# Дипломна работа

на тема

**Уеб приложение за покупко-продажба на автомобили**

Дипломант: **Александър Александров Свиларов**

ученик от 12А клас в

ЧПГДН "СофтУни Светлина"

Научен ръководител: **Калоян Костадинов**

Рецензент: **XXX XXX XXX**

Дата: **12 май 2023**

Сесия: май-юни 2023 г.

София, 2023 г.

*Указания:*

* *Вашата дипломна работа трябва да има обем от* ***минимум 30 страници****.*
* *При попълването на документа да се използва стил „****Normal****“, в който е настроен шрифта и размера, така, че да бъдат изпълнени критериите за форматиране.*
* *След попълването на основната информация, всички указания (текстовете в сиво) да се премахнат.*
* *За формулирането на отделните точки в съдържанието използвайте стиловете „****Heading 1****“ (за основни дялове), „****Heading 2****“ (за отделните точки в дяловете), „****Heading 3****“ (за отделните подточки във всяка точка на дяловете). За описателния текст в точките използвайте стил „****Normal****“. За кода да се използва стил „****Code****“.*
* *Всяка* ***фигура****,* ***картинка*** *и* ***таблица*** *да бъдат означени и номерирани. Използвайте пестеливо разполагане на текста спрямо фигурите/картинките, така, че при разпечатването да не се хаби излишен брой листи.*
* *Спазвайте изискванията за ползване и* ***цитиране*** *на чужди разработки (авторско право).*
* *След като приключите с работата по всички точки и заличите ненужните текстове, направете обновяване на* ***Съдържанието****, така, че всички нововъведени точки и подточки да придобият местоположение спрямо страниците*
* *В меню File 🡪 Info въведете* ***автора на документа****, така, че при проверка от страна на рецензента аворството да бъде потвърдено недвусмислено.*
* *Запишете* ***файла с подходящо име****: кратко заглавие + име на дипломанта + класа, примерно:* ***Polivna-sistema-Stefan-Kirov-12B-klas.docx****.*

# Съдържание

[Дипломна работа 1](#_30j0zll)

[Съдържание 3](#_1fob9te)

[Увод 4](#_3znysh7)

[Проблем 4](#_2et92p0)

[Цели на дипломния проект 4](#_tyjcwt)

[Задачи, произтичащи от целите 4](#_3dy6vkm)

[Глава 1. Проучване 6](#_1t3h5sf)

[Глава 2. Използвани технологии 7](#_4d34og8)

[Глава 3. Проектиране и имплементация 8](#_2s8eyo1)

[Архитектура на системата 8](#_17dp8vu)

[Имплементация на системата 8](#_3rdcrjn)

[Тестове 9](#_26in1rg)

[Внедряване (deployment) 10](#_lnxbz9)

[Глава 4. Ръководство за потребителя (смени заглавието) 11](#_35nkun2)

[Заключение 12](#_44sinio)

[Информационни източници 13](#_2jxsxqh)

[Приложение 1 / 2 / … 14](#_z337ya)

[Рецензия на дипломен проект 15](#_3j2qqm3)

# 

# Увод

DeltaCar е уеб приложение, което има за цел да улесни процеса на покупко-продажба на автомобили. То предоставя на потребителите възможност да намерят подходящия автомобил за тях, да сравняват различни марки и модели и да закупят автомобила онлайн.

Целта на проекта е да създаде едно лесно за използване приложение, което да предоставя на потребителите добра и удобна услуга. То ще бъде достъпно за всички, които желаят да купят или продадат автомобил, и ще съдържа полезна информация, която ще ги насочи към правилния избор.

Приложението ще предлага разнообразни филтри и опции за търсене, които да помогнат на потребителите да намерят автомобила, който отговаря на техните нужди и предпочитания. След като потребителите изберат автомобила, който искат да закупят, те ще могат да се свържат директно с продавача, за да се договорят за цената и да уточнят всички детайли по сделката. Например, потребителите могат да търсят автомобили по марка, модел, цена, година на производство, тип на двигателя, тип на горивната система, пробег и други параметри. Тези филтри помагат на потребителите да се фокусират върху тези автомобили, които отговарят на техните нужди. Системата за търсене на DeltaCar се отличава също и с възможността да се търсят автомобили по ключови думи, което е особено полезно за тези потребители, които търсят автомобил с даден проблем.

DeltaCar ще предлага и много други полезни функции за потребителите. Те ще могат да преглеждат описания на автомобилите, да разглеждат снимки и детайлни характеристики. Приложението ще предлага и интегрирани услуги за проверка на автомобили, които да помогнат на потребителите да се уверят, че сделката, която сключват, е сигурна и безопасна.

Комбинацията от множеството параметри за търсене и гъвкавостта на системата правят DeltaCar по-добро от другите сайтове за покупко-продажба на автомобили. Това приложение предоставя на потребителите иновативен начин за намиране на идеалния автомобил за тях.

## Проблем

## DeltaCar решава проблема на хората, които търсят удобен и бърз начин за покупка на автомобил. Сайтът ще предоставя богат избор от автомобили от различни марки и модели, с различни характеристики и цени.

## Проблемът, който сайтът решава, е, че традиционните начини за покупка на автомобили - от дилър или частен продавач, могат да бъдат трудни и времеемки за хората. При това, те може да не могат да намерят точно това, което търсят. DeltaCar предлага удобство и лесно търсене на автомобили в едно място, с информация за модела, състоянието и цената на всяко едно предложение. Освен това, сайтът осигурява безопасност и сигурност на покупката, като се грижи за всички необходими документи и проверки на автомобилите.

## Този проблем е актуален за милиони хора, които търсят нов или употребяван автомобил в света. Сайтът DeltaCar предоставя решение на този проблем за хора, които търсят лесен и бърз начин да закупят автомобил, както и за тези, които искат да продадат своя автомобил. Освен това, сайтът е полезен за хора, които търсят конкретен модел или марка автомобил, който може да бъде трудно да се открие в традиционните магазини или от частни продавачи.

## Цели на дипломния проект

Целите на дипломния проект DeltaCar са да разработи уеб базирана платформа за продажба на автомобили, която да предоставя на потребителите бърз и лесен начин за търсене и закупуване на автомобили, както и за продажба на автомобили.

Специфичните цели на проекта включват следното:

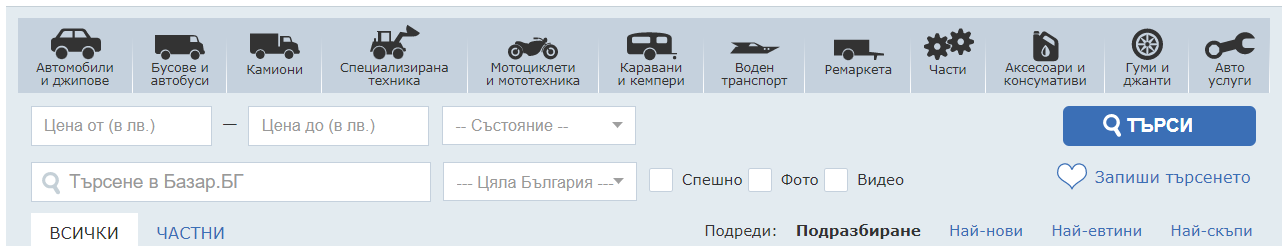
* Разработване на потребителски интерфейс, който да е удобен, лесен за използване и да предоставя ясна информация за всеки автомобил, който е наличен за продажба.
* Интегриране на търсачка за автомобили, която да позволява на потребителите да търсят автомобили по марка, модел, година на производство, цена и други характеристики.
* Интегриране на система за онлайн плащания, която да позволява на потребителите да закупуват автомобили директно през уебсайта. ***(Планирано)***
* Разработване на система за управление на продажбите, която да позволява на администраторите на сайта да управляват автомобилите, които са налични за продажба, както и да проследяват статуса на всяка продажба. ***(Планирано)***
* Интегриране на система за комуникация между потребителите и администраторите на сайта, която да позволява на потребителите да получават бърз и ефективен отговор на въпроси и да получават помощ при нужда. ***(Планирано)***

# Проучване

Закупуването или продажбата на автомобил може да бъде една времеемка и не винаги лесна задача. За да има успех продавачът, той трябва да притежава автомобил в добро състояние, привлекателен външен вид, добро описание, честно отношение към клиентите и тн. Преди съществуването на интернет пространството подобни сделки са се извършвали най-често в автокъщи – човек отива там, говори с човек, отговорен за продажбите, обяснява какво търси, изискванията си и предпочитанията си, след което бива насочен към няколко автомобила, с които автокъщата разполага и той избира този, който му допада най-много.

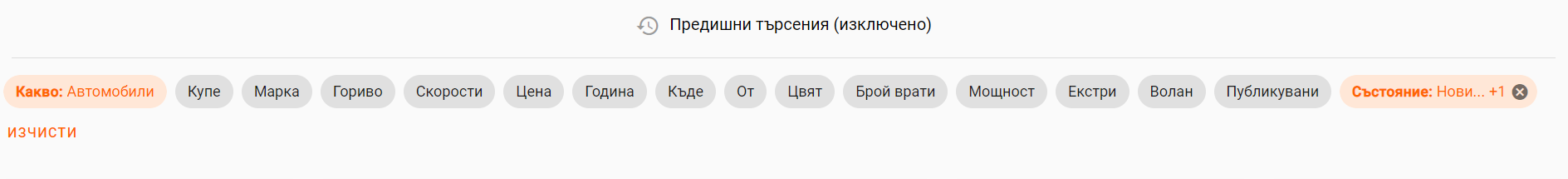
Един от основните проблеми на този начин за покупка на автомобил е времето, което отнема, без гарантиран резутат. Винаги може да се случи така, че автокъщата да не разполага с автомобил, който да покрива изискванията на даден клиент – тогава целият този процес започва отначало. След 2-3 различни срещи денят е загубен, а купувачът все още няма желаният автомобил. С развитието на интернет, подобни проблеми бързо намират своите решения – сферата на автомобилите не е изключение. В днешно време най-популярният начин за покупка на автомобил е онлайн.

В България имаме няколко основни сайта за тази цел – Mobile.bg, Cars.bg, Auto.bg, както и Bazar.bg и Olx.bg. Всеки един от тези сайтове има своите плюсове и минуси. В сайтове като Bazar.bg търсачката е силно ограничена по ключови думи, локация, състояние и цена:



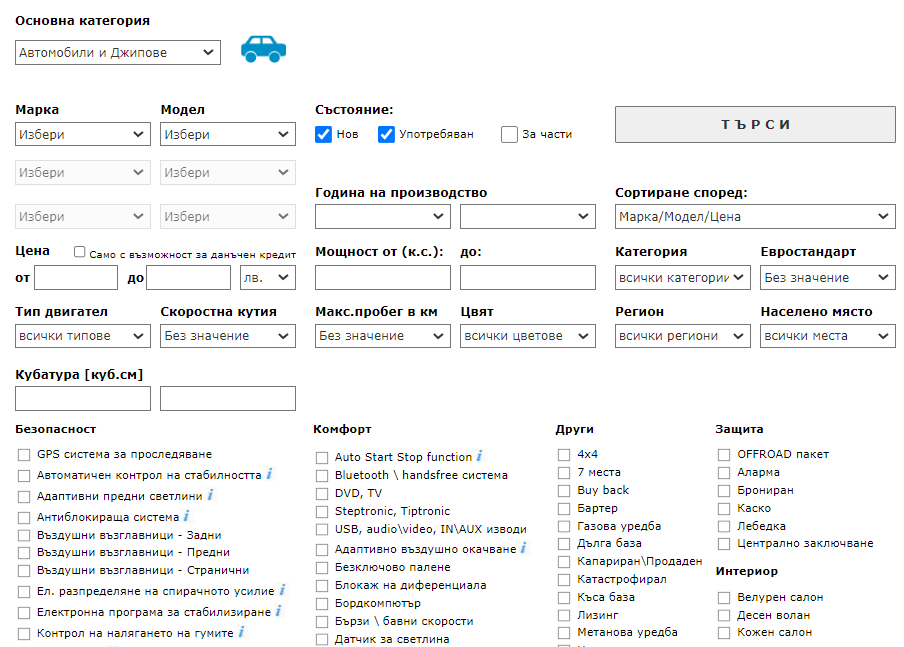
Фигура : Търсачката в Bazar.bg

В Auto.bg наблюдаваме подобна ситуация – ограничена търсачка, този път с малко повече опции:



Фигура : Търсачката в Auto.bg

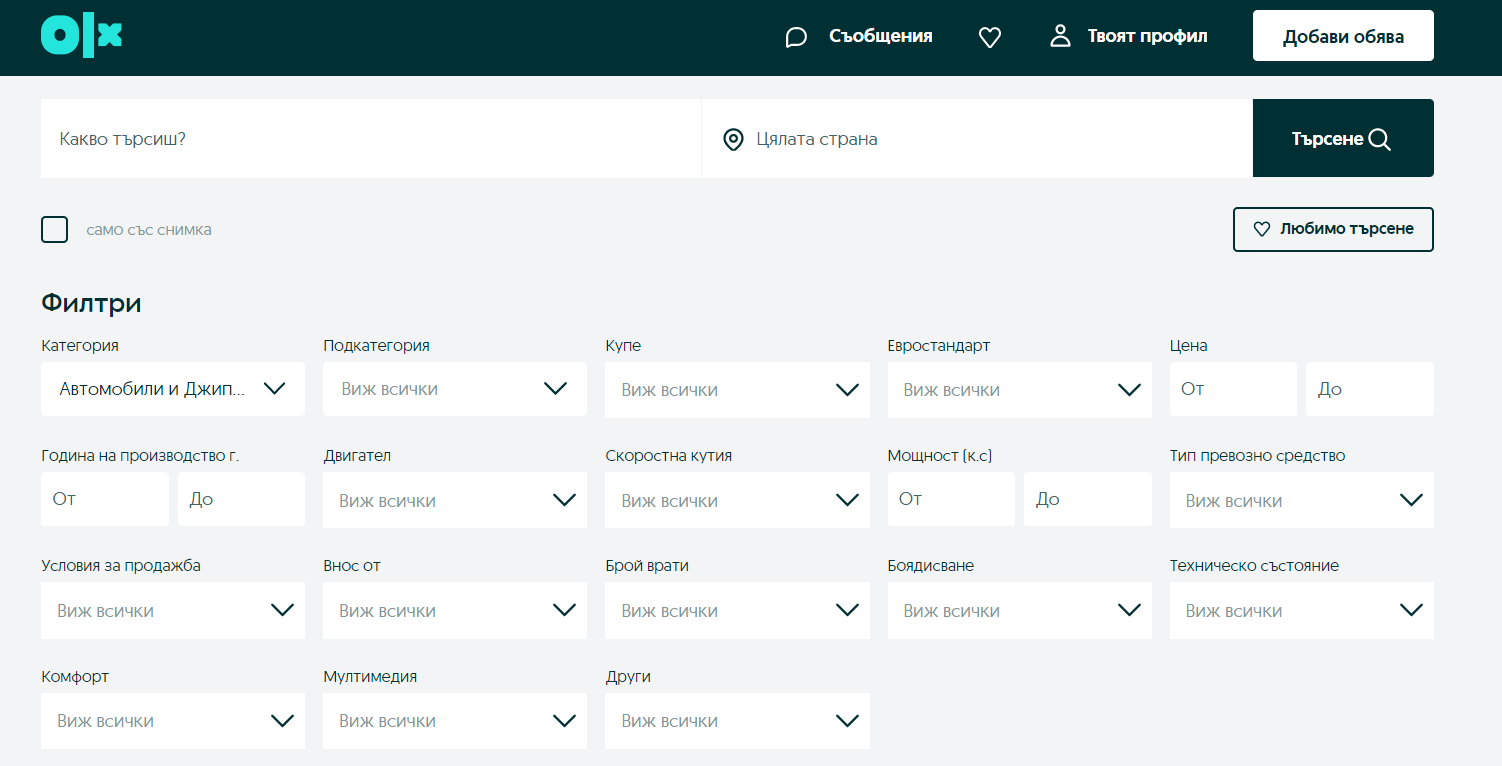
В сайтове като Mobile.bg ще забележим, че търсачката е сравнително по-обогатена, имайки редица опции, от които да изберем:



Фигура : Търсачката в Mobile.bg

Mobile.bg разполага с много добре разработена търсачка, която включва почти всичко, от което се нуждае един клиент. Единственото, което липсва, е търсенето по ключови думи – една важна подробност, която често бива забравена. Има хора, които търсят коли със специфичен проблем – например има механици, които търсят кола за „гарнитура на глава“, която да оправят и продадат за повече пари. DeltaCar се отличава от другите сайтове за покупко-продажба на автомобили благодарение на своята по-добре разработена система за търсене, която обединява различните видове търсачки, показани по-горе. Потребителите имат достъп до множество параметри, които могат да използват, за да намерят точно този автомобил, който отговаря на техните нужди и предпочитания. Функцията за за търсене по ключови думи е много важна част от системата за търсене, с която разполага DeltaCar.

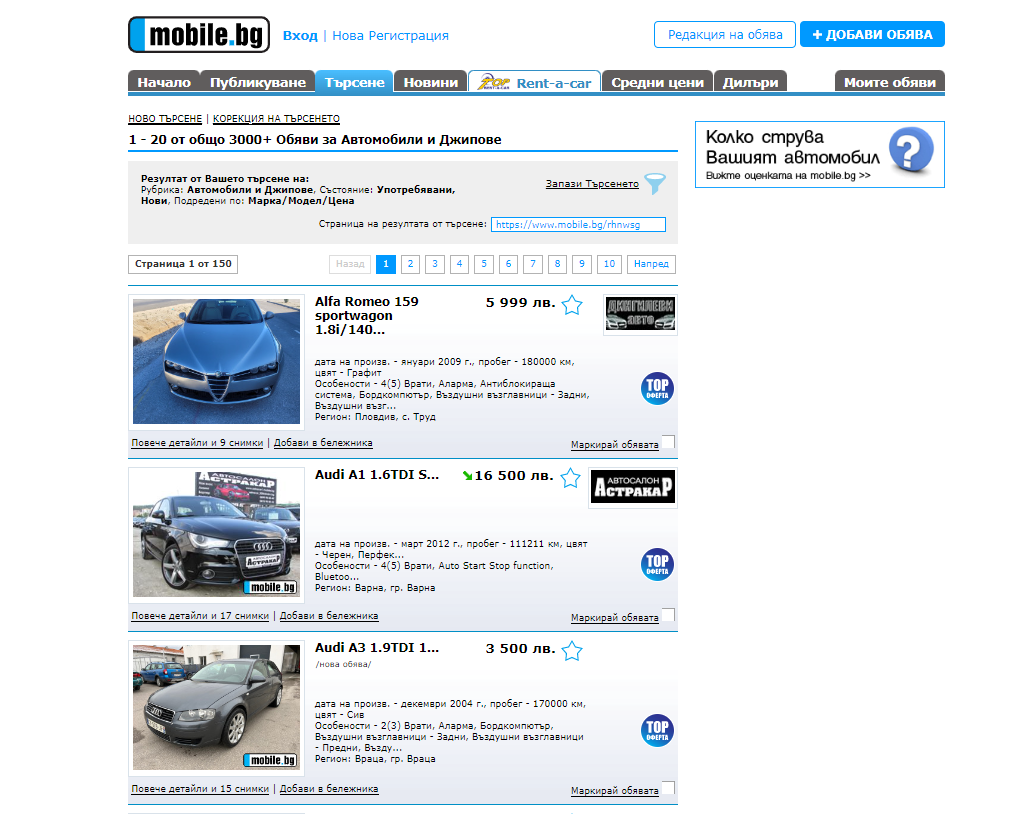
От всички подобни сайтове изненадващо най-добра търсачка има Olx.bg, като единственият проблем там, е че сайтът не е стриктно предназначен за покупко-продажба на автомобили.



Фигура : Tърсачката в Olx.bg

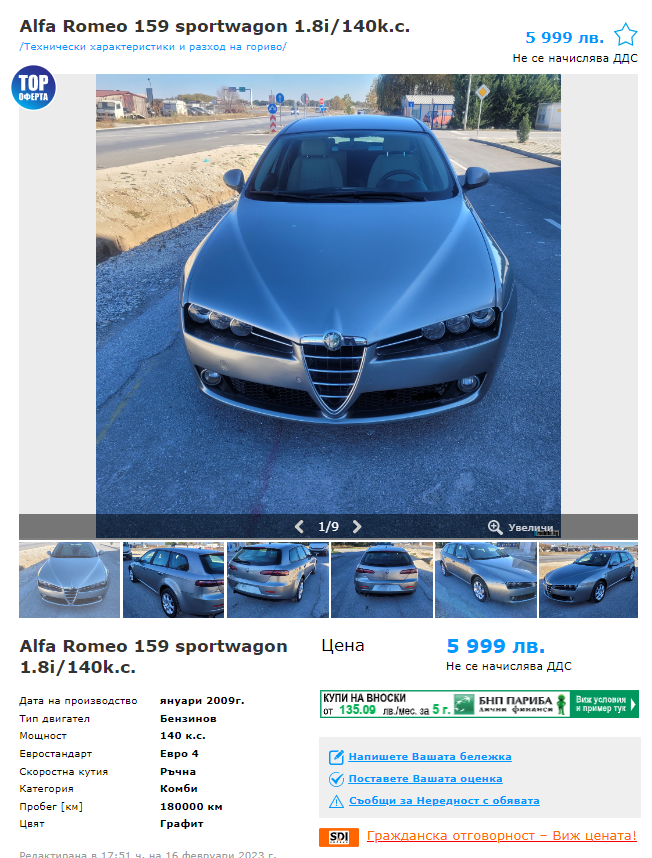
Нека разгледаме и дизайна на различните уебсайтове, които предлагат услуга за покупко-продажба на автомобили.

В Mobile.bg дизайнът е семпъл, но вече е леко остарял. Със сигурност не е в крак с тенденциите в уеб дизайна, но пък е лесен за разбиране и потребителите не се затрудняват да отгатнат как да навигират из сайта.

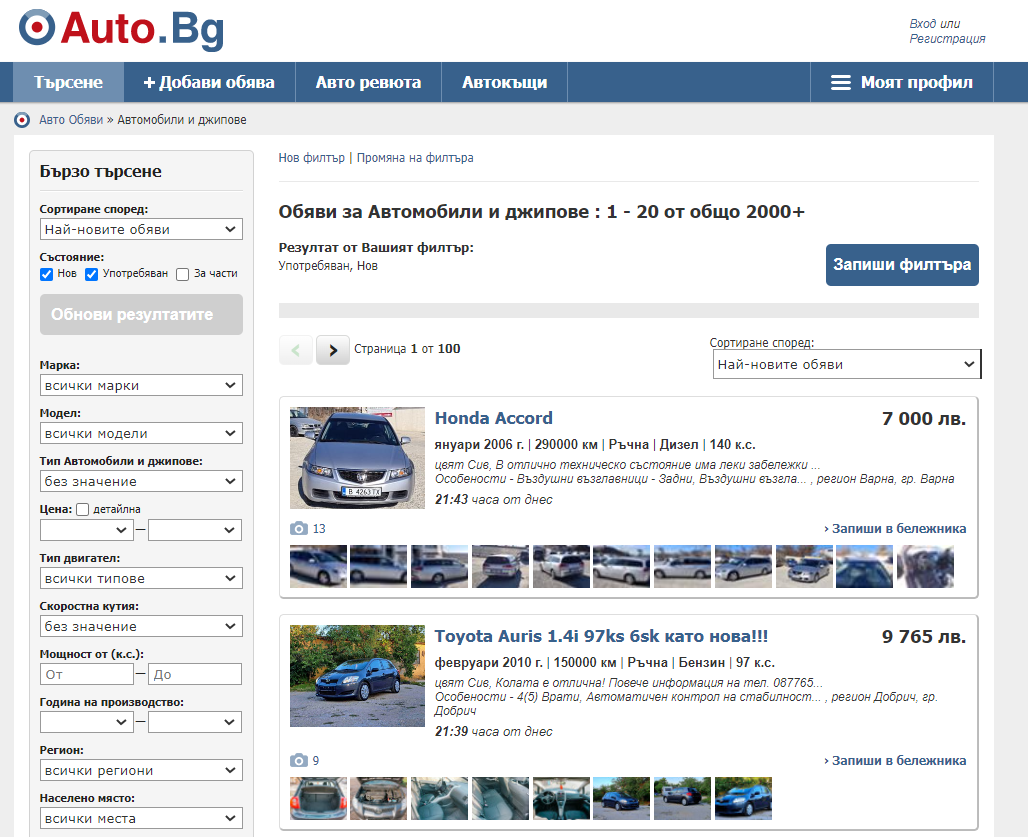


Фигура : Дизайн в Mobile.bg

Дизайнът в самата страница, където се намира обявата, е добре структуриран, като е предтавена най-важната за потребителя информация:

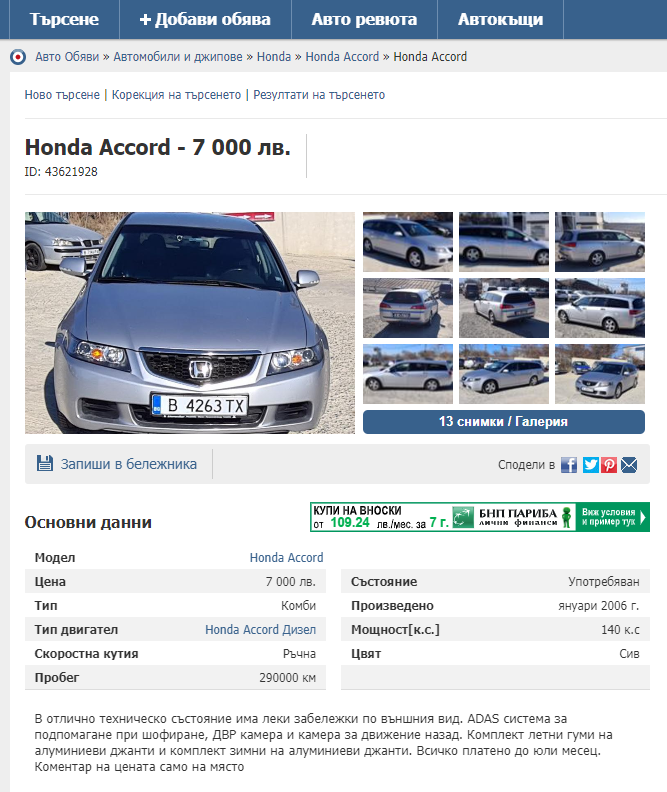


Фигура : Дизайн на обява в Mobile.bg

Дизайнът в Аuto.bg е подобен, но тук виждаме подобрение в Navigation Bar-a, както и опцията за Бързо Търсене, която е полезна.

Фигура : Дизайн в Auto.bg

Подобен дизайн наблюдаваме и в страницата на обявата:



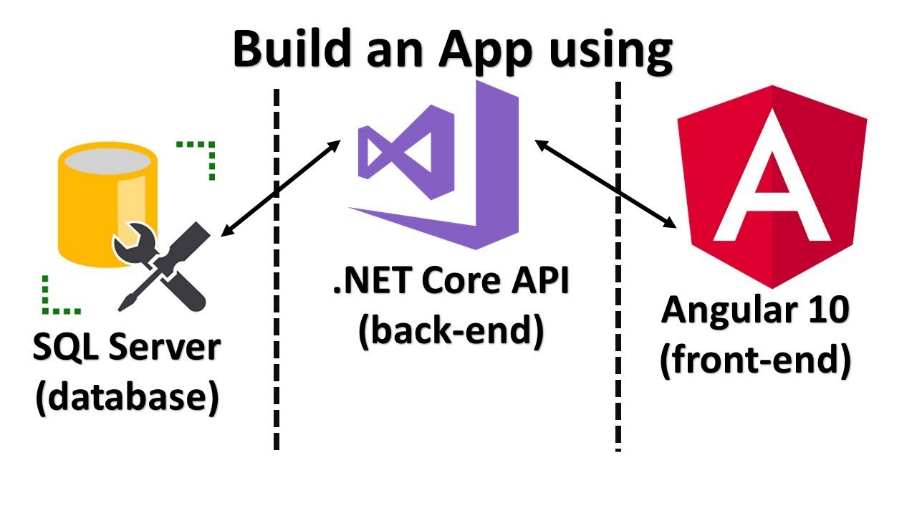
Фигура : Дизайн на обява в Auto.bg

# Глава 2. Използвани технологии

За да създадем уеб приложението, ще използваме различни технологии за backend-a и frontend-a. За backend-a ще използваме C# и Entity Framework, а за frontend-a – Angular (с Typescript).

# 2.1 Backend

Първата стъпка в разработката на уеб приложения е създаването на Backend-a, който случи за достъпването на данните, тяхното модифициране и т.н. В модела клиент-сървър клиентът обикновено се счита за frontend, а сървърът обикновено се счита за backend.



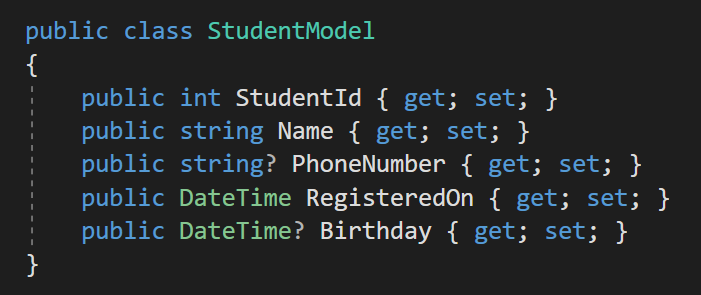
Фигура : Структурата на уеб приложението (технологии)

# 2.1.1 Какво е ORM и как функционира в Entity Framework?

ORM е начин за конвертиране на данни между релационна база данни и обектно-ориентиран език за програмиране, създавайки база данни с виртуални обекти, която може да се използва от обетно-ориентирания език за програмиране.

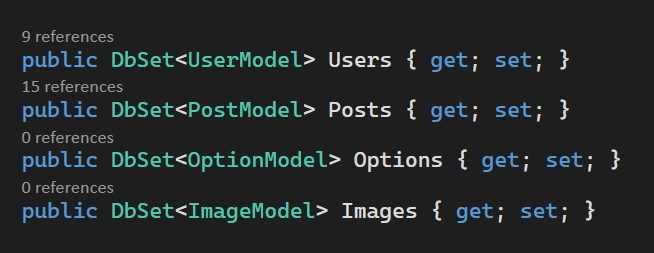
В Entity Framework това се случва чрез помощта на модели и миграции - процес на автоматично създаване и актуализиране на релационната база данни на основата на промените в моделите. Това означава, че EF Core може да проследява промените в модела на данните и да създава, актуализира или премахва таблиците и колоните в базата данни, когато е необходимо.

Един модел в Entity Framework представлява клас, който има различни property-та. Те могат да бъдат различни типове, като Entity Framework сам ще ги превърне в типове, които SQL Server да разбере – например тип ***string*** ще бъде превърнат в тип ***nvarchar***, тип ***DateTime*** ще бъде превърнат в ***datetime2*** и така нататък:



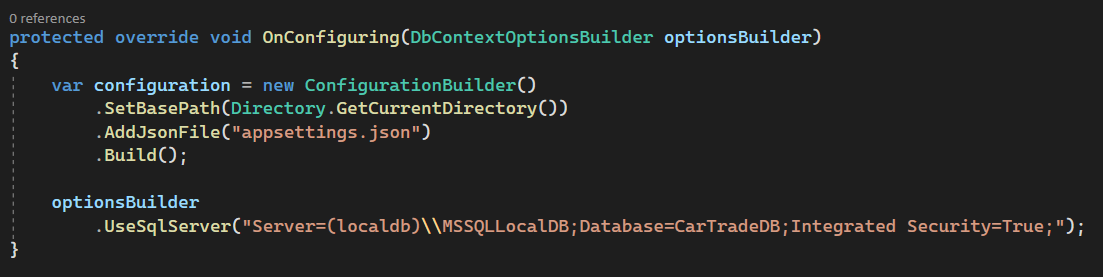
Фигура : Модел в Entity Framework

Тези модели можем да превърнем в таблици, използвайки т.нар ***DbSet***, където задаваме и желаното име на таблицата:



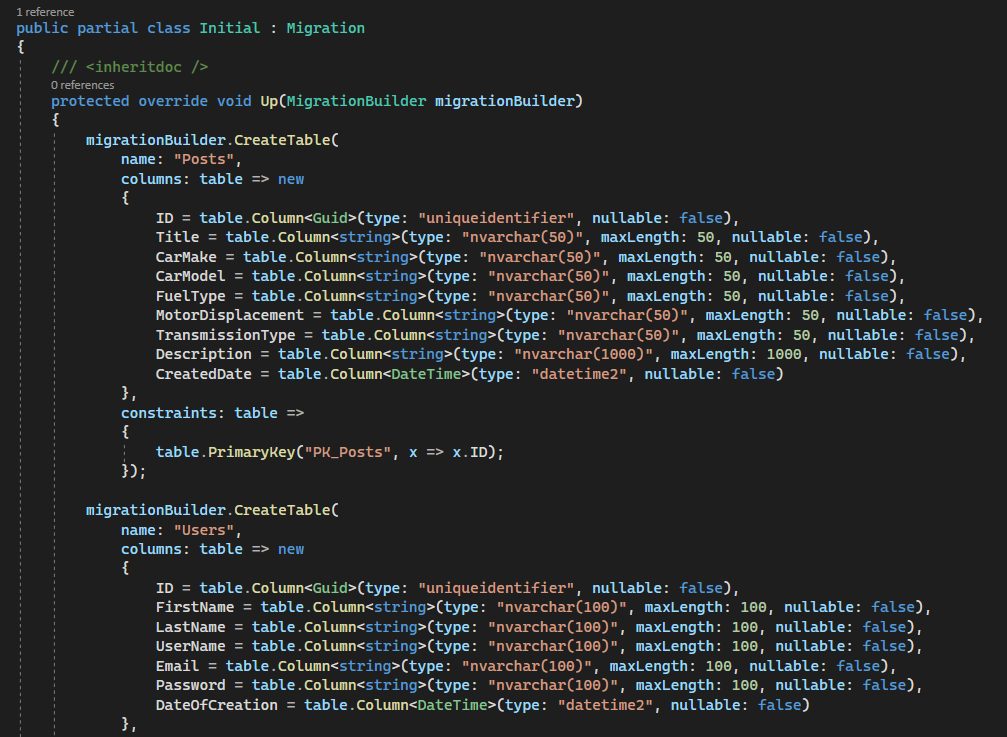
Фигура : Използване на DbSet в Entity Framework

Разбира се, трябва да имаме и конфигурация, коятно да посочи на Entity Framework как да намери / къде да създаде базата данни, която искаме да създаваме, използваме или модифицираме. Това можем да постигнем чрез метода **OnConfiguring**:



Фигура : Метод OnConfiguring в Entity Framework

След като имаме модел и сме решили, че той ще бъде таблица, задавайки го като DbSet, трябва да направим миграция. Миграцията в Entity Framework се извършва с помощта на инструмента за миграции на Entity Framework, който позволява на разработчиците да генерират т.нар миграционни скриптове и да ги приложат върху базата данни. Този процес включва изграждане на миграционна история, която записва всички промени в модела на данните, като например добавяне или премахване на таблица или колона. Когато се генерира нова миграция, инструмента за миграции автоматично генерира необходимите SQL команди за актуализиране на базата данни.

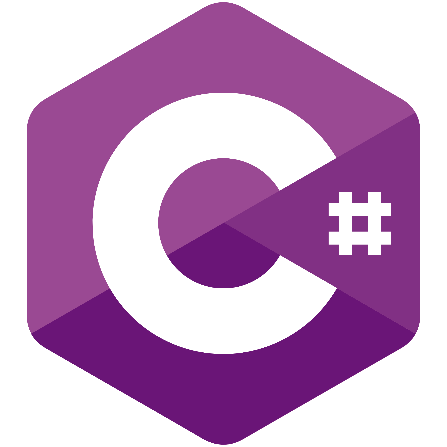


Фигура : Миграционен скрипт в Entity Framewok

# 2.1.2 Защо C# е добър при разработката на уеб приложения?

Ето някои от основните причини защо **C#** е сигурен избор при разработването на уеб приложения:

* Обектно-ориентиран подход: **C#** е език със строга типизация, който поддържа обектно-ориентирано програмиране. Това позволява на програмистите да създават модулни и структурирани приложения, които са лесни за поддръжка и мащабиране.
* **ASP.NET: ASP.NET** е платформа за създаване на уеб приложения, която използва **C#** като основен език за програмиране. **ASP.NET** предоставя мощни инструменти за създаване на сигурни и ефективни уеб приложения.
* Голяма общност: **C#** е един от най-популярните езици за програмиране, който има голяма и активна общност от разработчици. Това означава, че има много ресурси, като форуми, уроци и документация на разположение.
* Лесно мащабиране: **C#** езикът и **ASP.NET** платформата предлагат много възможности за мащабиране на уеб приложенията. Това означава, че програмистите могат да добавят нови функции и да променят кода на приложението, без да се налага да го изграждат отново.
* Сигурност: **C#** и **ASP.NET** са добре известни със своята сигурност. Например, **ASP.NET** предоставя вградена защита срещу най-различни видове атаки, като SQL инжекции, кръстосан скриптинг и други.



# 2.2 Frontend

След като backend-а е готов, стигаме до реализацията на приложението в уеб среда, за което ни е нужен Frontend. Frontend-a в едно уеб приложение е частта, която потребителите виждат и с която взаимодействат.

# 2.2.1 Angular и защо да използваме него?

След като backend-а е готов, стигаме до реализацията на приложението в уеб среда, за което ни е нужен Frontend. Frontend-a в едно уеб приложение е частта, която потребителите виждат и с която взаимодействат.

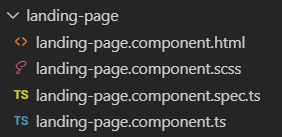
Angular е framework за разработка на едностранични приложения (SPA - Single Page Applications) с помощта на HTML, CSS и TypeScript. Angular е разработен от Google и предоставя много функционалности, които подобряват разработката на уеб приложения, като например:

* **Two-way data binding (Двупосочно свързване на данни)**: Двупосочното свързване на данни на Angular позволява на разработчиците да създадат връзка между модела и изгледа. Всички промени, направени в модела, автоматично се отразяват в изгледа и обратно.
* **Dependency injection (инжектиране на зависимости)**: Системата за инжектиране на зависимости на Angular позволява на разработчиците лесно да управляват зависимостите и да създават компоненти за многократна употреба.
* **Routing (маршрутизиране)**: Вграденият маршрутизатор на Angular позволява на разработчиците лесно да навигират между различните страници на тяхното приложение.
* **Directives (директиви)**: Директивите на Angular позволяват на разработчиците да създават персонализирани HTML тагове и атрибути, които могат да се използват за добавяне на функционалност към тяхното приложение.
* **Testing (тестване)**: Angular предоставя надеждна рамка за тестване, с която разработчиците могат да тестват своите приложения, гарантирайки, че те са надеждни и без грешки.

# 2.2.2 Компонентите в Angular

В Angular, компонентите са основните строителни блокове на приложението. Компонентите са самостоятелни единици, които обхващат HTML, CSS и JavaScript кода и позволяват на разработчиците да създават многократно използваеми UI компоненти за приложенията си.

Компонентите в Angular са създадени чрез декоратора **@Component**, който определя метаданните на компонента, като името, селектора, шаблона (template), стиловете и други. Те също така имат свойства (properties), които могат да бъдат свързани двустранно с данни в шаблона на компонента чрез двустранно свързване на данни (two-way data binding). Когато генерираме един компонент в Angular, файловете, които той геерира, са следните:



Фигура : Файловете, които съставят един компонент в Angular

Имаме 4 файла, като всеки един от тях има своята роля:

* Файл с разширение **.html** – в него се пише HTML кодът на компонента, неговата структура.
* Файл с разширение **.css** (**scss**) – в него се пише визуалното оформление на компонента, изгледът.
* Файл с разширение **.spec.ts** – в него компонентът се създава и инициализира.
* Файл с разширение **.ts** – в него можем да пишем frontend-ската логика, която искаме (валидации, пренасочване, повикване на функции и така нататък)

# 2.2.3 Service-ите в Angular

Service-ите в Angular са основнен начин за обмяна на данни между frontend-a и backend-a на дадено уеб приложение. Важно е тяхното използване, защото то подпомага лесното преизползване на вече написан код, което на свой ред е една от основните черти на оптимизацията.

В service-ите ние се свързваме с backend-a чрез така наречените **HTTP** **заявки**, като най-често използваните са следните:

* **GET**: Извлича информация или данни от сървъра за определен ресурс. Това е най-често използваният метод на HTTP заявка и често се използва за извличане на уеб страници, изображения и други видове файлове.
* **POST**: Подаване на данни към сървъра за обработка, обикновено като част от подаване на формуляр или за създаване на нов ресурс.
* **PUT**: Актуализира съществуващ ресурс на сървъра с нови данни.
* **DELETE**: Премахва ресурс от сървъра.

Именно чрез тези заявки разработчиците правят връзката между frontend-a и backend-a.



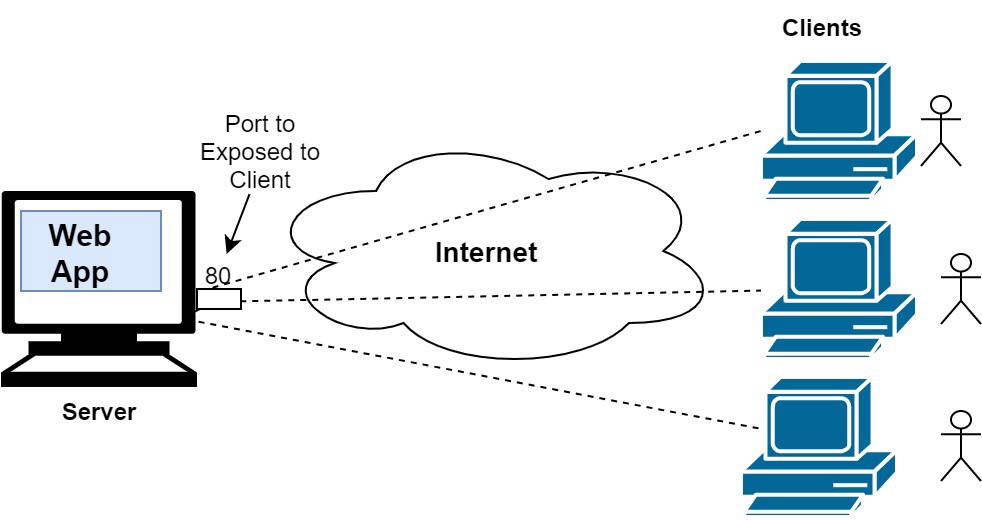
Фигура : Примерен service в Angular

# 3. Проектиране и имплементация

Сега ще обясня как е проектирано и имплементрано уеб приложението DeltaCar. Ще разгледаме логиката в backend-a, data-access layer-a, миграциите, моделите, frontend-а, как се осъществява връзката между клиентската и сървърната част и др.

## 3.1 Архитектура на системата

Архитектурата на проекта ще е client-server архитектура, която изглежда по следният начин:



Фигура : Client-Server архитектура

Client-server архитектурата е модел на софтуерен дизайн, където приложението е разделено на две основни компонента - клиент и сървър. Клиентът е софтуер, който изпълнява заявки към сървъра, за да получава информация или да извършва операции. Сървърът е софтуер, който отговаря на заявките на клиента и обработва информацията, която е необходима за тяхната изпълнение.

Системата работи като клиентите комуникират със сървъра чрез мрежата, като изпращат заявки за информация или изпълнение на операции. Сървърът получава тези заявки, обработва ги и изпраща отговорите обратно към клиентите. Това позволява на различни клиенти да работят едновременно с един и същи сървър, като споделят обща информация и ресурси.

Предимствата на този модел включват улеснена съвместна работа и управление на данни, както и по-лесно скалиране на приложенията. В допълнение, този модел предоставя по-голяма сигурност на данните, тъй като те са съхранени централизирано на сървъра и могат да бъдат защитени с подходящи механизми за достъп и сигурност.

## Имплементация на системата

*Описва се* ***имплементацията на системата*** *– основни компоненти (примерно сървърна част, mobile app, Web app), по-важни класове, слоеве (примерно data layer, UI layer), използвани външни библиотеки и фреймуърци.*

*Пример:*

*Подписването на документ в уеб браузъра с цифров сертификат от PFX файл се реализира чрез Java аплет “****DigitalSignerApplet****”. Той се състои основно от класа DigitalSignerApplet, който реализира основната му функционалност*

|  |
| --- |
| **DigitalSignerApplet.java** |
| **import java.applet.Applet;**  **import java.awt.\*;**  **…**  **public class DigitalSignerApplet extends Applet {**  **…**  **}** |

*Този клас имплементира следните стъпки: …*

*…*

*Подписването на документ в уеб браузъра със смарт карта се реализира чрез Java аплет “****SmartCardSignerApplet****”. Той работи по следния начин …*

*…*

*Обработката на подписания документ от страна на сървъра и верификацията на положения цифров подпис се извършва от специализирано уеб приложение. То е изградено по J2EE стандартите за уеб приложения с MVC framework “Struts” и работи по следния начин: …*

## Тестове

*Включва описание на* ***проведените тестове*** *(****ръчни****,* ***автоматизирани****), тяхната имплементация и резултати. Описва се и построената* ***CI система*** *(ако има такава).*

*Пример:*

*От страна на клиента системата успешно тествахме с:*

* *Java Plug-In 1.4 (за аплета с PKCS#12 хранилище) и Java Plug-In 1.5 (за аплета със смарт карта)*
* *уеб браузъри Internet Explorer 5.0, Internet Explorer 6.0, Mozilla 1.3, Mozilla 1.7, Firefox 1.0, Netscape Communicator 4.5 и Netscape 6.1 върху Windows 98, Windows 2000, Windows XP и Windows 2003 Server*
* *Mozilla 1.3 върху Red Hat Linux 8.0 (с графична среда KDE 3.0)*
* *смарт карта Utimaco Safeware с PKCS#11 драйвери Utimaco SafeGuard Smartcard Provider и карточетец ACR38*
* *цифрови сертификати от VeriSign, Thawte, GlobalSign и StampIT*

*Върху всички посочени платформи и уеб браузъри системата работи коректно и според очакванията...*

*Системата беше разгледана и тествана и от специалисти от Асоциация за информационна сигурност (ISECA), както и от експерти, работещи в областта на удостоверителните услуги от „Информационно обслужване” АД (StampIT). Изказаното мнение…*

## Внедряване (deployment)

*Представя се* ***начинът на достъпване на системата****. Тя може да бъде качена в Интернет, да се хоства в облачна среда като Azure или да се изтегля като инсталатор от публичен уеб сайт. Предоставя се* ***линк за достъп*** *и/или* ***инструкции*** *за достъп до приложението.*

*Пример:*

*За да изпълним приложението DocumentSigningDemoWebApp.war е необходимо да го инсталираме (deployment) на някой J2EE сървър или сървър за Java уеб приложения (Servlet container).*

*Ако използваме сървъра за J2EE уеб приложения Apache Tomcat, е достатъчно до копираме файла DocumentSigningDemoWebApp.war в директория %TOMCAT\_HOME%/webapps7 и да стартираме Tomcat. След това (при стандартна инсталация и конфигурация на Tomcat) приложението е достъпно от адрес: http://localhost:8080/DocumentSigningDemoWebApp/ (фигура 3-8):*

Картина, която съдържа текст

Описанието е генерирано автоматично

*Фигура 3-8. Уеб приложение за подписване на документи в уеб среда и проверка на подписи*

# Глава 4. Ръководство за потребителя (смени заглавието)

*Съдържа* ***ръководство на потребителя: как се инсталира, конфигурира и изпълнява системата****. Трябва да се опише програмния продукт от изискванията към хардуера, през инсталацията, до начално конфигуриране и описание на ключовите екрани и как се работи с тях (когато това не е очевидно), заедно с подходящи screenshots (максимум* ***4-6 страници****).*

*Пример:*

*Системата е достъпна като сорс код от следното GitHub хранилище:* [*https://github.com/nakov/NakovDocumentSigner*](https://github.com/nakov/NakovDocumentSigner)

*…*

*За да работи правилно аплетът, е необходимо той да бъде подписан. Можем да използваме следния скрипт за да си генерираме саморъчно-подписан сертификат, с който да подпишем след това аплета:*

|  |
| --- |
| **generate-certificate.bat**  **del SmartCardSignerApplet.jks**  **keytool -genkey -alias signFiles -keystore SmartCardSignerApplet.jks -keypass !secret -dname "CN=Your Company" -storepass !secret**  **pause** |

*…*

# Заключение

*Съдържа* ***обобщение на постиженията*** *в дипломния проект и дипломната работа и* ***възможностите за усъвършенстването*** *и обогатяването им (максимум* ***1-2 страници****).*

*Пример:*

*Системата* [*NakovDocumentSigner*](http://www.nakov.com/documents-signing/) *е работещ пример, който илюстрира един платформено независим подход за използване на цифрови подписи от Java-базирани уеб приложения. Тя решава проблемите, които …*

*Ще разгледаме най-важните проблеми и недостатъци на системата* [*NakovDocumentSigner*](http://www.nakov.com/documents-signing/)*: (изброяваме ги)…*

# Информационни източници

*Включва* ***цитираната*** *и* ***използвана*** *в записката на дипломния проект* ***литература****. Започва на отделна страница от основния текст.*

*Примери:*

*1. Николов А., Програмиране на С++, Техника, София, 1998.*

*2. John A., Main Principles of C++ Programming, International Journal of Programming, Vol. 35, No 5, May 2001, pp. 112-183.*

# Приложение 1 / 2 / …

*Съдържат* ***документация****, която не е намерила място в текста поради ограниченията в обема й или за по-добра прегледност подредба. В текста трябва да има* ***препратка*** *към всички приложения.*

*Пример:*

*В* ***" Приложение 1: Тестване, оценка и усъвършенстване"*** *е направена критична оценка на реализираната система…*

*…*

***Приложение 1: Тестване, оценка и усъвършенстване***

*Една от основните идеи, залегнали при проектирането и имплементирането на системата за цифрово подписване на документи в уеб среда, е тя да бъде платформено независима и да…*

***Експериментално тестване на системата***

*По време на тестовете на системата NakovDocumentSigner използвахме следния софтуер oт страна на сървъра:*

* *Windows XP SP2*
* *Java Runtime Environment версия 1.4.2*

# Рецензия на дипломен проект

|  |  |  |  |
| --- | --- | --- | --- |
| **Тема на дипломния проект** |  | | |
| **Ученик** |  | | |
| **Клас** |  | | |
| **Професия** |  | | |
| **Специалност** |  | | |
| **Ръководител- консултант** |  | | |
| **Рецензент** |  | | |
| **Критерии за допускане до защита на дипломен проект** | | **Да** | **Не** |
| Съответствие на съдържанието и точките от заданието | |  |  |
| Съответствие между тема и съдържание | |  |  |
| Спазване на препоръчителния обем на дипломния проект | |  |  |
| Спазване на изискванията за оформление на дипломния проект | |  |  |
| Готовност за защита на дипломния проект | |  |  |
| Силни страни на дипломния проект |  | | |
| Допуснати основни слабости |  | | |
| Въпроси и препоръки към дипломния проект |  | | |

**ЗАКЛЮЧЕНИЕ:**

Качествата на дипломния проект дават основание ученикът/ ученичката ................................................... да бъде допуснат/а до защита пред членовете на комисията за подготовка, провеждане и оценяване на изпит чрез защита на дипломен проект- част по теория на професията.

.........05.2023г. Рецензент:................................

град София