Молдавский государственный университет

Факультет математики и информатики

Департамент Информатики

**Курсовая работа**

Разработка RESTful API приложений с использованием

ASP.NET WEB API Core и языка программирования C#

**444.1 Информатика**

Выполнена студентом II курса,

специальности Информатика

**Чобану Артёмом, группа I1902**

Руководитель, преподаватель

Департамента Информатики

**Георгий Латул**

Кишинэу – 2021

Оглавление

[Введение 3](#_Toc70103998)

[Глава 1 4](#_Toc70103999)

[Глава 2 5](#_Toc70104000)

[Глава 3 5](#_Toc70104001)

[Заключение 5](#_Toc70104002)

[Литература и ссылки 5](#_Toc70104003)

[Вторая часть практики: Shadowing 5](#_Toc70104004)

[Литература: 5](#_Toc70104005)

[Заключение: 6](#_Toc70104006)

# Введение

На данный момент **веб-сервисы** – это наиболее часто создаваемый тип приложений. Без них не обходится ни один проект, даже самый мелкий. Кроме того, в крупных проектах несколько общающихся между собой веб-сервисов, объединённых в одно приложение. В последнее время идея веб-сервисов развивается в сторону микросервисной архитектуры – множества небольших веб-сервисов, объединённых в качестве модуля одного большего приложения.

В течение последних двадцати лет, было созданно несколько технологий и протоколов для создания веб-сервисов. Одним из ранее принятых способов было создание ASMX и WCF (Windows Communication Foundation) сервисов, использующих протокол SOAP (Simple Object Access Protocol).

На данный момент наиболее актуальным является создание **RESTful Web API** веб-сервисов, использующих протокол **HTTP** (HyperText Transfer Protocol). Это самый распространённый протокол передачи данных в современное время, который изначально был предназначен для передачи гипертекстовых документов.

Протокол HTTP в качестве дальнейшего развития получил расширение **HTTPS** (HyperText Transfter Protocol Secure), поддерживающего шифрование, благодаря чему является безопаснее.

**Цель данной работы** – создать пример наиболее правильной имплементации **WEB API** проекта на языке на платформе ASP.NET и с использованием языка программирования **C#**. В качестве примера будет создан веб-сервис, использующийся для новостного веб-сайта. Веб-сервис будет взаимодействовать с другими приложениями посредством HTTP **GET** и **POST** запросов, благодаря чему будет возможно создать множество разных приложений, взаимодействующих с ним.

Преимущество данного подхода – это независимость таких приложений от платформы: пока приложение правильно посылает веб-сервису HTTP запросы, удовлетворяющие его API, оно будет успешно работать вне зависимости от операционной системы и языка программирования. Подобный веб-сервис может работать как с Desktop приложениями, так и с веб-сайтами.

# Глава 1

1.1 Назначение и структура HTTP запроса.

**HTTP (HyperText Transfer Protocol)** – протокол 7 (прикладного) уровня сетевой модели OSI, изначально разработанный для доступа к гипертекстовым документам. Из-за этого, основными реализациями **клиентов** является браузеры.

HTTP реализует архитектуру “**Клиент-Сервер**”, в которой:

* **Клиент** – заказчик услуг, который создаёт подключение к серверу, посылает запрос и получает ответ.
* **Сервер** – поставщик услуг, принимающий запросы от клиента, и отправляющий ответ, если это необходимо.

Структура HTTP запроса:

* **Стартовая строка** (Request line – строка запроса) – содержит метаданные, связанные с конфигурацией запроса, такие как метод, версия HTTP, URI и другие.
* Например: “*GET /images/logo.png HTTP/1.1*”
* **Поля заголовка/Заголовок** (Header fields) – параметры запроса. Зачастую через них передаются и метаданные. Например: “Language: RU”, “Encoding: UTF-8”
* **Тело запроса** (Request body) – сами данные, отделяющиеся от заголовка пустой строкой. Тело не является обязательным. Часто данные передаются в формате **JSON**.

Последовательность сетевых транзакций “Запрос-Ответ” называется **сессией**. Вызов любого HTTP **метода** равноценен одной сессии.

Самые часто использующиеся методы HTTP:

* **GET** – запрос на получение данных. Может использоваться в качестве команд для сервера.

Пример GET запроса с параметром limit: /posts/get?limit=10

* **POST** – используется для отправки данных серверу. Данные помещаются в тело запроса, вследствие чего тело является обязательным для POST запросов.

Структура ответа:

**Код состояния** (Status Code) – трёхзначное целое число, возвращаемое сервером в ответе. Они делятся на 5 групп:

* 1xx – Информирование о процессе передачи (Informational).
* 2xx – Успешные (Successful).

Например:

200 – Ok – Запрос выполнен успешно.

* 3xx – Перенаправле ние (Redirection).
* 4xx – Клиентская ошибка (Client Error).

Например:

404 – Не найдено (Not Found) – ресурса по указанному URL не существует.

400 – Неверный запрос (Bad Request) – клиент неверно сформировал запрос.

* 5xx – Ошибка сервера (Server Error).

500 – Internal Server Error – внутренняя серверная ошибка в процессе обработки запроса.

1.2 Архитектура REST

**REST** **(Representational State Transfer)** – архитектурный стиль, использующийся для создания веб-сервисов. Веб-сервисы, следующие принципам данного архитектурного стиля, называются **RESTful**.

REST предоставляет шесть ограничений:

* Архитектура “**Клиент-Сервер**” –
* **Отсутствие состояние** (Statelessness) –
* **Кэширование** (Cacheability) – ответы сервера должны иметь указание, являются ли они кэшированными.
* **Система слоёв** (Layered System) – возможность использования нескольких веб-сервисов в таких целях как распределение нагрузки. В таком случае API также не должен меняться. Клиент не должен иметь возможность определить к какому из сервисов был послан запрос: к промежуточному узлу или напрямую.
* **Единообразие интерфейса** (Uniform Interface)

# Глава 2

ФЫВ

# Глава 3

фыв

# Заключение

фыв

# Литература и ссылки

фыв

# Вторая часть практики: Shadowing

ФВ

# Литература:

фывыфв

# Заключение:

Интернатура в компании Endava – это отличный способ вырасти профессионально, получить лучшее представление о мире Информационных Технологий, а также начать карьеру в IT.

Стажировка – это очень интересный и полезный опыт, позволяющий увидеть то, что действительно происходит при работе на реальном проекте, а не только в теории. Я увидел лучшие практики на C#, а также многие нюансы самого процесса разработки.

Кроме того, это возможность познакомиться с новыми людьми, которые к тому же являются профессионалами в своём деле. В самом начали интернатуры было сказано, что интерн может подойти к любому человеку и задать ему вопрос.

За завершение стажировки я получил сертификат об окончании.