**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

Факултет „Компютърни системи и технологии”

Катедра „Информационни технологии в индустрията”

Специалност: Информационни технологии в индустрията

**ДИПЛОМНА РАБОТА**

*Тема: Разработка на уеб услуга за музика*

Студент: Александър Ивайлов Иванов

Фак. No: 501217031

Научен ръководител: гл. ас. д-р Галя Павлова

София, 2021

**СЪДЪРЖАНИЕ**

[**1. Въведение в уеб системи** 4](#_Toc71634228)

[**1.1 История** 4](#_Toc71634229)

[**1.2 Предимства** 5](#_Toc71634230)

[**1.3 Езици за реализиране на уеб приложения** 7](#_Toc71634231)

[**2. Проектиране на уеб услугата** 18](#_Toc71634232)

**УВОД**

Предмет на тази дипломна работа е проектирането и разработването на back-end частта на уеб-базирано приложение/услуга за слушане на музика. То може да бъде използвано в Интернет при наличието на уеб браузър или като самостоятелно приложение от локален компютър. За реализацията ще използваме допълнителен софтуерен продукт за симулирането на такъв браузър на локално ниво, наречен Postman. Услугата може да се разглежда като аналог на популярното приложение Soundcloud, което е безплатно за ползване, или по-популярното, но платено такова, Spotify.

То ни предоставя възможност за слушане на милиони песни без да е нужно да ги съхраняваме наше устройство, стига да имаме постоянна връзка с Интернет. Услугата осигурява лесен начин за търсене, слушане, харесване и сортиране на песни и плейлисти, следване на други потребители, както и качване на наши песни и създаване на собствени плейлисти. Допълнителна функционалност е писането и харесването/нехаресването на коментари под всяка една песен.

За реализацията на уеб услугата са използвани технологии, съвместими с Java, а именно Spring Framework, Hibernate, JPA, Tomcat Server, MySQL, както и HTTP протокола за безопасна комуникация в Интернет. Използваната среда за разработка е IntelliJ IDEA върху Windows OS.

# **1. Въведение в уеб системи**

## **1.1 История**

Всичко започва с появата на уеб услугата в интернет (1990г.). В самото начало на виртуалната мрежа съдържанието и уеб страниците са били статични - представлявали са само HTML файлове. По-скоро брошури, на които един потребител поставя ръчно някакъв текст. В следващите години на уеб (~1993г.) към брошурата вече са се добавяли и картинки. Към този момент съдържанието, заедно с презентацията му са били едно цяло – HTML. Едва през 1995г. се появява специализираният за презентацията език CSS (Cascading Style Sheets), чрез който се описва форматирането и стила на HTML елементите. По същото време (1995-1997г.) се появява и възможността (CGI – Common Gateway Interface) за динамично генериране на съдържанието в уеб страниците чрез програмни и скриптови езици като Perl, Python, PHP, ASP, които работят на сървъра.

През последвалите години се появят още основни уеб технологии, променящи статичната среда на уеб такава. Една от тях е DOM (Document Object Model) или Документен Обектен Модел (1997-1998г.), с който съдържанието на HTML документа се поставя в дървовидна логическа структура от отделни обекти. А тези обекти, или части от съдържанието, вече може динамично да се променят чрез JavaScript(1995г.) и Ajax (1999г.) заявки, които не изискват презареждане на уеб страницата.

Дотук развитието на уеб технологиите преобръща статичната брошура в динамична и интерактивна уеб страница. Съдържанието и визуалното му представяне се генерират динамично, спрямо потребителя и неговите предпочитания и взаимодействие със страницата.

Интерактивността на уеб сайтовете подхранва взаимодействието на потребителите с тях. Но освен страхотна презентация, за да се задържи интересът и подтикне взаимодействието, сайтът вече трябва да предоставя уместно, качествено и най-вече полезно съдържание. А това от своя страна налага нуждата от добавяне и редовно обновяване на такова съдържание. В ранните години на уеб това би било непосилна задача за един единствен човек, който трябва да обновява ръчно HTML кода на десетките страници в сайта, ежедневно. В създаването на съдържание за сайта вече се включват повече хора, отговарящи за съответната информация – авторите да добавят публикации и помощни статии, маркетинг специалистите да обновяват промоциите, човешки ресурси да добавят обява за работа и други.

И тук е моментът, в който една система за управление на уеб съдържанието става жизнено необходима. Една система, която да предоставя методи за лесно управление на съдържанието в сайта от много потребители. Още през 1995-1997г. се появяват първите системи за управление на съдържанието, които са предимно за корпоративни и бизнес клиенти и са със затворен код, специална разработка.

По това време (1997-1998г.) в софтуерната индустрия вече се говори за Open Source (отворен код), а Netscape публикуват изходния код на своя софтуерен пакет Netscape Communicator. Малко след това е сформирана и инициативата Open Source Initiative, която насърчава създаването на софтуер с отворен код. Отворен код означава, че притежателят на авторските права на изходния код предоставя правата за обучение, промяна и разпространение на софтуера на всекиго и за всякакви цели.

И в началото на 2000г. започват да се появяват и първите системи за управление на уеб съдържанието (WCMS), които са с отворен код, като Drupal 2000г., WordPress 2003г., Joomla 2005г. Най известните системи с отворен код, специализирани за онлайн търговия, се появяват малко по-късно Magento 2008г., OpenCart 2010г., WooCommerce 2011г.

## **1.2 Предимства**

Уеб системите имат голям набор от предимства пред десктоп базираните системи, за които е нужно да бъдат инсталирани или разработени за конкретна операционна система и трябва да имат стриктни хардуерни изисквания, за да се подсигури тяхното правилно функциониране. Тази хардуерна зависимост както и mainframe terminal applications представлява неудобство за някои потребители, а и допълнителни разходи за набавянето на нужните изисквания за изпълнението им.

Уеб базираните системи предлагат единна база данни, което води до централизирана архитектура на уеб системите и позволява съхранението на всички данни на едно място. Това предотвратява например проблемите с консолидирането на данните от различните офиси на компанията или различни инсталации на съответната система (ако например се инсталира отделна база данни във всеки обект, в който се използва софтуерът).

Благодарение на единната база данни е възможно да получавате информация в реално време за състоянието на компанията. Единната база данни позволява и много хора да работят с един и същи запис – например актуална ценова информация, задължения на клиенти и др. Eдин потребител може да вкарва поръчките от клиенти, а друг да обработва стоката от склада, намиращ се в друг град, например.

Улесняват се процесите на инсталация, настройка и поддръжка. В общия случай инсталацията, настройката и поддръжката на една ERP системасе разделя на клиентска и сървърна. Когато вашата ERP\* система е уеб базирана вие не се нуждаете от инсталация на клиентската машина (работните места). При създаване на ново работно място единственото нещо, което е нужно е уеб браузър. Поддръжката и развитието на една ERP система са неотменна част от нейния жизнен цикъл. Поради това, че използвате уеб базирана ERP система, всички промени и модификации по системата се извършват само на едно място, без да е необходимо да се отразяват по всички работни места. Системата е една и се ползва от всички потребители от едно централно място. Това прави поддръжката й ефективна, бърза и евтина.

Имаме лесен достъп до тях. Благодарение на интернет, когато ERP системата ви е в уеб, не сте ограничени да работите с нея само в офиса от служебния си компютър – имате свободата да разполагате с цялата актуална информация, където и да се намирате. Възможно е да работите с вашата уеб ERP система буквално от всяко място чрез интернет – от летището, от хотела или от дома, можете да гледате справки, да модифицирате данните, да отчитате работата си или просто да наблюдавате процесите в компанията. Всички това се случва без нуждата от скъпа комуникационна инфраструктура, каквато изискват десктоп базираните ERP системи. Разбира се, винаги можете да ограничите достъпа до вашата ERP система само за потребители от вашата локална мрежа или чрез VPN. Също така вече не е задължително да използвате само персонален компютър за работа с уеб приложения. Поради това, че повечето последни модели мобилни телефонни разполагат с уеб браузър вашата ERP система лесно може да се направи достъпна за мобилни устройства.

Вместо ръчно да експортвате справки от вашата ERP система, всички ваши контрагенти могат да получават информация в реално време. Получавате запитвания от вашите клиенти директно от уеб сайта, предоставяте справка за оферти, падежи, плащания, поръчки и др. на вашите клиенти, както и справки за вашите доставчици и други контрагенти. Можете да публикувате на вашия уеб сайт различни резултати на компанията, като черпите информация направо от базата данни на ERP системата. Благодарение на новите технологии в уеб (Ajax, SOAP, XML и др.) можете да интегрирате вашата ERP система с външни приложения – публикуване на информация, връзка с външни системи за разплащане и т.н.

При уеб базираните системи липсват ограниченията по отношение на хардуерните характеристики или операционната система на компютъра, от който се ползват. Единственото изискване е активна интернет връзка и инсталиран браузър. Няма хардуерни изисквания. Не е нужно да заделяте голяма част от паметта ви за инсталирането и поддръжката на системата.

За разлика от десктоп приложенията, при който информацията се записва на компютър, USB носители или други външни устройства, уеб базираният софтуер се поддържа на професионално защитени и надеждни сървъри, които се намират в специални информационни центрове. Там се качват и съхраняват базите данни с информация. Няма нужда постоянно да се правят резервни копия на данни от програмата, тъй като архивирането на информацията е автоматично. Обикновено тя се архивира едновременно на няколко независими добре защитени устройства с памет. Така няма опасност от безвъзвратно изгубване на информация, заради кражба или повреда на компютъра, срив в софтуера и т.н.

Предвид факта, че уеб базираният софтуер е нает и се използва през интернет, не се налага да се правят първоначални разходи за закупуването на лицензи, мощни компютри, сървъри, както и за инсталацията му. Няма да се налага да се плаща и за актуализациите на счетоводната програма, техническа поддръжка и т.н. Месечната такса е единственият финансов ангажимент към доставчика на софтуер.

Лесно може да се сложи рестрикция за отделните потребители, тези който имат определен достъп до данни и операции и тези с ограничен достъп. Администраторите могат лесно да проследят извършените от други потребители дейности.

## **1.3 Езици за реализиране на уеб приложения**

Скриптовите езици бързо се превръщат в програмни езици с общо предназначение, поради тяхната функционалност. Те се използват масово там, където времето на разработчиците е по-ценно от времето за изпълнение на програмата. С тях се пишат програми, предназначени за:

Вграждане в Web-страници. Тези програми не се компилират, а се интерпретират от браузъра при зареждането на съответната страница на клиентския компютър.

С помощта на стандартните средства на скрипт - езика и на обектите, дефинирани от DOM, програмистът може да получи достъп до всеки елемент от Web-страницата и динамично да промени неговото съдържание и стилово оформяне.

Реализиране на разнообразни динамични ефекти като показване и скриване на съдържание, придвижване и позициониране на обекти, анимация и т.н. Съединяване в едно цяло различни пакети приложни програми. Създаване на големите Интернет приложения. Автоматизиране.

За да се разбере по-добре същността на скриптовите езици е необходимо да се изходи от второто им наименование – езици за сценарии.

Сценарий – последователност от операции, които потребителят може да извършва на компютъра. Сценариите обикновено се интерпретират, а не компилират.

Приложна програма сценарий (скрипт) — програма, която автоматизира някаква задача, която без сценария потребителя трябва да извърши ръчно. Тъй като сценариите се интерпретират от изходния код динамично при всяко изпълнение, те се изпълняват значително по-бавно от готови програми, транслирани в машинен код на етап на компилиране.

Динамично изпълними скриптовите езици четат инструкциите от файла на програмата на минимални функционални блокове и изпълняват тези блокове без да четат останалия код.

Предварително компилирани скриптовите езици – прочитат програмата в началото, компилират я цялата в машинен код или някакъв вътрешен формат и след това я изпълняват.

Какви са предимствата на скриптовите езици? Скриптовите езици притежават по-сложен инструментариум и поддържат по-прогресивни техники на програмиране. Това съкращава времето за разработка на програми и ги прави по ефективни. Скриптовите езици позволяват бързо да се откриват грешки в програмата и да се реализират доработки на кода без да се изчаква приключване на компилацията.

### **1.3.1 HTML**

HTML (HyperText Markup Language) или език за маркиране на хипертекст е основният език за описание на уеб страници. HTML е стандарт в Интернет, а правилата се определят от международния консорциум W3C. Текущата версия на стандарта е HTML5.

Описанието на документа става чрез специални елементи или маркери, които се състоят от етикети или тагове (HTML tags). HTML елементите са основната градивна единица на уеб страниците. Чрез тях се оформят отделните части от текста на една уеб страница, като заглавия, цитати, раздели, хипертекстови препратки и т.н. Най-често HTML елементите са групирани по двойки и имат отварящ таг <h1> и затварящ таг </h1>.

HTML е основната технология, контролираща това, което уеб браузърът показва на екрана. Основните категории команди на езика HTML, включени във версиите 3.2 и 4.01 са предназначени за: специфициране на стилови формати и управление на текстовия поток (FlowControl); включване на графични изображения (Images); създаване на хииервръзки (Links); интегриране на аудио с външни графични обекти (Sound and Maps); вмъкване на видео таг (Video) и на youtube клипове директно от тяхната страница чрез тага (Iframe); създаване на интерактивни формуляри (Forms); разделяне на документа на отделни полета (Frames); включване на външни приложения (Applet), написани на езика Java; осъществяване на връзка с външни информационни структури (CGI-script). HTML е език за маркиране, който уеб браузърите използват, за да интерпретират и създават текст, изображения и други материали. Неизпълнената характеристика за всяка точка се определя в браузъра, като могат да бъдат променяни или засилвани с допълнителни дизайнерски страници, наречени Cascading Style Sheets (CSS).

В повечето случаи HTML кодът е написан в текстови файлове и се хоства на сървъри, свързани към Интернет.

Тези файлове съдържат текстово съдържание с маркери – инструкции за браузъра за това как да се показва текстът. Предназначението на уеб браузърите е да могат да прочетат HTML документите и да ги превърнат в уеб страници. Браузърите не показват HTML таговете, а ги използват, за да интерпретират съдържанието на страницата.

Основното предимство на HTML е, че документите, оформени по този начин, могат да се разглеждат на различни устройства, а не само на екрана. Документът може да бъде правилно оформен и върху монитора на персонален компютър.

Създаването на HTML-базирана уеб страница може да се извърши с помощта на обикновен текстов редактор. Този начин изисква познаване на HTML тагове, така че те да бъдат интегрирани в текста, който ще се показва на страницата. Също така, често срещани са по-приятелски настроените инструменти, които не изискват от потребителя да притежава познания по HTML. Специализирани инструменти за създаване на HTML страници са някои от продуктите на Microsoft, FrontPage, Notepad, Notepad++, Sublime Text, NetBeans и други.

Пример 1. Обикновен HTML код

<!DOCTYPE html>

<html>

<head>

<title>Заглавие, което се визуализира в таба

на браузъра</title>

</head>

<body>

<h1>Заглавие</h1>

<p>Параграф, в който може да се публива

статия</p>

</body>

<html>

**Семантични елементи в HTML5:**

* Елемент <header> - представлява заглавие на елемент или група от елементи. Тага <header> позволява използването на няколко заглавия в един документ.
* Елемент <nav> - дефинира навигацията на сайта. Той е предназначен за големи блокове от навигационни връзки. Не всички връзки трябва да са в <nav> елемента, а само главното навигационно меню.
* Елемент <article> - представлява предмет/статия. Съдържанието в <article> трябва да има смисъл само по себе си и да може да се разпространява независимо и отделно от останалата част на сайта. Използва се най-често за публикация във форум; публикация в блог; новини; коментари и др.
* Елемент <section> е тематично групирано съдържание, обикновено със заглавие. Този елемент позволява влагане на нова сегмантична структура във вече съществуващ елемент.
* Елемент <aside> - представя съдържание което е странично от основното съдържание на страницата. Съдържанието в <aside> се счита за неважно и може да бъде пропуснато, ако сайта се възпроизвежда на устройство с малък екран като телефон.
* Елемент <footer> - може да съдържа информация за автора на страницата; навигация на сайта; авторските права върху използваните материали и връзки към други (подобни) публикации.

### **1.3.2 CSS**

CSS (Cascading Style Sheets) е език за описание на стилове, който се използва основно за описване на представянето на документ, написан на език за маркиране (HTML, XML). Официално спецификацията на CSS се поддържа от W3C (World Wide Web Consortium). CSS позволява да се определя как да изглеждат елементите на една HTML страница –

шрифтове, размери, цветове, фонове и др. CSS кодът се състои от последователност от стилови правила, всяко от които представлява селектор, последван от свойства и стойности. CSS е удобен начин за форматиране на текст, шрифтове, списъци, таблици и др.

Всяко CSS правило има две основни части: селектор и една или повече декларации. Ако трябва да сме още по-прецизни, всяка декларация се състои от комбинацията свойство и стойност. Синтаксисът е следния:



*Фиг.1.1 CSS правило*

От лявата страна на свойството се намира селекторът – това е частта от правилото, указваща за кой елемент от документа трябва да се приложат стиловете.

В примера са избрани всичките елементи на body. От дясната страна на селектора се намира декларацията, която представлява комбинация от свойството и стойност. Свойството е стилов атрибут, на който се задава стойност. Всяко свойство има стойност. CSS декларацията винаги завършва с точка и запетая, а групите от декларации са заградени от фигурни скоби.

Най-голямото улеснение, което предлага CSS е свързано с контрола на голям набор HTML документи, като контролирането на външния вид на страниците може да става по няколко начина.

Пример 1. CSS код с добавени стилове в </head> частта

<!DOCTYPE html>

<html>

<head>

<title>CSS като отделен файл </title>

<meta

http-equiv=

Content-Type

content="text/html; charset=UTF-8" />

<link type="text/css"/>

</head>

<body>

rel="stylesheet"

href="external.css"

<h1>Заглавие</h1>

<p >Параграф, в който може да се публива

статия</p>

</body>

<html>

Пример 2. CSS код с добавени стилове в </head> частта.

<!DOCTYPE html>

<html>

<head>

<title> вграден CSS файл </title>

<meta http-equiv= Content-Type content="text/html;

charset=UTF-8" />

<style type=”text/css”>

</style>

</head>

<body>

<h1>Заглавие</h1>

<p >Параграф, в който може да се публива статия</p>

</body>

<html>

Пример 3. CSS код със стилове в самите HTML елемент

<!DOCTYPE html>

<html>

<head>

<title> вграден CSS файл </title>

<meta

http-equiv=

Content-Type

content="text/html; charset=UTF-8" />

<style type=”text/css”>

</style>

</head>

<body>

<h1>Заглавие</h1>

<p style=”text-align: left;”>Параграф,

в който може да се публива статия</p>

</body>

<html>

При наличие на повече от един стил, специфициран за HTML елемент, ще се използва този, който има най-висок приоритет (четири е най-високият), както следва:

* + - 1. Стилът на браузъра по подразбиране.
      2. Външните стилове.
      3. Вътрешните стилове.
      4. Стиловете, вградените в самите HTML елементи.

Първоначално HTML етикетите са били създадени, за да определят съдържанието на документа. Или иначе казано, идеята била следната: таговете трябвало да “казват”: “Това е заглавие”, “Това е параграф”, “Това е таблица”, използвайки синтаксис от рода <h1> (за големи заглавия), <p> (за параграф). Постепенно с добавянето на нови спецификации към HTML и с увеличаване на изискванията към визията на сайта, ставало все по-трудно да се създават страници само с него.

Друго предимство на CSS е, че спестява работа и по-точно писане на код. Тъй като можем във външен файл да съхраним повтарящите се форматирания и така да не пишем всеки път атрибутите за тях. Освен това, по този начин при евентуална промяна на изгледа на едно форматиране, се налага само промяна в съответния стил. Например, ако имаме 10 страници с шрифт Arial, размер 2, цвят черен, и искаме да променим цвета, но не сме задали стил, то ще се наложи ръчно да го направим във всичките страници поотделно. Задавайки стил за конкретното форматира в отделен файл, трябва само да се поправи там – на едно единствено място.

Синтаксисът на CSS е съставен от три части: селектор (selector), характеристика (property) и стойност (value).

selector {property: value}

Селекторът е нормален HTML елемент (таг), който трябва да се промени, характеристиката е атрибутът за смяна, като всяка характеристика си има стойност. Характеристиката и стойността са отделени с двоеточие и са заградени с фигурни скоби:

body {color: black}

Ако стойността представлява низ съдържащ няколко думи – необходимо е да се огради с кавички:

p {font-family: "sans serif"}

Ако трябва да се специфицира повече от една характеристика, е необходимо да се разделят с “;”.

p {text-align:center;color:red}

Селекторите с еднакви характеристики могат да се групират по между си и така да се спести писане на код. Това става като се разделят със запетая. В следващия пример са групирани всички header елементи. След прилагането на стила съответния текст ще бъде оцветен в зелено:

h1,h2,h3,h4,h5,h6 { color: green }

CSS притежава още едно важно свойство, така нареченият клас Селектор (class selector). С него може да се зададат различни стилове за един и същи HTML елемент в уеб страницата. Ако например имаме два параграфа, единият от тях е с дясно подравняване а другият с централно, с клас селектор това би изглеждало така:

p.right {text-align: right}

p.center {text-align: center}

Друга полезна опция в CSS e id селекторът. Той се различава от класа селектор, по това че последния може да се приложи върху няколко елемента върху страницата, а id селекторът е приложим само за един. Той трябва да е уникален в рамките на уеб страницата. Долу е дефиниран стил, който свързва елемента <p> с id стойността “paragraph1”

p# paragraph1{text-align: center; color: red }

Добра практика е като се пише код да се оставят коментари, каквато възжможност притежава и CSS. Коментарът започва с "/\*" и завършва с "\*/". Всичко между тях се игнорира от браузъра.

### **1.3.3 JavaScript**

***Javascript*** е скриптов език за програмиране, създаден първоначално от *Netscape Communications Corporation* и предназначен за създаване на итерактивни Web страници. Базата за създаване и развитие на езика е една спецификация, носеща името **ECMA-262**. Предназначението на ***Javascript*** е да улесни динамичната обработка на HTML документи при клиента. Скрипт кодът се включва като част от HTML кода. Изпълнението на скрипта става след интерпретация от навигатора на клиента при изтегляне на страницата от сървъра.

Динамично управление на графика, формуляри, хронологичния списък на страниците са сред най-често използваните приложения на скрипт код, написан на ***Javascript***.

***Javascript*** е първият скриптов език за програмиране в среда Web и вече не е единствен. С подобни функции и възможности е ***VBScript*** *(Visual Basic Script)*, създаден от *Microsoft*.

Сходството в наименованията между езиците ***Java*** и ***Javascript*** не ги прави еднакви, нещо повече те са различни в много отношения. Не бива да се смята, че ***Javascript*** е някакво подмножество на езика ***Java***, те просто са два различни езика за програмиране.

Подобно на всеки друг скриптов език чрез ***Javascript*** се създава скрипт-код, който подлежи на интерпретиране.

***Javascript и Java***

Различията между ***Javascript*** и ***Java*** в най-общи линии се заключават в следното:

|  |  |
| --- | --- |
| **Javascript** | **Java** |
| Код интегриран в HTML страница. | Кодът не се интегрира в HTML страница (applet). |
| Кодът се интерпретира от навигатора в момента на изпълнение. | Първичният код се компилира преди изпълнение. |
| Програмният код е упростен, но и с много ограничения. Обектно-ориентиран. | Сложен обектно-ориентиран език за програмиране от високо ниво, почти без ограничения. |
| Достъп до обектите на навигатора. | Липсват средства за достъп до обектите на навигатора. |

***Javascript и HTML***

Приложението на ***Javascript*** изисква предварителни познания по HTML. Написаният код на ***Javascript*** се вмъква директно в кода на HTML страницата по следния начин:

|  |
| --- |
| **<html>**  **<head>**  **<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">**  **<title>Javascript</title>**  **</head>**  **<body>**  **<p>Текстов параграф в HTML код.</p>**  **<p>**  **<script language="JavaScript">**  **<!-- hide from old browsers**  **document.write("Текст, изведен на екран чрез JavaScript код!")**  **// -->**  **</script>**  **</p>**  **<p>Текстов параграф в HTML код.</p>**  **</body>**  **</html>** |

JavaScript кода се вмъква в HTML документа между двойката елементи **<SCRIPT>** и **</SCRIPT>.** Този елемент е въведен в стандарта HTML 3.2 и HTML 4.01. Когато срещне тагът **<SCRIPT>**, браузъра разбира че трябва да спре интерпретирането на HTML кода и да започне да обработва скрипта, намиращ се между **<SCRIPT>** и **</SCRIPT>**. Този скрипт не е задължително да бъде написан на JavaScript. Има и други езици за  писане на скриптове, например VBScript. Затова когато пишете отварящия таг за скрипт, трябва да укажете на браузъра на какъв език ще бъде скрипта. Ако този език е JavaScript трябва да напишете **<SCRIPT LANGUAGE="JavaScript">.**

Код на ***Javascript*** може де се вмъква динамично в момента на интерпретиране на HTML страница чрез следната команда:

|  |
| --- |
| **<SCRIPT SRC="javascript\_file.js" TYPE="text/JavaScript">** |

където, файлът **javascript\_file.js** съдържа код на ***Javascript***.

Основните **характеристики** на езика ***Javascript*** се заключават в следното:

Няколко основни типа данни:

* + Цифров тип, без ясно разграничаване между цели и реални;
  + Символен тип;
  + Логически тип;
  + Нулев тип;

• Слаба типизация в езика - една променлива може във всеки един момент да промени своя тип;

• Синтактична близост на езика с езика за програмиране С, почти същите оператори и инструкции;

• Езикът е основан на недоразвит, начален обектен модел, липсва понятието клас и не поддържа унаследяване;

• Няколко базови обекта за обработка на дати, таблици и математически пресмятания;

• Ограничени възможности за манипулиране на входа и изхода, и почти никакви за въздействие с мрежови операции;

**Елементи** на езика **Javascript** са следните:

• Лексика • Коментари • Типове данни • Променливи • Ключови думи

Като език за програмиране, JS поддържа **обектно-ориентираното програмиране**

и е един от малкото езици, които позволяват да създадете приложение във всяка една възможна област - от сайтове и мобилни приложения до сървърни и desktop приложения.

Най - популярните среди за интерпретация на JavaScript са уеб браузърите, които използвате всеки ден - Chrome, Firefox, Internet Explorer и т.н. Когато зареждате даден уеб сайт е много вероятно той да съдържа JavaScript файлове, които ще се изпълнят при отварянето на сайта и ще направят изживяването при разглеждането на сайта по - приятно и динамично. Твърде вероятно е ако в сайта има падащи менюта, анимации, регистрация на потребител, дразнещи реклами, то те да са реализирани именно чрез използването на езика JavaScript.

Често чуваме, че даден код се "изпълнява на клиента" - това ще рече, че JavaScript кодът, се изпълнява в нашия browser, който играе ролята на клиент, или приемник. За да има приемник, трябва да има и предавател. В технологичният свят тези предаватели се наричат сървъри. Можем да си представим сървърите като едни много мощни компютри, до които имат достъп много хора. Всички сайтове стоят на подобни сървъри - т.е. файловете, които помагат на сайта да изглежда такъв какъвто го виждате - картинки, текстове, JavaScript файлове, се намират на конкретен сървър. Нашият browser (клиент) се свързва към сървъра, на който се намира сайта, който посещавате, и сървърът изпраща обратно необходимите файлове, за да се визуализира пред нас. Можем да си представим, че нашият browser е радиото в колата ви, на което слушате любимата програма, а сървърът е сградата от където се излъчва това предаване, макар и комуникацията между двете да протича по различен начин.

Другият популярен интерпретатор за JavaScript e NodeJS. Можем да си го представим като приложение, което инсталираме на компютъра си и той започва да разбира от JavaScript по същият начин, по който и нашият browser разбира. По този начин можем да изпълняваме JavaScript код директно на компютъра си, без необходимостта от browser. Както казахме, сървърите са просто по-мощни компютри. Те започват да разбират от JavaScript по същият начин, по който и нашият компютър - просто им се инсталира NodeJS.

# **2. Проектиране на уеб услугата**

В настоящата глава ще разгледаме различните аспекти на Web-програмирането, неговите предимства и недостатъци, както и причините за неговата популярност.

Всички сме виждали Web-базирани e-mail системи като mail.yahoo.com, mail.bg и abv.bg. Те са чудесни примери за Web-приложения.

Както знаем, Web-приложенията представляват услуги, които работят на някакъв Web-сървър и предоставят на потребителите Web-базиран интерфейс, който се визуализира от Web-браузърите им. Комуникацията между потребителските Web-браузъри и Web-приложенията се основава на заявки и отговори и се извършва по протокол HTTP. Когато потребителят напише адреса на някое Web-приложение, неговият Web-браузър изпраща на съответния Web-сървър заявка за достъп до това Web-приложение и получава динамично генериран отговор във вид на HTML или друг формат, който браузърът разбира.

**Архитектура на Java-базираните Web-приложения**

Можем да разделим Java-базираните Web-приложения на две части:

- сървърска част – представлява съвкупност от Java компоненти и JSP страници, които обработват получените от потребителя данни и в зависимост от тях динамично генерират HTML документи, CSS стилове и JavaScript код

- клиентска част – представлява съвкупността от динамично генерираните HTML документи, CSS и JavaScript код, които се визуализират от Web-браузъра на потребителя и изграждат потребителския интерфейс на приложението.

Разглеждайки Web-приложенията по този начин, можем да ги определим като многопотребителски клиент-сървър приложения, предназначени за работа в Интернет.

Когато, обаче, се разработва по-сложно програмно приложение, се използват множество готови програмни елементи и класове, които образуват специализираните програмни среди. Най-широко разпространение в момента получава специализираната програмна среда, разработена от Sun Microsystems J2EE, с която съм реализирал и моето уеб приложение.

За реализирането на моето приложение съм използвал Spring Framework технологична рамка (framework) с отворен код (open source), чието основно ядро е неговият **IoC** (Inversion of Control) контейнер, който осигурява логически средства за конфигуриране и управление на Java обекти с помощта на reflection. Контейнерът е отговорен за управлението на жизнения цикъл на специфични обекти: създаването на тези обекти, извикването на инициализиращите им методи и конфигурирането им, свързвайки ги заедно. Обектите, създадени от контейнера се наричат още управлявани обекти или **бийнове** (beans). Контейнерът може да бъде конфигуриран чрез зареждане на XML файлове или специфични Java анотации от конфигуриращите класове. За по-лесна и безпроблемна работа съм използвал Java анотации за конфигуриране на контейнера.

Обектите(инстанциите на класовете) могат да бъдат получени чрез dependency injection. Dependency injection е софтуерен шаблон за дизайн, който опростява Обектно-ориентираното програмиране (ООП) чрез намаляване на зависимостите между отделните класове. Така при създаването на един проект, използвайки някой обектно-ориентиран език, се избягва зависимостта от конкретни имплементации, и се набляга повече на абстракции. Това се нарича още обръщане на контрола (inversion of control).

При dependency injection обектите се проектират по начин, при който те получават инстанции на обекти, от които зависят, за да изпълняват бизнес логика, от различен външен код, вместо да са конструирани вътре в него(благодарение на анотациите, ние оказваме, че когато имаме нужда от даден обект от даден клас Spring го създава за нас без да е нужно ние изрично да го създаване с ключовата дума new). Зависимият елемент определя на какъв софтуерен компонент зависи за да си свърши работата, а контейнерът решава какви конкретни класове отговарят на изискванията на зависимия обект, и ги предоставя като зависимости. По този начин контейнерът може да избира конкретните имплементации, които да се използват по време на изпълнение на програмата (run-time), а не по време на компилация (compile-time). По време на изпълнение (run-time) могат да бъдат създадени множество различни инстанции на даден софтуерен компонент, без значение каква конкретна е била инжектирана.

Описаното накратко до тук представлява ядрото на уеб приложението и на какъв принцип то ще работи. За изграждането му ще използваме архитектурния модел Model-View-Controller (MVC), който разделя дадено ссофтуерно приложение на три взаимносвързани части, така че да се отделят вътрешни представяния на информация(бизнес-логиката) от начините, по които информацията се представя на потребителите:  *контролери*, *изгледи*, *модели.*

Всичко казано до тук и всички допълнителни технологии, използвани за реализанирането на уеб приложението ще бъдат представени по-подробно в следващите подглави.

## 2.1 Технологии за реализиране на уеб услугата Soundcloud

Благодарение на това, че Java е един от най-популярните и използвани езици за програмиране има голям набор от допълнителни технологии, които дават изключително голяма възможност за създаването на сложни и мощни уеб приложения. Използваните от мен технологии за изграждането на Soundcloud уеб услугата са: Java EE, Spring Framework, Maven, JPA(Hibernate), JDBC, Lombok, Tomcat, MySQL.

### 2.1.1 Java EE

Платформата J2EE предлага компонентен подход към проектиране, разработка и внедряване на корпоративни приложения и е основно ориентирана за работа в Web среда, както в Интернет, така и в локални мрежи.

Чрез нея се разработват трислойни програмни приложения, които се разполагат на три различни места: машината на клиента, J2EE сървърната машина и машината, съдържаща базата данни с които работи приложението (‘задна’ част). J2EE представлява средния слой, който извършва основната обработка на данните (бизнес-логиката).

J2EE се състои от компоненти, които представляват програми, написани на Java, и се компилирани по същия начин, както стандартните Java програми (класове). Разликата между стандартните Java класове и J2EE компонентите е, че J2EE компонентите се асемблират (комплектоват) в J2EE приложение, проверени са за съответствие с J2EE спецификацията и са подготвени за управление от J2EE сървър(в нашия случай използваме Tomcat web server).

Най-просто можем да кажем, че Java SE се използва за разработването на desktop приложения и е основата на създаването на Java програми. Състои се от набор от инструменти за разработка, технологии за внедряване, библиотеки и набор от класове. Java EE е изграден върху Java SE и се използва за разработването на уеб и големи корпоративни приложения.

### 2.1.2 Tomcat

Както споменахме, за да създаден уеб услуга се нуждаем от сървър, до който клиентите да изпращат заявки, на базата на които той да предоставя някаква информация. Казано по друг начин Web-сървърите са приложения, които “слушат” на определен TCP порт (обикновено това е стандартният порт за протокола HTTP – 80), и отговарят на HTTP заявките, получени от клиентски приложения (най-често това са Web-браузърите). Платформата J2EE предоставя именно такъв сървър – Tomcat.

### 2.1.3 HTTP протокол