Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Бакалавърска програма
„Софтуерно инженерство”

Предмет: XML технологии за семантичен Уеб

Зимен семестър, 2025/2026 год.

Тема №49: “Каталог на автомобили“

Курсов проект

Автори:

Никола Георгиев, 4MI0600288

Боян Видолов 1MI0600250

януари, 2026 г.

София

Съдържание

1 Въведение

Условие на заданието

Настоящият проект има за цел да създаде каталог на автомобили, базиран на XML технологии, който представя различни марки, модели, двигатели и конкретни автомобилни конфигурации. Изискването включва дефиниране на структурата чрез DTD (Document Type Definition), която да гарантира коректността и йерархичната организация на данните. Всеки автомобил се описва чрез ключови характеристики — марка, модел, двигател, цвят, категория, скоростна кутия, година на производство и цена, като връзките между обектите се реализират с помощта на ID и IDREF атрибути, осигуряващи вътрешна референциална цялост.

Каталогът трябва не само да съхранява структурирана текстова информация, но и да включва графично съдържание – изображения на автомобили, внедрени чрез XML ентитети или пътища към файлове в отделна директория. Целта е да се постигне интеграция между данни и визуално представяне, аналогично на това в реален уеб каталог като mobile.bg. Проектът следва да поддържа поне 7–8 различни автомобила с разнообразие от марки и модели.

След дефинирането на XML документа и неговия DTD, съдържанието се преобразува чрез XSLT (Extensible Stylesheet Language Transformations) в XSL-FO (Formatting Objects) структура, която се използва за генериране на PDF документ. По този начин се демонстрира практически процесът на пренос на информация от данни към представяне, както и възможностите за автоматизирано визуализиране на XML данни в печатен формат.

Проектът изисква логическо разделяне на информацията на четири основни секции – марки, модели, двигатели и автомобили. Тази структура е отразена в DTD схемата, която описва зависимостите между отделните елементи. Например, всеки модел е свързан с конкретна марка чрез brandRef, всеки двигател – с модел чрез modelRef, а всеки автомобил – едновременно с модел и двигател чрез modelRef и engineRef.

Реализацията предполага стриктно спазване на XML стандартите: валидност на документите, коректно използване на ID/IDREF атрибути, ясно дефиниране на типове съдържание и поддържане на четимост чрез йерархично подреждане на тагове.

Изпълнението на заданието завършва с генериране на PDF, който представя всеки автомобил на отделна страница с включена снимка, технически характеристики и описателен текст. По този начин се демонстрира пълният жизнен цикъл на XML-базирано съдържание — от структурирани данни до визуализиране в готов документ.

Актуалност на темата

Използването на XML като основен формат за обмен и съхранение на данни е изключително актуално в съвременните информационни системи и уеб приложения. XML предлага независимост от платформи и езици за програмиране, което го прави подходящ за интеграция между различни системи – например между автомобилни бази данни, каталози и уеб интерфейси. В контекста на дигиталната търговия с автомобили, подобна структура е изключително ценна за автоматизирана обработка и публикуване на съдържание.

Темата е актуална и поради това, че XML, XSLT и XSL-FO са основополагащи технологии за структурирано публикуване на данни, които продължават да намират приложение в области като електронно правителство, техническа документация, каталози и дигитални медии. Чрез тях се осигурява устойчивост на данните и възможност за многократна употреба на едно и също съдържание в различни формати (HTML, PDF, ePub и др.).

В последните години се наблюдава тенденция към автоматизация на визуализацията на данни, където XSLT служи като междинен слой между чистите XML данни и крайното представяне. В този смисъл проектът илюстрира не просто теоретичното използване на XML, а и практическото му прилагане за създаване на динамични, форматирани документи с реални изображения и параметри.

Реалният пример с автомобилен каталог е особено показателен, тъй като подобни системи се използват ежедневно от сайтове за продажба на автомобили. XML-базиран подход позволява лесно обновяване на данни, добавяне на нови модели и автоматично генериране на актуализирани документи без ръчно форматиране. Това прави процеса мащабируем и икономически ефективен.

От гледна точка на образователния и изследователския аспект, проектът демонстрира интеграцията на три основни технологии – XML (структура), DTD (валидация) и XSLT/XSL-FO (преобразуване и визуализация). Така се обединяват теоретичните принципи на структурираното описание на данни с практическото умение за създаване на функционални и визуално завършени документи.

Актуалността на темата се подчертава и от факта, че XML продължава да бъде де факто стандарт за обмен на структурирани данни в индустриални и корпоративни среди, независимо от конкуренцията на JSON. XML предлага по-строга типизация, коментари, декларации и възможност за валидация чрез DTD или XML Schema, което го прави изключително подходящ за официални и дългосрочни проекти като този автомобилен каталог.

Проблем и неговото решение

Основният проблем, който се адресира с този проект, е неструктурираното съхранение и представяне на автомобилна информация. В класическите уеб каталози или бази данни често се наблюдава фрагментация на данни, липса на ясна йерархия и трудности при интегриране на текстова и графична информация. Това води до проблеми с автоматизацията, повторното използване на данни и генерирането на стандартизирани документи.

Друг проблем е неясната връзка между обектите – марки, модели, двигатели и конкретни автомобили. Без механизъм за референции е трудно да се гарантира, че даден модел принадлежи на правилната марка, че двигателят е свързан с коректния модел, и че всеки автомобил е правилно асоцииран с двигател и модел. Това е критично за точността на каталога и за последващи автоматизирани обработки като филтриране, сортиране и извличане на статистики.

Допълнителен проблем е интеграцията на графично съдържание с текстовите данни. Снимките на автомобили трябва да бъдат свързани с конкретните записи, така че при преобразуване в PDF или HTML всяка визуализация да отговаря на правилния автомобил. Липсата на стандартизиран подход може да доведе до несъответствия и грешки при визуализиране.

Решението на тези проблеми се реализира чрез структуриране на данните в XML документ, дефиниран чрез DTD. Чрез отделяне на четири основни секции — brands, models, engines и cars — се постига ясна йерархия и логическа организация. Атрибутите ID и IDREF осигуряват надеждни вътрешни връзки между елементите, което елиминира риска от некоректни препратки и дублиране на данни.

Графичното съдържание се решава чрез включване на пътища към изображения като атрибути на елементите <car> (picture-link) или чрез XML ентитети. Това позволява автоматично визуализиране при преобразуване с XSLT/XSL-FO и създава унифициран интерфейс за представяне на информацията. При генериране на PDF всяка страница съдържа както текстови данни, така и съответното изображение, запазвайки контекстната връзка между елементи.

В крайна сметка, проблемът с неструктурирани, разпокъсани и трудно управляеми данни се решава чрез комбинация от DTD за валидация, XML за структуриране и XSLT/XSL-FO за визуализация. Това осигурява пълна интеграция между данни и представяне, автоматизирана обработка и възможност за разширение на каталога с нови автомобили, модели или марки, без да се нарушава съществуващата структура.

Употреба на технологията XML

В основата на реализирания проект стои Extensible Markup Language (XML) – отворен стандарт на W3C, предназначен за описание и обмен на структурирани данни. XML позволява ясно разделяне на информацията от начина на нейното представяне, като всеки елемент има строго дефинирано значение, а не визуална функция. В настоящия каталог XML се използва за логическа организация на автомобилните данни, като за всеки обект (марка, модел, двигател, автомобил) са определени съответни елементи и атрибути, описани в DTD схемата.

Основната идея при използването на XML е разделението между съдържание и форма. Документът cars.xml съдържа единствено данни, без информация за визуалното им оформление. Това позволява гъвкавост – едни и същи

XML данни могат да бъдат визуализирани по различен начин чрез XSLT шаблони: като уеб страница (HTML), печатен документ (PDF) или дори JSON изход за API. Тази архитектура е пример за концепцията data–presentation separation, широко използвана в съвременните информационни системи.

В проекта XML изпълнява и ролята на централен контейнер за взаимно свързани обекти чрез атрибутите ID и IDREF. Тази функционалност е аналогична на релационна връзка между таблици в база данни. Например, всеки <model> елемент съдържа атрибут brandRef, който сочи към конкретен <brand> чрез неговото id. По този начин се изгражда йерархия от взаимозависими данни без нужда от дублиране, което оптимизира структурата и подобрява четимостта.

Друга ключова характеристика на XML, използвана тук, е разширяемостта. DTD описва минимално необходимата структура, но към нея лесно могат да се добавят нови елементи като <fuelType>, <driveTrain> или <warranty>, без да се наруши валидността на съществуващите данни. Това прави XML идеален избор за системи, подлежащи на развитие – например бъдещо добавяне на електрически автомобили, хибридни варианти или допълнителни метаданни.

В проекта XML съдържанието се комбинира с външни ресурси като изображения, представени чрез атрибут picture-link. Така се демонстрира интеграцията на XML с файлови системи и мултимедийни източници. Възможно е използването и на XML ентитети за вграждане на графично съдържание директно в документа, което би позволило пълна самостоятелност на XML файла без външни зависимости.

XML е избран не само като формат, но и като основа за трансформации и автоматизация. С помощта на XSLT, данните от XML файла се преобразуват в XSL-FO структура, която впоследствие се рендерира в PDF формат чрез инструменти като Apache FOP. Този процес демонстрира цялостната екосистема на XML технологията – от структуриране и валидация до визуализация и публикация в преносим, печатаем документ.

Структура на документа

Структурата на XML документа е проектирана така, че да осигурява ясна логическа последователност и йерархични зависимости между различните нива на автомобилната информация. Главният елемент <catalog> съдържа четири основни секции: <brands>, <models>, <engines> и <cars>. Тази четирислойна архитектура позволява лесно разширяване и поддръжка, като всяка секция изпълнява конкретна роля и е валидирана спрямо DTD.

Елементът <brands> дефинира списък от автомобилни производители, всеки с уникален id. Тези идентификатори се използват като външни референции в други секции, което осигурява нормализирана структура без повторение на данни. По аналогия с релационните бази данни, <brands> може да се разглежда като „първична таблица“, към която се свързват зависими обекти.

Следващата секция <models> съдържа автомобилни модели, всеки от които препраща към съответната марка чрез brandRef. Тази релация тип „много към едно“ позволява една марка да има множество модели, без да се нарушава целостта на структурата. Всеки модел също има собствен id, който по-късно се използва от <engines> и <cars>.

В секцията <engines> са описани различните двигатели, като всеки елемент съдържа името на двигателя (<name>) и мощността (<power>). Атрибутът modelRef посочва към кой модел принадлежи съответният двигател. Тази връзка е критична за правилната корелация между модели и техните технически параметри. Подобен подход имитира обектно-ориентирана йерархия, при която двигателят е свойство на модела.

Основната и най-важна част от документа е <cars>, която съдържа конкретни инстанции на автомобили. Всеки <car> комбинира информация от трите предходни секции чрез modelRef и engineRef, като добавя специфични характеристики: цена (<price>), описание (<description>), цвят (<color>), трансмисия (<transmission>), категория (<category>) и година (<year>). Атрибутът picture-link осигурява достъп до визуален файл от директорията images/.

Тази структура е изцяло валидирана от DTD файла (cars.dtd), който задава формални правила за съдържание, подредба и типове елементи. Така се гарантира, че XML документът е не само синтактично правилен, но и семантично последователен. В допълнение, XSLT трансформацията (cars.xsl) използва XPath изрази, за да навигира между елементите и да извлича коректните стойности при изграждането на PDF каталога, което демонстрира прецизен контрол върху структурата и логиката на документа.

2 Анализ на решението

2.1 Работен процес

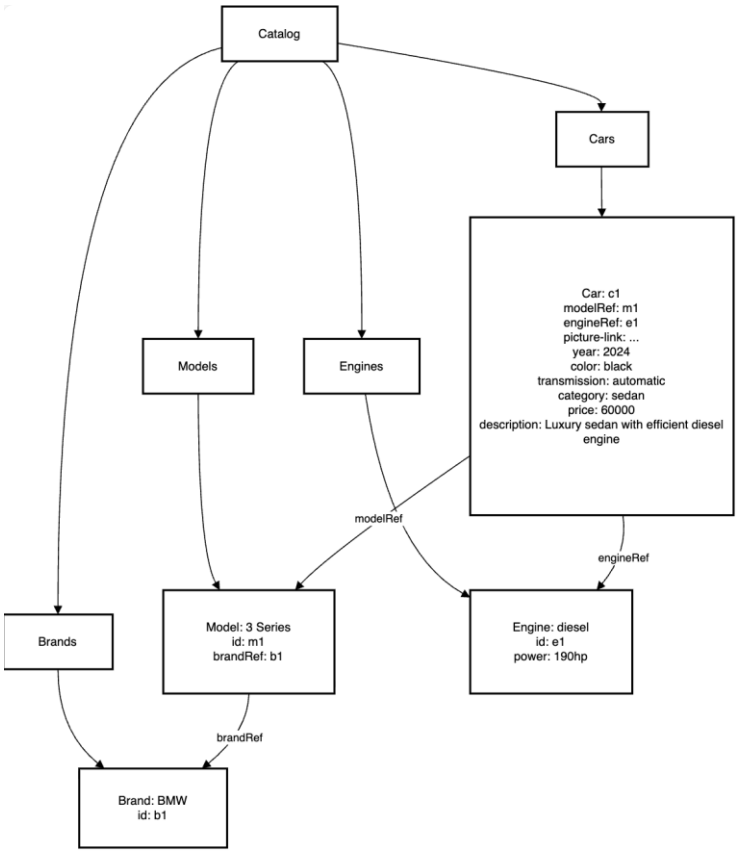
В рамките на проекта се разработва информационна система за представяне на автомобилен каталог, реализирана чрез XML технология и XSLT трансформация. Работният процес може да бъде разделен на три основни етапа – вход, обработка и изход.

Вход:Входното съдържание представлява структуриран XML документ (cars.xml), съдържащ данни за автомобилни марки, модели, двигатели и конкретни автомобили. Данните са събрани от публично достъпни източници (официални уебсайтове на производители и автомобилни медии). Всеки автомобилен обект включва характеристики като марка,

модел, година, тип двигател, мощност, скоростна кутия, цвят, категория и цена. В XML документа се включват и осем графични ресурса (снимки на автомобили), които са представени чрез URL връзки.
Обработка:Обработката на данните се извършва на няколко последователни етапа. Първо XML съдържанието се валидира спрямо DTD схема (cars.dtd), която гарантира правилна йерархична структура и коректни релации между обектите. След това чрез XSLT файл (cars.xsl) се извършва трансформация на XML документа в XSL-FO формат, който дефинира визуалното оформление и типографските правила за крайния документ. В процеса на трансформация се прилагат XPath изрази за достъп до конкретни елементи (напр. <brand>, <model>, <engine>, <car>), комбиниране на данни и подреждане в подходящ визуален ред.
Изход:Изходният резултат представлява PDF документ, генериран от XSL-FO чрез процесор като Apache FOP. Полученият PDF съдържа каталог на автомобили с илюстрации, технически спецификации и текстови описания. Всеки автомобил е представен в самостоятелен визуален блок, включващ снимка, име на модела, основни параметри и цена. Изходният документ е предназначен за печатна и дигитална употреба – например като брошура, онлайн каталог или вътрешен справочник на дилърска мрежа. Работният процес е проектиран така, че при всяка промяна на XML данните трансформацията може да се извършва автоматично, без нужда от ръчна намеса.

2.2 Структура на съдържанието

XML съдържанието е организирано в йерархична структура, която следва принципите на таксономично подреждане. Главният елемент е <catalog>,със следната структура:



Тази структура отразява взаимовръзките между категориите:
* <brand> (производител) е базов тип обект;
* <model> е подтип, свързан с конкретна марка;
* <engine> описва техническите параметри и се свързва с модел чрез engineRef;
* <car> е екземпляр, който обединява всички останали категории.
Типология и релации

* Марки (brand): 4 екземпляра – BMW, Audi, Skoda, Lexus.
* Модели (model): 8 екземпляра – по 2 за всяка марка.
* Двигатели (engine): 8 екземпляра с различен тип (бензинов, дизелов, хибриден, електрически).
* Автомобили (car): 8 крайни обекта, всеки съдържащ връзки към съответните модели и двигатели.
Характеристики и атрибути
Елемент Атрибути Тип на съдържанието Честота
<brand> id Текст (име на марка) 1..n
<model> id, brandRef Текст (име на модел) 1..n
<engine> id, type, power Текст/число 1..n
<car> id, modelRef, engineRef, picture-link Смесено (текст + мултимедия) 1..n
Йерархията и релациите между обектите позволяват да се представи онтологична мрежа от зависимости – например един двигател може да се използва в няколко модела, а една марка може да има множество модели. Структурата е проектирана така, че да може да се разширява с нови типове ресурси – напр. географски региони (страна на производство) или пазарни сегменти (SUV, хибрид, електромобил).

2.3 Тип и представяне на съдържанието

Съдържанието на проекта е смесен тип, включващ както текстови, така и графични (мултимедийни) ресурси.
Текстово съдържание
* Обем: 8 текстови обекта – описания на автомобили.
* Тип: Техническа и рекламна информация (описание, цена, параметри).
* Кодиране: UTF-8 (поддържа латиница и български език).
* Файлов формат: Вграден в cars.xml като PCDATA.
* Източник: Авторски текст, базиран на реални модели от сайтовете на BMW, Audi, Skoda и Lexus.
Графично съдържание
* Обем: 8 изображения, по едно за всеки автомобилен екземпляр.
* Тип: JPEG и WEBP изображения на автомобили (размери 500–900 px ширина).
* Източници:
  1. BMW 3 Series
  2. BMW X5 Protection
  3. Audi A4
  4. Audi Q5
  5. Skoda Kodiaq
  6. BMW Cover Sample
  7. Generic Sports Car
  8. TopGear Concept Car

Всички изображения са използвани единствено за учебни цели с илюстративен характер.
Файлова организация
* cars.xml – основен XML файл (около 15 KB, UTF-8).
* cars.dtd – DTD схема за структурна валидация (2 KB).
* cars.xsl – XSLT шаблон за визуална трансформация (5 KB).
* catalog.pdf – финален изходен документ, генериран от XSL-FO (≈1 MB).
Представяне
PDF изходът представя всеки автомобил в отделен визуален блок, включващ:
* заглавие с марка и модел (форматирано като Heading 2);
* снимка (в центъра на страницата);
* таблица с технически данни;
* кратко текстово описание и цена.
Форматът на представяне е подходящ както за печатна форма, така и за дигитално разпространение, включително уеб каталози и интерактивни PDF документи.

3 Дизайн (4-5 стр.)

3.1 Архитектура и използване на XML технологиите

Дизайнът на решението е базиран върху многостепенна XML архитектура, включваща три основни слоя: структурен (XML + DTD), логически (XSLT) и представителен (XSL-FO → PDF). Тези слоеве комуникират последователно, като осигуряват

пълен цикъл – от структуриране на данните, през трансформация и визуализация, до генериране на краен документ в професионален формат.
В центъра на архитектурата стои основният XML документ (cars.xml), който съдържа структурирани данни за автомобилни марки, модели, двигатели и автомобили. Всеки елемент има уникален идентификатор (ID), което позволява връзки между логически свързани обекти. Например, моделът се свързва с марка чрез brandRef, а автомобилът — с модел и двигател чрез modelRef и engineRef. Тази релационна структура позволява в XML документа да се моделират сложни зависимости, аналогични на релационна база данни, но при това в самостоятелна, самодефинираща се структура.
Логическата архитектура е проектирана така, че да поддържа многоизточниково съдържание. Това означава, че XML може да бъде обогатяван с нови секции или външни документи чрез включвания на външни entities. Например:

```
<!ENTITY engineData SYSTEM "engines.xml">
<!ENTITY modelData SYSTEM "models.xml">
```

Тези декларации позволяват модулност и разделяне на съдържанието на отделни файлове, като при валидиране и трансформация те се третират като част от основния документ. Така системата остава разширяема и лесна за поддръжка.
XML технологията е използвана не просто като носител на данни, а като описателен език за цялата логика на каталога. Всеки елемент и атрибут в XML има ясно семантично значение, което се използва от XSLT при генерирането на крайния документ. Например <engine type="diesel" power="190hp"/> не е просто текстов запис, а структурирано описание на техническа характеристика, което XSLT интерпретира и визуализира в таблица с етикет и стойност.
Архитектурно, решението следва модела „данни – логика – визуализация“ (Data → Transformation → Presentation). XML документът осигурява суровите данни, DTD — синтактична и структурна валидация, XSLT — логика за обработка и агрегиране на информацията, а XSL-FO — визуалното представяне, което се компилира до PDF чрез процесор като Apache FOP. Всяка промяна в XML автоматично се отразява в крайния документ без нужда от ръчно редактиране

```
Визуално може да се представи следната схема на архитектурата:
[cars.xml] → [DTD Validation] → [XSLT Transformation] → [XSL-FO Output] → [PDF Document]
    ↑           |                   |                   |
  Entities    ID/IDREF            XPath/XSLT          FO Formatting
```

Тази архитектура осигурява пълна проследимост на данните от източника до визуалния резултат, като всяка стъпка има конкретна роля в потока на информацията.

3.2 Структура на XML и DTD: Entities, релации и валидация

В основата на XML дизайна стои принципът за логическа модуларност и референтна цялост. Всеки основен тип обект – марка, модел, двигател, автомобил – е представен чрез собствен контейнерен елемент и дефиниран чрез DTD. Например, колекцията от марки е описана така:

```
<!ELEMENT brands (brand+)>
<!ELEMENT brand (name)>
<!ATTLIST brand id ID #REQUIRED>
```

Това определя, че всяка марка има единствено име и уникален идентификатор. Впоследствие моделите реферират тези идентификатори:

```
<!ELEMENT models (model+)>
<!ELEMENT model (name)>
<!ATTLIST model id ID #REQUIRED brandRef IDREF #REQUIRED>
```

Атрибутът brandRef гарантира, че всеки модел принадлежи на валидна марка. Валидацията, извършена от XML парсера спрямо DTD, ще отчете грешка, ако стойността на brandRef не съвпада с някой id в <brand>.
Същият принцип се прилага и при описанието на двигателите:

```
<!ELEMENT engine (name, type, horsepower)>
<!ATTLIST engine id ID #REQUIRED>
```

и при автомобилите:

```
<!ELEMENT car (year, color, transmission, price)>
<!ATTLIST car id ID #REQUIRED modelRef IDREF #REQUIRED engineRef IDREF #REQUIRED picture-link CDATA #IMPLIED>
```

По този начин се изгражда логическа мрежа от взаимосвързани елементи, която може да бъде навигирана чрез XPath в XSLT.
XML entities са използвани за централизирано управление на ресурси и често срещани стойности. Например, дефинирането на изображение като неразпарсван entity:

```
<!NOTATION jpeg SYSTEM "image/jpeg">
<!ENTITY bmw3 SYSTEM "images/bmw3.jpg" NDATA jpeg>
```

позволява да се извиква това изображение навсякъде в XML чрез &bmw3;. Това е полезно, когато едно и също изображение се използва в няколко различни елемента. Entities могат също да представляват текстови шаблони, например често използвани фрази или единици мярка, което редуцира дублирането на информация в документа.
Валидацията чрез DTD осигурява структурна консистентност преди всяка трансформация. Например, ако XML съдържа:

```
<model id="m5" brandRef="b99">
```

а не съществува

```
<brand id="b99">
```

процесорът ще прекъсне изпълнението и ще съобщи грешка:Error: IDREF attribute brandRef references an unknown ID 'b99'. Това гарантира, че логическите връзки между елементите винаги остават валидни и предотвратява грешки при визуализацията.
DTD файлът изпълнява и документаторска роля — в него са описани всички възможни елементи и техните връзки. Това позволява лесна поддръжка и добавяне на нови категории (например „страна на произход“ или „тип каросерия“) без нарушаване на съществуващата структура.
Така XML + DTD слоят осигурява стабилна и проверима база за следващия етап — трансформацията чрез XSLT.

3.3 Трансформация и XSLT съдържание

След като XML е валидиран, следващият ключов етап е XSLT трансформацията, реализирана чрез файл cars.xsl. Основната роля на XSLT е да комбинира свързана информация от различни секции на XML (чрез ID/IDREF връзки) и да я форматира в XSL-FO документ, който дефинира визуалното оформление на крайния PDF.
XSLT започва с шаблон за кореновия елемент:

```
        <xsl:template match="/catalog">
 <fo:root>
  <fo:layout-master-set>
   <fo:simple-page-master master-name="A4" page-height="29.7cm" page-width="21cm" margin="2cm">
    <fo:region-body/>
   </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="A4">
   <fo:flow flow-name="xsl-region-body">
    <xsl:apply-templates select="cars/car"/>
   </fo:flow>
  </fo:page-sequence>
 </fo:root>
</xsl:template>
```

Този шаблон дефинира основната страница и указва, че за всеки <car> трябва да се приложи нов шаблон.

Шаблонът за <car> е сърцевината на трансформацията. Той събира информация от свързани елементи чрез XPath:

```
<xsl:template match="car">
 <xsl:variable name="model" select="/catalog/models/model[@id=current()/@modelRef]"/>
 <xsl:variable name="brand" select="/catalog/brands/brand[@id=$model/@brandRef]"/>
 <xsl:variable name="engine" select="/catalog/engines/engine[@id=current()/@engineRef]"/>

 <fo:block space-after="1cm">
  <fo:external-graphic src="{@picture-link}" content-width="12cm"/>
  <fo:block font-size="16pt" font-weight="bold" color="#003366">
   <xsl:value-of select="$brand/name"/> — <xsl:value-of select="$model/name"/>
  </fo:block>
  <fo:block font-size="10pt" font-family="Helvetica">
   Engine: <xsl:value-of select="$engine/type"/> (<xsl:value-of select="$engine/horsepower"/>)
   Transmission: <xsl:value-of select="transmission"/>
   Price: <xsl:value-of select="concat(price, ' €')"/>
  </fo:block>
 </fo:block>
```

```
</xsl:template>
```

Този код комбинира три нива на информация: марка, модел и двигател. Използването на променливи и относителни XPath изрази гарантира, че при всяка итерация за <car> се извличат коректните стойности.
XSLT използва и условни конструкции за по-гъвкаво представяне:

```
<xsl:if test="contains($engine/type, 'electric')">
  <fo:block color="green">Eco Friendly Vehicle</fo:block>
</xsl:if>
```

Това добавя динамична логика, базирана на съдържанието на XML.
Резултатът от XSLT е XSL-FO документ — междинен формат, който съдържа всички инструкции за оформление: шрифтове, отстъпи, цветове и разположение на изображения. Този FO документ се обработва от FOP процесор, който генерира крайния PDF файл. В него всяка кола е представена на отделен визуален блок, със снимка, технически параметри и цена, напълно оформени чрез XSL-FO.
Цялостно, XSLT слоят изпълнява функцията на интелигентна интеграция: той свързва логическите единици в XML, прилага стилови правила и създава визуален документ с професионално оформление. Това демонстрира пълната сила на XML технологиите — от декларативно структуриране на информация до автоматизирана трансформация и типографски контрол върху резултата.

4 Използване на изкуствен интелект

1. Изкуствен интелект в имплементацията на решението
Изкуственият интелект беше използван основно за обучителни цели. Той разясняваше детайли относно използваните технологии, когато възникваше нужда от пояснения или се появяваха въпроси. Промптовете бяха от типа: „Обяснете X“, „Дайте пример за X“, „Защо X не работи?“, „Как да се направи X?“. Всеки отговор се разглеждаше като обучителна информация, а не като информация, която може директно да се приложи в имплементацията на решението, тъй като все още не може да се гарантира интегритетът на резултатите, които изкуственият интелект генерира. Извън обучителните въпроси изкуствен интелект не беше използван.
2. Изкуствен интелект в документацията
Изкуственият интелект беше използван, за да форматира документацията и да я изчисти от правописни и граматични грешки. Промптовете бяха от типа: „Коригирай грешките в документацията“. Изходът от тези промптове може да се наблюдава в текста.

5 Тестване

Първичната тестова колекция включва следните реални/контролни файлове: един главен XML (cars.xml) с 8 автомобила, DTD схема (cars.dtd), XSLT шаблон (cars.xsl) и папка images/ със 8 изображения. За допълнителни сценарии създадохме още 4 помощни XML файла — един с липсващи ID (негативен случай), един с дублирани ID (негативен), един с частични данни (липсващи picture-link или engineRef) и един голям генеративен файл с 1000 автомобила за performance тестове. Тези документи позволиха да покрием функционални, гранични и натоварващи тестове.

Структурната (синтактична) валидация беше първият етап — всеки XML файл се валидира спрямо cars.dtd. За целта използвахме xmllint и неговата опция за DTD валидация. Команда за единичен файл:

```
xmllint --noout --dtdvalid cars.dtd cars.xml
```

Това бързо откриваше несъответствия като неизвестни IDREF стойности, липсващи задължителни елементи или грешна подредба на елементите (например ако <car> няма <price>). Резултатът на валидацията беше записван в лог файл за последващ анализ.

След това изпълнихме серия от негативни тестове: документ с дублирани id стойности (проверка за уникалност), документ с modelRef сочещ към не-съществуващ id (проверка за референтна цялост) и документ с неправилен ред на елементи (напр. <year> преди <price> ако DTD определя обратното). Във всички тези случаи очаквахме xmllint да отчете грешка; успешен тест е такъв, при който инструментът открива и сигнализира грешката, преди да бъде позволена трансформация.

Функционалната проверка на трансформацията включваше изпълнение на XSLT върху валидни XML документи и инспекция на междинния XSL-FO (.fo) файл. Използвахме xsltproc (или Saxon за XSLT 2.0/3.0 сценарии) за да получим FO, след което инспектирахме синтаксиса и логиката в FO файла (наличие на fo:external-graphic, коректни fo:block заглавия и правилни XPath резултати). Командата за единична трансформация:

```
Xsltproc cars.xsl cars.xml > cars.fo
```

Проверихме, че за всеки <car> в .fo има съответен блок и че всички променливи, въведени в XSLT ($model, $engine) са коректно попълнени — това се прави с grep/awk или ръчен преглед за няколко пробни случая.

Рендер тестовете се фокусираха върху генериране на PDF и визуална QA: използвахме Apache FOP за генериране на PDF и след това правихме визуален преглед за коректност на оформлението, съотношение на изображение/текст, разделяне на страници и липса на препълване. Командата беше:

```
fop cars.fo catalog.pdf
```

При визуална проверка търсихме: липсващи изображения (съобщения в log), коректно форматиране на цените (формат-число), и дали всеки car започва на отделна страница (наличие на break-before="page"). Пробвахме и генериране на FO директно от FOP (XML + XSL) за да валидираме pipeline-а в едно изпълнение.

За гранични случаи направихме тестове на устойчивост: файл с 1000 автомобила и големи изображения, за да наблюдаваме паметови/времеви характеристики при трансформация и рендериране. Измервахме времето за XSLT (xsltproc / Saxon) и FOP, наблюдавахме пиково потребление на RAM и изписвахме логове за грешки при OOM. Тези резултати ни помогнаха да определим препоръчителни лимити (размер на изображение, препоръчителна компресия) за production среда.

Проверихме и специални случаи, свързани с мултимедия: документи с липсващи picture-link и такива с невалидни пътища. Тестовата логика включваше условни пътища в XSLT (xsl:if test="@picture-link") и fallback към placeholder изображение. Валидирането на наличност на файл се правеше чрез малък скрипт, който сверява picture-link срещу файловата система (или HTTP статуса при външни URL). Ако снимката липсва, процесът записва предупреждение и въвежда placeholder в FO.

Регресионните тестове бяха автоматизирани чрез скрипт, който изпълняваше валидация → трансформация → рендер и след това сравняваше MD5/SHA256 чексуми на генерираните PDF за идентичност при непроменени входни данни. По този начин откривахме странични промени в шаблона или инструмента. Команда-пример за pipeline скрипт:

```
xmllint --noout --dtdvalid cars.dtd $1 || exit 1
xsltproc cars.xsl $1 > tmp.fo
fop tmp.fo ${1%.xml}.pdf
```

Освен техническите тестове направихме и визуална/функционална валидация с чеклист: проверка за локализация/кодиране (UTF-8), проверка на шрифтове в FO/PDF, проверка за семантично представяне (заглавие, маркировка на година, цена) и accessibility-елементи (метаданни в PDF: заглавие, автор, език). Тези проверки гарантират, че крайният документ не само е валиден технически, но и отговаря на очакванията за потребителско представяне.

Накрая въведохме процес за логване и отчитане: всички тестови изпълнения записват stdout/stderr в лог директория, съдържаща валидационни съобщения, XSLT предупреждения и FOP рендер логове. Това позволява лесна диагностика при откриване на дефекти и служи за история при регресии. Критерий за приемане беше: всички валидни тестови XML файлове минават без грешки през валидация и трансформация и генерират PDF, докато всички негативни случаи анулират pipeline-а с ясно описана грешка.

6 Заключение и възможно бъдещо развитие

Проектът демонстрира успешното приложение на XML технологии за изграждане на структурирана, взаимосвързана и валидирана база данни за автомобилен каталог. Чрез използването на XML документи, DTD за валидация и XSLT/XSL-FO за визуализация бе постигната пълна интеграция между структурирани данни и тяхното представяне в PDF документи. Решението показва как декларативни формати като XML могат да заменят традиционни релационни бази за специфични сценарии на документация и каталогизация.

Основното предимство на използвания подход е гъвкавостта и разширяемостта на данните. Добавянето на нови марки, модели или двигатели не изисква промяна на съществуващия код за визуализация, а само актуализация на XML документа. Връзките между обектите се поддържат чрез ID/IDREF, което осигурява висока консистентност на данните и намалява риска от логически грешки при трансформацията.

DTD валидацията осигури строго структурно следване на правилата и предотврати грешки като липсващи или дублирани идентификатори. Тази стъпка е особено важна за автоматизирана трансформация с XSLT, където несъответствията биха довели до празни блокове или счупени връзки. Същевременно DTD предлага ограничена гъвкавост в сравнение с XML Schema (XSD), който би позволил дефиниране на типове данни, pattern matching и по-сложни ограничения.

XSLT слоят успешно агрегират и комбинира информацията от различни секции на XML документа, осигурявайки динамична и семантично правилна визуализация. XSL-FO форматът предоставя пълен контрол върху разположението,

шрифтовете, цветовете и изображенията в PDF, което е изключително важно за професионално оформление на каталога. Това демонстрира силата на XML трансформациите за създаване на готови за печат документи от структурирани данни.

Ограничеността на използваните технологии се проявява основно в обема и производителността. При големи файлове (стотици или хиляди записи) трансформацията с XSLT и рендерирането чрез FOP може да изисква значителни ресурси на паметта и CPU. За големи каталози или динамично съдържание може да се наложи използването на бази данни и сървърни технологии за кеширане и индексиране.

Алтернативи на XML + XSLT включват JSON базирани решения с фронтенд визуализация чрез JavaScript и библиотеки като React или Vue. Тези подходи са по-гъвкави за динамично съдържание, осигуряват бързи клиентски рендери и лесна интеграция с REST API. Основният им недостатък е липсата на стандартизиран, декларативен подход за генериране на печатни документи с пълен контрол върху оформление, който XSL-FO предлага.

Подобни решения в практиката намират приложение в автомобилната индустрия, електронни каталози, справочници и документи с висок визуален и семантичен компонент. XML и XSLT са използвани в корпоративни документи, електронни каталози и техническа документация, където последователността, валидността и визуализацията са критични.

За бъдещо развитие проектът може да бъде разширен с динамична интеграция на данни от външни източници — например API за актуални цени, наличности или нови модели. Това би изисквало добавяне на скриптове за автоматично обновяване на XML или миграция към XML база данни като eXist-db, която поддържа XPath/XQuery заявки.

Допълнителни подобрения включват разширяване на DTD или преминаване към XML Schema за дефиниране на по-сложни типове данни, валидация на формати за цена, дата и цветове, както и интеграция на повече мултимедийни ресурси като видеа или 3D модели. Това ще повиши информативността и реалистичността на каталога.

В заключение, проектът демонстрира, че XML технологиите предоставят надеждно, структурирано и разширяемо решение за изграждане на каталози, където данните и визуализацията са строго отделени, но логически свързани. Чрез комбинацията от DTD, XML, XSLT и XSL-FO е постигнат професионален резултат, който може лесно да се адаптира и разширява в бъдеще.

Като насоки за бъдещо развитие, може да се препоръча: интеграция на динамични данни чрез API, преминаване към XSD за по-сложна валидация, внедряване на автоматизирани тестове за големи обеми и добавяне на интерактивни елементи при електронни версии на каталога, без да се нарушава структурната цялост на XML документа. Това ще позволи решението да остане актуално, надеждно и практично за различни сценарии на употреба.

7 Разпределение на работата

Работата по проекта се проведе изцяло съвместно. Двамата участници бяха прочели заданието предварително, и съвместно имплементираха решението му в стил Pair-Programming.

- Първо се изгради прототип за XML, XSLT, DTD документите.
- В последствие се разписа дизайнът и изискванията съответно за документите.
- Имплементира се XML документа.
- Имплементира се XSLT документа.
- Имплементира се DTD документа.
- Създаде се github репозитория.
- Написа се документацията.

8 Използвани литературни източници и Уеб сайтове

Информация за коли:

- https://www.autoevolution.com/cars/

Използвани снимки:

- https://cache2.arabwheels.ae/system/car_generation_pictures/29388/original/Cover.jpg?1733354366
- https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSH2c6WS-aaa2Oem9-GhJ-IZhQFYlZNVrhaYg&s
- https://www.bmw.co.za/content/dam/bmw/common/all-models/3-series/sedan/2018/inform/bmw-3series-3er-inform-lines-01-01.jpg
- https://mediapool.bmwgroup.com/cache/P9/202308/P90520309/P90520309-the-new-bmw-x5-protection-vr6-08-23-599px.jpg
- https://stimg.cardekho.com/images/carexteriorimages/930x620/Audi/A4/10548/1757137106350/front-left-side-47.jpg?impolicy=resize&- imwidth=420

- https://article.images.consumerreports.org/image/upload/t_article_tout/v1753967736/prod/content/dam/CRO-Images-2025/Cars/CR-Cars-InlineHero-2025-Audi-Q5-f-driving-7-25
- https://www.topgear.com/sites/default/files/2024/10/DSC02736.jpg
- https://images.autox.com/uploads/2025/09/Skoda-Kodiaq-Race-Blue-1758285005922.webp