



УНИВЕРЗИТЕТ СИНГИДУНУМ
ФАКУЛТЕТ ЗА ИНФОРМАТИКУ И РАЧУНАРСТВО

АПЛИКАЦИЈА ЗА АГЕНЦИЈУ ЗА ИЗДАВАЊЕ ПУТНИЧКИХ ВОЗИЛА

- дипломски рад -

Ментор:

Доц. др Александар Јевремовић

Кандидат:

Јелена Ћелић

Београд, 2016.



ФАКУЛТЕТ ЗА ИНФОРМАТИКУ И РАЧУНАРСТВО

УНИВЕРЗИТЕТ СИНГИДУНУМ

ФАКУЛТЕТ ЗА ИНФОРМАТИКУ И РАЧУНАРСТВО

Београд, Данијелова 32

Кандидат: Јелена Ћелић

Број индекса: 2012/200822

Тема: Апликација за агенцију за издавање путничких возила

Задатак: Апликација треба да омогући административном особљу агенције да води евиденцију о изнајмљивању путничких возила, као и о возном парку и подацима везаним за трошкове одржавања возила. Апликација мора да врши контролу приступа и да дозволи само корисницима који имају исправне параметре за приступ да се пријаве на апликацију. Од опција које морају да буду имплементирани су могућност додавања новог возила у возни парк, са свим основним подацима о возилу. Одређени подаци треба да буду увезени са записом о возилу из других табела у бази података, а не да се уносе као слободан текст. За поља која су слободан текст, обавезно вршити проверу исправности садржаја који се уноси помоћу регуларних израза. Апликација треба да приказује која возила из возног парка су тренутно на располагању, а која су издата на реверс у датом тренутку.

Датум одобрења рада:

Београд, ____ . ____ . ____.

МЕНТОР:

ДЕКАН:

Апликација за агенцију за издавање путничких возила

Сажетак: У овом дипломском раду представићу апликацију за агенцију за издавање путничких возила једноставног, прегледног и функционалног дизајна која је намењена и прилагођена свим корисницима Интернет услуга.

Садржај

1	Увод	3 -
2	HTML	5 -
2.1	Ознаке	6 -
2.2	Хипер-везе	7 -
2.3	Уређивање текста	8 -
2.3.1	Нумерисане и ненумерисане листе	10 -
2.3.2	Табеле	12 -
2.4	Формулари.....	13 -
2.4.1	Поља за унос текста.....	13 -
2.4.2	Радио-дугмад.....	14 -
2.4.3	Падајуће листе	14 -
2.4.4	Поља за означавање.....	15 -
2.4.5	Дугмад	15 -
3	CSS.....	17 -
3.1	Селектори	17 -
3.2	Имплементација.....	19 -
4	JavaScript.....	20 -
4.1	Имплементација.....	20 -
4.2	Променљиве	21 -
4.2.1	Низови	22 -
4.3	Оператори.....	22 -
4.4	jQuery	23 -
5	Bootstrap	24 -
5.1	Структура	24 -
5.2	Имплементација.....	25 -
5.3	Грид систем.....	25 -

6	RНР	- 27 -
6.1	Синтакса	- 27 -
6.2	Управљачке структуре	- 29 -
6.2.1	Условно гранање if/else	- 29 -
6.2.2	Вишесмерно гранање switch/case	- 29 -
6.2.3	Цикличне структуре for/while	- 30 -
6.3	Обрада података из формулара	- 31 -
6.4	Рад са фајловима	- 31 -
6.5	Рад са базама података	- 32 -
6.5.1	Додавање података у базу	- 33 -
6.5.2	Учитавање података из базе	- 33 -
6.5.3	Измена података у бази	- 34 -
6.5.4	Уклањање података из базе	- 34 -
7	MVC	- 35 -
8	Апликација	- 37 -
8.1	Развој апликације	- 37 -
8.1.1	Преглед мапе апликације	- 37 -
8.1.2	Преглед функција корисника	- 37 -
8.1.3	Преглед модела почетне стране	- 38 -
8.1.4	Преглед модела базе података	- 40 -
8.2	Функције апликације	- 42 -
8.2.1	Пријава	- 42 -
8.2.2	Додавање путничког возила	- 44 -
8.2.3	Измена путничког возила	- 46 -
8.2.4	Изнајмљивање возила	- 49 -
8.2.5	Слање контакт форме	- 53 -
9	Закључак	- 55 -
10	Литература	- 56 -

1 Увод

У прошлости услуга изнајмљивања путничких возила се искључиво односила на иностране госте којима је због природе свог пословања био неопходан повољан, безбедан и практичан превоз. Данас, изнајмљивање путничких возила је у сталном успону због различитих потреба људи, било да је у питању пословни састанак или годишњи одмор, прослава или венчање, било да је квар на возилу или чињеница да у данашњем несигурном времену не располажу сопственим превозом, људи се окрећу изнајмљивању путничких возила.

Идеја овог дипломског рада је креирање управо такве веб апликације за онлајн (енгл. *online*) изнајмљивање путничких возила ради омогућавања квалитетног и сигурног превоза за све.

Апликација за агенцију за издавање путничких возила првенствено поседује прегледност и привлачност за корисника, која се огледа у једноставном и функционалном дизајну веб апликације, пружању прегледа каталога са широким асортиманом путничких возила и омогућавању да услуга изнајмљивања истих буде поједностављена и ефикасна. Како изнајмљивање путничких возила захтева остављање личних података, неопходно је било обезбедити заштиту истих, ради проверавања баланса новчаних средстава и извршавања регуларне испоруке путничких возила. Треба напоменути да је ова веб апликација прилагођена свим нивоима интернет корисника, јер није усмерена уско једној групи, већ широј групи корисника.

Главни циљ веб апликације за изнајмљивање путничких возила је пружање услуге изнајмљивања путничких возила корисницима којима је неопходно да пређу одређену километражу у одређеном временском периоду, уз могућност одабира путничког возила из свих класа и категорија ради пружања најбољих и најквалитетнијих услуга. Такође, пружање корисницима једноставну испоруку путничког возила на најближем месту и у најкраћем року.

Један од додатних циљева веб апликације је надомештање недостатака које имају друге апликације за изнајмљивање путничких возила, нуђењем погодностима као што су:

- Једноставна манипулација категоријама
- Управљањем корисничким налозима
- Дозвољавање изнајмљивање путничких возила само пунолетним лицима са валидном возачком дозволом ради заштите интегритета и ауторизације
- Омогућавање само администратору, особи са привилегијама, да врши додавање, измену, брисање и изнајмљивање возила.
- Добро организован и естетски привлачан кориснички интерфејс (енгл. *interface*)
- Могућност покретања веб апликације на различитим уређајима и веб претраживачима

За пројектовање и имплементирање веб апликације за издавање путничких возила користила сам различите интернет и веб технологије. У овом раду представићу језике *HTML*, *CSS*, *JavaScript* и *PHP* без којих ова апликација не би поседовала динамичан и функционалан кориснички интерфејс, систем *MySQL* који је омогућио управљање са релационим базама података, алат *Bootstrap* који је допринео и олакшао развој веб форми и напредних веб компоненти и *MVC* архитектура која је поделила поједине делове апликације у компоненте које су коришћене за израду веб апликације за издавање путничких возила.

AutoRent Početna Cenovnik Uslovi Iznajmljivanje Kontakt Prijava



Najam

Lokacija ---

Kategorija ---

☐ 4+1 putnika

☐ 5 brzina

☐ 5 vrata

☐ Dečije sedište

☐ GPS

☐ Klima

☐ Prva pomoć

☐ Pun rezervoar

☐ Zimske gume

Pretraga



Renault Captur

Renault Captur

50 €

Naše usluge

Pružamo
00h-24h

Nudimo
Putno osiguranje

Dajemo
Gartis GPS navigacija

© 2016 - AutoRent

2 HTML

Језик за означавање хипертекста (енгл. *HyperText Markup Language*, *HTML*), грешком често сврстан у групу програмских језика, је прилично лаган и прост језик означавања, који има велику и значајну примену у свету због неизоставне потребе за креирањем обогаћених веб страница услед непрестаног развијања интернета и веба, који су постали део нашег свакодневног живота и пословања.

Овај стандард омогућио је дефинисање изгледа веб страница, успоставу веза (линкова) међу документима и креирање и мењање обичних текстуланих фајлова у било ком програму за уређивање текста.

Два неопходна елемента које *HTML* документ треба да поседује су декларација његовог типа и сам *html* елемент:

```
<!DOCTYPE html >
```

Декларација типа документа служи да веб претраживачу клијента пружи информацију о којој верзији *HTML-a* је реч, односно на који начин треба интерпретирати садржај. Тренутно актуелна верзија *HTML* стандарда је 5 која је у поређењу са преходним верзијама веома поједностављена.

Сам *html* елемент унутар *HTML* документа представља његов садржај, а састоји се из две основне ознаке *head* и *body* које представљају заглавље и тело документа:

```
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

У заглављу документа чувају се информације о самом документу (наслов, кључне речи, опис, везе ка стиливима и скриптовима и сл.) док се у телу налази садржај који се приказује у веб претраживачу.

2.1 Ознаке

Предефинисане одговарајућим *HTML* стандардом, ознаке (енгл. *tag*) служе за дефинисање одговарајућих елемената у којима се садржај налази у *HTML* документима. Све ознаке се пишу у оквиру изломљених заграда “< >” и обично се налазе у пару као отварајуће и затварајуће ознаке, односно свака ознака мора бити затворена истом ознаком са додатком косе црте “/” унутар заграда:

```
<html>
...
</html>
```

Постоје случајеви када ознаке представљају крајње чворове и када се ово правило изузима:

```
<br >
```

Предност ознака *HTML* стандарда се огледа у могућности да поједини елементи у себи могу имати друге елементе или текстуални садржај:

```
<table>
  <tr>
    <td> Поље 1 </td>
    <td> ... </td>
  </tr>
  <tr>
    <td> Поље 1.1 </td>
    <td> ... </td>
  </tr>
</table>
```

Такође, у томе што се различитим атрибутима могу додељивати елементи унутар њихових ознака. Атрибути се увек пишу у првом делу, након што је ознака отворена и састављени су из два дела *назив*=”*вредност*” :

```
<img src= “slika.jpg” alt= ”опис слике” />
```

Још једна од могућности *HTML* стандарда која је више значајна програмерима, него самом програму је унос текстуалних ознака (коментара), које се најчешће користе као подсетници или као објашњење кода. Главна карактеристика коментара је да се неће приказивати у веб претраживачу:

```
<!--Наслов -->

<h1> Ауто изнајмљивање</h1>

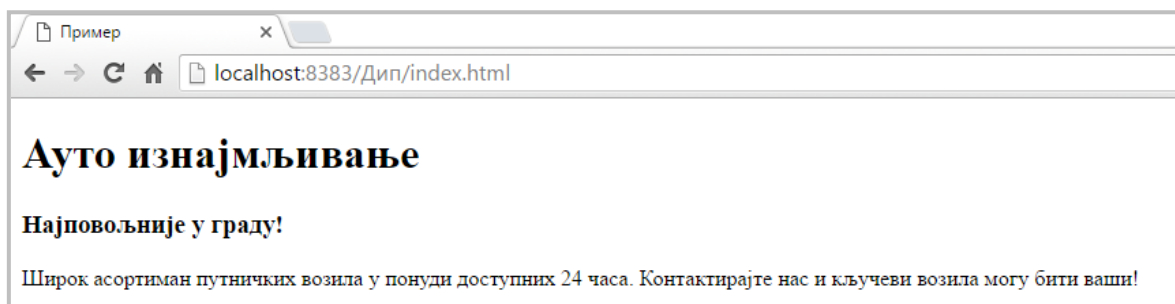
<!-- Поднаслов -->

<h3> Најповољније у граду!</h3>

<!-- Параграф -->

<p>Широк асортим путничких возила у понуди доступних 24ч.</p>

<p>Контактирајте нас и кључеви возила могу бити ваши!</p>
```



Слика 1: Резултат обраде ознака у веб претраживачу

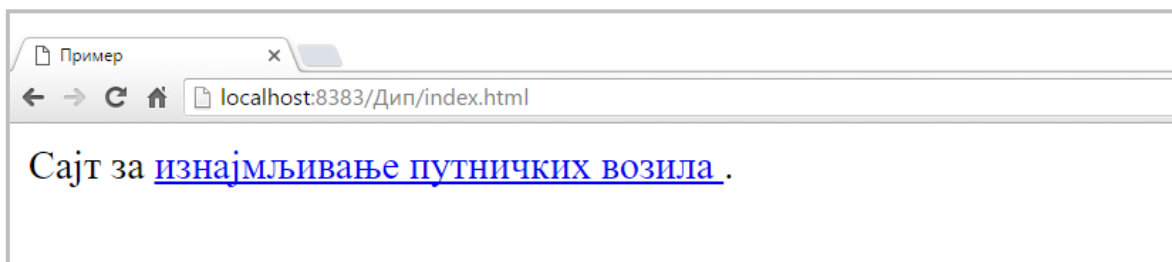
2.2 Хипер-везе

Главна карика за проналажење жељених информација су хипер-везе јер омогућују повезивање *HTML* документа на вебу. Постоје два типа повезивања.

Први тип је повезивање са саставним ресурсима самог документа где се првенствено користе инструкције у заглављу документа.

Други тип подразумева повезивање докумената путем хипер-веза коришћењем тела документа. Овакве везе пружају да се неком елементу у телу документа придружи веза ка неком другом документу. Дефинишу се уз помоћ ознаке *a* и *href* које представљају одредиште везе:

```
Сајт за <a href= "http://www.autorent.rs" > изнајмљивање путничких возила </a> .
```



Слика 1: Повезивање докумената путем хипер-веза

Такође, хипер-везе осим ка другим документима, могу да повезују и делове документа који се налазе у самом том истом документу. У том случају се у оквиру одредишта хипер-везе новоди идентификатор одређеног дела самостално или у комбинацији са адресом удаљеног документа.

2.3 Уређивање текста

Креирање докумената са текстуалним садржајем који се може аутоматски повезивати је главна намена постојања *HTML* језика. Због овога *HTML* нуди широк дијапазон могућности за формирање текстуалног садржаја.

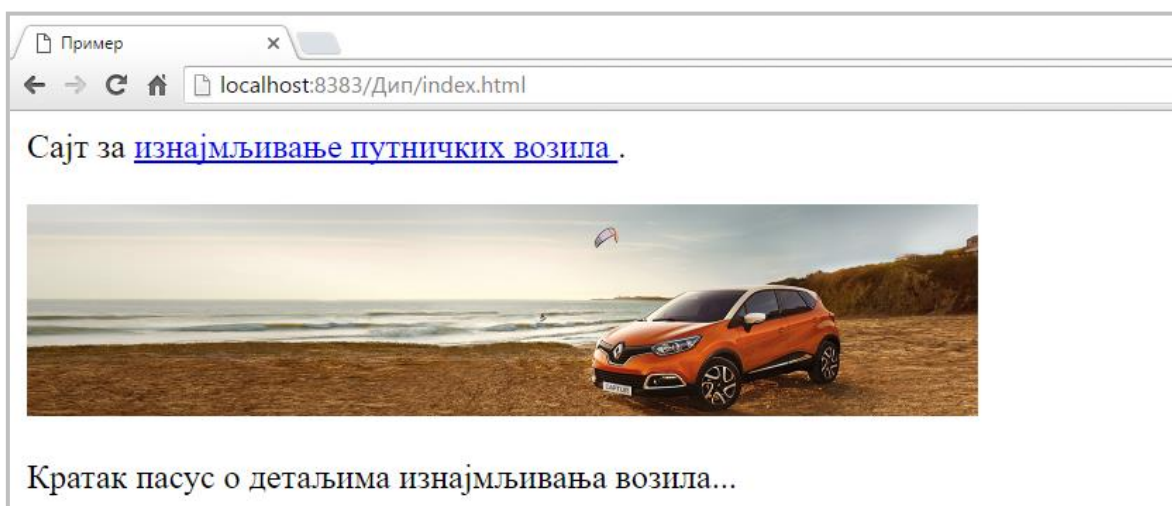
За правилно приказивање текста у *HTML* документима, како би веб претраживач имао информацију на који је начин текст кодован, у заглавље документа додају се ознаке *meta* са вредношћу *UTF-8* за атрибут *charset*:

```
<html>
  <head>
    ...
    <meta charset= "UTF-8">
    ...
  </head>
</html>
```

За организацију текста у *HTML* документима као основна јединица се користи параграф чија је ознака *p* и унутар кога се може додати једна или више реченица, слике, линкови и друго:

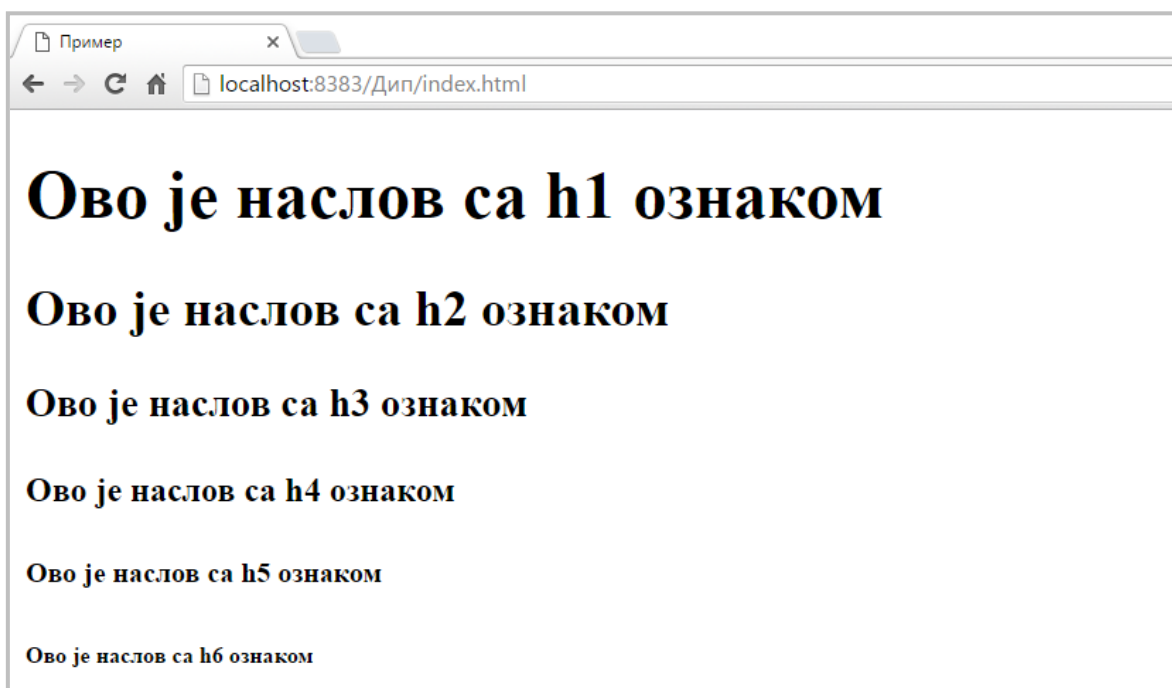
```
<p> Сајт за
<a href= "http://www.autorent.rs"> изнајмљивање путничких возила </a>.
</p>

<p>Кратак пасус о детаљима изнајмљивања возила...</p>
```



Слика 1: Резултат интерпретирања наведеног кода

За избор величине слова која ће се користити у наслову и поднаслову документа, *HTML* језик поседује 6 нивоа заглавља чије су ознаке дефинисане од *h1* до *h6*:

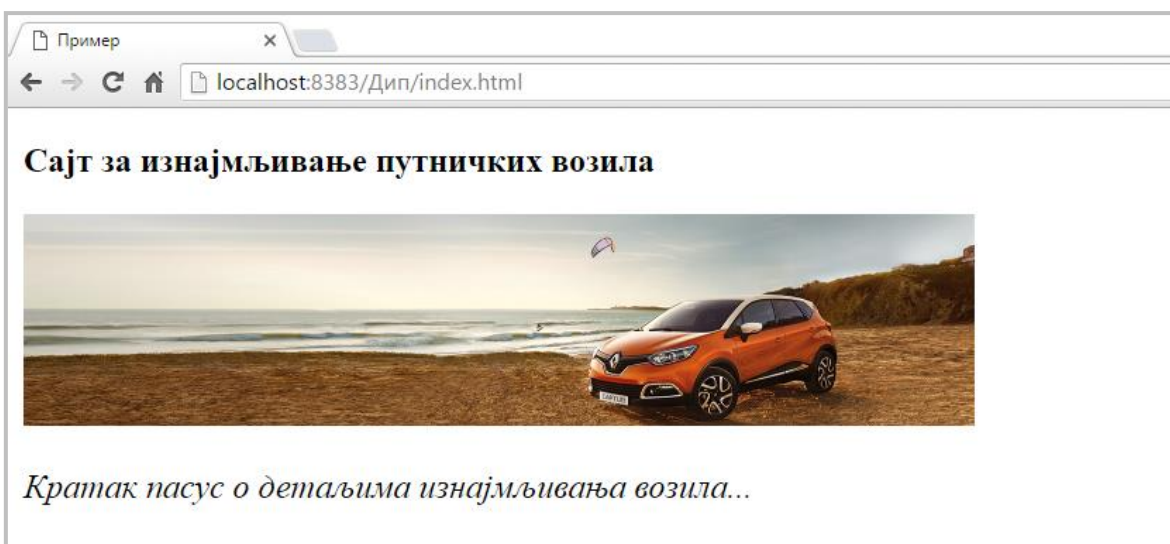


Слика 2: Пример свих нивоа наслова

За специјално наглашавање одређеног дела текста у *HTML* документима користе се логичке ознаке *em* (блаже наглашавање курзивом) и *strong* (посебно наглашавање подебљаним словима). Такође, постоје и физичке ознаке за наглашавање текста *b* и *i*. Разликују се од логичких ознака јер не зависе од веб претраживача:

```
<p> <strong> Сајт за </strong>
<a href= "http://www.autorent.rs" > изнајмљивање путничких возила </a>.
</p>

<p> <em> Кратак пасус о детаљима изнајмљивања возила...</em> </p>
```



Слика 3: Блаже и посебно наглашавање текста

Остале ознаке за уређивање текста у HTML документима су:

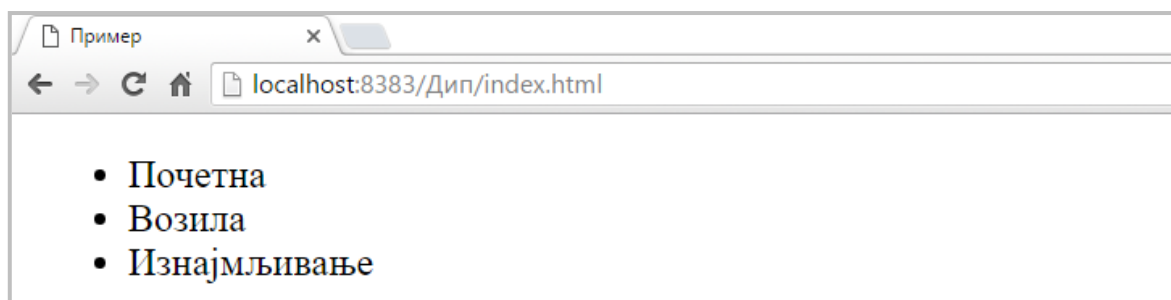
- **<sub>** дефинише потписан текст
- **<sup>** дефинише натписан текст
- **<u>** дефинише подвучен текст
- **
** прави ред размака у текстуалном облику
- **<hr>** креира хоризонталну линију
- **<div>** дели страницу на различите секције/блокове

2.3.1 Нумерисане и нумерисане листе

Још једна битна одлика *HTML* језика је могућност креирања листа са набрајањем или без набрајања. Листе се могу комбиновати у било ком броју нивоа кроз додавање нове листе унутар ставке постојеће листе.

За креирање нумерисане листе (енгл. *Unordered List*) користи се ознака **** у оквиру које се ставке додају ознаком ****:

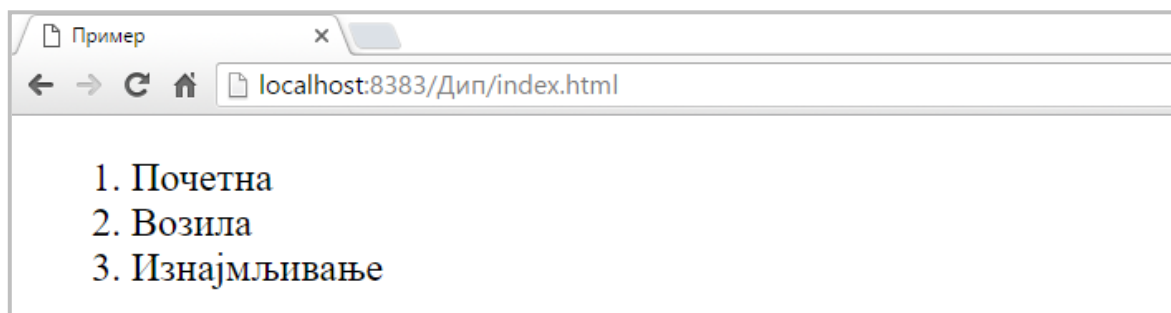
```
<ul>
  <li>Почетна</li>
  <li>Возила</li>
  <li>Изнајмљивање</li>
</ul>
```



Слика 1: Приказ ненумерисане листе

За креирање нумерисане листе (енгл. *Ordered List*) користи се ознака `<ol>` у оквиру које се ставке додају ознаком `<li>` и у којима интерпретер сам води рачуна о нумерацији ставки:

```
<ol>
  <li>Почетна</li>
  <li>Возила</li>
  <li>Изнајмљивање</li>
</ol>
```

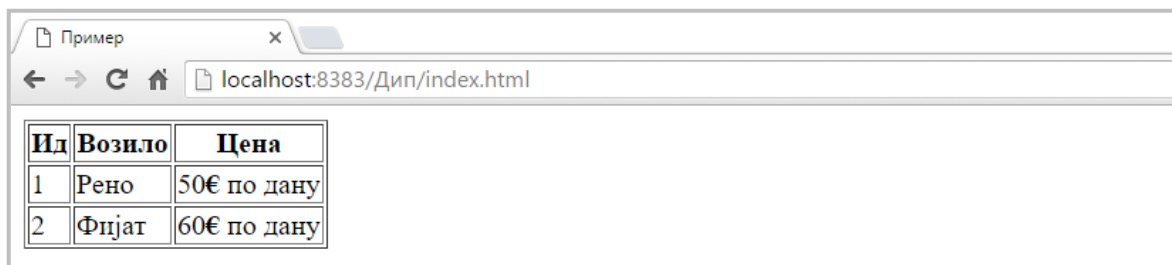


Слика 2: Приказ нумерисане листе

2.3.2 Табеле

Посебна пажња се обраћа на табеле јер због њихове намене представљају један од најчешће коришћених садржаја у *HTML* документима. Креирају се помоћу ознаке `<table>`, а раздвајање у редове и колоне врши се помоћу ознака `<tr>` и `<td>`. Исто као и у случају параграфа текста, и табеле могу садржати текст, слику, листу па чак и нову табелу:

```
<table border="1">
  <tr>
    <th>Ид</th>
    <th>Возило</th>
    <th>Цена</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Рено</td>
    <td>50<b>&euro; </b> по дану</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Фијат</td>
    <td>60<b>&euro; </b> по дану</td>
  </tr>
</table>
```



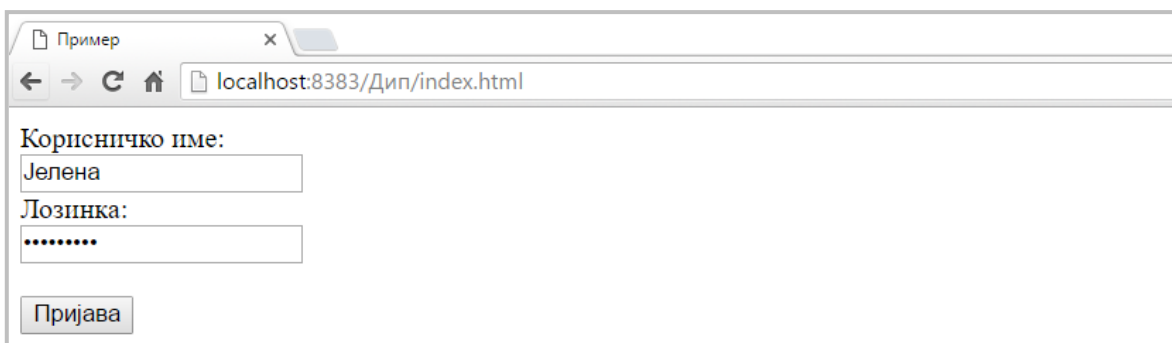
Ид	Возило	Цена
1	Рено	50€ по дану
2	Фијат	60€ по дану

Слика 1: Приказ табеле

2.4 Формулари

Данас су неизоставни део сваке веб апликације јер омогућују преузимање података од корисника и ограничавају унос вредности. Формулари се креирају ознаком `<form>`, а појединачни елементи формулара ознаком `<input>`:

```
<form>
  Корисничко име:<br>
  <input type="text" name="име"><br>
  Лозинка:<br>
  <input type="password" name="лозинка"><br>
  <button type="submit"> Пријава</button>
</form>
```



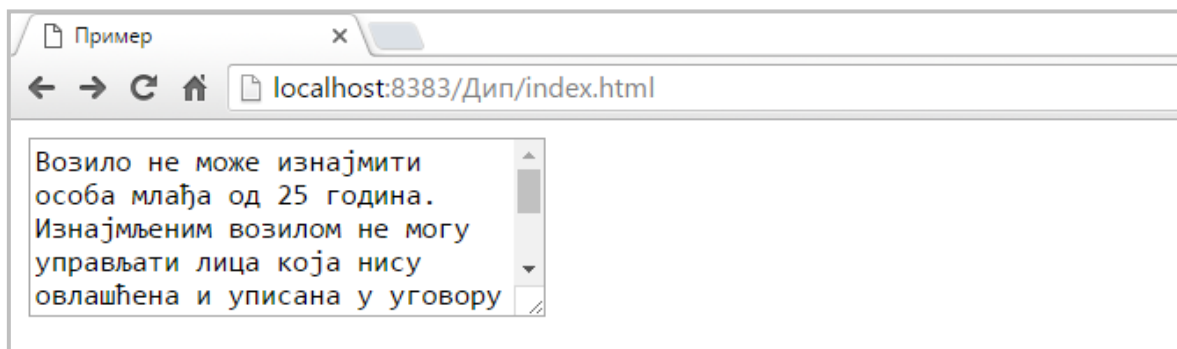
Слика 1: Приказ базичног формулара

2.4.1 Поља за унос текста

Примењују се три основна поља за унос текста од стране корисника: унос текста у једном реду, унос текст у више редова и унос скривеног текста.

За унос текста у једном реду и скривеног текста користи се ознаке из претходног примера, док се за унос текста у више редова користи ознака `<textarea>` и атрибути `name`, `cols`, и `rows`:

```
<textarea name="опис" cols="30" rows="5"> . . .</textarea>
```

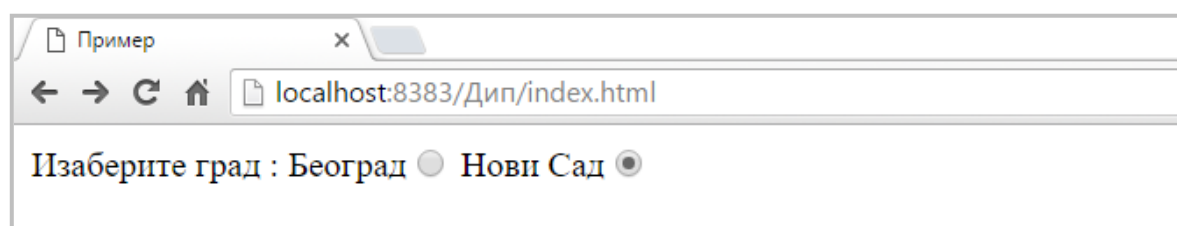


Слика 1: Приказ формулара за унос текста у више редова

2.4.2 Радио-дугмад

Користе се када је потребно одабрати једну од више понуђених опција. Додавањем ознаке `<input>` и атрибута `type` са вредношћу `radio` добија се адекватан формулар:

Изаберите град : Београд `<input type="radio" name="град" value="б">`
Нови Сад `<input type="radio" name="град" value="нс">`

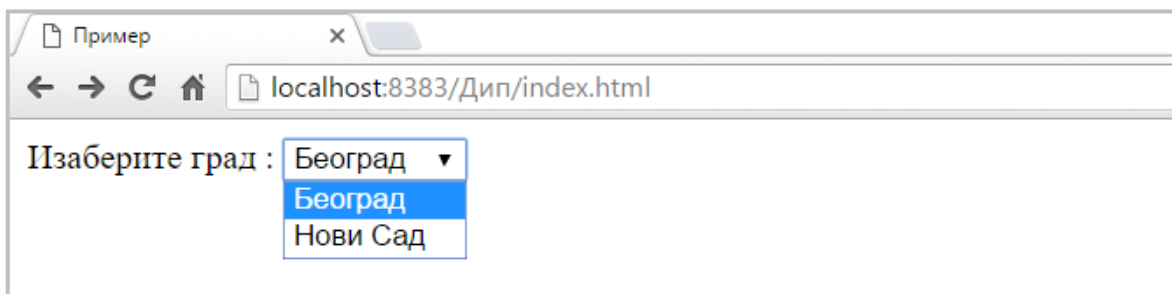


Слика 1: Приказ радио-дугмад

2.4.3 Падајуће листе

Креирају се као алтернатива радио-дугмади јер понекад ради естетике и функционалности веб апликације, радио-дугмад нису најбоље решење. Додавањем ознаке `<select>` са атрибутом `name` и ознаке `<option>` са одговарајућом вредношћу добија се жељени формулар:

Изаберите град :
`<select name="град">`
 `<option value="б">Београд</option>`
 `<option value="нс">Нови Сад</option>`
`</select>`



Слика 1: Приказ падајуће листе

2.4.4 Поља за означавање

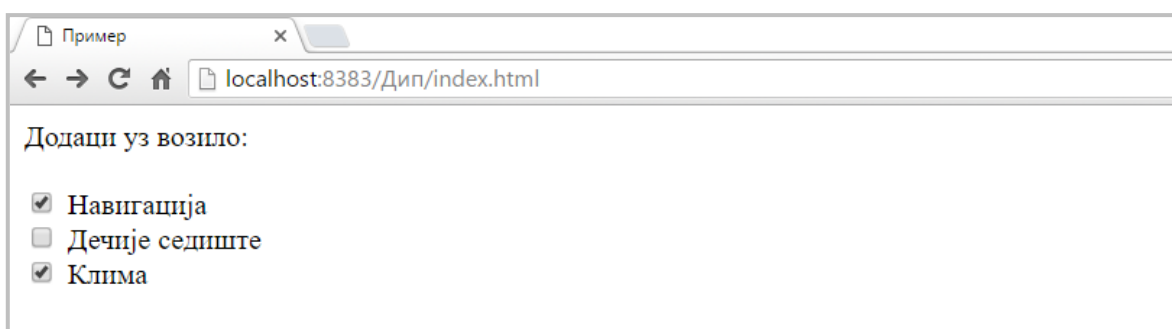
Употребљавају се када је потребно одабрати више понуђених вредности. Додавањем ознаке `<input>` код кога је тип постављен на `checkbox` креира се формулар:

Додаци уз возило: `<br>`

`<input type="checkbox" name="навигација">` Навигација`<br>`

`<input type="checkbox" name="дечије_седиште">` Дечије седиште`<br>`

`<input type="checkbox" name="клима">` Клима



Слика 1: Приказ падајуће листе

2.4.5 Дугмад

Омогућују слање података ка серверу која је најчешћа употреба формулара. Међутим, дугмад се могу користити за покретање одређене акције или за ресетовање унетог садржаја. Додавањем ознаке `<button>` и одговарајућег типа креира се дугме, а атрибутом `method` дефинише се метод слања података који може бити `GET` или `POST`. У датом примеру приказаћу дугме за слање података са типом `submit`, ознаком `<action>` (дефинише који ће програм са серверске стране преузети послате податке) и методом `post`:

```
<form name="input" action="http://www.autorent.rs/iznajmljivanje.php"
method="post">
    Изнајмљивање возила:<br>
    Име: <br><input type="text" name="име" >
    Презиме: <br><input type="text" name="презиме" >
    Телефон: <br><input type="tel" name="телефон" >
    Возило:<br><input type="text" name="возило" >
    <button type="submit" value="изнајми"> Изнајми </button>
</form>
```

Пример

localhost:8383/Дип/index.html

Изнајмљивање возила:

Име:

Презиме:

Телефон:

Возило:

Изнајми

Слика 1: Приказ слања садржаја формулара

3 CSS

Језик за дефинисање стилова који одређује и формира изглед *HTML* елемената. *CSS* (*CascadingStyleSheets*) поседује три основне карактеристике за дефинисање класа за изглед, боје и фонтова чији елементи омогућавају приступачнији и флексибилнији садржај. Коришћењем стилског језика избегава се константно понављање ознака и због одређеног броја енглеских речи, синтакса је крајње једноставна. Састоји се из декларације који садржи атрибуте и вредности које се групишу унутар витичастих заграда “{ }” и завршавају се ознаком “;” :

```
селектор{  
    атрибут1: вредност1;  
    атрибут2: вредност2;  
    ...  
}
```

Једноставност *CSS* синтаксе се огледа и у њеном писању које може бити једнолинијски или вишелинијски. У примеру се може видети да се коментари у *CSS* језику пишу ознакама “/” и “*”, који се доста разликују од коментара у *HTML*-у:

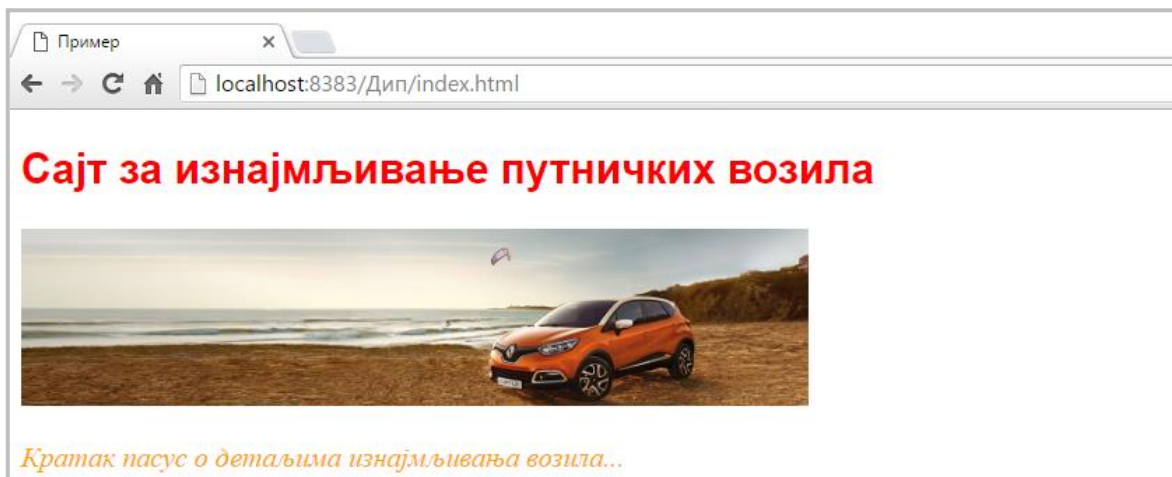
```
/* једнолинијска синтакса */  
селектор { атрибут1: вредност1; атрибут2: вредност2; ... }  
  
/* вишелинијска синтакса */  
селектор{  
    атрибут1: вредност1;  
    атрибут2: вредност2;  
    ...  
}
```

3.1 Селектори

Постоје три основне врсте селектора: *HTML* селектори, *ID* селектори и селектори класе.

HTML селектори представљају називе ознака који се користе у *HTML* документу на које ће бити примењени:

```
/* CSS */
h2{
    color: orangered;
    font-family: sans-serif;
}
p {
    color: orange;
    font-style: italic;
}
<!-- HTML -->
<h2> Сајт за изнајмљивање путничких возила </h2>
<p> Кратак пасус о детаљима изнајмљивања возила...</p>
```



Слика 1: Резултат интерпретирања наведеног кода

ID селектори чији називи почињу тарабом “#” и чија намена је примењивање одређеног стила на јединствен елемент у *HTML* документу:

```
/* CSS */
#naslov {
    color: red;
}
<!-- HTML -->
<h2 id="naslov">Сајт за изнајмљивање путничких возила </h2>
```

Селектори класе (енгл. *class*) чији називи ових стилова почињу тачком “.” и који се користе за креирање стилова који се позивају у *class* атрибутима *HTML* елемената:

```
/* CSS */  
.naslov {  
    color: red;  
}  
  
<!-- HTML -->  
<h2 class="naslov">Сајт за изнајмљивање путничких возила </h2>
```

3.2 Имплементација

CSS стилове је могуће користити унутар *HTML* документа на три начина.

Интерни стил у коме се дефинише стил унутар сваког *HTML* документа посебно, навођењем исказа унутар *style* атрибута који је потребно променити. Најједоставнији али најмање препоручљив начин за употребу CSS јер је код нечитљив и тежак за проналажење грешака и измене:

```
<head>  
    <style>  
        a {color: #fff;}  
    </style>  
</head>
```

Уметнути стил који је дефинисан у заглављу *HTML* документа и који је бољи од претходног јер одваја приказ од садржине. Међутим, може представљати проблем при евентуалној измени, зато је препоручљиво његово коришћење у случајевима када је потребно приметити стил само на поједином *HTML* елементу.

```
<a style="color:#fff"> Линк </a>
```

Екстерни стил издваја стилове у засебан CSS фајл где се садржина и приказ одвајају у потпуности и где се стилови могу користити на више различитих *HTML* докумената мењањем само једног фајла. Због овога је данас препоручљив стил:

```
<head>  
    <link href="/css/main.css" rel="stylesheet" type="text/css" />  
</head>
```

4 JavaScript

Објектно оријентисан скриптни језик намењен динамичкој изradi веб апликација јер диктира како ће се елементи странице понашати. Укључивањем овог извршног језика пружа се већа контрола и интеракција са корисником. Комбиновањем са *HTML*-ом и *CSS*-ом, *JavaScript* чини неизбежну компоненту без које се ниједна модерна апликација не може замислити. Један од основних разлога настанка овог језика који се извршава на страни корисника, јесте проверавање веродостојности података, међутим данас *JavaScript* пружа много више.

4.1 Имплементација

Као и *CSS* језик, *JavaScript* се у документ може убацити на више начина, тачније два. Писањем директно у *HTML* код у *<head>* секцији у којој се скрипта одмах учитава, или смештањем унутар *<body>* секције *HTML* документа где се скрипта извршава чим веб претраживач наиђе на њу:

```
<html>  
  <body>  
    <script type="text/javascript">  
      document.write("JavaScript!");  
    </script>  
  </body>  
</html>
```

Други начин је смештање скрипте у посебан фајл (екстензија ".js") који се потом позива у *HTML* документу. Овакав начин пружа да се једна иста скрипта користи у више страница:

```
<script type="text/javascript" src="/js/main.js"></script>
```

Такође, унос коментара који се користе ради лакшег сналажења у коду је у *JavaScript* –у омогућен на више начина:

```
// Коментар у једној линији кода
```

```
/* Коментар који садржи  
више линија кода */
```

4.2 Променљиве

Како би се запамтиле вредности које се касније користе у другим деловима програма, *JavaScript* као и остали програмски језици користи променљиве.

Променљиве нису ништа друго до имена која садрже вредности. Та имена могу почињати великим словима (А - З), малим словима (а - з), или доњом цртом “_”. Остали карактери могу се састојати, поред наведених и од цифара (0 - 9) :

```
Возило  
_возило  
Возило_1  
изнајмљивањеВозила  
ВОЗИЛО
```

JavaScript, и ако обраћа пажњу на велика и мала слова (енгл. *case-sensitive*), не захтева специфицирање типа вредности променљиве, већ дозвољава због поседовања интерпретатора који аутоматски конвертује тип података који се налази у променљивој. За декларисање променљивих користи се ознака *var*:

```
var x;  
  
var y = 28;  
  
var изнајмљивање = true;  
  
var возило = "Пежо 206";
```

Разликују се две врсте променљивих:

Локалне променљиве које се дефинишу унутар неке функције и доступне су само унутар исте:

```
function test() {  
    var a = 10;  
}
```

Глобалне променљиве које се дефинишу изван функције и њима се може приступити из било које функције:

```
var a = 5;  
function test() {  
    var b = a + 10;  
}
```

4.2.1 Низови

Посебна врста променљивих који имају могућност смештања вишеструких вредности. Дакле, то су објекти који се састоје из назива низа и индекса који представља редни број низа и који се налази између угластих заграда “[]”:

```
низ [0]
низ [1]
низ [2]
низ [3]
...
```

Низови се могу писати на више начина, додељивањем вредности одмах приликом уноса или накнадно у току даљег кодовања:

```
// декларисање празног низа
var ауто = new Array();

// декларисање низа који ће имати 5 елемената
var ауто = new Array(5);

// декларисање низа чији индекс почиње са 10
var ауто = [10];

// декларисање низа са додељивањем вредности
var ауто = new Array("Рено", "Пежо", "Југо");
```

4.3 Оператори

Креирају се за трансформисање једне или више вредности у једну резултирајућу вредност. *JavaScript* ради са операторима који су подељени у следеће категорије:

Аритметички оператори представљају математичке операторе који се користе за разна рачуњања:

Оператор	Опис
+	сабирање
-	одузимање
*	множење
/	дељење
++	инкрементација
--	декрементација

Логички оператори користе се за извођење логичке операције над логичким операндима:

Оператор	Опис
&&	логичко и
	логичко или
!	логичка негација

Оператори поређења користе се за поређење две вредности према једнакости, неједнакости или нумеричкој вредности:

Оператор	Опис
==	једнако
===	строго једнако
!=	неједнако
!==	строго неједнако
<	мање
<=	мање или једнако
>	веће
>=	веће или једнако

Оператори доделе користе се за додељивање вредности променљивих:

Оператор	Пример	Исто као
=	<code>x = y</code>	
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>

4.4 jQuery

Када се говори о *JavaScript*-у неизоставно је поменути и његову најпопуларнију библиотеку *jQuery*. Креирана са наменом да буде надоградња *JavaScript*-а омогућила је његову једноставност и бољу интеракцију између *JavaScript*-а и других програмских језика. *jQuery* такође поједностављује и друге компликоване ствари у *JavaScript*-у као што су *AJAX* и *DOM* манипулација.

Предности употребе *jQuery* библиотеке су лакоћа употребе, понашање слично у свим веб претраживачима, велик број додатних библиотека, екстензија, корисника и програмера. Наравно и *jQuery* библиотека има недостатке јер ју је неопходно учитати приликом сваког првог учитавања веб странице.

5 Bootstrap

Представља софтвер отвореног кода (енгл. *open-source*), односно колекцију *HTML*-а, *CSS*-а и *JavaScript*-а, креиран са циљем да олакша и омогући бржу израду веб апликација. Пружа модерну типографију, форме, дугмади, табеле, навигацију и све потребне елементе за креирање веб апликације. Оно што ову софтверску платформу чини све популарнијом је њено потпуно подржавање израде прилагодљивих веб страница на различитим резолуцијама и димензијама екрана тј. прилагођавање изгледа у зависности од уређаја на којем се приказује. Управо што је потребно апликацији за агенцију за издавање путничких возила како би поседовала мобилност у којој корисници не би били ограничени.

5.1 Структура

Као што сам већ поменула, *Bootstrap* је комбинација *HTML*-а, *CSS*-а и *JavaScript*-а и они уједно чине његову структуру. Садржи три директоријума:

- **css фолдер** који поседује већ формиране класе које се примењују на елементе *HTML* документа
- **fonts фолдер** који садржи око 200 стандардизованих фонт-иконица (енгл. *glyphicons*) које се једноставно примењују тако што се под атрибутом *class* упише назив фонт-иконице.
- **js фолдер** који има *JavaScript* функције које омогућује разне акције, анимације и ефекте.

Структура bootstrap-a/

```

css/
├── bootstrap.css
├── bootstrap.min.css
├── bootstrap-theme.css
└── bootstrap-theme.min.css

js/
├── bootstrap.js
└── bootstrap.min.js

fonts/
├── glyphicons-halflings-regular.eot
├── glyphicons-halflings-regular.svg
├── glyphicons-halflings-regular.ttf
└── glyphicons-halflings-regular.woff
    
```

5.2 Имплементација

Први начин коришћења *Bootstrap*-а, који је и најчешћи, је преузимање софтвера и његово смештање у директоријум пројекта. Остаје само позивање жељеног елемента у *HTML* документ који се позива као и сваки други екстерни елемент:

```
<!-- укључивање CSS датотеке -->  
  
<link href="bootstrap.min.css" rel="stylesheet">  
  
<!-- укључивање JavaScript датотеке -->  
  
<script src="assets/js/bootstrap.min.js"></script>
```

Други начин подразумева укључивање датотека у *HTML* документ са сервера, без преузимања фајлова:

```
<!-- укључивање CSS датотеке -->  
  
<link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/  
/css/bootstrap.min.css">  
  
<!-- укључивање JavaScript датотеке -->  
  
<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/  
/js/bootstrap.min.js"></script>
```

5.3 Грид систем

Главна одлика *Bootstrap*-а јер олакшава прилагођавање веб апликације разлитим величинама екрана (монитори, лаптоп рачунари, таблет уређаји и мобилни телефони). У грид систему цела страница је подељена на дванаест колона (збир колона не сме бити већи од тога) па се на овај начин једноставно може одређивати ширина одређених секција на страници само позивањем одређене класе:

```
<div class="container">  
  <div class="row">  
    <div class="col-md-6 "> Лева половина странице</div>  
    <div class="col-md-6 "> Десна половина странице</div>  
  </div>  
</div>
```

Грид систем поседује пет нивоа прилагодљивости:

- Веома мали (**.col-xs-**) чија резолуција мања од 544px
- Мали (**.col-sm-**) чија резолуција већа или једнака од 544px
- Средњи (**.col-md-**) чија резолуција већа или једнака од 768px
- Велики (**.col-lg-**) чија резолуција већа или једнака од 992px
- Веома велики (**.col-xl-**) чија резолуција већа или једнака од 1200px

Предности оваквог система су:

- Олакшава и убрзава програмирање
- Омогућава једноставнији развој веб форми и интерфејса
- Индетично се приказује на свим веб читачима
- Једноставан за употребу уколико се познају основе *CSS* и *JavaScript*
- Софтвер отвореног кода

Недостаци грид система и Bootstrap-а уопште су:

- Не даје веб апликацији оригиналан изглед чиме визуелно веб апликације подсећају једна на другу
- Неопходно је да се сваки пут учита приликом учитавања странице чиме се страница учитава спорије
- Захтева одговарајућу верзију *jQuery* библиотеке тако да ажурирање и дотеривање веб апликације не може да се изведе без додатних дорада

6 PHP

Специјализовани скриптни језик *PHP* (енгл. Hypertext Preprocessor) користи се за креирање динамичког веб садржаја у коме клијент шаље серверу стандардан захтев за одређеним садржајем који се обрађује одговарајућом програмском процедуром на серверу. На овакав начин *PHP* код који је генерисао садржај се не може видети али зато нуди неограничене могућности развоја веб апликација у којима се приказује чист *HTML* код. По синтакси има сличности програмском језику *C* јер садржи исте функције, што подразумева да се једна акција може извести коришћењем више различитих функција.

6.1 Синтакса

Свака *PHP* датотека садржи ознаке “<?php” и “?” на почетку и крају између којих се налази *PHP* код:

```
<?php
    ...
?>
```

У *PHP* језику коментари се могу писати на више различитих начина као што је приказано у мом следећем примеру:

```
<?php

// Први начин писања једноредног коментара

# Други начин писања једноредног коментара

/* Начин писања
    вишередног коментара */
?>
```

Променљиве у *PHP* језику се представљају помоћу префикса “\$” иза које следи назив променљиве. У називу променљиве нема размака па је неопходно користити доњу црту “_” за одвајање речи. Имена су осетљива на мала и велика слова и могу се писати словом или доњом цртом иза којег може да следи неограничен број слова, бројева или доњих црта:

```
$возило
$ИзнајмљивањеВозила
$_возило
$Изнајмљивање_возила
```

У основне типове променљивих у *PHP* језику спадају:

- **Логичка вредност** (енгл. *boolean*)
- **Целобројна вредност** (енгл. *integer*)
- **Децимална вредност** (енгл. *float*)
- **Низ симбола** (енгл. *string*)

У сложене типове променљивих у *PHP* језику спадају:

- **Низови елемената** (енгл. *array*)
- **Објекти** (енгл. *object*)

У посебне типове променљивих у *PHP* језику спадају:

- **Ресурси** (енгл. *resource*)
- **Променљиве које немају вредност** (енгл. *NULL*)

Две основне функције за исписивање променљивих су *print* и *echo* :

```
// Функција прослеђује само један аргумент
print ( $arg)
// Функција прослеђује више аргумента
echo ( $arg1 [, $arg2 ... ] )
```

За разлику од променљивих, константама се не могу мењати вредности, не могу додељивати други типови података сем основних и не могу се писати префиксом "\$", док правило за имена константи важи као и код променљивих:

```
define ("назив_константе", вредност [ ... ] ) ;
```

PHP има доста уграђених функција које омогућавају извршавање одређеног блока наредби, које се понављају, али поседује могућност дефинисиња сопствених. Најчешћи разлози због којих се користе функције су избегавање поновног писања већ написаног дела кода и лакшег разумевања кода. Декларишу се помоћу кључне речи *function*, а називе функција се додељују према одрђеним правилима у којима се препоручује да функција асоцира на акције које она треба да реализује:

```
function име_функције (листа_параметара)
{
    блок наредби;
}
```

6.2 Управљачке структуре

Представљају груписање наредби којима се под одређеним условима одређује ток извршавања. Представљају и једну наредбу у облику блока која се поставља између витичастих заграда “{ }”.

6.2.1 Условно гранање if/else

У зависности од потребе, условно гранање као једна од основних управљачких структура, може имати различите форме. Подразумева дефинисање основног услова и њему придрженог скупа наредби који ће се извршити уколико је услов испуњен. Уколико услов није испуњен, наставља се испитивање следећих услова све до последње клаузуле *else* :

```
<?php
    if (условни_израз_1)
    {
        блок_наредби_1;
    }
    elseif (условни_израз_2)
    {
        блок_наредби_2;
    }
    else
    {
        последњи_блок_наредби;
    }
?>
```

6.2.2 Вишесмерно гранање switch/case

Употребљава се када је неопходно једну исту променљиву или израз упоредити са различитим вредностима, те у зависности од резултата поређења, извршити различите акције. Ова структура редом пореди различите случајеве (енгл. *case*) могућих очекиваних вредности, све док се не наиђе на случај када су те две вредности једнаке. Даље, извршавају се сви блокови наредби, по реду, све до последњег, или се прекида извршавање структуре (енгл. *break*):

```
<?php

switch (израз)
{
    case "вредност_1":
        блок_наредби_1;
    case "вредност_2":
        блок_наредби_2;

    default:
        последњи_блок_наредби;
}

?>
```

6.2.3 Цикличне структуре for/while

Користе се за понављање одређеног блока наредби већи број пута који може бити унапред познат или не. За понављање блока наредби унапред познати број пута користи се циклична структура *for*:

```
<?php

for (израз1; израз2; израз3)
{
    блок_наредби;
}

?>
```

У случајевима када није унапред познат број понављања блока наредби примењује се циклична структура *while*:

```
<?php

do
{
    блок_наредби;
}
while (условни_израз)

?>
```

6.3 Обрада података из формулара

Путем *HTTP* метода *GET* и *POST* могу се обрађивати вредности из *HTML* формулара. Приступ њима у *PHP* програмском језику се врши путем суперглобалних променљивих *\$_GET* и *\$_POST*. За слање података серверу користи се метод *POST*, а за њихов приступ променљива *\$_POST*:

```
<!-- HTML -->
<form action="slanje.php" method="post">
  <input type="text" name="име"/>
  <button type="submit"> Слање</button>
</form>

// PHP
<?php
  echo $_POST ['име'];
?>
```

За приступ подацима добијени путем *GET* метода врши се коришћењем суперглобалне променљиве *\$_GET*:

```
<?php
  echo $_GET ['име'];
?>
```

Такође, приступ подацима из *HTTP* захтева може се уједно и вршити путем суперглобалне променљиве *\$_REQUEST*:

```
<?php
  echo $_REQUEST ['име'];
?>
```

6.4 Рад са фајловима

PHP програмски језик има више функција за подршку различитим приступима рада са фајловима. Две основне функције које омогућују уписивање садржаја и његово ишчитавање из фајлова су *file_put_contents* и *file_get_contents*:

```
// Уписивање садржаја у фајл
$фајл = "Неки текстуални садржај";
file_put_contents('текст.txt', $фајл);

// Учитавање садржаја фајла
$фајл= file_get_contents('текст.txt');
echo $фајл ;
```

Међутим, код великих података, као и код података који константно пристижу потребно је направити везу са фајлом, а затим учитавати или исписивати у мање блоковне садржаје. Због тога се користе функција *fopen* за отварање фајла која поседује четири могућа параметра и функција *fwrite* за писање садржаја у фајл:

```
// Учитавање садржаја фајла
$фајл = fopen( $име_датотеке", $начин_приступања
[, $користити_путању_за_спољашње_датотеке = false [, $ресурс]] );

// Уписивање садржаја у фајл
fwrite ( $ фајл, $садржај [, $дужина_у_бајтовима] );
```

6.5 Рад са базама података

Једна од најважнијих и најбитнијих функција *PHP* програмског језика је рад са базама података. Да би се то омогућило потребно је успоставити везу са *MySQL* базом за коју *PHP* језик има уграђену подршку:

```
<?php

// Креирање конекције
$bp = mysqli_connect ($servername, $username, $password, $db_name);

// Провера конекције
if (!$bp) {
    die("Неуспела конекција: " . mysqli_connect_error());
}
echo " Конекција успешна!";

// Раскид везе са базом
mysqli_close($bp);

?>
```

PHP функција за успоставу везе састоји се из четири вредности коју враћа *mysqli_connect()*:

- **\$servername** представља *IP* адресу компјутера
- **\$username** представља корисничко име
- **\$password** представља лозинку
- **\$db_name** представља име базе података

Пошто је *MySQL* сервер потпуно засебан део, треба се предвидети могућност да је сервер недоступан или да постоји посебна дозвола приступа или да унесени параметри нису исправни. У том случају је неопходна провера конекције помоћу исказа *if* у којој функција *mysqli_connect_error* пријављује грешку ако постоји.

Успешно успостављена веза са сервером базом података је активна све до краја извршења захтева уколико се посебно не захтева њено раскидање наредбом `mysqli_close()`.

6.5.1 Додавање података у базу

Када је омогућена активна веза са сервером и креирана табела следи додавање података у базу *PHP* наредбом за извршавање *mysqli_query* упита у комбинацији са *MySQL* наредбом *INSERT*:

```
<?php

// Креирање конекције
$bp = mysqli_connect ($servername, $username, $password, $db_name);

// Провера конекције
if (!$bp) {
    die("Неуспела конекција: " . mysqli_connect_error());
}
echo "Конекција успешна!";

// Додавање података у базу
$sql = "INSERT INTO име_табеле (колона1, колона2, колона3)
VALUES ('вредност1', 'вредност2', вредност3)";

if (mysqli_query($bp, $sql)) {
    echo "Подаци успешно креирани!";
} else {
    echo "Грешка: " . $sql . "<br>" . mysqli_error($bp);
}

// Раскид везе са базом
$mysqli_close($bp);

?>
```

6.5.2 Учитавање података из базе

Још једна у низу од могућности манипулисања подацима чија операција је веома једноставна уз употребу наредбе *SELECT* јесте учитавање података из базе:

```
<?php

// Учитавање података из базе
$sql = "SELECT колона1, колона2, колона3 FROM име_табеле ";
$resultat = mysqli_query($bp, $sql);

while($red = mysqli_fetch_object($resultat)) {
    echo " Колона1: " . $red ["колона1"].
        " Колона2: " . $red ["колона2"].
        " Колона3: " . $red ["колона3"]. "<br>";
}

?>
```

6.5.3 Измена података у бази

За разлику од додавања и учитавања података из базе, измена података представља најсложенију операцију руковања подацима због избора података којих треба изменити, учитавања истих и примењивања њихових нових вредности. Наредба је врло проста и гласи *UPDATE*:

```
<?php

// Измена података у бази
$sql = "UPDATE име_табеле SET колона1=' вредност2'
      WHERE колона1=' вредност1' ";

?>
```

6.5.4 Уклањање података из базе

Још једна од комплексних операција руковођења подацима која се врши у једном кораку наредбом *DELETE*:

```
<?php

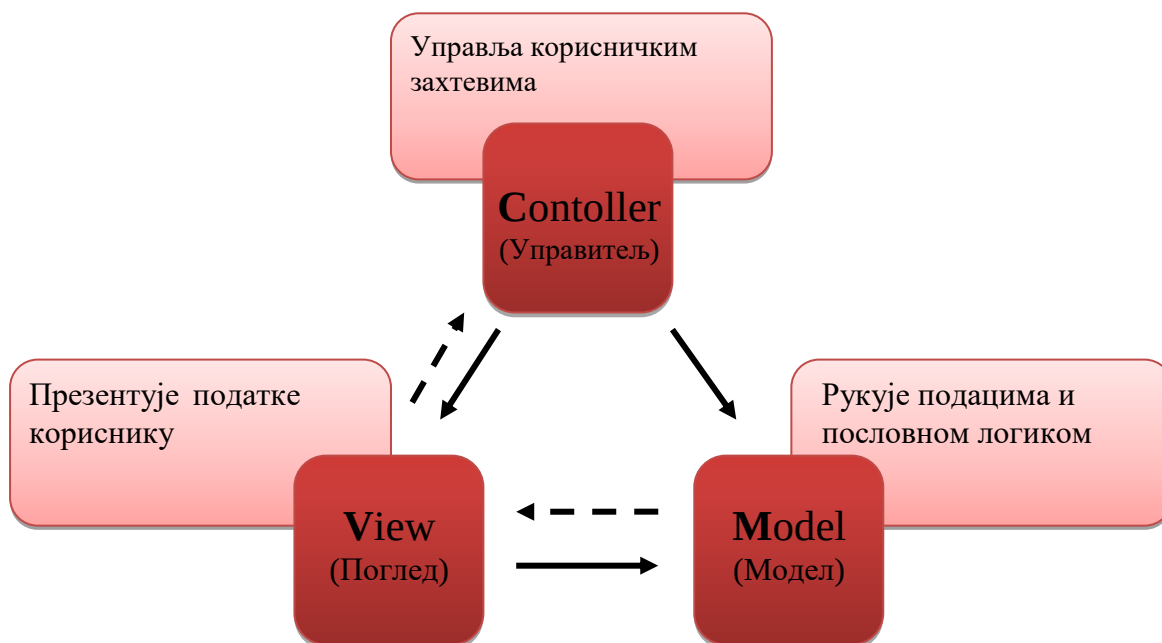
// Брисање података из базе
$sql = "DELETE FROM име_табеле WHERE колона1=' вредност1' ";

?>
```

7 MVC

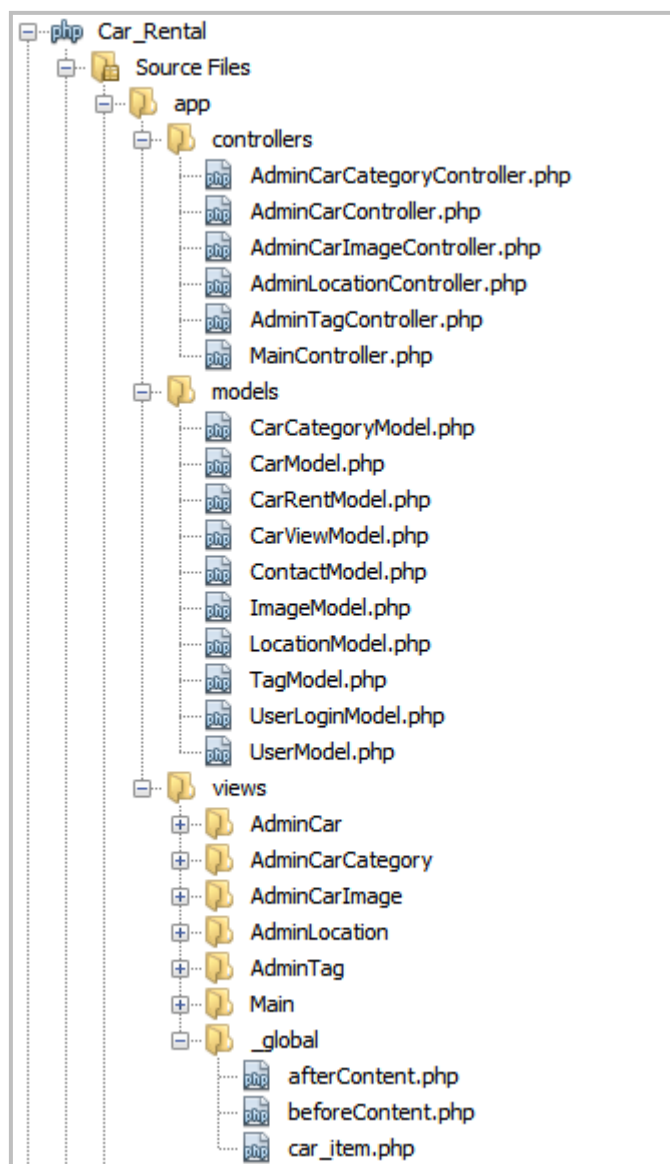
Софтверска архитектура, дизајнерски патерн (енгл. *pattern*) или модел. Све су ово називи којима можемо дефинисати концепт *MVC* (енгл. *Model–View–Controller*), чија је намена одвајање поједних делова веб апликације у компоненте зависно о њиховој намени. Овакав систем је погодан јер услед честих допунских захтева корисника омогућава мењање корисничког интерфејса независно од пословне логики. *MVC* модел се састоји од три целине:

- **Управитељ** (енгл. *Controller*) садржи главну контролу програма и одговоран је за његов ток. Такође, одговоран је за управљање корисничких захтева (*GET* или *POST*). Најчешће управитељ позива одређени модел за задатак и затим одабира одговарајући поглед.
- **Модел** (енгл. *Model*) садржи главне програмске податке које директно позива управитељ, док их модел обрађује и прослеђује њему.
- **Поглед** (енгл. *View*) садржи корисничко окружење апликације и обезбеђује различите начине за презентовања података које добија од модела. Као последњи слој *MVC* архитектуре садржи *HTML*, *CSS*, *JavaScript* итд. и видљив је од стране корисника, док су управитељ и модел сакривени.



Слика 1: MVC модел. Пуна црта приказује директну повезаност док испрекидана индиректну повезаност кроз посматрача

У имплементацији веб апликације за агенцију за издавање путничких возила користила сам MVC патерн чија је структура фолдера и фајлова приказана на следећој слици (Слика 2):



Слика 2: Структура фолдера и фајлова веб апликације за агенцију за издавање путничких возила

8 Апликација

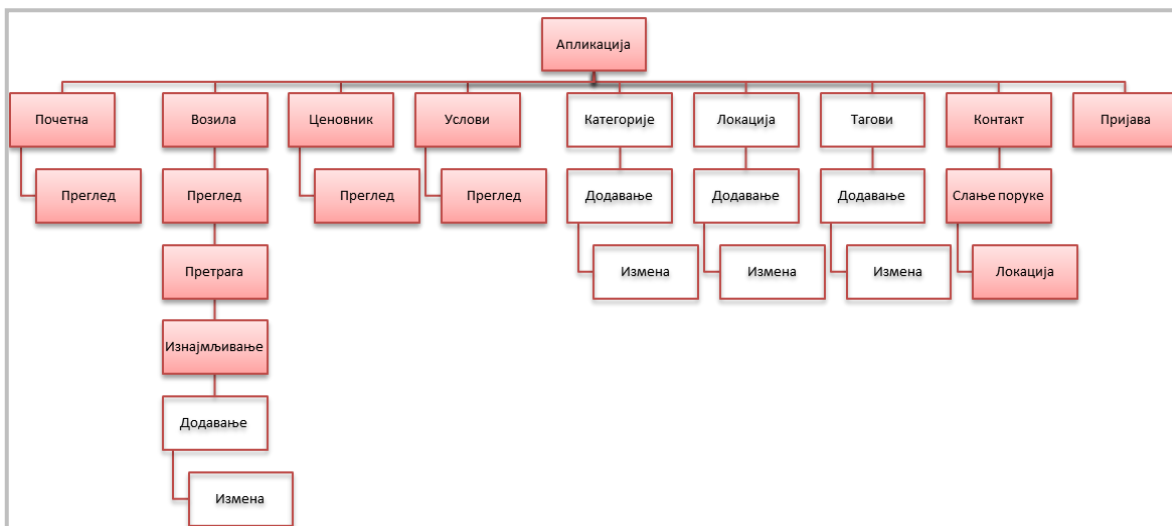
Након сажетог описа веб технологија које сам користила за имплементацију веб апликације за агенцију за изнајмљивање путничких возила на реду је њено приказивање од момента развоја до саме имплементације.

8.1 Развој апликације

Ради бољег разумевања услуга и функција које пружа веб апликација неопходно је приказати њене моделе понашања, базе података и почетне стране.

8.1.1 Преглед мапе апликације

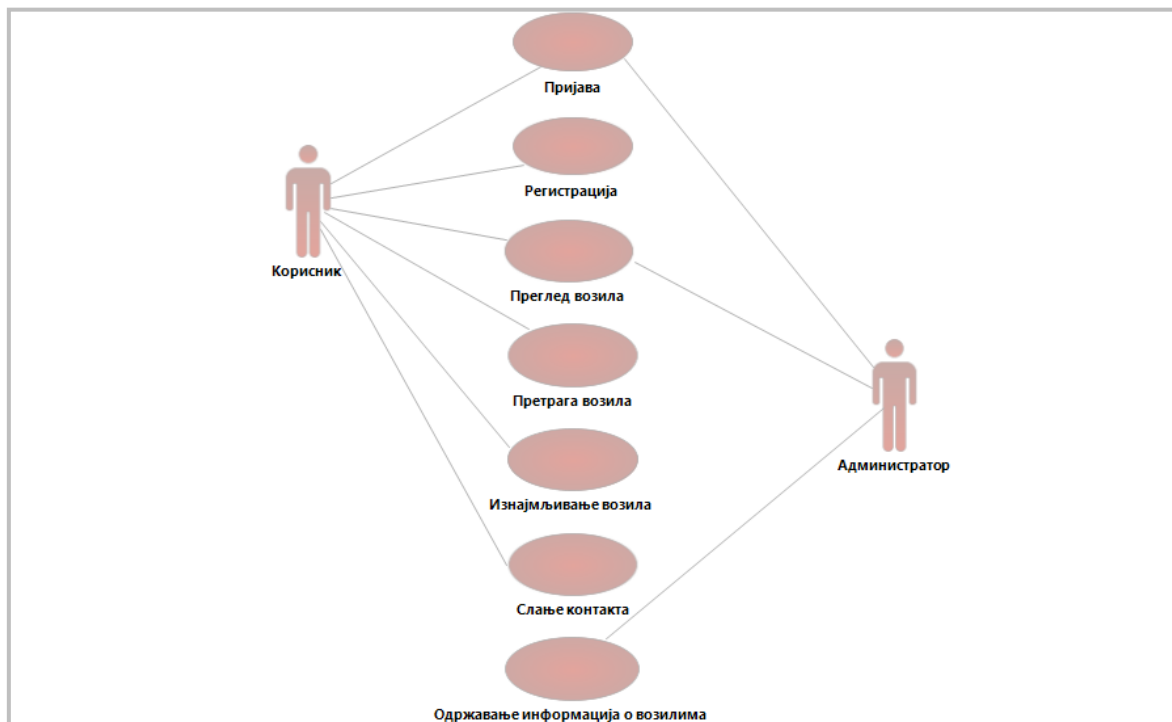
Служи за приказ свих страница веб апликације у оквиру којих корисник и администратор имају одвојене прегледе и различите функције:



Слика 1: Мапа веб апликације у којој обојена поља представљају функције корисника, а необојена функције администратора

8.1.2 Преглед функција корисника

Омогућује приказ функционисања веб апликације са корисничке стране у оквиру кога корисник поседује могућност за пријавом или регистрациом, прегледом, претраживањем и/или изнајмљивањем возила и слањем контакт форме:

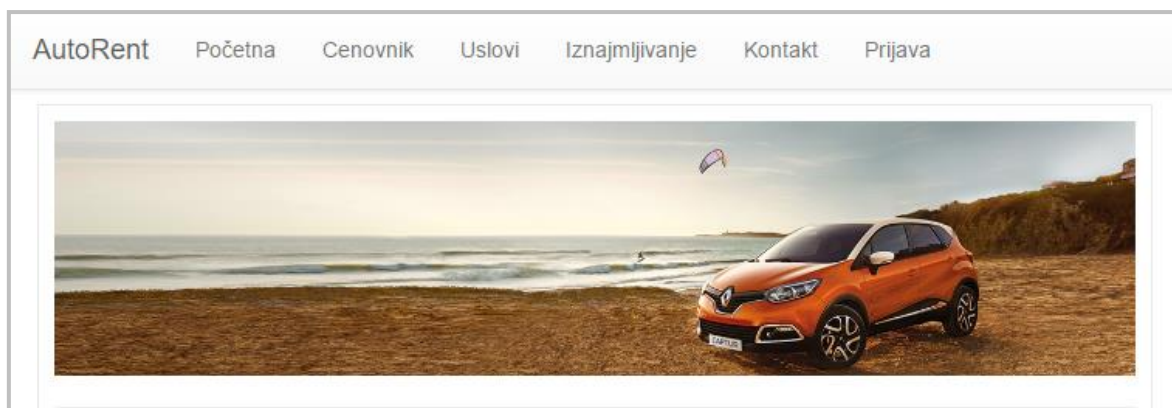


Слика 1: Дијаграм случајева коришћења

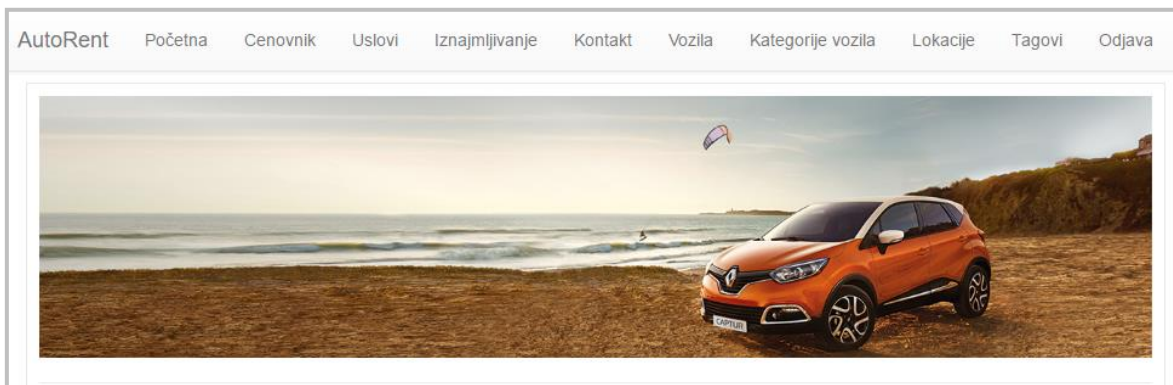
8.1.3 Преглед модела почетне стране

Прва страница коју корисник види приликом отварања веб апликације јесте почетна страница. Зато је од великог значаја сам њен дизајн који подразумева прегледност, континуираност и адекватност садржаја који сам се трудила да испоштујем у самој апликације за изнајмљивање возила. Због тога почетна страница садржи три целине:

Навигациони мени чија је обимност страница сведена на минимум и који се разликује за кориснике и администратора:



Слика 1: Навигациони мени корисника



Слика 2: Навигациони мени администратора

Главни садржај који садржи једноставну форму претраге возила по жељеним критеријумима:

Najam

Lokacija

Beograd

Kategorija

Kupe

☐ 4+1 putnika
 ☒ 5 brzina
 ☐ 5 vrata
 ☐ Dečije sedište
 ☐ GPS

☐ Klima
 ☐ Prva pomoć
 ☐ Pun rezervoar
 ☐ Zimske gume

Pretraga

Renault Twingo

Renault Twingo

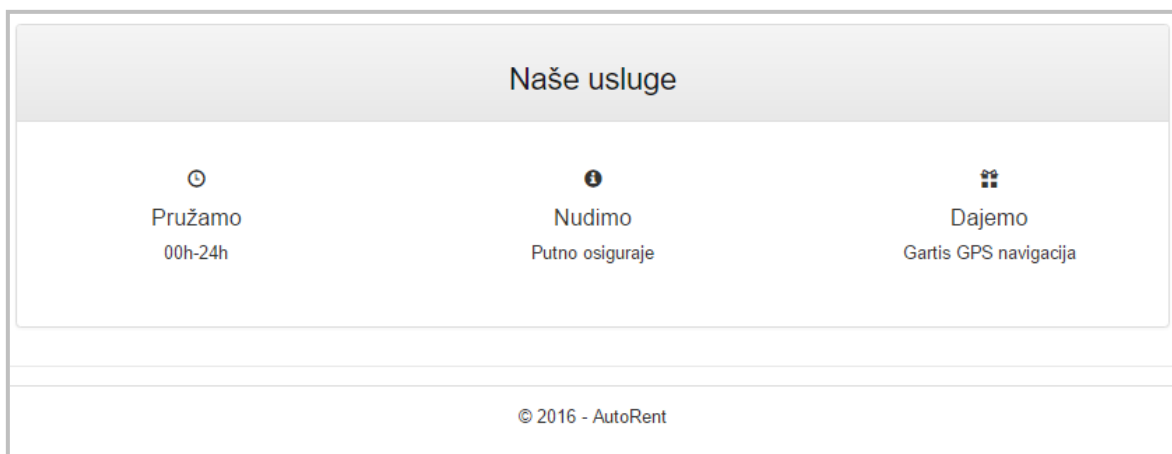
49 €

☒ 4+1 putnika
 ☒ 5 brzina
 ☒ 5 vrata
 ☒ Dečije sedište
 ☒ GPS

☒ Klima
 ☒ Prva pomoć
 ☒ Pun rezervoar
 ☒ Zimske gume

Слика 3: Форма претраге возила

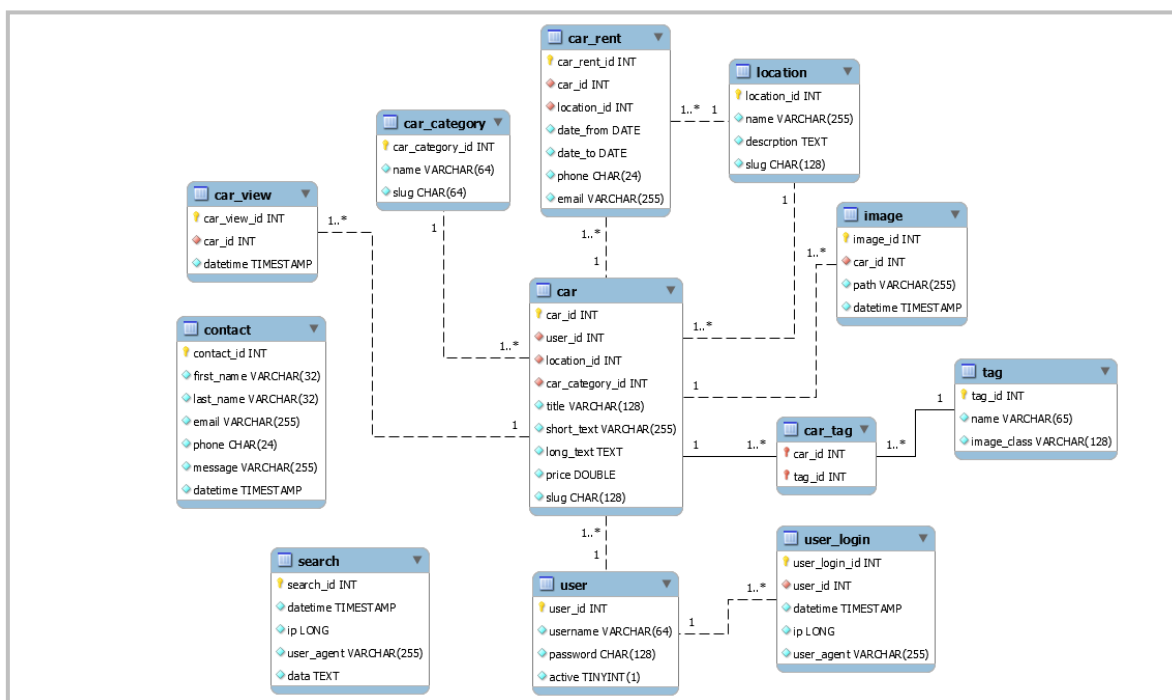
Заглавље (енгл. *footer*) странице које приказује посебне погодности и права апликације за изнајмљивање возила:



Слика 4: Заглавље почетна странице

8.1.4 Преглед модела базе података

Приказује све табле базе података веб апликације и њихове међусобне везе:



Слика 1: Модел базе података

У следећим листинзима детаљно су приказане поједине табеле из базе података веб апликације за изнајмљивање путничких возила:

Табела **корисник** (енгл. *user*) са примарним кључем *user_id* и атрибутима *username*, *password* и *active*:

user - Table										
			Table Name: user		Schema: car_rental					
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
user_id	INT	☒	☒	☐	☐	☒	☐	☒	☐	
username	VARCHAR(64)	☐	☒	☒	☐	☐	☐	☐	☐	
password	CHAR(128)	☐	☒	☐	☐	☐	☐	☐	☐	
active	TINYINT(1)	☐	☒	☐	☐	☒	☐	☐	☐	1

Табела **возило** (енгл. *car*) са примарним кључем *car_id* , атрибутима *title*, *short_text*, *long_text*, *price*, *slug* и спољним кључевима *location_id*, *user_id* и *car_category_id*:

car - Table										
			Table Name: car		Schema: car_rental					
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
car_id	INT	☒	☒	☐	☐	☒	☐	☒	☐	
user_id	INT	☐	☒	☐	☐	☒	☐	☐	☐	
location_id	INT	☐	☒	☐	☐	☒	☐	☐	☐	
car_category_id	INT	☐	☒	☐	☐	☒	☐	☐	☐	
title	VARCHAR(128)	☐	☒	☐	☐	☐	☐	☐	☐	
short_text	VARCHAR(255)	☐	☒	☐	☐	☐	☐	☐	☐	
long_text	TEXT	☐	☒	☐	☐	☐	☐	☐	☐	
price	DOUBLE	☐	☒	☐	☐	☒	☐	☐	☐	
slug	CHAR(128)	☐	☒	☐	☐	☐	☐	☐	☐	

Табела **катеорије возила** (енгл. *car_category*) са примарним кључем *car_category_id* и атрибутима *name* и *slug*:

car_category - Table										
			Table Name: car_category		Schema: car_rental					
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
car_category_id	INT	☒	☒	☐	☐	☒	☐	☒	☐	
name	VARCHAR(64)	☐	☒	☒	☐	☐	☐	☐	☐	
slug	CHAR(64)	☐	☒	☒	☐	☐	☐	☐	☐	

Табела **изнајмљивање возила** (енгл. *car_rent*) са примарним кључем *car_rent_id*, атрибутима *date_from*, *date_to*, *phone*, *email* и спољним кључевима *location_id* и *car_id* :

car_rent - Table										
			Table Name: car_rent		Schema: car_rental					
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
car_rent_id	INT	☒	☒	☐	☐	☒	☐	☒	☐	
car_id	INT	☐	☒	☐	☐	☒	☐	☐	☐	
location_id	INT	☐	☒	☐	☐	☒	☐	☐	☐	
date_from	DATE	☐	☒	☐	☐	☐	☐	☐	☐	
date_to	DATE	☐	☒	☐	☐	☐	☐	☐	☐	
phone	CHAR(24)	☐	☒	☒	☐	☐	☐	☐	☐	
email	VARCHAR(255)	☐	☒	☐	☐	☐	☐	☐	☐	

8.2 Функције апликације

Представљају срж саме веб апликације јер захваљујући њима администратор може добити и доставити информације од корисника неопходне за изнајмљивање путничких возила што је и главна намена ове веб апликације. У зависности од привилегија разликујемо корисничке функције од администраторски и најбитније ће бити приказане у наставку мог рада.

8.2.1 Пријава

Представља први и најважнији корак администратора у процесу коришћења веб апликације јер помоћу пријаве добија сва права и привилегије за коришћењем података веб апликације. Да би пријава била успешна потребно је да постоји администраторски налог у бази података апликације и да се унесе валидно име и лозинка приликом пријаве:

+ Options

←

→

▼

	user_id	username	password	active
✎ ✂ ✖ EditCopyDelete 1admin7fbbf6c476fc642cb89db25a2abf9dba069044ac98c207edf8...				1

⬆

☐

Check all

With selected:

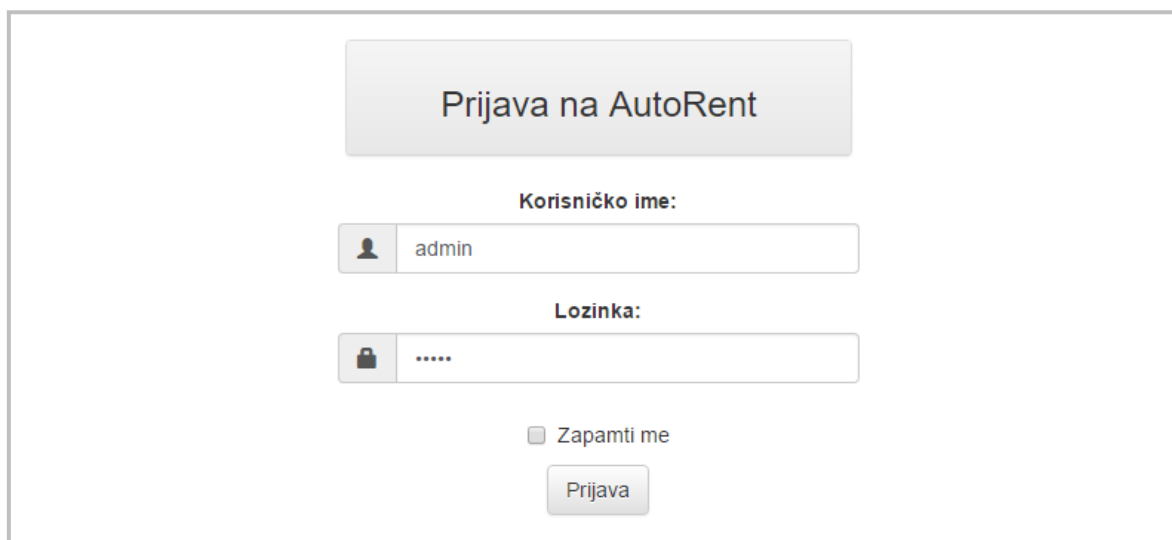
✎

✂

✖

EditCopyDeleteExport

Слика 1: Приказ администраторског налога у бази података



Prijava na AutoRent

Korisničko ime:

Lozinka:

☐ Zapamti me

Слика 2: Пријава администратора на веб апликацију

Такође, у главном управљачу веб апликације у оквиру *PHP* фајла треба да постоји функција пријаве (енгл. *login*):

```
/**
 * Овај метод проверава да ли постоје подаци за пријаву послати
 * HTTP POST методом, врши њихову валидацију, проверава постојање
 * корисника са тим приступним параметрима и у случају да све провере прођу
 * без грешке метод креира сесију за корисника и преусмерава корисника на
 * стандардну (енгл. default) руту.
 * @return void метод не враћа ништа, већ користи return наредбу за
 * прекид извршавања у одређеним ситуацијама.
 */

function login() {
    if ($_POST) {
        $username = filter_input(INPUT_POST, 'username');
        $password = filter_input(INPUT_POST, 'password');

        if (!preg_match('/^[a-z0-9]{4,}$/', $username) or ! preg_match('/^{5,}$/',
        $password)) {
            $this->set('message', 'Погрешно унето корисничко име или лозинка. ');
            return;
        }

        $passwordHash = hash('sha512', $password . Configuration::SALT);
        $user = UserModel::getActiveUserByUsernameAndPasswordHash($username,
        $passwordHash);

        if (!$user) {
            $this->set('message', 'Корисничко име или лозинка нису валидни или
            корисник није активан. ');
            return;
        }

        Session::set('user_id', $user->user_id);
        Session::set('username', $username);
        Session::set('user_ip', filter_input(INPUT_SERVER, 'REMOTE_ADDR',
        FILTER_FLAG_IPV4));
        Session::set('user_agent', filter_input(INPUT_SERVER,
        'HTTP_USER_AGENT'));

        Misc::redirect('');
    }
}
```

8.2.2 Додавање путничког возила

Функција администратора која пружа унос новог путничког возила на основу марке и модела возила, кратког и детаљног описа и на основу кључних речи:

Spisak svih putnickih vozila			
Dodati novo vozilo			
Id	Vozilo	Slug	Opcije
1	Renault Captur	renault-captur	Izmena Slika
8	Renault Clio	renault-clio	Izmena Slika
2	Renault Duster	renault-duster	Izmena Slika
4	Renault Grand Scenic	renault-grand-scenic	Izmena Slika
10	Renault Laguna	renault-laguna	Izmena Slika
12	Renault Laguna Karavan	renault-laguna-karavan	Izmena Slika
16	Renault Megane	renault-megane	Izmena Slika
14	Renault Megane Cabrio	renault-megane-cabrio	Izmena Slika
15	Renault Megane G7	renault-megane-g7	Izmena Slika
6	Renault Twingo	renault-twingo	Izmena Slika

Слика 1: Приказ свих постојећих возила

Dodavanje novog putnickog vozila

Ime

Slug

Kratak tekst

Detaljan opis

Cena po danu

Lokacija

Kategorija

Specijalni dodaci/tagovi
☒ 4+1 putnika ☐ 5 brzina ☐ 5 vrata ☐ Dečije
sedište ☒ GPS ☐ Klima ☒ Prva pomoć ☐ Pun
rezervoar ☒ Zimske gume

Id	Vozilo
1	Renault Captur
8	Renault Clio
2	Renault Duster
4	Renault Grand Scenic
17	Renault Kadjar
10	Renault Laguna
12	Renault Laguna Karavan
16	Renault Megane
14	Renault Megane Cabrio
15	Renault Megane G7
6	Renault Twingo

Слика 2: Додавање новог путничког возила

У управљачу *AdminCarController* посебно креираног за управљањем подацима о возилима потребно је направити функцију додавања возила (енгл. *add*) како би се омогућило уношење новог возила:

```
/**
 * Овај метод приказује формулар за додвање или врши додавање ако су подаци
 * послати HTTP POST методом.
 * @return void
 */

public function add() {
    $this->set('locations', LocationModel::getAll());
    $this->set('categories', CarCategoryModel::getAll());
    $this->set('tags', TagModel::getAll());

    if (!$_POST) return;

    $title = filter_input(INPUT_POST, 'title');
    $slug = filter_input(INPUT_POST, 'slug');
    $short_text = filter_input(INPUT_POST, 'short_text');
    $long_text = filter_input(INPUT_POST, 'long_text');
    $price = floatval(filter_input(INPUT_POST, 'price'));

    $location_id = filter_input(INPUT_POST, 'location_id',
        FILTER_SANITIZE_NUMBER_INT);
    $car_category_id = filter_input(INPUT_POST, 'car_category_id',
        FILTER_SANITIZE_NUMBER_INT);

    $scar_id = CarModel::add($title, $slug, $short_text, $long_text, $price,
        $location_id, $car_category_id, Session::get('user_id'));

    if ($scar_id) {
        $tag_ids = filter_input(INPUT_POST, 'tag_ids',
            FILTER_SANITIZE_NUMBER_INT, FILTER_REQUIRE_ARRAY);

        foreach ($tag_ids as $tag_id) {
            CarModel::addTagToCar($scar_id, $tag_id);
        }

        Misc::redirect('admin/cars/');
    } else {
        $this->set('message', 'Дошло је до грешке приликом додавања путничког
            возила у базу података. ');
    }
}
```

Такође, у моделу веб апликације неопходно је направити идентичну функцију као код управљача:

```
/**
 * Метод врши унос нове ставке у бази података за путничко возило.
 * @param type $title
 * @param type $slug
 * @param type $short_text
 * @param type $long_text
 * @param type $price
 * @param type $location_id
 * @param type $car_category_id
 * @return int|boolean ID број путничког возила или
 * FALSE у случају грешке
 */

public static function add($title, $slug, $short_text, $long_text, $price,
    $location_id, $car_category_id, $user_id) {

    $SQL = 'INSERT INTO car (title, slug, short_text, long_text, price,
        location_id, car_category_id, user_id) VALUES (?, ?, ?, ?, ?, ?, ?, ?)';

    $prep = DataBase::getInstance()->prepare($SQL);
    $res = $prep->execute([$title, $slug, $short_text, $long_text, $price,
        $location_id, $car_category_id, $user_id]);

    if ($res)
        return DataBase::getInstance()->lastInsertId();
    else
        return false;
}
```

8.2.3 Измена путничког возила

Још једна функција администратора која омогућује мењање података о путничким возилима на основу истих параметра као и код додавања возила:

Izmena vozila

Ime

Slug

Kratak tekst

Detaljan opis

Cena po satu

Lokacija

Kategorija

Specijalni dodaci/tagovi

☐ 4+1 putnika
☐ 5 brzina
☐ 5 vrata
☐ Dečije sedište

☐ GPS
☐ Klima
☐ Prva pomoć
☐ Pun rezervoar
☐ Zimske gume

Id	Vozilo
1	Renault Captur
8	Renault Clio
2	Renault Duster
4	Renault Grand Scenic
17	Renault Kadar
10	Renault Laguna
12	Renault Laguna Karavan
16	Renault Megane
14	Renault Megane Cabrio
15	Renault Megane G7
6	Renault Twingo

Слика 1: Измена путничког возила

Као и код убацивања новог возила у управљачу *AdminCarController* потребно је направити функцију измене возила (енгл. *edit*):

```

/**
 * Овај метод приказује формулар за измену или врши измену ако су
 * подаци послати HTTP POST методом.
 * @return void
 */

public function edit($id) {
    $this->set('locations', LocationModel::getAll());
    $this->set('categories', CarCategoryModel::getAll());
    $this->set('tags', TagModel::getAll());

    $car = CarModel::getById($id);

```

```
if (!$car) {
    Misc::redirect('admin/cars/');
}

$car->tags = CarModel::getTagsForCarId($id);
$car->tag_ids = [];
foreach ($car->tags as $tag) {
    $car->tag_ids[] = $tag->tag_id;
}
$this->set('car', $car);

if (!$_POST) return;

$title = filter_input(INPUT_POST, 'title');
$slug = filter_input(INPUT_POST, 'slug');
$short_text = filter_input(INPUT_POST, 'short_text');
$long_text = filter_input(INPUT_POST, 'long_text');
$price = floatval(filter_input(INPUT_POST, 'price'));

$location_id = filter_input(INPUT_POST, 'location_id',
    FILTER_SANITIZE_NUMBER_INT);
$car_category_id = filter_input(INPUT_POST, 'car_category_id',
    FILTER_SANITIZE_NUMBER_INT);

$res = CarModel::edit($id, $title, $slug, $short_text, $long_text, $price,
    $location_id, $car_category_id);

$tag_ids = filter_input(INPUT_POST, 'tag_ids',
    FILTER_SANITIZE_NUMBER_INT, FILTER_REQUIRE_ARRAY);

CarModel::deleteAllTags($id);
foreach ($tag_ids as $tag_id) {
    CarModel::addTagToCar($id, $tag_id);
}

if ($res) {
    Misc::redirect('admin/cars/');
} else {
    $this->set('message', 'Дошло је до грешке приликом измене података о
        путничком возилу.');
```

Исто тако потребно је и модел измене пројектовати:

```
/**
 * Метод мења садржај записа у бази података за изабрано возило.
 * @param type $id
 * @param type $title
 * @param type $slug
 * @param type $short_text
 * @param type $long_text
 * @param type $price
 * @param type $location_id
 * @param type $car_category_id
 * @return boolean
 */

public static function edit($id, $title, $slug, $short_text, $long_text, $price,
    $location_id, $car_category_id) {

    $SQL = 'UPDATE car SET title = ?, slug = ?, short_text = ?, long_text = ?,
        price = ?, location_id = ?, car_category_id = ? WHERE car_id = ?;';

    $prep = DataBase::getInstance()->prepare($SQL);

    return $prep->execute([$title, $slug, $short_text, $long_text, $price,
        $location_id, $car_category_id, $id]);
}
```

8.2.4 Изнајмљивање возила

Једна од најважнији и најбитнијих функција коју корисник може да користи јесте изнајмљивање возила. Веб апликација пружа кориснику могућност најма возила на више места уз олакшице у виду брзе претраге траженог возила:

Najam

Lokacija

Novi Sad

Kategorija

Džip

☐ 4+1 putnika
 ☐ 5 brzina
 ☐ 5 vrata
 ☐ Dečije sedište
 ☐ GPS
 ☐ Klima

☐ Prva pomoć
 ☐ Pun rezervoar
 ☐ Zimske gume

Pretraga



Renault Duster

Renault Duster

70 €

Слика 1: Претрага путничког возила по задатим критеријумима

Кориснику се нуди могућност претраге возила на основу три критеријума (локације, категорије и кључних речи возила). Метод управљача за форму претраге возила се имплементира по сличном принципу додавања и измене возила, док се модел претраге знатно разликује. У наставку је дат део кода:

```
/**
 * Метод враћа резултате претраге по датим критеријумима
 * @param int $location_id
 * @param int $car_category_id
 * @param array $tag_ids
 * @return array
 */

public static function homePageSearch($location_id, $car_category_id, $tag_ids) {
    if ($location_id != -1 and $car_category_id != -1 and count($tag_ids) > 0) {
        $tag_placeholders = [];
        for ($i = 0; $i < count($tag_ids); $i++) {
            $tag_placeholders[] = '?';
        }
        $tag_placeholder_string = implode(' ', $tag_placeholders);
    }
}
```

```
$SQL = 'SELECT * FROM car ' .  
      'WHERE location_id = ? AND car_category_id = ? AND car_id IN  
      (SELECT car_id FROM car_tag WHERE tag_id IN  
      (' . $tag_placeholder_string . ')) ' . 'ORDER BY `title`';  
$prep = DataBase::getInstance()->prepare($SQL);  
$niz = [  
    $location_id,  
    $car_category_id  
];  
$niz = array_merge($niz, $tag_ids);  
$prep->execute($niz);  
return $prep->fetchAll(PDO::FETCH_OBJ);  
}  
}
```

Сам процес изнајмљивања жељеног возила се врши у пар једноставних корака, што је још једна одлика ове веб апликације. За имплементацију функције најма возила такође је потребно направити посебан управљач и модел по принципу на претходне:

The screenshot shows a web form titled "Iznajmljivanje vozila" (Car Rental) within the "AutoRent" application. The form is contained within a light gray box and has a title bar. It includes the following fields and elements:

- Od:** A date input field with a calendar icon and placeholder "dd-----yyyy".
- Do:** A date input field with a calendar icon and placeholder "dd-----yyyy".
- Vozilo:** A dropdown menu with a car icon and placeholder "---".
- Lokacija:** A dropdown menu with a location pin icon and placeholder "---".
- Email:** A text input field with an email icon and placeholder "Unesite email adresu".
- Telefon:** A text input field with a phone icon and placeholder "Unesite broj telefona".
- Preko 25 godina:** A checkbox.
- Iznajmi:** A button at the bottom right of the form.

Слика 2: Форма најма путничког возила

```
/**
 * Управљач чији метод приказује формулар за изнајмљивање
 * путничког возила послати HTTP POST методом.
 * @return void
 */

function carRent() {
    $this->set('cars', CarModel::getAll());
    $this->set('locations', LocationModel::getAll());

    if ($_POST) {
        $date_from = filter_input(INPUT_POST, 'date_from');
        $date_to = filter_input(INPUT_POST, 'date_to');
        $car_id = filter_input(INPUT_POST, 'car_id',
            FILTER_SANITIZE_NUMBER_INT);
        $location_id = filter_input(INPUT_POST, 'location_id',
            FILTER_SANITIZE_NUMBER_INT);
        $email = filter_input(INPUT_POST, 'email',
            FILTER_SANITIZE_EMAIL);
        $phone = filter_input(INPUT_POST, 'phone');

        $car_rent_id = CarRentModel::add($car_id, $location_id, $date_from,
            $date_to, $phone, $email);

        if ($car_rent_id) {
            $this->set('message', 'Возило је успешно изнајмљено.');
        } else {
            $this->set('message', 'Дошло је до грешке. Позовите администратора');
        }
    }
}

/**
 * Модел чији метод врши изнајмљивање изабраног путничког возила
 * @param type $car_id
 * @param type $location_id
 * @param type $date_from
 * @param type $date_to
 * @param type $phone
 * @param type $email
 * @return boolean
 */

public static function add($car_id, $location_id, $date_from, $date_to, $phone,
    $email) {
    $SQL = 'INSERT INTO car_rent (car_id, location_id, date_from, date_to,
        phone, email) VALUES (?, ?, ?, ?, ?, ?)';
```

```
$prep = DataBase::getInstance()->prepare($SQL);

$res = $prep->execute([$car_id, $location_id, $date_from, $date_to, $phone,
$email]);

if ($res)
    return DataBase::getInstance()->lastInsertId();
else
    return false;
}
```

8.2.5 Слање контакт форме

Подразумевана функција корисника која омогућује слање поруке администратору уколико постоје питања или потреба за додатним информисањем.

first_name	last_name	email	phone	message
Ana	Anic	ana@gmail.com	0848485806	/
Petar	Petrovic	petar@gmail.com	08411385828	/
David	Kukolj	dkukolj@gmail.com	084065060	Poštovani,

Слика 1: Приказ послате поруке корисника у бази података

Kontaktirajte nas

Ime: David

Prezime: Kukolj

Email: dkukolj@gmail.com

Telefon: 064065060

Poštovani,

Želeo bih da znam da li imate u ponudi Renault Kadiar?

Pozdrav!

Pošalji

Слика 1: Контакт форма

У наставку су дати примери управљача и модела контакт форме:

```
/**
 * Управљач чији метод приказује формулар за контакт форму
 * @return void
 */

function contact() {
    if ($_POST) {
        $first_name = filter_input(INPUT_POST, 'first_name');
        $last_name = filter_input(INPUT_POST, 'last_name');
        $email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);
        $phone = filter_input(INPUT_POST, 'phone');
        $message = filter_input(INPUT_POST, 'message');

        $contact_id = ContactModel::add($contact_id, $first_name, $last_name, $email,
        $phone, $message );
        if ($contact_id) {
            $this->set('message', 'Контакт је успешно послат.');
        } else {
            $this->set('message', 'Дошло је до грешке. Позовите администратора');
        }
    }
}

/**
 * Модел чији метод врши слање корисничке поруке
 * @param type $contact_id
 * @param type $first_name
 * @param type $last_name
 * @param type $email
 * @param type $phone
 * @param type $message
 * @return boolean
 */

public static function add($contact_id, $first_name, $last_name, $email, $phone,
    $message ) {

    $SQL = 'INSERT INTO contact (contact_id, first_name, last_name, email, phone,
    message) VALUES (?, ?, ?, ?, ?, ?)';
    $prep = DataBase::getInstance()->prepare($SQL);
    $res = $prep->execute([$contact_id, $first_name, $last_name, $email, $phone,
    $message]);
    if ($res)
        return DataBase::getInstance()->lastInsertId();
    else
        return false;
}
```

9 Закључак

HTML језик за означавање хипертекста представља темељ веб стандарда и са пуним правом сврстава се у веома моћан језик за структурирање. У почетку је био језик са доста ограничених и слабих могућности, а данас његова опсежност и подршка је у сталном напретку. Тренутно доступна верзија *HTML 5*, поседује много новина које су ми пружиле приказивање апликације на интернету и олакшале комбиновање различитих технологија и тиме омогућиле изградњу јаче и богатије веб апликације за издавање путничких возила.

Стилски језик *CSS* данас представља основу дизајна сваког *HTML* документа. Омогућује растерећење ознака унутар *HTML* странице јер се користи за обликовање и презентацију, односно опис документа написаног у *HTML* програмском језику. Најбољи је језик за стилизовање и као такав је незаменљив. Упозната са тим, користила сам га за прављење привлачне и естетски допадљиве веб апликације и тиме испунила један од задатих циљева овог рада.

JavaScript језик допринео је веб апликацији за изнајмљивање путничких возила, динамичност и већу интеракцију између ње и њених корисника. Као један од најпопуларнијих скрипних језика на вебу пружио ми је неограничену креативност и тиме допринео јединствен изглед веб апликације. Слободно могу рећи да је један моћан инструмент на пољу веб дизајна из простог разлога што омогућава бољи дизајн, читање и писање *HTML* елемената и препознавање веб претраживача као и самог корисника.

Интерфејс између серверске стране и корисника *Bootstrap* је најпопуларнији алат за израду веб апликација. Разлози за његово коришћење у имплементацији веб апликације за изнајмљивање путничких возила су компатибилност са свим веб претраживачима, стандардизовање прилагодљивог дизајна на свим уређајима и обезбеђивање јединствених компоненти које ова веб апликација изискује.

PHP годинама важи за један од популарнијих програмски језика (одмах иза језика *Java*, *C*, *C++*) због своје универзалности. Зато је он био мој избор приликом израде веб апликације јер омогућује рад на већини веб сервера и готово свим оперативним системима и платформама. Веома моћан, поуздан и стабилан језик отвореног кода чијом је употребом решена функционалост веб апликације као један од њених најбитнијих и најважнијих задатака.

MVC модел као облик архитектуре веб апликације коришћен је за организацију делова исте у три групације компонената где свака групација компоненти има своју посебну намену. Оваква архитектура олакшала је независан развој, тестирање и одржавање веб апликације за изнајмљивање путничких возила.

Дигитална револуција одиграла је велику улогу у нашим животима. Данас не постоји особа која не користи услуге интернета за добијање и размењивање информација. Због ове чињенице неопходно их је учинити доступним. Апликација за агенцију за изнајмљивање путничких возила ради на том принципу. Својим једноставним, прилагодљивим и функционалним дизајном омогућује свим корисницима приступачност информација за лако, ефикасно и економично изнајмљивање возила.

10 Литература

1. Веиновић, М. и Јевремовић, А. (2013) Интернет технологије, прво издање, Универзитет Сингидунум, Београд
2. Веиновић, М. и Јевремовић, А. (2013) Рачунарске мреже, треће издање, Универзитет Сингидунум, Београд
3. Веиновић, М., Шимић, Г., Јевремовић, А., и Франц, И. (2013) Базе података, прво издање, Универзитет Сингидунум, Београд
4. Његуш, А. (2015) Пословни информациони системи, седмо издање, Универзитет Сингидунум, Београд
5. Мишковиц, В. (2016) Практикум- Интернет и Веб технологије, Универзитет Сингидунум, Београд
<http://predmet.singidunum.ac.rs/course/view.php?id=261>