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 “Менеджер игр”»

Обучающийся 4311 ГильфановаК.М.

(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 2022

ОГЛАВЛЕНИЕ

[1. ПОСТАНОВКА ЗАДАЧИ 3](#_Toc104279798)

[2. МОДЕЛЬ ДАННЫХ СИСТЕМЫ 4](#_Toc104279799)

[3. МАКЕТЫ HTML-СТРАНИЦ СИСТЕМЫ 5](#_Toc104279800)

[3.1 Макет главной страницы 5](#_Toc104279801)

[4. СТРУКТУРНАЯ СХЕМА МОДУЛЕЙ ПО, СОСТАВЛЯЮЩИХ СИСТЕМУ 5](#_Toc104279802)

[4.1 Структурная модель приложения. Модель MVC 5](#_Toc104279803)

[4.2 Структурная схема HTML страниц на стороне клиента 6](#_Toc104279804)

[5. ДИАГРАММА КЛАССОВ РАЗРАБОТАННОЙ СИСТЕМЫ 8](#_Toc104279805)

[6. ОБЗОР ПО, ВЫБРАННОГО ДЛЯ РЕАЛИЗАЦИИ СИСТЕМЫ 9](#_Toc104279806)

[6.1 Среда разработки – Sublime Text 9](#_Toc104279807)

[6.2 Веб-браузер Yandex Browser и инструменты разработчика 9](#_Toc104279808)

[6.3 Программная платформа Node.js 10](#_Toc104279809)

[6.4 СУБД MongoDB 10](#_Toc104279810)

[7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 11](#_Toc104279811)

[7.1 Главная страница сайта 11](#_Toc104279812)

[7.2 Страница личного кабинета 12](#_Toc104279813)

[ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ 13](#_Toc104279814)

[ПРИЛОЖЕНИЕ 1.Файл server.js 14](#_Toc104279815)

[ПРИЛОЖЕНИЕ 2.Файл index.html 14](#_Toc104279816)

[ПРИЛОЖЕНИЕ 3.Файл list.html 16](#_Toc104279817)

[ПРИЛОЖЕНИЕ 6. Файл app.js 16](#_Toc104279818)

[ПРИЛОЖЕНИЕ 7. Файл users.js 20](#_Toc104279819)

[ПРИЛОЖЕНИЕ 7. Файл user.js 21](#_Toc104279820)

[ПРИЛОЖЕНИЕ 8. Файл receipt.js 21](#_Toc104279821)

[ПРИЛОЖЕНИЕ 9. Файл admin.html 22](#_Toc104279822)

[ПРИЛОЖЕНИЕ 10. Файл admin.js 22](#_Toc104279823)

ВВЕДЕНИЕ

В наше время игры пользуются большой популярностью, существует множество игр разных жанров. Очень легко запутаться в таком разнообразии. Именно для этого существуют менеджеры игр, которые хранят информацию об играх которые вы бы хотели поиграть . На сайте вы можете записать название игры и ссылку на ее источник, чтобы не забыть и сыграть позже.

# 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 менеджера игр «Abstergo» (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 менеджера игр «Abstergo»

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. 1). Основная идея AJAX – возможность для приложения получать информацию и отдавать ее другим компьютерам без перезагрузки веб-страницы.



Рис. 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

## 3.1 Макет главной страницы

Макет главной страницы разрабатываемого Интернет-ресурса, выполненный в приложении “Paint”, изображен на рис. 2:

Изображение выглядит как текст

Автоматически созданное описание  
Рис. 2. Макет главной страницы

# 4. СТРУКТУРНАЯ СХЕМА МОДУЛЕЙ ПО, СОСТАВЛЯЮЩИХ СИСТЕМУ

## 4.1 Структурная модель приложения. Модель MVC

Наша система использует шаблон «модель – представление – контроллер» (Model – View – Controller (MVC)) (см. рис. 3).



Рис. 3. 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

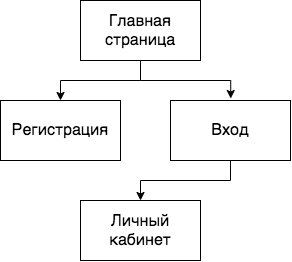


Рис. 4. Структурная схема клиента

# 5. ДИАГРАММА КЛАССОВ РАЗРАБОТАННОЙ СИСТЕМЫ

Диаграмма классов разработанной системы изображена на рис. 5:

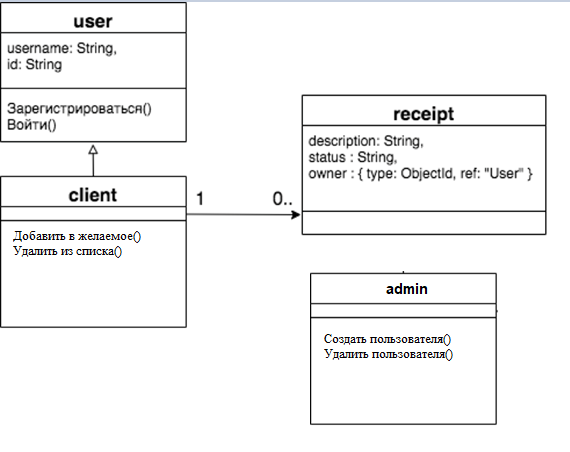


Рис. 5. Диаграмма классов разработанной системы

# 6. ОБЗОР ПО, ВЫБРАННОГО ДЛЯ РЕАЛИЗАЦИИ СИСТЕМЫ

## 6.1 Среда разработки – Sublime Text

Sublime Text — проприетарный текстовый редактор. Поддерживает плагины на языке программирования Python.

Разработчик позволяет бесплатно и без ограничений ознакомиться с продуктом, однако программа уведомляет о необходимости приобретения лицензии.

Некоторые возможности:

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

Sublime Text 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

Для запуска программы необходимо в адресной строке веб-браузера необходимо ввести адрес сайта менеджера игр «Abstergo» и нажать клавишу Enter. Будет запущена главная страница сайта (см. рис. 6).

Изображение выглядит как текст

Автоматически созданное описание

Рис. 6. Главная страница сайта

В «шапке» страницы изображена эмблема клиники и её название

В центральной части страницы находится поле для регистрации и/или входа на сайт. Перед вами находится поле ввода, в которое нужно ввести ваше имя пользователя (сайт укажет вам, если вы ввели его неправильно). Если вы впервые посещаете сайт, вам нужно нажать на кнопку регистрации, после чего вы получите уведомление (рис. 7)

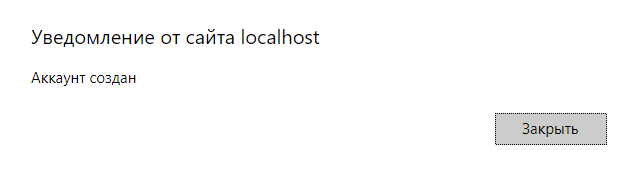


Рис. 7. Уведомление при регистрации на сайта

Нажав кнопку "закрыть" вы автоматически попадете в раздел личного кабинета.

Если же вы уже ранее регистрировались в системе сайта, вам нужно будет нажать на кнопку "войти", после чего вы также окажетесь в личном кабинете.

## 7.2 Страница личного кабинета

После того, как будет выполнен вход в личный кабинет, перед вами появится страница личного кабинета. В центральной части страницы находится поле с четырьмя вкладкам: история игр(рис 8), список желаемого, вкладка добавления в желаемое (рис. 9) и вкладка для выхода из личного кабинета

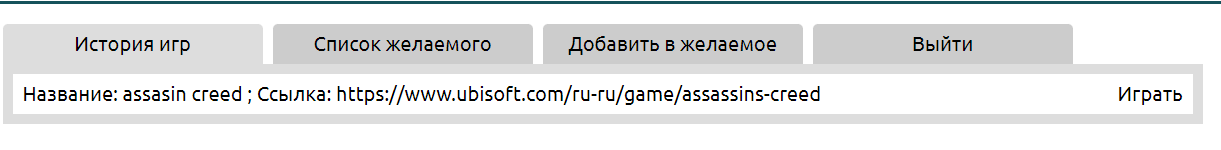


Рис. 8. Вкладка "История игр"

Изображение выглядит как текст

Автоматически созданное описание

Рис. 9. Вкладка "Добавить в желаемое"

На вкладке со списком желаемого (рис.11) рядом с каждой записью есть кнопка: если вы еще не играли в эту игру, то на кнопке будет написано "Играть", иначе – "Удалить из списка". Удалить можно только уже сыгранную игру!

# 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

var express = require("express"),

http = require("http"),

mongoose = require("mongoose"),

ReceiptController = require("./controllers/receipts\_controller.js"),

UsersController = require("./controllers/users\_controller.js"),

app = express();

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 Books в Mongo

mongoose.connect('mongodb://localhost/books', {

useNewUrlParser: true,

useCreateIndex: true,

useUnifiedTopology: true

}).then(res => {

console.log("DB Connected!")

}).catch(err => {

console.log(Error, err.message);

});

app.get("/receipts.json", ReceiptController.index);

app.get("/receipts/:id", ReceiptController.show);

app.post("/receipts", ReceiptController.create);

app.put("/receipts/:id", ReceiptController.update);

app.delete("/receipts/:id", ReceiptController.destroy);

app.get("/users/:username/receipts.json", ReceiptController.index);

app.post("/users/:username/receipts", ReceiptController.create);

app.put("/users/:username/receipts/:id", ReceiptController.update);

app.delete("/users/:username/receipts/:id", ReceiptController.destroy);

app.get("/users.json", UsersController.index);

app.post("/users", UsersController.create);

app.get("/users/:username", UsersController.show);

app.put("/users/:username", UsersController.update);

app.delete("/users/:username", UsersController.destroy);

# ПРИЛОЖЕНИЕ 2.Файл index.html

<!doctype html>

<html>

<head>

<title> Менеджер игр "Abstergo" </title>

<link rel="stylesheet" type="text/css" href="css/reset.css">

<link rel="stylesheet" type="text/css" href="css/style.css">

<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Ubuntu&subset=latin,cyrillic">

<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Droid+Sans&subset=latin,cyrillic" >

<link rel="stylesheet" type="text/css" href="fonts/1.css">

<script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>

<script type="text/javascript" src="js/users.js"></script>

</head>

<body>

<header>

<div class="container">

<a href="/index.html"><img src="/images/abstergo.png" alt="Логотип сайта" width="20%">

</a>

<nav>

<a class ="title" href="index.html" >Менеджер игр "Abstergo"</a>

</nav>

</div>

</header>

<main>

<div class="authorization">

<p>Введите номер вашей учетной записи</p>

</div>

</main>

<footer>

<div class="container">

<div class="contact">

<p>Менеджер игр "Abstergo"</p>

<p>Лучший менеджер игр для вас!</p>

<p> г.Казань, ул.Гоголя, 20</p>

<p> 2022 Abstergo</p>

</div>

</div>

</footer>

</body>

</html>

# ПРИЛОЖЕНИЕ 3.Файл list.html

<!doctype HTML>

<html>

<head>

<title> Менеджер игр "Abstergo" </title>

<link rel="stylesheet" type="text/css" href="/css/reset.css">

<link rel="stylesheet" type="text/css" href="/css/style.css">

<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Ubuntu&subset=latin,cyrillic">

<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Droid+Sans&subset=latin,cyrillic" >

<link rel="stylesheet" type="text/css" href="/fonts/1.css">

<script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>

<script type="text/javascript" src="/js/app.js"></script>

</head>

<body>

<header>

<div class="container">

<a href="/index.html"><img src="/images/abstergo.png" alt="Логотип сайта" width="20%">

</a>

<nav>

<a class ="title" href="index.html" >Менеджер игр "Abstergo"</a>

</nav>

</div>

</header>

<main>

<div class="container">

<div class="tabs">

</div>

<div class="content">

</div>

</div>

</main>

<footer>

<div class="container">

<div class="contact">

<p>Менеджер игр "Abstergo"</p>

<p>Лучший менеджер игр для вас!</p>

<p> г.Казань, ул.Гоголя, 20</p>

<p> 2022 Abstergo</p>

</div>

</div>

</footer>

</body>

</html>

# ПРИЛОЖЕНИЕ 6. Файл app.js

var liaWithEditOrDeleteOnClick = function (Receipt, callback) {

var $ReceiptListItem = $("<li>").text(Receipt.description),

$ReceiptEditLink = $("<a>").attr("href", "Receipts/" + Receipt.\_id),

$ReceiptRemoveLink = $("<a>").attr("href", "Receipts/" + Receipt.\_id);

$ReceiptEditLink.addClass("linkEdit");

$ReceiptRemoveLink.addClass("linkRemove");

if (Receipt.status === 'В желаемом') {

$ReceiptEditLink.text("Играть");

$ReceiptEditLink.on("click", function() {

var newDescription = Receipt.description + "Сыграно";

if (newDescription !== null && newDescription.trim() !== "") {

$.ajax({

"url": "/Receipts/" + Receipt.\_id,

"type": "PUT",

"data": { "description": newDescription, "status": 'Куплено'},

}).done(function (responde) {

Receipt.status = 'Сыграно';

callback();

}).fail(function (err) {

console.log("Произошла ошибка: " + err);

});

}

return false;

});

$ReceiptListItem.append($ReceiptEditLink);

}

else {

$ReceiptRemoveLink.text("Удалить из списка");

$ReceiptRemoveLink.on("click", function () {

$.ajax({

url: "/Receipts/" + Receipt.\_id,

type: "DELETE"

}).done(function (responde) {

callback();

}).fail(function (err) {

console.log("error on delete 'Receipt'!");

});

return false;

});

$ReceiptListItem.append($ReceiptRemoveLink);

}

return $ReceiptListItem;

}

var main = function (ReceiptObjects) {

"use strict";

var tabs = [];

tabs.push({

"name": "История игр",

"content": function(callback) {

$.getJSON("Receipts.json", function (ReceiptObjects) {

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

for (var i = ReceiptObjects.length-1; i>=0; i--) {

var $ReceiptListItem = liaWithEditOrDeleteOnClick(ReceiptObjects[i], function() {

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

});

$content.append($ReceiptListItem);

}

callback(null, $content);

}).fail(function (jqXHR, textStatus, error) {

callback(error, null);

});

}

});

tabs.push({

"name": "Список желаемого",

"content": function(callback) {

$.getJSON("Receipts.json", function (ReceiptObjects) {

var $content, i;

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

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

if (ReceiptObjects[i].status === 'В желаемом') {

var $ReceiptListItem = liaWithEditOrDeleteOnClick(ReceiptObjects[i], function() {

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

});

$content.append($ReceiptListItem);

}

}

callback(null, $content);

}).fail(function(jqXHR, textStatus, error) {

callback(error, null);

});

}

});

tabs.push({

"name": "Добавить в желаемое",

"content":function () {

$.get("Receipts.json", function (ReceiptObjects) {

// создание $content для Добавить

$("main .content").append($place);

var $place = $("<h3>").text("Введите название игры: "),

$input = $("<input>").addClass("description"),

$place2 = $("<h3>").text("Введите ссылку на сайт игры: "),

$input2 = $("<input>").addClass("gamelink"),

$button = $("<button>").text("Добавить"),

$content1 = $("<ul>"),

$content2 = $("<ul>");

$content1.append($input);

$content2.append($input2);

$("main .content").append($place);

$("main .content").append($content1);

$("main .content").append($place2);

$("main .content").append($content2);

$("main .content").append($button);

function btnfunc() {var description = ('Название: ' + $input.val() + ' ; Ссылка: ' + $input2.val() +' '),

linkgame = ($input2.val()),

newReceipt = {"description":description, "status": 'В желаемом', "gamelink":linkgame};

$.post("Receipts", newReceipt, function(result) {

$input.val("");

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

});

}

$button.on("click", function() {

btnfunc();

});

$input.on('keydown',function(e){

if (e.which === 13) {

btnfunc();

}

});

});

}

});

tabs.push ({

"name": "Выйти",

"content":function() {

document.location.href = "/index.html";

}

});

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("Receipts.json", function (ReceiptObjects) {

main(ReceiptObjects);

});

})

# ПРИЛОЖЕНИЕ 7. Файл users.js

var main = function (UsersObjects) {

var asdf = 0;

"use strict";

var $input = $("<input placeholder='имя пользователя'>").addClass("username"),

$butRegister = $("<button>").text("Зарегистрироваться"),

$butLogin = $("<button>").text("Войти");

$butRegister.on("click", function() {

var username = $input.val();

var example = /^[а-я]{3,20}(?<!000)$/ui;

console.log(username);

var check = example.test(username);

console.log(check);

if (check) {

if (username !== null && username.trim() !== "") {

var newUser = {"username": username};

$.post("users", newUser, function(result) {

console.log(result);

UsersObjects.push(newUser);

}).done(function(responde) {

console.log(responde);

alert('Аккаунт создан');

$butLogin.trigger("click");

}).fail(function(jqXHR, textStatus, error) {

console.log(error);

if (jqXHR.status === 501) {

alert("Такой пользователь уже зарегестрирован\nВведите другое имя пользователя");

} else {

alert("Произошла ошибка\n"+jqXHR.status + " " + jqXHR.textStatus);

}

});

}

}

else {

alert('Имя пользователя должен состоять из строчных букв и быть длиной не больше 20 символов\nПовторите ввод')

$input.val("");

}

});

$butLogin.on("click", function() {

var username = $input.val();

if (username !== null && username.trim() !== "") {

var loginUser = {"username": username};

$.ajax({

'url': '/users/'+username,

'type': 'GET'

}).done(function(responde) {

window.location.replace('users/' + username + '/');

}).fail(function(jqXHR, textStatus, error) {

console.log(error);

alert("Произошла ошибка\n"+jqXHR.status + " " + jqXHR.textStatus);

});

}

});

$("main .authorization").append($input);

$("main .authorization").append($butLogin);

$("main .authorization").append($butRegister);

}

$(document).ready(function() {

sessionStorage.setItem('key', 'value');

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

main(UsersObjects);});});

# ПРИЛОЖЕНИЕ 7. Файл user.js

var mongoose = require("mongoose");

var UserSchema = mongoose.Schema({

username: String,

id: String

});

var User = mongoose.model("User", UserSchema);

module.exports = User;

# ПРИЛОЖЕНИЕ 8. Файл receipt.js

var mongoose = require("mongoose"),

ObjectId = mongoose.Schema.Types.ObjectId;

var ReceiptSchema = mongoose.Schema({

description: String,

status : String,

owner : { type: ObjectId, ref: "User" }

});

var Receipt = mongoose.model("Receipt", ReceiptSchema);

module.exports = Receipt;

# ПРИЛОЖЕНИЕ 9. Файл admin.html

<!doctype html>

<html>

<head>

<title> Менеджер игр "Abstergo" </title>

<link rel="stylesheet" type="text/css" href="css/reset.css">

<link rel="stylesheet" type="text/css" href="css/style.css">

<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Ubuntu&subset=latin,cyrillic">

<link rel="stylesheet" type="text/css" href="http://fonts.googleapis.com/css?family=Droid+Sans&subset=latin,cyrillic" >

<link rel="stylesheet" type="text/css" href="fonts/1.css">

<script src="http://code.jquery.com/jquery-3.4.1.min.js"></script>

<script type="text/javascript" src="js/admin.js"></script>

</head>

<body>

<header>

<div class="container">

<a href="/index.html"><img src="/images/abstergo.png" alt="Логотип сайта" width="20%">

</a>

<nav>

<a class ="title" href="index.html" >Менеджер игр "Abstergo"</a>

</nav>

</div>

</header>

<main>

<div class="authorization">

<p>Введите имя пользователя</p>

</div>

</main>

<footer>

<div class="container">

<div class="contact">

<p>Менеджер игр "Abstergo"</p>

<p>Лучший менеджер игр для вас!</p>

<p> г.Казань, ул.Гоголя, 20</p>

<p> 2022 Abstergo</p>

</div>

</div>

</footer>

</body>

</html>

# ПРИЛОЖЕНИЕ 10. Файл admin.js

var main = function (UsersObjects) {

var asdf = 0;

"use strict";

var $input = $("<input placeholder='имя пользователя'>").addClass("username"),

$butRegister = $("<button>").text("Зарегистрировать"),

$butDestroy = $("<button>").text("Удалить");

$butRegister.on("click", function() {

var username = $input.val();

var example = /^[а-я]{3,20}(?<!000)$/ui;

console.log(username);

var check = example.test(username);

console.log(check);

if (check) {

if (username !== null && username.trim() !== "") {

var newUser = {"username": username};

$.post("users", newUser, function(result) {

console.log(result);

UsersObjects.push(newUser);

}).done(function(responde) {

console.log(responde);

alert('Аккаунт создан');

$butLogin.trigger("click");

}).fail(function(jqXHR, textStatus, error) {

console.log(error);

if (jqXHR.status === 501) {

alert("Такой пользователь уже зарегестрирован\nВведите другое имя пользователя");

} else {

alert("Произошла ошибка\n"+jqXHR.status + " " + jqXHR.textStatus);

}

});

}

}else {

alert('Имя пользователя должен состоять из строчных букв и быть длиной не больше 20 символов\nПовторите ввод')

$input.val("");}});

$butDestroy.on("click", function() {

if ($input.val() !== "") {

if ($input.val() !== null && $input.val().trim() !== "") {

var username = $input.val();

$.ajax({

'url': '/users/'+username,

'type': 'DELETE',

}).done(function(responde) {

console.log(responde);

$input.val("");

alert('Пользователь удален');

}).fail(function(jqXHR, textStatus, error) {

console.log(error);

alert("Произошла ошибка\n"+jqXHR.status + " " + jqXHR.textStatus);

});

}

}

});

$("main .authorization").append($input);

$("main .authorization").append($butDestroy);

$("main .authorization").append($butRegister);

}

$(document).ready(function() {

sessionStorage.setItem('key', 'value');

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

main(UsersObjects);

});

}