МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ КРОССПЛАТФОРМЕННОЙ УРБАНИСТИЧЕСКОЙ ВЕБ-ИГРЫ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Васильева Михаила Владимировича

Научный руководитель
ст. преподаватель _____ М. И. Сафрончик

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные города — сложные саморазвивающиеся системы. Урбанистика — набор принципов, помогающих организовать максимально комфортные для жизни города, в которых учтены потребности и интересы людей из самых разных категорий и социальных групп. Одним из принципов урбанистики является обязательное участие горожан в планировании городской среды. Активные граждане определяют круг проблем и контролируют их решение. Это позволяет достичь наилучших результатов.

Целью работы является разработка серверной части кроссплатформенной урбанистической веб-игры, предназначенной для обучения игроков в данной сфере. Были поставлены следующие задачи:

- проектирование базы данных;
- разработка модуля авторизации;
- разработка модуля списка игр;
- разработка модуля активных игр;
- разработка модуля хаба;
- разработка модуля взаимодействия с игрой;
- разработка модуля игры;
- реализация двустороннего обмена данными между клиентом и сервером для модуля игры.

Для реализации этих задач был выбран язык программирования C# и фреймворк ASP.NET Core, благодаря которому была реализована кроссплатформенность [1]. В качестве базы данных была выбрана MySQL, для работы с которой использовался пакет Entity Framework Core. Двусторонний обмен данными между клиентом и сервером реализован по протоколу WebSocket, с использованием пакета SignalR.

Бакалаврская работа состоит из введения, трех глав, заключения, списка используемых источников и приложений. В первом разделе описана теоретическая часть и правила игры. Второй раздел содержит описание технологического стека проекта. В третьем разделе описана практическая часть работы: архитектура проекта и реализация модулей игры.

1 Общая информация об игре

Данная игра относится к виду учебно-деловых игр [2]. Основная цель таких игр — научить участников ориентироваться в различных ситуациях, учитывать возможности и состояния других людей, устанавливать с ними контакты, влиять на их интересы [3]. Существует много разновидностей учебно-деловых игр. Например, экономические или бизнес-деловые игры, но ни одной урбанистической игры, на момент ее разработки, найти не удалось. Большинство таких игр существуют в формате настольной или компьютерной игры с необходимостью установки.

В качестве примеров учебно-деловых игр можно привести игры:

- «Тренинг по ощущению бизнеса» — настольная игра и ее электронная версия «Собственный КАПИТАЛ». Они представляют виртуальную экономико-управленческую модель организации производства и сбыта продукции с привлечением средств инвесторов.
- Компьютерные деловые игры серии «БИЗНЕС-КУРС» — суть игр в управлении виртуальным предприятием, действующим в условиях конкуренции.
- «Монополия» — экономическая настольная игра. Цель игры остаться единственным игроком, рационально используя стартовый капитал. Также имеет многочисленные компьютерные воспроизведения.

Было решено создать урбанистическую веб-игру, в которой будет максимально автоматизирован процесс игры, оставляя игроку только возможность принятия решений. Формат веб-игры избавляет пользователя от необходимости установки на компьютер, а также дает возможность входа в игру в любой момент. Автоматизация процесса игры повышает удобство и доступность для игроков. Благодаря чему, появляется возможность проведения онлайн или офлайн мероприятий, на которых любой желающий сможет в нее сыграть без долгих объяснений.

В основе этой игры лежат бизнес-деловые пошаговые игры, в которых любой игрок может влиять на ход игры посредством принятия определенных решений и для того чтобы показать их правильность было решено разделить игроков на две команды, конкурирующие между собой. По итогам игры побеждает та команда, которая принимала решения, оказывающие наиболее положительный эффект на общую среду, в которой находятся игроки этой

команды.

1.1 Участие в игре

Перед тем как принять участие в игре, пользователю необходимо зарегистрироваться. Если же у пользователя уже есть учетная запись, он может войти в нее с использованием электронной почты и пароля. Далее пользователь может ознакомиться со списком запланированных игр. Пользователь может зарегистрироваться на любую из игр, где еще есть свободные места. После регистрации появляется возможность входа в игру для ознакомления с общей информацией об игре и ожидания ее начала.

1.2 Стадии игры

Игра состоит из стадий, некоторые из них повторяются. На каждую стадию отводится определенное время, по истечению которого игра переходит в следующую стадию. Стадии игры:

- ожидание начала;
- выборы мэра;
- ход;
- городской совет;
- закончена.

На рисунке 1 представлена блок-схема смены стадий игры.

1.3 Игра

Перед началом игры игроки делятся на две команды, распределение игроков между командами происходит случайным образом при регистрации на игру. Каждая команда представляется городом, имеющим набор характеристик, изменяющихся по ходу игры.

1.4 Город и карта города

В игре может быть всего два города, каждый их них представляется следующими параметрами:

- название;
- мэр;
- баланс казны;
- население;

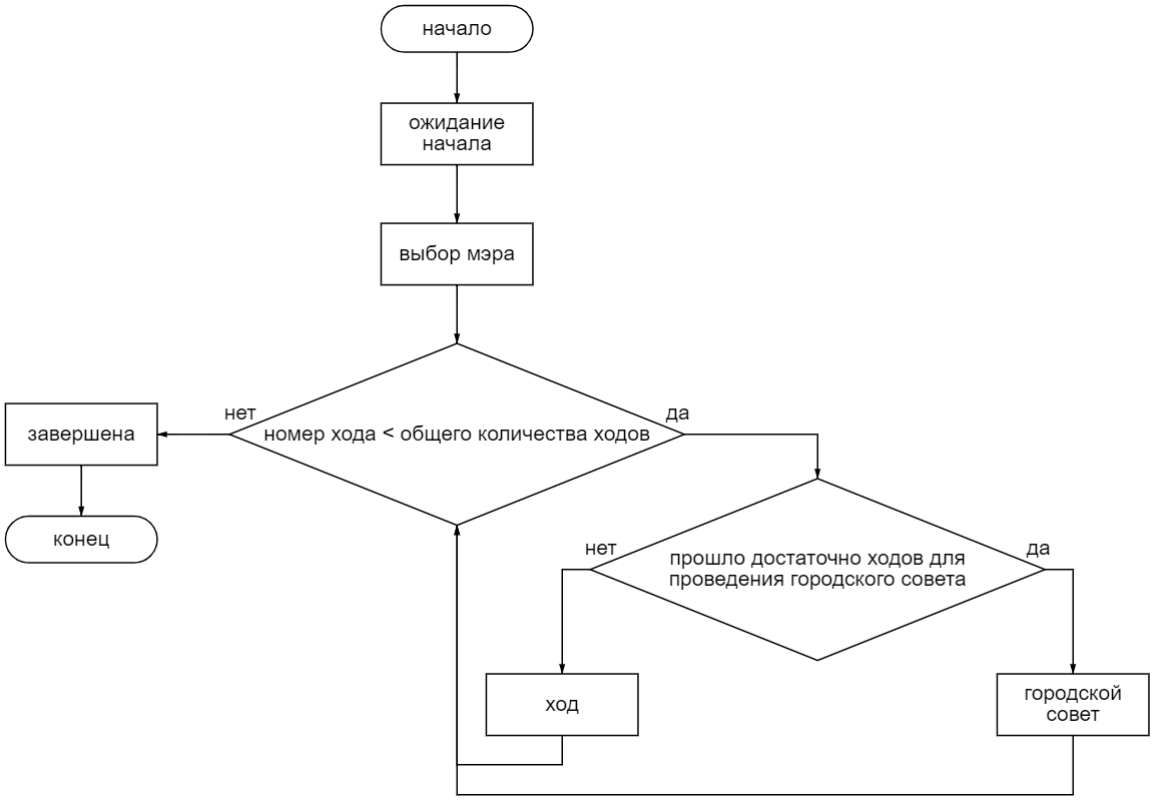

Рисунок 1 – Блок-схема смены стадий игры

— налоговая система;
— рейтинг;
— карта.

1.4.1 Карта города

Карта города представляет собой упорядоченный набор из 24-х клеток. Каждая клетка может быть одного из следующих типов:

— проблема;
— налог;
— зарплата;
— переезд.

Клеток типа «проблема» может быть всего 20, типа «налог» и «зарплата» по одной, и клеток типа «переезд» — две.

1.4.2 Проблема

Всего в городе может быть 22 проблемы. Проблема — это отражение ситуации в городе по определенному признаку, например, «безопасность» или «здравоохранение». Проблема имеет следующие свойства:

— название;
— значение показателя;
— автоухудшение;
— набор состояний;
— набор решений.

В зависимости от значения показателя проблемы определяется ее состояние. Существуют следующие состояния:

— плохое — при значении показателя от 0 до 2;
— нейтральное — при значении показателя от 3 до 7;
— хорошее — при значении показателя от 8 до 10.

Каждое действие оказывает эффект определенного типа. Всего три типа эффектов: перемещение по карте, изменение баланса игрока и изменение дохода игрока. Значение эффекта может быть как положительным, так и отрицательным. Таким образом, в зависимости от значения показателя, определяется текущее состояние проблемы. Когда игрок останавливается на клетке этой проблемы, ему предлагается выбор из действий, относящихся к текущему состоянию проблемы. Решения проблемы предназначены для увеличения значения показателя проблемы. Чем оно выше, тем лучше состояние проблемы. Каждое решение можно применить всего один раз за игру. Всего решений у каждой проблемы города может быть три.

1.4.3 Мэр

У города есть три кандидата в мэры, каждый из них имеет влияние на несколько проблем города. Влияние на показатель проблемы может быть как положительным, так и отрицательным.

1.5 Городской совет

Во время городского совета игрокам предлагается ознакомиться со всеми проблемами города и их состоянием. Игрок может выбрать одно или несколько из предложенных решений для затронувших его проблем. Количество решений, за которые можно проголосовать, ограничено параметрами игры, а также балансом казны города. По окончанию городского совета принимаются все решения, за которые проголосовали игроки города, в порядке уменьшения количества голосов и на которые достаточно средств в казне этого города.

2 Использованные технологии

В качестве технологического стека было принято решение выбрать:

— язык программирования C#;
— фреймворк ASP.NET Core;
— база данных MySQL;
— пакет Entity Framework Core;
— пакет SignalR;
— пакет JwtBearer;
— пакет Swashbuckle для ASP.NET Core.

2.1 ASP.NET Core

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету [4]. Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget.

2.2 Entity Framework Core

Entity Framework (EF) Core — это простая, кроссплатформенная и расширяемая версия технологии доступа к данным Entity Framework с открытым исходным кодом, разработанная компанией Microsoft [5]. EF Core поддерживает множество различных систем баз данных. Таким образом, можно через EF Core работать с любой СУБД, если для нее имеется нужный провайдер [6]. В данном случае для работы с базами данных MySQL использовался провайдер MySql.EntityFrameworkCore, разработанный компанией Oracle. Для работы с функциями миграции [7] использовался дополнительный пакет EntityFrameworkCore.Tools.

2.3 SignalR

SignalR Core представляет библиотеку от компании Microsoft, которая предназначена для создания приложений, работающих в режиме реального времени [8]. SignalR использует двунаправленную связь для обмена сообщениями между клиентом и сервером. Наилучшим механизмом для взаимодействия является WebSockets, также имеется поддержка Server-Sent Events [9] и Long Polling [10].

2.4 JwtBearer

В Web API механизм авторизации полагается преимущественно на JWT-токены. JWT (или JSON Web Token) представляет собой веб-стандарт, который определяет способ передачи данных о пользователе в формате JSON в зашифрованном виде [11].

2.5 Swashbuckle

Пакет Swashbuckle для ASP.NET Core позволяет легко добавить в проект документацию Swagger [12]. В дополнение к генератору Swagger также содержит встроенную версию удобного интерфейса swagger-ui.

3 Разработка серверной части проекта веб-игры

Архитектура проекта представлена на рисунке 2. Клиентская часть реализована с использованием библиотеки React. Для коммуникации между клиентом и сервером были использованы два протокола: HTTP [13] и WebSocket [14]. Для коммуникации вне модуля игры использован классический для REST-API [15] протокол HTTP. В качестве базы данных был сделан выбор в пользу MySQL.

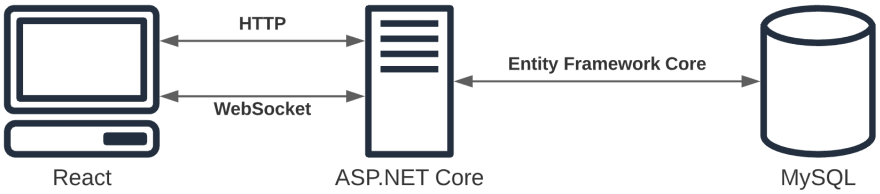

Рисунок 2 – Архитектура проекта

При создании проекта использован шаблон веб-API для ASP.NET Core. Структура проекта во многом напоминает MVC [16], но в ней отсутствуют представления.

3.1 Подключение базы данных

Конфигурация подключения к базе данных описывается в файле конфигурации проекта appsettings.json. В классе $Startup$, в методе $ConfigureServices$ добавим контекст подключения, для того чтобы затем получать его в конструкторе контроллера через механизм внедрения зависимостей [17].

3.2 Подключение SignalR

Для подключения SignalR необходимо в классе $Startup$, в методе $ConfigureServices$ задействовать сервисы SignalR, а также в методе $Configure$ связать запросы и класс хаба $GameHub$. Этот класс будет обрабатывать все запросы.

3.3 Авторизация с помощью JWT-токенов

Создадим специальный класс $AuthenticationOptions$, в котором будут описаны свойства для генерации токена [18]. Для того чтобы встроить функциональность JWT-токенов в конвейер обработки запроса необходимо использовать компонент $JwtBearerAuthenticationMiddleware$.

SignalR на стороне клиента не добавляет токен в заголовки запроса. И при обращении к хабу токен фактически будет посылаться как параметр строки запроса. Добавим событие $OnMessageReceived$, в котором передадим строку токена из параметров запроса в параметры контекста, для дальнейшей валидации токена [19].

Дополнительно добавим событие $OnTokenValidated$, для проверки существования пользователя в базе данных.

3.4 Расширения класса Controller

Для удобства обработки ошибок контроллеров был написан универсальный обработчик исключений, метод расширения [20] $ExceptionHandling$ для класса $Controller$. Данный метод производит простое логирование ошибок в консоль, а также проводит проверку типа исключения, возвращая соответствующее сообщение об ошибке. Еще один расширяющий метод $GetUserId$ — возвращает идентификатор пользователя из объекта $Claims$.

3.5 Авторизация и регистрация

Авторизация и регистрация реализована в контроллере $AuthController$. PUT-запрос $SignIn$ позволяет выполнить авторизацию. POST-запрос $SignUp$ реализует возможность регистрации.

3.6 Игра

Состояние игры может быть одним из следующих:

- создана;
- ожидает начала;
- в процессе;
- пауза;
- закончена;
- завершена с ошибкой.

Было принято решение дополнить стадии, описанные в правилах. Так как на каждую стадию отводится определенное время, и они идут подряд друг за другом, возникает необходимость выделения дополнительного времени на расчет данных. Перед началом очередной стадии нужно подготовить данные, отправляемые на клиентскую часть всем игрокам. В конце стадии необходимо обработать данные, полученные с клиентской части. И выделить дополнитель-

ное время на ожидание данных с клиентской части для того, чтобы компенсировать задержки интернет-соединения, в случае если пользователь выполнил какое-либо действие в последний момент. Необходимо дополнить лишь стадии «выбор мэра», «ход» и «городской совет». Таким образом каждая стадия будет включать в себя набор из основной и дополнительных стадий:

1. подготовка;
2. стадия;
3. ожидание;
4. расчет результатов.

При этом стоит отметить, что время, выделенное на основную стадию, остается неизменным, дополнительные стадии используют отдельное время, отображаемое игроку на клиентской части как пауза между стадиями игры. Также добавилась стадия «подождите» — это стадия по умолчанию. Сразу после начала игра переходит в эту стадию. Она же будет отправляться на клиентскую часть вместо стадий «подготовка», «ожидание» и «расчет результатов», так как на клиентской части игроку эта информация не нужна. На рисунке 3 представлена блок-схема смены стадий игры.

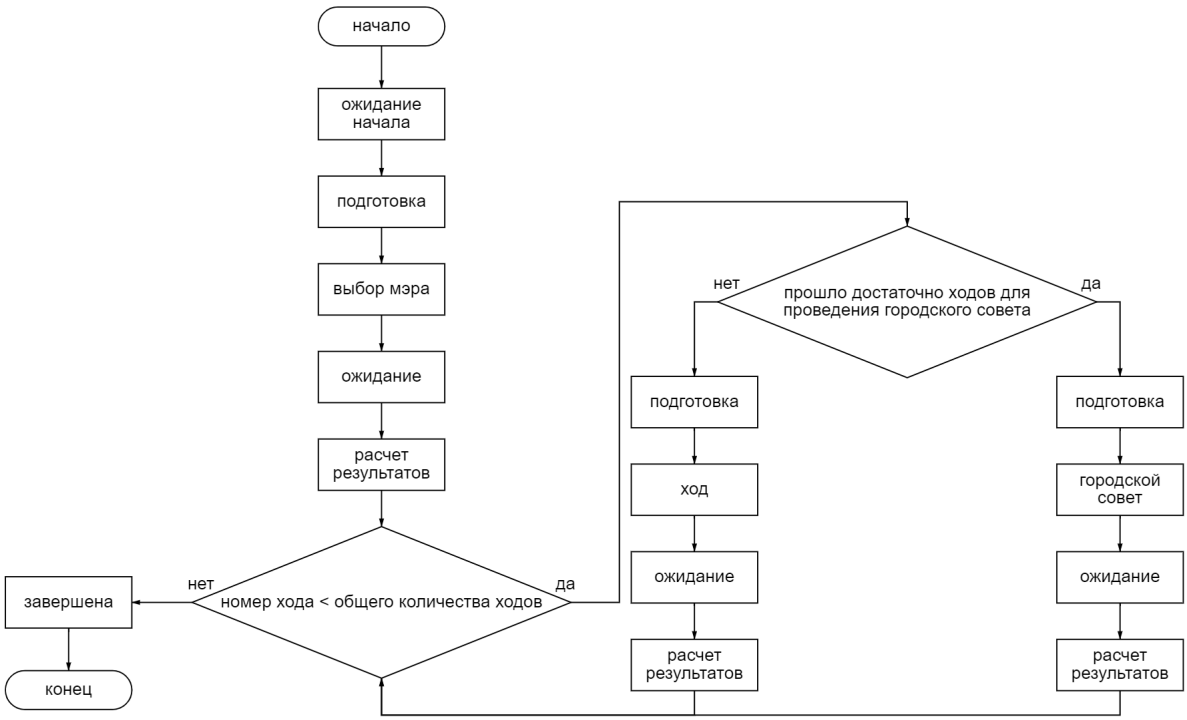

Рисунок 3 – Блок-схема смены стадий игры

3.7 Модуль списка игр

Для реализации модуля был создан контроллер $GameController$. GET-запрос $list$ возвращает список игр, находящихся в одном из состояний: «ожидание начала», «в процессе» и «пауза». POST-запрос $registration$ выполняет регистрацию на игру.

3.8 Модуль активных игр

Сервис $GameCenterService$ реализует этот модуль. Данный сервис предназначен для запуска всех активных игр и передачи им команд, полученных хабом от клиентской части. Так как игра коммуницирует с клиентской частью через хаб, необходимо передать в сервис его контекст [21].

3.9 Модуль хаба

Хаб состоит из набора типовых методов, принимающих команды от клиентской части. Набор параметров для каждой команды хаба отличается. Реализованы следующие методы:

- отключение клиента от хаба;
- выбор мэра;
- выбор действия во время стадии «ход»;
- смена города;
- запуск события смены города;
- выбор решений во время стадии «городской совет».

3.10 Модуль взаимодействия с игрой

Данный модуль описывает взаимодействия с активной игрой. Контроллер $PlayGameController$ реализует функционал этого модуля. POST-запрос $enter$ выполняет вход в игру и возвращает информацию о ней.

3.11 Модуль игры

Данный модуль реализует основную логику игры, он описан в классе $PlayGameService$. Первоначально выполняется проверка возможности запуска игры и загрузка всех необходимых данных. Далее запускается цикл по стадиям игры. В начале каждой стадии вычисляются необходимые данные, которые будут отправлены на клиентскую часть. Во время стадии выполняется сбор данных о действиях игроков, а в конце — обработка полученных данных.

ЗАКЛЮЧЕНИЕ

В результате выполнения бакалаврской работы была разработана серверная часть кроссплатформенной урбанистической веб-игры, были выполнены следующие задачи:

- спроектирована база данных;
- разработаны модули авторизации, списка игр, активных игр, хаба, взаимодействия с игрой и игры;
- реализован двусторонний обмен данными с клиентом для модуля игры.

В процессе разработки были применены следующие технологии:

- ASP.NET Core;
- Entity Framework Core;
- JWT;
- SignalR.

В дальнейшем планируется разработать следующие модули:

- Чат — предназначен для общения игроков команды во время игры.
- СМИ — информирование игроков команды о событиях, произошедших в городе во время игры. События возникают при попадании игрока на клетку типа «проблема», принятии решения на городском совете и при создании вручную администратором или модератором игры.
- Административная панель — предназначена для настройки параметров игры, наблюдения и управления активной игрой.

Процесс внедрения программного продукта на виртуальный сервер производится через терминал с помощью bash-скрипта, который предоставляет пользователю понятный и удобный интерфейс взаимодействия. Скрипт устанавливает все необходимые пакеты для работы программного продукта, загружает и запускает приложение. Дальнейшее взаимодействие через терминал не требуется.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Прайс, М.* C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов / М. Прайс. — СПб.: Питер.

2 *Коджаспирова, Г. М.* Педагогика / Г. М. Коджаспирова. — Москва: Юрайт, 2015.

3 *Гребенюк, О. С.* Теория обучения / О. С. Гребенюк. — Москва: Юрайт, 2019.

4 *Прайс, М.* C# 8 и .NET Core. Разработка и оптимизация / М. Прайс. — СПб.: Питер.

5 *Фримен, А.* Entity Framework Core 2 для ASP.NET Core MVC для профессионалов / А. Фримен. — Москва: Вильямс, 2019.

6 Поставщики баз данных [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/ef/core/providers/ (Дата обращения 01.05.2021). Загл. с экр. Яз. рус.

7 Обзор миграций [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/ef/core/managing-schemas/migrations/ (Дата обращения 01.05.2021). Загл. с экр. Яз. рус.

8 *Vemula, R.* Real-Time Web Application Development / R. Vemula. — New York City: Apress, 2017.

9 Server-sent events [Электронный ресурс]. — URL: https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events (Дата обращения 02.05.2021). Загл. с экр. Яз. англ.

10 Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP [Электронный ресурс]. — URL: https://datatracker.ietf.org/doc/rfc6202/ (Дата обращения 02.05.2021). Загл. с экр. Яз. англ.

11 *Fagerberg, J.* ASP.NET Core 2.2 MVC, Razor Pages, API, JSON Web Tokens & HttpClient / J. Fagerberg. — Amazon Digital Services LLC - KDP Print US, 2019.

12 Swashbuckle [Электронный ресурс]. — URL: https://github.com/domaindrivendev/Swashbuckle.WebApi (Дата обращения 04.05.2021). Загл. с экр. Яз. англ.

13 Hypertext Transfer Protocol – HTTP/1.1 [Электронный ресурс]. — URL: https://datatracker.ietf.org/doc/rfc2616/ (Дата обращения 04.05.2021). Загл. с экр. Яз. англ.

14 The WebSocket Protocol [Электронный ресурс]. — URL: https://datatracker.ietf.org/doc/rfc6455/ (Дата обращения 04.05.2021). Загл. с экр. Яз. англ.

15 Representational State Transfer (REST) [Электронный ресурс]. — URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (Дата обращения 04.05.2021). Загл. с экр. Яз. англ.

16 Общие сведения ASP.NET Core MVC [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview (Дата обращения 04.05.2021). Загл. с экр. Яз. рус.

17 Внедрение зависимостей в ASP.NET Core [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/aspnet/core/fundamentals/dependency-injection (Дата обращения 05.05.2020). Загл. с экр. Яз. рус.

18 JWT Validation and Authorization in ASP.NET Core [Электронный ресурс]. — URL: https://devblogs.microsoft.com/aspnet/jwt-validation-and-authorization-in-asp-net-core/ (Дата обращения 07.05.2020). Загл. с экр. Яз. англ.

19 Проверка подлинности и авторизация в ASP.NET Core SignalR [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/aspnet/core/signalr/authn-and-authz (Дата обращения 07.05.2020). Загл. с экр. Яз. рус.

20 Методы расширения (Руководство по программированию в C#) [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/extension-methods (Дата обращения 07.05.2020). Загл. с экр. Яз. рус.

21 ASP.NET Core узла SignalR в фоновых службах [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/aspnet/core/signalr/background-services (Дата обращения 08.05.2020). Загл. с экр. Яз. рус.