05.13.18

ИССЛЕДОВАНИЕ МОДЕЛИ ОБРАЩЕНИЯ К WEB-СЛУЖБЕ ASP.NET ЧЕРЕЗ ПРОКСИ СБОРКУ НА ПРИМЕРЕ РАЗРАБОТКИ СЕРВИСА ПО АРЕНДЕ ВИДЕОФИЛЬМОВ

Н.Д. Маслов, Е.В. Попова

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» имени В. И. Ульянова (Ленина) (СПбГЭТУ «ЛЭТИ»), г. Санкт-Петербург

В статье рассматривается исследование модели соединения Web-приложения и Web-службы напрямую и через прокси. Рассматривается кроссплатформенная высокопроизводительная среда ASP.NET. При разработке используется паттерн MVC. Формирование запросов реализуется с помощью архитектурного стиля REST и протокола передачи гипертекста HTTP. Строится модель взаимодействия Web-приложения и Web-службы. Целью данной работы является исследование данной модели взаимодействия напрямую и через прокси. Были разработаны Web-приложение, являющееся сервисом по аренде видеофильмов и Web-служба, функциональное назначение которой – работа с базой данных Web-сервера. Для реализации прокси сборки было создан отдельный проект. В настройках проекта были прописаны протокол передачи данных, хост API, порт и роутинги перенаправления запросов. Исследование модели проводилось с помощью программного обеспечения Postman и инструментов браузера. Результаты исследования показывают плюсы и минусы использования прокси при взаимодействии Web-приложения и Web-службы. Данная статья поможет заказчикам, разработчикам Web-приложений выбрать модель соединения Web-приложения и Web-службы, соответствующую решаемым задачам.

Ключевые слова: ASP.NET, Web-служба, архитектурный стиль REST, API, прокси-сборка.

ВВЕДЕНИЕ

Для решения бизнес задач на современном этапе развития многие компании разрабатывают веб-приложения, используя различные технологии. ASP.NET – это платформа для разработки Web-приложений, Web-служб, Web-API на базе .Net Framework [1].

Для обмена данными между Web-службой и Web-приложением используется архитектурный стиль REST (Representational State Transfer) [2]. Приложения REST обращаются к функциям HTTP-запроса для публикации, чтения и удаления данных, тем самым используя полную функциональность HTTP CRUD (Create Read Update Delete).

Среда разработки ASP.NET MVC [3] опирается на паттерн MVC (Model-View-Controller), который соответствует архитектурному шаблону: «модель-представление-контроллер» и сочетает в себе эффективность и методы гибкой разработки.

Актуальность работы связана с нарастанием значимости обработки удаленных вызовов в Web-технологиях. При разработке больших приложений требуется быстрое и надёжное взаимодействие клиентской и серверной частей. Для получения критериальных оценок этого взаимодействия, необходимо построить модель и исследовать её на примере реального Web-приложения и Web-службы.

Целью данной работы является исследование модели взаимодействия между клиентом и службой напрямую и через прокси. Для реализации поставленной цели был разработан сервис без использования сторонних ресурсов. В научных работах по схожей тематике при создании служб и приложений на ASP.NET используются или сторонние прокси [4,5] или следующие решения: DynamicProxy от Castle [6], RealProxy от MS [7], кодогенерация на базе T4 [8,9], создание модуля с помощью PostSharp [10]. Proxy от Castle взаимодейтсвует только с виртуальными членами класса, RealProxy изменяет классы сервиса, кодогенерация эффективно используется для мелких задач, PostSharp – платный. В этих решениях заложен функционал готового прокси, что является удобным, но не гибким вариантом при решении задач, кроме того, требуется установка дополнительных пакетов. Поэтому при реализации модели создан оригинальный прокси, в котором используется встроенный пакет ASP.NET core Builder, доступный вместе со стандартными пакетами нового проекта ASP.NET.

Новизна работы заключается в связке используемых технологий: ASP.NET для разработки прокси, MVC паттерн для разработки приложения, REST-архитектура для создания Web-службы, Postman для проведения исследования. Данные технологии и результаты исследования могут послужить основой для создания программных продуктов для вновь создаваемых сервисов, и наглядно проиллюстрировать плюсы и минусы использования прокси в выбранных технологиях.

ОСНОВНАЯ ЧАСТЬ

Web-служба — это изолированный модуль кода, выполняемый на Web-сервере [11], реализуемый без графического интерфейса, с лёгкой поддержкой функциональности и масштабируемости. Связь между Web-приложением и Web-службой можно осуществить напрямую и с использованием прокси. На рисунке 1 представлена схема соединения Web-приложения с Web-службой.

Соединение через прокси имеет следующие преимущества:

- Сокрытие IP-адреса. Клиент будет иметь доступ только к прокси и не будет знать о службах, используемых на сервере, что повышает информационную безопасность приложения.
- Фильтрация запросов к серверу. Прокси позволяет осуществлять блокировку нежелательных запросов от определенных ресурсов.
- Кэширование. При повторении одинаковых запросов прокси позволяет отвечать клиентам, кэшированными запросами без повторного обращения к серверу, это увеличивает скорость ответа клиенту и снижает нагрузку на сервер.
- Маршрутизация запросов. Прокси может осуществлять маршрутизацию запросов при использовании нескольких служб. С точки зрения клиента запросы будут идти на один и тот же адрес. Помимо этого, маршрутизация позволяет упростить масштабирование и равномерно распределяет нагрузку между несколькими инстансами сервера.

К недостаткам использования прокси относятся возможность кражи личных данных пользователя при реализации доступа к прокси злоумышленником.

Web-приложение выполняет роль MVP (Minimum Viable Product) сервиса по аренде видеофильмов. В приложении пользователь может зарегистрироваться, а при наличии аккаунта - авторизоваться. После успешной авторизации пользователь заполняет форму аренды, и ему доступны разные тарифы для разных фильмов.

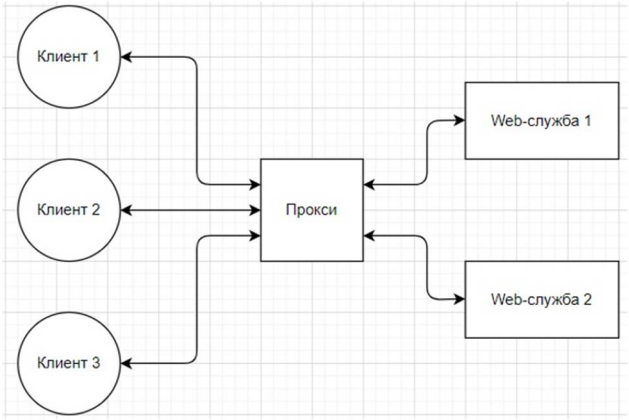

Рис. 1. Схема связи клиентов и служб через прокси

Пользователь с правами администратора может добавлять, редактировать представление и удалять видеофильмы. На основании этих технологий была создана модель взаимодействия Web-приложения и Web-службы представлена на рисунке 2.

Функциональным назначением web-службы является работа с базой данных. Служба принимает различные представления через методы GET, POST, PUT и DELETE. Логика Web-службы разделена на три контроллера:

- контроллер фильмов для действия с фильмами, получение списка фильмов, создание нового фильма исправление и удаление существующего;
- контроллер покупателя для действий с моделью базы данных покупателя, получения списка покупателей, создание нового покупателя, исправления существующего покупателя и удаления;
- контроллер аренды фильмов, предназначенного для того, чтобы пользователь мог арендовать фильм.

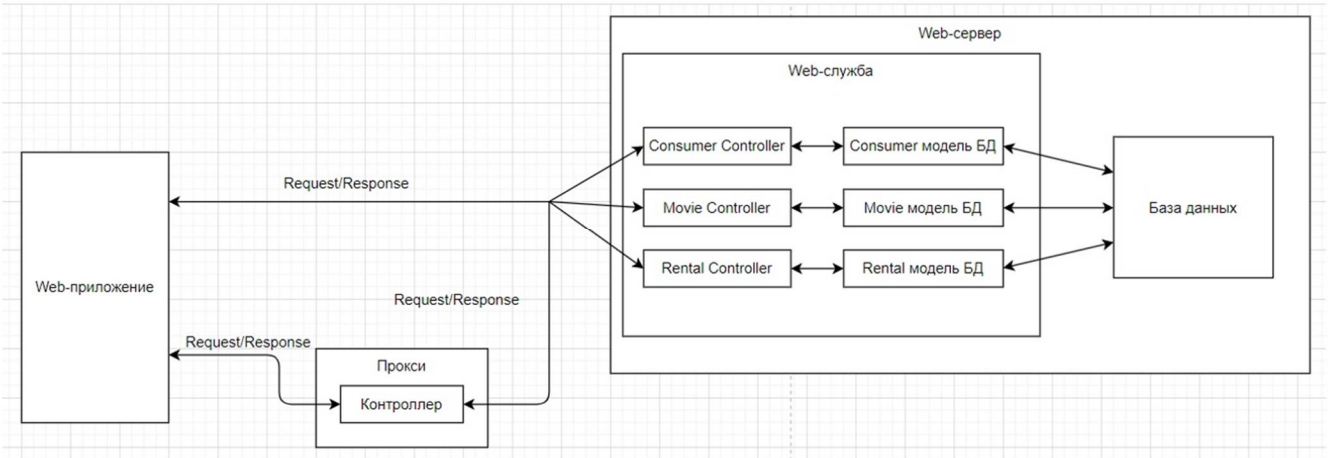

Рис. 2. Модель взаимодействия Web-приложения и Web-службы

В случае, когда запрос от web-приложения к web-службе отправляется через прокси, запрос сначала идёт в контроллер прокси, где определяется к какому контроллеру web-сервера идёт обращение и после этого происходит переадресация на web-сервер.

Приложение и созданное прокси коннектилось перенаправлением URL-адресов форм и запросов на получение данных на прокси (рис. 3).

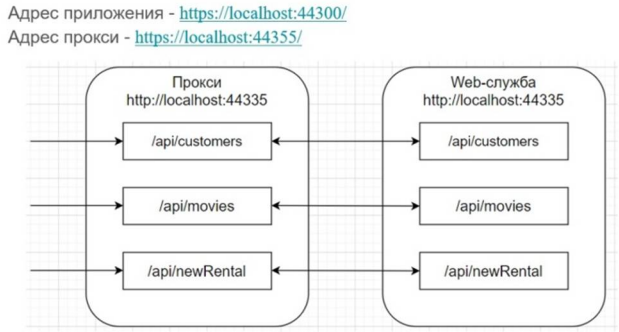

Рис. 3. Представление перенаправления запросов

Для исследования модели взаимодействия проверим время получения отклика на запросы через прокси и напрямую, возможности авторизованного пользователя, получение адреса Web-службы с учетом прокси.

ИССЛЕДОВАНИЕ МОДЕЛИ

Исследование модели проводилось на следующем аппаратном обеспечении: Windows 10, 64-разрядная (10.0, сборка 19041), ОЗУ – 16гб, процессор – Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz (8CPUs). В качестве инструмента исследования использовалось программное обеспечение Postman [12], которое позволяет производить запросы к службам, отслеживать данные запроса и его параметры, а также инструменты разработчика браузера, которые позволяют отслеживать запросы Web-приложения.

Postman, помимо совершения обычных запросов, даёт возможность написания API-тестов. Во вкладке «Tests», можно специфицировать, какие конкретно данные о запросах нужно исследовать.

Для определения скорости ответа службы при обращении к ней напрямую была написана функция, листинг которой приведен на рисунке 4. Функция sendRequest позволяет отправить тысячу запросов к службе с промежутком в сто миллисекунд и вывести усреднённое время ответа. Для тестирования запросов через прокси аналогичная функция содержит другой адрес запроса.

```
iterations = 1000;
delay = 100;
responseTimes = [];

i=0;

function sendRequest() {
    pm.sendRequest({
        url: "https://localhost:44300/api/movies",
        method: 'GET'
    }, function (err, res) {
        pm.test("Response time is " + res.responseTime, function(){
            pm.expect(err).to.equal(null);
            pm.expect(res).to.have.property('code', 200);
            responseTimes.push(res.responseTime);
        });
        if (i < iterations - 1) {
            i++;
            setTimeout(sendRequest, delay);
        }
        else {
            averageResponseTime = average(responseTimes);
            pm.test("Average response time is " + averageResponseTime + " ms; the number of iterations is " + iterations);
        }
    });
}
sendRequest();
function average(nums) {
    return nums.reduce((a, b) => (a + b)) / nums.length;
};
```

Рис. 4. Листинг функции для получения времени ответа

В связи с тем, что запросы обрабатываются локально, время ответа меньше, чем при удаленном соединении, но разница времени ответов напрямую и через прокси прослеживается.

На рисунках 5 и 6 представлены GET-запросы напрямую и с использованием прокси, реализуемые с помощью программы Postman.

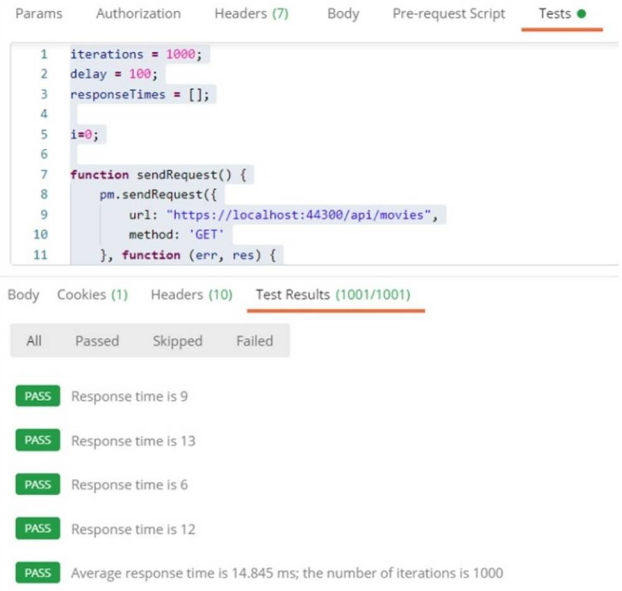

Рис. 5. GET-запросы напрямую к службе

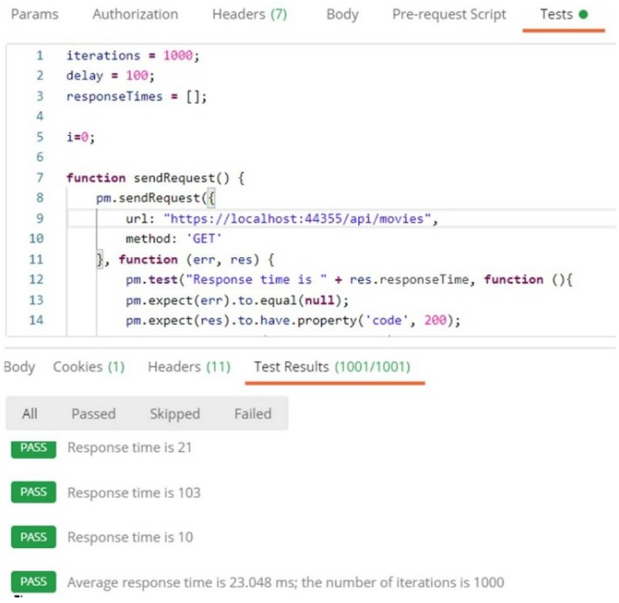

Рис. 6. GET-запросы через прокси к службе

Аналогично были протестированы POST-запросы. Программное обеспечение генерирует среднее время запросов.

Произведена проверка API на безопасность. Только авторизированный пользователь должен иметь возможно получать, изменять и удалять данные. Для того, чтобы использовать программный токен для авторизации в запросах, обратились к инструментам браузера.

После авторизации, в инструментах разработчика на вкладке «Cookie», в программном токене пользователя убрали заголовок, передающий куки в запросе, и сделали запрос к службе на получение списка пользователей. Результат неавторизованного запроса представлен на рисунке 7.

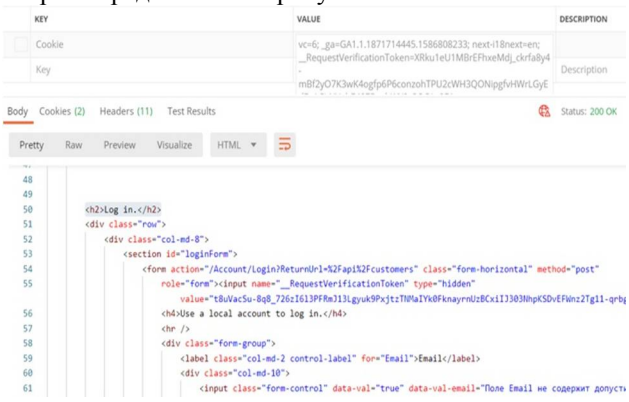

Рис. 7. Неавторизованный запрос

Как видно из тела ответа службы, неавторизированный запрос в ответ получил страницу логина, то есть произошло перенаправление на страницу с началом авторизации. Если добавить заголовок куки с токеном авторизации в ответ придёт списков всех покупателей, то есть авторизованный запрос будет реализован. Запросы к службе через прокси ведут себя аналогичным образом.

Проверим, какую информацию может получить пользователь при запросе о службе (рис.8) с использованием прокси.

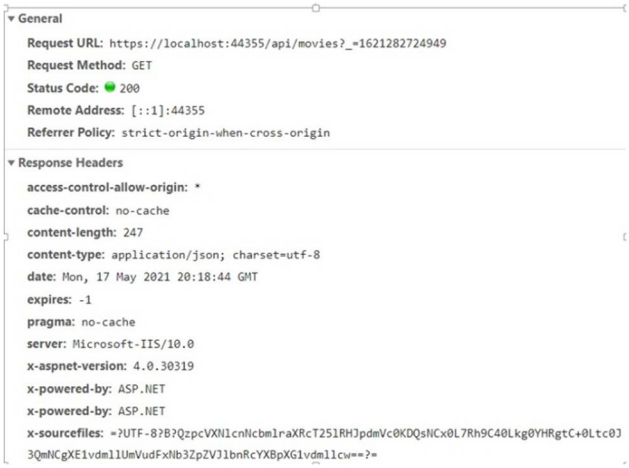

Рис. 8. Информация о запросе

Как видно из полей главной информации и заголовков запроса, пользователь не видит адреса Web-службы. Вместо этого присутствует адрес прокси, а значит сокрытие адреса службы работает. При запросе напрямую адрес службы определяется.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ

Результаты исследования представлены в таблице 1. Исследование модели показало, что в среднем ответ службы через прокси происходит медленнее, это связанно с увеличением RTT (Road trip time) [13] вследствие промежуточного узла при передаче запроса. Так как исследование проходило локально это не критично, но если запросы делать на реальных серверах, которые могу находиться в разных регионах, то разница по времени может быть ощутима.

Табл. 1 Результаты исследования

	Среднее время 1000 GET запросов	Среднее время 1000 POST запросов	Неавторизованный пользователь	Сокрытие адреса Web-службы
Напрямую	14.845 ms	23.048 ms	Перенаправление на страницу логина	Нет
Через прокси	28.405 ms	37.927 ms	Перенаправление на страницу логина	Да

При авторизованных и неавторизованных запросах результаты исследования показали, что запросы ведут себя аналогичным образом при запросе напрямую и через прокси. Авторизированный пользователь может выполнять действия с моделями базы данных, а

неавторизированный пересылается на страницу логина для авторизации.

Изучение информации, доступной пользователю о выполняемом запросе, показывает, что сокрытие адресов службы при использовании прокси работает.

ЗАКЛЮЧЕНИЕ

На основании полученных результатов можно заключить, что внедрение прокси увеличивает время запросов, что может сильно влиять на скорость работы всего приложения. Эта плата за плюсы применения прокси, такие как сокрытие адресов служб, кэширование, фильтрация и маршрутизация запросов.

Внедрение прокси должно быть обусловлено решаемыми задачами, например, упрощением работы с несколькими службами, которые расположены на разных серверах; увеличением безопасности, путём добавления дополнительных токенов при пересылке запроса через прокси; снижением нагрузки на серверы путём кэширования пользовательских запросов.

Результаты исследования могут повлиять на выбор варианта взаимодействия приложения и службы напрямую или через прокси уже на этапе проектирования программного продукта. Дальнейшие исследования могут заключаться в использовании протокола SOAP (Simple Object Access Protocol) в данной модели [14,15].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Троелсон Э., Джепкинск Ф. Язык программирования C# и платформы .Net и .Net Core [Текст] / Э.Троелсон, Ф.Джепкинск пер. с английского ООО «Диалектика», 2018. – 1316 с.

2. Патни Санджей, Pro RESTfull APIs [Текст] / Санджей Патни, США: Изд-во «Apress», 2017. – 130 с.

3. Руководство по ASP.NET MVC5 [Электронный ресурс] – 2017. – Режим доступа: https://metanit.com/sharp/mvc5/

4. Техническая документация Microsoft [Электронный ресурс] – 2021. – Режим доступа: https://docs.microsoft.com/ru-ru/documentation/

5. Zanid Haytam ASPECT ORIENTED PROGRAMMING USING PROXIES IN ASP.NET CORE [Электронный ресурс] – 2018. – Режим доступа: https://blog.zhaytam.com/2020/08/18/aspnetcore-dynamic-proxies-for-aop/

6. RealProxy Class [Электронный ресурс] – 2021. – Режим доступа: https://docs.microsoft.com/en-us/dotnet/api/system.runtime.remoting.proxies.realproxy?redirectedfrom=MSDN&view=netframework-4.8

7. AskDev.ru — спроси разработчика [Электронный ресурс] – 2021. – Режим доступа: https://askdev.ru/q/dinamicheskiy-proksi-server-castle-ne-perehvatyvaet-vyzovy-metodov-pri-vyzove-iz-klassa-326646/

8. T4 (Text Template Transformation Toolkit) Code Generation - Best Kept Visual Studio Secret [Электронный ресурс] – 2021. – Режим доступа: https://www.hanselman.com/blog/t4-text-template-transformation-toolkit-code-generation-best-kept-visual-studio-secret

9. Симан М. Внедрение зависимостей в .NET [Текст] / М. Симон – СПб.: Питер, 2014 – 464 с.

10. Less Boilerplate More Clarity [Электронный ресурс] – 2021. – Режим доступа:https://www.postsharp.net

11. Создание учетной записи-посредника веб-службы [Электронный ресурс] – 2017. – Режим доступа: https://docs.microsoft.com/ru-ru/sql/reporting-services/report-server-web-service/net-framework/creating-the-web-service-proxy?view=sql-server-2017

12. POSTMAN Learning Center [Электронный ресурс] – 2021. – Режим доступа: https://learning.postman.com/docs/publishing-your-api/documenting-your-api/

13. Determining TCP Initial Round Trip Time [Электронный ресурс] – 2018. – Режим доступа: https://blog.packet-foo.com/2014/07/determining-tcp-initial-round-trip-time/

14. SOAP — Краткое руководство [Электронный ресурс] – 20018. – Режим доступа: https://coderlessons.com/tutorials/akademicheskii/izuchite-soap/soap-kratkoe-rukovodstvo

15. SOAP Version 1.2 Part 0: Primer (Second Edition) [Электронный ресурс] – 2007. – Режим доступа: https://www.w3.org/TR/soap12-part0/

Маслов Никита Дмитриевич – студент, магистр, Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» имени В. И. Ульянова (Ленина) (СПбГЭТУ «ЛЭТИ»), г. Санкт-Петербург, e-mail: nikitamas1998@mail.ru.

Попова Елена Владимировна – к.т.н., доцент кафедры «МО ЭВМ», Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» имени В. И. Ульянова (Ленина) (СПбГЭТУ «ЛЭТИ»), г. Санкт-Петербург, e-mail: serana5@inbox.ru.

INVESTIGATION OF THE WEB SERVICE ACCESS MODEL ASP.NET USING A PROXY ASSEMBLY ON THE EXAMPLE OF DEVELOPING A VIDEO RENTAL SERVICE

N.D. Maslov, E.V. Popova
Saint-Petersburg Electrotechnical University ETU "LETI", Saint-Petersburg

Abstract – The article considers the study of the connection model of a Web application and a Web service directly and through a proxy. A cross-platform high-performance environment is considered ASP.Net. The MVC pattern is used during development. Request generation is implemented using the REST architectural style and the HTTP hypertext transfer protocol. A model of interaction between a Web application and a Web service is being built. The purpose of this work is to study this model of interaction directly and through a proxy. A Web application was developed, which is a video rental service and a Web service, the functional purpose of which is to work with a Web server database. A separate project was created to implement the build proxy. In the project settings, the data transfer protocol, API host, port and request forwarding routings were registered. A previously developed service was used as an API. The model was studied using Postman software and browser tools. The results of the study show the pros and cons of using a proxy when interacting with a Web application and a Web service. This article will help customers, developers of Web applications to choose a model for connecting a Web application and a Web service that corresponds to the tasks being solved.

Index terms: ASP.NET, Web service, REST architectural style, API, proxy assembly.

REFERENCES

1. Troelson E., Dzhepkinsk F. C # programming language and platforms.And Clean .The network core [Text] / E. Troelson, F. Dzhepkinsk per. from English LLC "Dialectics", 2018 – 1316 p.
2 .Patni Sanjay, about the API interface [Text] / Sanjay Putney, USA: Publishing house "Press" 2017, - 130 p.
3. Guide to ASP.NET MVC5 [Electronic resource] - 2017. – The access mode: https://metanit.com/sharp/mvc5/
4. Technical documentation Microsoft [Electronic resource] - 2017. – The access mode: https://docs.microsoft.com/ru-ru/documentation/
5. Zanid Haytam ASPECT ORIENTED PROGRAMMING USING PROXIES IN ASP.NET CORE [Electronic resource] - 2018. – The access mode: https://blog.zhaytam.com/2020/08/18/aspnetcore-dynamic-proxies-for-aop/
6. AskDev.ru — ask the developer [Electronic resource] - 2018. – The access mode: https://askdev.ru/q/dinamicheskiy-proksi-server-castle-ne-perehvatyvaet-vyzovy-metodov-pri-vyzove-iz-klassa-326646/
7. RealProxy Class [Electronic resource] - 2021. – The access mode: https://docs.microsoft.com/en-us/dotnet/api/system.runtime.remoting.proxies.realproxy?redirectedfrom=MSDN&view=netframework-4.8
8. T4 (Text Template Transformation Toolkit) Code Generation - Best Kept Visual Studio Secret [Electronic resource] - 2021. – The access mode: https://www.hanselman.com/blog/t4-text-template-transformation-toolkit-code-generation-best-kept-visual-studio-secret
9. Siman M. Embedding dependencies in.NET [Text] / M. Siman – SPb.: Piter, 2014 – 464 p.
10. Less Boilerplate More Clarity [Electronic resource] - 2021. – The access mode: https://www.postsharp.net
11. Creating an intermediary account of the web service [Electronic resource] - 2017. – The access mode: https://docs.microsoft.com/ru-ru/sql/reporting-services/report-server-web-service/net-framework/creating-the-web-service-proxy?view=sql-server-2017
12 POSTMAN Training Center [Electronic resource] - 2021. – The access mode: https://learning.postman.com/docs/publishing-your-api/documenting-your-api/
13. Determining TCP Initial Round Trip Time [Electronic resource] - 2018. – The access mode: https://blog.packet-foo.com/2014/07/determining-tcp-initial-round-trip-time/
14. SOAP — Quick guide [Electronic resource] - 2018. – The access mode: https://coderlessons.com/tutorials/akademicheskii/izuchite-soap/soap-kratkoe-rukovodstvo
15. SOAP Version 1.2 Part 0: Primer (Second Edition) [Electronic resource] - 2018. – The access mode: https://www.w3.org/TR/soap12-part0/

Maslov Nikita Dmitrievich – student, Master's Degree, Saint-Petersburg Electrotechnical University ETU "LETI", Saint-Petersburg, e-mail: nikitamas1998@mail.ru.

Popova Elena Vladimirovna - PhD in Technical Sciences, Associate Professor at the Department of "MO EVM", Saint-Petersburg Electrotechnical University ETU "LETI", Saint-Petersburg,e-mail: serana5@inbox.ru.