Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**Иркутский национальный исследовательский**

**технический университет**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

Допускаю к защите

|  |  |
| --- | --- |
| Руководитель |  |
|  | подпись |
|  | В.А. Харахинов |
|  | И.О. Фамилия |

|  |
| --- |
|  |
| Разработка клиент-серверного приложения для работы с  нереляционной базой данных |
| наименование темы |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

|  |
| --- |
| Объектно-ориентированные базы данных |

|  |
| --- |
| 1.05.00.00 ПЗ |
| обозначение документа |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИСМб-19-1 |  |  |  | И.А. Малиновцев |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Нормоконтроль |  |  |  |  |  | В.А. Харахинов |
|  |  |  |  | подпись |  | И.О. Фамилия |
| Курсовой проект защищен с оценкой | | | |  | | |

Иркутск 2022 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

ЗАДАНИЕ

на курсовое проектирование

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| По курсу | Объектно-ориентированные базы данных | | | | | | | | | | | | | | | | | |
| Студенту | Малиновцеву И.А. | | | | | | | | | | | | | | | | | |
|  | (фамилия, инициалы) | | | | | | | | | | | | | | | | | |
| Тема проекта | | Разработка клиент-серверного приложения для работы с нереляционной базой данных | | | | | | | | | | | | | | | | |
| Исходные данные | | | Вариант 5 | | | | | | | | | | | | | | | |
| Рекомендуемая литература | | | | | | | |  | | | | | | | | | | |
| 1. Сосинская С.С. «Разработка клиента по технологии MVC»: Методические указания по выполнению курсового проекта» Электронный каталог кафедры вычислительной техники. | | | | | | | | | | | | | | | | | | |
| Графическая часть на | | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ листах. | | | | | | | | | | | | | | |
| Дата выдачи задания | | | | « | 12 | » | ноября | | | | | 2022 г. | | | | |
|  | | | | | | | | | | | | | | | | | | |
| Задание получил | | | | | | | | |  | | | |  | | И.А. Малиновцев | | | |
|  | | | | | | | | | подпись | | | |  | | И.О. Фамилия | | | |
|  | | | | | | | | | |  |  | | |  | |  | |  |
| Дата представления проекта руководителю | | | | | | | | | | « | 24 | | | » | | декабря | | 2022 г. |
|  | | | | | | | | | | | | | | | | | | |
| Руководитель курсового проектирования | | | | | | | | |  | | | |  | | В.А. Харахинов | | | |
|  | | | | | | | | | подпись | | | |  | | И.О. Фамилия | | | |

**Содержание**

[Введение 4](#_Toc121692660)

[Индивидуальный вариант 5](#_Toc121692661)

[1. Структура базы данных 6](#_Toc121692662)

[1.1 Концептуальная модель данных 6](#_Toc121692663)

[1.2 Физическая модель данных 6](#_Toc121692664)

[2. Проектирование информационной системы 8](#_Toc121692665)

[3. Контроллеры 9](#_Toc121692666)

[4. Представления 10](#_Toc121692667)

[4.1 Index 10](#_Toc121692668)

[4.2 Room 10](#_Toc121692669)

[4.3 Flat 11](#_Toc121692670)

[5 Сценарии 13](#_Toc121692671)

[5.1 room.js 13](#_Toc121692672)

[5.2 flat.js 15](#_Toc121692673)

[6. WebApi 21](#_Toc121692674)

[7. Таблица тестов 22](#_Toc121692675)

[8. Результаты тестирования 24](#_Toc121692676)

[Заключение 28](#_Toc121692677)

[Список использованных источников 29](#_Toc121692678)

Введение

Целью курсового проекта является знакомство с фреймворком ASP NET MVC Framework для создания веб-приложения по используемому паттерну Model-view-controller (MVC) и использования NoSQL базы данных MongoDB.

В качестве предметной области был взят жилой квартал, включающий следующие сущности:

1. Жильцы;
2. Комнаты;
3. Квартиры;
4. Здания.

Индивидуальный вариант

Имеются жильцы (имя, фамилия, дата рождения) и комнаты (тип комнаты, площадь). Из этих объектов составляются квартиры (жильцы, комнаты), входящие в здания (квартиры, адрес).

Выходные документы:

выдать список квартир, в которых проживает житель;

выдать список зданий, в котором есть квартиры, соответствующие заданными типам комнат.

1. Структура базы данных

* 1. Концептуальная модель данных

Концептуальная модель данных содержит 4 сущности и 3 связи. Первая модель описывает отношение между сущностями Комнаты и Квартиры. Они являются обязательными и имеют связь «многие ко многим». Вторая модель описывает отношение между Жильцами и Квартирами, имеющую связь «многие ко многим». Жильцы обязательно должны быть прописаны в квартире, но квартира не обязательно должна иметь жильцов. Третья модель описывает отношения между Квартирами и Зданиями. Она имеет связь «один ко многим» и является обязательной.

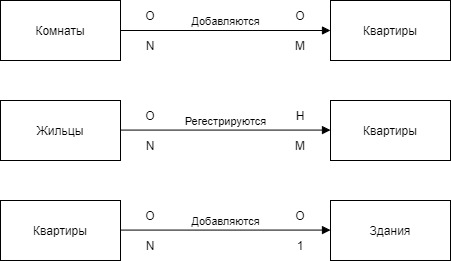


Рисунок 1 – Концептуальная модель данных

1.2 Физическая модель данных

В результате применения инструмента draw.io в отношении разработки логической модели базы данных, основываясь на концептуальной модели (см. рис. 1) была разработана физическая модель, представленная на рисунке 2.

Таблица 1 – Структура модели «Комнаты»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Null | Примечание |
| \_id | ObjectId | Нет | PK, Index |
| roomType | String | Нет |  |
| area | Int32 | Нет |  |

Таблица 2 – Структура модели «Жильцы»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Null | Примечание |
| \_id | ObjectId | Нет | PK, Index |
| name | String | Нет |  |
| lastname | String | Нет |  |
| birthDate | Date | Нет |  |

Таблица 3 – Структура модели «Квартиры»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Null | Примечание |
| \_id | ObjectId | Нет | PK, Index |
| Rooms | Array< ObjectId > | Нет | FK |
| Residents | Array< ObjectId > | Да | FK |

Таблица 4 – Структура модели «Квартиры»

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Тип | Null | Примечание |
| \_id | ObjectId | Нет | PK, Index |
| Flats | Array< ObjectId > | Нет | FK |
| Address | String | Нет |  |

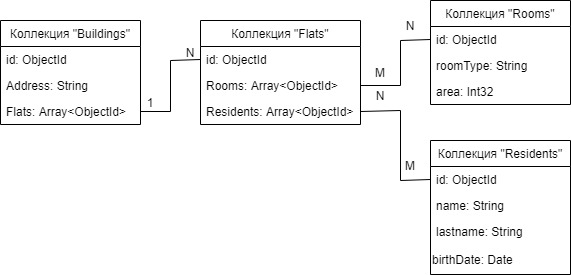


Рисунок 2 – Физическая модель данных «buildDB»

2. Проектирование информационной системы

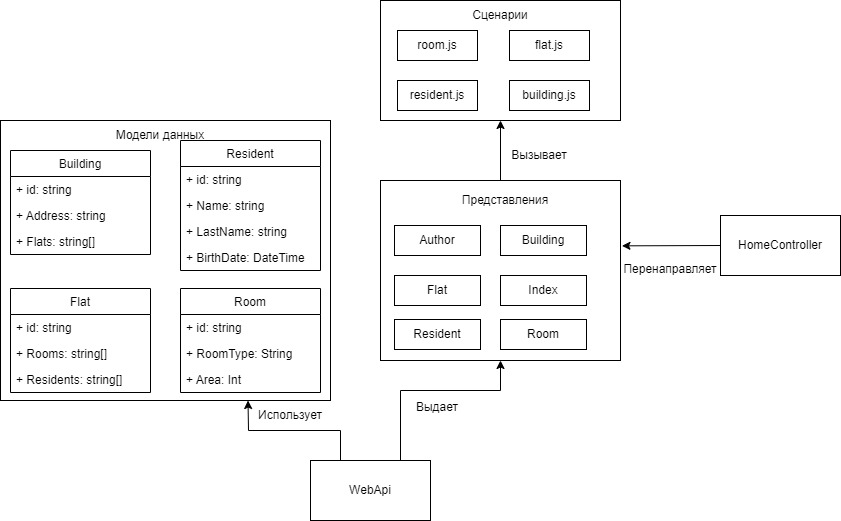


Рисунок 3 – Модель ИС

3. Контроллеры

В программе реализован 1 контроллер: HomeController, который занимается маршрутизацией сайта. Листинг:

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Resident()

{

return View();

}

public IActionResult Room()

{

return View();

}

public IActionResult Flat()

{

return View();

}

public IActionResult Building()

{

return View();

}

public IActionResult Author()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

4. Представления

Для реализации ИС было создано 6 представлений (см. рис. 3). Представление Index содержит базовую информацию о работе. Представление Author содержит информация об авторе. Room позволяет взаимодействовать с моделью Room и выполнять выдачу первого документа. Resident позволяет взаимодействовать с моделью Resident. Flat позволяет взаимодействовать с моделью Flat. Building позволяет взаимодействовать с моделью Building и выполнять выдачу второго документа.

4.1 Index

@{

ViewData["Title"] = "Главная страница";

}

<div class="text-center">

<h1 class="display-4">Добро пожаловать!</h1>

<h2 class ="display-4">Курсованя работа по дисциплине Объектно-Ориентированные Базы Данных</h2>

</div>

<div>

<p>Вариант 5.</p>

<p>Предметная область: Жилой квартал</p>

<p>Модель данных включает в себя:</p>

<ul>

<li>Жильцов</li>

<li>Комнаты</li>

<li>Квартиры</li>

<li>Здания</li>

</ul>

</div>

4.2 Room

@{

ViewData["Title"] = "Комнаты";

}

<h1>@ViewData["Title"]</h1>

<div>

<h2>Список комнат</h2>

<div>

<input class="form-control" type="hidden" id="roomId" />

<p>

Тип комнаты:<br />

<select id="roomType" class="form-select">

<option>Зал</option>

<option>Ванная</option>

<option>Кухня</option>

<option>Комната</option>

<option>Прихожая</option>

</select>

</p>

<p>

Площадь:<br />

<input class="form-control" id="roomArea" type="number" min="10"

max ="100" value="10" step="10"/>

</p>

<p>

<button class="btn btn-primary" id="saveBtn" onclick="addRoom()">Сохранить</button>

<button class="btn btn-warning" id="resetBtn" onclick="reset()">Сбросить</button>

</p>

</div>

<table class="table table-striped">

<thead>

<tr>

<th scope="col">Тип комнаты</th>

<th scope="col">Площадь</th>

<th></th>

</tr>

</thead>

<tbody>

</tbody>

</table>

<script>

</script>

</div>

<script src="~/js/room.js" asp-append-version="true"></script>

4.3 Flat

@{

ViewData["Title"] = "Квартиры";

}

<h1>@ViewData["Title"]</h1>

<div style="display: flex; justify-content: space-evenly">

<div >

<input id="flatId" type="hidden"/>

<p >

Комнаты:<br />

<select id="cmbRoom" class="form-select" onchange="cmbRoomChange()">

</select>

<input id="roomInfo" class="form-control" disabled="disabled" />

<button class="btn btn-primary" id="btnRoomAdd" onclick="AddRoom()">Добавить</button>

<table class="table table-striped" id="roomTable">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Тип комнаты</th>

<th scope="col">Площадь</th>

<th></th>

</tr>

</thead>

<tbody id="roombody">

</tbody>

</table>

</p>

</div>

<div>

<p>

Жильцы:<br />

<select id="cmbResident" class="form-select" onchange="cmbResidentChange()">

</select>

<input id="residentInfo" class="form-control" disabled="disabled" />

</p>

<button class="btn btn-primary" id="btnResidentAdd" onclick="AddResident()">Добавить</button>

<table class="table table-striped" id="residentTable">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Имя</th>

<th scope="col">Фамилия</th>

<th scope="col">Дата рождения</th>

<th></th>

</tr>

</thead>

<tbody id="residentbody">

</tbody>

</table>

</div>

</div>

<div>

<p>Список квартир </p>

<button class="btn btn-primary" id="btnFlatAdd" onclick="AddFlat()">Сохранить квартиру</button>

<button class="btn btn-warning" id="resetBtn" onclick="Reset()">Сбросить</button>

<table class="table" id="mainTable">

<thead>

<tr>

<th>id</th>

<th></th>

</tr>

</thead>

<tbody>

</tbody>

</table>

</div>

<script src="https://yastatic.net/jquery/3.3.1/jquery.min.js"></script>

<script src="~/js/flat.js" asp-append-version="true"></script>

5 Сценарии

Для взаимодействия с представлениями были седаны сценарии на языке JavaScript. Каждый сценарий назван также, как и преставление.

5.1 room.js

async function getRooms() {

const response = await fetch("/api/Rooms", {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const rooms = await response.json();

console.log(rooms);

const rows = document.querySelector("tbody");

rooms.forEach(room => rows.append(row(room)));

}

}

async function getRoom(id) {

const response = await fetch(`/api/Rooms/${id}`, {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const room = await response.json();

document.getElementById("roomId").value = room.id;

document.getElementById("roomType").value = room.roomType;

document.getElementById("roomArea").value = room.area;

}

else {

const error = await response.json();

}

}

async function createRoom(roomRoomType, roomArea) {

const response = await fetch("/api/Rooms", {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify({

roomType: roomRoomType,

area: parseInt(roomArea, 10)

})

});

if (response.ok == true) {

const room = await response.json();

document.querySelector("tbody").append(row(room));

}

else {

const error = await response.json();

console.log(error.message);

}

}

async function editRoom(roomId, roomRoomType, roomArea) {

const response = await fetch("/api/Rooms", {

method: "PUT",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify({

id: roomId,

roomType: roomRoomType,

area: roomArea

})

});

if (response.ok === true) {

const room = await response.json();

document.querySelector(`tr[data-rowid='${room.id}']`).replaceWith(row(room));

}

else {

const error = await response.json();

console.log(error.message);

}

}

async function deleteRoom(id) {

var flats = await GetFlats(id);

if (flats.length > 0) {

alert("Эта комната находится в квартирах");

return;

}

const response = await fetch(`/api/Rooms/${id}`, {

method: "DELETE",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const room = await response.json();

document.querySelector(`tr[data-rowid='${room.id}']`).remove();

}

else {

const error = await response.json();

console.log(error.message);

}

}

async function GetFlats(roomId) {

const response = await fetch(`/api/Flats/Rooms/${roomId}`, {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const flats = await response.json();

return flats;

}

}

function reset() {

document.getElementById("roomId").value = "";

}

function row(room) {

const tr = document.createElement("tr");

tr.setAttribute("data-rowid", room.id);

const typeTd = document.createElement("td");

typeTd.append(room.roomType);

tr.append(typeTd);

const areaTb = document.createElement("td");

areaTb.append(room.area);

tr.append(areaTb);

const linksTd = document.createElement("td");

const editLink = document.createElement("button");

editLink.append("Изменить");

editLink.className = "btn btn-info";

editLink.addEventListener("click", async () => await getRoom(room.id));

linksTd.append(editLink);

const removeLink = document.createElement("button");

removeLink.append("Удалить");

removeLink.className = "btn btn-danger";

removeLink.addEventListener("click", async () => await deleteRoom(room.id));

linksTd.append(removeLink);

tr.appendChild(linksTd);

return tr;

}

async function addRoom() {

const id = document.getElementById("roomId").value;

const type = document.getElementById("roomType").value;

area = document.getElementById("roomArea").value;

if (type.length == 0) {

alert("Вы не выбрали тип");

return;

}

if (area.length == 0) {

alert("Вы не ввели площадь");

return;

}

if (isNaN(area)) {

alert("Введите числовое значение площади");

return;

}

area = parseInt(area, 10);

if (area < 10 || area > 100) {

alert("Введите корретное значение площади");

return;

}

if (id === "")

await createRoom(type, area);

else

await editRoom(id, type, area);

reset();

}

getRooms();

5.2 flat.js

async function GetFlats() {

const response = await fetch("/api/Flats", {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const flats = await response.json();

console.log(flats);

const rows = document.querySelectorAll("tbody")[2];

flats.forEach(flat => rows.append(FlatRow(flat)));

await GetFlat(flats[0].id);

var table = document.getElementById("mainTable");

var row = table.rows;

row[1].bgColor = "green";

var idField = document.getElementById("flatId");

idField.value = flats[0].id;

}

}

async function GetFlat(id) {

const response = await fetch(`/api/Flats/${id}`, {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const flat = await response.json();

const rooms = [];

for (var i = 0; i < flat.rooms.length; i++) {

rooms.push(await GetRoom(flat.rooms[i]));

}

const roomRows = document.querySelectorAll("tbody")[0];

const residentRows = document.querySelectorAll("tbody")[1];

const residents = [];

for (var i = 0; i < flat.residents.length; i++) {

residents.push(await GetResident(flat.residents[i]));

}

rooms.forEach(room => roomRows.append(roomRow(room)));

residents.forEach(resident => residentRows.append(rowResident(resident)));

}

else {

// если произошла ошибка, получаем сообщение об ошибке

const error = await response.json();

console.log(error.message); // и выводим его на консоль

}

}

async function setRoomsCmb() {

const response = await fetch("/api/Rooms", {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const rooms = await response.json();

var roomInfo = document.getElementById("roomInfo");

if (rooms.length < 1) {

roomInfo.value = "Не добавлены комнаты";

return;

}

var cmbRoom = document.getElementById("cmbRoom");

for (var i = 0; i < rooms.length; i++) {

var t = document.createElement("option");

t.value = rooms[i].id;

t.textContent = rooms[i].roomType;

cmbRoom.appendChild(t);

}

roomInfo.value = `Тип комнаты: ${rooms[0].roomType} Площадь: ${rooms[0].area}`;

}

}

async function GetRoom(id) {

const response = await fetch(`/api/Rooms/${id}`, {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const room = await response.json();

return room;

}

}

async function cmbRoomChange() {

var e = document.getElementById("cmbRoom");

var id = e.options[e.selectedIndex].value;

var room = await GetRoom(id);

var roomInfo = document.getElementById("roomInfo");

roomInfo.value = `Тип комнаты: ${room.roomType} Площадь: ${room.area}`;

}

async function setResidentsCmb() {

const response = await fetch("/api/Residents", {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const residents = await response.json();

var roomInfo = document.getElementById("residentInfo");

if (residents.length < 1) {

roomInfo.value = "Не добавлены жители";

return;

}

var cmbResident = document.getElementById("cmbResident");

for (var i = 0; i < residents.length; i++) {

var t = document.createElement("option");

t.value = residents[i].id;

t.textContent = residents[i].lastName;

cmbResident.appendChild(t);

}

roomInfo.value = `Имя: ${residents[0].name} | Фамилия: ${residents[0].lastName}

| Дата рождения: ${residents[0].birthDate}`;

}

}

async function GetResident(id) {

const response = await fetch(`/api/Residents/${id}`, {

method: "GET",

headers: { "Accept": "application/json" }

});

const resident = await response.json();

return resident;

}

async function cmbResidentChange() {

var e = document.getElementById("cmbResident");

var id = e.options[e.selectedIndex].value;

var roomInfo = document.getElementById("residentInfo");

var resident = await GetResident(id);

roomInfo.value = `Имя: ${resident.name} | Фамилия: ${resident.lastName}

| Дата рождения: ${resident.birthDate}`;

}

async function AddRoom() {

var e = document.getElementById("cmbRoom");

var id = e.options[e.selectedIndex].value;

var room = await GetRoom(id);

document.querySelectorAll("tbody")[0].append(roomRow(room));

}

function roomRow(room) {

const tr = document.createElement("tr");

tr.setAttribute("data-rowid", room.id);

const idTd = document.createElement("td");

idTd.append(room.id);

tr.append(idTd);

const typeTd = document.createElement("td");

typeTd.append(room.roomType);

tr.append(typeTd);

const areaTb = document.createElement("td");

areaTb.append(room.area);

tr.append(areaTb);

const linksTd = document.createElement("td");

const removeLink = document.createElement("button");

removeLink.append("Удалить");

removeLink.className = "btn btn-danger";

removeLink.addEventListener("click", async () => await DeleteItem(room.id));

linksTd.append(removeLink);

tr.appendChild(linksTd);

return tr;

}

async function AddResident() {

var e = document.getElementById("cmbResident");

var id = e.options[e.selectedIndex].value;

var roomResidents = document.getElementsByTagName("tbody")[1].rows;

for (var i = 0; i < roomResidents.length; i++) {

var td = roomResidents[i].getElementsByTagName("td");

if (id === td[0].innerText) {

alert("Этот житель уже прописан в этой квартире");

return;

}

}

var resident = await GetResident(id);

document.querySelectorAll("tbody")[1].append(rowResident(resident));

}

function rowResident(resident) {

const tr = document.createElement("tr");

tr.setAttribute("data-rowid", resident.id);

const idId = document.createElement("td");

idId.append(resident.id);

tr.append(idId);

const nameTd = document.createElement("td");

nameTd.append(resident.name);

tr.append(nameTd);

const lastNameTd = document.createElement("td");

lastNameTd.append(resident.lastName);

tr.append(lastNameTd);

const bdTb = document.createElement("td");

bdTb.append(resident.birthDate);

tr.append(bdTb);

const linksTd = document.createElement("td");

const removeLink = document.createElement("button");

removeLink.append("Удалить");

removeLink.className = "btn btn-danger";

removeLink.addEventListener("click", async () => await DeleteItem(resident.id));

linksTd.append(removeLink);

tr.appendChild(linksTd);

return tr;

}

async function DeleteItem(id) {

document.querySelector(`tr[data-rowid='${id}']`).remove();

}

async function click(flat, tr) {

Reset();

var idField = document.getElementById("flatId");

idField.value = flat.id;

var table = document.getElementById("mainTable");

var row = table.rows;

for (var i = 0; i < row.length; i++) {

row[i].bgColor = "white";

}

var id = flat.id;

tr.bgColor = "green";

await GetFlat(id);

}

function FlatRow(flat) {

const tr = document.createElement("tr");

tr.setAttribute("data-rowid", flat.id);

const idId = document.createElement("td");

idId.append(flat.id);

const linksTd = document.createElement("td");

tr.append(idId);

const removeLink = document.createElement("button");

removeLink.append("Удалить");

removeLink.className = "btn btn-danger";

removeLink.addEventListener("click", async () => await DeleteFlat(flat.id));

linksTd.append(removeLink);

tr.appendChild(linksTd);

tr.addEventListener("click", async () => await click(flat, tr));

return tr;

}

async function DeleteFlat(id) {

var bs = await GetBuildings(id);

if (bs.length > 0) {

alert("Эта квартира находится в зданиях");

return;

}

const response = await fetch(`/api/Flats/${id}`, {

method: "DELETE",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const flat = await response.json();

document.querySelector(`tr[data-rowid='${flat.id}']`).remove();

}

else {

const error = await response.json();

console.log(error.message);

}

}

async function AddFlat() {

let rooms = [];

var roomRows = document.getElementsByTagName("tbody")[0].rows;

for (var i = 0; i < roomRows.length; i++) {

var td = roomRows[i].getElementsByTagName("td");

rooms.push(td[0].innerText)

}

let residents = [];

var roomResidents = document.getElementsByTagName("tbody")[1].rows;

for (var i = 0; i < roomResidents.length; i++) {

var td = roomResidents[i].getElementsByTagName("td");

residents.push(td[0].innerText)

}

if (rooms.length == 0) {

alert("Вы не выбрали комнаты");

return;

}

var idField = document.getElementById("flatId");

if (idField.value === "") {

await CreateFlat(rooms, residents)

}

else {

await EditFlat(idField.value, rooms, residents)

}

Reset();

}

async function EditFlat(id, rooms, residents) {

const response = await fetch("/api/Flats", {

method: "PUT",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify({

id: id,

rooms: rooms,

residents: residents

})

});

if (response.ok === true) {

const flat = await response.json();

document.querySelector(`tr[data-rowid='${flat.id}']`).replaceWith(FlatRow(flat));

}

else {

const error = await response.json();

console.log(error.message);

}

}

async function CreateFlat(rooms, residents) {

const response = await fetch("/api/Flats", {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify({

rooms: rooms,

residents: residents

})

});

if (response.ok == true) {

const flat = await response.json();

document.querySelectorAll("tbody")[2].append(FlatRow(flat));

}

else {

const error = await response.json();

console.log(error.message);

}

}

function Reset() {

$("#roombody").empty();

$("#residentbody").empty();

var table = document.getElementById("mainTable");

var row = table.rows;

for (var i = 0; i < row.length; i++) {

row[i].bgColor = "white";

}

var idField = document.getElementById("flatId");

idField.value = "";

}

async function GetBuildings(buildId) {

const response = await fetch(`/api/Buildings/Flats/${buildId}`, {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const buildings = await response.json();

return buildings;

}

}

GetFlats();

setRoomsCmb();

setResidentsCmb();

6. WebApi

Для взаимодействия с MongoDB было создано Api средствами ASP.Net MVC. Оно представляет из себя набор из 4 методов, реализующий CRUD. Частичный листинг:

private static void MapResident(WebApplication app, IMongoDatabase db, string collectionName)

{

//Добавим конечную точку, которая обрабатывает GET по пути

app.MapGet($"/api/{collectionName}", () =>

db.GetCollection<Resident>(collectionName).Find("{}").ToListAsync());

//Если обратиться по id, то получаем другую конечную точку

app.MapGet("/api/"+collectionName+"/{id}", async (string id) =>

{

var resident = await db.GetCollection<Resident> (collectionName)

.Find(r => r.Id == id)

.FirstOrDefaultAsync();

// если не найден, отправляем статусный код и сообщение об ошибке

if (resident == null) return Results.NotFound(new { message = "Пользователь не найден" });

// если пользователь найден, отправляем его

return Results.Json(resident);

});

//Если получаем Delete по этому адресу, вызывается это

app.MapDelete($"/api/"+collectionName+"/{id}", async (string id) =>

{

var resident = await db.GetCollection<Resident>(collectionName).FindOneAndDeleteAsync(r => r.Id == id);

// если не найден, отправляем статусный код и сообщение об ошибке

if (resident is null) return Results.NotFound(new { message = "Пользователь не найден" });

return Results.Json(resident);

});

//При получении запроса с методом POST, идем сюда

app.MapPost($"/api/{collectionName}", async (Resident resident) => {

// добавляем пользователя в список

await db.GetCollection<Resident>(collectionName).InsertOneAsync(resident);

return resident;

});

//Если приходит put по этому адресу, идем сюда

app.MapPut($"/api/{collectionName}", async (Resident residentData) => {

var resident = await db.GetCollection<Resident>(collectionName)

.FindOneAndReplaceAsync(p => p.Id == residentData.Id, residentData, new() { ReturnDocument = ReturnDocument.After });

if (resident == null)

return Results.NotFound(new { message = "Пользователь не найден" });

return Results.Json(resident);

});

}

7. Таблица тестов

Результаты тестирования работоспособности реализованного приложения приведены в таблице 4.

Таблица 4 – Тесты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Название таблицы | Название  теста | Входные  данные | Выходные  данные |
| 1 | Жильцы | Добавление | Имя, Фамилия, 08.12.2022 | Обновление таблицы |
| 2 | Редактирование | Новое имя, Новая фамилия, 06.12.2022 | Обновление таблицы |
| 3 | Удаление | Удаление записи №3 | Обновление таблицы |
| 4 | Комнаты | Добавление | Зал. 50 | Обновление таблицы |
| 5 | Редактирование | Кухня. 40 | Обновление таблицы |
| 6 | Удаление | Удаление записи №6 | Обновление таблицы |
| 7 | Квартиры | Добавление | Зал, Кухня, ыфвфыв | Обновление таблицы |
| 8 | Редактирование | Зал, Кухня, asdasd | Обновление таблицы |
| 9 | Удаление | Удаление записи №3 | Обновление таблицы |
| 10 | Здания | Добавление | 638abf0b72521aed81accd65, Адрес | Обновление таблицы |
| 11 | Редактирование | 638abf0b72521aed81accd65, Новый адрес | Обновление таблицы |
| 12 | Удаление | Удаление записи №3 | Обновление таблицы |
| 13 | Первый запрос | Выполнение запроса | 6388816ee72cd74701899f92 | Документ |
| 14 | Второй запрос | Выполнение запроса | Зал, Кухня | Документ |

8. Результаты тестирования

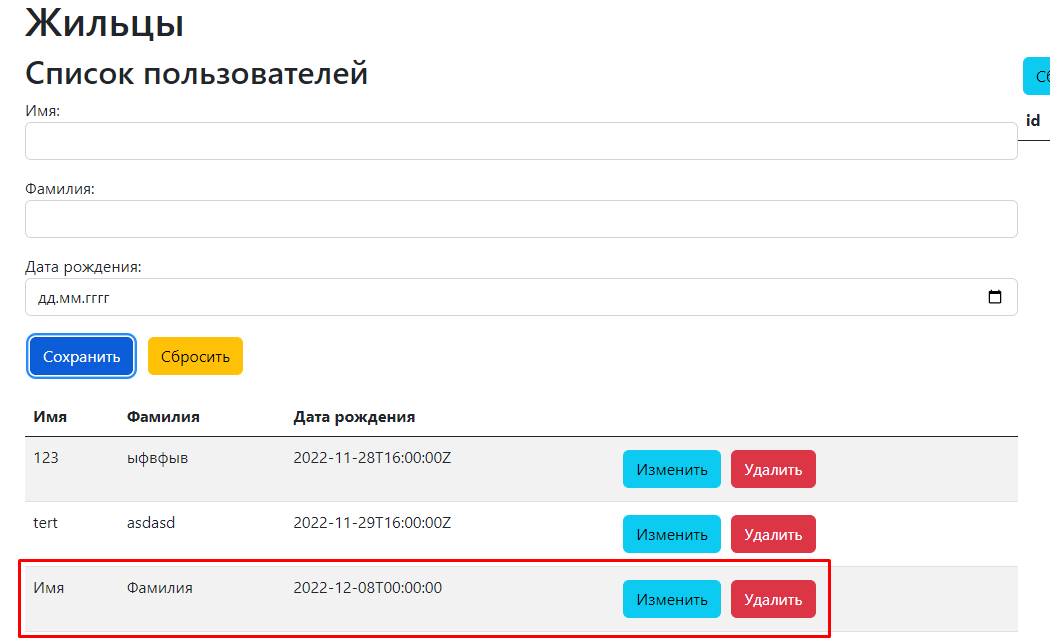


Рисунок 4 – Результат теста №1

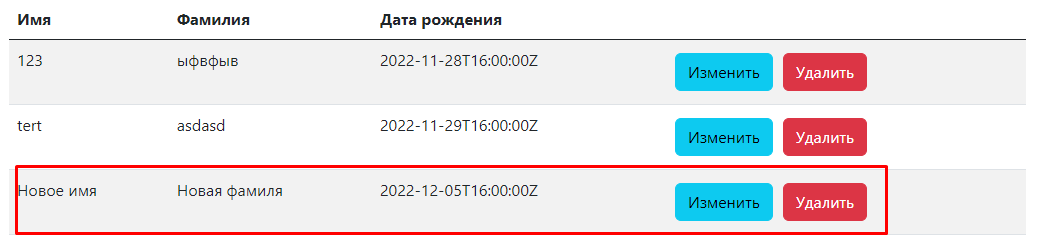


Рисунок 5 – Результат теста №2

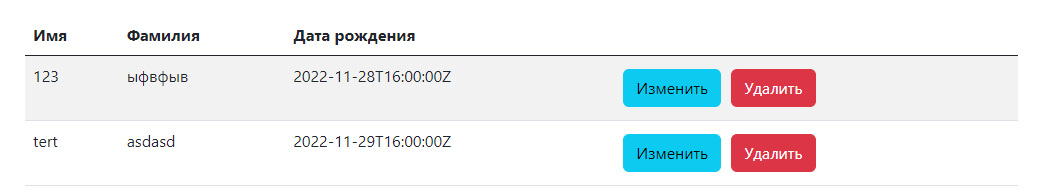


Рисунок 6 – Результат теста №3

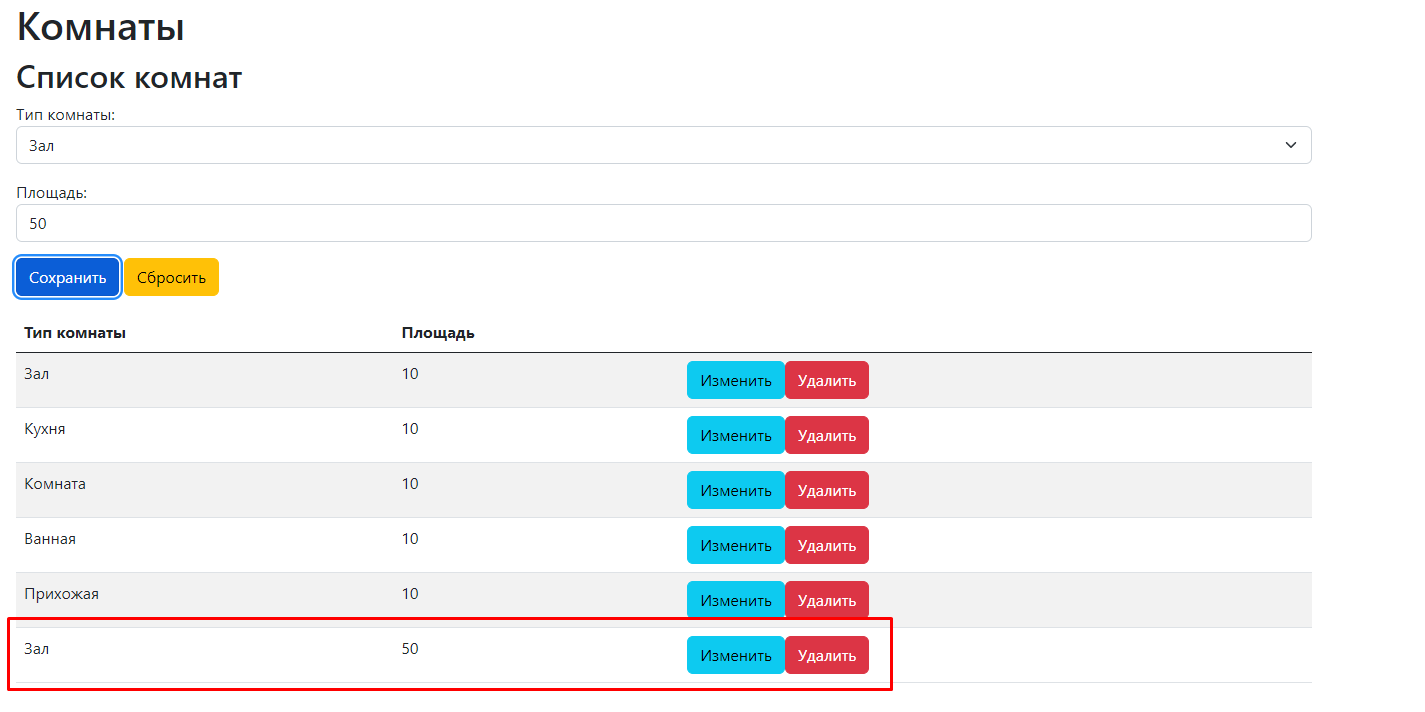


Рисунок 7 – Результат теста №4



Рисунок 8 – Результат теста №5

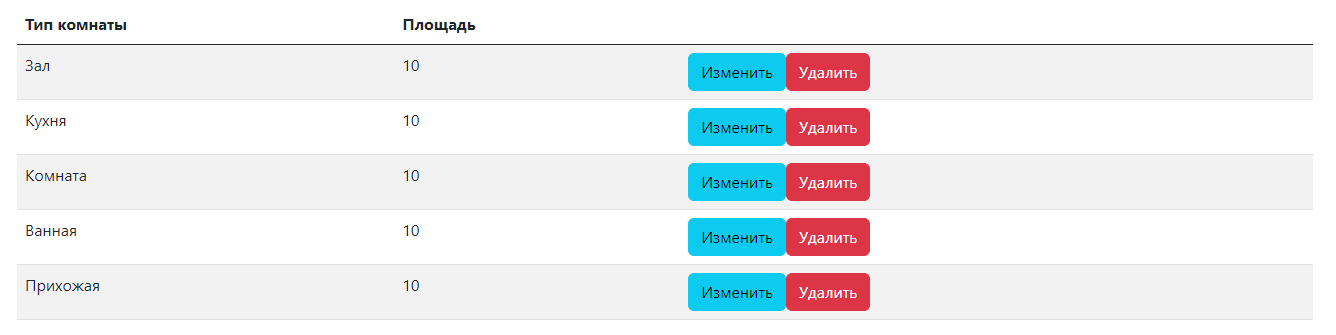


Рисунок 8 – Результат теста №6

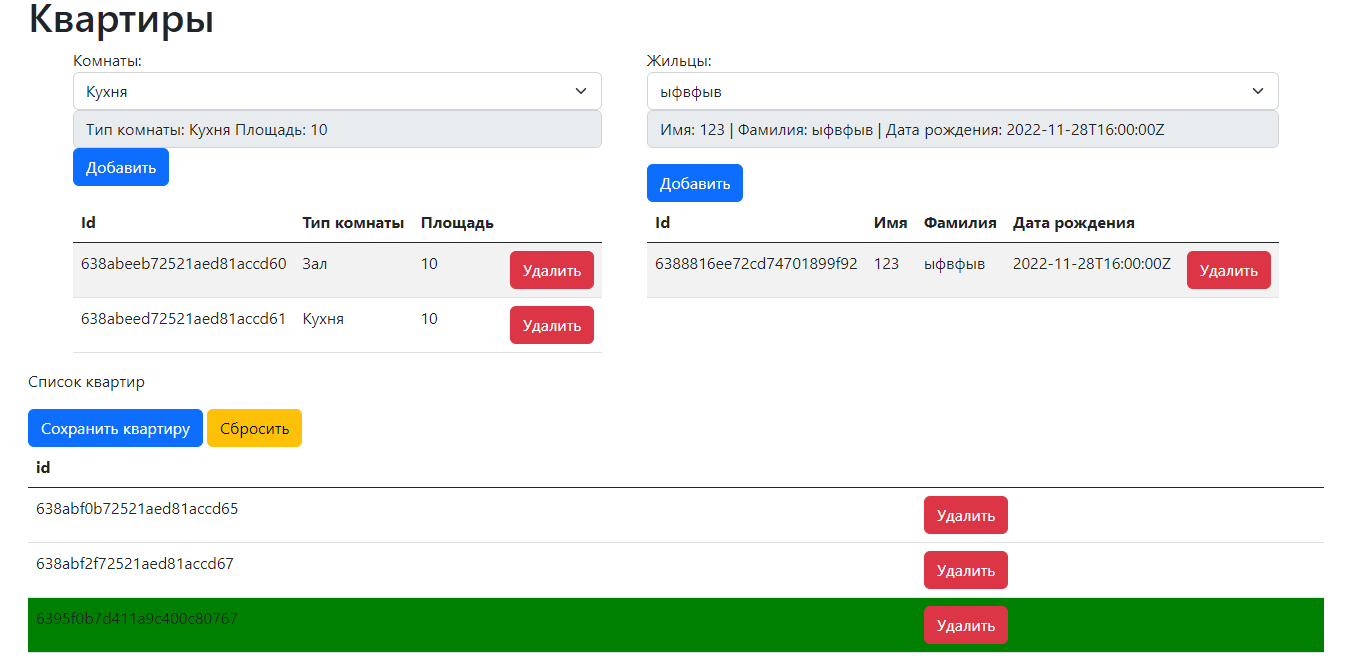


Рисунок 9 – Результат теста №7

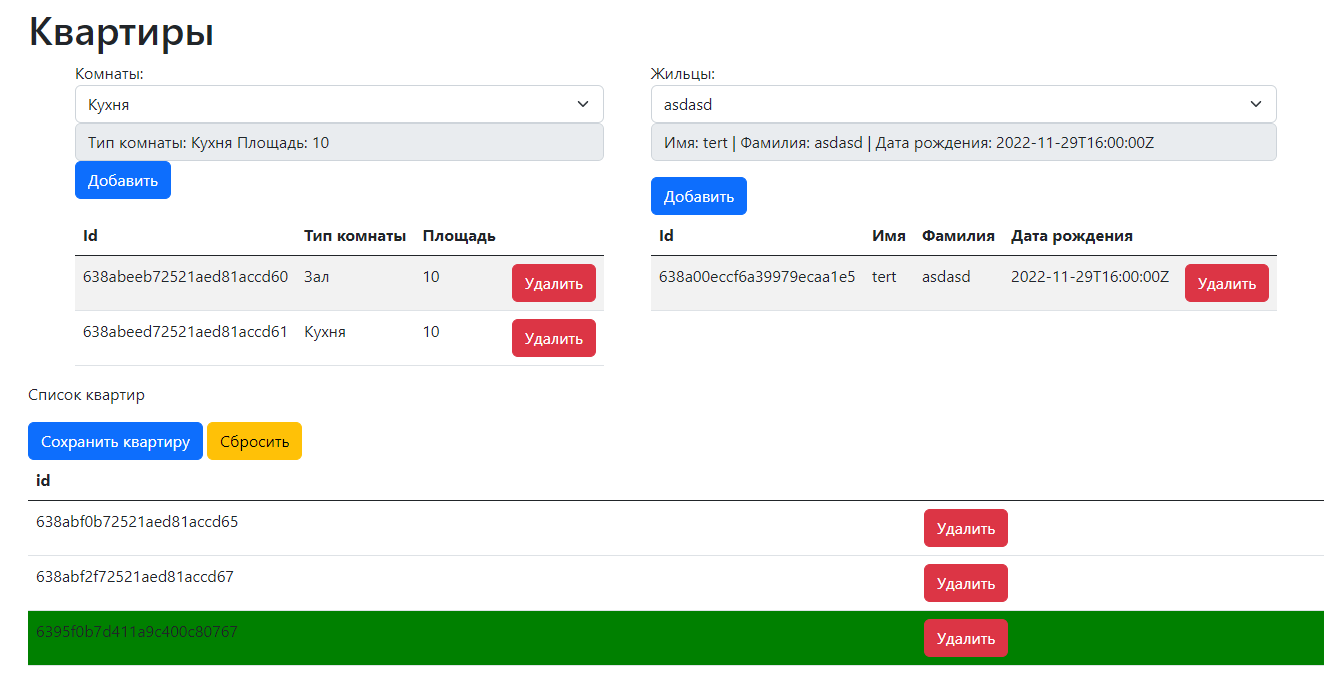


Рисунок 10 – Результат теста №8

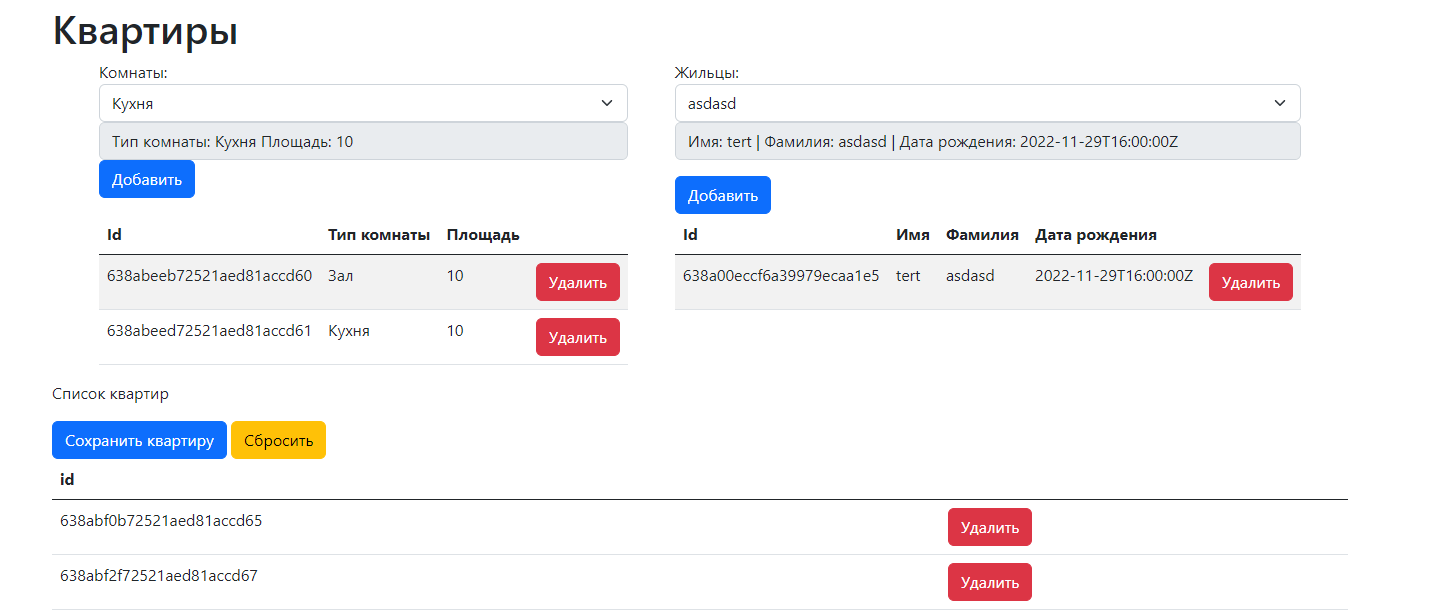


Рисунок 11 – Результат теста №9

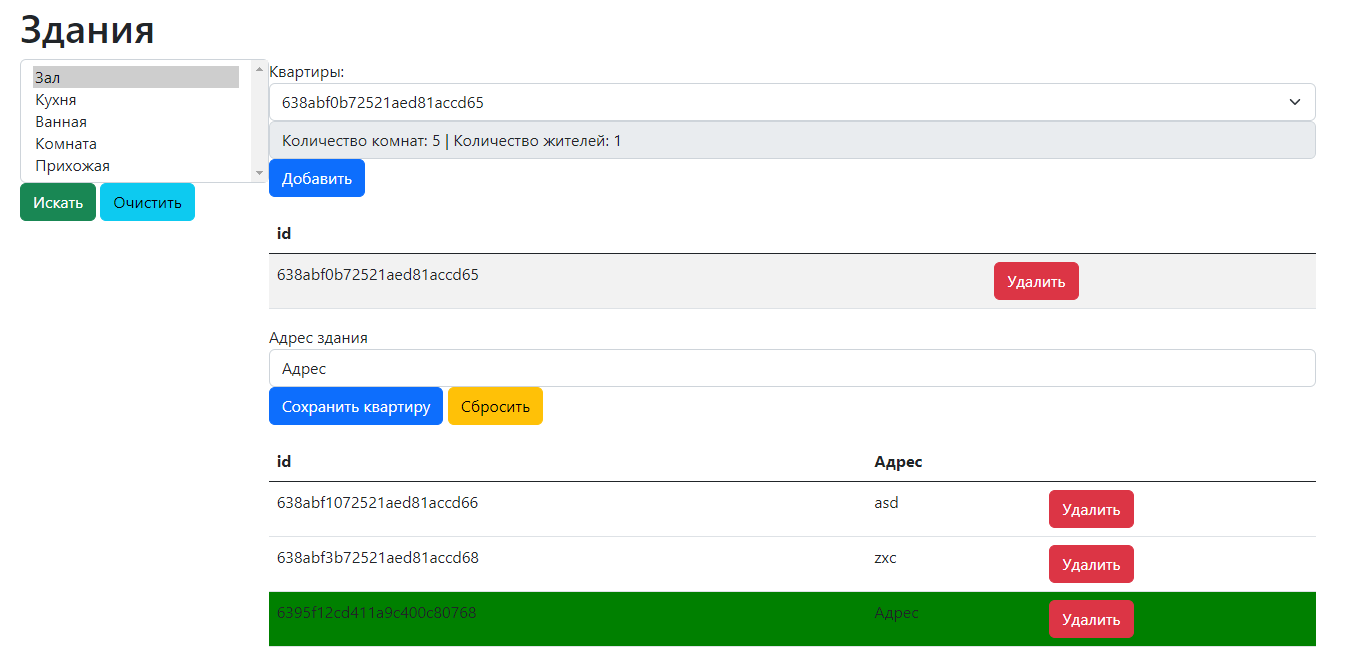


Рисунок 12 – Результат теста №10

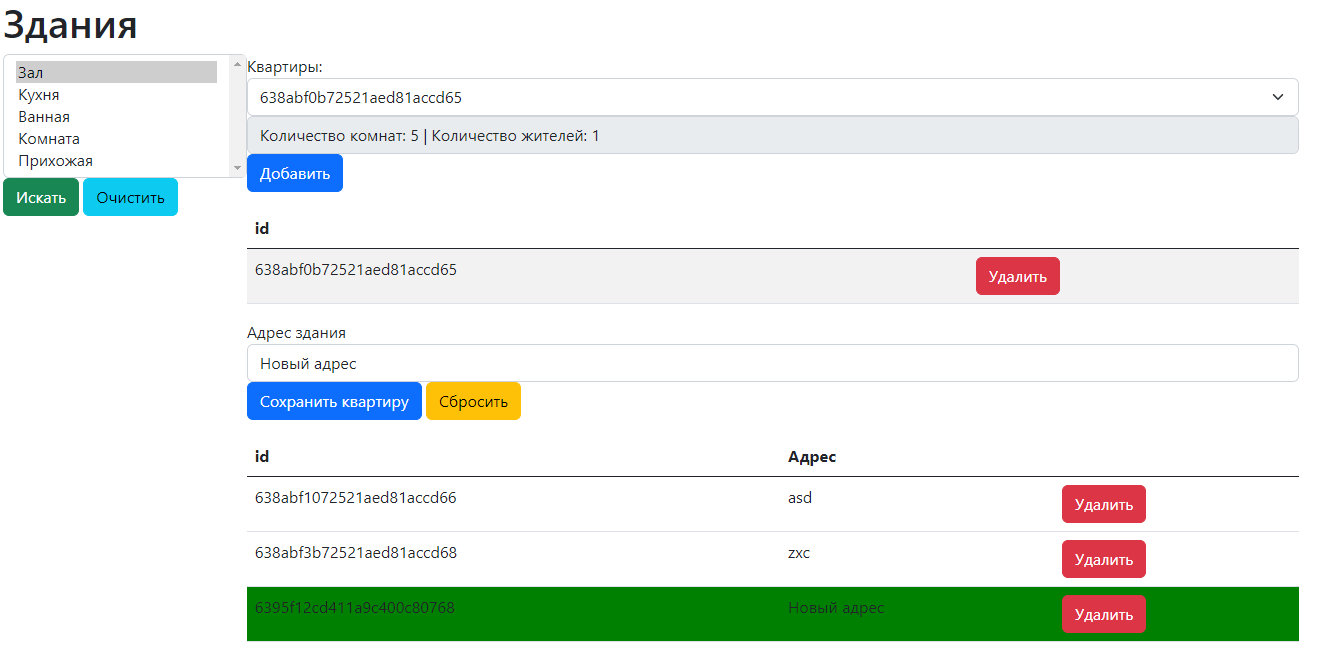


Рисунок 13 – Результат теста №11

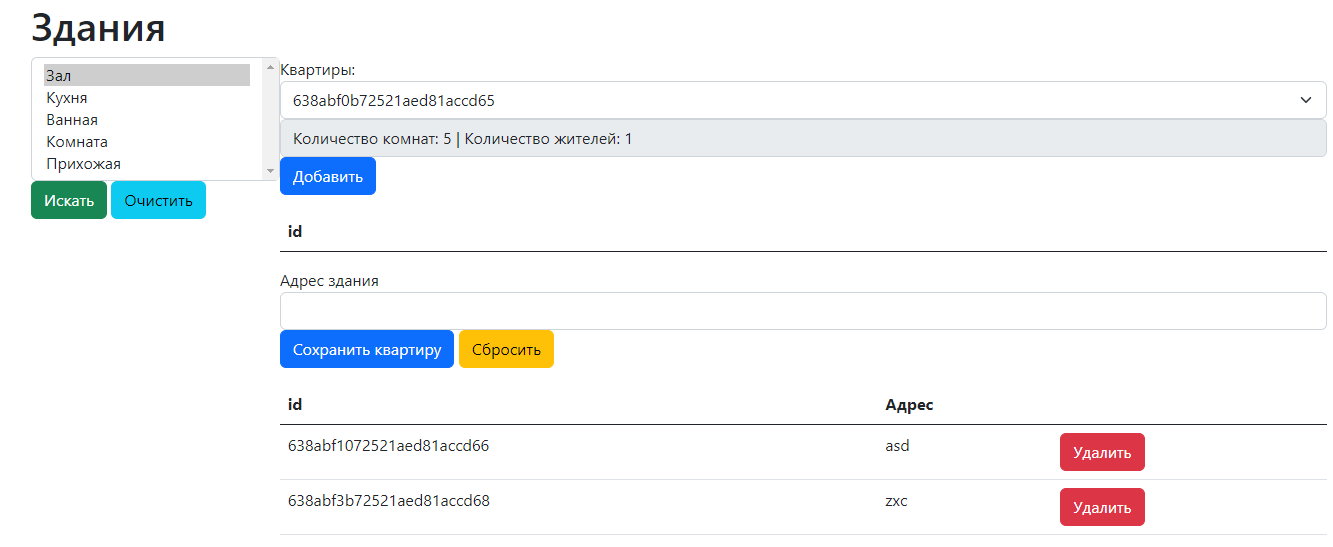


Рисунок 14 – Результат теста №12

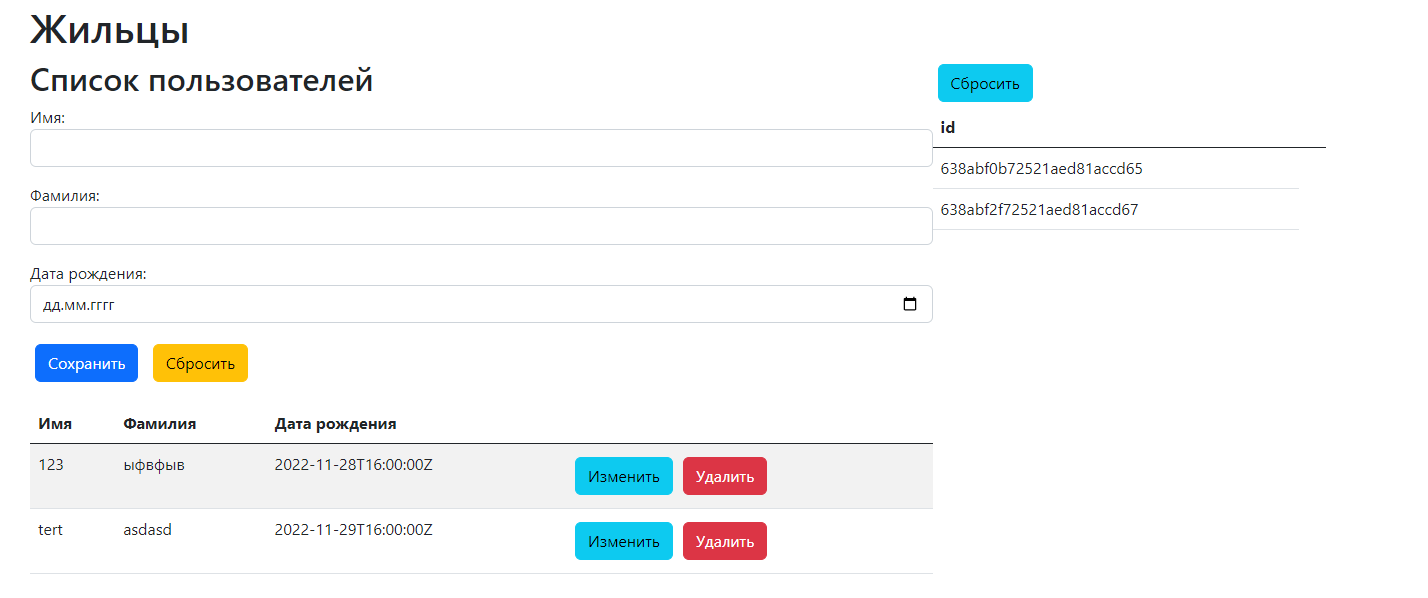


Рисунок 15 – Результат теста №13

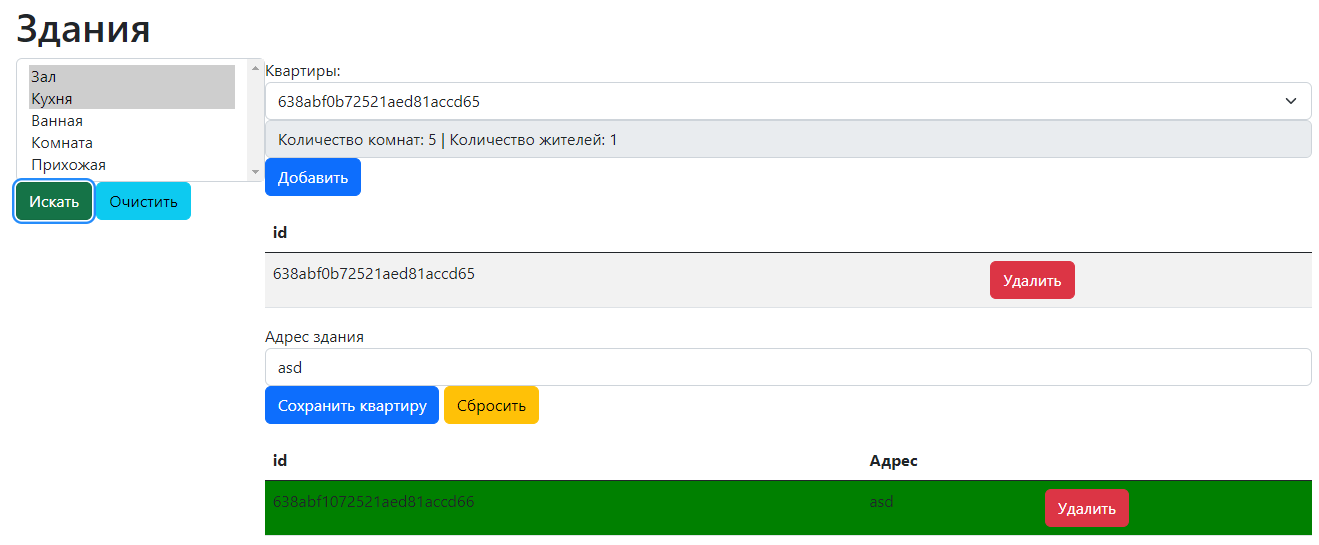


Рисунок 16 – Результат теста №14

Заключение

В процессе разработки веб-клиента были получены и закреплены практические навыки проектирования баз данных, получены новые навыки в использовании фреймворка ASP NET MVC.

В ходе выполнения проекта было разработано 14 тестов, где проверялась работоспособность реализованных операций и запросов.

Из проблем проекта стоит выделить повторяемость кода, которую можно решить путем шаблонизации некоторых участков страниц контроллеров.

Проект может быть продолжен, если исправить недостатки текущей версии приложения, улучшить положительные стороны и добавить новые функции.

Поставленные проектом задачи выполнены, все цели достигнуты.

Список использованных источников

1. Стасышин В. М., Стасышина Т. Л. Базы данных: Технологии доступа 2-е изд., испр. и доп. Учебное пособие для академического бакалавриата, 2019. 164 c. ISBN: 978-5-534-09888-4 Серия: Профессиональное образование. URL: https://biblio-online.ru/viewer/bazy-dannyhtehnologii-dostupa-415342#page/1.
2. Голицына О. Л. Базы данных: учебное пособие для вузов по направлению 230700 "Прикладная информатика" / О. Л. Голицына, Н. В. Максимов, И. И. Попов, 2012. 399 с.
3. Сосинская С. С. Использование языка C# в различных информационных технологиях: учебник для студентов вузов, обучающихся по направлению "Информационные системы и технологии / Сосинская, 2014. 367 с.
4. Шубина М.А. Управление данными: учебное пособие для студентов направлений подготовки 09.03.02 и 09.04.02 «Информационные системы и технологии» / Шубина М.А, 2016. 132 c. ISBN: 978-5-9239-0832-9. URL: https://e.lanbook.com/book/74029#book\_name.
5. Сосинская С.С. «Разработка клиента по технологии MVC»: Методические указания по выполнению курсового проекта» Электронный каталог кафедры вычислительной техники (дата обращения: 07.12.2021).
6. Руководство по ASP NET Core 5 [Электронный ресурс] // Метанит [Сайт], URL: https://metanit.com/sharp/aspnet5/ (Дата обращения: 25.11.2021)
7. MVC framework: большое введение для начинающих [Электронный ресурс] // Хабр [Сайт], URL: https://habr.com/ru/post/49718/ (Дата обращения: 1.12.2021)