**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА   
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ**

**ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**НИЖЕГОРОДСКИЙ ИНСТИТУТ УПРАВЛЕНИЯ – филиал РАНХиГС**

Факультет управления

Кафедра Информатики и информационных технологий

Направление подготовки / специальность: 09.04.03 Прикладная информатика

Направленность (профиль) / специализация: Информационные технологии в ГМУ

**Отчет по лабораторным работам**

|  |  |
| --- | --- |
| по дисциплине: | Лингвистическое обеспечение web-взаимодействия |
|  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
|  |  | **АВТОР** |
|  |  | Обучающийся 2 курса группы Ик-721 |
|  |  | заочной формы обучения |
|  |  |  |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_ Соколов Дмитрий Александрович |
|  |  | *(подпись) (фамилия, инициалы)* |
|  |  |  |
|  |  | **РУКОВОДИТЕЛЬ** |
|  |  | Кандидат физико-математических наук |
|  |  | *(ученая степень, ученое звание)* |
|  |  | оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. |
|  |  | *(дата защиты)* |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_ Орлов М.Л. |
|  |  | *(подпись) (фамилия, инициалы)* |

Нижний Новгород, 2025г.

Оглавление

[Практическое задание 1 3](#_Toc188559917)

[Задание 1 3](#_Toc188559918)

[Задание 2 5](#_Toc188559919)

[Задание 3 8](#_Toc188559920)

[Практическое задание 2 11](#_Toc188559921)

[Задание 1 11](#_Toc188559922)

[Задание 2 12](#_Toc188559923)

[Задание 3 14](#_Toc188559924)

[Задание 4 16](#_Toc188559925)

[Задание 5 18](#_Toc188559926)

[Задание 6 20](#_Toc188559927)

[Задание 7 24](#_Toc188559928)

[Практическое задание 3 26](#_Toc188559929)

[Задание 1 26](#_Toc188559930)

[Задание 2 31](#_Toc188559931)

[Практическое задание 4 36](#_Toc188559932)

[Задание 1 36](#_Toc188559933)

[Задание 2 37](#_Toc188559934)

[Практическое задание 5 38](#_Toc188559935)

# Практическая работа №1

**Задание №1**

Для выполнения задания создаем документ 1.xml. Для начала определяем основной тег <TOVAR>.

Затем добавил дочерние теги <NAME> и <PRACE>.

Внутрь дочерних тегов заносим информационную часть.

Указываем пролог - версию языка XML, кодировки документа и автономности. Необходимо указать кодировку UTF-8 для корректного отображения русского языка.

Результат работы - текстовое содержимое документа (рис.1):

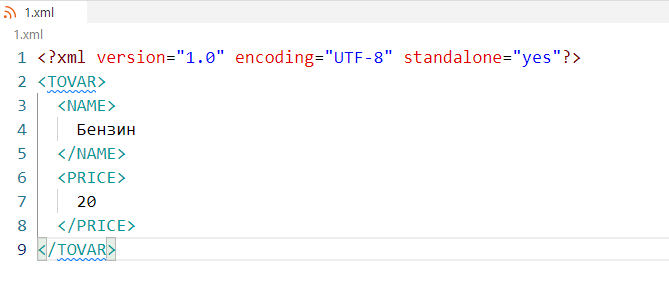


Рис. 1 Текстовое содержимое документа 1.xml

Далее проверяем структуру файла в браузере (рис.2).

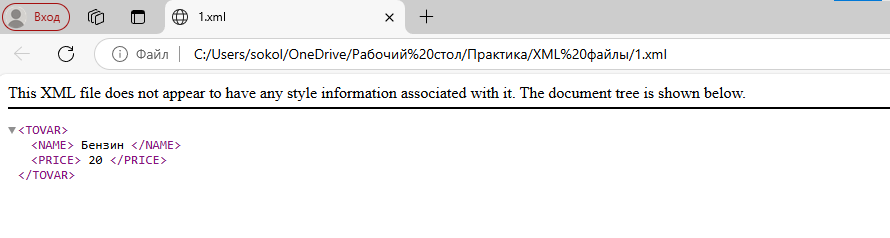


Рис. 2 Текстовое содержимое документа 1.xml

**Задание №2**

Для упрощения создания xml документов устанавливаем приложение XmlPad (рис.3).

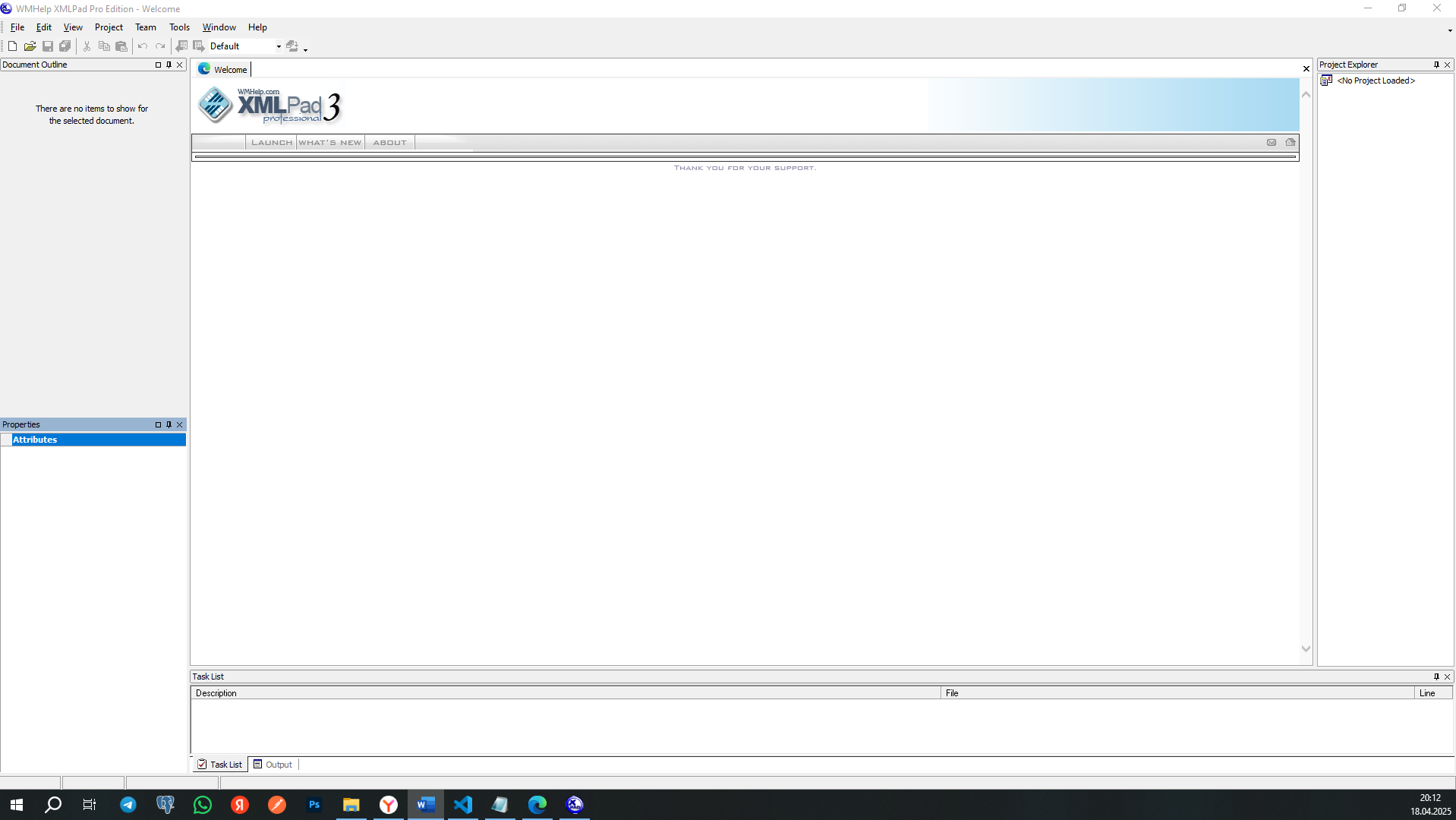
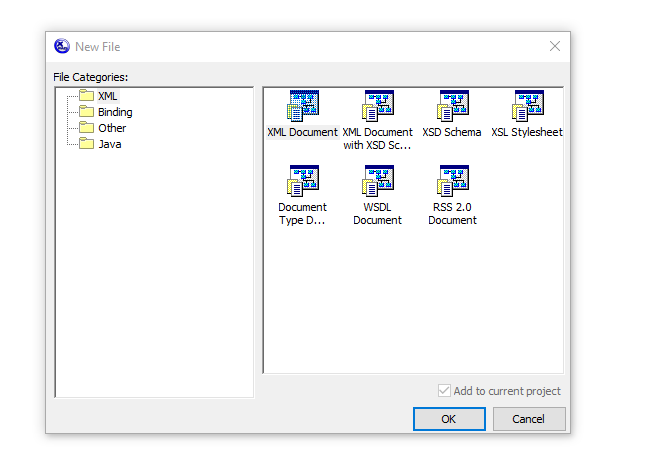
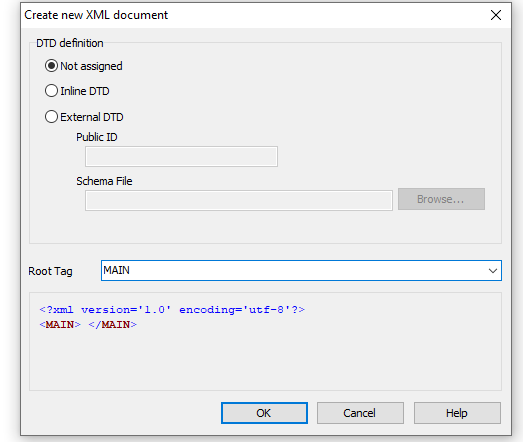


Рис. 3 Редактор XmlPad

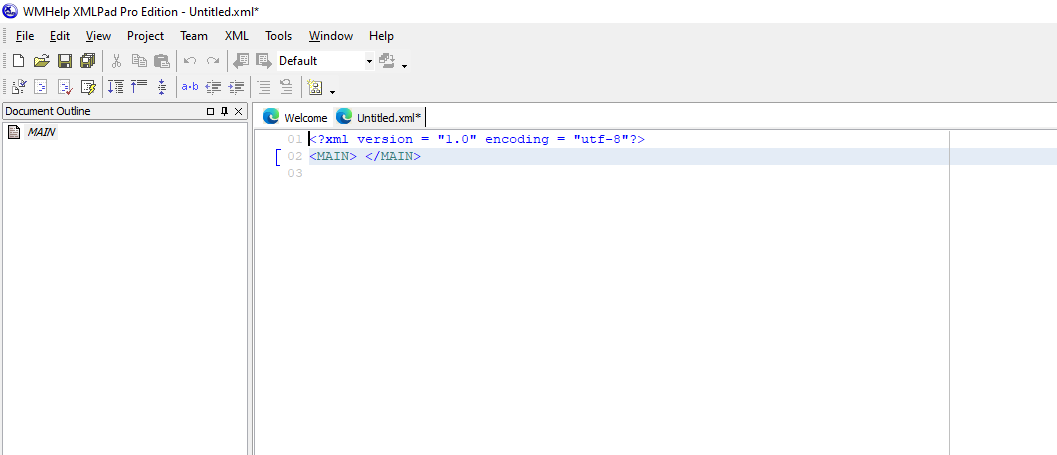
Далее создаем новый файл через контекстное меню, выбрав категорию XML (рис.4).



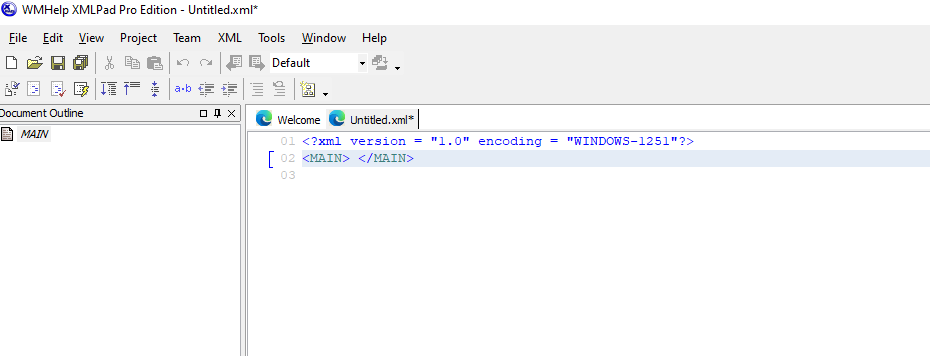
Затем указываем дополнительные параметры и главный тег <MAIN> (рис.5).



Проверяем созданный документ (рис.6).

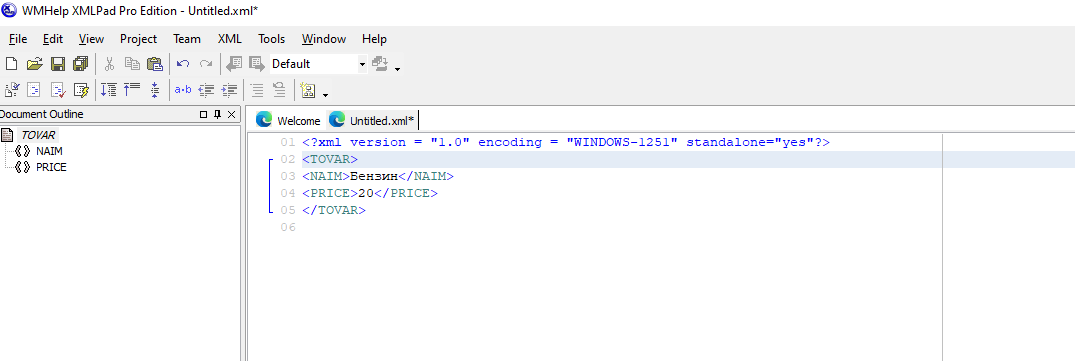


Меняем кодировку следуя заданию через пункт меню «Advanced Save Option» (рис.7).

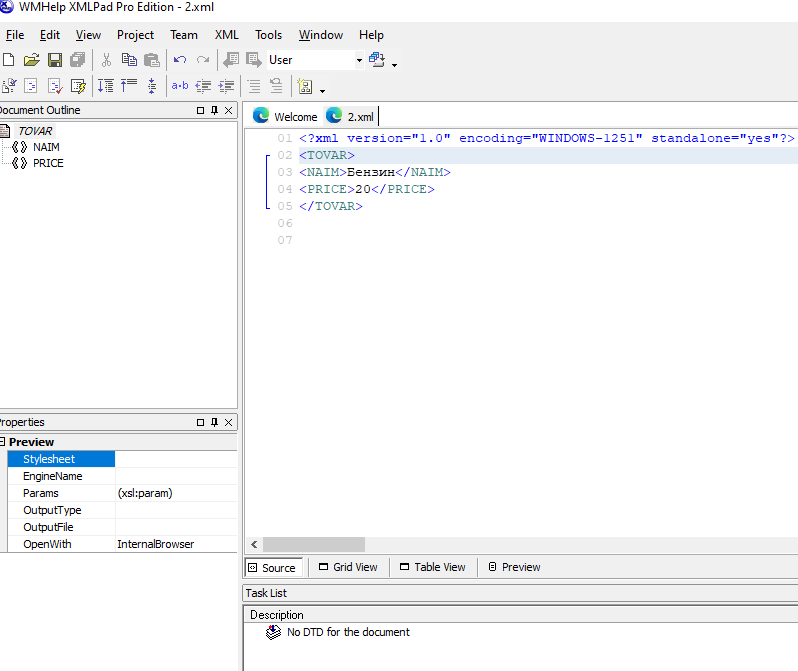


Проверяем изменения (рис. 8).

В первой строке прописываем атрибут standalone, удаляем тег <MAIN> и добавляем разметку (рис. 9).



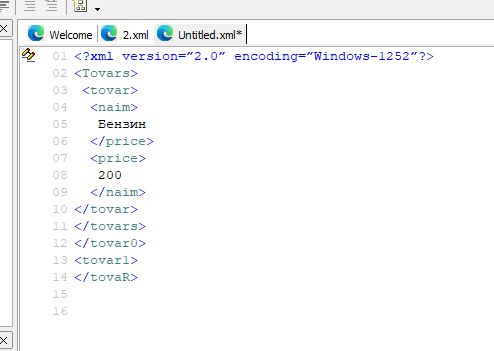
Выполняем проверку структуры документа с помощью программы XmlPad по алгоритму XML => Validate. Проверяем task list, данная надпись означает верное заполнение документа (рис. 10).



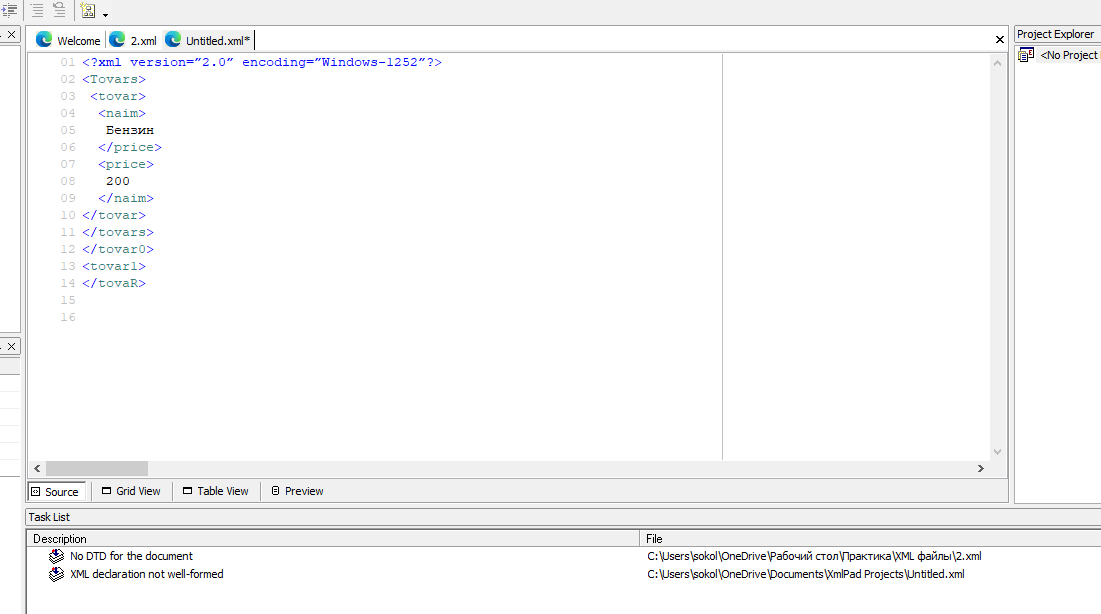
В конце сохраняем документ 2.xml в папку.

**Задание №3**

Создаем новый документ и добавляем содержимое из задания (рис. 12).



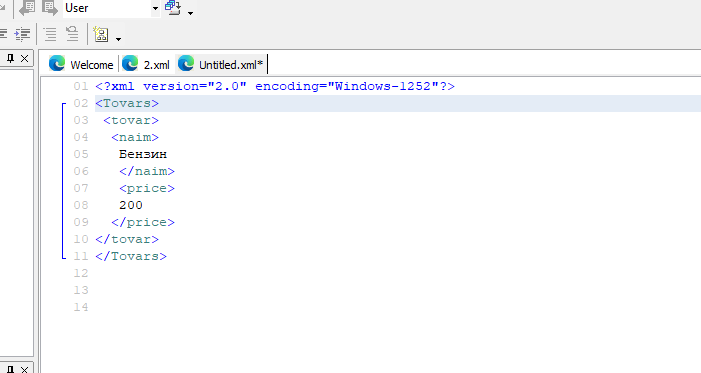
Проанализируем ошибки с помощью программы XmlPad, обращаем внимание на Task List (рис. 13).



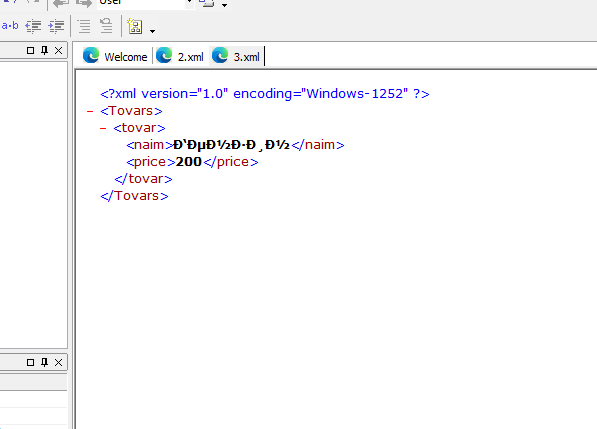
Устраняем стандартную ошибку псевдокавычек и получаем следующую ошибку из-за несоблюдения вложенности (рис. 14).



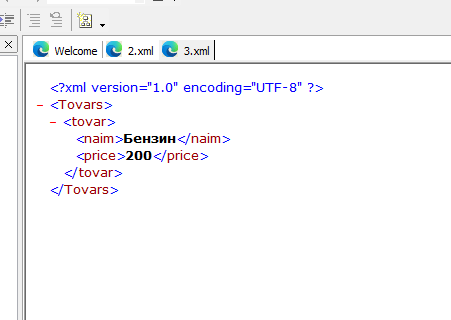
Анализируем разметку, анализируем, учитывая правила, что есть основной тег, все теги должны закрываться и совпадать по имени (регистр тоже важен). После устранения ошибок смотрим на результат (рис. 15).



Проверим вид документа с помощью вкладки Preview, предварительно изменив версию на 1.0, т.к 2.0 выдает ошибку (рис. 16).



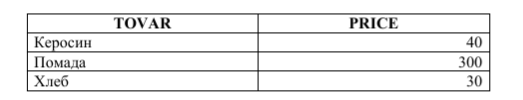
Такой результат из-за кодировки, меняем на UTF-8, поддерживающая русские символы (рис. 17).



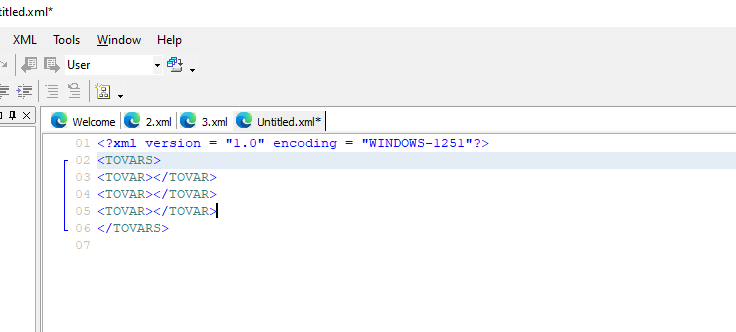
**Практическая работа №2**

**Задание №1**

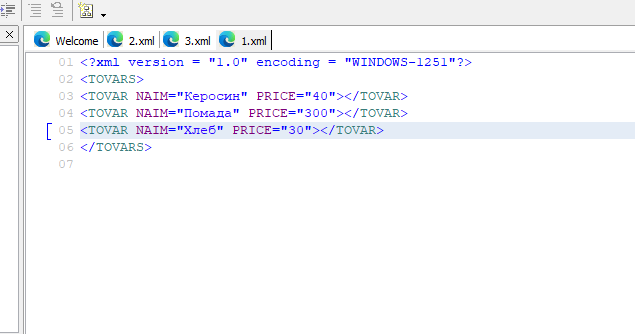
Создадим файл xml, который будет содержать следующие данные (рис. 18):



Создаем базовую структуру с тегом TOVARS, меняем кодировку, добавляем тег TOVAR по кол-ву товаров (рис. 19).



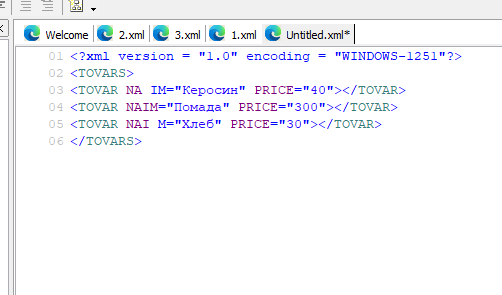
Затем добавляем атрибуты NAIM и PRICE в 3 товара (рис. 20).



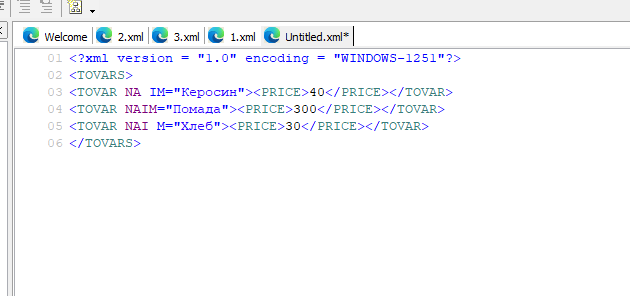
Не забываем провести валидацию, убеждаемся, что ошибки отсутствуют.

**Задание №2**

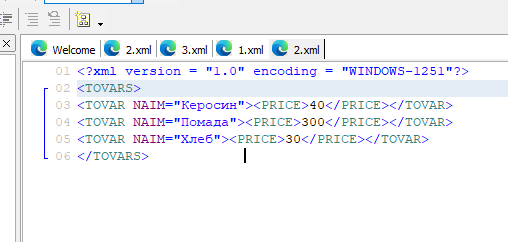
Создаем документ следующей структуры (рис.21).



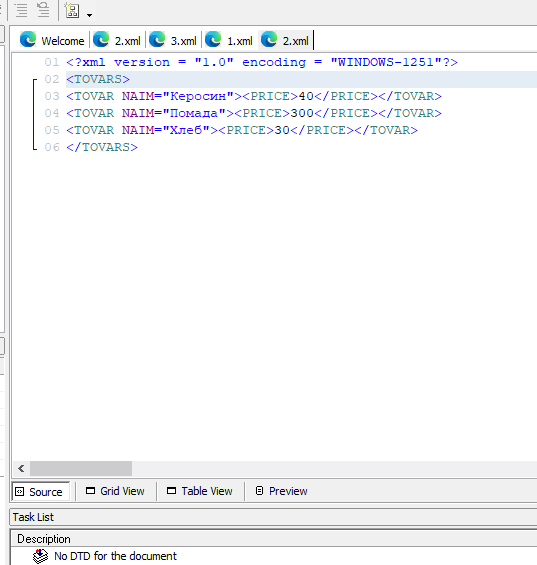
Меняем атрибут PRICE, на вложенный тег (рис. 22):



Получаем ошибку форматирования, указывающую на 3 строку, корректируем атрибуты NAIM, убираем лишние пробелы (рис.23).



Убеждаемся в отсутствии ошибок (рис. 24).

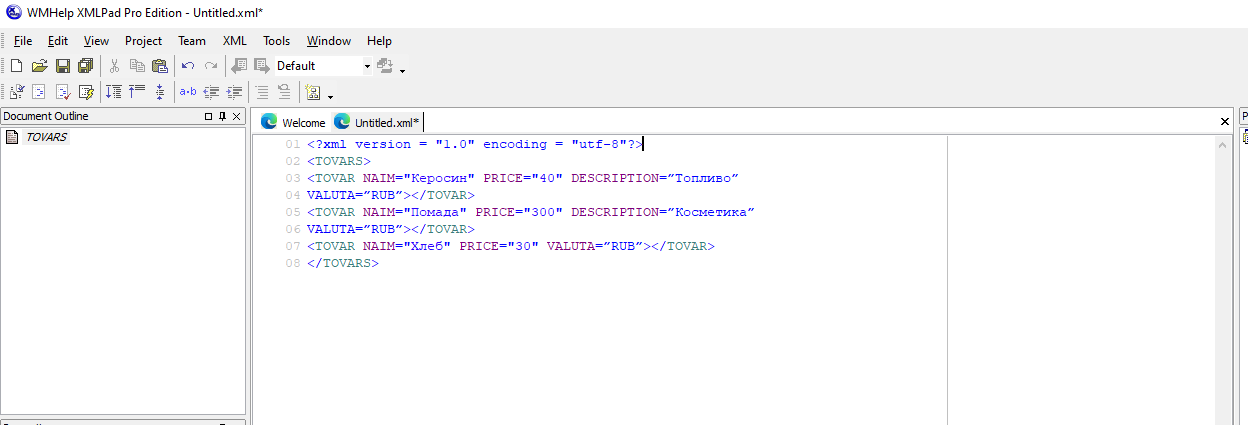


**Практическая работа №3**

**Задание №1**

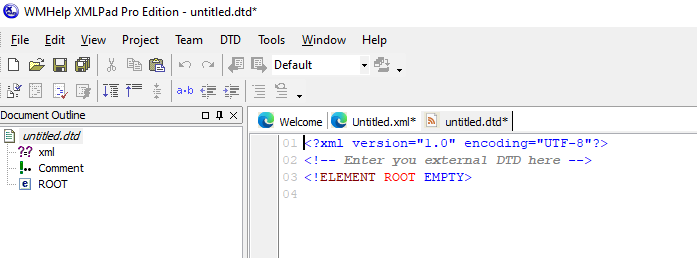
Создаем новый документ и структуру из задания - TOVARS и TOVAR– задать в виде элементов, PRICE, NAIM, DESCRIPTION и

VALUTA – в виде атрибутов (рис. 25).

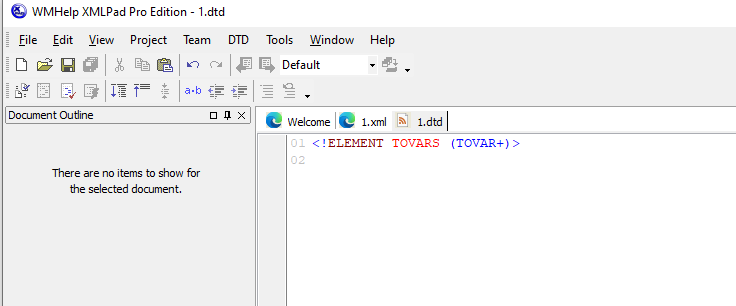


Далее создадим схему DTD. DTD (Document Type Definition) — это способ определения структуры XML-документа. С помощью DTD можно задать правила, которым должен соответствовать XML-документ: какие элементы в нем допустимы, в каком порядке они могут встречаться, какие атрибуты допускаются у элементов и т. д.

Для создания схемы DTD в программе XMLPad требуется создать новый документ, с типом Document Type Difinition (рис. 26)



Для корректной работы схемы требуется удалить всю информацию и ввести заново. Ввод информацию в схему начинается с указания корневого элемента и всех его дочерних элементов, при этом получается схема (рис. 27).

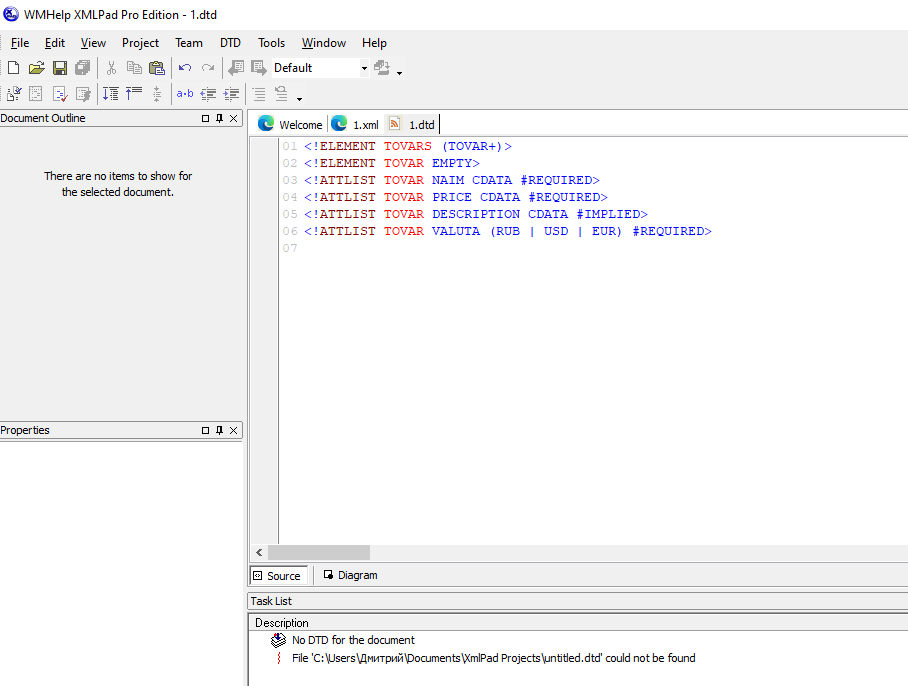


Было введено имя корневого тега TOVARS и имя дочернего элемента (TOVAR).

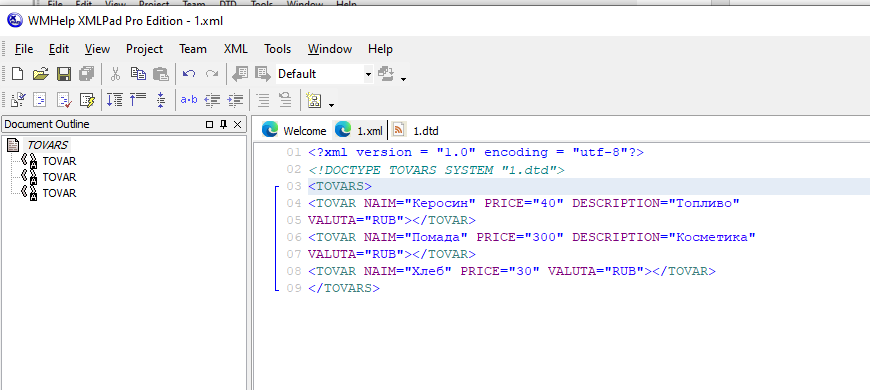
Символ + после имени дочернего элемента означает, что он может повторяться несколько раз и будет включен в документ не менее одного раза.

Затем требуется задать атрибуты NAIM, PRICE, DESCRIPTION, VALUTA

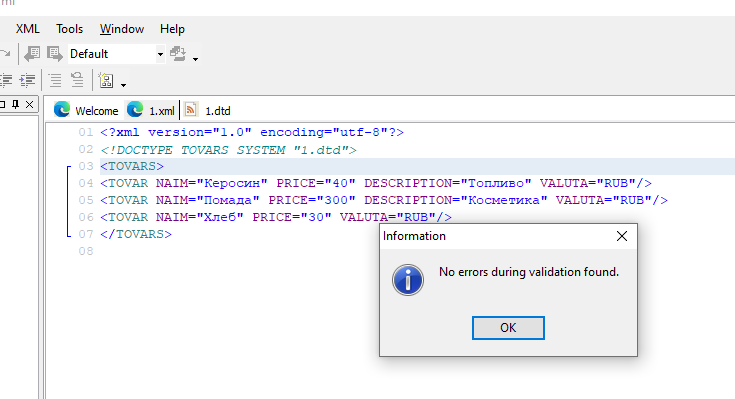
элемента TOVAR, причем атрибут DESCRIPTION являются необязательным, атрибут VALUTA является обязательным и может принимать одно из 3-х значений (RUB, USD, EUR). Вид схемы показан на рис. 28.



Теперь необходимо подключить схему к документу 1.xml, добавляем в заголовке <!DOCTYPE TOVARS SYSTEM "1.dtd"> (рис. 29).

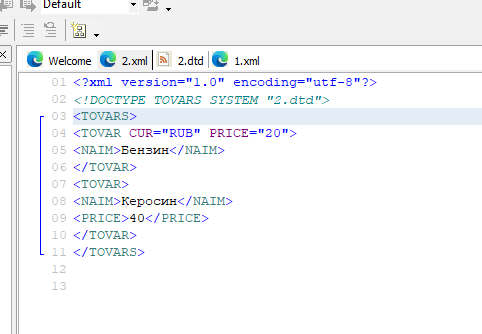


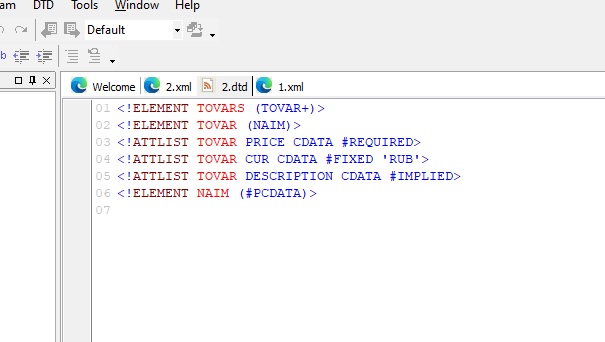
После подключения делаем валидацию и получаем сообщение об отсутствии ошибок т.к разметка файла xml соответствует схеме DTD (рис. 30)



**Задание №2**

Проверяем схему на рис. 31 на соответствие схеме на рис.32.

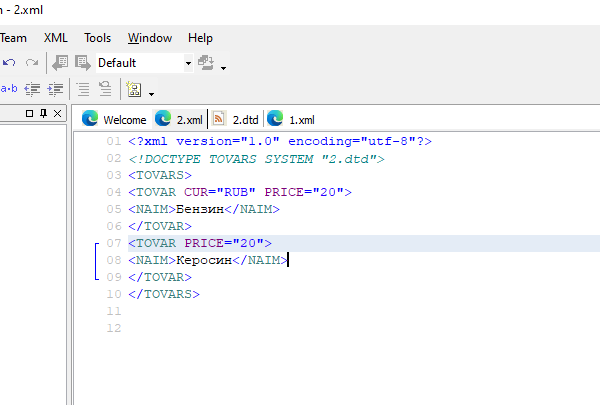


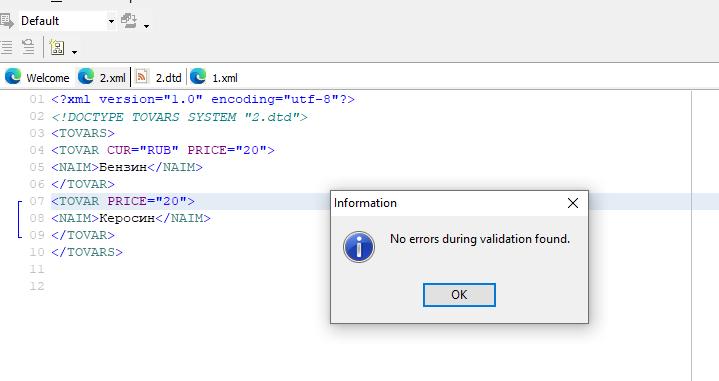


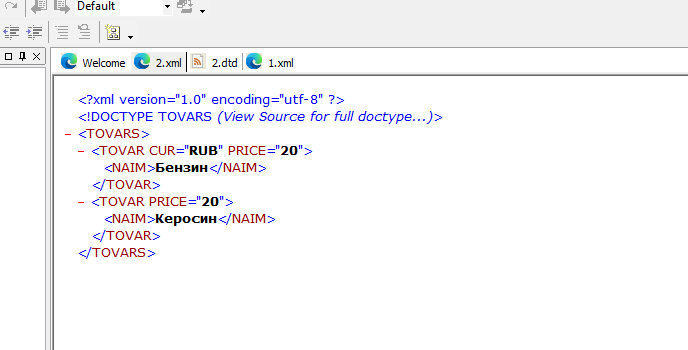
Выполнив валидацию, получаем множество ошибок (рис. 33).



Видим, что требуется атрибут PRICE для товара, а дочерний элемент прайс нам не нужен, т.к. в схеме указан именно атрибут, исправляем, делаем валидацию, просматриваем документ (рис. 34-36).







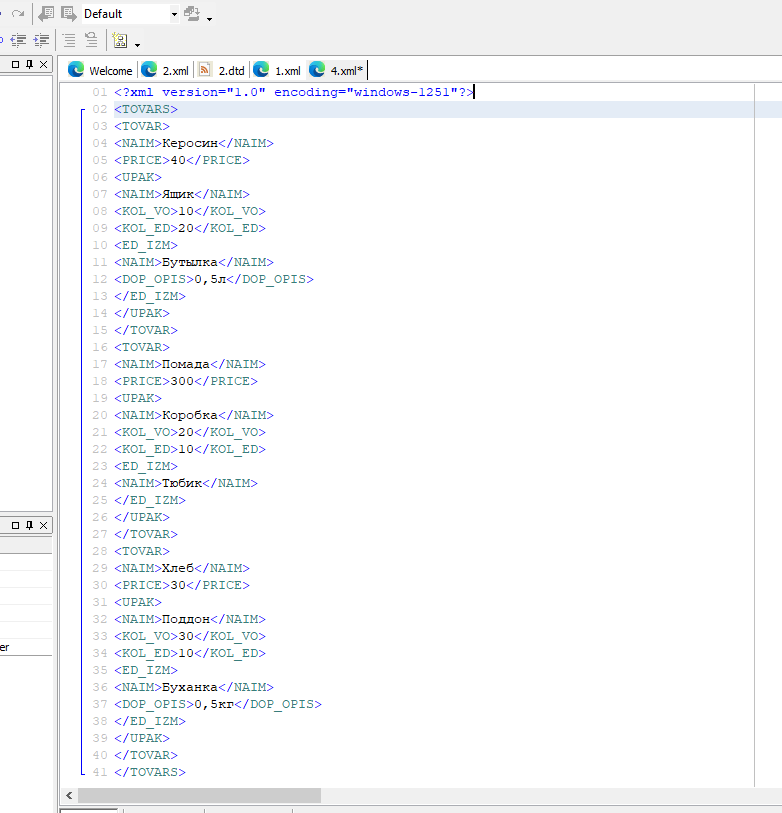
**Практическая работа №4**

**Задание №1**

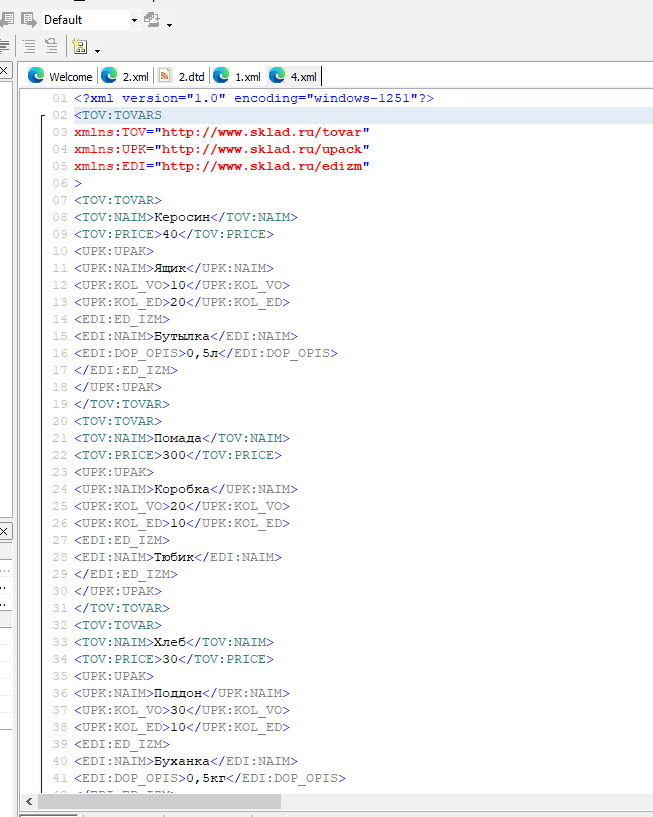
Создаем документ с применением пространства имен. В документе будет храниться информация о наличии товара на складе, каждый из товаров характеризуется наименованием (NAIM), ценой (PRICE) и упаковкой. Каждая упаковка характеризуется наименованием (NAIM) и количеством единиц товара в упаковке (KOL\_ED) и количеством упаковок в наличии (KOL\_VO). Также каждая упаковка характеризуется единицами измерения товара (ED\_IZM), в свою очередь единицы измерения товара (ED\_IZM)характеризуются наименованием (NAIM) и дополнительным описанием (DOP\_OPIS).

Дополнительное описание является необязательным элементом.

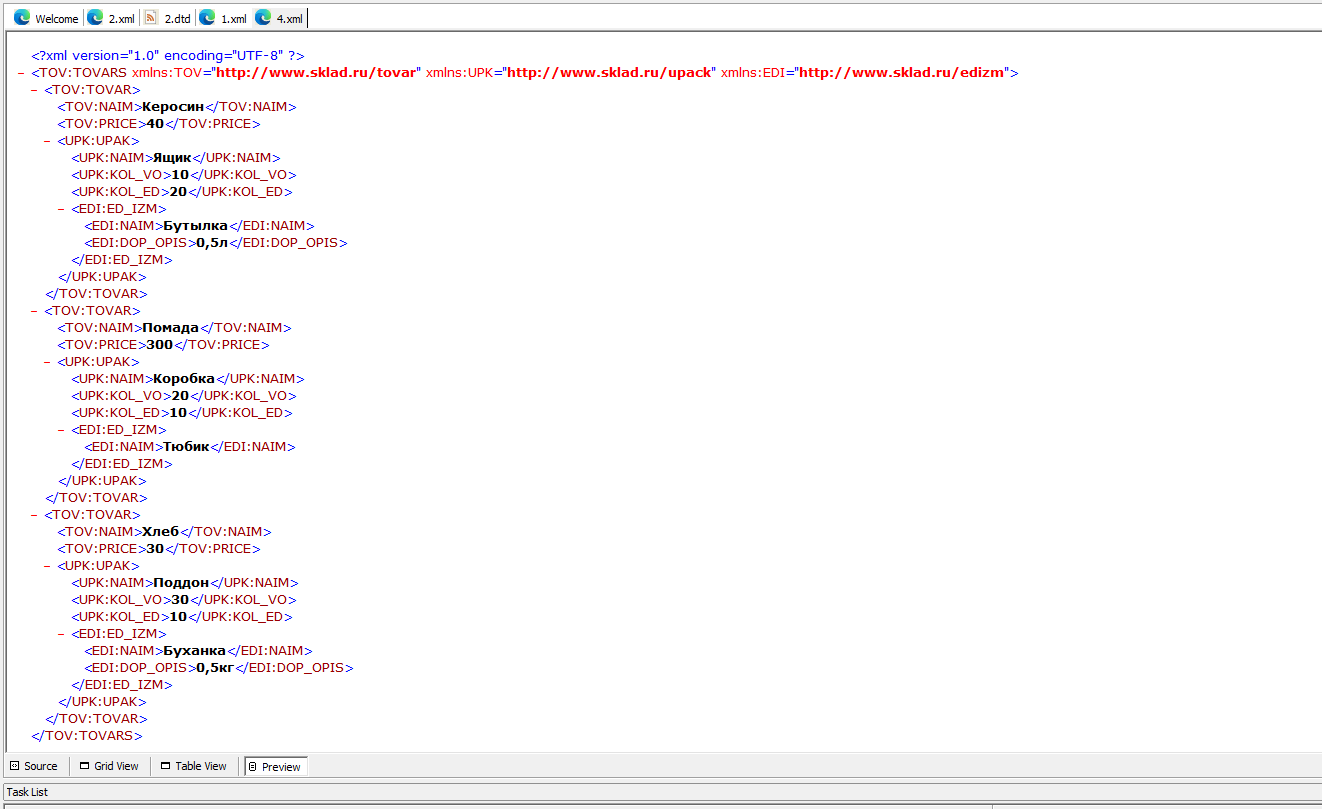
Создадим документ без пространства имен (рис. 37).



Получается много одинаковых элементов, во избежание ошибок, добавим пространство имен, документ примет вид указанный на рисунке 38.



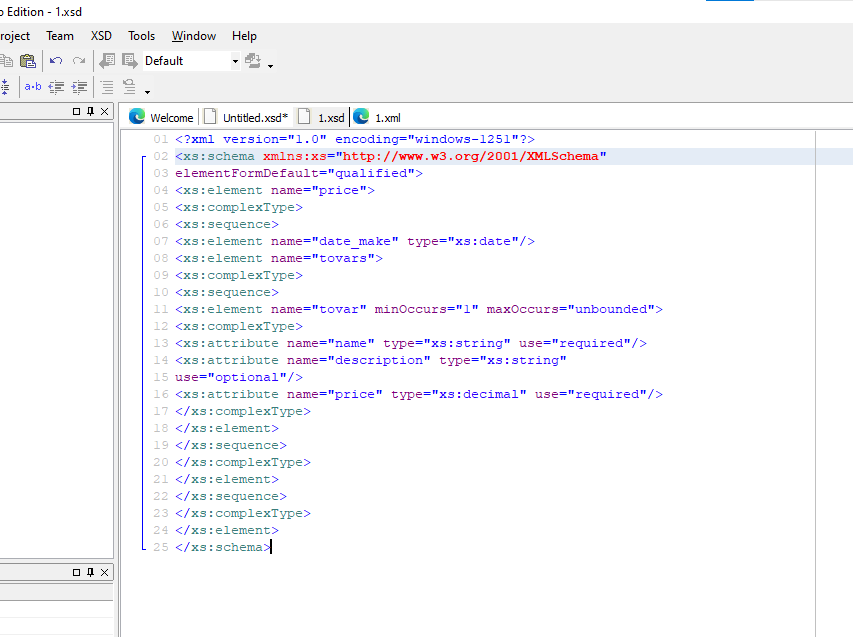
Теперь каждый документ относится к своему пространству имен. Просмотрим документ (рис. 39)



**Практическая работа №5**

**Задание №1**

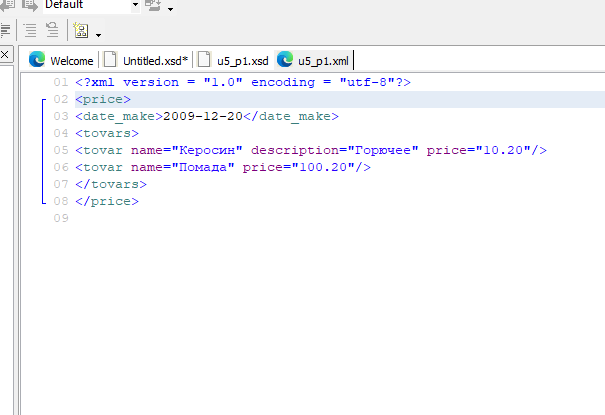
Требуется создать для опубликования на сайте прайс-листа товаров документ XML на основе схемы XSD. Схема представлена ниже (рис. 40):



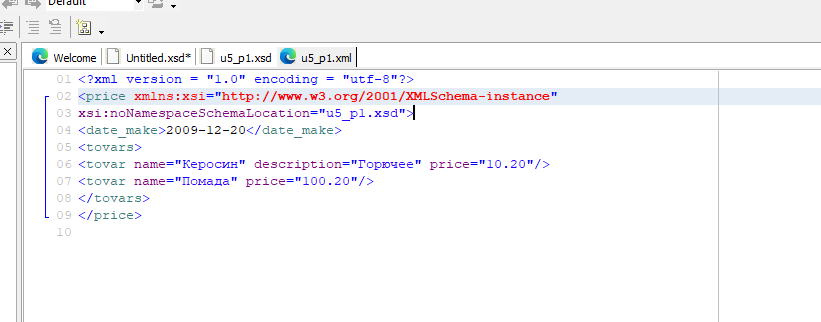
XSD (XML Schema Definition) — это язык, с помощью которого описывают структуру, содержимое и правила XML-документа. Проще говоря, это «инструкция», как должен выглядеть правильный XML-файл: какие элементы разрешены, в каком они порядке, какие у них типы данных, какие значения допустимы и т.д.

XSD — это более современная и мощная альтернатива DTD.

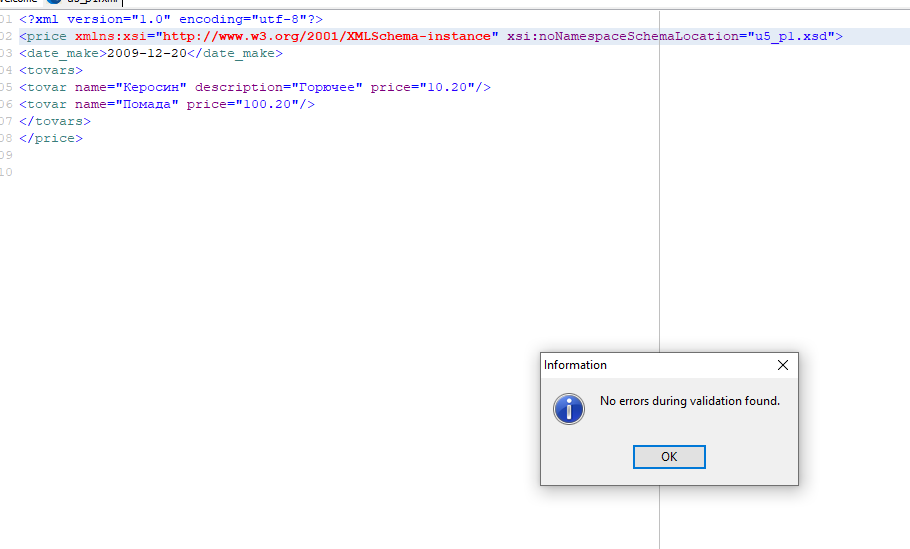
На основе данной схемы создаем XML документ (рис. 41):



Затем подключаем схему (рис. 42)



Теперь можно проверить документ xml на ошибки, т.к схема подключена (рис. 43).



Ошибки отсутствуют, но для проверки, сознательно внесем неверные значения в PRICE, например добавим букву (рис. 44).

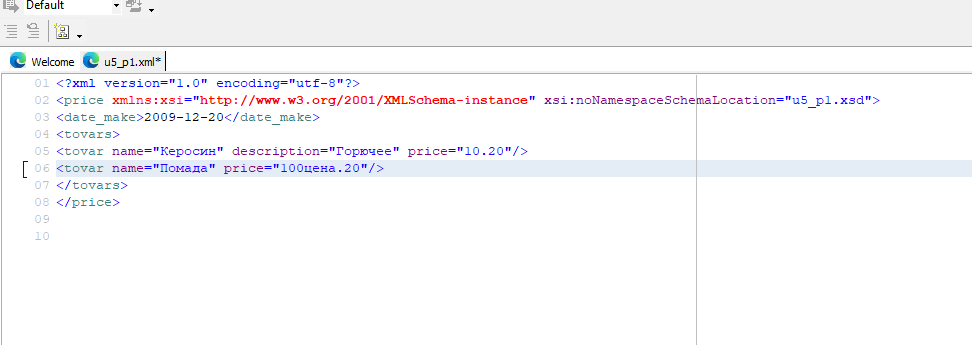
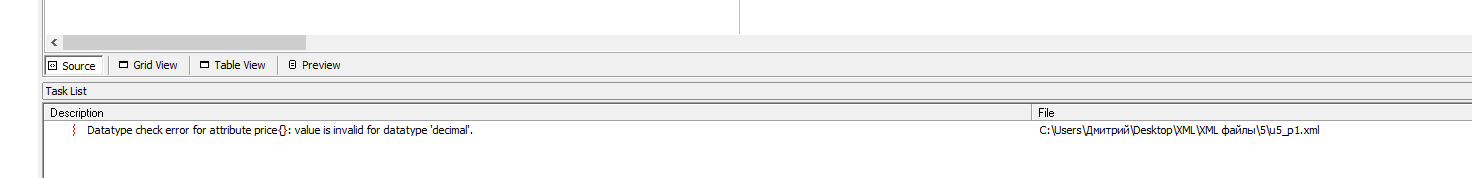
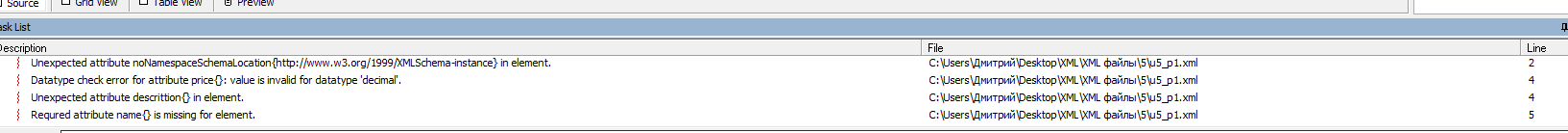


Схема работает, т.к. получаем ошибку (рис. 45) о несоответствии типов.



**Задание №2**

Схема XML соответствует схеме задания 1, но файл xml (рис. 46) отличается и после валидации получаем много ошибок, которые необходимо исправить (рис. 47).



Первая ошибка расположена на строке 2 – пространство имен указано неверно (должен быть указан 2001, а не 1999). После исправления количество ошибок не уменьшилось. Далее требуется исправить ссылку на пространство имен, правильное название - XMLSchema-instance После исправления ошибки и проверки содержимого, вид документа показан на рис. 48 Видно, что количество ошибок уменьшилось.



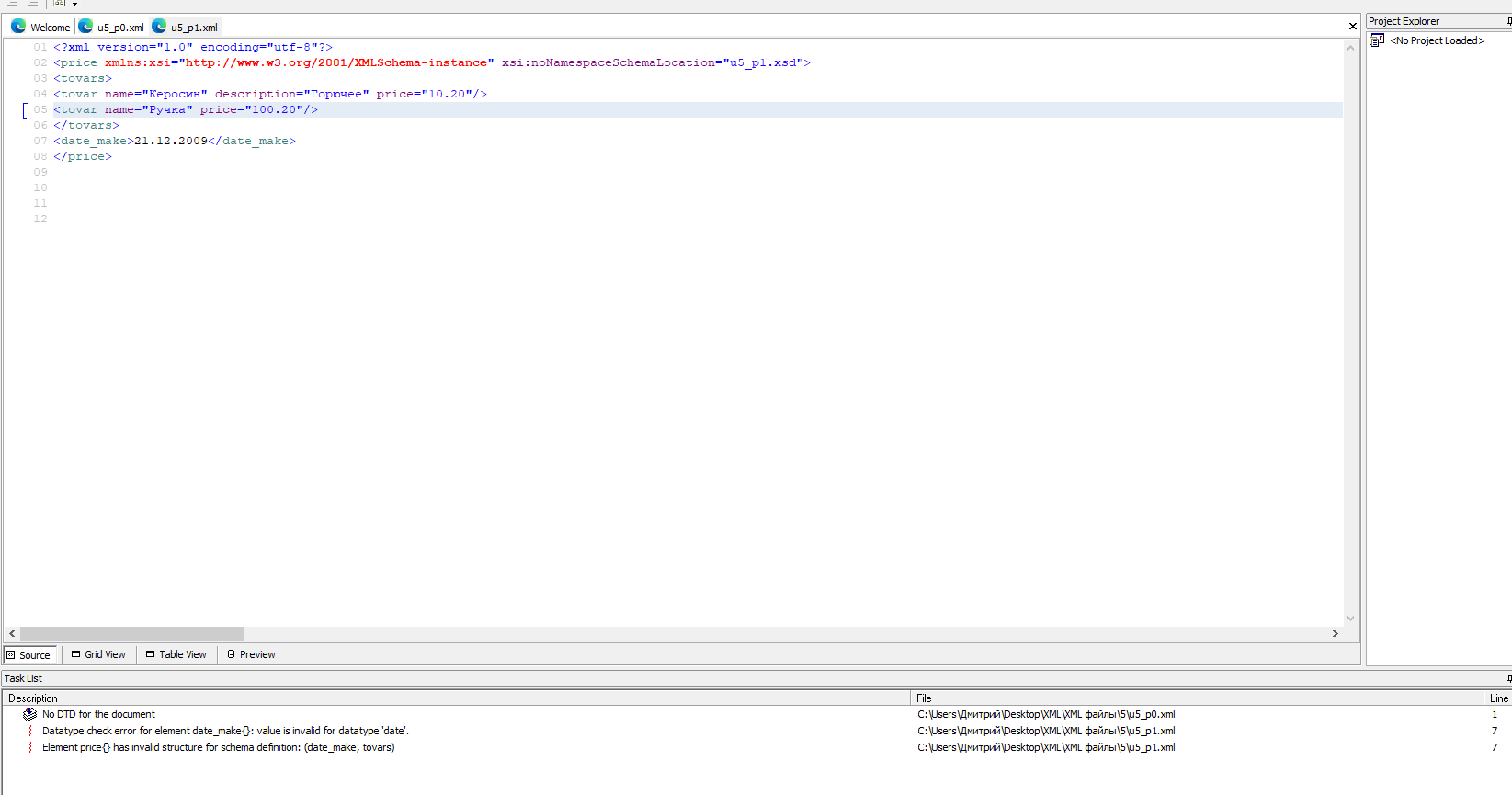
Одна из ошибок вызвана тем, что на строке 5 в атрибуте price информационное содержимое не соответствует числовому типу (разделитель целой и дробной части указан –, а не точка). После исправления этой ошибки можно исправить имя атрибута descritton на

description . После исправления этих ошибок и дополнительной проверки документа количество ошибок изменилось (рис. 49)

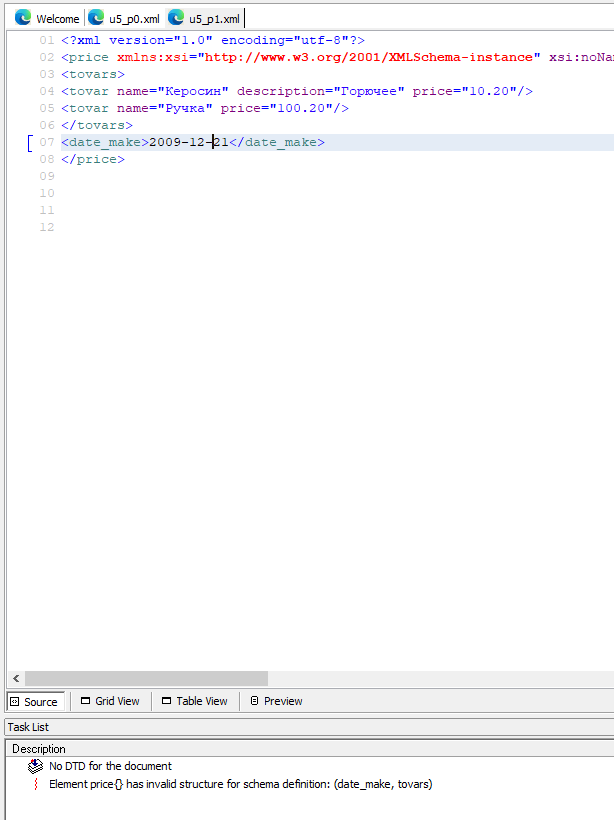


Следующая ошибка возникла на строке 10 – нет обязательного атрибута с именем name.

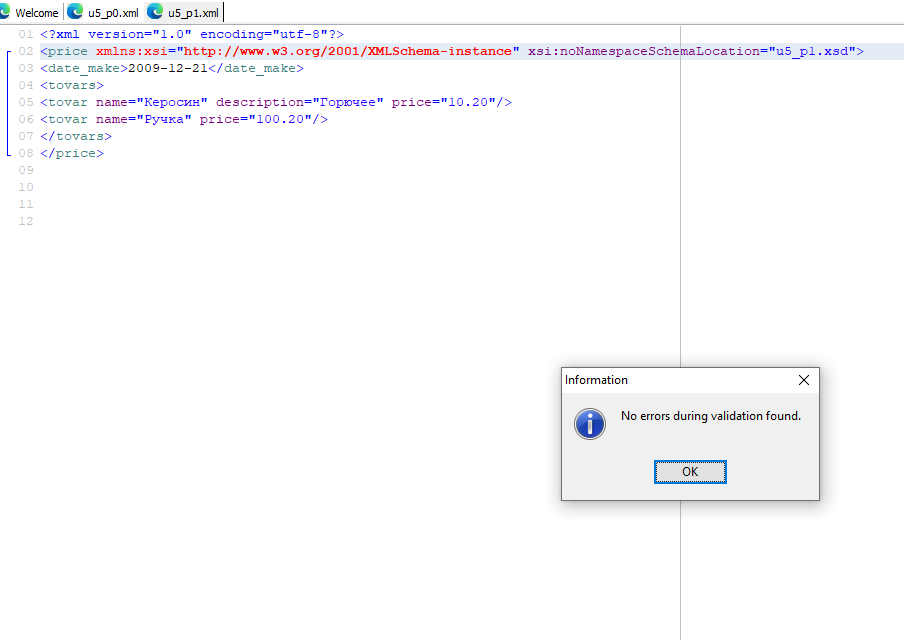
Требуется добавить этот атрибут и заново проверить содержимое документа. В результате проверки осталось 2 ошибки на строке 8 (рис. 50).



Одна из ошибок вызвана неправильным форматом даты – дата представлена в формате дд.мм.гггг в то время как требуется – гггг-мм-дд После изменения формата даты и проверки ошибок осталась одна ошибка (рис. 51)



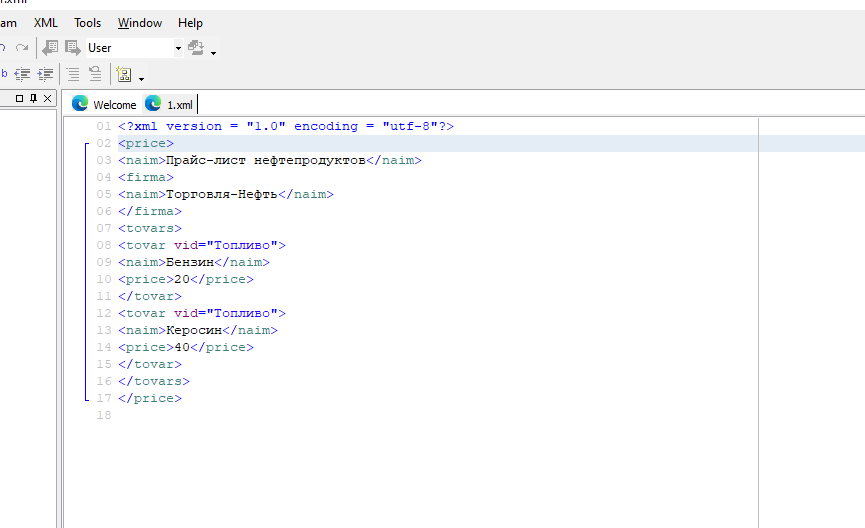
Ошибка вызвана тем, что элемент date\_make должен быть указан «выше» по тексту документа, чем элемент tovars. После изменения положения данного элемента документ соответствует схеме (рис. 52)



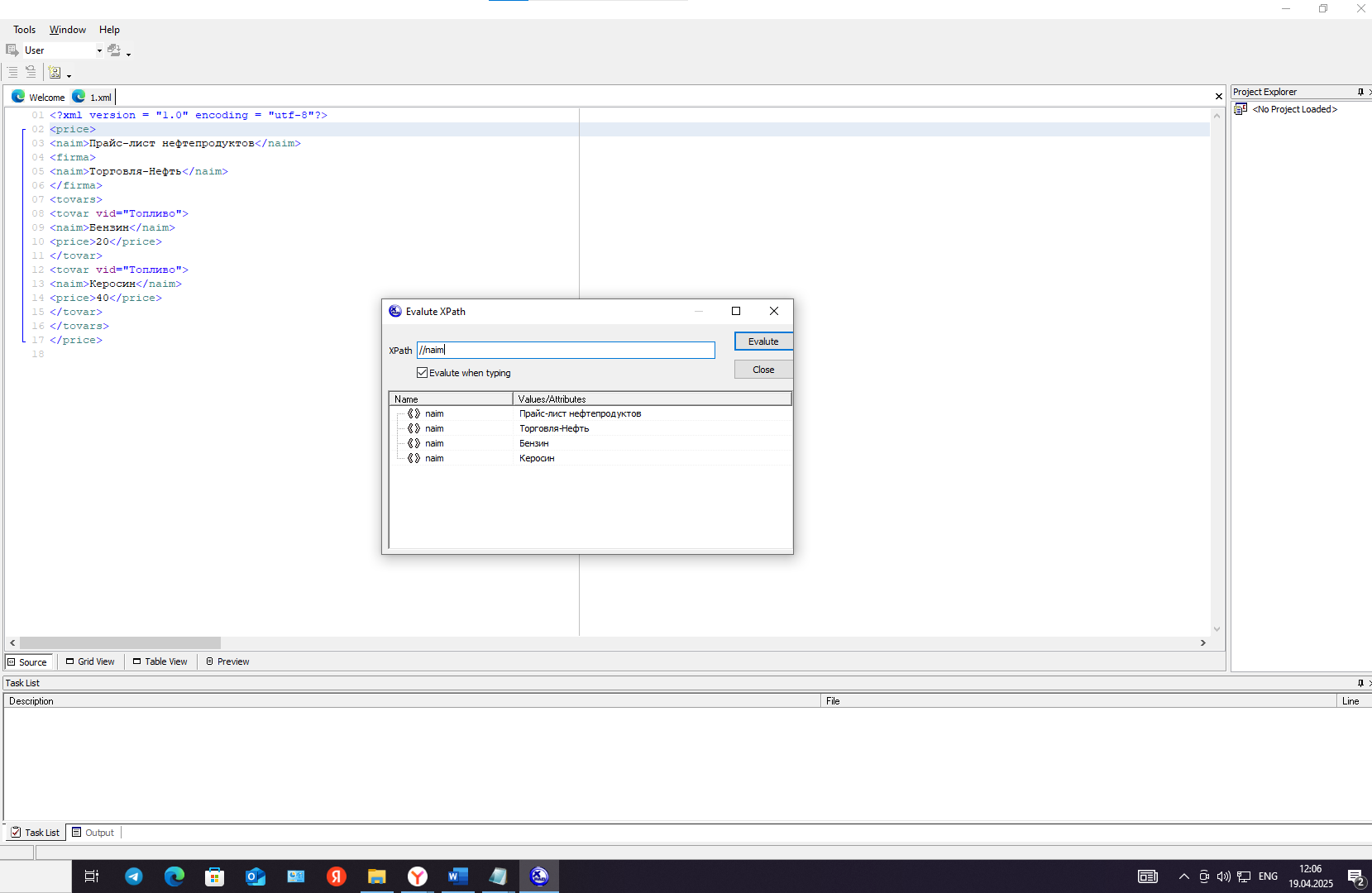
**Практическая работа №6**

**Задание №1**

Требуется найти все элементы с именем naim в документе 1 (рис. 53):



Для поиска в документе с применением XPATH выражения требуется выполнить пункт Evaluate XPATH меню XML. При этом откроется окно для ввода XPATH выражения, показанное на рис. 54

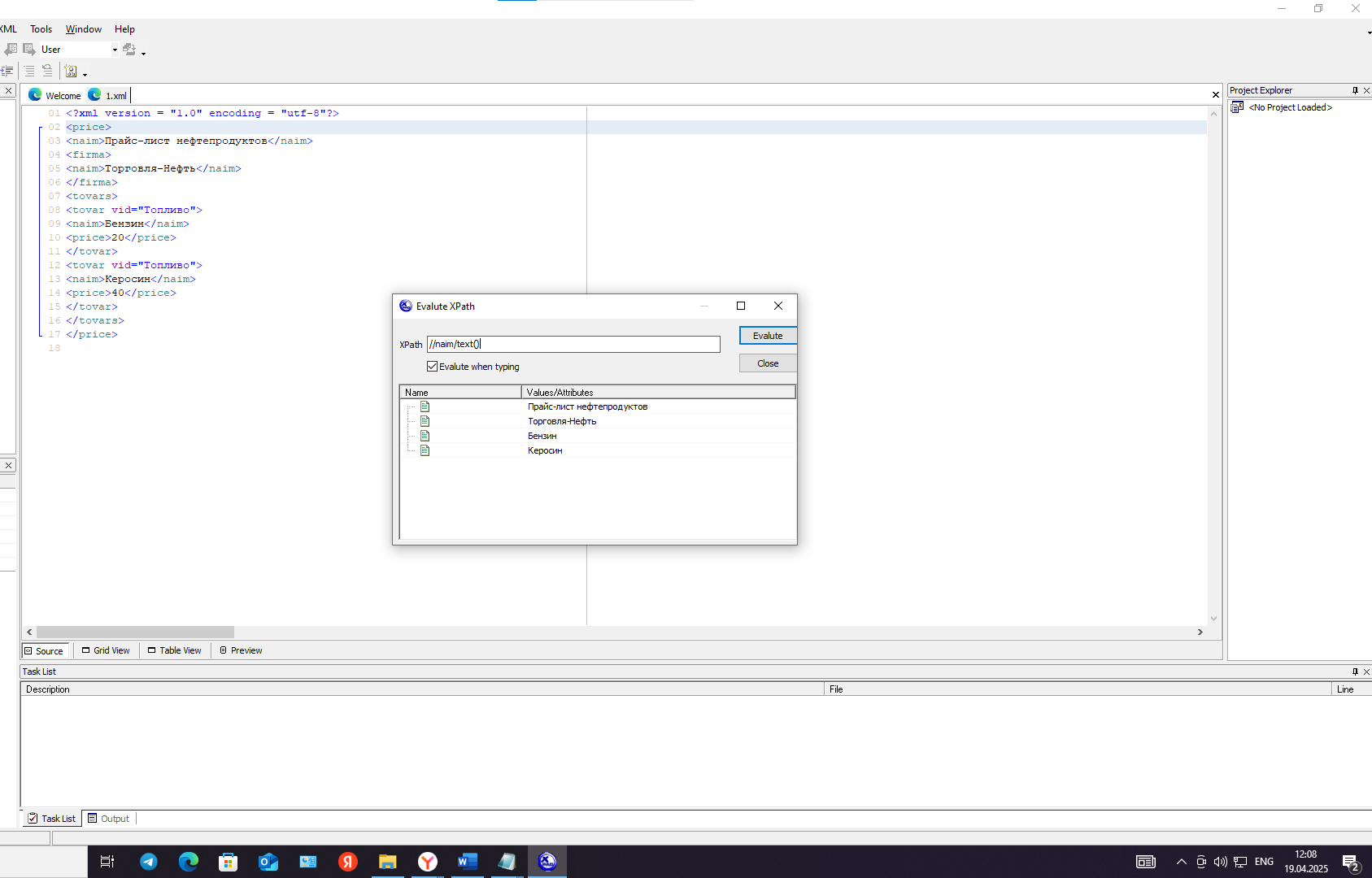


Полученный результат совпадает с ожидаемым, из чего можно сделать вывод о правильности решения задачи.

**Задание №2**

Для Документа 1 из задания 1 требуется найти информационное содержание всех элементов naim.

Начало выражения соответствует выражению из задания 1.Выражение принимает следующий вид: //naim/text() (рис. 55)

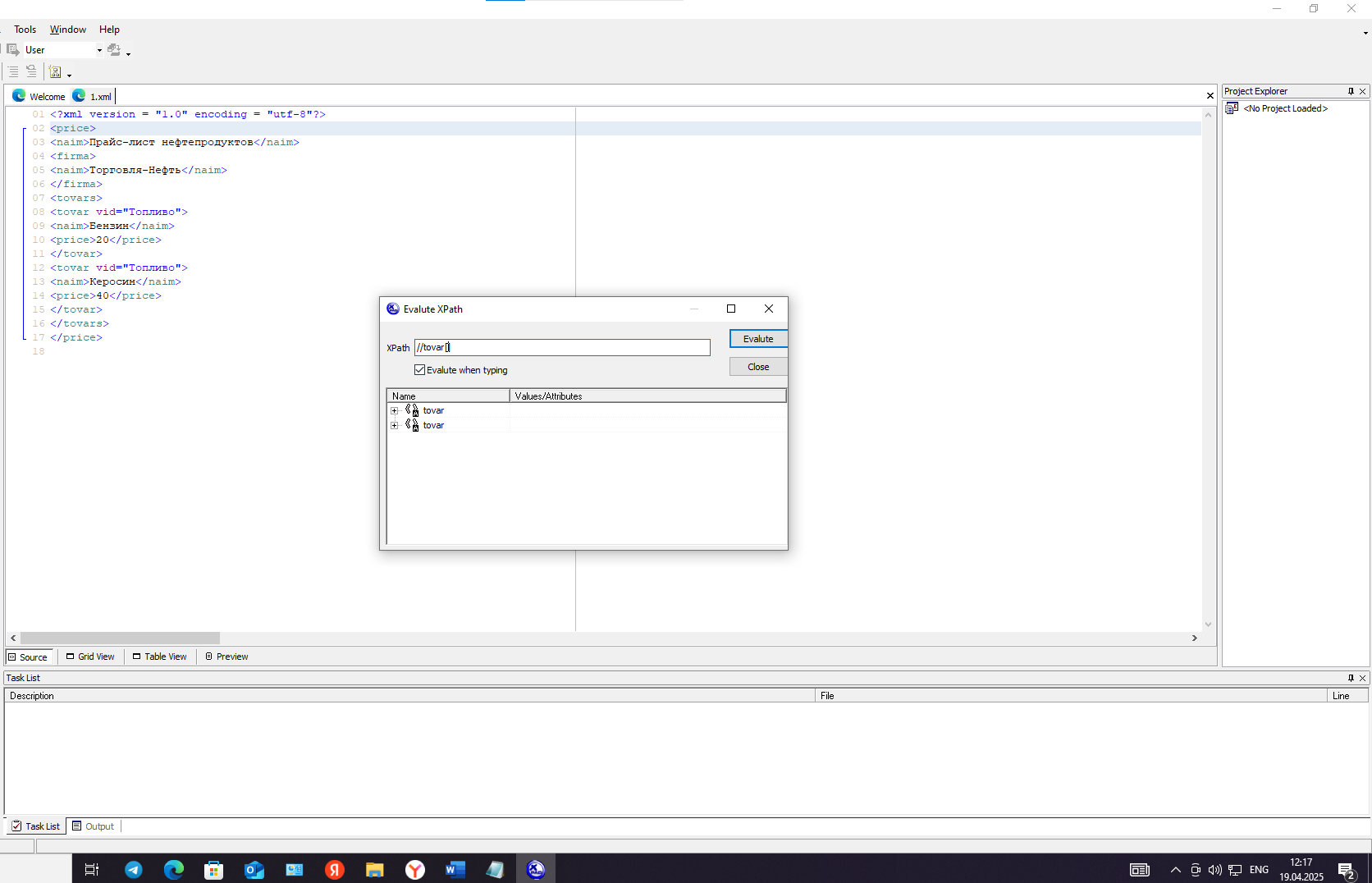


**Задание №3**

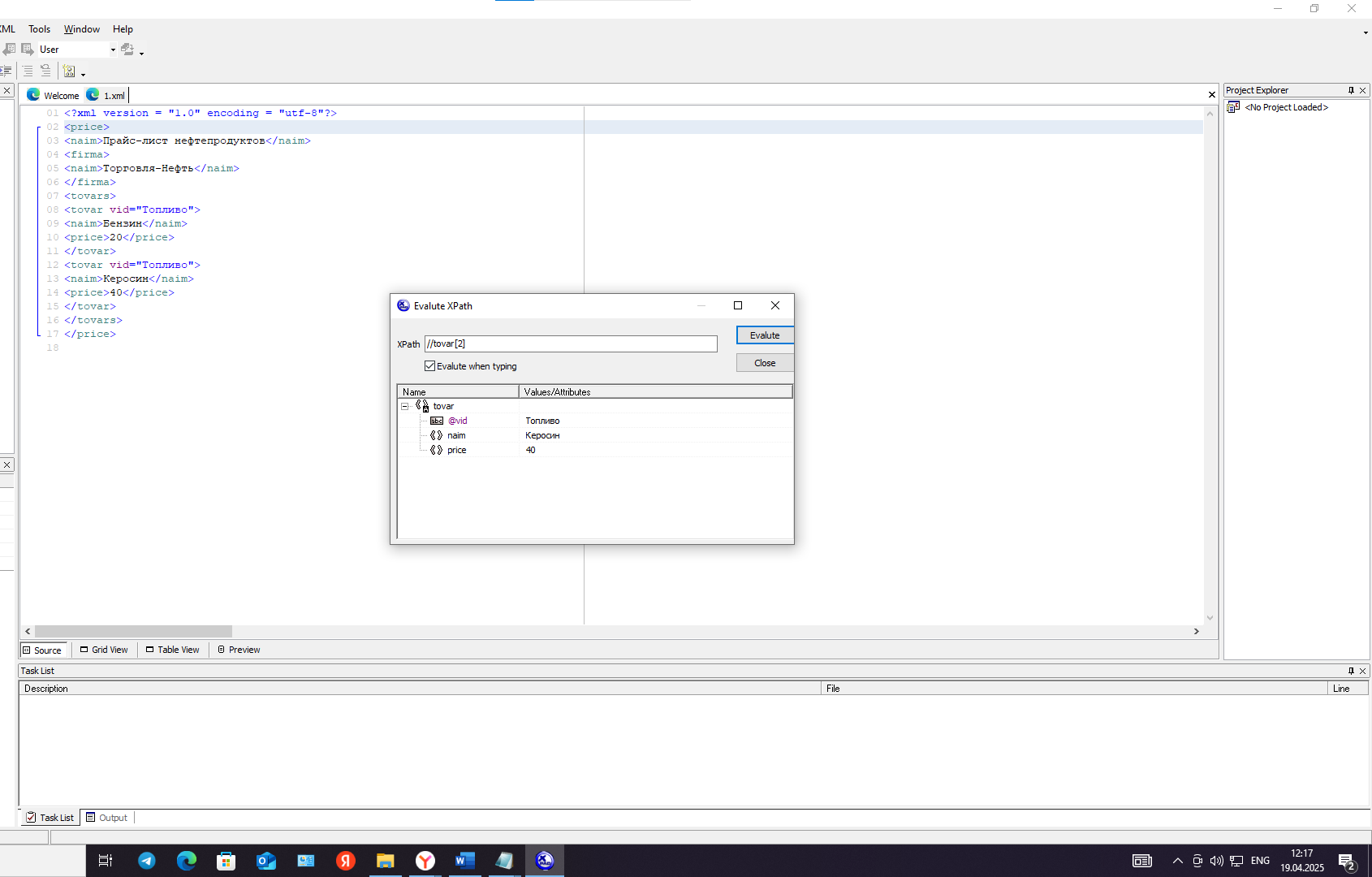
Найти все элементы с именем tovar и положением с номером 2.

Составление выражения – для нахождения элементов tovar используется следующее выражение: //tovar Для нахождения элемента с заданным положением используется номер в квадратных скобках. Следовательно, полное выражение - //tovar[2]

Все товары (рис. 56):



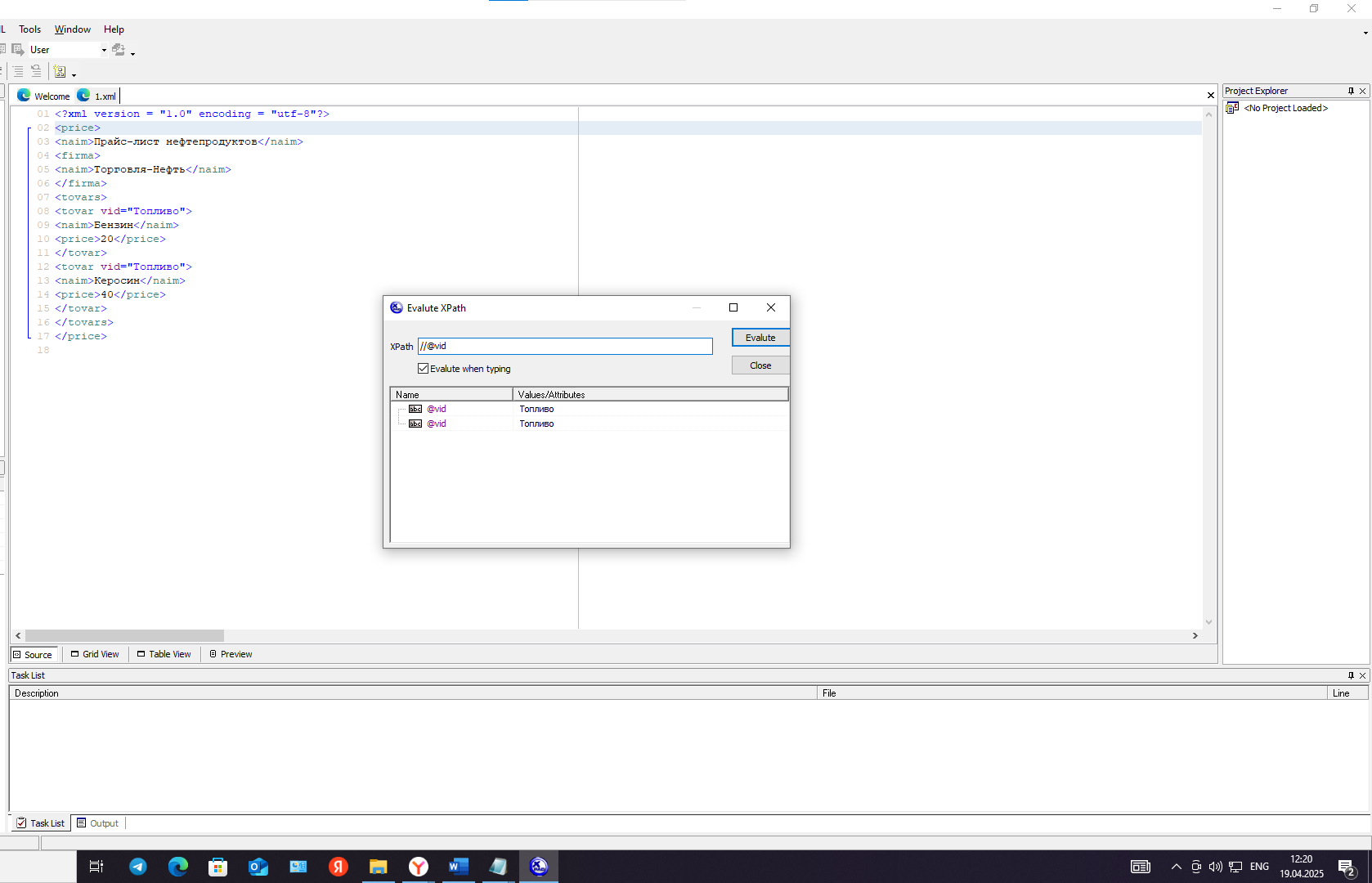
Именно второй товар (рис. 57):



**Задание №4**

Найти атрибуты vid для всех элементов. Для этого составим выражение //@vid.

Результат поиска приведен на рисунке 58.

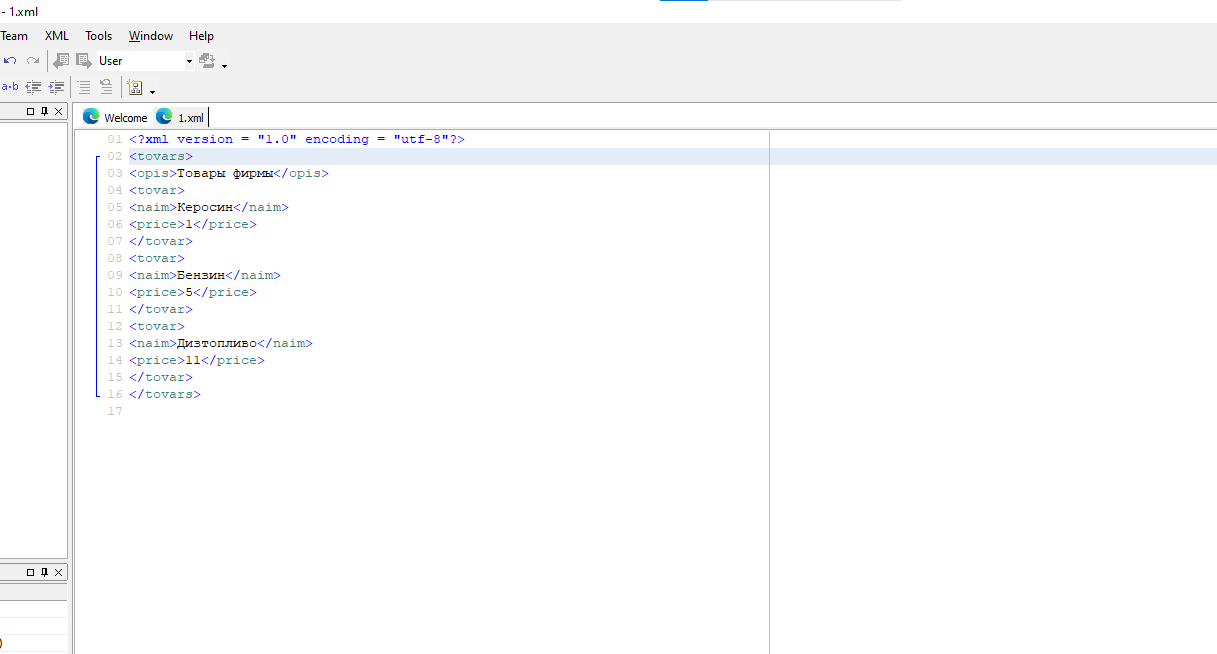


**Практическая работа №7**

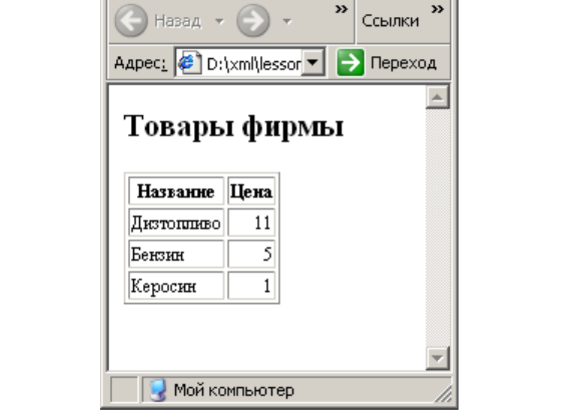
**Задание №1**

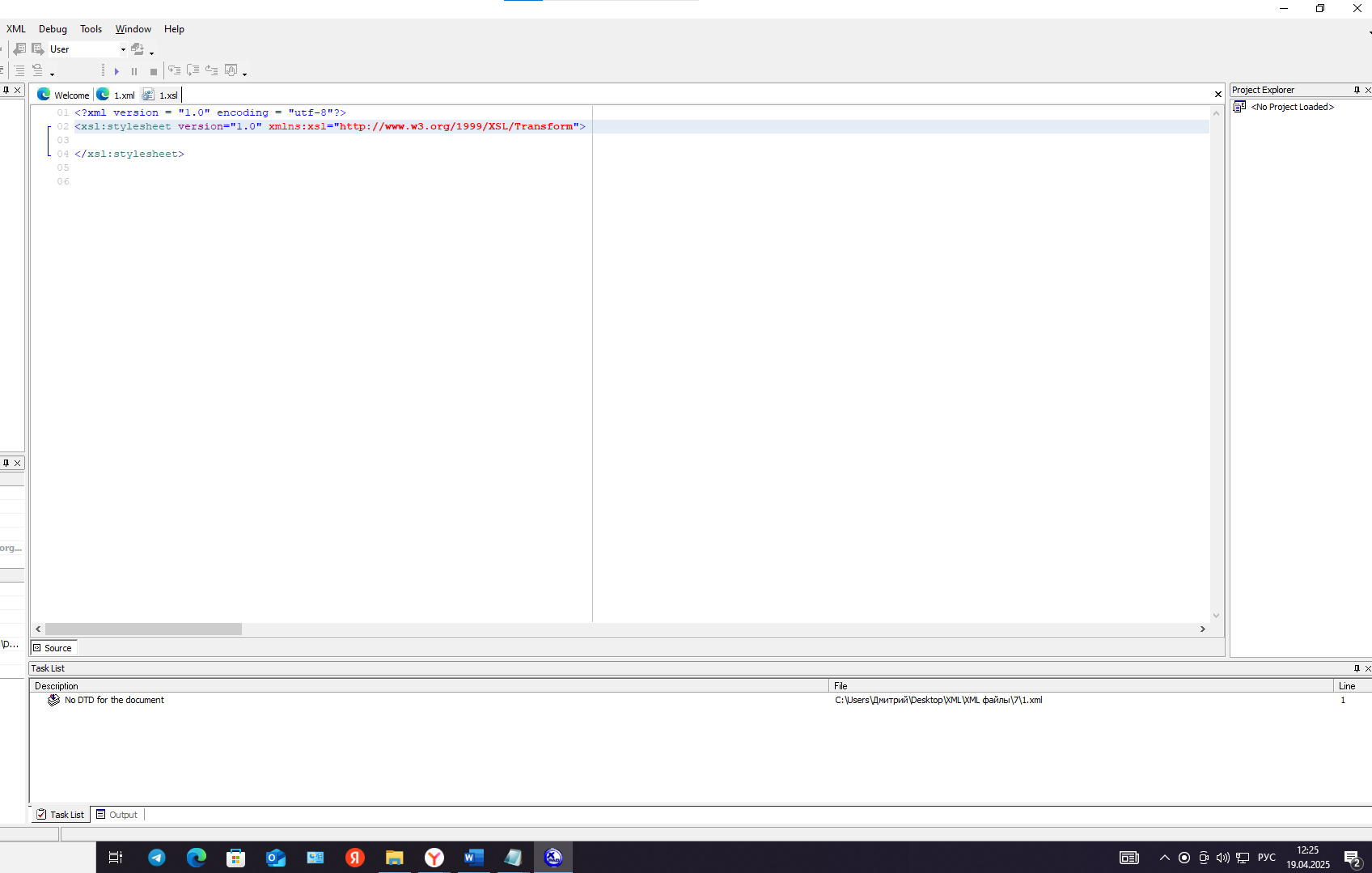
Преобразование XML-документа в HTML-документ с использованием XSL-файла.

Был создан документ следующего содержания (рис. 59):

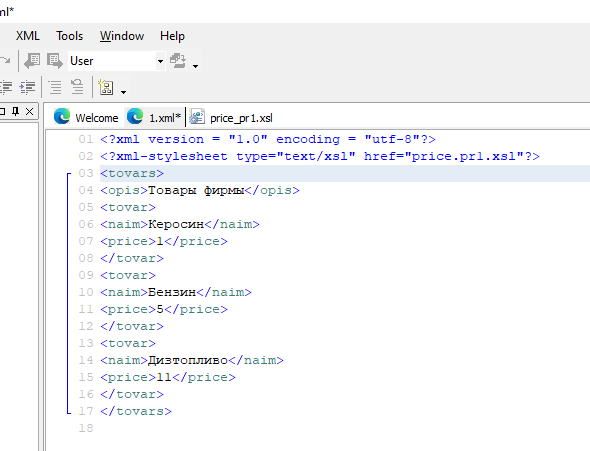


Для создания html документа следующего содержания (рис.60) необходимо продумать все html элементы и создать XSL файл (рис. 61).





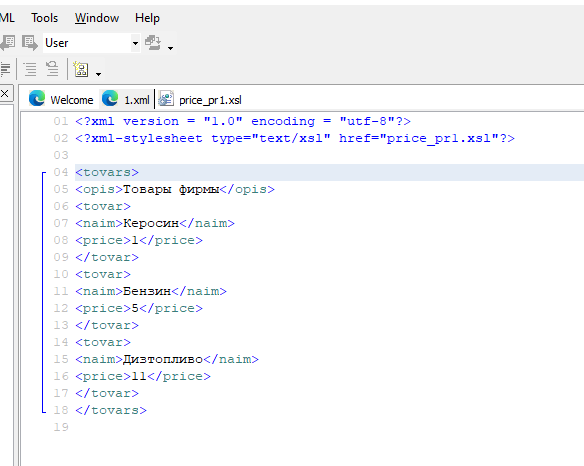
Затем подключим файл XSL в XML (рис. 62)



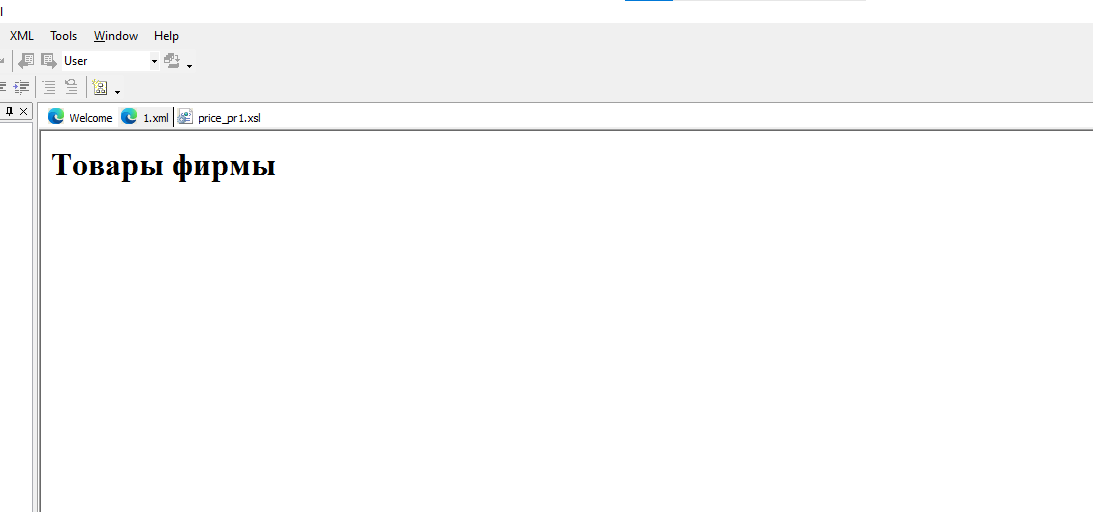
Затем потребуется добавить правило соответствия для корневого элемента XPath - <xsl:template match=”/”>. Элемент должен иметь начальный и конечный теги. Вид документа показан на рис. 63



На следующем шаге требуется добавить правило соответствия для элемента opis, который является дочерним элементом для элемента tovars. Также потребуется указать теги заголовка первого уровня <H1> и обозначить применение правил соответствия <xsl:apply-templates>. Вид полученного XSL-файла показан на рис. 64



Проверим работоспособность в файле Preview (рис.65).



Следующим этапом будет формирование таблицы. Для этого потребуется указать правило, которое будет отбирать элементы tovar и выводить информационное содержимое подчиненных элементов naim и price. Это правило будет выглядеть следующим образом (рис. 66):