Увод

Динамичното развиване на интернет дава възможност на хората да не си пазят информацията на своя компютър, а директно да я запазват по различни сайтове. Пример за това е запазването на изображения във „Facebook“. Предимсвото от запазването на информация „online“ е, че така тя бива достъпвана много лесно, не заема място на вашия компютър и също така при повреда на компютъра Ви, информацията не изчезва.Всичко от което се нуждаете за достъп до вашата информация е компютър и достъп до интернет.

Целта на настоящата дипломна работа е създаване и реализиране на система за управление на снимки. Системата трябва да поддържа работа с много потребители, за които се запазва набор от снимки разделени в отделни директории.

Системата(сайта) ще бъде качена в интернет на реално IP. Достъп до системата ще има всеки потребител, имащ интернет или физически достъп до нея. Единственото ограничение за използване на системата е потребителя да бъде регистриран в нея.

Благодарности

Специални благодарности дължа на моя брат Ивайло Радев, без когото създаването на „Фотоалбума“ нямаше да е възможно в настоящия му вид.

# Глава 1 - Състояние на проблема по литературни данни.

Бързият прогрес в развитието на интернет и използването му от все повече хора, налага необходимостта от повече сайтове за съхранение на информация. Интернет се превръща във все по-важно средство в ежедневието на човека. В наши дни хора и компании все повече се обръщат към онлайн услугите за съхранение и архивиране на своите поверителни документи и данни. Онлайн съхранението на данни има няколко предимства в сравнение с локалното съхранение. С други думи имате достъп до вашите данни от всяко място с интернет връзка, а те са защитени от загуба поради кражба или злополука.

Има много онлайн услуги за съхранение на файлове и изборът коя да изберете може да бъде труден. Преди да изберете конкретна услуга за съхранение на документи, първо направете някои проучвания. Определете вашите нужди и внимателно преценете опциите си.

* **Определете вашите нужди**

Преди да изберете измежду различните услуги за съхранение на документи ще се наложи първо да определите какви са вашите изисквания. Какво ще съхранявате или архивирате – снимки, документи, или дори цялата операционна система? Знаейки това, от което имате нужда, ще можете да изберете и услуга, която подхожда най-много на вашите нужди и изисквания.

* **Проверете вашата интернет връзка**

Преди да изберете онлайн услуга за съхранение, проверете дали вашата интернет връзка може да я поддържа. Много хора избират услуга и чак след това осъзнават, че тяхната пропускателна способност не е достатъчна.

* **Уверете се, че вашата операционна система е съвместима**

Повечето онлайн услуги за съхранение и архивиране са съвместими с операционните системи Mac и Windows, а потребителите на Linux и другите операционни системи имат много по-малко възможности. Преди да изберете определена услуга, проверете нейната съвместимост с операционната система на компютрите Ви.

* **Започнете с безплатната версия на услугата, която изберете**

Един много умен ход е преди да се обвържете с платената версия на някоя услуга първо да тествате известно време безплатната ѝ версия. Това ще ви даде възможност да опознаете добрите и лоши страни на услугата и да прецените дали си струва да плащате за нея.

Има много добри и не чак толкова сайтове за съхранение на снимки. Избора кой да използвате зависи изцяло от нуждите Ви. Например ако искате просто да съхранявате снимките си онлайн, без никакви други функционалности, то тогава “Facebook” може би не би бил толкова добър избор. Това е така, защото “Facebook” предлага много други функционалности освен запазване на снимки, които вас не биха Ви интересували. В случая някой по-прост сайт който предлага само съхранение на снимки би бил много по-удачен избор, като например „Пикаса“.

В настоящия случай ние ще разгледаме сайт който предоставя възможността за съхранение на снимки. Сайта симулира стукртура на файлова система на компютъра, където подредбата на снимки и директории илюстрират подредба на „дърво“.

Той е изключително подходящ за запазване на снимки, защото потребителя ще бъде улеснен от подредбата му, тъй като интерфейса му наподобява изключително много този на файловата система на компютъра. Сайта ще има минимални операции върху снимките и потребителите няма да са объркани от сложността на сайта, а ще са фокусирани само въру операциите които искат да извършат върху снимките.

Предимства от използването му са:

* бърз достъп до вашите снимки
* не е нужно да съхранявате снимките на компютъра си
* лесен за използване
* не заема място на компютъра
* достъп от различни краища на света
* достъп от различни компютри
* при повреда или кражба на компютъра, информацията Ви се запазва
* лесно споделяне на снимките си с приятел или близък

## 1.1. Сигурност

Това е най-важният аспект, от който зависи дали потребителите ще предпочетат да запазват информацията си онлайн. Когато я пазите локално, то риска да загубите нещо е много голям. Представете си, че Ви се повреди компютъра, изгори Ви твърдия диск или още по-лошо – бъде откраднат. Тогава информацията Ви се загубва, а в последния случай могат и да злоупотребят с нея. Точно тук е най-положителната част на запазването на информацията онлайн – сигурността. Няма нужда да пазите информация по флашки, дискове или други устройства. Запазвате я онлайн и после забравяте да мислите за нея. Знаете, че може да бъде достъпвана по-всяко време и Вие решавате дали да е само от Вас или искате и други хора да разберат за нея.

## 1.2. Анализ на съществуващи системи

Има и други компании, които използват подобен софтуер за споделяне на снимки. Нека разгледаме най-често срещаните такива продукти:



Фиг.1 Лого на „Facebook“

“Facebook” е може би най-популярния сайт за споделяне на снимки. Това се дължи на разнообразиeтo от услуги които предлага “Facebook”. Там освен възможност за споделяне на снимки, може също да се възползвате от опция като „чат“, търсене на други потребители, коментиране на снимки и други.

Недостатък е достъп до личната Ви информация.



Фиг. 2 Лого на “Flickr.com”

Основната функция на “Flickr” е споделяне и разглеждане на снимки. Сайтът има изключително приятен дизайн и е лесен за използване. Може да бъде използван от таблети и мобилни устройства, като дизайна му е такъв, че се наглася според това от какво устройство е използван.



Фиг.3 Лого на „Picasa.com“

Друг също често използван сайт за споделяне на снимки е “Picasa”. Той също разполага с приятен дизайн, като основната му функция е запазване на снимки. Това с което се отличава този сайт е, че е фокусиран само към споделяне на снимки и това прави сайта прост като функционалност, което е перфектно за потребител, който иска да използва сайт само за качване на снимки.

## 1.3. Сравнение на направения сайт с друг такъв

Нека сега разгледаме предимства и недостатъци на направения сайт с друг подобен такъв.

Сайтове които предоставят възможността за споделяне на снимки са много, затова ние ще се спрем само на един, който може би е най-популярния напоследък, а именно “<https://www.facebook.com/>“.

Нека първо разгледаме “Facebook”

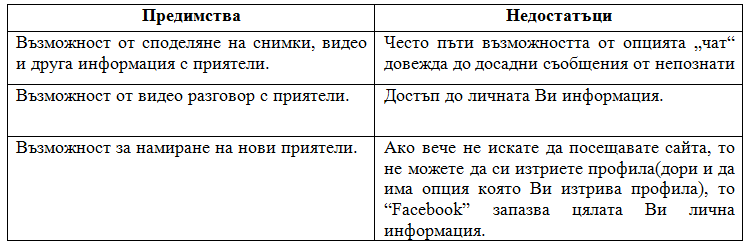


Табл.1 Предимста и недостатъци на “Facebook”

Нека сега разгледаме предимствата на сайта предложен в този документ.

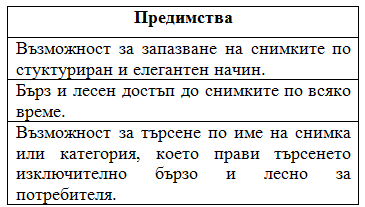


Табл.2 Предимствата на „Фотоалбум“

Като цяло къде да споделяте вашите снимки или друга информация, изключително много зависи от това какви нужди имате. Ако искате сайт в който просто да складирате вашите снимки без да се безпокоите от това, личната Ви информация да бъде запазвана, без да искате да търсите нови приятели, то тогава проекта „Фотоалбум“ е перфектен за тази цел. От друга страна , ако искате да се запознавате с нови хора, да си споделяте и коментирате снимките, то „Facebook“ е по-добра опция.

## 1.4. WEB приложения

### 1.4.1. **WEB сървър и WEB браузър**

Мрежовите приложения се състоят от сървър и клиент. Когато разглеждаме WEB приложения, това са WEB сървър и WEB браузър, като те са свързани чрез Интернет или Интранет мрежа. Стандартният протокол за предаване на данни между тях е създаденият през 1991 година Hypertext Transfer Protocol (HTTP), а стандартният тип документ – Hypertext Markup Language (HTML). Важна особеност на HTTP е, че не съдържа контрол на сесиите между участниците и за всяка заявка се създава нова сесия, осъществява се транзакция от клиента към сървъра, отговор от сървъра и се затваря сесията.

### 1.4.2. WEB сайт

WEB сайтът представлява съвкупност от страници, кодирани по начин подходящ за разпространение и бързо възпроизвеждане. Отделните страници или документи се свързват посредством хипервръзки (линкове). Можем да различим два основни вида WEB сайтове – статични и динамични.

#### 1.4.2.1. Статични WEB сайтове

Статичните WEB сайтове са съвкупност от прости документи, директно кодирани във формат HTML (Hypertext Meta Language). Поправката на тези сайтове става чрез директно променяне на кода на страницата. Те могат да извеждат само предварително въведена статична информация и не могат да изпълняват логически действия. При тях потребителят посочва адрес, който желае да бъде визуализиран в своя WEB браузър (това може да е както IP адрес 127.0.0.1, така и домейн–име като [www.mozila.org](http://www.mozila.org)) и се изпраща GET заявка по HTTP на стандартният 80 порт към избраният сървър. Повечето WEB сървъри са конфигурирани така, че да улеснят потребителите и при заявка към директория да върнат определен файл от нея, вместо лист със съдържаните файлове. За разглежданите статични WEB приложения това най-често е файл с име index или main и разширение html или htm. Ако потребителят поиска конкретен файл, сървърът го изпраща обратно към WEB браузъра. След като получи файла, WEB браузърът го анализира и визуализира. Характерно за статичните WEB приложения е, че всички заявки към един WEB адрес връщат един и същ резултат, и не позволяват на потребителите да модифицират данните разположени на сървъра като използват само WEB приложението.

#### 1.4.2.2. Динамични WEB сайтове

Динамичните WEB сайтове позволяват на потребителя да взаимодейства с тях като въвежда и модифицира данни. Тази функционалност се развива върху основата на приложение, което WEB сървърът използва при генериране на приета заявка от потребителя и в зависимост от дадени параметри в нея, генерира валиден HTML документ. При динамичните приложения потребителят може да се обърне директно към файл, съдържащ изпълним код (JAVA, PHP) или друг път, който сървърът да пренасочи до желаното действие. Сървърът, вместо да предаде съдържанието на файла, както е при статичните сайтове, го предава на модул, който го изпълнява и връща резултата от изпълнението към клиента.

Като динамични WEB сайтове е реализирана и дипломната ми работа. Така са реализирани и много други динамични сайтове, за които идентификацията и поддържането на сесия е изключително важно. Това се поема отново от приложението, работещо на сървъра, като на клиента освен заявената страница се изпраща и 'бисквитка' (cookie). Тя се използва за идентификация и следене на действията на даден потребител по време на отделната сесия на WEB сайта. Това ни позволява лесно да осигурим влизане на потребителя и извеждане на съдържание, съгласно неговото ниво на оторизация или лични предпочитания.

Друга особеност на съвременните WEB приложения е, че за съхранение на динамичната информация се използват релационни бази данни.

Върху основата на компоненти като: сървър, WEB приложение и база данни могат да се изградят редица WEB приложения. За всички тях е характерно, че WEB браузърът се използва само за визуализация на WEB страници. Така можем да разглеждаме WEB браузъра като тънък клиент (“thin client”), не съдържащ никаква логика. Това означава, че за всяка модификация на визуализираното съдържание е необходима нова заявка към WEB сървъра, както и, че програмистът няма никаква възможност за задаване на функционалност върху елементите в WEB страницата. Със създаването на JAVA, PHP и други програмни езици се предоставя възможност за задаване на функционалности в рамките на WEB страницата.

Статичните WEB сайтове намират приложение най-често в рекламни и информативни сайтове, където няма чести актуализации и сложна функционалност. Динамичните WEB сайтове се използват във всички области, където има честа актуализация и диалог с потребителя.

### 1.4.3. HTTP

HTTP (Hypertext Transfer Protocol) е комуникационен протокол от приложното ниво в OSI модела, който се използва за предаване на информация през World Wide Web. Първоначално е бил развит като стандартен метод за публикуване на HTML страници.

При HTTP разглеждаме заявка и отговор между клиент и сървър, където клиентът често е WEB браузър, или друго потребителско приложение, а другият участник – сървърът често създава WEB страниците и изпраща необходимите ресурси към тях като картинки и други файлове.

Към HTTP съществуват 8 различни типа заявки (HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT) като при разработка на WEB приложения най-често се използват GET и POST заявки, тъй като е възможно или клиентът, или сървърът да не поддържат останалите. В рамките на разработеното приложение са използвани само тези два типа заявки. За по-добра яснота от това какво е HTTP, можем да погледнем [4].

## 1.5. Цел на дипломната работа

След като направихме сравнения с друг подобен сайт и описахме основните функции които ще предоставя, вече можем по-точно да дефинираме целта на настоящата дипломна работа.

Целта на дипломната работа е създаване на софтуерна система която да предоставя възможността за споделяне на изображения. Основните функции които ще предоставя софтуерната система са:

* Управление на потребители и техните профили
* Управление на категории
* Управление на снимки

# Глава 2 - Теоретично решение на поставената задача.

## 2.1. Преглед на средствата за програмна реализация

### 2.1.1. HTML

#### 2.1.1.1. Същност

HTML (Hypertext Markup Language) е основният език за създаване на WEB страници. Чрез него се описва структурата на текстови документи като отделни сегменти от него се визуализират по различен начин. Определянето на различните сегменти се извършва чрез етикети, често наричани тагове като за начало на таг се използва '<' , а за край '>'. Чрез HTML също до определена степен може да се опише семантиката на документа. HTML позволява внедряването на допълнителни мултимедийни елементи и скриптове .

Като история на HTML можем да посочим създадената през 1980 година от Тим Бърнърс Лий система ENQUIRE за споделяне на документи в CERN. Първата спецификация на HTML е публикувана през 1991 година като днес е популярна 5.0 версията на стандарта. Характерно за развитието му е това, че се увеличава изискването за валидност на подредбата на елементите в документа. В ранните години на стандарта, неправилна подредба е довела до усложняване на браузърите и до вариации при визуализацията им.

За да се използва дефинираната като стандарт DTD (Document Type Definition) и да не се използва режима Quirks Mode на браузъра, HTML документът трябва да започне с валидна декларация на типа документ (Document Type Declaration -“DOCTYPE”).

#### 2.1.1.2. Структура на HTML документ

Структурата на HTML документите се изгражда на основата на елементи. Те имат две основни свойства – атрибути и съдържание. Атрибутите и съдържанието са различни според дефиницията на HTML DTD. Всеки елемент има начало и край – например <table> и </table>, а други елементи като <br/> и <hr/> нямат съдържание и затова не са им необходими начален и краен таг.

Всеки HTML документ дефинира елемент <html> на най-високо ниво. В този елемент се разполагат елементите <head> и <body>. В елемента <head> поставяме информация за връзки като CSS дефиниции, JavaScript файлове и други. В <body> елемента се намира съдържанието на HTML документа. Така празен и валиден html документ би имал вида:

<html>

<head>

</head>

<body>

</body>

</html>

На допълнителни детайли за HTML елементи и създаването на HTML документи няма да се спираме. Все пак допълнителни детайли около HTML5, който е последната версия на HTML можем да видим в [7].

### **2.1.2. CSS**

#### 2.1.2.1. Същност

Каскадните стилови таблици (Cascading Style Sheets или CSS) се използват за определяне визуализацията на структуриран документ като най-често служат за изграждане на дизайна на WEB страници, написани на HTML. Самата технология позволява изграждане на дизайна на произволен XML документ чрез XLST технология и не е ограничена само до WEB технологии.

Едно от големите предимства на CSS е, че позволява да се разделя презентацията на документ от неговото съдържание. Това води до лесна поддръжка на дизайна на много HTML страници. Друго предимство е, че може да се използват едновременно от авторите и потребителите на WEB страници чрез подходящи настройки на WEB браузъра. Много добри препоръки за най-добре използване на CSS код е даден в [5].

#### ***2.1.2.2. Синтаксис***

Синтаксисът на CSS е изключително лесен. Всеки CSS съдържа списък от правила за форматиране като се определя селектор на елементи и атрибути, които трябва да му бъдат приложени. Ще разгледаме синтаксиса в следните примери:

TABLE.title

{

COLOR: #993366; FONT: 18px verdana, arial, shans-serif; font-weight: bold;

}

Фиг.4 Синтаксис на CSS

Създава се клас, който се наследява от всички блокови елементи, намиращи се вътре в 'ТABLE' елементa, на които е зададено 'class = title’ и се правят съответните настройки. Той определя заглавието да е оцветено по определен начин (#993366), големината на шрифта да е 18 пиксела, а вида му да е verdana или arial или shans-serif, удебелен.

TABLE.dividers

{

width: 100%; color: #FFFFFF; height: 3px; border: 1px solid #CC66CC;

}

Фиг.5 Продължение на фиг.4

Създава се клас, който определя всички разделителни черти да с широчина 100%, цвят на буквите -#FFFFFF; височина-3 пиксела; рамка 1 пиксел, която е плътна и е с цвят #CC66CC.

TABLE.just

{

text-align: justify;

}

Фиг.6 Продължение на фиг.5

Създава се клас, който определя двустранното подравняване на елементи.

Каскадността на стиловите таблици се изразява в това, че към един елемент се записват всички атрибути, които се напасват с него.

Стиловите таблици се добавят към документ чрез следната примерна конструкция:

<link rel="stylesheet" href="css/mainstyle.css" type="text/css">.

Важна възможност на стиловите таблици е дефинираното им използване само при печат на WEB страници или при визуализиране чрез други устройства.

### **2.1.3. Същност на Javascript**

**JavaScript** е интерпретиран език за програмиране, разпространяван с повечето Уеб браузъри. Поддържа обектно-ориентиран и функционален стил на програмиране. Създаден е в Netscape през 1995-та. Най-често се прилага към HTML -а на Интернет страница с цел добавяне на функционалност и зареждане на данни. Може да се ползва също за писане на сървърни скриптове NodeJS, както и за много други приложения. JavaScript не трябва да се бърка с Java, съвпадението на имената е резултат от маркетингово решение на Netscape. Javascript е стандартизиран под името EcmaScript.

JavaScript е програмен език, който позволява динамична промяна на поведението на браузъра в рамките на дадена HTML страницата. JavaScript се зарежда, интерпретира и изпълнява от уеб браузъра, който му осигурява достъп до Обектният модел на браузъра. JavaScript функции могат да се свържат със събития на страницата (например: движение/натискане на мишката, клавиатурата или елемент от страницата и други потребителски действия). Javascript е най-широко разпространеният език за програмиране в Интернет. Прието е JavaScript програмите да се наричат скриптове. Изключително полезно и ефикасно е да се използват препоръките за писане на код на JavaScript, които са дадени в [2].

#### ***2.1.3.1. Разлики с Java***

Освен съвпадението в част от името, двата езика нямат кой знае какви прилики, дори са разработени от различни корпорации ( Java е дело на SUN, a JavaScript е разработка на Netscape). Java е популярен език за програмиране не само на Интернет приложения, но и на самостоятелни програми за различни платформи. Интернет приложенията на Java се наричат аплети. Те са файлове с разширение .class и се вмъкват в HTML документа между таговете <APPLET> и</APPLET>. Тук няма да се спираме подробно на Java аплетите.

#### 2.1.3.2. Възможности на JavaScript

JavaScript може да влияе на почти всяка част от браузъра. Браузъра изпълнява JavaScript кода в цикъла на събития т.е. като резултат от действия на потребителя или събития в браузъра. Основни задачи в повечето JavaScript приложения са:

* Зареждане на данни чрез AJAX (ще разгледаме AJAX след малко)
* Ефекти с изображения и HTML елементи: скриване/показване, пренареждане, влачене, слайд шоу, анимация и много други.
* Управление на прозорци и рамки.
* Разпознаване на възможностите на браузъра.
* Използване на камерата и микрофона.
* Създаване на 3D графики WebGL.
* По-добър и гъвкав потребителски интерфейс

Какво не може да се прави с помощта на JavaScript :

* Не може да се записва информация на потребителския компютър или отдалечения сървър.
* Не може да се запазва информация директно в отдалечена база данни.
* Не може да се стартират локални приложения.

#### ***2.1.3.3. Интерпретиране***

JavaScript кодът може да се вмъква в HTML документа между двойката елементи <SCRIPT> и</SCRIPT>. Когато срещне тага <SCRIPT>, браузърът разбира, че трябва да спре интерпретирането на HTML кода и да започне да обработва скрипта, намиращ се между <SCRIPT> и </SCRIPT>. Този скрипт не е задължително да бъде написан на JavaScript. Има и други езици за писане на скриптове, например VBScript. Но езикът по подразбиране е JavaScript. В следния пример създаваме уеб страница в която е вмъкнат JavaScript код, който да изписва на екрана "Здравей!"

<!DOCTYPE html>

<HTML>

<HEAD>

<TITLE>javascript</TITLE>

</HEAD>

<BODY>

<SCRIPT>

document.write(“Здравей!”);

</SCRIPT>

</BODY>

</HTML>

Фиг.7 Пример за валиден JavaScript код

#### 2.1.3.4. Предимства и недостатъци на Ajax

**Ajax**  е похват в уеб разработките за създаване на интерактивни уеб приложения. Предимството на ***Ajax*** е, че посредством използването му уеб страните се зареждат по-бързо. Посредством асинхронен обмен на малки порции данни „зад кадър“ могат да се променят само частично информации на уеб страницата. По този начин се намалява количеството информация, която се трансферира между сървъра и клиента. Асинхронността позволява да не бъде необходимо да се презарежда цялата страница отново. По този начин се повишава интерактивността, скоростта и функционалността на страниците.

Сам по себе си ***Ajax*** не е технология, а термин за употребата на група технологии. За първи път се споменава през февруари 2005 г. от Джеси Джеймс Гарет в книгата му [3].

Предимства

* Няма нужда от презареждане на страницата .
* Времената за отговор се скъсяват.
* Броят заявки към сървъра намалява, защото скриптовете и *CSS*-файловете се зареждат само веднъж.
* Посредством променливи в *JavaScript* може да се запази състоянието на програмата

Недостатъци

* Не се поддържа от стари уеб браузъри, както и от някои мобилни телефони
* При натискане на бутона за връщане назад на браузъра приложението не се връща в предишното състояние, а в състоянието при зареждането му
* При запазване на отметка в браузъра се запазва отметка към приложението в началното му състояние
* Повечето търсачки не изпълняват код на *JavaScript* , поради което не цялото съдържание на сайтове с *Ajax* се индексира

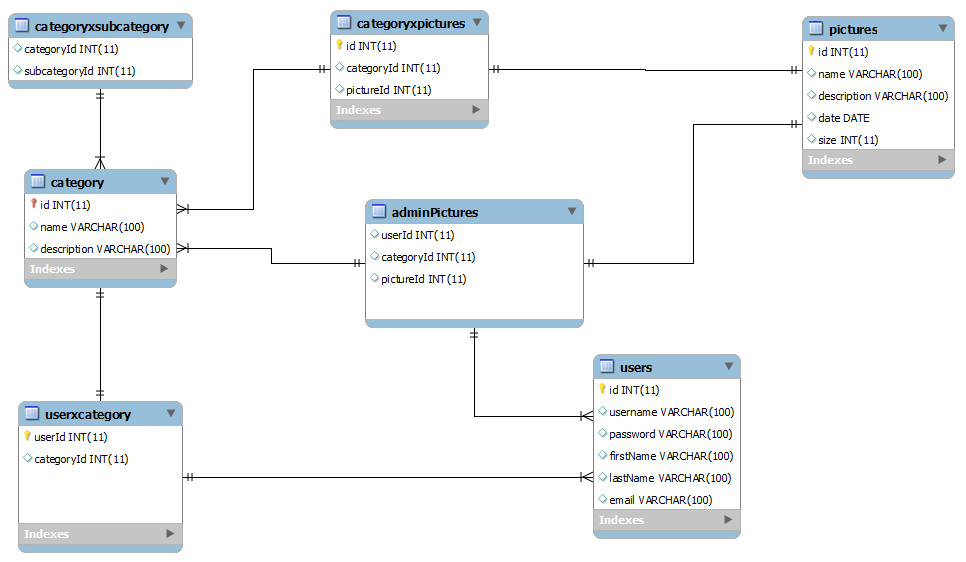
## **2.2. Релационни бази от данни**

Повечето съвременни информационни системи използват релационни бази от данни (RDB) за съхранение на огромния обем от информация.

Системите за управление на релационни бази от данни (RDBMS) предоставят един унифициран механизъм за бърз достъп към избрани части от данните в базата. Релационният модел на данни се базира върху идеята за представяне на данните в таблична форма. Схемата на една таблица се състои от името на таблица и множество от именувани колони. Имената на колоните се наричат променливи или атрибути. Действителната таблица (един екземпляр на схемата), наричана още релация, представлява едно именувано множество редове. Всеки елемент на една таблица съдържа в колонката за атрибута А някаква стойност от дефиниционната област 15 Dom(A) на колоната А. Един атрибут може да бъде произволен тип – цели или реални числа, низове, дати и т.н. Подредбата на редовете и колоните в таблицата не е от значение.

Данните във всяка клетка на една таблица обикновено са атомарни стойности, т.е. не е прието използването на списък от стойности за една клетка. Това означава, че ако искаме да представим информация за хора, тяхната възраст и телефонни номера, ние не можем да пазим в един атрибут няколко телефонни номера на един и същ човек. Ограниченият по този начин модел има така наречената първа нормална форма. При прокетиране на „Фотоалбум“ от настоящата дипломна работа ще считаме, че базата от данни, от която ще се правят извадки, отговаря за първа нормална форма.

Дори в относително малки организации, използваните релационни бази от данни могат да имат стотици таблица и хиляди атрибути. По тази причина, управлението на схемата на базата от данни може да се окаже една доста сложна задача. Понякога се твърди, че за целите на анализа на данни е достатъчно всички тези таблици да бъдат комбинирани в една масивна таблица с наблюдения или „универсална таблица“, като по този начин ИЗД анализаторът не трябва да се грижи, че данните се намират някъде в базата от данни. Обаче, проверката на тази теза дори върху прости примери показва, че това не е осъществимо: универсална таблица ще бъде толкова голяма, че операциите върху нея ще бъдат извънредно скъпи.

Ето как изглежда нашата база от данни и връзките които са изградени между таблиците:

Фиг.8 Диаграма на базата от данни и връзките между таблиците

### 2.2.1. Език на структурирани заявки (MySQL)

За управлението на данните не е достатъчно само да можем да описваме структурата им и да ги съхраняваме с използване на тази структура. Трябва ни още възможност да извличаме данни по желание от създадената база. Ще разгледаме накратко езика на структурните заявки (MySQL), който предлага удобен механизъм за извличане на данни от базата, като използва логически условия за дефиниране на критерии за търсене. В системите за управление на бази от данни MySQL е стандартът, приет от повечето производители на подобни системи. MySQL реализира едно надмножество от релационна алгебра, като по този начин осигурява една удобна и компактна нотация за боравене с множества от данни. В настоящия раздел ще разгледаме само базовата структура на MySQL програмите.

Базовата конструкция на SQL е израз или заявка от тип “select-from-where” (избери-от-където), която има обикновено следната най-проста форма:



Фиг.9 Базова конструкция от тип на „select-from-where“

Тук всяка ri е таблица, а всеки Aj е атрибут. Интуитивното значение е, че за всеки възможен избор на редове t1, …, tk от таблиците r1, r2, …, rk ние проверяваме, дали условията са верни. Ако да, то редът, състоящ се от стойностите на атрибутите Aj , се връща в резултата.

Вторият ред на заявката - клаузата **from** - описва таблиците, към които тази SQL конструкция трябва да бъде приложена. Третият ред, съдържащ клаузата **where**, описва условията, на които редовете на тези таблици трябва да отговарят, за да бъдат включени в резултата. Първият ред - клаузата **select** - определя кои атрибути от указаните таблици трябва да фигурират в резултата. Тя съответства на операцията проектиране (π) от релационната алгебра (не на операцията за избор - σ). Клаузата **where** се използва за представяне на условията за избор, участващи в операциите избор и съединение. За операцията избор условията за избор са представени като списък от условия в клаузата **where**, разделени от такива ключови думи като **and**, **or** или **not**. Например, избор на всички потребители от таблицата с име “Users”, на които първото име е „Тодор“, може да бъда направен чрез заявката:



Фиг.10 Пример за взимане на всички потребители, на които първото им име е „Тодор“

За да се „гмурнем“ по - надълбоко и да разучим по – задълбочено MySQL, [10] би ни свършила чудесна работа.

## 

## 2.3. Езици за създаване на динамични WEB – сайтове( предимства и недостатъци)

### **2.3.1 JAVA**

В днешно време Java е един от най-популярните езици за програмиране. На него пишат милиони разработчици по цял свят. Най-големите световни софтуерни корпорации като IBM, Oracle, Google и VMware базират своите решения на Java платформата и използват Java като основен език за разработка на своите продукти.

Езикът Java е първоначално разработен и поддържан от Sun Microsystems, но през 2006 г. Java платформата става проект с отворен код и в момента се поддържа и развива от световната Java общност съвместно със Sun. Благодарение на отворения код популярността на Java постоянно се увеличава и броят Java разработчици непрекъснато расте.

Езикът Java се разпространява заедно със специална среда, върху която се изпълнява, наречена Java Runtime Environment (JRE). Тази среда включва т. нар. Java виртуална машина (JVM) и пакет стандартни библиотеки, предоставящи базова функционалност. Благодарение на нея Java програмите са преносими и след като веднъж бъдат написани, след това почти без промени могат да работят на стотици хардуерни платформи и операционни системи.

Java  е обектно-ориентиран език за програмиране. Кодът, написан на Java не се компилира до машинен код за определен процесор, а до специфичен за езика код, наречен байт код. Поради това за изпълнението на програма, написана на Java е необходима т. нар. виртуална машина.

Подобна реализация има своите предимства и недостатъци. Сред главните предимства са:

* лесната поносимост между различните платформи (софтуерни или хардуерни) - веднъж написана и компилирана, една Java-програма може да бъде стартирана на компютри независимо от архитектурата или от операционната им система. За целта е необходимо само да се инсталира виртуална машина за съответната платформа;
* допълнителните действия, извършвани от виртуалната машина като освобождаване на паметта от обекти(класове) които не се използват(Garbage collector), проверка за размерността на масивите;
* възможността за контрол на правата на потребителя на ниво виртуална машина
* първоначално заделяна на heap, част от паметта резервирана за джава
* висока степен на сигурност поради факта, че програмистите не работят директно с паметта.

Главен недостатък е необходимостта от допълнителни ресурси (под формата на процесорно време и памет) за изпълнението на самата виртуална машина.

Съществуват и компилатори, които превеждат байт кода до машинен код. Недостатъка при използването им е, че тогава програмата не може да се стартира на различни платформи.

Ранните версии на виртуалните машини страдат от слаба производителност поради необходимостта от компилиране на байт кода до машинен код. Този проблем е решен в значителна степен след въвеждането на JIT компилатора във виртуалната машина.

В по – сериозни детайли няма да навлизаме. Книги за JAVA има изключително много. Лично аз смятам, че най – полезната книга, която слага основите на JAVA e [1]. Обхваща най – важните неща и са обяснени изключително подробно.

### **2.3.2 Същност на JSP**

*Java базирани Web-приложения*

Едно Java базирано Web-приложение представлява съвкупност от сървлети, JSP страници (разширение на сървлетите, което позволява в HTML документи да се вгражда Java код), Java архиви и други файлове, които заедно изграждат една обща Web-базирана система.

*Производителността на сървлети и JSP*

От гледна точка на ефективността сървлетите и JSP-тата превъзхождат вече остарялата CGI технология, защото изпълнението на сървлет (или JSP) не води до създаване на нов процес в операционната система, което е традиционно бавна операция. Java виртуалната машина стои постоянно заредена в паметта и когато се извика някой сървлет, той просто се изпълнява от нея и резултатът се връща на клиента, без да се създава нов процес за обработка на заявката.

Веднъж изпълнен, сървлетът остава в компилиран вид активен в паметта, чакайки ново извикване. Това значително повишава производителността, защото не е необходимо при всяка заявка сървлетът да се чете от файловата система и да се инстанцира неговият клас. Когато едновременно постъпят няколко заявки за един и същ сървлет, той се изпълнява едновременно в няколко нишки(threads), за което се грижи сървъра.

Всъщност по подобен на сървлетите начин работят всички съвременни Web-технологии като Perl, PHP, ASP и ASP.NET. При тези технологии обикновено интерпретаторът на съответния скрипт език стои като модул зареден постоянно в паметта на Web-сървъра и се включва при клиентска заявка. За по-голяма ефективност често се прилага и друг подход – не се използва интерпретатор, а компилатор, който компилира кода до някакъв междинен код и виртуална машина, която изпълнява този междинен код.

*Средствата на J2EE за изграждане на Web-приложения*

Съвкупността от средства за работа със сървлети и JSP-та, който ни дава J2ЕЕ платформата предлага доста вградени удобства. Едно от тях е вече споменатото автоматично управление на многопотребителския достъп, което дава възможност за съхранение на различни данни за потребителя в рамките на неговата сесия. Освен това извличането и декодирането на параметрите е силно улеснено, дава се възможност за достъп на по-високо ниво до HTTP хедърите, за пренасочване на потребителския браузър, за достъп до общи за приложението данни, за обмяна на данни между приложенията, както и достъп до ресурси, принадлежащи на Web-контейнера (сървърът, изпълняващ сървлетите и JSP-тата).

*Преносимост на Java-базираните Web-приложения*

Java базираните Web-приложения са лесно преносими. В повечето случаи всичко, което е необходимо за да се прехвърли едно приложение от един сървър на друг, е да се прехвърли един единствен файл. Дори новият сървър да е от друг производител и да работи на друга операционна система, рядко се налага да се извършват промени или допълнителни настройки за да заработи Web-приложението на новия сървър. Тази изключително добра съвместимост се дължи на стандартите за Web-приложения, които се дават от платформата J2EE и се спазват стриктно от почти всички производители на Web-приложения и Web-контейнери.

*Сигурност и надеждност на Java-базираните Web-приложения*

Сигурността и надеждността на Java базираните Web-приложения е завидно добра. Това се дължи най-вече на надеждността и сигурността на самата Java платформа. Уязвимости като препълване на буфери са изключени, а проблеми с лошо декодиране на данни, неправилна работа с базата данни, непозволен достъп до паметта и още много други при Java платформата са много по-рядко срещани, отколкото при други платформи.

На практика тенденцията е повечето езици за програмиране с общо предназначение да преминават към архитектури, близки до архитектурата на Java. Компютрите станаха толкова бързи, че вече използването на виртуална машина, интерпретатор или някаква друга среда, която управлява изпълнението на кода и не му позволява да прави „лоши неща” се е наложило като стандартен модел за изпълнение на програмния код. Такъв е моделът на изпълнение и в Java и в .NET и в Perl и в PHP и в други платформи.

Езиците за програмиране в платформите Java и .NET имат и други предимства по отношение на сигурността и надеждността – те са силно типизирани, а това не позволява възникване на проблеми с неправилно използване на указатели. В тези езици има и автоматично управление на паметта и ресурсите, а това силно ограничава проблемите със сигурността и надеждността, свързани с неправилно заделяне, освобождаване на памет и управление на ресурси.

*Java-базираните Web-приложения и свободния софтуер*

Още едно важно предимство на Java сървлетите, JSP страниците и Java-базираните Web-приложения е, че те могат да се използват напълно безплатно, включително и в комерсиални проекти. За работата им не е необходимо закупуването на скъп сървърски софтуер, защото има достатъчно добри безплатни Web-контейнери и J2EE сървъри за приложения (application servers), идеални за малкия и средния бизнес. Примери за такива безплатни сървъри са Web-контейнерът Tomcat (http://jakarta.apache.org/tomcat/), който ще разгледаме след малко и сървърът JBoss (http://www.jboss.org), който е пълна имплементация на J2EE спецификацията. Допълнителни детайли за JSP-тата можем да намерим в [8].

### **2.3.3. Същност на Java Servlet**

Сървлета е програмен клас на Java, използван за разширяване на възможностите на даден сървър.Въпреки че сървлетите могат да отговорят на всякакви видове заявки, те често са използвани за разширяване на приложенията, хоствани от уеб съврър, така че те могат да се мислят и като Java аплети, които работят на съвръри, вместо на уеб браузъри. Тези видове сървлети са копие на Java, както и на други Уеб динамични технологии като PHP и ASP.NET. Основни познания по Java Servlet можем да придобием от [6].

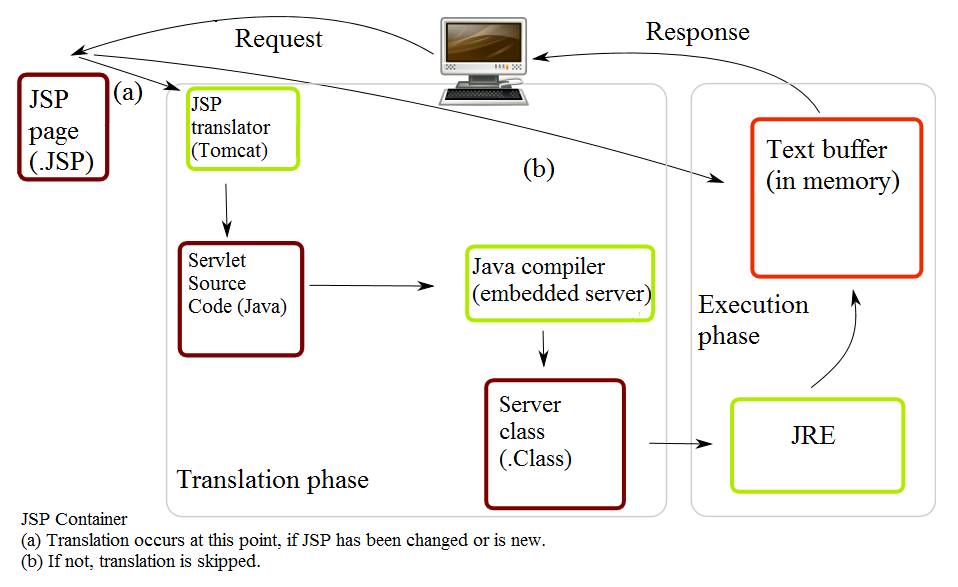
*Сървлетите се използват най-често за:*

- обработват или съхраняват данни, които са подадени през HTML форма.

- Осигуряване на динамично съдържание, пример за това са резултатите от заявка към база от данни.

*Жизнен цикъл на сървлета*

Три са основните методи на сървлета.Това са init(), service(), и destroy(). Те са имплементирани от от всеки сървлет и се извикват в специфично време от сървъра.



Фиг.11 Жизнен цикъл на сървлета

Като бъде изпълнен за първи път JSP файла, JSP кода(който е подебен на HTML кода) се конвертира до Java код, който от своя страна е съвместим с Java Serlvet-а. След това се компилира до файла на класът и се изпълнява Java сървъра(в случая Tomcat) като Java Servlet. По този начин JSP-тата се изпълняват като java servlet-и, въпреки че кода им е много близък до този на html-a.

## 

## 2. 4. Етапи при работа с динамичен сайт

Работа с динамичен сайт, създаден на JAVA се осъществява със следните стъпки:

- Заявка от клиента (потребителя) към сървъра. Тази заявка може да бъде показване на определена страница, извадка от данни или оторизация на потребителя.

- Сървърът от своя страна проверява заявката дали може да бъде изпълнена директно, без използването на база данни. Ако може – се изпълняват само определени функции, незасягащи данните, ако ли не – е необходимо:

- Обръщане към базата данни и изпълнение на необходимите заявки за получаването на филтрирана информация, която отново се връща на сървъра.

- Получената информация се филтрира и обработва спрямо кода на WEB сайта. След изпълнение на всички функции е генериран HTML файл, който има формата на единична страница.

- Получения файл се изпраща на клиента и се остава в режим на чакане на следващото действие.

Схемата на действие е дадена на следната фигура:



Фиг.12 Етапи при работа с динамичен сайт

## 2.5. Избор на програмни средства

На базата на изложените досега предимства и недостатъци на различните езици за създаване на динамични WEB сайтове моят избор се спря на JAVA. Главните причините, поради които се спрях на него са:

* защото той е един от най-разпространените езици в Интернет пространството;
* предлага всички възможности за създаване на динамичен съвременен сайт;
* защото е безплатен;
* езикът не зависи от операционната система;
* за разлика от много други скрипт езици, JAVA се разпространява с вграден компилатор, който компилира кода на JAVA и може да открива грешки в него;
* друго значително предимство на JAVA е неговата преносимост, JAVA е съвместим с произволни комбинации от софтуери това е, което го прави преносим;
* може да работи с почти всяка комбинация от операционна система, WEB сървър и сървър за база данни;

Избраната база данни е MySQL, тъй като тя има ясно структурирани заявки, инсталацията е лесна, съвместима е с много други приложения и езици за програмиране, както за WEB така и за десктоп програмиране и е безплатна.

## **2.6. Функции които се предоставят пред потребителя**

Потребителите имат няколко възможности, след като са успяли да влязат успешно в сайта.Техните възможности са:

*-добавяне на снимка*

*-изтриване на снимка*

*-добавяне на категория*

*-изтриване на категория*

*-промяна на парола*

*-промяна на профил*

*-изтриване на профил*

*-търсене на снимка и категория*

**Функция за въвеждане на снимки:**

От дясната страна има меню с опции от които едната е „Добави нова снимка“, при което потребителя се изпраща на нова страница на която има форма за избор на файл и съответно полета за име на снимка и описание на снимка.След като потребителя е готов с изискваната информация, има бутон „Запази“, при което снимката не се показва на потребителя, а отива в секцията на администратора за одобряване.Чак след като администратора одобри снимката, тя ще бъде показана на потребителя.Както вече обяснихме по-рано, идеята администратора да одобрява снимките е направена като мярка за сигурност, за да не бъдат качвани снимки с неподходящо съдържание.

**Функция за изтриване на снимки:**

Когато потребителя отиде с мишката върху снимката, в горния ляв ъгъл се появява едно „кошче“ което се използва за изтриване на снимките.След като потребителя щракне върху него, му се показва съобщение дали наистина желае да изтрие снимката.Ако потребителя натисне „Да“, то снимката се изтрива от сайта и от базата данни, при отказ потребителя се връща в последното си състояние.

**Функция за добавяне на категория:**

От дясната страна на сайта има меню с опции, от които едната е „Добави нова категория“, при което потребителя бива изпратен на нова страница с форма, която съдържа „Име на категорията“ и „Описание на категорията“. След като потребителят попълни формата, има бутон за запазване на информацията, при което се показва категория с въведените от потребителя име и описание.

**Функция за изтриване на категория:**

От дясната страна на сайта има меню с опции, от които едната е „Изтрий текущата категория“, при което ако потребителя я натисне, то той бива питан дали наистина желае да изтрие текущата категория, ако той натисне „Да“, то цялата информация в категорията ще бъде изтрита.Това означава, че ще бъдат изтрити всички снимки, категории и подкатегории в текущата категория.При отказ потребителя бива върнат в последното си състояние.

***Забележка:*** Важно е да се отбележи, че потребителят има право да изтрие всички категории без “root” категорията(главната категория), която се създава автоматично при създаването на потребителя.

**Функция за промяна на профил:**

От дясната страна на сайта има меню с опции, от които едната е „Промяна на профил“, който изпраща потребителя до нова страница с форма която да попълни.В тази форма има полета за промяна на име, фамилия и имейл, при което ако потребителят въведе нова информация за някое от тези полета(или всички) и натисне бутона „Запази“, то информацията ще бъде актуализирана.При отказ потребителя бива върнат в последното си състояние.

Може би Ви прави впечатление, че липсват полета за промяна на “username”(потребител) и парола.Това е така, защото е забранена промяната на “username”, тъй като той се води уникален.Колкото до промяната на парола, вече я обяснихме малко по-нагоре в секцията за „Смяна на парола“

**Функция за търсене:**

Търсенето на снимки и категории става по име.В горния десен ъгъл на сайта има поле за търсене, в което ако потребителя напише някакъв текст и натисне бутона „Enter“, то ще бъдат изобразени всички категории и снимки на които името им съдържа текста който е въведен в полето за търсене(или нищо ако няма снимка или категория, която да съдържа въведения текст)

## 2.7. UML Диаграми

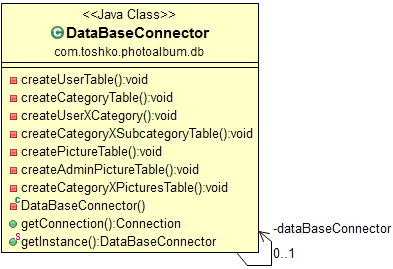
В дипломната работа съм използвал MVC pattern, който е разделен на 3 отделни части: Model, View и Controller.

Model – това е ядрото на приложението, предопределено от областта, за която се разработва. Обикновено това са данните от реалния свят, които сме моделирали и над които искаме да работим - да въвеждаме, променяме, визуализираме и т.н.Трябва да се направи разлика между реалния обкръжаващ ни свят и въображаемия абстрактен моделен свят, който е продукт на нашият разум.

View – това е тази част от изходния код на приложението, отговорна за показването на данните от модела. Например, изгледът може да се състои от PHP шаблонни класове, JSP страници,  ASP страници, JFrame наследници в Swing приложение. Зависи от това какъв графичен интерфейс се прави и каква платформа се използва.

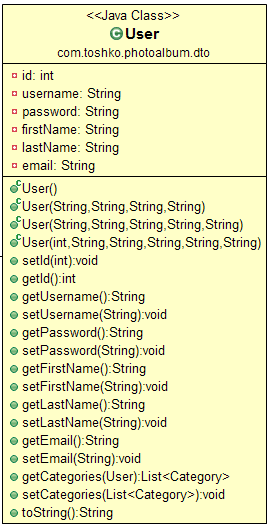
Controller - тази част от сорс кода(клас или библиотека), която взима данните от модела или извиква допълнителни методи върху модела, предварително обработва данните и чак след това ги дава на View частта. Също така когато се прави уеб графичен интерфейс това би довело до много лесна модификация на HTML кода дори от човек, който не е програмист - той ще гледа на шаблона просто като на обикновена HTML страница.

По – долу сме показали част от най-важните класове в проекта изобразени чрез UML диаграми.



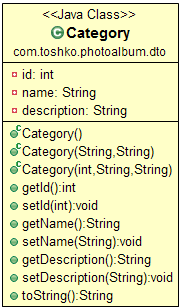
Фиг.13 Диаграма на класа “DataBaseConnector”

Класът „DataBaseConnector“ е направен по примера на “Singleton”. Може да имаме само една инстанция на този клас(което се вижда и от диаграмата). В него имаме 8 “private” метода, които отговарят за създаването на всяка една от таблиците в базата от данни и 2 публични. Първият публичен метод, който е “getConnection()” отговаря за вземане на връзка към базата от данни, а „getInstance()“ служи за вземане на инстанция на класа.



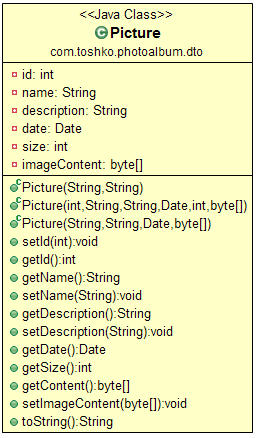
Фиг.14 Диаграма на класа „User“

В класът „User“ пазим информация за всеки потребител. Има 4 конструктора, които приемат различни параметри, според това как е използван класа „User“, в останалите класове. Останалата информация която имаме в този клас са методи, които служат за слагане и съответно взимане на „id“, „потребител“, „парола“, “първо име”, „фамилно име“ и „имейл“, които се отнасят пряко до потребителя. Има и 2 метода („setCategories“, “getCategories”), които служат за слагане и вземане на всички категории за даден потребител.



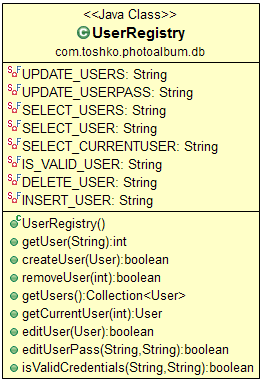
Фиг.15 Диаграма на класа “Category”

Подобно на „User“, в класът „Category“ пазим информация за всяка една категория. Имаме методи за слагане и взимане на “id”, “име” и „описание“, които се отнасят пряко за категорията. Също така имаме 3 конструктора, които приемат различни параметри според това, как е използван след това класът „Category“.



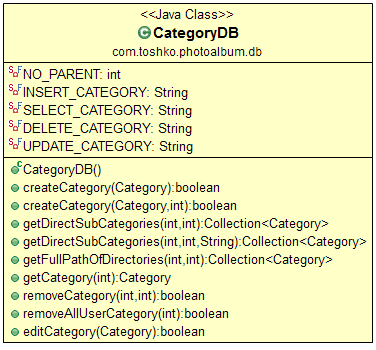
Фиг.16 Диаграма на класа „Picture“

И тук ситуацията е подобна. Пазим цялата необходима информация за една снимка. Имаме 3 конструктора които приемат различни параметри според това как е използван класа по-нататък. Също така имаме методи за слагане и вземане на „id“, „име“, „описание“, „съдържание на снимка“ и методи за взимане на „дата на създаване на снимката“ и „големина на снимката“.



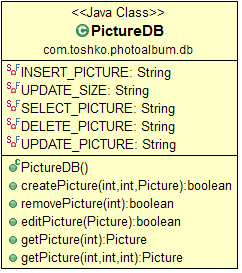
Фиг.17 Диаграма на класа „UserRegistry“

В класът „UserRegistry“ имаме заявки към базата от данни съответно за „обвновяване на информацията за даден потребител“, „обновяване на паролата за потребител“, „взимане на всички потребители“, „взимане на даден потребител“, „взимане на текущия потребител“, „проверка дали даден потребител е регистриран“, „изтриване на потребител“ и „добавяне на потребител“. За всяка една от заявките към базата от данни имаме съответстващ метод, който връща резултат на база на това какво е върнала заявката.



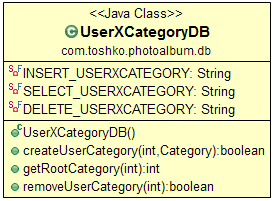
Фиг.18 Диаграма на класа „CategoryDB“

В класът „CategoryDB“ имаме заявки за „добавяне на категория“, „взимане на дадена категория“, „изтриване на категория“ и „обновяване на категория“. Съответно имаме метод за всяка една от заявките и отделно имаме методи за „взимане на всички подкатегории за дадена категория“, „взимане на всички „родители“ за дадена категория“ и „изтриване на всички категории за даден потребител“.



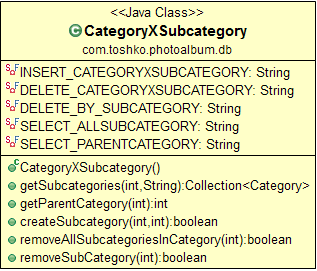
Фиг.19 Диаграма на класа „PictureDB“

В класът „PictureDB“ имаме заявки към базата от данни за: „добавяне на снимка“, „взимане на снимка“, „изтриване на снимка“, „обновяване на снимка“ и за „обновяване на големина на снимката“. За всяка една от заявките имаме еквивалентен метод, който връща резултата от заявките.



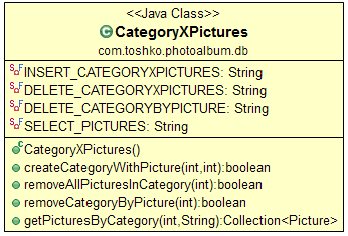
Фиг.20 Диаграма на класа „UserXCategoryDB“

Класът „UserXCategoryDB“ се грижи да пази информацията за „id“-то на потребителя и “id”-то на категорията. В този клас имаме заявки към базата от данни за: „добавяне на „id“-то на текущия потребител и „id“-то на текущата категория“, „взимане на “id”-то на “root” категорията“ и „изтриване на категория за даден потребител“. За всяка заявка имаме евквивалентен на нея метод.



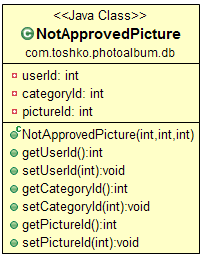
Фиг.21 Диаграма на класа „CategoryXSubcategory“

По подобие на класът „UserXCategory“ и тук имаме заявки за „добавяне на “id”-то на дадена категория и “id”-то на нейната подкатегория“, „изтриване на дадена категория“, „изтриване на дадена подкатегория“, „взимане на всички подкатегории“ и „взимане на категорията „родител““. За всяка от заявките имаме еквивалентни на тях методи.



Фиг.22 Диаграма на класа „CategoryXPicture“

И тук запазваме информация за текущото id на дадена категория и id-то на дадена снимка в тази категория. Имаме заявки за добавяне на id-то на категорията и id-то на подкатегорията, изтриване на id на категория, изтриване на id на подкатегория, взимане на всички снимки по дадено id. За всички заявки имаме еквивалентни на тях методи, които връщат резултат на база на това какъв резултат за върнали заявките.



Фиг.23 Диаграма на класа „NotApprovedPicture“

В класът „NotApprovedPicture“ имаме конструктор, който приема като параметри „userId“, “categoryId” и “pictureId”. Също така имаме методи които се грижат за вземане и слагане на „id за потребителя“, „id за категория“ и „id за снимка“.

За запознаване с това какво са UML диаграми, как се правят връзките между тях и къде намират най-голямо приложение можем да видим в [9].

## **2.8. Избор на среда за разработка**

Цялата система е разработена на JAVA, като са използвани различни технологии, които средата позволява. Във системата са използвани технологии като HTML, CSS, JavaScript, JSP, Servlet, Java API, Apache TomCat Server и база данни MySQL.

### 2.8.1. Алтернативи

Алтернативна среда за разработка е .NET 2.0 Framework. Средата поддържа перфектно API за разработка на уеб сайтове. Тя като алтернатива има предимства и недостатъци.

Предимства:

* Добра поддръжка и лесно закачване(data binding) на обекти и контроли.
* Внедрена база данни. Лесно администриране на базата.

Недостатъци:

* Средата е платена. За да бъде използвана, трябва да бъде както средата за разработка, така и сървърът(IIS), който средата ще използва
* Средата е разработена от комерсиална гледна точка. Тя работи перфектно със IIS, но за съжаление само с него. Всеки друг сервър внася допълнителни проблеми за разработчика.
* Сървърът, който .Net Framework използва работи само на операционна система Windows, което прави нашият сайт зависим от Windows и работещ единствено и само на него.
* Средата главно използва MSSQL, който отново е платен и е платформено зависим.

### 2.8.2. Операционна система

Системата е мулти платформена и работи на всякаква система. Единсвено се изисква да се инсталира JDK1.5 или по-висока версия и Apache TomCat. Сърърът (Apache TomCat) е написан ицяло на Java което го прави независим от платформата, на която ще бъде инсталиран.

### 2.8.3. Система за управлние на база данни

Системата използва за управление базата данни MySQL. Тя е изцяло написана на Java, което я прави платформено независима. Позволява внедряване на базата във кода, който пишем, което я прави пъти по-бърза от други бази работещи с Java приложения. Може да вика Java методи през рефлекция което прави приложението да може да изполва предимството на базата и да не се ограничава с него. Базата данни MySQL e напълно безплатна и може да бъде изтеглена както базата така и документация за нея от: <http://www.mysql.com>

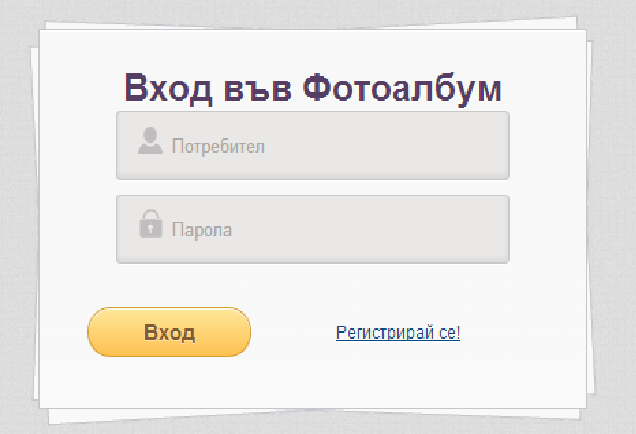
# Глава 3 - Описание на апаратната и софтуерната част

## 3.1. Потребителско ръководство

### 3.1.1. Потребителски панел

Потребителския панел отговаря за съдържанието на видимата част за потребителите от самата интернет страница.Той трябва да е възможно най-прост, за да може да не отегчава потребителя от безкрайни кликания на мишката, както и да има добър външен вид, за да може потребителите да искат да се върнат пак.

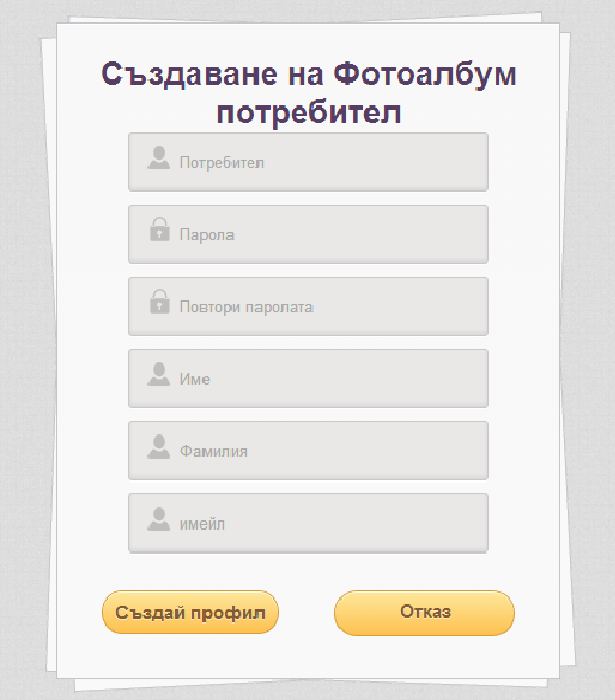
В нашия случай, преди потребителя да бъде допуснат в сайта, има “Login” страница, където да бъде аутентикиран(разпознат), която изглежда по следния начин:



Фиг.24 „Login“ страница

Потребителят трябва да въведе своя „username“ и парола за да бъде допуснат или ако няма такива да се регистрира.

Страницата за регистрация изглежда по следния начин:



Фиг.25 Страница за регистрация

В случай на избор за регистрация от страна на потребителя, той ще бъде изпратен на тази страница и ще трябва да попълни съотвените полета. Всичките полета са задължителни и трябва да бъдат попълнени коректно, в противен случай на потребителя ще бъдат изкарвани съобщения за грешка.

## 3.2. Административен панел

Административният панел, администрация или „back end“ са само част от наименованията на тази част от едно интернет приложение, която отговаря за съдържанието на видимата част за потребителите от самата интернет страница. В административния панел на едно уеб приложение може да се редактира, добавя или трие част от съдържанието на уеб страницата.

Администрацията на сайта е характерна само и единствено за динамичните сайтове, статичните сайтове сами по себе си няма как да бъдат администрирани по друг начин освен да се правят директни промени в HTML кода. Това е и удобството на администрацията, тя трябва да изглежда напълно достъпна за администратор, така че да може бързо и лесно да оперира с нея, намалява се и рискът от грешки в HTML кода.

Административния панел остава скрит от външния свят, но и той сам по себе си трябва да има представителност дори и пред администратора, трябва да отговаря на общоприетите стандарти за използваемост (usability standards), а те са:

лесен и бърз за научаване;

лесен и бърз за използване;

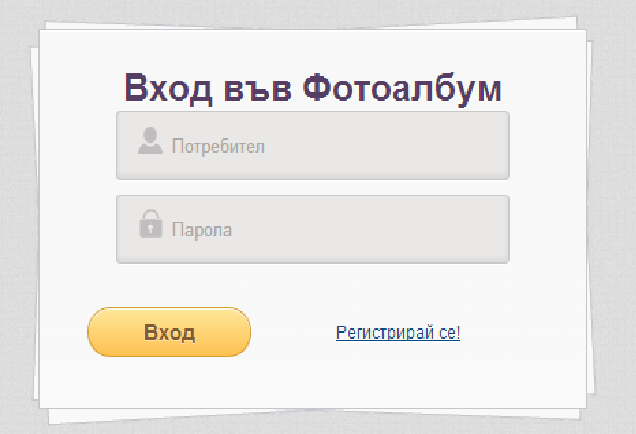
приятна и разбираема визия.

Тъй като искаме администрацията да остава скрита за посетителите на сайта, поставяме изискване администраторът да въвежда име и парола при всяко влизане в нея. Също така тя може да е разделена на нива на достъп, т.е. ако има повече от един човек достъп до нея. В нашия случай имаме само 1 ниво за достъп – администратор.

Сега ще разгледаме основните административни панели

### 3.2.1 Влизане в административния панел

Ето как изглежда тази част от администрацията, която е входа към същинската част на администрацията:



Фиг.26 „Login“ страница за администратор

Това е страницата която се появява, където администраторът трябва удостовери самоличността си чрез потребителско име и парола, следващата стъпка е според името и паролата да получи съответното ниво на достъп. В случай, че администратора сгреши потребителското име или паролата ще се появи съобщение за грешка, в което ще се съдържа информация за вида на грешката. Съобщението за грешка се използва почти навсякъде в администрацията, където дадена форма не е попълнена правилно или има липсващи полета. В този случай има само две възможни грешки, едната е ако администраторът е оставил празно полето за потребителско име или за парола и втората е, както споменахме по горе, ако сгреши потребителското мето или паролата или и двете. Ето и как изглежда съобщението за грешка:

C:\Users\todor.radev\Desktop\Untitled3.png

Фиг.27 Съобщение за грешка при невалиден потребител или парола

Всички съобщения за грешка в администрацията следват този шаблон. Съществува и съобщение за успешно извършена операция, но тук то не е необходимо, защото при опит за влизане в административния панел има само два варианта: успешен и неуспешен. След като разгледахме неуспешния вариант за влизане в административния панел, нека сега разгледаме и успешния. При въвеждане на правилно потребителско име и парола, администраторът бива препратен към вътрешната страница на администрацията за управление на снимките. В този случай не излиза съобщение за успешен вход в администрацията, защото резултатът, че администратора е вътре е достатъчно доказателство.

### 3.2.2. Излизане от администрацията

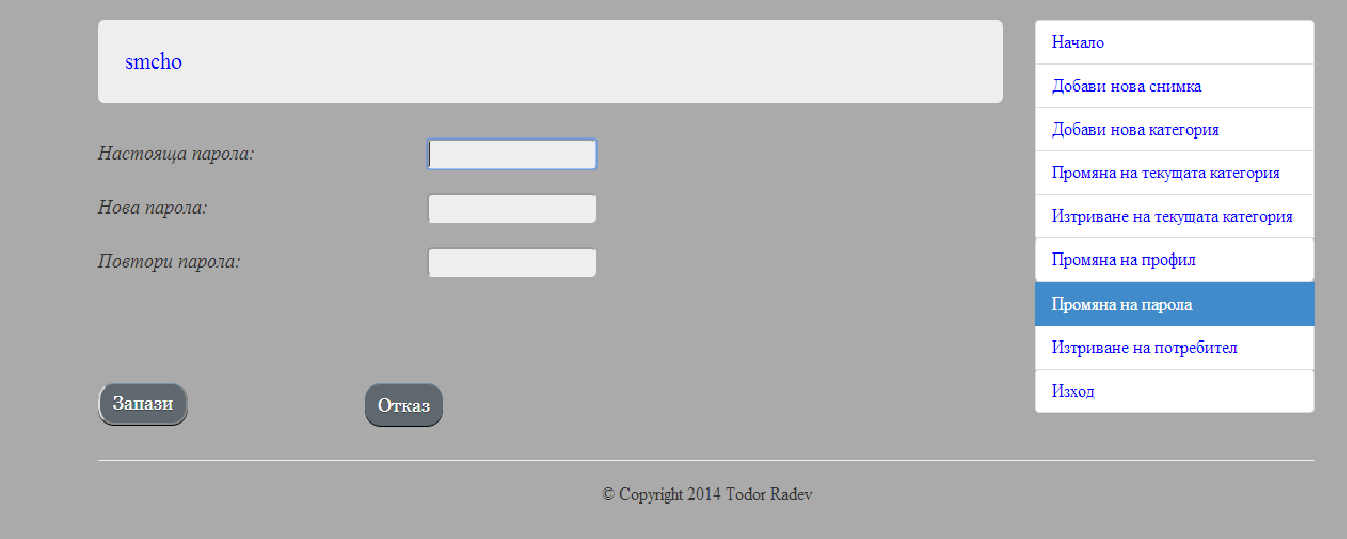
Както има вход към администрацията така има и изход. Изходът е необходим, за да не може да се влезе обратно в нея от друг човек ползващ същия компютър. Това е задължително условие на което трябва да отговаря всяка администрация и изобщо всяка система изискваща вход чрез потребителско име или парола. В нашата администрация излизането става чрез натискане на бутонът „Изход“ в горната дясна част на страницата. Съществува и автоматичен начин за излизане в случай, че администраторът забрави да излезе от нея след като си е свършил работата. Този начин е чрез т.нар. „изтичане на сесията“, това ще рече, че ако няма активност (не се натискат никакви линкове и не се пращат форми с информация кум сървъра, който обслужва административния панел и сайтът) в текущата страница в продължителност на 30 мин., то тогава при натискане на линк, администраторът ще бъде препратен обратно към страницата за вход. Съществуват значително по интелигентни системи, които запомнят текущото състояние на работа за потребителя(администратора) в администрацията и след обратно влизане в нея могат спокойно да си продължат работата. Но това е мярка която се взима в значително по сериозни системи, като банковите системи, системите за трансфер на криптирани файлове и много други. В нашия случай говорим за съвсем елементарна администрация, в която в най-лошия случай, потребителят ще се наложи да стигне до снимката която е разглеждал или директорията в която е бил наново, което би отнело не повече от 1 минута.

## 3.3. Администратор

Основната цел на администратора е да одобрява/отказва снимките.Той има достъп до всички снимки на всеки един потребител и само той може да реши дали да одобри дадената снимка или да я откаже.Идеята да има администратор за изработения сайт е, за да може да се спести качването на снимки с неподходящо съдържание от страна на потребителите.

## 3.4. Смяна на парола

Тази секция е предназначена за смяна на паролата за влизане в административния панел. Тя е най-често срещаните във всяка една администрация на сайт, който изисква потребителско име и парола за достъп. В тази секция се влиза посредством линка „Промяна на парола”, който се намира в горния десен ъгъл на менюто и изглежда така:



Фиг.28 Секция за „Смяна на парола“

В секцията за смяна на паролата потребителя трябва да въведе текущата си парола и новата парола, която трябва да бъде въведена два пъти. След като изпълни правилно изискването за смяна на паролата и натисне бутона „Запази“, тази форма ще бъде изпратена до web server-а и съответно ако няма проблеми с въведените данни (грешна текуща парола, разлика в повтарянето на новата парола и т.н.) на страницата ще излезе съобщение, че е обновена паролата успешно:



Фиг.29 Съобщение при успешно сменена парола

В случай на проблем с некоректно въведени данни, възжмоните грешки ще бъдат:

* *неправилна стара парола;*



Фиг.30 Съобщение при грешно въведена текуща парола

* *несъвпадение между полето на новата парола и полето за повторение на новата парола;*



Фиг.31 Съобщение при несъвпадение между полетата за „Нова парола“ и „Повтори парола“

* *Новата парола е по-малка от 6 символа.*



Фиг.32 Съобщение, че новата парола е по-малка от 6 символа.

**Защо потребителя трябва да въвежда два пъти нова парола?**

Идеята с въвеждането на новата парола два пъти е изключително ползотворно за потребителите ,защото полето за паролата е засекретено(символите които въвежда потребителя се изобразяват със звездички) и не се вижда докато се въвежда. Така ако потребителя допусне чисто техническа грешка в написването й, няма как да разбере. Затова идеята с повтаряне на парола е изключително успешна и се използва в почти всички системи в които има секция за смяна на паролата.

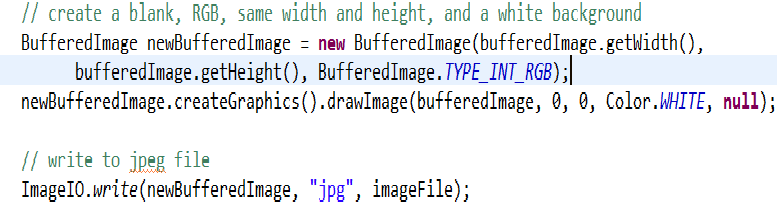
## **3.5. Компресиране на снимки**

Един от проблемите, с който трябва да се справят сайтовете които предоставят възможността за качване на снимки е намаляне на големината на снимките или така нареченото компресиране на снимки. Какво става ако потребителите започнат да качват снимки по 7,8 или дори 10мб и ние не ги компресираме? Това би било абсурдно. Много бързо бихме си запълнили свободното място на твърдия диск.

Чрез компресия на снимките сме си решили проблема и в нашия случай. Как става това? Когато потребителя качи своя снимка, без значение колко е голяма и в какъв формат е, ние я превръщаме в „jpg” формат. Така снимката намаля своята големина драстично. Единствената загуба е, че се губи от качеството на снимката, което би се забелязало само на монитор с много голяма резолюция. Пример за компресирана снимка има и в моя сайт. Снимка която е 7мб, след компресиране големината и става 474кб. Това спестява много място на твърдия диск.

Така ние не поставяме ограничение на потребителите за размера на снимките, ние ги смаляваме вместо тях ако е необходимо.

На фиг. 3.5.1. е показан метода, който извършава компресирането на снимката.



Фиг.33 Метода write(), който прави компрасирането на снимките

Java ни предоставя метода write() в класа ImageIO, на който му подаваме като параметри „снимката която ще компресираме, формата в който искаме да я направим(в нашия случай jpg и файла в който да я запише.

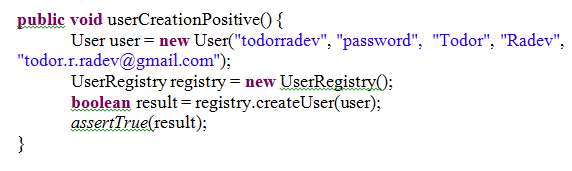
# Глава 4 – Изчислителна част, функционално тестване на софтуерната част.

## **4.1. Функционални тестове**

### **4.1.1. Тестове за потребител**

В тази секция ще разгледаме няколко позитивни и негативни теста на някой от ключовите методи за проекта.

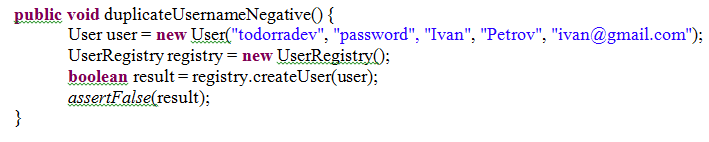
Първият тест, който ще направим е за *успешно* създаден потребител:

****

Фиг.34 Тест за успешно създаден потребител

Теста минава успешно, тъй като въведената от нас информация е абсолютно коректна.

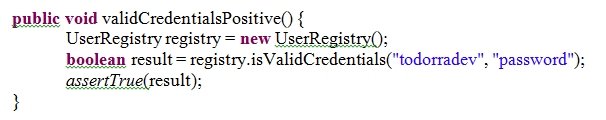
Нека сега пробваме да създадем потребител със същия “username” .

****

Фиг.35 Тест за създаване на потребител със съществуващ вече username

Както очаквахме, резултатът ни е „false“ – отрицателен, тъй като “username”-а е уникален и не можем да имаме 2 потребителя с един и същ “username”.

Следващият ни тест ще симулира потребител, който опитва да се “log”-не в нашия сайт и системата прави проверка дали въведените от него „потребител“ и „парола“ са валидни.

****

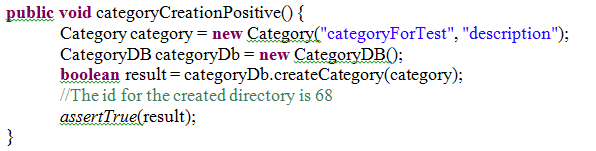
Фиг.36 Проверка дали са валидни въведените потребител и парола

Създаедният по рано от нас потребител с полета за достоверност - “username”="todorradev" и парола="password" е въведен в този тест. Както очаквахме теста преминава успешно с резултат „true“.

### **4.1.2. Тестове за категории**

В тази секция ще разгледаме тестове които се отнасят насочени към създаването и взимането съответно на категории и подкатегории.

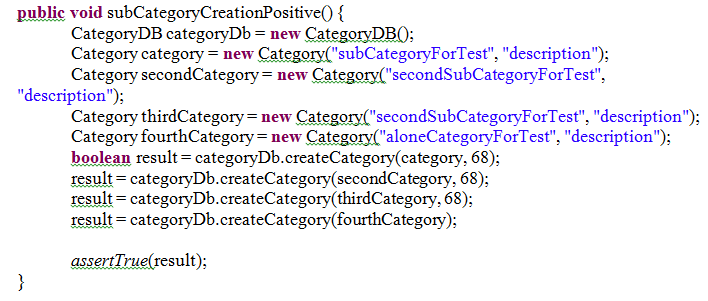
Първият тест който ще разгледаме е успешно създадена категория.



Фиг.37 Тест за успешно създаване на категория

Създаваме си категория с име = "categoryForTest" и описание = "description". Системата ни дава „id“ с номер = 68(“id”-тата се раздават от системата автоматично, като започват от „1“ и при всяка следваща създадена категория следват „2“, „3“ и т.н. .

Нека сега разгледаме тест, в който създаваме нови категории, които ще се намират в създадената вече категория „categoryForTest“ и една която няма да има “parent” категория (ще разберем по-късно защо). Тези нови категории(без последната) ще се явяват подкатегории за „categoryForTest“.

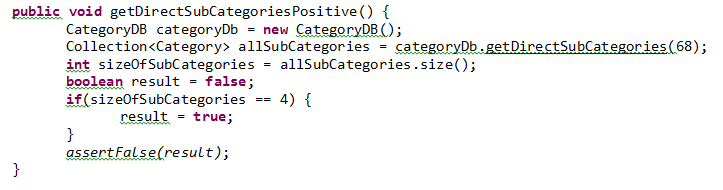
****

Фиг.38 Тест за успешно създаване на подкатегории

Създаваме 4 нови категории, за които при създаването им, на 3 от тях задаваме да се намират в категория с номер „68“(която е създадената преди малко от нас „categoryForTest“).Последната (4та) категория я създаваме без да има „parent” категория.

След изпълнение на теста виждаме, че резултата който получаваме е „true“, т.е. е преминал успешно.

Последният тест който ще направим е да видим колко на брой са подкатегориите в „categoryForTest“.

****

Фиг.39 Взимане на броя подкатегории в „categoryForTest“

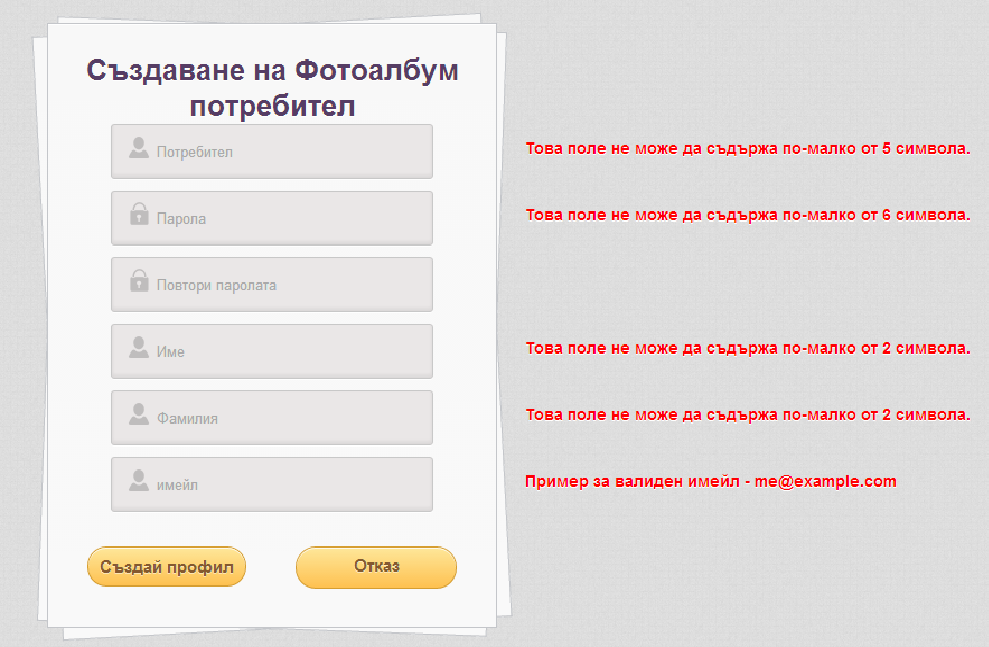
Както виждаме резултата след изпълнение на теста е „false“, което е очакваното поведение, тъй като ние имаме 3 подкатегории на категорията „categoryForTest“. Последната 4та категория която създадохме в предния тест, няма „parent“ категория.

## **4.2. Test cases**

Ще покажем положителни и отрицателни тестове за някой от основните функционалности на сайта.

### **4.2.1. Страницата за регистриране**

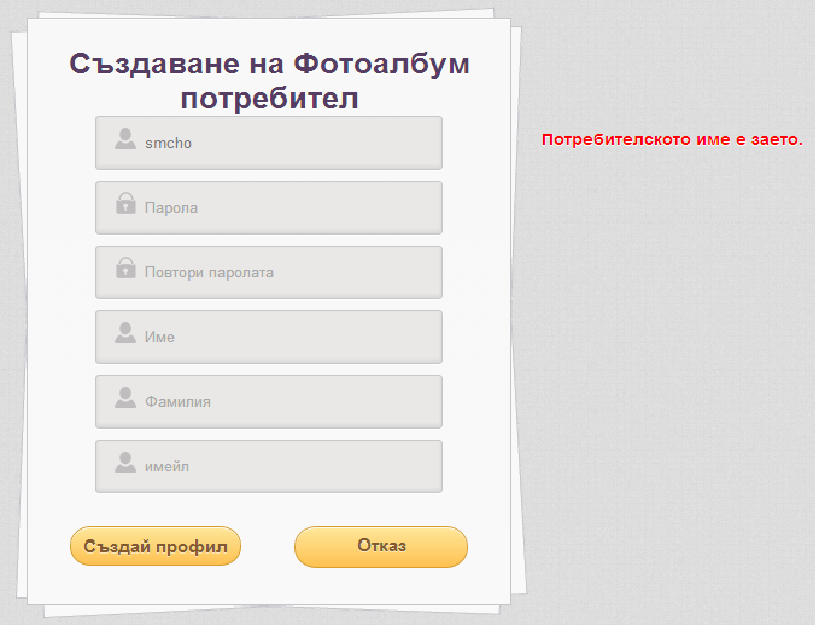
Ще започнем с отрицателен тест в секцията за регистриране.Нека се опитаме да направим регистрация като оставим всички полета празни.Резултатът ще бъде следния:



Фиг.40 Опит за регистрация при невъведени данни

Както виждаме получаваме съобщение за грешка на всяко поле, без това за „Повтори парола“, тъй като там ще получим грешка само ако сме въвели някаква парола в полето за „Парола“ или „Повтори парола“ и те не съвпадат.

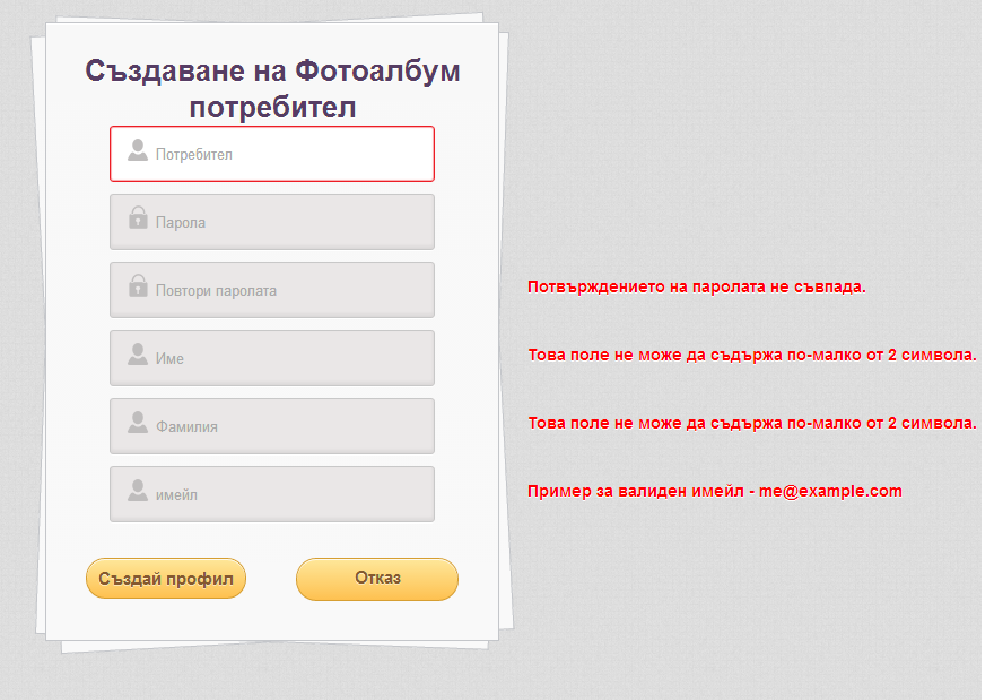
Нека сега направим опит за регистрация с потребител, който вече съществува.



Фиг.41 Опит за регистрация при вече съществуващо потребителско име.

Както виждаме получаваме съобщение за грешка срещу полето за „потребител“, която гласи „Потребителското име е заето.“

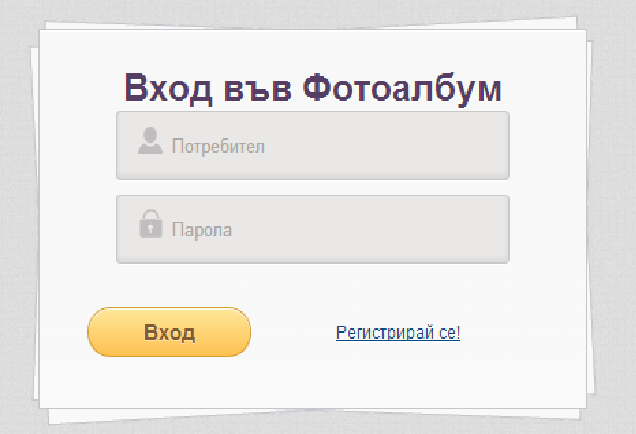
Нека сега пробваме да въведем коректни данни за „Потребител“ и за „Парола“ и да видим какво ще се случи.



Фиг.42 Опит за регистрация при коректно въведени данни за „потребител“ и „парола“

Както очаквахме получаваме грешки на всички полета без коректно въведените данни за „Потребител“ и „Парола“.Този път получаваме грешка и за „Повтори паролата“, тъй като системата отчита, че имаме въведена парола и очаква потвърждение за такава.

Последния ни опит от секецията за регистрация ще бъде с коректно въведени данни за всички полета и съответно успешно създаване на потребител.В този случай системата няма да ни покаже съобщение,че профила ни е бил създаден успешно, защото очевидно след като не сме получили съобщение за грешка, то той е създаден успешно.В случая системата ни изпраща до “Login” страницата чакаща да си въведем новосъздадените „потребител“ и „парола“.”Тя изглежда по следния начин:

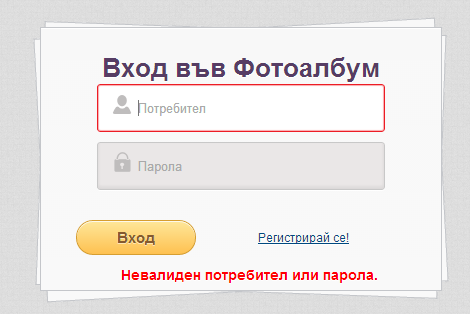


Фиг.43 Препращане до „Login“ страницата при успешна регистрация

## 4.3. Страница за „Login“

Нека сега разгледаме един положителен и един отрицателен тест на страницата за „Login“.

Първо ще разгледаме отрицателния случай.Има няколко варианта, в които можем да въведем некоректно данни.Можем да оставим едно от полетата за „Потребител“ или „Парола“ празни(или и двете), можем да въведем грешен потребител или грешна парола.Във всеки един от случаите системата ще ни покаже следната страница:

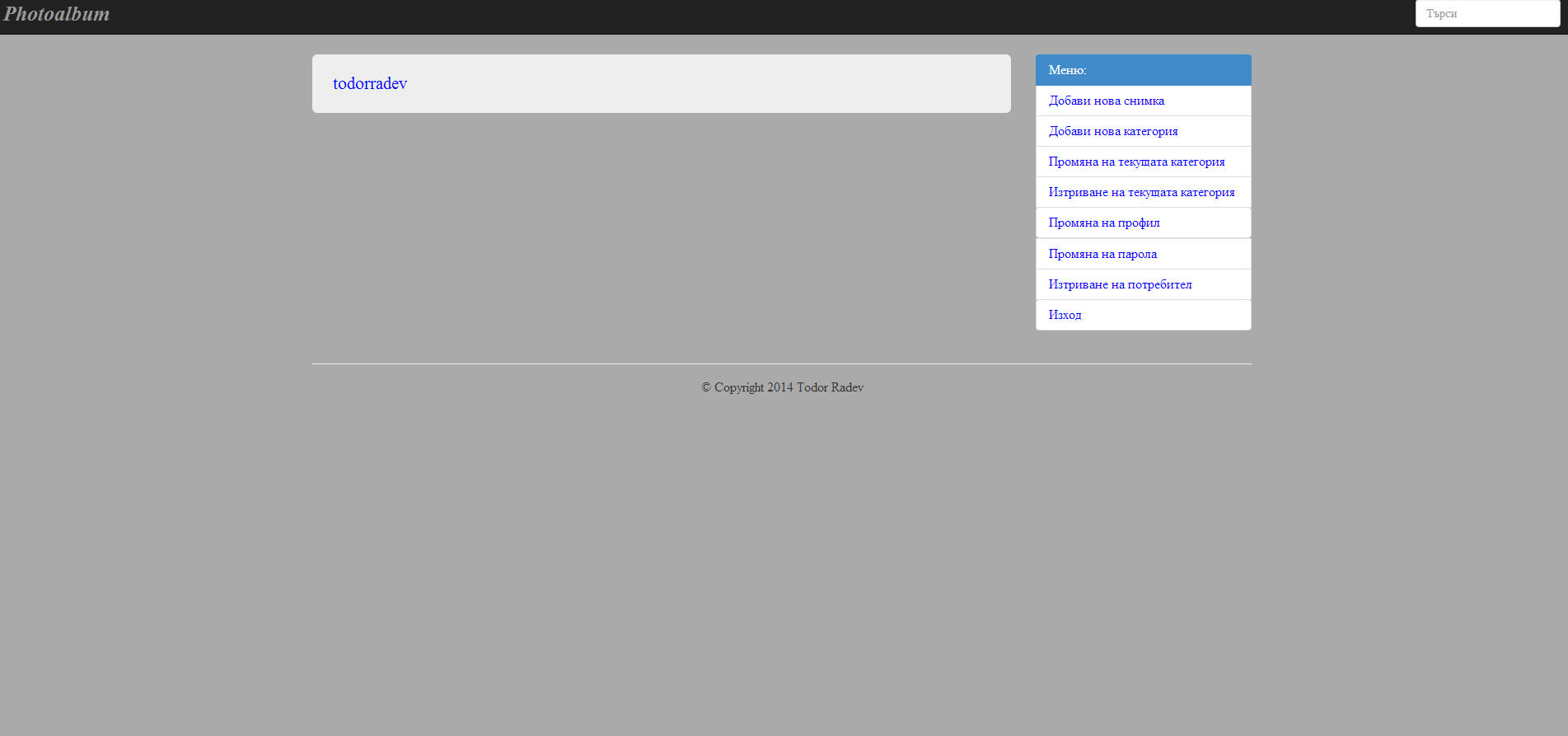


Фиг.44 Опит за влизане в системата при невалиден потребител или парола

Както виждаме сами под бутона за вход, получаваме съобщение което гласи: „Невалиден потребител или парола“.Системата не ни казва кое от полетата е сгрешено, с цел предпазване от „хакерски“ намеси.

Нека се разгледаме и другия случай, в който сме си въвели коректно данните.

След като сме създали своя профил успешно и въведем правилните потребител и парола, то системата ни изпраща до главната ни страница в профила, която изглежда по следния начин:



Фиг.45 Изглед на главната страница при успешно влизане в системата

Какво имаме на нашата основна страница след успешно влизане в системата?

В средата на страницата се появява една сива лента, в лявата на част на която е изписан нашия „username“, който сме записали при регистрирането ни.Това е своеобразен път по който да се движим, при създаване на нови категории.Пример за такъв път можем да видим в следващия пример.

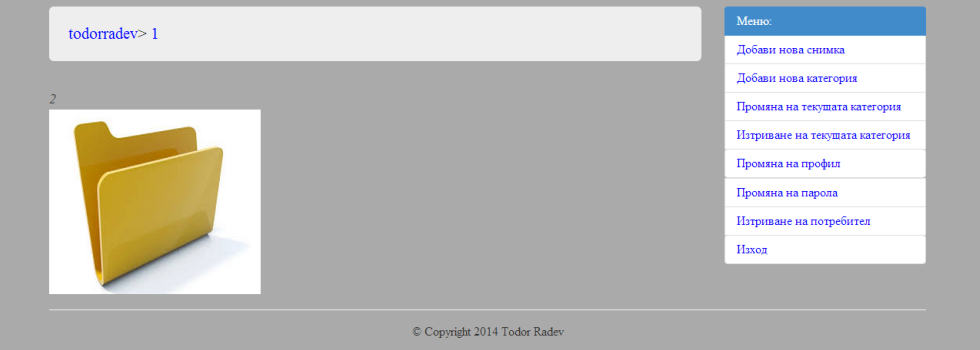
## 4.4. Възможни опции при създаден вече потребител

### **4.4.1. „Път“**



Фиг.46 Показване на едната от опциите – „Път“ която се намира в светло сивата лената

Ето тук ние сме си създали нова категория, в която ако влезем, пътят ни би изглеждала по следния начин:



Фиг.47 Продължение на фиг.46

Създаваме си нова категория, на която „родителят“ и е „1“, на който “родителят” му е „todorradev“ и пътят ни вече изглежда така:

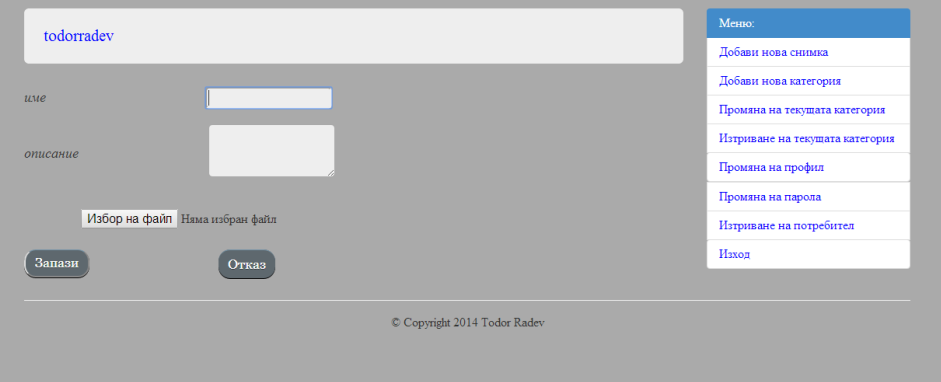


Фиг.48 Продължение на фиг.47

Така ние вече се намираме в последната ни създадена катеогиря с име „2“.Сега ние ако искаме да се върнем в „1“ или в главната ни категория „todorradev“, всичко което трябва да направим е да щракнем на самото име.

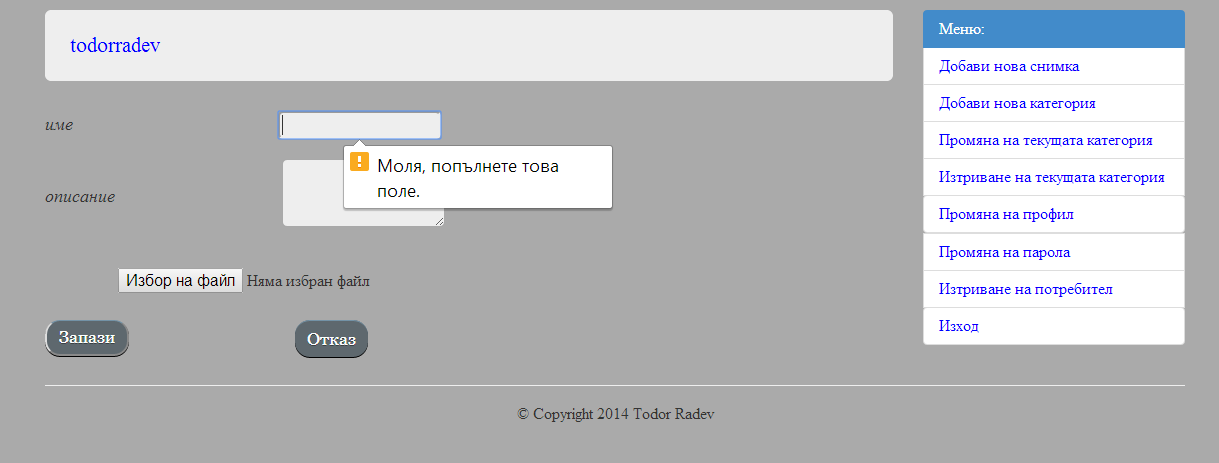
### **4.4.2. Меню**

От дясната страна имаме меню с опции, които да изберем.Нека направим тест със създаване на снимка.Страницата на която ни изпраща при избор на опция „Добави нова снимка“ изглежда по следния начин:



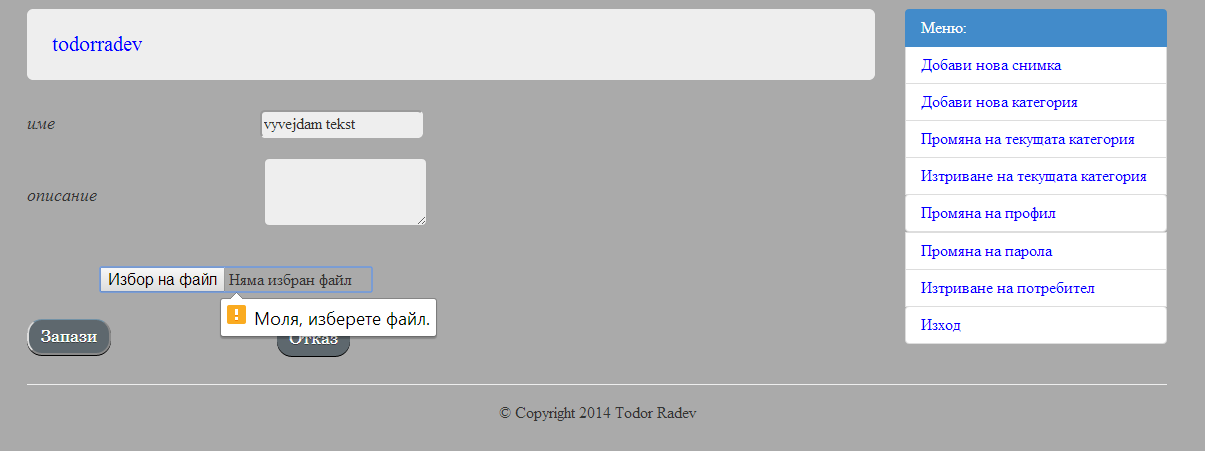
Фиг.49 Изглед на секцията за „Добавяне на снимка“

За да успеем да добавим успешно нова снимка, ние трябва **задължилтено** да попълним полето за „име“, както и да сме избрали снимка.В противен случай, ние ще получим съобщение, което гласи:



Фиг.50 Опит за добавяне на снимка с невъведени данни за полето „име“

Или:



Фиг.51 Опит за добавяне на снимка при неизбран файл

След като въведем коректно всички данни, то нашата снимка бива изпратена до администратора за одобрение.Чак след получим одобрение от администратора нашата снимка се появява.Пример за успешно качена снимка:



Фиг.52 Изглед на успешно добавена снимка

# Глава 5 - Икономическа оценка на резултатите от дипломната работа

За реализацията на проекта са използвани само безплатни версии на съответния софтуер тъй като те са абсолютно достатъчни за осъществяване на настоящите цели. За нуждите на една голяма организация обаче, безплатните версии на софтуера може да се окажат недостатъчни.

От всички софтуери които съм използвал в настоящата дипломна работа единствено MySQL има платен еквивалент.

## 5.1. Сравнение на безплатната версия на MySQL с неговия платен еквивалент

За наше щастие безплатната версия на MySQL ни е напълно достатъчна, тъй като тя предлага всичко необходимо. Ето и някои от предимствата които са ни предоставени от платената версия:

* Лесен за използване – MySQL е изключително лесен за ползване. Нужни са ни само основни познания по SQL
* Сигурен - MySQL притежава изключително силни защитни стени, което позволява сигурност от намеса на хакери
* Бърз – MySQL е една от най-бързите база от данни. Други които са също много бързи са Sybase и Oracle.

Съществената разлика на платената версия с тази на безплатната е, че в платената версия на MySQL имаме поддръжка. В настоящия проект обаче, това по никакъв начин няма да ни повлияе.

Останалите използвани софтуери, които съм използвал съществуват само в безплатна версия, което е изключително изгодно от икономическа гледна точка.

Като заключение в тази глава можем да кажем, че избора на софтуер, който е използван спрямо цена и ефективност е изпълнен. Едиственото което можем да подобрим е да използваме платената версия за MySQL, но то ще се окаже неефективно, тъй като предимствата му спрямо безплатната версия са изключително малко и неефективни в нашия случай, което само ще увеличи нашите разходи, но няма да подобри качеството.

# Глава 6 – Приложимост на дипломната работа.

За уеб приложенията изключително важно нещо са тяхната приложимост. В противен случай времето и разходите биват прахосвани напразно.

Сайтът който съм реализирал в дипломната работа намира изключително широко приложение. Конфигурацията и поддръжката му са изключително прости, базите данни са лесни за научаване от хора които не са се сблъсквали досега с тях. Лекият дизайн който има и използваните технологии при създаването на приложението позволява да се използват сървъри със сравнително евтин хардуер, което би повишило търсенето и използването му. Едно от най-големите предимства на сайта е изключителната гъвкавост при желание или нужда от разширяването му и добавянето на нови функционалности и услуги. Друго сериозно предимство на сайта е възможността от използването му както през комютри така и през телефонни устройства и таблети. Дизайна му е направен така, че уеб приложението променя размерите си според резолюцията на устройството за да се адаптира към него. Също така голямо предимство е широкото разпространение на сайта под различни операционни системи. Много сайтове имат ограничение да бъдат използвани само под Windows заради избора на програмни средства(пример е C#, или MSSQL), които са на Microsoft. В нашия случай, ние сме използвали софтуер, който може да се използван под всички операционни системи и това прави сайта изключително гъвкав, и не налага ограничение в използването му.

# Глава 7 – Изводи и претенции за самостоятелно получени резултати.

## 7.1. Претенции за оригинално получени резултати

* Съставени са функционални тестове, които тестват положителните и отрицателни резултати при създаване на потребител. Също така проверяват дали потребителят е въвел коректно username и парола
* Съставени са функционални тестове, които тестват положителни и отрицателни резултати при създаване на категории и подкатегории. Също така има проверка дали подкатегориите са записани коректно в дадена категория
* Проверка за броя на всички подкатегории в дадена категория
* Създадени са Test Cases, които показват нагледно как ще реагира системата при определени действия на даден потребител. Обхванати са основните функционалности на сайта
* Създадена е отделна страница за администратор, който да проверява съдържанието на снимките за всеки един потребител. Администраторът има права за одобряване или изтриване на дадена снимка

Като извод от направената дипломна работа може смело да кажем, че сме прокетирали един добре структуриран и технически издържан уеб сайт. Използвали сме наложили се на пазара софтуери, които ни гарантират безупречна работа и качество. Разбира се времената се променят изключително бързо и след няколко години софтуера който сме използвали ще е остарял и ще са излезли нови по-добри версии. Затова трябва постоянно да се осведомяваме за новите софтуери на пазара и затова кои са най-качествените продукти. Настоящата дипломна работа може да е от помощ на студенти, които са решили да се занимават с уеб приложения. В бъдеще уеб приложението може да се променя, да се използват допълнителни софтуери, допълнителна логика, но ние имаме основата и можем само да надграждаме. В заключение можем да кажем, че поставената задача в дипломната работа е изпълнена.

## 7.2. Възможности за бъдещо развитие

Неща които могат да се подобрят в сайта са:

* създаване на рекламна секция. Това би бил изключително добър избор от маркетингова гледна точка
* Да се направи системата да изглежда по-добре под стари версии на IE(Internet Explorer).
* Да се направи мобилно приложение за сайта.
* Да се уведомяват потребителите при одобрение или изтриване(като се заяви причината) на качената от тях снимка чрез съобщение.
* Оптимизация на одобряването на снимки от страна на админа. В момента админа ръчно минава и одобрява/изтрива снимките, но ако имаме 1000 или повече потребителя, то тогава ще му е трудно да ги преглежда навреме.
* Възможност от споделяне на снимки между потребители. В момента споделянето на снимки става след като бъде изпратен URL-а на снимката чрез skype, email и т.н.

# **Използвана Литература**

[1]B. Eckel, “Thinking in Java”, 1998.

[2] D. Flanagan, “JavaScript: The Definitive Guide, Sixth Edition”, 2011.

[3] D. Garet, “Ajax: A New Approach to Web Applications”, 2005.

[4] D. Gourley, B. Totty, “HTTP: The Defintive Guide”, 2002.

[5] E. Meyer, “CSS The Definitive Guide”, 2006.

[6] J.Hunter, W. Crawford, “Java servlet programming”, 2001.

[7] M. Pilgrim, “Dive into HTML5”, 2009.

[8] O. Reilly, “JavaServer Pages”, 2002.

[9] S. Mishra. "Visual Modeling & Unified Modeling Language (UML) : Introduction to UML". Rational Software Corporation, 1997.

[10] S. Pachev, “Understanding MySQL Internals”, 2007.

# Списък на възприетите преводи на английски термини

В тази таблица са събрани всичките възприети в дипломната работа преводи от английски термини на български език.

|  |  |
| --- | --- |
| Английски термин | Възприет превод на български език |
| Application servers | Сървъри за приложения |
| Controller | Контролер |
| Cookie | Бисквитка |
| Data Binding | Лесно закачване |
| Login | Вход |
| Model | Модел |
| Private | Личен |
| Root | Корен, главен |
| Thin client | Тънък клиент |
| Тhread | Нишка |
| Usability standards | Стандарти за използване |
| Username | Потребител |
| View | Изглед |

Табл.3 Списък на всички английски термини

# Списък на използваните съкращения

В тази таблица са събрани всички съкращения, използвани в изложението. Някои от тях са добре утвърдени сред компютърните специалисти, други са въведени в настоящата работа с цел улесняване на записа.

|  |  |
| --- | --- |
| Съкращение | Пълно наименование |
| CSS | Cascading Style Sheets |
| HTML | Hyper Text Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IP | Internet Protocol |
| JSP | Java Server Pages |
| Java | Език за програмиране |
| MySQL | Език за бази данни |
| MVC | Model View Controller |
| PHP | Език за програмиране |

Табл.4 Списък на използваните съкращения

Съдържание

[Глава 1 - Състояние на проблема по литературни данни. 3](#_Toc391367942)

[1.1. Сигурност 4](#_Toc391367943)

[1.2. Анализ на съществуващи системи 5](#_Toc391367944)

[1.3. Сравнение на направения сайт с друг такъв 6](#_Toc391367945)

[1.4. WEB приложения 7](#_Toc391367946)

[1.4.1. *WEB сървър и WEB браузър* 7](#_Toc391367947)

[1.4.2. WEB сайт 8](#_Toc391367948)

[1.4.2.1. Статични WEB сайтове 8](#_Toc391367949)

[1.4.2.2. Динамични WEB сайтове 8](#_Toc391367950)

[1.4.3. HTTP 10](#_Toc391367951)

[1.5. Цел на дипломната работа 10](#_Toc391367952)

[Глава 2 - Теоретично решение на поставената задача. 11](#_Toc391367953)

[2.1. Преглед на средствата за програмна реализация 11](#_Toc391367954)

[2.1.1. HTML 11](#_Toc391367955)

[2.1.1.1. Същност 11](#_Toc391367956)

[2.1.1.2. Структура на HTML документ 11](#_Toc391367957)

[2.1.2. CSS 12](#_Toc391367958)

[2.1.2.1. Същност 12](#_Toc391367959)

[2.1.2.2. Синтаксис 13](#_Toc391367960)

[2.1.3. Същност на Javascript 14](#_Toc391367961)

[2.1.3.1. Разлики с Java 15](#_Toc391367962)

[2.1.3.2. Възможности на JavaScript 15](#_Toc391367963)

[2.1.3.3. Интерпретиране 16](#_Toc391367964)

[2.1.3.4. Предимства и недостатъци на Ajax 16](#_Toc391367965)

[2.2. Релационни бази от данни 17](#_Toc391367966)

[2.2.1. Език на структурирани заявки (MySQL) 19](#_Toc391367967)

[2.3. Езици за създаване на динамични WEB – сайтове( предимства и недостатъци) 20](#_Toc391367968)

[2.3.1 JAVA 20](#_Toc391367969)

[2.3.2 Същност на JSP 21](#_Toc391367970)

[2.3.3. Същност на Java Servlet 24](#_Toc391367971)

[2. 4. Етапи при работа с динамичен сайт 26](#_Toc391367972)

[2.5. Избор на програмни средства 27](#_Toc391367973)

[2.6. Функции които се предоставят пред потребителя 27](#_Toc391367974)

[2.7. UML Диаграми 30](#_Toc391367975)

[2.8. Избор на среда за разработка 36](#_Toc391367976)

[2.8.1. Алтернативи 36](#_Toc391367977)

[2.8.2. Операционна система 37](#_Toc391367978)

[2.8.3. Система за управлние на база данни 37](#_Toc391367979)

[Глава 3 - Описание на апаратната и софтуерната част 38](#_Toc391367980)

[3.1. Потребителско ръководство 38](#_Toc391367981)

[3.1.1. Потребителски панел 38](#_Toc391367982)

[3.2. Административен панел 39](#_Toc391367983)

[3.2.1 Влизане в административния панел 40](#_Toc391367984)

[3.2.2. Излизане от администрацията 42](#_Toc391367985)

[3.3. Администратор 43](#_Toc391367986)

[3.4. Смяна на парола 43](#_Toc391367987)

[3.5. Компресиране на снимки 46](#_Toc391367988)

[Глава 4 – Изчислителна част, функционално тестване на софтуерната част. 47](#_Toc391367989)

[4.1. Функционални тестове 47](#_Toc391367990)

[4.1.1. Тестове за потребител 47](#_Toc391367991)

[4.1.2. Тестове за категории 48](#_Toc391367992)

[4.2. Test cases 49](#_Toc391367993)

[4.2.1. Страницата за регистриране 50](#_Toc391367994)

[4.3. Страница за „Login“ 53](#_Toc391367995)

[4.4. Възможни опции при създаден вече потребител 55](#_Toc391367996)

[4.4.1. „Път“ 55](#_Toc391367997)

[4.4.2. Меню 56](#_Toc391367998)

[Глава 5 - Икономическа оценка на резултатите от дипломната работа 58](#_Toc391367999)

[5.1. Сравнение на безплатната версия на MySQL с неговия платен еквивалент 58](#_Toc391368000)

[Глава 6 – Приложимост на дипломната работа. 59](#_Toc391368001)

[Глава 7 – Изводи и претенции за самостоятелно получени резултати. 59](#_Toc391368002)

[7.1. Претенции за оригинално получени резултати 59](#_Toc391368003)

[7.2. Възможности за бъдещо развитие 60](#_Toc391368004)

[Използвана Литература 61](#_Toc391368005)

[Списък на възприетите преводи на английски термини 61](#_Toc391368006)

[Списък на използваните съкращения 62](#_Toc391368007)