback(null, $content);

                        }).done(function(responde) {
                                callback(null, $content);
                                console.log("responde : " + responde[0].textStatus);
                        }).fail(function (jqXHR, textStatus, error) {
                                callback(error, null);
                        });
                }
        });

        // добавляем вкладку Сегодня в меню
        tabs.push({
                "name": "Сегодня в продаже",
                "content": function(callback) {
                        $.getJSON("/meals.json", function (mealObjects) {
                                var $content;
                                $content = $("<ul>");
                                for (var i = 0; i < mealObjects.length; i++) {
                                        if (mealObjects[i].status === 'Есть в меню')
{
```

```
                                        var $mealListItem = addMealToMenu(mealObjects[i], function() {
                                            $(".tabs a:nth-child(2) span").trigger("click");
                                        });
                                        var $text = $("<p>").text("");
                                        var $ingredients = $("<li>").text("Состав: " + mealObjects[i].tags);
                                        var $price = $("<li>").text("Цена: " + mealObjects[i].price + " рублей");
                                        $content.append($text);
                                        $content.append($mealListItem);
                                        $content.append($ingredients);
                                        $content.append($price);
                                    }
                                }
                                callback(null, $content);
                            }).fail(function(jqXHR, textStatus, error) {
                                callback(error, null);
                            });
                        }
                    });

                    // добавляем вкладку Нет в меню
                    tabs.push({
                        "name": "Нет в продаже",
                        "content": function(callback) {
                            $.getJSON("/meals.json", function (mealObjects) {
                                var $content;
                                $content = $("<ul>");
                                for (var i = 0; i < mealObjects.length; i++) {
                                    if (mealObjects[i].status === 'Сегодня не в меню') {
                                        var $mealListItem = addMealToMenu(mealObjects[i], function() {
```

```
                                        $(".tabs a:nth-child(3) span").trigger("click");
                                    });
                                    var $text = $("<p>").text("");
                                    var $ingredients = $("<li>").text("Состав: " + mealObjects[i].tags);
                                    var $price = $("<li>").text("Цена: " + mealObjects[i].price + " рублей");
                                    $content.append($text);
                                    $content.append($mealListItem);
                                    $content.append($ingredients);
                                    $content.append($price);
                                }
                            }
                            callback(null, $content);
                        }).fail(function(jqXHR, textStatus, error) {
                            callback(error, null);
                        });
                    }
                });

                tabs.forEach(function (tab) {
                    var $aElement = $("<a>").attr("href",""),
                        $spanElement = $("<span>").text(tab.name);
                    $aElement.append($spanElement);
                    $("main .tabs").append($aElement);

                    $spanElement.on("click", function () {
                        $(".tabs a span").removeClass("active");
                        $spanElement.addClass("active");
                        $("main .content").empty();
                        tab.content(function (err, $content) {
                            if (err !== null) { alert ("Возникла проблема при обработке запроса: " + err); }
                            else { $("main .content").append($content); }
```

```
                });
                return false;
            });
        });

        $(".tabs a:first-child span").trigger("click");
    }

    $(document).ready(function() {
        $.getJSON("/meals.json", function (mealObjects) {
            var $userPhoto;
            var login  = Cookies.get('CurrentUser');
            var pathToImgOfUser = Cookies.get('PathToImgOfUser');
            console.log(pathToImgOfUser);

            $.ajax({
                url: pathToImgOfUser,
                type:'HEAD',
                error: function()
                {
                    console.log("изображения нет!!!");
                    pathToImgOfUser = "/images/user_default.png";
                    $userPhoto = $("<img>").attr("src",
pathToImgOfUser).addClass("main_image");
                    $("header .image").append($userPhoto);
                },
                success: function()
                {
                    console.log("изображение есть!!!");
                    $userPhoto = $("<img>").attr("src",
pathToImgOfUser).addClass("main_image");
                    $("header .image").append($userPhoto);
                }
            });
```

```
                var $place = $("<p>").text("Здравствуйте, " + login + "!");
                $("header .username").append($place);
                console.log(login);
                main(mealObjects);
        });
});
```

ПРИЛОЖЕНИЕ 8. Файл guest.js

```
var organizeByTags = function (mealObjects) {
    // создание пустого массива для тегов
    var tags = [];
    // перебираем все задачи meals
    mealObjects.forEach(function (Receipt) {
        // перебираем все теги для каждой задачи
        Receipt.tags.forEach(function (tag) {
            // убеждаемся, что этого тега еще нет в массиве
            if (tags.indexOf(tag) === -1) {  tags.push(tag); }
        });
    });
    var tagObjects = tags.map(function (tag) {
        // здесь мы находим все блюда с тегом
        var mealsWithTag = [];
        mealObjects.forEach(function (Receipt) {
            if (Receipt.tags.indexOf(tag) !== -1) { mealsWithTag.push(Receipt.description); }
        });
        // мы связываем каждый тег с объектом, который содержит название тега и массив
        return { "name": tag, "meals": mealsWithTag };
    });
    return tagObjects;
};

var mealListItem = function (Meal, callback) {
    var $content = $("<ul>");
    var $mealListItem = $("<li>");

    $mealListItem.text(Meal.description);

    var $image = $("<li align = center>");
    $content.append($image);
    let pathToImgOfMeal = Meal.pathToImg;
```

```
            $.ajax({
                url: pathToImgOfMeal,
                type:'HEAD',
                error: function() {
                    pathToImgOfMeal = "/images/user_default.png";
                    $image.append($("<img>").attr("src",
pathToImgOfMeal).addClass("main_image"));
                },
                success: function() {
                    $image.append($("<img>").attr("src",
pathToImgOfMeal).addClass("main_image"));
                }
            });

            var $ingredients = $("<li>").text("Состав: " + Meal.tags);
            var $price = $("<li>").text("Цена: " + Meal.price + " рублей");
            $content.append($mealListItem);
            $content.append($ingredients);
            $content.append($price);
            return $content;
        }

        var main = function (mealObjects) {
            "use strict";
            // создание пустого массива с вкладками
            var tabs = [];
            // добавляем вкладку Меню
            tabs.push({
                "name": "Меню",
                "content": function(callback) {
                    var arr = [];
                    var $content = $("<ul>");
                    $.getJSON("/menus.json", function (menusObjects) {
                        for (var i = menusObjects.length-1; i>=0; i--) {
                            let size = menusObjects[i].meals.length;
```

```
                    if (size != 0) { arr.push(menusObjects[i]);
}
                }
                callback(null);
            }).fail(function (jqXHR, textStatus, error) {
                callback(error, null);
            });

            $.getJSON("/meals.json", function (mealsObjects) {
                for (var i = arr.length-1; i>=0; i--) {
                    var $textInput =
$("<h3>").text(arr[i].nameOfMenu);
                    $textInput.addClass("nameOfMenu");
                    let id = arr[i]._id
                    $content.append($textInput);

                    for (var j = 0; j<=mealsObjects.length-1; j++)
{
                        if (mealsObjects[j].owner == id) {

                            var $mealListItem =
mealListItem(mealsObjects[j], function() {
                                $(".tabs a:nth-child(1)
span").trigger("click");
                            });


        $content.append($mealListItem);

        $content.append($("<p>").text(""));
                        }
                    }
                }
                callback(null, $content);
            }).fail(function (jqXHR, textStatus, error) {
```

```
                callback(error, null);
            });


        }
    });

    // добавляем вкладку Сегодня в меню
    tabs.push({
        "name": "Сегодня в продаже",
        "content": function(callback) {
            // $.getJSON("/meals.json", function (mealObjects) {
                var arr = [];
                var $content = $("<ul>");
                $.getJSON("/menus.json", function (menusObjects)
{
                    for (var i = menusObjects.length-1; i>=0; i--)
{
                        let size =
menusObjects[i].meals.length;
                        if (size != 0) {
arr.push(menusObjects[i]);      }
                    }
                    callback(null);
                }).fail(function (jqXHR, textStatus, error) {
                    callback(error, null);
                });


                $.getJSON("/meals.json", function (mealsObjects) {

                    for (var i = arr.length-1; i>=0; i--) {
                        console.log("test 1");
                        var $textInput =
$("<h3>").text(arr[i].nameOfMenu);
```

```
                                        $textInput.addClass("nameOfMenu");
                                        let id = arr[i]._id
                                        $content.append($textInput);
                                        var count = 0;
                                        for (var j = 0; j<=mealsObjects.length-
1; j++) {
                                            console.log("test 2");
                                            if (mealsObjects[j].status ==
'Есть в меню' && mealsObjects[j].owner == id) {
                                                count++;
                                                var $mealListItem =
mealListItem(mealsObjects[j], function() {
                                                    $(".tabs a:nth-
child(2) span").trigger("click");
                                                });


        $content.append($mealListItem);

        $content.append($("<p>").text(""));
                                            }
                                        }

                                        if (count == 0) {
                                            $content.append($("<p
align=center>").text("сегодня из этого меню ничего нет :("));
                                        }
                                    }
                                    callback(null, $content);
                                }).fail(function (jqXHR, textStatus, error) {
                                    callback(error, null);
                                });

                                callback(null, $content);
```

```
        }
    });

    // добавляем вкладку Поиск по ингредиентам
    tabs.push({
        "name": "Ингредиенты",
        "content":function (callback) {
            $.get("meals.json", function (mealObjects) {
                // создание $content для Теги
                var organizedByTag = organizeByTags(mealObjects), $content;
                organizedByTag.forEach(function (tag) {
                    var $tagName = $("<h3>").text(tag.name);
                        $content = $("<ul>");
                    tag.meals.forEach(function (description) {
                        var $li = $("<li>").text(description);
                        $content.append($li);
                    });
                    $("main .content").append($tagName);
                    $("main .content").append($content);
                });
                callback(null,$content);
            }).fail(function (jqXHR, textStatus, error) { callback(error, null); });
        }
    });

    tabs.forEach(function (tab) {
        var $aElement = $("<a>").attr("href",""),
            $spanElement = $("<span>").text(tab.name);
        $aElement.append($spanElement);
        $("main .tabs").append($aElement);

        $spanElement.on("click", function () {
            var $content;
```

```
                $(".tabs a span").removeClass("active");
                $spanElement.addClass("active");
                $("main .content").empty();
                tab.content(function (err, $content) {
                    if (err !== null) { alert ("Возникла проблема при обработке запроса: " + err); }
                    else { $("main .content").append($content); }
                });
                return false;
            });
        });

        $(".tabs a:first-child span").trigger("click");
    }

    $(document).ready(function() {
        $.getJSON("menus.json", function (menusObjects) {
            var $place = $("<p>").text("Здравствуйте, гость!");
            $("header .username").append($place);
            // console.log(username);
            main(menusObjects);
        });
    });
```

ПРИЛОЖЕНИЕ 9. Файл admin.js

```
var liaWithEditOrDeleteOnClick = function (User, callback) {

    var $content = $("<ul>");

    var $userListItem = $("<li>").text("ФИО: " + User.username + ", Логин: " + User.login + ", Пароль: " + User.password);
    $content.append($userListItem);
    //поменять проверку с имени на тип
  if (User.login != "admin") {
     var $userRemoveLink = $("<a>").attr("href", "/users/" + User.login);

        var $image = $("<li align = center>");
        let pathToImgOfUser = User.pathToImg;

        $.ajax({
            url: pathToImgOfUser,
            type:'HEAD',
            error: function()
            {
                // console.log("изображения нет!!!");
                pathToImgOfUser = "/images/user_default.png";
                $image.append($("<img>").attr("src", pathToImgOfUser).addClass("main_image"));
                $content.append($image);
            },
            success: function()
            {
                // console.log("изображение есть!!!");
                $image.append($("<img>").attr("src", pathToImgOfUser).addClass("main_image"));
                $content.append($image);
            }
        });
```

```
                var $userPhotoPath = $("<li>").text("Путь к изображению: " +
User.pathToImg);
                $content.append($userPhotoPath);

            $userRemoveLink.addClass("linkRemove");
            $userRemoveLink.text("Удалить");
            $userRemoveLink.on("click", function () {
              $.ajax({
                url: "/users/" + User.login,
                type: "DELETE"
              }).done(function (responde) {
                callback();
              }).fail(function (err) {
                console.log("Произошла ошибка: " + err.textStatus);
              });
              return false;
            });
            $userListItem.append($userRemoveLink);
          }

            return $content;
        }

        var addMealToMenu = function(Receipt, callback) {

            var $content = $("<ul>");

            var $mealListItem = $("<li>"),
                $mealEditLink = $("<a>").attr("href", "/meals/" + Receipt._id),
                $mealReturnLink = $("<a>").attr("href", "/meals/" +
Receipt._id),
                $menusList = $("<select name = 'menus'>"),
                $addMealToMenu = $("<a>").attr("href", "/meals/" +
Receipt._id),
```

```
                $removeMealFromMenu = $("<a>").attr("href", "/meals/" +
Receipt._id);
                $removeMeal = $("<a>").attr("href", "/meals/" + Receipt._id);


        $mealListItem.text(Receipt.description);


        $mealEditLink.addClass("linkEdit");
        $mealReturnLink.addClass("linkRemove");
        $menusList.prepend('<option value="-1">Выберете меню</option>');
        $menusList.addClass("linkList");
        $addMealToMenu.addClass("linkEdit");
        $removeMealFromMenu.addClass("linkEdit");
        $removeMeal.addClass("linkRemove");

        var $image = $("<li align = center>");
        let pathToImgOfMeal = Receipt.pathToImg;
        $content.append($image);
        $.ajax({
            url: pathToImgOfMeal,
            type:'HEAD',
            error: function()
            {
                pathToImgOfMeal = "/images/user_default.png";
                $image.append($("<img>").attr("src",
pathToImgOfMeal).addClass("main_image"));
            },
            success: function()
            {
                $image.append($("<img>").attr("src",
pathToImgOfMeal).addClass("main_image"));
            }
        });
```

```
            if (Receipt.status === 'Есть в меню') {
                $mealEditLink.text("Сегодня не подаем");
                $mealEditLink.on("click", function() {
                    var newDescription = Receipt.description + " [Сегодня не
в меню]";
                    if (newDescription !== null && newDescription.trim()
!== "") {
                        $.ajax({
                            "url": "/meals/" + Receipt._id,
                            "type": "PUT",
                            "data": { "description": newDescription,
"status": 'Сегодня не в меню',"price" : Receipt.price, "owner" : Receipt.owner }
                        }).done(function (responde) {
                            Receipt.status = 'Сегодня не в меню';
                            callback();
                        }).fail(function (err) {
                            console.log("Произошла ошибка: " + err);
                        });
                    }
                    return false;
                });
                $mealListItem.append($mealEditLink);
            }
            else {
                $mealReturnLink.text("Сегодня подаем");
                $mealReturnLink.on("click", function () {
                    var oldDes = Receipt.description
                    var newDescription = oldDes.split(' [Сегодня не в
меню]').join(' ');
                    var newDescription = newDescription.substring(0,
newDescription.length - 1);
                    $.ajax({
                        "url": "/meals/" + Receipt._id,
                        "type": "PUT",
```

```
                    "data": { "description": newDescription, "status":
'Есть в меню',"price" : Receipt.price, "owner" : Receipt.owner }
                }).done(function (responde) {
                    Receipt.status = 'Есть в меню';
                    callback();
                }).fail(function (err) {
                    console.log("Произошла ошибка: " + err);
                });
                return false;
            });
            $mealListItem.append($mealReturnLink);
        }

        //добавляем меню в список
        $.getJSON("/menus.json", function (menusObjects) {
            for (var i = menusObjects.length-1; i>=0; i--) {
                $menusList.append('<option value="' +
menusObjects[i]._id + '">' + menusObjects[i].nameOfMenu + '</option>');
            }
        }).fail(function (jqXHR, textStatus, error) {
            callback(error, null);
        });

        $removeMeal.text("Удалить блюдо");
        $removeMeal.on("click", function() {
            $.ajax({
                "url": "/meals/" + Receipt._id,
                "type": "DELETE"
            }).done(function (responde) {
                callback();
            }).fail(function (err) {
                console.log("Произошла ошибка: " + err);
            });
            return false;
        });
```

```
            $mealListItem.append($removeMeal);

            if (Receipt.owner === null) {
                $mealListItem.append($menusList);
                $addMealToMenu.text("Добавить в меню ");
                $addMealToMenu.on("click", function() {
                    console.log("выбрано меню: " + $menusList.val());
                    if ($menusList.val() == -1) {
                        alert ("Вы не выбрали меню!!!");
                    } else {
                        var selectedMenuID = $menusList.val(); //id
выбранного Меню
                        $.ajax({
                            "url": "/meals/" + Receipt._id,
                            "type": "PUT",
                            "data": { "description": Receipt.description,
"status": Receipt.status, "price" : Receipt.price, "owner" : selectedMenuID },
                        }).done(function (responde) {
                            callback();
                        }).fail(function (err) {
                            console.log("Произошла ошибка: " + error);
                        });
                        $.getJSON("/menus.json", function (menusObjects)
{
                            console.log(menusObjects);
                            var newArray = [];
                            for (var i = menusObjects.length-1; i>=0; i--)
{
                                console.log(newArray);
                                if (menusObjects[i]._id ==
selectedMenuID) {
                                    console.log("меню найдено");
                                    newArray =
menusObjects[i].meals;
                                    newArray.push(Receipt);
```

```
                                                    console.log(newArray);
                                                    $.ajax({
                                                        "url": "/menus/" +
menusObjects[i]._id,
                                                        "type": "PUT",
                                                        "data": { "meals":
newArray, "nameOfMenu" :  menusObjects[i].nameOfMenu},
                                                    }).done(function (responde) {
                                                        // callback();
                                                    }).fail(function (err) {
                                                        console.log("Произошла
ошибка: " + error);
                                                    });
                                                }
                                            }
                                            // callback();
                                        }).fail(function (jqXHR, textStatus, error) {
                                            callback(error, null);
                                        });
                                    }

                                    return false;
                                });
                                $mealListItem.append($addMealToMenu);
                            } else {
                                $removeMealFromMenu.text("Удалить из меню");
                                $removeMealFromMenu.on("click", function() {
                                    let mealID = Receipt._id;
                                    $.ajax({
                                        "url": "/meals/" + Receipt._id,
                                        "type": "PUT",
                                        "data": { "description": Receipt.description,
"status": Receipt.status, "price" : Receipt.price, "owner" : null },
                                    }).done(function (responde) {
                                        callback();
```

```
                    }).fail(function (err) {
                        console.log("Произошла ошибка: " + err);
                    });

                    $.getJSON("/menus.json", function (menusObjects) {
                        console.log(menusObjects);
                        var check = 0;
                        for (var i = menusObjects.length-1; i>=0; i--) {
                            var oldArray = [];
                            var newArray = [];
                            console.log("menuID : " +
menusObjects[i]._id);
                            let menuID = menusObjects[i]._id; //id Меню
                            let menuName =
menusObjects[i].nameOfMenu;
                            oldArray = menusObjects[i].meals;
                            for (var j = 0; j<=oldArray.length; j++) {
                                if (oldArray[j] == Receipt._id) {
                                    console.log("меню найдено,
удаляем блюдо из него");
                                    check = 1;
                                } else {
                                    console.log("добавляем в
массив newArray");
                                    newArray.push(oldArray[j]);
                                }
                            }
                            console.log(newArray);
                            if (check === 1 ) {
                                $.ajax({
                                    "url": "/menus/" + menuID,
                                    "type": "PUT",
                                    "data": { "meals": newArray,
"nameOfMenu" : menuName},
                                })
```

```
                                        .done(function (responde) {
                                            // callback();
                                        })
                                        .fail(function(jqXHR, textStatus, error)
{
                                            if (jqXHR.status === 501) {
console.log("Блюдо с таким названием уже существует!"); }
                                            else { console.log("Произошла
ошибка!\n"+jqXHR.status + " " + jqXHR.textStatus);     }
                                        });

                                        console.log("i = " + i);

                                            break;

                                    }
                                }
                                // callback();
                            }).fail(function (jqXHR, textStatus, error) {
                                callback(error, null);
                            });

                            return false;
                        });
                        $mealListItem.append($removeMealFromMenu);
                    }

                    var $ingredients = $("<li>").text("Состав: " + Receipt.tags);
                    var $price = $("<li>").text("Цена: " + Receipt.price + " рублей");
                    var $mealPhotoPath = $("<li>").text("Путь к изображению: " +
Receipt.pathToImg);
                    $content.append($mealListItem);

                    var $nameOfMenu = $("<li>");
                    $.getJSON("/menus.json", function (menusObjects) {
```

```
                for (var i = menusObjects.length-1; i>=0; i--) {
                        if (menusObjects[i]._id == Receipt.owner) {
                                nameOfMenu = menusObjects[i].nameOfMenu
                                $nameOfMenu.text("Меню: " + nameOfMenu)
                        }
                }
                if (Receipt.owner == null) {
                        $nameOfMenu.text("Меню: не указано")
                }
        });

        $content.append($nameOfMenu);
        $content.append($ingredients);
        $content.append($price);
        $content.append($mealPhotoPath);
        return $content;
    }

    var editOrDeleteMenu = function(Menu, callback) {
        var $menuListItem = $("<li>").text(Menu.nameOfMenu + " (" + "" +
Menu.meals.length + " блюд)"),
                $editMenu = $("<a>").attr("href", "/menus/" + Menu._id),
                $removeMenu = $("<a>").attr("href", "/menus/" + Menu._id);
        $editMenu.addClass("linkEdit");
        $removeMenu.addClass("linkRemove");

        $editMenu.text("Переименовать меню");
        $editMenu.on("click", function() {
                var newNameOfMenu = prompt("Введите новое
наименование для задачи", Menu.nameOfMenu);
                $.ajax({
                        "url": "/menus/" + Menu._id,
                        "type": "PUT",
                        "data": { "meals": Menu.meals, "nameOfMenu":
newNameOfMenu },
```

```
            }).done(function (responde) {
                callback();
            }).fail(function (err) {
                console.log("Произошла ошибка: " + err.val);
            });
            return false;
        });

        $menuListItem.append($editMenu);

        $removeMenu.text("Удалить меню");
        $removeMenu.on("click", function() {
            $.ajax({
                "url": "/menus/" + Menu._id,
                "type": "DELETE"
            }).done(function (responde) {
                callback();
            }).fail(function (err) {
                console.log("Произошла ошибка: " + err);
            });
            return false;
        });
        $menuListItem.append($removeMenu);

        return $menuListItem;
}

var main = function(userObjects) {
        "use strict";
        // создание пустого массива с вкладками
        var tabs = [];

        // добавляем вкладку Пользователи
        tabs.push({
            "name": "Пользователи",
```

```
                "content": function(callback) {
                        $.getJSON("/users.json", function (userObjects) {
                                var $content = $("<ul>");
                                $content.append($("<p>").text(""));
                                for (var i = userObjects.length-1; i>=0; i--) {
                                        // console.log("test");
                                        var $userListItem =
liaWithEditOrDeleteOnClick(userObjects[i], function() {
                                                $(".tabs a:first-child
span").trigger("click");
                                        });
                                        $content.append($userListItem);
                                        $content.append($("<p>").text(""));

                                }
                                callback(null, $content);
                        }).fail(function (jqXHR, textStatus, error) {
                                callback(error, null);
                        });
                }
        });

        // создаем вкладку Добавить пользователя
        tabs.push({
                "name": "Новый сотрудник",
                "content":function () {
                        $.get("/users.json", function (userObjects) {
                                // создание $content для Добавить
                                var $loginInput = $("<input>").addClass("login"),
                                        $passwordInput =
$("<input>").addClass("password"),
                                        $usernameInput =
$("<input>").addClass("username"),
                                        $userPhotoPathInput =
$("<input>").addClass("pathToImg"),
```

```
$button = $("<button>").text("Добавить"),
$contentForLogin = $("<ul>"),
$contentForPassword = $("<ul>"),
$contentForUsername = $("<ul>"),
$contentForImg = $("<ul>");

$loginInput.attr('placeholder','Логин');
$passwordInput.attr('placeholder','Пароль');
$usernameInput.attr('placeholder','ФИО');
$userPhotoPathInput.attr('placeholder','Путь
к изображению (папка клиента, начиная с /images/...)');

$contentForLogin.append($loginInput);
$contentForPassword.append($passwordInput);
$contentForUsername.append($usernameInput);
$contentForImg.append($userPhotoPathInput);
$("main .content").append($("<p>"));
$("main .content").append($contentForLogin);
$("main .content").append($contentForPassword);
$("main .content").append($contentForUsername);
$("main .content").append($contentForImg);
$("main .content").append($button);

function btnfunc() {
    var login = $loginInput.val();
    var password = $passwordInput.val();
    var username = $usernameInput.val();
    var pathToImg = $userPhotoPathInput.val();
    // создаем нового пользователя
    $.ajax({
        url: pathToImg,
        type:'HEAD',
        error: function()
        {
```

```
                                        console.log("изображения
нет!!!");
                                        alert('Изображения по
указанному пути не существует!')
                                    },
                                    success: function()
                                    {
                                        console.log("изображение
есть!!!");

                                        if (login !== null && login.trim()
!== "" &&
                                        password !== null &&
password.trim() !== "" &&
                                        username !== null &&
username.trim() !== "" &&
                                        pathToImg !== null &&
pathToImg.trim() !== "" ) {
                                            var newUser = {"login":
login, "password" : password, "username" : username, "pathToImg" : pathToImg};
                                            $.post("/users", newUser,
function(result) {
                                                console.log(result);

        userObjects.push(newUser); // отправляем на клиент
                                                $loginInput.val("");

        $passwordInput.val("");
                                                $(".tabs a:nth-
child(2) span").trigger("click");

        }).done(function(responde) {

        console.log(responde);
```

```
                                                    alert('Аккаунт
успешно создан!')
                                                }).fail(function(jqXHR,
textStatus, error) {
                                                    console.log(error);
                                                    if (jqXHR.status
=== 501) {
                                                     alert("Такой
пользователь уже существует!\nИзмените логин и повторите "
                                                            +
"попытку");
                                                    }
                                                    else {
alert("Произошла ошибка!\n"+jqXHR.status + " " + jqXHR.textStatus);    }
                                                });
                                            } else {
                                                alert('Введите все
данные!')
                                            }
                                        }
                                    });
                                }
                                $button.on("click", function() { btnfunc(); });
                                $('.tags').on('keydown',function(e){ if (e.which ===
13) { btnfunc(); } });
                            });
                        }
                });

                //вкладка "Все меню"
                tabs.push({
                    "name": "Все меню",
                    "content": function(callback) {
                        var $content = $("<ul>");
                        $.getJSON("/menus.json", function (menusObjects) {
```

```
                        var $text = $("<p>").text("");
                        $content.append($text);
                        for (var i = menusObjects.length-1; i>=0; i--) {
                            var $menuListItem = editOrDeleteMenu(menusObjects[i], function() {
                                $(".tabs a:nth-child(3) span").trigger("click");
                            });
                            $content.append($menuListItem);
                        }
                        callback(null, $content);
                    }).fail(function (jqXHR, textStatus, error) {
                        callback(error, null);
                    });
                }
            });

            //вклакда "Создать меню"
            tabs.push({
                "name": "Создать меню",
                "content":function (callback) {
                    $.getJSON("/menus.json", function(menuObjects) {
                        var $input = $("<input>").addClass(""),
                            $button = $("<button>").text("Добавить"),
                            $content = $("<ul>");
                        $content.append($input);
                        $input.attr('placeholder','Название для нового меню');
                        $("main .content").append($("<p>"));
                        $("main .content").append($content);
                        $("main .content").append($button);
                        function btnfunc() {
                            var nameOfMenu = $input.val();
                            if (nameOfMenu !== null && nameOfMenu.trim() !== "") {
```

```
                                        // создаем новое меню
                                        var newMenu = {"nameOfMenu" :
nameOfMenu};
                                        $.post("/menus", newMenu,
function(result) {
                                            console.log(result);
                                            menuObjects.push(newMenu); //
отправляем на клиент
                                            $input.val("");
                                            $(".tabs a:nth-child(4)
span").trigger("click");
                                        }).done(function(responde) {
                                            console.log(responde);
                                            alert('Меню успешно
создано!')
                                        }).fail(function(jqXHR, textStatus,
error) {
                                            console.log(error);
                                            if (jqXHR.status === 501) {
alert("Меню с таким названием уже существует!"); }
                                            else { alert("Произошла
ошибка!\n"+jqXHR.status + " " + jqXHR.textStatus); }
                                        });
                                    } else {
                                        alert('Введите все данные!')
                                    }
                                }
                                $button.on("click", function() { btnfunc(); });
                                $('.tags').on('keydown',function(e){ if (e.which ===
13) { btnfunc(); } });


                            }).fail(function (jqXHR, textStatus, error) {
                                console.log(error);
                                callback(error, null);
```

```
                });

            }
        });

        //вкладка "Все блюда"
        tabs.push({
            "name": "Все блюда",
            "content": function(callback) {
                $.getJSON("/meals.json", function (mealObjects) {
                    var $content = $("<ul>");
                    for (var i = mealObjects.length-1; i>=0; i--) {
                        var $mealListItem = addMealToMenu(mealObjects[i], function() {
                            $(".tabs a:nth-child(5) span").trigger("click");
                        });
                        $content.append($("<p>").text(""));
                        $content.append($mealListItem);
                    }
                    callback(null, $content);

                }).fail(function (jqXHR, textStatus, error) {
                    callback(error, null);
                });

            }
        });

        // создаем вкладку Добавить блюдо
        tabs.push({
            "name": "Добавить блюдо",
            "content":function () {
                $.get("/meals.json", function (mealObjects) {
                    // создание $content для Добавить
```

```
                var $input = $("<input>").addClass("description"),
                    $tagInput = $("<input>").addClass("tags"),
                    $priceInput = $("<input>").addClass("price"),
                    $pathToImg = $("<input>").addClass("pathToImg"),
                    $button = $("<button>").text("Добавить"),
                    $content = $("<ul>");

                $input.attr('placeholder','Название для нового блюда');
                $tagInput.attr('placeholder','Ингредиенты (через запятую)');
                $priceInput.attr('placeholder','Стоимость (в рублях)');
                $pathToImg.attr('placeholder','Путь к изображению (папка клиента, начиная с /images/...)');

                $content.append($input);
                $content.append($tagInput);
                $content.append($priceInput);
                $content.append($pathToImg);

                $("main .content").append($("<p>"));
                $("main .content").append($content);
                $("main .content").append($button);

                function btnfunc() {
                    var description = $input.val(),
                        tags = $tagInput.val().split(","),
                        price = $priceInput.val(),
                        pathToImg = $pathToImg.val();


                    $.ajax({
```

```
                                        url: pathToImg,
                                        type:'HEAD',
                                        error: function()
                                        {
                                            console.log("изображения
нет!!!");
                                            alert('Изображения по
указанному пути не существует!')
                                        },
                                        success: function()
                                        {
                                            console.log("изображение
есть!!!");

                                        if (description !== null &&
description.trim() !== "" &&
                                            $tagInput.val() !== null &&
$tagInput.val().trim() !== "" &&
                                            price !== null && price.trim()
!== "" &&
                                            pathToImg !== null &&
pathToImg.trim() !== "") {
                                            // создаем новый элемент
списка задач
                                            var newMeal =
{"description":description, "tags":tags, "status": 'Есть в меню', "price": price,
"pathToImg" : pathToImg};


                                            $.post("/meals", newMeal,
function(result) {
                                                console.log(result);

        mealObjects.push(newMeal); // отправляем на клиент
                                                $input.val("");
```

```
                                        $tagInput.val("");
                                        $priceInput.val("");
                                        $pathToImg.val("");
                                        $(".tabs a:nth-child(6) span").trigger("click");
                                    })
                                    .done(function(responde){
                                        console.log(responde);
                                        alert('Блюдо успешно создано!')
                                    })
                                    .fail(function(jqXHR, textStatus, error) {
                                        console.log(error);
                                        if (jqXHR.status === 501) { alert("Блюдо с таким названием уже существует!"); }
                                        else if (jqXHR.status === 500) { alert("Проверьте правильность введенных данных!"); }
                                        else { alert("Произошла ошибка!\n"+jqXHR.status + " " + jqXHR.textStatus); }
                                    });


                                } else {
                                    alert('Введите все данные!')
                                }
                            }
                        });

                        }
                        $button.on("click", function() { btnfunc(); });
                        $('.tags').on('keydown',function(e){
                            if (e.which === 13) { btnfunc(); }
                        });
                    });
```

```
            }
        });

        tabs.forEach(function (tab) {
            var $aElement = $("<a>").attr("href",""),
                $spanElement = $("<span>").text(tab.name);
            $aElement.append($spanElement);
            $("main .tabs").append($aElement);

            $spanElement.on("click", function () {
                $(".tabs a span").removeClass("active");
                $spanElement.addClass("active");
                $("main .content").empty();
                tab.content(function (err, $content) {
                    if (err !== null) { alert ("Возникла проблема при
обработке запроса: " + err); }
                    else { $("main .content").append($content); }
                });
                return false;
            });
        });

        $(".tabs a:first-child span").trigger("click");
    }

    $(document).ready(function() {
      $.getJSON("/users.json", function (userObjects) {
        var $place = $("<p>").text("Здравствуйте, админ!");
        $("header .username").append($place);
        main(userObjects);
      });
    });
```

ПРИЛОЖЕНИЕ 10. Файл user.js

```
var mongoose = require("mongoose");
// Это модель Mongoose для пользователей
var UserSchema = mongoose.Schema({
        username: String, //имя пользователя
        login: String, //Логин для входа
        password : String,
        pathToImg : String,
        usertype: Number //0 - сотрудник, 1 - админ
});
var User = mongoose.model("User", UserSchema);
module.exports = User;
```

ПРИЛОЖЕНИЕ 11. Файл meal.js

```
var mongoose = require("mongoose"),
        ObjectId = mongoose.Schema.Types.ObjectId;
        // mongoose.Schema.Types.Buffer
// Это модель mongoose для списка блюд меню
var MealSchema = mongoose.Schema({
        description: String,
        status : String,
        price : Number,
        tags: [ String ],
        pathToImg : String,
        owner : { type: ObjectId, ref: "Menu" }
});

var Meal = mongoose.model("Meal", MealSchema);
module.exports = Meal;
```

ПРИЛОЖЕНИЕ 12. Файл menu.js

```
var mongoose = require("mongoose"),
        ObjectId = mongoose.Schema.Types.ObjectId;

// Это модель mongoose для меню, состоящего из списка блюд
var MenuSchema = mongoose.Schema({
        nameOfMenu: String,
        meals : [ {type: ObjectId, ref: "Meal"} ]
});

var Menu = mongoose.model("Menu", MenuSchema);
module.exports = Menu;
```

```
// обратите внимание на то, что нужно перейти в папку,
// в которой находится каталог models
var User = require("../models/user.js"),
    Meal = require("../models/meal.js"),
    UsersController = {};

UsersController.index = function (req, res) {
    console.log('Вызвано действие: UsersController.index');
    User.find(function (err, users) {
        if (err !== null) { res.json(500, err); }
        else { res.status(200).json(users); }
    });
};

// Отобразить пользователя (админ и сотрудник)
UsersController.show = function(req, res) {
    console.log('Вызвано действие: отобразить пользователя ' +
req.params.login);
    User.find({'login': req.params.login}, function(err, result) {
        if (err) { console.log(err); }
        else if (result.length !== 0) {
            // открыть страницу админа или сотрудника?
            if (req.params.login == "admin") {
res.sendfile('./client/admin.html'); }
            else { res.sendfile('./client/worker.html'); }
        }
        else { res.sendStatus(404); }
    });
};

// Создать нового пользователя
UsersController.create = function(req, res) {
    console.log('Вызвано действие: создать пользователя ' +
req.body.login);
```

```
            var username = req.body.username;
            var login  = req.body.login;
            var password  = req.body.password;
            var img = req.body.pathToImg;
            console.log(username + ", " + login) + ", " + password;
            User.find({"login": login}, function (err, result) {
               if (err) { console.log(err); res.send(500, err); }
                    else if (result.length !== 0) {
                            res.status(501).send("Пользователь уже существует");
                       console.log(err);
                       console.log("Пользователь уже существует");
                    }
                     else {
                            // создание нового пользователя
                       var newUser = new User({ "login": login, "username": username,
"password": password, "pathToImg" : img});
                       newUser.save(function(err, result) {
                          console.log(err);
                          if (err !== null) { res.json(500, err);  }
                                  else { res.json(200, result); console.log(result);  }
                       });
                    }
            });
      };

      // Обновить логин существующего пользователя
      UsersController.update = function (req, res) {
            console.log("Вызвано действие: обновить пользователя");
            var login = req.params.login;
            console.log("Старое имя пользователя: " + username);
            var newLogin = {$set: {login: req.body.login}};
            console.log("Новый логин пользователя: " + newLogin);
            User.updateOne({"login": login}, newLogin, function (err,user) {
                  if (err !== null) { res.status(500).json(err); }
                  else {
```

```
                        if (user.n === 1 && user.nModified === 1 && user.ok
=== 1) {
                                console.log('Пользователь изменен');
                                res.status(200).json(user);
                        }
                        else { res.status(404); }
                }
        });
};

// Удалить существующего пользователя
UsersController.destroy = function (req, res) {
        console.log("Вызвано действие: удалить пользователя");
        var login = req.params.login;
        User.find({"login": login}, function (err, result) {
                if (err) { console.log(err); res.send(500, err); }
                else if (result.length !== 0) {
        console.log("Удаляем все блюда, созданные с 'owner': " +
result[0]._id);
                        // удалять блюда, созданные сотрудником
необязательно, это может сделать и админ
                        User.deleteOne({"login": login}, function (err, user) {
                                if (err !== null) { res.status(500).json(err); }
                                else {
                                        if (user.n === 1 && user.ok === 1 &&
user.deletedCount === 1) { res.status(200).json(user); }
                                        else { res.status(404).json({"status": 404}); }
                                }
                        });
        }
                else { res.status(404).send("Пользователь не существует");
console.log(err);    }
        });
}
```

```
module.exports = UsersController;
```

ПРИЛОЖЕНИЕ 14. Файл meals_controller.js

```
// обратите внимание на то, что нужно перейти в папку,
// в которой находится каталог models
var Meal = require("../models/meal.js"),
    User = require("../models/user.js"),
    MealsController = {};

MealsController.index = function (req, res) {
    console.log('Вызвано действие: MealsController.index');
    Meal.find(function (err, meals) {
        if (err !== null) { res.json(500, err); }
        else { res.status(200).json(meals); }
    });
};

MealsController.create = function (req, res) {

    console.log('Вызвано действие: создать блюдо ' +
req.body.description);
    var description = req.body.description;
        status = req.body.status,
        price = req.body.price,
        tags = req.body.tags,
        pathToImg = req.body.pathToImg;

    Meal.find({"description": description}, function (err, result) {
      if (err) { console.log(err); res.send(500, err); }
        else if (result.length !== 0) {
            res.status(501).send("Блюдо уже существует");
          console.log(err);
          console.log("Блюдо уже существует");
      }
        else {
            // создание нового блюда
            var newMeal = new Meal({
```

```
                    "description": description,
                    "status" : status,
                    "price" : price,
                    "tags": tags,
                    "pathToImg" : pathToImg,
                    "owner" : null
                });
                newMeal.save(function (err, result) {
                    console.log(result);
                    if (err !== null) { console.log(err); res.json(500, err);
}
                    else { res.status(200).json(result); }
                });
            }
        });


    };

    MealsController.show = function (req, res) {
        console.log('Вызвано действие: отобразить блюдо ' +
req.params.description);
        // это ID, который мы отправляем через URL
        var id = req.params.id;
        // находим блюдом с этим ID
        Meal.find({"_id":id}, function (err, Meal) {
            if (err) { console.log(err); }
            else if (err !== null) { res.status(500).json(err); }
            else {
                if (Meal.length !== 0) { res.status(200).json(Meal[0]); }
                else { res.sendStatus(404); }
            }
        });
    };
```

```
MealsController.destroy = function (req, res) {
    var id = req.params.id;
    Meal.deleteOne({"_id": id}, function (err, Meal) {
        if (err !== null) { res.status(500).json(err); }
        else {
            if (Meal.n === 1 && Meal.ok === 1 && Meal.deletedCount === 1) { res.status(200).json(Meal); }
            else { res.status(404).json({"status": 404}); }
        }
    });
}

MealsController.update = function (req, res) {
    var id = req.params.id;
    console.log('Вызвано действие: обновить блюдо ' + id);
    var newDescription = {$set: {description: req.body.description, status: req.body.status, owner: req.body.owner || null}};
    Meal.updateOne({"_id": id}, newDescription, function (err,Meal) {
        if (err !== null) { res.status(500).json(err); }
        else {
            if (Meal.n === 1 && Meal.nModified === 1 && Meal.ok === 1) { res.status(200).json(Meal); }
            else { res.status(404).json({"status": 404}); }
        }
    });
}

module.exports = MealsController;
```

ПРИЛОЖЕНИЕ 15. Файл menus_controller.js

```
var Menu = require("../models/menu.js"),
   Meal = require("../models/meal.js"),
       User = require("../models/user.js"),
       MenuController = {};


MenuController.index = function (req, res) {
       console.log('Вызвано действие: MenuController.index');
       Menu.find(function (err, menus) {
              if (err !== null) { res.json(500, err); }
              else { res.status(200).json(menus); }
       });
};

MenuController.create = function (req, res) {
       console.log('Вызвано действие: создать новое меню ' +
req.body.nameOfMenu);
       var nameOfMenu = req.body.nameOfMenu;
       var meals = req.body.meals
       Menu.find({"nameOfMenu": nameOfMenu}, function (err, result) {
          if (err) { console.log(err); res.send(500, err); }
              // else if (result.length !== 0) {
              //       res.status(501).send("Такое Меню уже существует");
          //     console.log(err);
          //     console.log("Меню уже существует");
          // }
              else {
                     // создание нового меню
                 var newMenu = new Menu({"nameOfMenu" : nameOfMenu,
"meals" : meals});
                 newMenu.save(function(err, result) {
                    console.log(err);
                    if (err !== null) { res.json(500, err);  }
                                  else { res.json(200, result); console.log(result);  }
```

```
            });
        }
    });
};

MenuController.show = function (req, res) {
    console.log('Вызвано действие: отобразить меню ' + req.params.id);
    // это ID, который мы отправляем через URL
    var id = req.params.id;
    Menu.find({"_id" : id}, function (err, result) {
        if (err) { console.log(err); }
        else if (err !== null) { res.status(500).json(err); }
        else {
            if (result.length > 0) { res.status(200).json(result[0]); }
            else { res.send(404); }
        }
    });
};



MenuController.destroy = function (req, res) {
    var id = req.params.id;
    console.log("Вызвано действие: удалить меню " + id);
    Menu.deleteOne({"_id" : id}, function (err, result) {
        if (err !== null) { res.status(500).json(err); }
        else {
            // console.log(result.n);
            // console.log(result.ok);
            // console.log(result.deletedCount);
            if (result.n === 1 && result.ok === 1 &&
result.deletedCount === 1) { res.status(200).json(result); }
            else { res.status(404).json({"status": 404}); }
        }
    });
```

```
    }

    MenuController.update = function (req, res) {
        var id = req.params.id;
        console.log('Вызвано действие: обновить меню ' + id);
        var newMenu = {$set: {nameOfMenu: req.body.nameOfMenu, meals:
req.body.meals || []}};
        Menu.updateOne({"_id": id}, newMenu, function (err,result) {
            if (err !== null) {
                console.log(err);
                res.status(500).json(err); }
            else {
                if (result.n === 1 && result.nModified === 1 && result.ok
=== 1) { res.status(200).json(result); }
                else { res.status(404).json({"status": 404}); }
            }
        });
    }

    module.exports = MenuController;
```