Софийски университет Св. Климент Охридски

Факлултет по математика и информатика

Катедра „Информационни технологии”

ДИПЛОМНА РАБОТА

тема:

ИНФОРМАЦИОННА СИСТЕМА С МОБИЛЕН ДОСТЪП

Дипломант: П е т ъ р Х р и с т о в С в е т и е в

Специалност: Разпределени системи и мобилни технологии

Фак. номер : M21836

Научен ръководител: доц. д-р Васил Георгиев

гр. София
2007 г.

Съдържание:

Въведение

Основно изискване към всяка информационна система е да осигури актуална информация до широк кръг от потребители. Използването на глобаната мрежа Интернет дава възможност данните да се разпространяват бързо между неограничен брой хора. Относително ниската цена и бързата инсталация са направили постоянния достъп до Интернет обичаен за офиса и дома. Но има случаи, в които получаването на информация от системата е от особена важност за потребители в движение. Тогава постоянната връзка с Интернет е значително по-трудно осъществима. Тези хора, обаче са постоянно свързани с друга мрежа – мобилната мрежа на своя клетъчен оператор. Този проект цели да направи връзка между двете най-популярни световни мрежи в момента – Интернет и мобилните телефони.

Настоящата разработка представлява уеб базирана система с възможност за лесно управление на съдържанието, която поддържа достъп през персонален компютър и мобилно устройство. Предоставя се възможност за попълването на съдържание от произволно избрана предметна област, без да е необходимо изменение на функционалността на системата. В резултат на това продуктът е подходящ за използване и управление от широк кръг потребители, които не е нужно да притежават специфични компютърни познания.

Разработената система притежава следните характеристики:

- централизирана база данни за съхранение на информацията;
- уеб интерфейс за обновяване на цялото съдържание на системата и възможност за приложение в различни предметни области;
- възможности за прилагане на системата за отдалечено обучение
- различни права за посетителите;
- достъп до информацията през Интернет;
- компактен XHTML MP интерфейс за достъп до системата през мобилен телефон чрез WAP или i-mode;
- предоставяне на информация чрез SMS, WAP Push SMS и MMS.

Части от съставения XHTML Mobile Profile код и събраните теоретични материали са използвани при съставянето на упражнения за курса по WAP/WML/XHTML във Факултета по математика и информатика към Софийския университет.

Дипломната работа се състои от следните части:

- **Въведение**.

Кратко представяне на проекта и структурата на дипломната работа

- **Глава 1 – Обзор на проблемната област и съществуващи реализации.**

В тази глава се описват проблемните области, които обхваща настоящния проект и се прави анализ на няколко съществуващи приложения за всяка от тях.

- **Глава 2 – Технологии за реализация.**

Подробно се разглеждат всички технологии, които се използват за реализиране на проекта. Накратко са описани популярни Интернет технологии и по-подробно са разгледани технологиите за мобилен Интернет и съобщения.

- **Глава 3 – Функционално описание.**

Тази глава разглежда етапа на проектиране на базата данни и всички операции, които могат да се извършат с помощта на системата: обновяване на съдържанието, потребителски права, тестове, интерфейс за WAP и i-mode, изпращане на SMS, MMS и WAP Push SMS.

- **Глава 4 – Програмна реализация.**

Тук са описани алгоритмите и функциите, които реализира приложението. По-важните алгоритми са представени в блок-схеми.

- **Глава 5 – Ръководство за потребителя и приложения на проекта.**

В тази глава детайлно се описва начина на работа на системата, както и местата, където тя е приложена.

- **Заключение и възможности за разширение на проекта**

Равносметка на създаденото приложение и анализ на възможностите за подобрения.

- **Използвана литература**
- **Приложение**

Упражнения за курса по WAP/WML/XHTML във ФМИ.

Приложен е диск с програмния код на продукта.

Глава 1. Обзор на проблемната област и съществуващи реализации

1.1. Проблемна област.

Информационна система се нарича автоматична или ръчна система, която обхваща хора, машини и/или методи, организирани да събират, обработват, предават и разпространяват данни. [1].

В съвременния свят необходимостта от информираност е изключително важна за почти всички организации. Ето защо, информационните системи намират все по-голямо приложение. Днес, информацията е станала един от петте основни ресурса за успеха на едно предприятие, наред с човешкия ресурс, парите, материалите и машините. [2].

За да е успешна една информационна система, трябва да притежава следните основни качества:

- възможност за бързо и ефективно въвеждане на информацията;
- данните в нея да са правилно структурирани, така че да може да се извлича необходимата информация в последствие;
- адаптивност към нови условия;
- бърза и точна обработка на информацията;
- бързо разпространение на данните до оторизираните за достъп потребители;
- надеждна защита от неоторизиран достъп.

Настоящият проект, на име Comgate, поставя за цел удовлетворяването на тези качества като за постигането ú, той обединява в себе си качества на няколко основни групи системи:

Система за управление на съдържанието (Content Management System)

Представлява софтуерна система, използвана за управление на съдържание. Включва компютърни файлове, изображения, аудио файлове, електронни документи и Интернет съдържание. Идеята на CMS е да се използват тези данни както в офиса, така и извън него през мрежата [3].

Частен случай на такава система, за Интернет приложения, е т. нар. Интернет система за управление на съдържанието (Web Content Management System, съкратено WCMS). Използва се за съхранение, контролиране и публикуване на специфична документация в Интернет, напр. новини, ръководства за управление, технически ръководства, упътване за продажби и маркетингови брошури. Може да поддържа следните черти:

- вкарване и създаване на документи и мултимедийни материали;
- идентифициране на потребителите;
- потребители с различни роли;
- известяване за промени по съдържанието;
- създаване на различни версии;
- автоматизирано публикуване на съдържанието;
- отделено съдържание от форматиране и възможности за изменения във външния вид, без промяна на съдържанието [4].

Дефинират се два типа системи за управление на съдържанието:

а) статични – това са системи, които се използват за еднократно генериране на съдържанието (Интернет сайт);

б) динамични – системи, които улесняват потребителя в създаването на съдържанието (Интернет сайта) и след това могат да се използват за динамичното му обновяване.

Основните предимства на системите за управление на съдържанието са:

- универсалност – с могат да се прилагат за различни предметни области;
- възможност да се управлява съдържанието, без да са необходими специализиран компютърни познания;
- доказани са с времето;
- добър програмен код – най-често са разработени в продължителен период от време от екип от програмисти, затова са изчистени са от грешки в кода;
- професионална поддръжка и богата документация (главно за платените).

Като недостатъци на системите за управление на съдържанието могат да се посочат:

– изискват много повече ресурси (както всички универсални системи) ;

– изискват постоянен контрол на сървъра – при малки промени в сървъра е много вероятно да пропадне цялата CMS, която е настроена за него;

– изисква се поддръжка на софтуера от трети лица – в бъдеще може да се окаже, че системата не работи с нови версии на сървъра, на операционната система или поддръжката за съответната CMS вече не съществува;

– така създадените сайтове са по-трудни за откриване от търсачките (особено за безплатните WCMS) – изтъкват повече информация за системата, отколкото за съдържанието на сайта.

Настоящият проект притежава следните общи черти със система за управление на съдържание: позволява прилагане в различни предметни области, автоматизирано добавяне на текстове, подреждане в различни категории, файлове, тестове, потребители с различни права, отделено съдържание от форматиране.

Система за отдалечено обучение (Virtual Learning Environment)

Представлява софтуерен продукт проектиран да улесни преподавателите във воденето на курсове, основно като помага в управлението на курса. Системата може да документира показаните резултати от всеки от учениците по време на курса. Основно се използват като допълнение на класическото обучение „лице в лице” [5].

Обикновено тези системи включват в себе си:

– учебни материали;

– форуми;

– чат;

– тестове;

– упражнения;

– блог;

– RSS емисии;

– ограничение на достъпва до учебните материали;

– възможнст за администриране на учебните материали.

Comgate съдържа следните черти на система за отдалечено обучение: отделни права за гости, студенти и преподаватели, възможност за администриране на съдържанието, групиране на темите по категории и подкатегории, възможност за съставяне и решаване на тестове и статистики за резулттатите.

Система за мобилнен достъп

Представлява програмент продукт, който адаптира данните и ги доставя във вид, удобен за тяхното получаване до клетъчен телефон, PDA или друго мобилно устройство. Ако съдържанието се доставя през Интернет, то трябва да се адаптира за големината на дисплея, оперативната памет, процесора и скоростта на връзката на мобилното устройство.

Друга добра възможност за предоставяне на данни е през услугите на мобилния оператор. Почти всички съвременни мобилни апарати могат да използват услугата „кратки текстови съобщения" (Short Message Service – SMS), тя се предоставя от мобилния оператор с т. нар. център за изпращане на съобщения (SMS Center). А прилагането на услугата от трети лица може да стане с протокол за обмен на съобщения между центрове за изпращане на съобщения – short message peer-to-peer protocol (SMPP). Повече възможности дава услугата „мултимедийни съобщения" (Multimedia Messaging Service – MMS), при която освен текст може да се изпраща и изображение, звук и видео.

Като цяло, системите за мобилен достъп съдържат някои от следните възможости:

- Съръжание съвместимо с WAP или i-mode;
- активизиране на клиентското устройство за достъп до WAP от страна на сървъра – т. нар. WAP Push;
- доставяне на съдържание при заявка на клиента;
- асинхронно изпращане на кратки съобщения – SMS и/или MMS;
- адаптиране на e-mail за мобилно устройство – протоколи Blackberry и/или iMail;
- RSS емисии;
- доставяне на съдържание чрез интерактивно гласово меню – Interactive Voice Response (IVR).

Comgate съдържа следните черти на система за мобилен достъп: поддържа версия на Интернет сайт за мобилен телефон, приема заявки с SMS, предава информация с SMS, MMS, WAP Push SMS.

Следват примери за изброените системи. Ще отбележим, че Comgate не цели да притежава всички качества на продуктите, които ще опишем, а взима само част от предимствата на всеки от тях.

1.2. Съществуващи реализации на системи за управление на съдържание.

1.2.1. Joomla и Mambo

Joomla! е една от най-популярните съвременни системи за управление на съдържанието с отворен код, за публикуване на съдържание в Интернет или интранет. Създадена е на базата на кода на друга CMS – Mambo, след като част от нейните разработчици се отделят и решават да създадат нов самостоятелен продукт. Писана е на програмен език PHP с MySQL база данни. Първата версия на Mambo излиза през март 2000 г., а на Joomla – на 17 август 2005 г. И двата продукта са лицензирани под GPL лиценз.

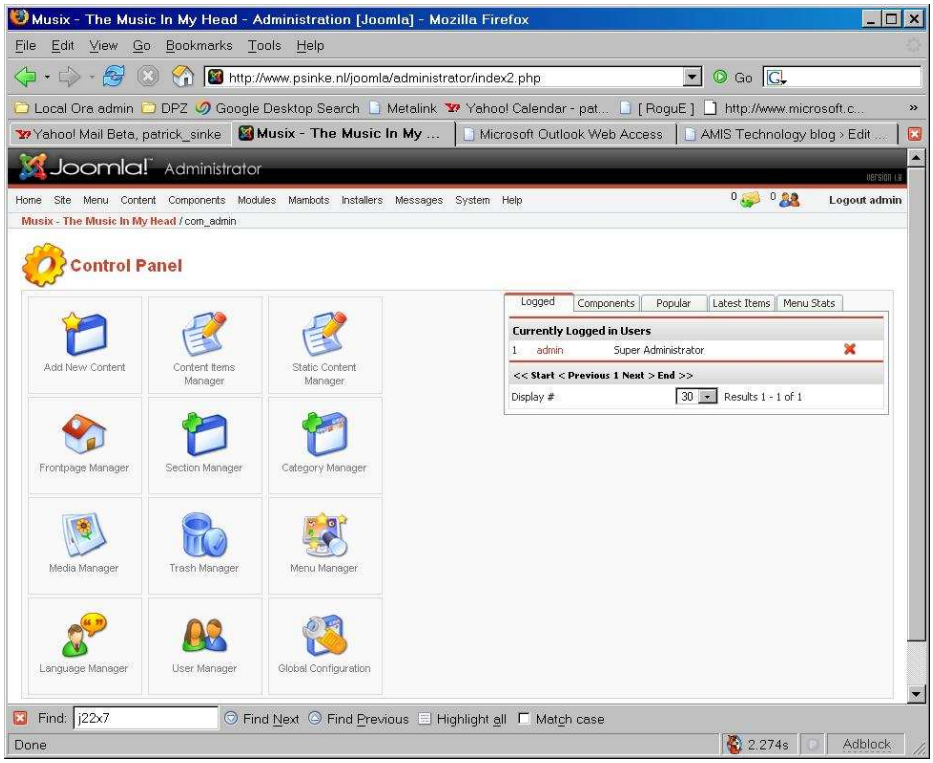
Възможностите на тези CMS са следните:

- Кеширане на страници за подобрение на производителността
- Създаване на RSS емисии
- Версии за печат на страниците,
- Създаване на флаш новини
- Блогове
- Анкети
- Търсачка
- Версии на различни езици
- WYSIWYG редактор – позволява редакторът да вижда реално как ще изглежда въвеждания от него текст (What you see is what you get) [6],[7].

Различните части на системата са създадени като модули. Това позволява лесно прилагане на допълнителни разширения, т. нар. plugins. Разширенията са разработени от различни хора за различни приложения. Реализирани са например разширения като WikiBot или WAP изглед. Всички разширения са достъпни на сайта на Joomla [6]. Отвореният код позволява и разработката на компоненти от различни

програмисти. Това са, например, компоненти за backup на сайта, календар, подобрения за по-лесно намиране на страницата и други.

Joomla ver 1.0 е създадена на базата на Mambo 4.5.2.3 с изчистване на някои грешки в кода и пропуски в сигурността. И двете програми продължават да поддържат еднакъв потребителски интерфейс и подобни настройки по подразбиране. Разлика има при реализирането на поддръжка на различни езици. При Joomla това се прави с ‘ini’ формат на транлиращите файлове, докато при Mambo използва функции ‘gettext’, които заемат повече ресурси. Тескстът в Joomla е изцяло кодиран в UTF-8 формат. Новост при Joomla са и някои нови възможности за атентификация като LDAP и Gmail както и някои клиент-сървър приложения. При Joomla се поддържа изцяло MySQL 5 както и възможност за мигриране към друга база данни [6].



Фиг. 1.1. Панел за управление на Joomla.

Сравнение с Comgate:

Comgate също е реализиран с технологиите PHP и MySQL. Това позволява приложение на различни сървъри и използване на голямо количество компоненти с отворен код. Подобна е и системата за редактиране, от администратора не се изискват програмистки познания, той вижда редактирания текст по начина, по който той ще се публикува след това. Реализирана е също форма за търсене, с която да се извлича вече въведен в системата текст. Инсталацията на Comgate и Joomla е също сходна – не се инсталират никакви програми, а само се вкарват модулите на

системата в директорията на уеб сървъра. Настоящото приложение не цели да постигне всички възможности на Joomla. Може да се приеме за предимство на Comgate, че е по-компактен от Joomla. Той е по-малък по обем и има по-малко модули, което го прави по-прост за настройка и работа.

1.2.2. Face Control

Face Control комерсиална система за управление на съдържанието, разработена от българската фирма Елидо ООД. Дава възможност на потребителите да публикуват и управляват информацията на уеб страниците си. За ползването на системата не се изисква специализирана компютърна грамотност. Всички секции и отделни информации, създадени с Face Control са изцяло динамични – могат да бъдат създавани или модифицирани от потребителя. Промени във визуализацията и графичния дизайн се осъществяват без да оказват влияние на структурата и съдържанието на страницата.

Системата се използва от над 250 български и чужди сайта, сред които сайта на сайта на правителството, сайта на RFI България, сайта на катедра Теоретична електротехника към ТУ и др.

Разработен е във версии Standart, Enterprise и Heavy-duty и се продава на цени от 600 до 5000 EUR, заедно с инсталация и поддръжка. [8].

Възможности:

- различни езици;
- прикачване на документи, картинки и флаш анимации;
- статистика за посещенията на страницата;
- автоматично изпращане на съобщения до абонираните потребители;
- регистрирани потребители и различни права;
- търсачка;
- кеширане и поддръжка на натоварване до 2000 заявки на минута (за Heavy-duty версията).

Face Control се инсталира директно на уеб сървъра като завършен продукт, без да е необходимо допълнително програмиране, тестване или настройване на програмния код.

Стравнение с Comgate:

Замисълът на Comgate също е да не се ангажират редакторите на страницата с компютърни познания. Подобна на Face Control е и системата за регистрация и права на потребителите в създадените сайтове. За разлика от Face Control, при Comgate не

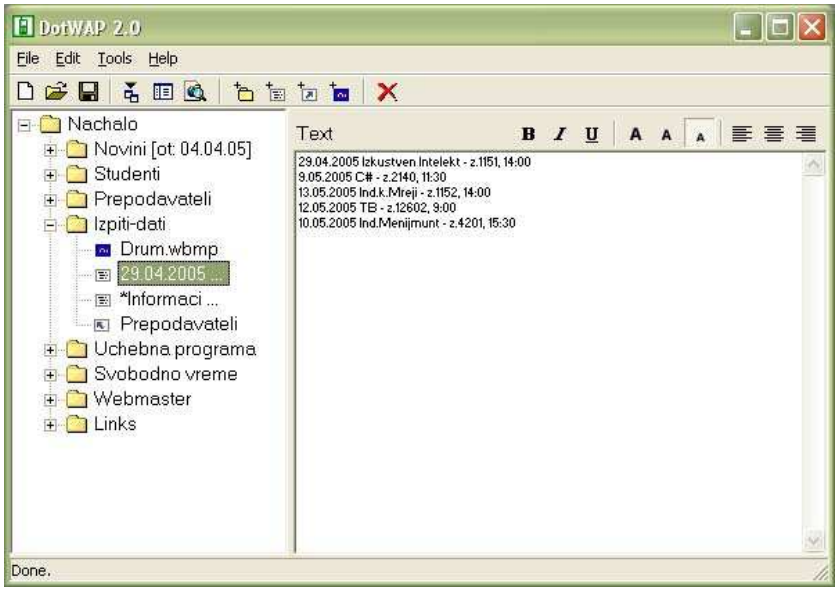
се изисква инсталация на специална програма на сървъра. Това прави Comgate независима от операционната система.

1.3. Съществуващи реализации на системи за мобилен достъп.

1.3.1. DotWAP

DotWAP е безплатен софтуер на фирма Inteis d. o. o. за публикуване на WAP съдържание. Представлява система за управление на съдържанието, която автоматично генерира WAP сайт с дървовидна структура и го публикува в хостинг на http://tagtag.com.

Потребителят инсталира програма на своя компютър, с която локално въвежда съдържанието на страницата. Интерфейсът е показан на Фиг. 1.2. Създават се множество директории (възли от дървовидната структура на сайта), които представляват вътрешни страници от сайта. Във всяка директория може да се поставят текст, картинки, външни препратки или поддиректории. Задават се предварително настройки за максимален размер на мобилната страница, като DotWAP автоматично разделя големите страници. DotWAP има и вграден емулатор, с който може да се види как ще изглежда създаваната WAP страница на мобилен телефон.



Фиг. 1.2. Интерфейс на DotWAP

Сравнение с Comgate:

Потребителят, който въвежда съдържанието в DotWAP не е необходимо да разбира от WML, нито от HTML. Същата е идеята и при Comgate. И в двата случая мобилното съдържание се генерира автоматично. Но за разлика от Comgate, при DotWAP се генерират статични WML страници с произволни имена, които в този вид се качват на сървъра. Това затруднява съставянето на вътрешни линкове извън дървовидната структура или външни препратки към произволно вътрешно съдържание на страницата. Друг недостатък на DotWAP е невъзможността да се правят промени по дизайна на генерираните мобилните страници.

1.3.2. focus-news.com

Сайтът на информационна агенция Фокус (http://www.focus-news.net/) е един от най-популярните портали за новини в България. Една от причините за това е възможността за достъп до информацията както през компютър, така и през мобилен телефон. Сайтът притежава web, WAP и i-mode версии както и възможност за поръчка на съдържание с SMS.

Фиг. 1.3. Изглед на WAP версията на focus-news през Openwave Emulator

Съдържанието на WAP и i-mode порталите е разделено на няколко групи, които са видими при зареждането на страницата. Всяка новина се отваря в нова страница, ако съдържанието е прекалено голямо, тя се разделя на няколко страници. Потребителят може да отваря различните линкове както с бутона „Accept” на своя телефон, така и с цифровите бутони от клавиатурата.

За да получи новина чрез SMS, потребителят трябва да изпрати SMS от своя мобилен телефон до определен номер. Поддържа се абонамент за SMS новини. Абонираните потребители не е нужно да изпращат заявка с SMS, а получават SMS-и при всяка излезла новина от желания от тая критерий. В този случай се заплаща за входящ SMS.

Сравнение с Comgate:

Comgate притежава версия за мобилни телефони, много близка до тази на focus-news. По аналогичен начин големите страници се разделят, за да могат да се заредят от мобилните телефони. Мобилната версия при Comgate поддържа и сесии, което позволява автентификация на потребителите и достъп до различно съдържание

на различните потребители. Изпращането на SMS във focus-news се извършва през SMS центровете на мобилните оператори в България. При Comgate изпращането на SMS и MMS се извършва с SMPP протокол и международен SMS център. Това позволява централизирано управление на услугата и възможност за изпращане на съобщения до мобилни телефони извън България.

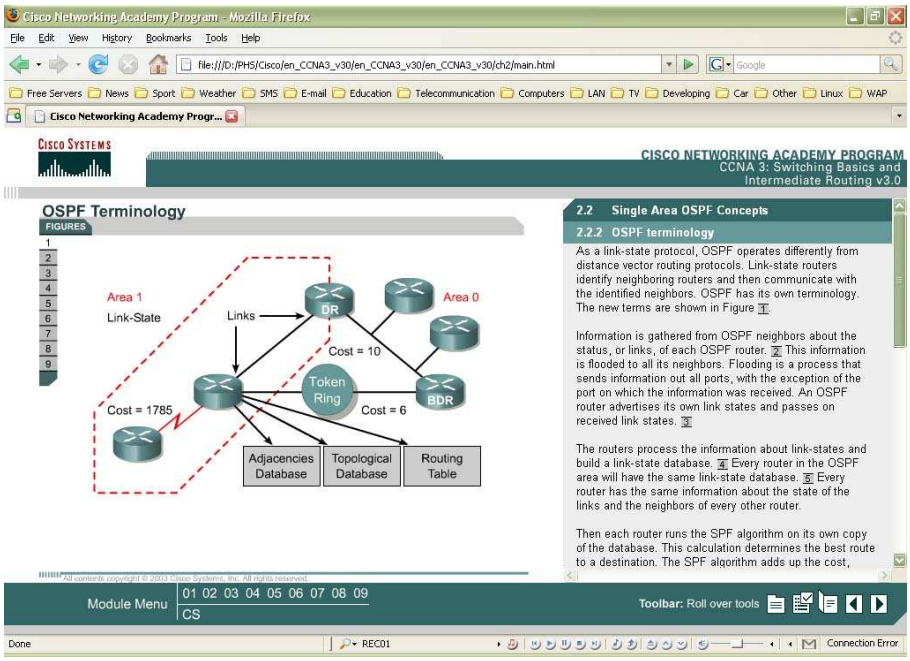
1.4. Съществуващи реализации на системи за отдалечено обучение.

1.4.1. Cisco Networking Academy

Система за представяне на материали, упржнения и тестове за студенти в мрежовите академии на Cisco Systems (http://netacad.cisco.com). Системата не е публична, ето защо можем да ра разгледаме част от възможностите ú.

Притежава поне три типа потребители – администратори, преподаватели и студенти. Администраторите (вероятно разделени на различни групи) управляват съдържанието в системата. Те качват нови версии на учебните материали, създават нови учебни курсове или редактират съдържанието на вече съществуващите материали. Задачата на преподавателите е да съставят и управляват класове от студенти. В един клас има група студенти, които в даден момент изучават един и същ учебен курс. Преподавателите могат да изключват или добавят студенти в класа. Те притежават информация за представянето на всеки студент и избират критерий за оценяването на студентите и преминаването им в следващ курс. Преподавателите имат достъп и до допълнително съдържание от Cisco, необходимо им за провеждането на курсовете и упражненията на живо. Студентите получават необходимите материали за завършването на курса. Студентите полагат тестове през системата, които служат за самоконтрол и за оценяване на представянето в курса.

За взимането на сертификат на Cisco е необходимо да се преминат няколко курса по определен материал. Съдържанието на всеки курс е разделено на определен брой модули. Всеки модул се състои от около 20-30 теми, групирани в няколко глави. На един екран на компютъра се показва съдържанието на текущата тема. (Фиг. 1.4)



Фиг. 1.4. Интерфейс на Cisco Network Academy

Освен текста, на екрана се представят пояснителни картинки и анимации. Всеки модул завършва с тестови въпроси за самоподготовка, които са подобни на тези, които биха се паднали на финалния изпит. Финалният изпит е също под формата на тест, като резултатът се определя като процент от верните отговори. След приключването на теста, студентът може да види точно кои теми и от кои модули са направените му грешки, за да ги прочете отново.

Сравнение с Comgate:

Работата на Comgate е много близка до системата на Cisco за определен курс. По аналогичен начин курсът е разделен на модули и теми и има тестове за всеки модул. За разлика от Cisco, които целят един курс да е достъпен за много класове в различни държави и преподаватели, целта на една инсталация на Comgate е да е в помощ само за един преподавател, в един курс. Преподавателят сам решава какво съдържание да публикува и може да го изменя по време на работата в зависимост от личните си виждания и резултатите на студентите. Така се избягва основния недостатък на Cisco академиите: бавното обновяване и тромаво отстраняване на неточности в материалите и тестовете.

1.4.2. ARCADE

ARCADE (Architecture for Reusable Courseware Authoring and DElivery – архитектура за многократно използване, съставяне и предаване на учебни

материали) е система за електронно обучение използвана от студенти и преподаватели във Факултета по математика и информатика в Софийския университет (http://62.44.100.133:8080/arcade/index.jsp). Комбинира уеб-базирани средства за управление на курсове и учебни материали, управление на потребителите, средства, улесняващи комуникацията, онлайн задачи за самостоятелна работа, както и тестване. Разработена е с UML технология, което дава възможност за лесно разширение на системата.

Аркейд поддържа пет различни групи потребители – студенти, инструктори, автори на курсове, курсови администратори, системни администратори, които съответстват на главните участници в образователния процес. Включването на потребителите към някоя от тези групи им позволява да ползват съответната функционалност.

Системата предлага на участниците в образователния процес различни средства, улесняващи комуникацията – вътрешна електронна поща, форум, чат, съобщения и виртуални места за съхранение на документи.

Сравнение с Comgate:

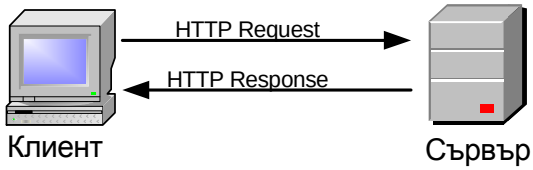
И Аркейд, и Comgate поддържат публикуване на учебни материали на страницата на курса. Предимство при Comgate е по-гъвкавия метод за управлението на съдържанието, който позволява материалите не само да се свалят от Интернет, а и да се четат он-лайн през страницата на курса. Предимство на Аркейд е поддръжката на форум, чат и виртуална кутия за сърхранение на документи. Като допълнителни средства за комуникация, Comgate предлага SMS, WAP Push SMS, MMS и достъп до съдържанието през мобилен телефон.

Глава 2. Технологии за реализация

2.1. Технологии за предаване на данни през Интернет

2.1.1. Основни концепции

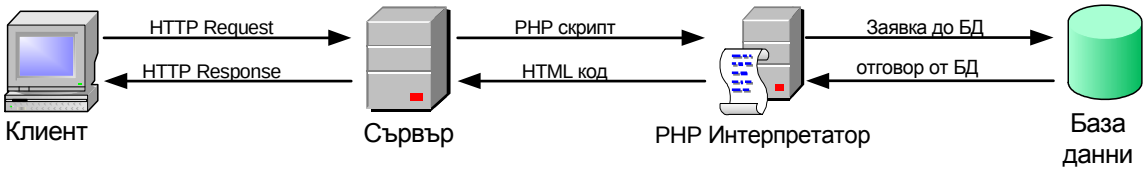
Интернет приложението представлява програмна система, работеща на уеб сървър и предоставяща на потребителите уеб базиран интерфейс, който се визуализира от уеб браузърите им. Комуникацията между потребителските браузъри и приложението на сървъра се осъществява на базата на протокола HTTP (Hyper Text Transfer Protocol) или WSP (Wireless Socket Protocol).



Фиг. 2.1. Връзка между уеб-браузър и уеб-сървър

HTTP и WSP са протоколи от приложно ниво на TCP/IP модела за комуникация, които се използват за реализиране на услугите WWW (World Wide Web) и WAP (Wireless Application Protocol) в Интернет. Клиентите изпращат заявка за определен ресурс от сървъра. Той от своя страна връща отговор с искания ресурс (ако в възможно да го достави) или код за грешка. (Фиг. 2.1.)

Динамичните Интернет страници най-често осъществяват достъп до база данни. Общата структура на уеб приложение на езика PHP с база данни е представена на фиг. 2.2.



Фиг. 2.2. Архитектура на PHP приложение с база данни

Потребителският уеб браузър изпраща HTTP заявка за определена уеб страница. Уеб сървърът получава заявката, извлича файла и го подава на PHP интерпретатора за обработка. При обработката на скрипта, ако се срещне команда за за свързване с базата данни и обработване на заявка, се отваря конекция до сървъра за база данни (напр. MySQL сървър) и се изпраща съответната заявка. Заявката се

обработва и се изпраща обратно отговор до интерпретатора. Крайният резултат от работата на интерпретатора е HTML код (или някоя от разновидностите за хипертекст), който се връща на сървъра и той от своя страна го праща на клиента с HTTP отговора. [9].

Предимствата от този модел са едновременно за клиентската и за сървърната страна на приложението. През Интернет се предава за клиента само изходния резултат от скрипта. По този начин се пести трафик и се увеличава скоростта на зареждането на приложението от страна на клиента. Резултатът от сложна обработка на данните може да се визуализира безпроблемно и от устройства със слаби системни показатели и бавна връзка към Интернет. Независимо от използваната платформа, необходим е само уеб браузър за достъп до ресурсите. Приложението не може да осъществява достъп до паметта и други ресурси от компютъра на потребителя, ето защо е напълно безопасно за него. От друга страна, самият скрипт остава скрит за крайния потребител. Така кодът върху сървъра е защитен от външните потребители, без да има опасност от недобронамерена промяна или кражба на интелектуална собственост.

2.1.2. Уеб сървър

Уеб сървърът е програма, която използвайки модела клиент-сървър и приложния протокол за комуникация HTTP, обслужва файловете, които формират Уеб страниците за потребителите на Интернет. Компютрите на потребителите съдържат HTTP клиенти (уеб браузъри), които изпращат заявки към уеб сървъра. [10].

Съществуват множество варианти за Интернет сървър в зависимост от платформата, на която ще се инсталира – Microsoft IIS, PWS, Omni HTTPd, Tomcat и др. Най популярен, използван от над 2/3 от Интернет сайтовете по света, според изследване на Неткрафт ([11]) е сървърът Apache.

Web сървърът Apache е създаден от The Apache Group. Той е проект с отворен код. Работи както под Unix операционни системи, така и под Windows. Безпроблемно може да се стартира приложение с Apache, PHP и MySQL на машина под Windows и след това да се прехвърли в Linux без почти никакви промени в скриптовете. Тъй като Apache е с отворен код, всеки може да пише програмен код, който да разширява функционалността му. [12].

Apache включва пълен HTTP уеб сървър, създаден да обслужва бързо динамично съдържание. В Apache, PHP може да бъде компилиран като модул и съответно уеб приложение комбинацията от Apache и PHP е по-стабилна и по-надеждна от останалите. Бързаната и стабилната работа са основните причини този проект да се препоръчва да работи върху Apache. И все пак, при коретно стартиране на PHP и MySQL, Comgate може да работи без проблемно и върху други платформи.

2.1.3. База данни

Типа на базата от данни е определен на базата на планирането на конкретното приложение с оглед на възможности, поддръжка и бъдещо развитие. Съществуват две основни направления при базите от данни. Единият тип са така наречените бази от данни от по-нисък клас, към които спадат Microsoft Access, FoxPro и др. Те не изизскват специална поддръжка и мощен хардуер, но производителността там е ограничена и не могат да се обработват много голямо количесво данни. За настоящия проект такова условие е неприемливо. Другият тип бази данни са т. нар. сървъри за баси данни – MySQL, Postgre SQL, Microsoft SQL, Oracle и др.

Базата данни MySQL поддържа сравнително по-малко данни от останалите (пр. Oracle), но е лесна за управление, разпространява се безплатно, поддържа SQL и многопотребителска работа, няма изисквания за операционната система на сървъра, работи стабилно и надеждно с Apache уеб сървър и е достатъчно бърза и мощна за малки и средни бази от данни. MySQL е силно мащабируема и лесна за администриране [12]. Не е необходим опитен администратор на база данни, нито за инсталирането на MySQL, нито за нейното управление. MySQL поддържа клиентски приложни интерфейси за широк набор от програмни езици (като Perl, C, PHP и т.н.), Клиентските програми, извършващи достъп до данните в MySQL база данни, могат да се напишат с помощта на тези интерфейси, това позволява безпроблемна работа на настоящото приложение, както и безпроблемно мигриране към друг програмен език.

Релационната база данни MySQL се основава на езика SQL. Език за програмиране, проектиран специално за операции с бази данни. Той е общ език за много машини за бази данни. Общата форма на SQL се нарича ANSI-SQL. Както всеки друг език за програмиране, той има типове данни, ключови думи и оператори. SQL се използва след като се свързва с базата данни, за да се управляват таблиците и съдържаните в тях данни.

MySQL е многопотребителски, многонишков сървър, който осигурява бърз достъп, както и ограничения на достъпа само на оторизираните за целта потребители.

MySQL съхранява всяка таблица като отделен файл в директорията за базата данни. Максималният размер на една таблица може да бъде от 4GB до максималният размер на файл, поддържан от използваната операционна система.

2.1.4. Програмен език от страна на сървъра

За настоящото приложение е необходим език от високо ниво за създаване на скриптове от страна на сървъра, който работи с база данни, прави необходимите обработки и изчисления и генерира като резултат различни скриптове, които се визуализират от страна на клиента. Съществуват множество програмни езици за целта – CGI, Perl, PHP, ASP, ASP.NET, Java/JSP и др. Възможностите на съвременните версии на тези езици са почти изравнени и затова изборът на език от страна на сървъра до голяма степен е субективен и се определя от възможностите на програмистите, както и наличния хардуер и софтуер.

PHP (PHP Hypertext Preprocessor) е сървърен скриптов език, проектиран специално за създаване на Интернет базирани приложения. PHP кодът се вгражда в (X)HTML или WML страница и се изпълнява при всяко нейно зареждане. Кодът се интерпретира от уеб сървъра и генерира изходни данни, които посетителят на сайта вижда.

PHP е създаден през 1994 г. от Рамус Лерфорд като продукт с отворен код. Оригиналният код е достъпен за използване, промяна и разпространение безплатно. Така, макар да е създаден само от един човек, той непрекъснато се усъвършенства, за да се превърне днес в един от най-популярните и ефективни езици за създаване на уеб приложения.

Основните фактори, поради които PHP е избран за разработването на настоящото приложение са следните:

– **Производителност**. PHP значително надминава по производителност своите конкуренти. Той се справя отлично с обслужване на много посещения едновременно дори хардуерът на сървъра да не е достатъчно бърз.

– **Интеграция с база данни.** PHP притежава интерфейс за много различни системи бази данни.

– **Преносимост.** PHP може да работи еднакво върху UNIX-базирани операционни системи и върху Windows.

– **Вградени библиотеки.** PHP е проектиран специално за работа с уеб и за целта притежава голямо количество вградени библиотеки и функции за решаването на стандартни задачи.

– **Отворен код.** Потребителят може сам да добавя нови функционалности на езика. Самият продукт е безплатен.

– **Наличен код.** Съществува огромно количесгво приложения разработени на PHP, които се разпространяват безплатно в Интернет. [12].

2.1.5. Скриптове при клиента

HTML и XHTML MP

HTML (HyperText Markup Language) представлява описателен език за представяне на форматиран текст. Създаден е като част от Интернет услугата World Wide Web от Тим Баренс-Лий.

HTML документите представляват изцяло текстови файлове, като в тях освен текста, който съдържат са вмъкнати и инструкции за форматиране наречени тагове, които указват как точно да се изобрази текста по време на визуализацията. Могат да се указват хипервръзки, които да сочат към произволни отдалечени ресурси, зададени чрез URL (Uniform Resource Location)., за да могат потребителите лесно да преминават през различни документи без да въвеждат тяхните Интернет адреси.

Стриктното спазване на стандартите на езика е от основно значение за еднаквото представяне на HTML страниците в различните браузъри. На базата на HTML и XML (eXtensible Markup Language) е създаден XHTML, който е съвместим с HTML, но стриктно спазва правилата за валиден XML.

За представяне на съдържание за браузърите при съвременните мобилни телефони се използва езика XHTML MP (eXtensible HyperText Markup Language Mobile Profile). Той е описателен език, дефиниран в спецификацията WAP 2.0 на Open Mobile Alliance (OMA). Използва се заедно с WAP CSS (WAP Cascading Style Sheet или WCSS).

XHTML MP е подмножество на XHTML, създаден основно на базата на XHTML Basic с допълнителни елементи и атрибути, както и с възможност за използване на

каскадни стилове. Основна цел на XHTML MP е да обедини технологиите за създаване на страници за мобилни телефони с тези от World Wide Web.

CSS и WCSS

Cascading Style Sheets (CSS) представлява език за описание на начина на визуализация и позициониране на елементите на HTML. Разработен е от W3C(World Wide Web Consorcium).

Чрез CSS се дефинират стилове, които се използват след това в HTML докумнтите за форматиране на текста. При необходимост форматирането на един HTML или няколко документи, използващи CSS може бързо и лесно да се променят, като се променят само стиловете в CSS файла без да се променят HTML файловете. [9]

Подмножество на CSS е WCSS (от WAP Cascading Style Sheets, или Wireless Cascading Style Sheets или Wireless Profile Cascading Style Sheets). Представлява дефинирани в WAP 2.0 каскадни стилове, които се използват с XHTML MP, базиран на CSS. [13].

JavaScript и ESMP

JavaScript е скриптов език, който се изпълнява от уеб браузъра на потребителя и позволява динамична манипулация на елементите на визуализираните в браузъра HTML документи. Разработен и приложен първоначално от Netscape Communications. рограмният код, написан на JavaScript, се вгражда в HTML документите или се записва във външен файл и се изпълнява от интерпретатора на JavaScript при определени събития. От съображения за сигурност JavaScript има достъп до ресурси, които са свързани с визуализираните в момента HTML документи в браузъра. От JavaScript не може да се отварят файлове от диска на клиента, да се използват сокет връзки, да се комуникира с други приложения и т. н.

Първоначално Microsoft решава да разработи своя версия на езика – JScript. Едновременното развитие на двата езика води до големи несъвместимости между браузърите и техните различни версии. Браузърът Opera, от своя страна, пренебрегва част от скриптовете с цел по-бързо зареждане на страниците.

Европейската асоциация на производителите на компютри (European Computer Manufacturers Association – ECMA) се оптива да въведе общ стандарт за скриптовете на различни разработчици – т. нар. ECMA Script. Въпреки появилите се общи стандарти, все още има големи различия в интерпретирането на езика от

различните браузъри и е необходимо внимателно използване или съставяне на различни версии на едно приложение, които да работят за различните браузъри. [9]

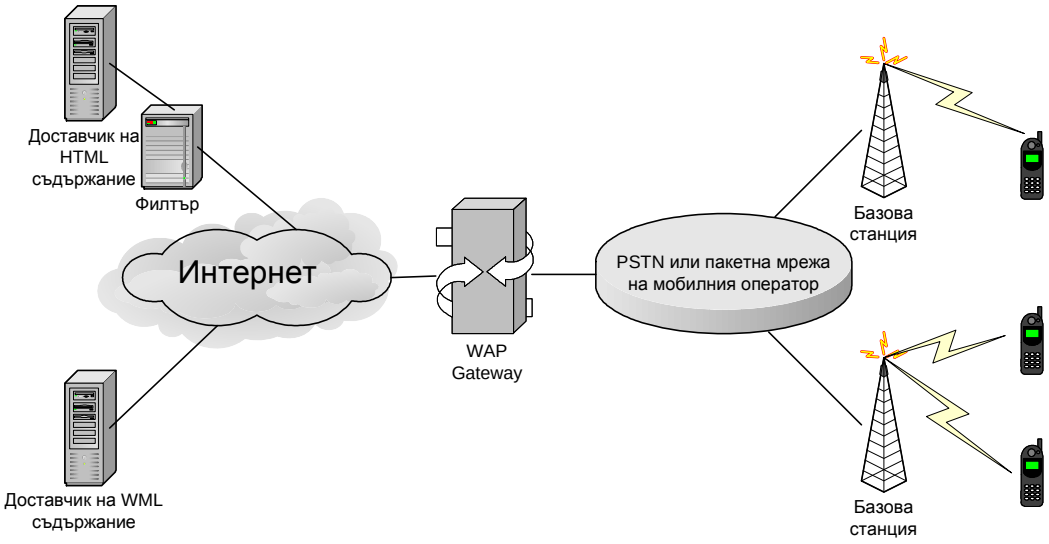
Във версия 1.1 на езика за мобилни браузъри XHTML MP се допуска използването на скриптов език и от страна на клиента. В спецификацията на XHTML MP ([14]) се препоръчва използването на скриптов език ECMAScript Mobile Profile (ESMP). Ако клиентската страна поддържа скриптови езици, то непременно един от тях трябва да бъде ESMP. По ESMP е много близък до JavaScript и поддържа цялата функционалност на по-стария език за клиентски скриптове при мобилни телефони – WMLScript.

2.2. Технологии за предаване на данни в мобилна мрежа

2.2.1. Достъп през мобилен браузър – WAP и i-mode

WAP (Wireless Application Protocol) представлява стандарт за пренос на информация до потребители на мобилни устройства и същевременно платформа за съответния тип приложения, приет първоначално от Ericsson, Nokia, Motorola и Unwired Planet и днес стандартизиран от WAP форум (www. wapforum.org). [15]. Версия 1.0 на стандарта излиза през 1998 г., текущата версия е 2.0, публикувана през 2001 г.

Архитектурата на стандарта WAP е представена на Фиг. 2.3.



Фиг. 2.3. Архитектура на WAP

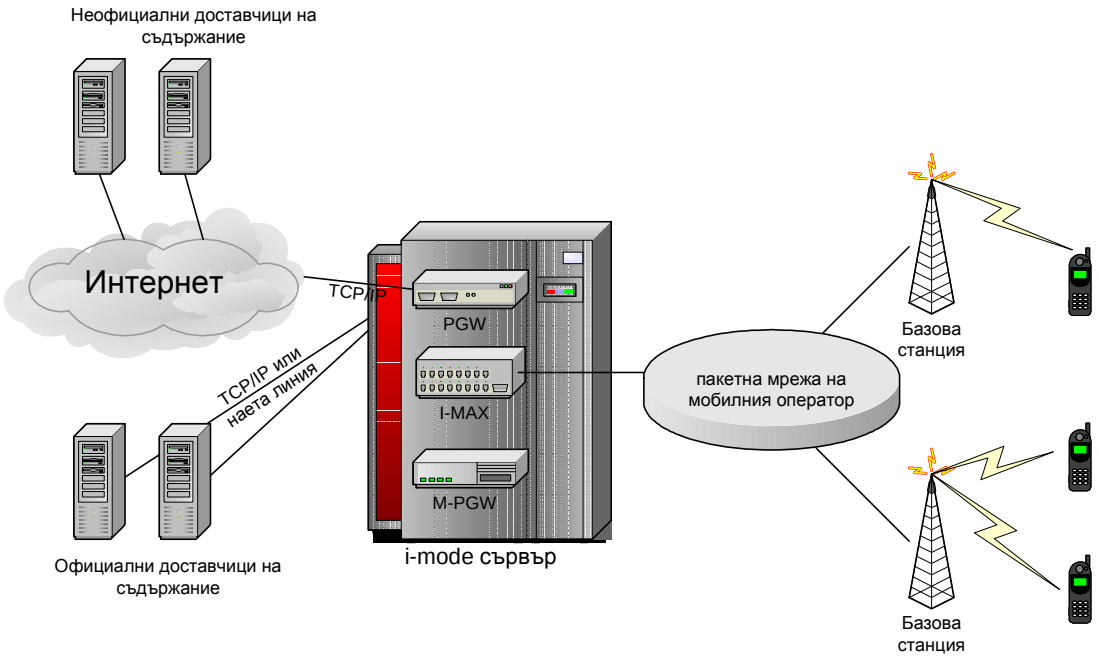
Мобилните станции обменят информация с доставчиците на съдържание посредством т. нар. WAP Gateway. WAP гейтуея е свързан от една страна с мрежата Интернет, а от друга – с мрежата на мобилния оператор. Съдържанието се намира на сървъри, които са отдалечени и общуват чрез TCP/IP с WAP Gateway. Мобилните станции правят заявка към Гейтуея. Той, от своя страна, предава заявката към уеб сървъра и връща отговорът му към мобилното устройство. Ако уеб приложението извежда HTML, то трябва да премине през филтър и да се преобразува на WML или XHTML MP, както и да се намали обема му, така че да може да бъде прието от мобилното устройство.

I-mode (Information mode)

Стандарт за доставяне на мобилно съдържание, разработен от японския мобилен оператор NTT DoCoMo през 1999 г. Целта му е да осигури по лесен начин съдържание на потребителите, без да се изискват от тях да правят настройки или да имат познания по Интенет. Базиран е на собствен протоколен стек и работи само с i-mode сертифицирани телефони като постига по-ефективно използване на капацитета на мобилната мрежа.

На ниско ниво се разчита на пакетна мрежа на мобилния оператор (напр. GPRS, PHS и др.). Разработен е собствен протокол на транспортно ниво – TLP (Transport Layer Protocol), който за радио-пакетни мрежи е по-ефективен от стандартния за Интернет TCP. При TLP ефективността се постига с по-голямо съотношение на данните към header полетата на протокола. Премахнати са голяма част от хедърите от TCP, които са ненужни за i-mode. Премахнат е също така тристъпковия handshake за установяване на сесия, тъй като при i-mode данните не се изпращат през Интернет, а локално до i-mode сървър през високонадеждна мрежа. [16].

Архитектурата на i-mode е представена на фиг. 2.4.



Фиг. 2.4. Архитектура на i-mode

Мобилните станции обменят информация с i-mode сървър през пакетната мрежа (GPRS) на своя оператор. I-mode сървърът включва в себе си няколко компонента, по-важните от които са:

- PGW (Packet Gateway Module) – гейтуей за пакетите, който извършва филтриране, компресиране и преобразуване на съдържанието от Интернет за i-mode мрежата.

- M-PGW (Message Packet Gateway Module) – гейтуей за съобщенията, използва се за услутите iMail и MMS през i-mode.

- I-MAX (Interface-Mobile Access Exchanges) – интерфейс за връзка с мобилните станции. Предава и приема заявки от мобилните станции, както и вече преработените данни. [16].

Доставчиците на съдържание в i-mode се разделят на два вида – официални и неофициални. Официалните сайтове са проверени и одобрени от мобилния оператор предоставящ i-mode услугата. Сървърите на тези сайтове обикновено са при моблиния оператор или до тях има наета линия или друг тип гарантиран капацитет на връзката. Това гарантира бърз и надежден обмен на данни между i-mode сървъра и тези доставчици на съдържание. Линкове към тези сайтове има на основното меню на i-mode портала. Всички други достачици на съдържание се свързват с i-mode сървъра през Интернет и се наричат неофициални. Достъп до тях клиентът извършва

чрез въвеждане на адреса им в своя мобилен браузър. Настоящето приложение може да се достъпва през i-mode като неофициален сайт.

2.2.2. Технологии за изпращане на съобщения

Настоящият проект реализира изпращане на съобщения от сървъра към мобилни телефони чрез услугата „Кратки текстови съобщения" на мобилните оператори.

SMS

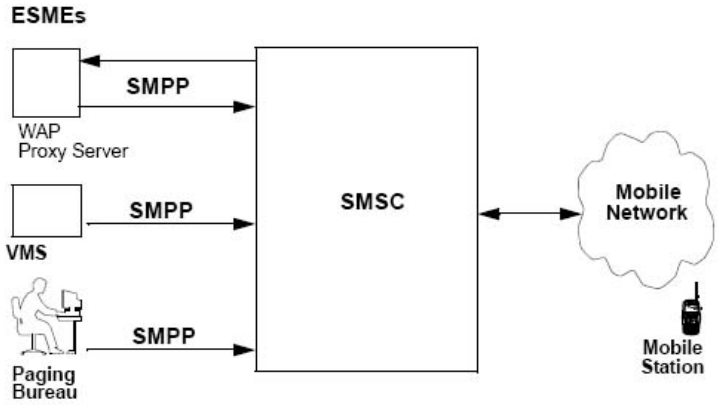
Short Message Service (SMS) е услуга на мобилните оператори за предаване на кратки съобщения към и от мобилните телефони. През 1985 г. е включен в стандарта GSM [17]. Предаването на съобщения се осъществява от Short Message Service Center (SMSC) чрез частта MAP (Mobile Apllication Part) на стандарта SS7. Стандартната големина на едно съобщение е 1120 бита. При стандартното GSM кодиране (GSM 03.38) всеки символ се представя със 7 бита, което позволява големина от 160 стандартни символа. За нестандартни азбуки (напр. кирилица) се използва Unicode кодиране от 16 бита за символ, което позволява до 70 символа за едно съобщение.

Стандартът SMS се поддържа от организацията 3GPP (3rd Generation Partnership Project).

SMPP

Short Message Peer to Peer Protocol (SMPP) е телекомуникационен протокол за обмен на кратки текстови съобщения. Той представлява интерфейс между центърът за изпращане на кратки текстови съобщения (SMSC) и външен доставчик (ESME – External Short Message Entity). Разработен е от компанията Aldiscon (закупена по-късно от Logica) и стандартизиран през 1999 г. от организацията SMS Forum (www.smsforum.net). В момента най-разпространена поддръжка има SMPP версия 3.3, а най-новата версия на протокола е 5.0.

Съществуват несъществени разлики във версиите на протокола. Най-голямото подобрение е направено във версия 3.4. Там е добавена възможност за предаване и изпращане на съобщения в една и съща сесия. За постигане на целите си и за най-голяма съвместимост с международните SMS центрове, настоящото приложение използва SMPP версия 3.4.



Фиг. 2.5. Приложение на SMPP в мобилните мрежи

Чрез SMPP се установява сесия между ESME и SMSC, чрез обмяна на протоколни единици (Protocol Data Units (PDUs)) за заявки и отговори (request/response). Протоколът работи върху TCP/IP и X.25 мрежи. [18].

На фиг. 2.5 са представени основните случаи за използване на SMPP.

– При услугата WAP, когато за носител на данните се използва SMS (много рядко се използва вече, основно WAP върви през CSD или GPRS)

– При услугите Гласова поща или Пропуснати повиквания. Когато разговор с даден абонат не може да бъде осъществен (телефонът му е изключен или извън обхват), централата генерира SMS с известяване за пропуснатото повикване, което се изпраща, когато мобилната станция на абоната се регистрира отново в мрежата.

– За т. нар. Paging услуги. При настъпване на дадено събитие се изпраща SMS към група потребители, абонирани за известяване. Това е приложението на SMPP и в настоящия проект.

Сесията между ESME и SMSC може да има следните състояния:

- OPEN – съществува връзка на мрежово ниво, но все още няма заявка на сесия.
- BOUND_TX – ESME може да изпраща съобщения
- BOUND_RX – ESME може да получава съобщения
- BOUND_TRX – ESME може да получава и изпраща съобщения в една и съща сесия (за SMPP v.3.4 или по-висока)
- CLOSE – затворена е сесията.

По-важните PDU на SMPP са следните:

▪ `bind_transmitter` – заявка от ESMP за отваряне на сесия в SMSC за изпращане на съобщения

▪ `bind_transmitter_resp` – отговор от SMSC на заявка `bind_transmitter`.

▪ `bind_receiver, bind_receiver_resp` – за сесия за получаване на SMS.

▪ `bind_tranceiver, bind_tranceiver_resp` – за отваряне на сесия за получаване и изпращане на съобщения.

▪ `unbind` – за затваряне на сесия.

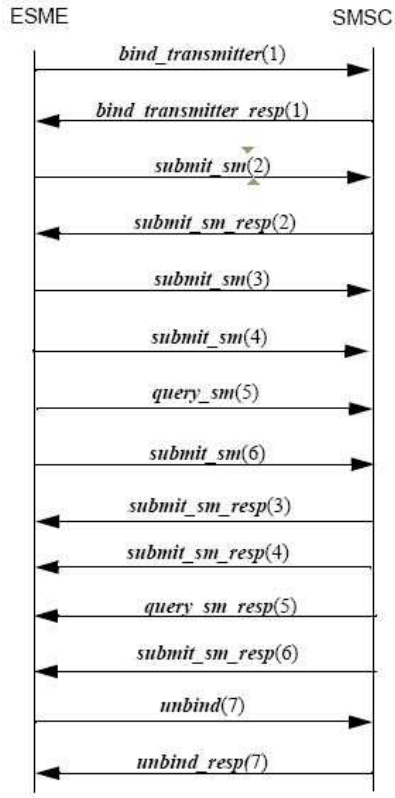
▪ `submit_sm` – използва се от ESME за предаване на съобщение за изпращане. Съдържа полета за подаващ номер и получаващ номер. Ако не се попълни подаващ адрес, SMSC ще сложи своя номер там. В този PDU се предава и текста на самото съобщение, както и периода на валидност, големина, кодиране, дата на получаване и др.

▪ `submit_sm_resp` – отговор от SMSC, че е получил заявка за изпращане на съобщение.

▪ `submit_multi, submit_multi_resp` – PDU-та за предаване на съобщение до много потребители едновременно.

▪ `deliver_sm, deliver_sm_response` – PDU за предаване на съобщение от SMSC към ESME.

▪ `data_sm, data_sm_resp` – PDU за предаване на данни – за случаите, в които се използва SMS като носител на данни за WAP.



Фиг. 2.6. Примерна SMPP сесия за изпращане на SMS

▪ `enquire_link, enquire_link_resp` – използват се и от ESME, и от SMSC за проверка на комуникационната връзка между тях на приложно ниво.

Примерна сесия между ESME и SMSC за изпращане на съобщене е представена на фиг. 2.6.

WAP Push

WAP Push представлява специално кодирано съобщение, което включва линк към WAP адрес. Може да се предаде с различни носители, сред които GPRS и SMS.

В повечето GSM мрежи, обаче не се поддържа активация на GPRS от страна на оператора, затова WAP Push се предава през SMS.

Съществуват два типа WAP Push съобщения.

– Service Indicationa (WAP Push SI) – известяват потребителя, че може да достъпи WAP съдържание.

– Service Load (WAP Push SL) – принуждават телефона автоматично да отвори съдържанието, без предупреждение. Не винаги работят, поради различни нива на защита на мобилния телефон.

WAP Push съобщенията са във вид на XML. Стандартно такова съобщение има следния вид:

```
<si>
  <indication href="<адрес към който насочва>" action="<вид на
индикацията>">
    Текст на полученото съобщение
  </indication>
</si>
```

Тъй като XML генерира много голям обем служебна информация (имена на елементи и атрибути), SMS съобщенията се кодират в двоично в WBXML (WAP Binary XML).

WBXML (WAP Binafy eXtensible Markup Language)

WBXML (WAP Binary XML) представлява бинарен формат на XML, който позволява компресиране на документа за пренасяне без да се измени същинската структура на XML. Мобилните устройсва поддържащи WAP Push имат вградено WBXML DTD и могат да разбират и възпроизвеждат предаваната информация.

Компресирането се постига по два начина. Всеки елемент се кодира със специфичен код и се пренася само кода, вместо името на елемента. Например, за WAP Push SI елементите са кодирани по следния начин:

Таг	Двоична стойност
si	0x45
indication	0xC6
info	0xC7
item	0XC8

Таблица 1.1. Кодиране на елементите в WBXML.

Друг метод за компресиране е да се заменят основните атрибути и известни стойност с двоични стойности. Например, атрибутът `action` на елемента `indication`, който показва нивото на индикация има няколко изброими стойности, които се кодират по следния начин:

Атрибут	Двоична стойност
`action="signal-none"`	`0x05`
`action="signal-low"`	`0x06`
`action="signal-medium"`	`0x07`
`action="signal-high"`	`0x08`
`action="signal-delete"`	`0x09`

Таблица 1.2. Кодиране на атрибути в WBXML

Като резултат XML документите могат значително да се компресират, така че да се изпрати целия документ с един SMS (1120 бита). [19].

MMS (Multimedia Messaging Service)

Стандарт за съобщения през мобилен телефон, който позволява изпращане на съобщения, които съдържат мултимедийна информация – картинки, аудио, видео, форматиран текст. Съдържанието се форматира във вид на SMIL който се интерпретира от мобилния телефон, поддържащ MMS. Предава се на мобилния телефон чрез WAP Push SMS.

SMIL (Synchronized Multimedia Integration Language)

XML-базиран език за представяне на поточни данни с комбиниране на аудио, видео, изображения и текст. Позволява съставяне на поредици и едновременно изпълнение на няколко потока като по този начин могат да се съберат отделните обекти в един единствен документ. С помощта на SMIL се реализира услугата MMS (Multimedia Messaging Service), която позволява предаване на мултимедийни съобщения между клетъчни телефони или между компютър и клетъчен телефон. [20]

Глава 3. Функционално описание

3.1. Общи изисквания към системата

Настоящата система трябва да се изработи така, че да притежава възможност отдалечен достъп и управление на голямо количество информация. Представянето трябва да е независимо от предметната област и достъпно през Интернет и мобилен телефон. Необходимо е да се предвидят средства вкараната в системата информация да може да се използва за обучение и проверка на знания. Най-общо, условията, на които трябва да отговаря тази система са следните:

- **Лесна за използване**

С помощта на лесен и атрактивен графичен потребителски интерфейс авторите могат да опростят създаването на текстове, въвеждането на снимки и мултимедиини файлове – използвайки стандартен текстов редактор за създаването и редактирането и зареждането на необходимите текстови документи. Редакторът трябва да има съответните права и да попълни форми с необходимата информация. При добавяне или изтриване на нова информация, системата автоматично обновява и главното меню.

- **Централизирано съхранение на информацията**

Съдържанието да съхранява на сървъра в база данни. Централно съхранение означава, че съдържанието може да бъде използвано на много места в уеб сайта и форматирано за всяко устройство по различен начин – за уеб браузър, мобилен браузър, SMS, MMS, принтер и др.

- **Различни права за достъп**

Външните посетители могат единствено да се запознават с представянето на материалите. За да ги четат, те трябва да се регистрират и автентифицират като учащи. Учащите не са оторизирани да променят съдържанието, а само администраторът-инструктор има пълен контрол върху работата на системата. Това гарантира сигурна и ефективна работа на приложението.

- **Динамично съдържание**

Всички страници са изцяло динамични и могат лесно да се обновяват в реално време от оторизираните за това потребители.

- **Поддръжка през Интернет**

Възможност за редактиране навсякъде, по всяко време. Необходим е единствено достъп до Интернет и уеб браузър.

- **Дизайн без наличие на технически умения от автора**

Хора със средни познания по текстова обработка могат да създадат и публикуват текст в реално време. Никакви познания по HTML, WML и други компютърни езици за представяне на съдържание не са нужни. Съдържанието може директно да се прехвърли от текстообработваща програма, като системата автоматично го преобразува във вид на Интернет страница.

- **Интуитивен интерфейс за редактиране**

Оторизираните за редакция потребители имат могат по различен начин да достигнат до раздела за управление на съдържанието – включително, могат да променят съдържанието, което текущо са отворили само с натискането на един бутон.

- **Устойчивостта на дизайна е запазена**

Съдържанието от всеки автор се представя със същия дизайн. Избягват се субективни виждания на авторите и се запазва цялостния облик на сайта.

- **Навигацията се генерира автоматично**

Менютата се генерират автоматично въз основа на съдържанието на базата данни. Така се премахва опасността от оставяне на линковете да сочат несъществуващи страници.

- **Допълнителни източници**

Могат да се включат материали за четене, и линкове към външни източници в библиотеки и в Интернет. Възможност за използване на външни файлове – текстови материали, мултимедия, програми, които да се приложат към материалите и да са достъпни за ползващите системата.

- **Онлайн тестове**

Страницата предлага решаването на тестове за самоконтрол и формиране на оценка директно през Интернет. Автоматично генериране на статистически резултати за тестовете направени от всеки потребител.

- **Разнообразни начини за достъп до информацията**

Информацията може да се достъпва от компютър свързан с Интернет, през мобилен браузър, както и чрез заявка с SMS.

3.2. Структура на базата данни

Базата от данни е съвкупност от взаимосвързани, съхранявани заедно данни, с минимален излишък, които могат да бъдат използвани за едно или много приложения [21]. Тя трябва да бъде проектирана така, че да обслужва както тях, така и други бъдещи нужди. Базата от данни е основата на една цялостна информационна система. Проектирането ú е извършено в три основни етапа – концептуален дизайн, логически дизайн и физически дизайн .

3.2.1. Концептуален дизайн

Този етап предполага моделирането на базата данни като се определят видовете обекти и взаимоотношенията между тях, каквито са в реалния свят, без да се свързват с определена семантика на базата данни. Идеята е да се получи модел, приложим във всякакъв вид система за управление на база данни.

Основната роля в системата е отредена на потребителите. Потребителите са поне два вида – ученици и администратори (подържащи съдържанието на сайта). Нерегистрираните потребители имат права само за четене на представянето на темите. Услугите на системата са достъпни само за регистрирани потребители и това налага съхранение на информацията за тях в базата данни. От гледна точка на сигурността е задължително съхранението на потребителско име и парола в криптиран вид, с които потребителят извършва автентификация пред системата.

За учениците се съхранява и допълнителна информация като име и електронен адрес. Администраторите предоставят същите данни на системата, но трябва да се записва и тяхната роля в системата. Според критериите за привеждане на базата в първа нормална форма, налага се разглеждането на различните “действащи лица” като отделен обект.

Според изискванията на концептуалния модел информацията, която ще бъде обработвана в системата е обособена като обекти, които нормално съществуват в реалния свят (entities). Следва описание на споменатите обекти, операциите, в които те участват и атрибутите им. (Таблица 3.1.)

Обект	Операции	Атрибути
Потребител	✓ Регистрация ✓ Промяна на лични данни ✓ Попълване на тест ✓ Преглед на резултати ✓ Редактиране на съдържание	✓ Интентификатор ✓ Име ✓ Електронен адрес ✓ Мобилен телефон ✓ Статут
Категория		✓ Номер ✓ Заглавие ✓ Въведение
Тема		✓ Номер ✓ Категории, в които е включена ✓ Заглавие ✓ Съдържание
Файл		✓ Номер ✓ Теми, в които е включен ✓ Описание на файла
Изображение		✓ Номер ✓ Теми, в които е включено ✓ Описание
Връзка (link)		✓ Номер ✓ Тема ✓ URL ✓ Описание
Въпрос		✓ Номер ✓ Тема, към която се отнася ✓ Текст на въпроса
Отговор		✓ Номер ✓ Номер на въпроса, за който е предназначен ✓ Текст на отговора ✓ Верен ли е
Резултат		✓ Номер ✓ Категория, към която се отнася теста ✓ Потребител, на който принадлежи ✓ Дата ✓ Брой въпроси в теста ✓ Брой верни ✓ Репорт

Таблица 3.1. Обекти и атрибути.

Концептуалният дизайн изисква и определяне на взаимоотношенията на описаните обекти. Релациите могат да са в следните три категории:

- Едно към едно – един елемент от даден обект е свързан с точно един елемент от друг обект.
- Едно към много – един елемент от обект е свързан с няколко елеманта от друг обект. Вторият обект трябва да притежава външен ключ за връзка с първия.
- Много към много – много елементи от обект са в отношение с много елементи от друг такъв.

Взаимотношенията, между наличните обекти са описани в Таблица 3.2

Стартов обект	Целеви обект	Свързващ атрибут	Тип релация
Потребител	Резултати	ID на потребител	1-N
Категория	Резултати	Номер на тема	1-N
Категория	Тема	Номер на категория, Номер на тема	N-N
Тема	Файлове	Номер на тема, Номер на файл	N-N
Тема	Изображения	Номер тема, Номер на файл	N-N
Тема	Връзки	Номер на тема	1-N
Тема	Въпроси	Номер на тема	1-N
Въпрос	Отговори	Номер въпрос	1-N

Таблица 3.2. Отношения между обекти.

В настоящия проект връзките са от тип едно към много и много към много.

3.2.2. Логически дизайн

При преминаването към логическия модел често се налагат промени в концептуалния модел с цел подобряване производителността на операциите при работа с базата данни. Отчитат се няколко параметъра, свързани с оптимизацията на модела. Стойност на операция – отчита брой обръщания към обект или използвани релации до изпълнение на операцията. Изисквания за съхранение – броя байтове, нужни за съхранение на данните. Необходимо е да се направят възможните опростявания на модела. Това предполага отстраняване на излишъците на информация, обединяване, сливане на обекти и релации с цел повишаване ефективността на операциите. Всички тези действия водят до създаването на

същинския логически модел. Обектите се превръщат в таблици, а самите атрибути в колони на таблиците, всеки уникален идентификатор на обект се превръща в първичен ключ. Логическия модел описва още типа на данните, фиксирани ключове, възможност за индексиране и нулиране на полета, стойности по подразбиране.

Изграждането на логическия модел на базата данни наложи създаването на дванадесет таблици. При именованието на таблиците и техните атрибути за удобство се създават някои условности, които да улесняват създаването на заявки към базата от данни:

– Имената на таблиците са винаги с множествено число, защото индикират група от обекти.

– Името на първичния ключ се състои от единственото число от името на таблицата с допълнение "_ID" – напр. таблица "`users`" – първичен ключ "`user_ID`"

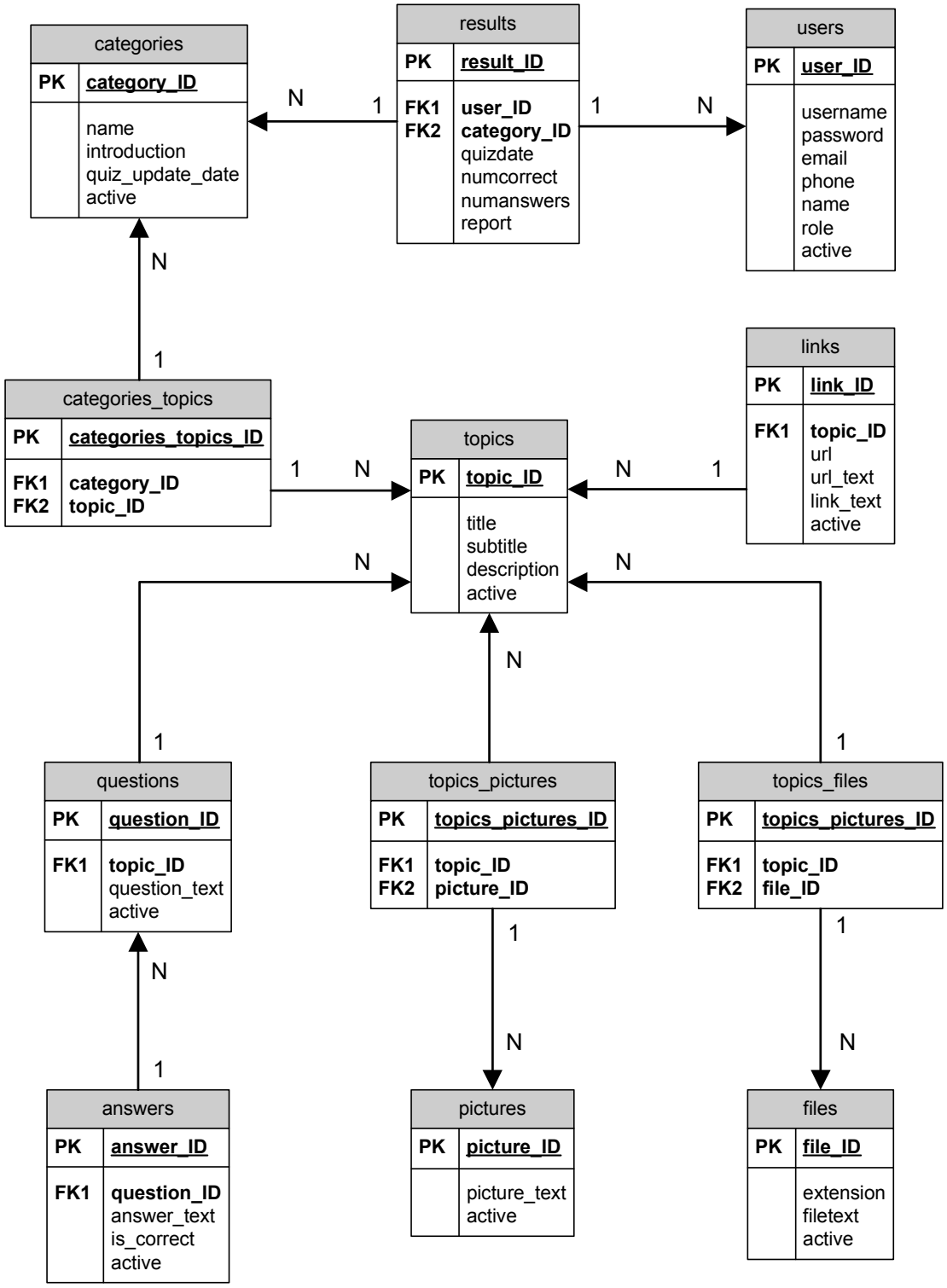
– Името на поле, уникално за таблицата се състои от единственото число от името на таблицата с допълнение някаква спецификация – напр. таблица "files" - Поле "`file_text`".

– Имената на полетата, които са първични ключове съвпадат с имената на първичния ключ от таблицата, с която правят връзка. Например поле `topic_ID` е първичен ключ в таблица topics и поле със същото име е вторичен ключ от таблица questions. Това е така, тъй като двете таблици са свързани едно към много и за конкретния ред от таблица topics отговарят един или повече реда от таблица questions със същата стойност на `topic_ID`.

– Имената на свързващите таблици са съставени от имената на двете таблици, които свързват. Например свързващата таблици между таблици `categories` и `topics` е "`categories_topics`".

На фиг. 3.1 е показана структурата на базата данни.

Връзките в таблиците са от типа "едно към много" и „много към много". За реализацията на връзки от типа много към много са използвани свързващи таблици. Такива са таблиците `categories_topics`, `topics_pictures`, `topics_files`. За реализацията на връзки от типа едно към много са използвани вторични ключове за връзка с първичния ключ от друга таблица. На фиг. 3.1. полетата, които са първични ключове са означени с PK, а вторичните – с FK.



Фиг. 3.1. Схема на базата данни..

Следва описание на създадените таблици и техните полета.

- **Таблица “users”:**

Всеки ред от тази таблица съответства на регистрацията на един потребител. Полето „user_id е primary key. Таблицата се състои от следните полета:

user_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се;

Уникален идентификатор на потребителя, при регистрация му се генерира автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

username – тип VARCHAR с дължина 32, ненулево, без стойност по подразбиране.

Потребителско име, с което се е регистрирал потребителя.

password – тип VARCHAR с дължина 32; ненулево, без стойност по подразбиране.

Тук се записва криптираната парола на потребителя.

email – тип VARCHAR с дължина 100; ненулево, без стойност по подразбиране.

Полето служи за записване адреса на електронната поща на потребителя. Това поле е задължително при регистрация на потребител, прави проверка за попълването му с валидна стойност за електронен адрес.

phone – тип VARCHAR с дължина 20; ненулево, без стойност по подразбиране.

В това поле се записва телефонния номер на потребителя и се използва за изпращане на текстови съобщения при заявка или при настъпване на определено събитие.

names – тип VARCHAR с дължина 100, ненулево, без стойност по подразбиране.

В това поле се записва реалното име на потребителя, то не е задължително при регистрация.

role – тип INTEGER с дължина 3, има стойност по подразбиране 0.

Определя статута на потребителя - 0 за учащ и 1 за поддържащите наличните материали (администратори) .

active – тип TINYINT с дължина 1 , има стойност по подразбиране 1.

При стойност 0 потребителят е деактивиран от администратора и не може да използва системата.

- **Таблица “categories”:**

Тази таблица съдържа, наличните в момента теми от материалите в избраната предметна област. Индексът е category_id, primary key за таблицата.

Полетата са следните:

category_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на категорията, при създаването ú се назначава автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

name – тип VARCHAR с дължина 50, ненулево, без стойност по подразбиране;

Сърханява заглавието на категорията, което се появява в менюто на старницата.

introduction – от тип TEXT с нулева стойност по подразбиране.

Съдържа кратко представяне на категорията.

quiz_update_date – тип DATETIME, с нулева стойност по подразбиране.

Там се записва точната дата и час на последното обновяване на въпрос от текущия тест към темата, чрез вградената в MySQL функция NOW().

active – тип TINYINT (1) , има стойност по подразбиране 1.

При стойност 0, темата е неактивна и не се вижда от потребителите.

- **Таблица “topics”:**

Таблицата съдържа въведените теми в системата. В една категория има няколко теми, като дадена тема може да се съдържа в няколко категории. Това определя необходимостта релацията между таблици „categories” и “topics” да е от типа „много към много”. Таблицата съдържа следните полета:

topic_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се;

Уникален идентификатор на темата, при създаването ú се назначава автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

title – тип VARCHAR с големина 100, без стойност по подразбиране.

Това е името на темата (topic), което се появява в автоматично генериран списък в началото на уводната статия на темата (introduction) и в меню, от което може да се достъпи съдържанието на темата.

subtitle – тип VARCHAR с големина 100, без стойност по подразбиране;

Подзаглавие, което ако е въведено се появява при извеждането на темата.

description – от тип TEXT с нулева стойност по подразбиране.

Съдържа текста на темата, въведен от администратора.

active – тип TINYINT(1) , има стойност по подразбиране 1.

При стойност 0, подтемата е неактивна и не се вижда от потребителите.

- **Свързваща таблица “categories_topics”**

Чрез тази таблица се осъществява връзката „много към много” между таблици „categories” и “topics”. Съдържа собствен първичен ключ и идентификаторите на категориите и темите като вторични ключове.

categories_topics_ID– тип INTEGER с дължина 5, ненулево, самоувеличаващо се;

Първичен ключ в таблицата.

category_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на категорията. Полето се използва за вторичен ключ в таблицата.

topic_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на тема. Вторичен ключ в таблицата.

- **Таблица „links”**

В нея се съхранява информация за всяка добавена към подтемата препратка, всяка заема един запис от таблица и има собствен уникален номер.

Полето „link_id” е primary key.

link_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на препратка към друга web старница, при прикачването се назначава автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

topic_ID – тип INTEGER с дължина 5, ненулево, стойност по подразбиране 0.

Това е foreign key и отговаря на topic_ID от таблицата topics. Идентифицира подтемата, на която принадлежи.

url - тип VARCHAR с големина 100, без стойност по подразбиране.

Тук се пази уникален локатора на ресурса (Unified Resource Locator). Стойността се използва от уеб браузъра за зареждане на външна страница.

url_text – тип VARCHAR с големина 100, без стойност по подразбиране.

Това е текста, който потребителя вижда, изобразява се с различен цвят, може да е както реалното url, така и вякакъв друг текст, въведен от администратора.

link_text – от тип TEXT, няма стойност по подразбиране.

Появява се под съответният линк, като негово описание.

active – тип TINYINT(1) , има стойност по подразбиране 1.

При стойност 0, връзката е неактивно и не се вижда от потребителите.

- **Таблица „files”**

В нея се съхранява информация за всеки прикачен в системата файл, като всеки един заема точно един запис и има собствен уникален номер. На сървъра те се записват със собственото си разширение и ункално име свързано с техния идентификатор.

file_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на файл, при прикачването се генерира автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

extension – тип VARCHAR с дължина 16, без стойност по подразбиране.

Съхранява името на разширението (или разширенията) на файла.

file_text – от тип TEXT, няма стойност по подразбиране.

Текст, описващ файла. Появява се в категориите, към които принадлежи.

active – тип TINYINT(1) , има стойност по подразбиране 1.

При стойност 0, файлът е неактивен и не се вижда от потребителите.

- **Свързваща таблица „topics_files”**

Чрез тази таблица се осъществява връзката „много към много” между таблици „topics” и “files”. Съдържа собствен първичен ключ и идентификаторите на темите и файловете като вторични ключове.

topics_files_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Първичен ключ в таблицата.

topic_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на тема. Вторичен ключ в таблицата за връзка с таблица topics.

file_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на файл. Вторичен ключ в таблицата за връзка с таблица files.

- **Таблица “pictures”**

В нея се съхранява информация за всяко прикачено изображение в системата, като всяко едно заема точно един запис и има собствен уникален номер.

picture_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на изображение, при прикачването се назначава автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

picture_text – от тип TEXT, няма стойност по подразбиране.

Ако е въведен, този текст се появява под съответното изображение.

active – тип TINYINT(1) , има стойност по подразбиране 1.

При стойност 0, изображението е неактивно и не се вижда от потребителите.

- **Свързваща таблица „topics_pictures”**

Чрез тази таблица се осъществява връзката „много към много” между таблици „topics” и “pictures”. Съдържа собствен първичен ключ и идентификаторите на темите и изображенията като вторични ключове.

topics_pictures_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Първичен ключ в таблицата.

topic_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на тема. Вторичен ключ в таблицата за връзка с таблица topics.

picture_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се;

Уникален идентификатор на изображение. Вторичен ключ в таблицата за връзка с таблица pictures.

- **Таблица „questions”**

В нея се съхраняват въпросите за тестовете към всяка тема, към която има въведени такива. Всекиа заема един запис от таблица и има собствен уникален номер.

question_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на въпрос, при създаването му се назначава автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

topic_ID – тип INTEGER с дължина 5, ненулево, стойност по подразбиране 0.

Това е foreign key и отговаря на topic_id от таблицата topics. Идентифицира темата, на която принадлежи съответния въпрос. При съставянето на теста администраторът избира първоначално категория и после при всеки въпрос от

падащо меню избира някоя от темите на съответната категория. В доклада за резултата от теста, полето се използва за генериране на линк към съответната тема.

question_text – тип TEXT, няма стойност по подразбиране.

Тук се съхранява текста на въпроса, при създаването му.

active – тип TINYINT(1) , има стойност по подразбиране 1.

При стойност 0, въпросът е неактивен и не се вижда от потребителите.

- **Таблица „answers”**

В нея се съхраняват отговорите на всеки въпрос. Всеки заема един запис от таблица и има собствен уникален номер. Полето „answer_id” е primary key. Връзката с таблица „questions” е от тип “едно към много”, тъй като към всеки въпрос има предложени от 2 до 8 възможни отговора. Таблицата съдържа следните полета:

answer_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на въпрос, при създаването му се назначава автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

question_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Това е foreign key и отговаря на question_ID от таблицата questions. Идентифицира въпроса, на който принадлежи този отговор. За всеки въпрос от таблица “questions” съответстват няколко отговора.

answer_text – тип TEXT, няма стойност по подразбиране.

Тук се съхранява текста на предложения отговор, който е въведен от администратора. При създаването на въпрос администраторът избира колко варианта на отговор да се предложат, интерфейсът предлага избор между два и осем отговора. Всеки е отделен запис в тази таблица.

is_correct – тип TINYINT , няма стойност по подразбиране.

Със създаването на въпрос и избора на броя въпроси се генерират толкова записа в тази таблица, колкото са възможните отговори. Администраторът избира от падащ списък на интерфейса за създаване на тестове кой пореден отговор е верният. Ако даден отговор е верен стойността на полето is_correct е 1, за всички останали, стойността е 0. Полето се използва за проверка на тестовете.

active – тип TINYINT(1) , има стойност по подразбиране 1.

При стойност 0, отговорът е неактивен и не се вижда от потребителите.

❖ **Таблица “results”**

Тази таблица съхранява история за резултатите от проведени тестове на регистрираните потребители. Един тест се прави от много различни потребители, ето защо таблицата прави връзка от типа „едно към много” с таблица users. От друга страна има тест за всяка категория, ето защо таблицата прави връзка „едно към много” и с таблица categories.

Таблица results съдържа следните полета:

result_ID – тип INTEGER с дължина 5, ненулево, самоувеличаващо се.

Уникален идентификатор на резултат от един проведен тест, при завършването на тест, стойността му се генерира автоматично, според текущата до момента стойност. Полето се използва за първичен ключ в таблицата.

category_ID - тип INTEGER с дължина 5, стойност по подразбиране 0.

Това е foreign key и отговаря на category_ID от таблицата categories. Идентифицира темата, на която принадлежи теста.

user_ID – тип INTEGER с дължина 5.

Това е foreign key и отговаря на user_ID от таблицата users. Идентифицира потребителя, на който принадлежи резултата.

quiz_date – тип DATETIME, с нулева стойност по подразбиране.

В това поле се записва точната дата и час на провеждане на теста към темата, чрез вградената в MySQL функция `NOW()`.

num_correct – тип INTEGER с дължина 3, няма стойност по подразбиране.

В това поле се записват броя на верни отговори за даден тест при изчисляването на реззултата.

num_answers – тип INTEGER с дължина 3, няма стойност по подразбиране.

Съдържа общия брой на въпросите, включени в един тест. Използва се за изчисляване на процента верни отговори.

report – тип TEXT, няма стойност по подразбиране.

В това поле се сърханява подробен доклад за резултатите от теста. Показва всички сгрешени въпроси с отговора, посочен от потребителя и с верния отговор. Съдържа и връзка към темите, към които принадлежат сгрешените въпроси.

3.2.3. Физически модел

Физическият модел предполага разглеждане на системата в контекста на MySQL. При разваботване на текущото приложение е използвана система за управление на база данни MySQL версия 5.0. По подразбиране поддръжка на MyISAM таблици е вградена в продукта след версия 2.23, така че всички таблици на конкретната системата са създадени като MyISAM таблици.

В MySQL цялата информация, включително мета-информацията се съхраняват във вид на файлове върху трвърдия диск. Под мета-информация разбираме служебна информация или информация за самите данни. За всяка една MyISAM таблица от базата се пази отделен файл, който има три инстанции, съответно с разширение “.frm”,”.myd”,”.myi”. Файловете с разширение “.frm” съхраняват структурата на таблиците т.е. имена на колони, техните типове и др. Файловете с разширение “.myi” съхраняват информация за ключове и индексите на таблицата, стойности по подразбиране, опции за нулиране и т. н., а тези с “.myd” същинстката информация в таблицата.

3.2.4. Нормализация на системата

Процесът на нормализиране на базата данни преминава през три последователни стъпки познати като нормални форми. Привеждане на базата данни в първа нормална форма се свързва с елиминиране на повтарящи се групи информация чрез разделянето им в отделни таблици. За всеки обект е създадена отделна таблица, което не допуска съхраняването на една и съща информация на няколко места. Тук се появява и първия проблем по време на логическия дизайн при описанието на вече създадените обекти. Обектът Файлове налага съхранение на информация за достъпни на сървъра прикачени файлове. Съхранението на връзките към други страници (линкове) изисква освен описание на съдържанието им и съхранение на URL. За да не се получи таблица съдържаща множество нулеви полета, бяха създадени отелни такива.

Дефинирани са всички първични ключове на таблици, с това системата е приведена в първа нормална форма.

Съгласно изискването за втора нормална форма, базата данни трябва да е вече в първа нормална форма и периодично повтарящи се атрибути на обекти да са изнесени в отделни таблици. Така е удовлетворено изискването всички атрибути да са зависими от първичния ключ и базата е приведена във втора нормална форма.

Третата нормална форма изисква базата да е вече във втора нормална форма и липсата на транзитивни зависимости, премахване на колоните, които не са пряко зависими от първичния ключ. За конкретната база данни това условие е налице, с което може да се счита че тя е нормализирана.

3.3. Описание на работата на приложението

3.3.1. Потребители и потребителски роли

Потребителските роли в системата се представят като актьори, а работата им със системата – като случаи на употреба. Така едно множество от физически потребители, имащи една и съща роля, се представя като един актьор. Това не изключва възможността един физически потребител да бъде регистриран два пъти в системата с две различни роли.

Според изискванията за конкретното приложение, то има три различни вида потребители:

- **Гост**

Най-ниско ниво в йерархията на системата. Получава само обща информация за системата, как тя се използва и какво съдържа. Има достъп до описанието на материалите, но не може да разглежда самото съдържание, нито да прави тестове. Влиза в системата без авторизация. Действията, които може да извършва са:

- регистрация;
- автентификация;
- преглед на интро на категориите;
- преглед на обща информация за системата.

- **Учащ**

Получава достъп до всички материали на сайта и възможност да извършва самопроверка с тестове върху отделните глави.

Изисква се регистрация и атентификация при влизане. Регистрацията може да е свободна или да се прави от администратор, в зависимост от приложението на проекта. Действията, които може да извършва учащ в системата са:

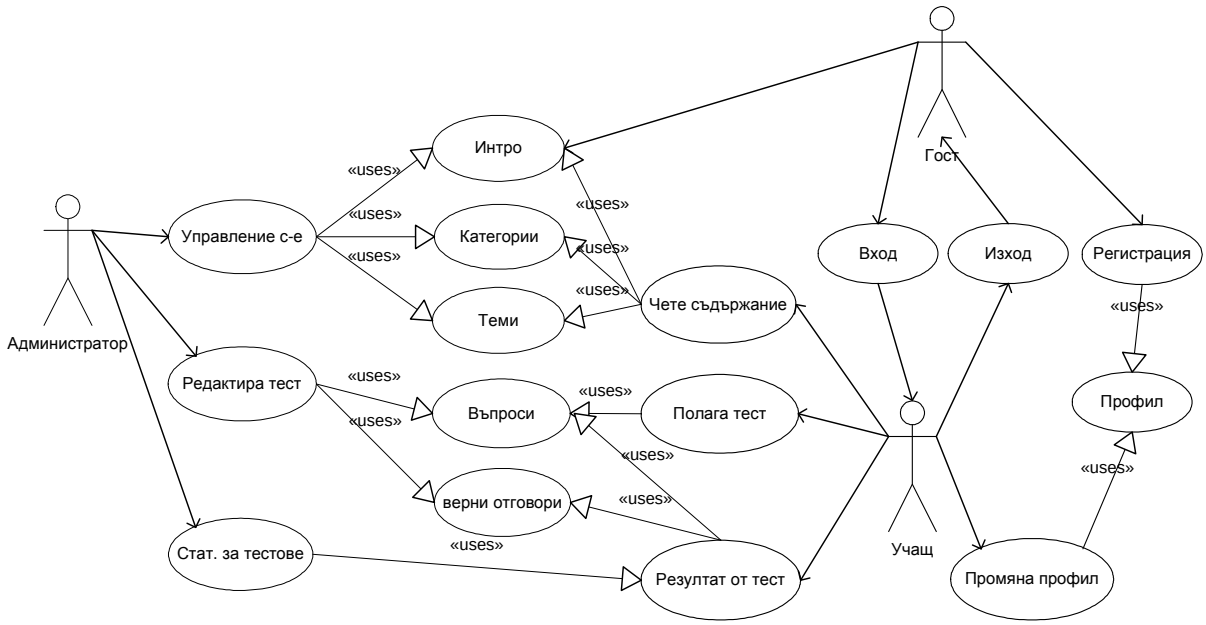
- преглед на интро на категориите;
- четене на теми през Интернет и мобилен телефон;
- заявка за дефиниция чрез SMS

– решаване на тест за категория;
– резултати и подробен доклад на тест;
– статистически резултати от всички тестове;
– промяна на данни от профила си;
– изход от системата.

➢ **Администратор**

Най-високо ниво в йерархията на системата. Създава и обновява информацията в системата. Акаунтът му се създава от системния администратор в базата данни, като се смени статуса на даден регистриран потребител. Влиза в системата като другите регистрирани потребители. Администраторът притежава всички привилегии на „учащ”, като допълнително има и следните други права:

– добавяне/изтриване на категория;
– редакция на интро на категория;
– добавяне/изтриване на тема;
– добавяне/изтриване/редакция на текст, файлове, картинки, външни връзки към тема;
– съставяне на тест за дадена категория;
– преглед на резултатите от тестовете на всички потребители.



Фиг. 3.2. Актьри и техните взаимодействия със системата.

На фиг. 3.2. е представено какви са общите за всички тях взаимодействия със системата (сценарии). Ще отбележим, че са показани само уникалните действия за потребителите. Когато дейтвие може да се извърши от повече от един актьор, то е показано само при актьора, който е на по-ниско ниво в йерархията на системата.

Възможно е с действията си актьорите взаимно да си влияят. Така например, ако едновременно учащ полага тест, а администратор актуализира съдържанието на теста, резултатите от теста няма да са коректни. При програмната реализация са взети мерки за предотвратяването на нежелани резултати при взаимоотношенията между актьорите.

3.3.2. Организация на съдържанието

Информацията в системата е структурирана в определен брой категирии (categories), всяка от които съдържа по няколко теми (topics).

Една категория (category) може да се сравни с въпрос от конспект по дадена учебна дисциплина. Съдържа множество от дефиниции на понятия, описания, извеждания и т. н. За всяка категория има кратко описание и съдържание на темите, които включва.

Тема (topic), в симсъла на системата, представлява един от многото проблеми, разглеждани в дадена категория. В една тема може да се дефинира дадено понятие, да се обясни процес, изведе формула и т. н. Дадена тема може да се съдържа в повече от една категория. Освен текст, в една тема се съдържат множество допълнителни ресурси – картинки, външни препратки (links) и файлове. Всеки ресурс може да се прилага в неограничен брой теми.

Администраторът има достъп до интерфейс за добавяне, редактиране и триене на текстовете и ресурсите. Той решава кои ресурси да участват в дадена тема, както и кои теми да включва дадена категория.

3.3.3. Тестове

Системата предвижда възможност за съставяне и решаване на тестове за самоконтрол на учащите.

Всеки тест съдържа определен брой от въпроси с два или повече отговора. При съставянето на теста администраторът определя за всеки въпрос към коя тема се отнася. Студентът може да избере да решава тест за определена категория и тогава

се селектират само тестовите въпроси от тази категория. Времето за решаването на теста се определя от броя на въпросите.

При изчисляване на резултатите се сравняват посочените от студента отговори с отговорите от базата данни, които предварително са дефинирани като верни от администратора. Като резултат от теста, потребителят получава процента от верните отговори, както и доклад за грешките, които е допуснал с връзки към съдържанието на съответни теми. Резултатите могат да се видят през Интернет или да се изпращат с e-mail или SMS. В своята страница, всеки потребител разполага със статистика за резултатите от тестовете и датите, в които ги е положил. Администраторът разполага с информация за резултатите на всички потребители върху даден тест.

3.3.4. Редактиране на съдържанието

Съдържанието на системата е изцяло динамично и може да се редактира от администратора. При първото стартиране в системата няма никакви данни. Администраторът разполага с интерфейс за добавяне на текстове, картинки, файлове, външни препратки (links) и тестови въпроси, през който вкарва материалите за избраната от него предметна област. Самият текст може да се копира директно от текстообработваща програма (напр. Word или Excel), като системата автоматично преобразува съдържанието и го вкарва в базата данни. Също автоматично, според избраните от администратора опции, се изгражда менюто за мобилната и Интернет версията на сайта.

В процеса на работа, освен в описания интерфейс, администраторът може да променя съдържание и от произволна заредена страница с натискане на бутон „Редактиране”. В този случай, той вижда какво е текущото съдържание на теамата и как ще изглежда то след редакцията.

3.3.5. Версия на сайта за мобилни телефони

Системата притежава олекотена версия за достъп през браузър на мобилно устройство. Взети са предвид по-малките системни възможности, които преитежават тези устройства. През мобилната версия може да се види основна информация за системата, както и съдържанието на категориите и темите. От съдържанието на темите са извадени мултимедийните файлове, които са тежки за зареждане и не могат да се представят от повечето мобилни устройства. Освен стандартни линкове,

са предвидени и т. нар. access keys (клавиши за достъп), с които навигирането из сайта е по-лесно. Текстът в големите теми автоматично се разделя на няколко страници, така че да е по-лесен за зареждане и по-удобен за четене през мобилния дисплей. Използва се изцяло unicode кодиране за кирилицата, за да има пълна съвместимост при визуализиране на страницата от всички мобилни браузъри.

За регистрираните в системата потребители, мобилният интерфейс дава възможност за вход с потребителското име и парола и поддържа сесия. След автентифициране в системата, потребителите имат достъп до резултатите и съкратен доклад за всеки направен от тях тест.

3.3.6. Изпращане на съобщения

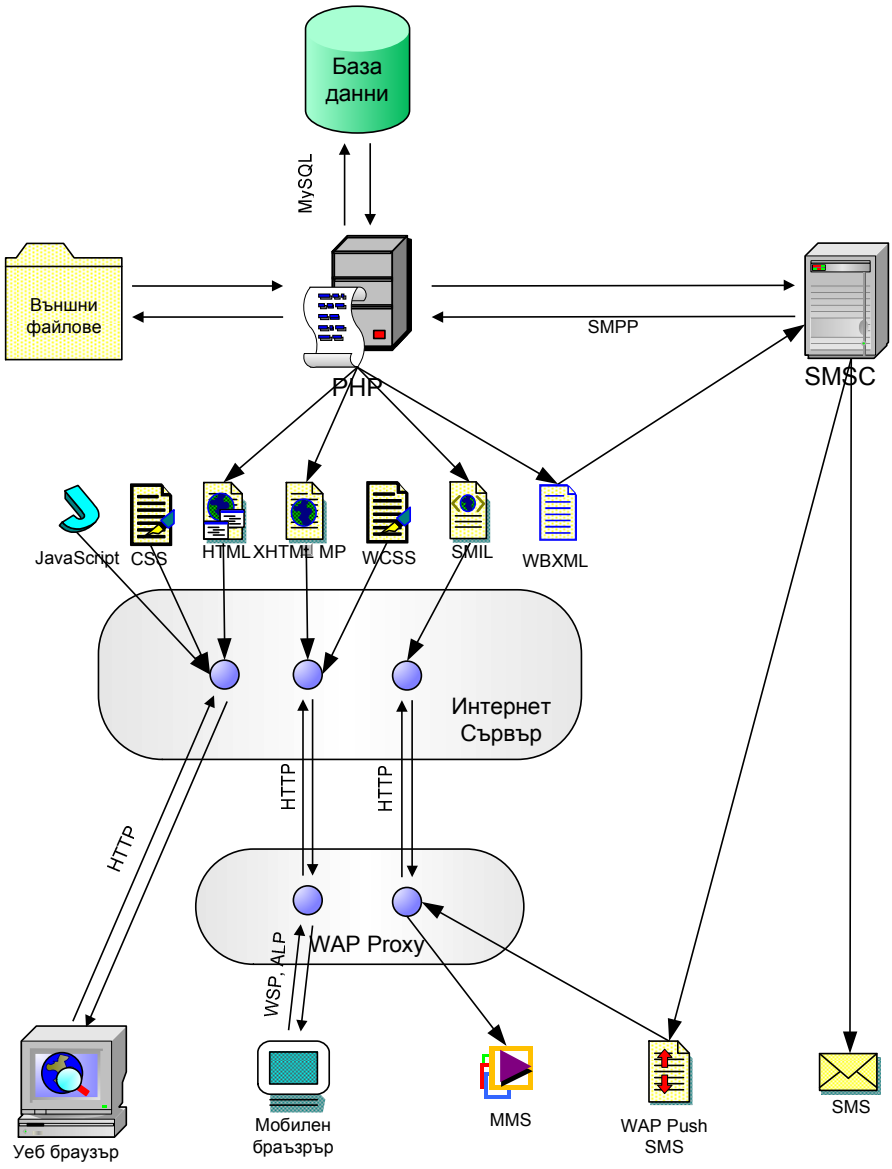
Настоящият проект осъществява конекция към сървър за изпращане на съобщения и реализира изпращане на кратки текстови съобщения (SMS), WAP Push SMS и мултимедийни съобщения (MMS).

При регистрация в системата всеки потребител задава своя мобилен номер. След това, той може да изпраща различни заявки към системата от своя мобилен телефон чрез SMS. При получаване на входящ SMS, приложението проверява дали изпращащият номер е регистриран в системата и дали текстът от съобщението е валидна заявка. Ако са налице условията, предава нужната информация. По този начин потребителят може бързо да се информира за резултати от тестове, да получи определена дефиниция или да бъде информиран при съставяне на нова категория или излизане на нов тест.

Глава 4. Програмна реализация

4.1. Обща схема на работа на приложението

На фиг. 4.1. е представена общата схема на работа на приложението.



Фиг. 4.1. Схема на работа на приложението.

При своята работа, PHP приложението се свързва със сървър за база данни, Интернет сървър и сървър за изпращане на съобщения (Short Message Service Center) и осъществява няколко различни интерфейса за достъп и извеждане на съдържанието.

При интерфейса за уеб браузър, потребителят зарежда интернет страницата през своя браузър като изпраща HTTP заявка до сървъра. Заедно с адреса на изискваната страница, той може да предаде входни данни на програмата чрез GET или POST заявки. PHP програмата се изпълнява на сървъра като извиква помощни функции от други файлове. По време на изпълнението си се свързва с базата данни с помощта на вградените за целта функции. Отговорът от MySQL сървъра се съдържа във върнатия резултат на функциите. Изходният резултат от изпълнението на PHP програмата е HTML код. Браузърът на клиента сваля този код, заедно с изискваните от него JavaScript и CSS файлове, и визуализира резултата.

При работа с мобилен браузър, клиентът комуникира с прокси сървър (WAP Gateway или i-mode сървър), чрез протоколи WSP (Wireless Socket Protocol) или ALP (Application Layer Protocol), които работят подобно на HTTP протокола. Прокси сървърът предава заявките до Интернет сървъра и препраща върнатото съдържание, което е в XHTML MP и WCSS формат.

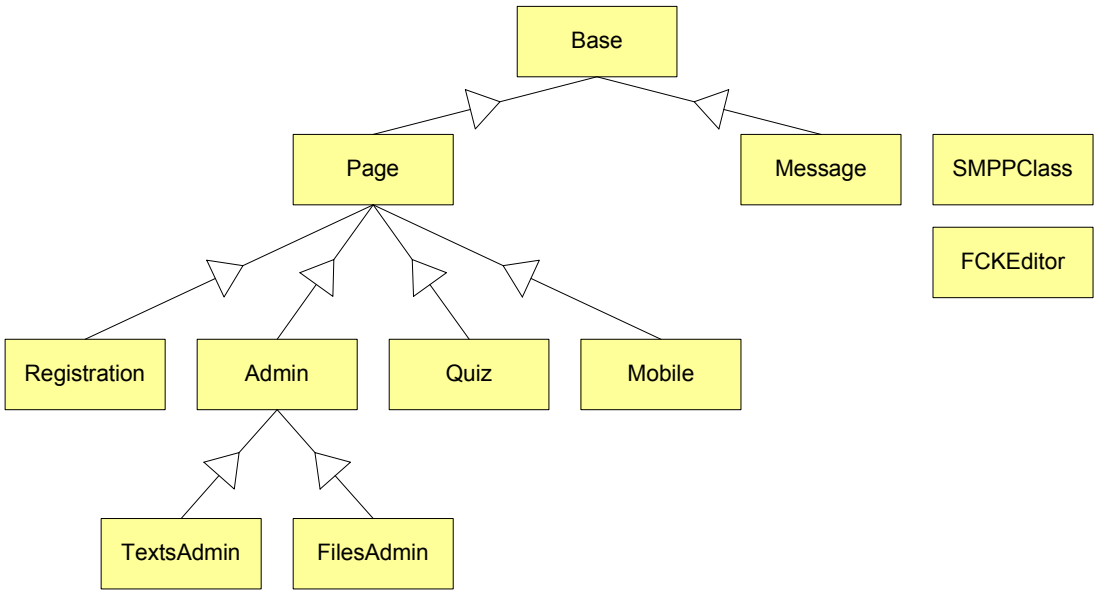
За изпращане на текстово съобщение (SMS), програмата комуникира с SMSC чрез протокола SMPP. SMS центърът се грижи за предаването на съобщението през мобилната мрежа.

За изпращане на мултимедийно съобщение (MMS) е необходимо по заявка на сървъра, клиентът да изтегли и визуализира съобщението. За целта, първоначално програмата генерира SMIL (Synchronizes Multimedia Integration Language) код и го съхранява на сървъра. След това генерира SMS с текст, кодиран във формат WBXML (Wireless Binary eXtended Markup Language), което се интерпретира от мобилния телефон като WAP Push SMS. В WBXML файла е записан URL адреса на SMIL файла от сървъра. Мобилният телефон се свързва със сървъра (отново посредством WAP прокси), изтегля файла и го визуализира чрез своя MMS интерпретатор.

4.2. Описание на класовете

Схема на организация на класовете е представена на фиг. 4.2.

В клас **Base** са реализирани методите за осъществяване на вход и изход на данни. Това включва функциите за работа с MySQL, както и функции за обработка и филтриране на данните, преди да се покажат или съхранят.



Фиг. 4.2. Организация на класовете.

Клас **Page** наследява класа Base, като добавя методи за визуализиране на съдържанието на страниците. Във всяка страница се извиква инстанция на този клас и методи, които визуализират по един и същи начин статичните части в началото и в края на сайта.

Клас **Registration** е наследник на класа Page. Добавя методи необходими за извършване на регистрация, за смяна на лични данни и за възстановяване на изгубена парола.

Клас **Admin** е наследник на класа Page. В него са дефинирани методи за добавяне, изтриване и променяне на ресурсите на системата

Клас **TextsAdmin** е наследник на клас Admin като добавя методи, специализирани за обработка на текстовете в страницата

Клас **FilesAdmin** е наследник на клас Admin, като добавя методи, специализирани за обработка на външни файлове.

Клас **Quiz** наследява клас Page и добавя методи за визуализиране на тестове, както и за изчисляване на резултати и оформяне на статистики.

Клас **Mobile** наследява клас Page.като предефинира функциите за представяне на съдържанието, така че то да е подходящо за представяне на мобилен телефон. Добавя също функции за кодиране на съдържанието, така че правилно да се визуализират шрифтовете.

Клас **Message** наследява клас Base като добавя методи за изпращане и визуализиране на текстови и мултимедийни съобщения.

Клас **SMPPClass** е клас съдържа методи за осъществяване на конекция с SMSC чрез протокол SMPP. Реализиран е на базата на безплатно разпространяван с GNU лиценз клас разработен от ChimIT Communications (http://chimit.nl), с някои модификации, необходими за настоящото приложение.

Клас **FCKEditor** е външен клас, разработен от Frederico Caldeira Knabben и разпространяван свободно с GNU лиценз (http://www.fckeditor.net). Използва се за осъществяване на потребителски интерфейс за въвеждане на текстове и мултимедия.

4.3. Реализация на основните функции на приложението

4.3.1. Регистрация и автентификация на потребители

Регистрацията се осъществява в скрипра register.php, където се създава инстанция на класа Registration.

По важни методи от класа са следните:

```
Registration::validate_register_form($username,$password,$password2,$email,$phone
,$names)
```

Данните се валидират от страна на сървъра. За валидация се използват регулярни изрази. Тук се извършват следните проверки:

– проверява се чрез регулярен израз валидността на потребителското име:

```
if(!ereg('^[a-zA-Z0-9_\.\-]{3,16}$', $username))
 $error.= "Please enter username between 3 and 16 latin letters, digits,
'.', '_' or '-‘'.<br/>";
```

– проверява се уникалността му:

```
$username=addslashes($username);
  $result = mysql_query("SELECT * FROM users WHERE username='$username'");
  if (!$result) return  "Query problem!";
  if (mysql_num_rows($result)>0) $error="The username is taken!";
```

– проверка за формата и дължината на паролата;

– проверка за съвпадение при повторно въвеждане на паролата;

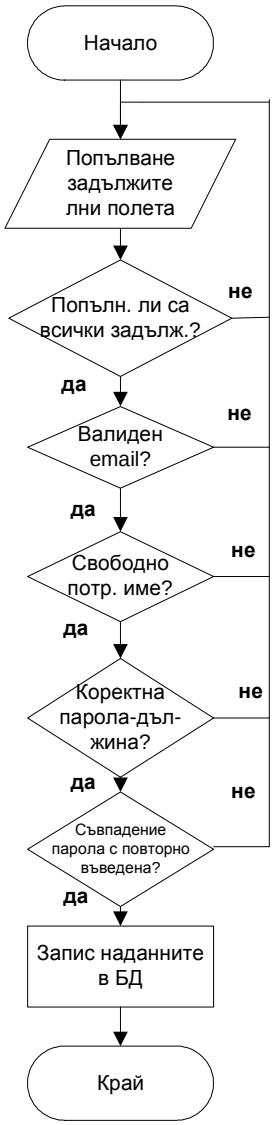
– проверка за валиден формат на електронния адрес – използва се допълнителна функция `Registration::valid_email($address)`, която проверява следния регулярен израз:

```
if (ereg('^[a-zA-Z0-9_\.\-]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$', $address))
      return true;
else  return false;
```

– проверка за валиден телефонен номер.

`Registration::register_user(...)`

В скрипта register.php се прави проверка за резултата на горната функция, ако всички проверки са минали успешно се прибавя нов запис в базата данни; Алгоритъмът е представен с блок схема (фиг. 4.3).



Фиг. 4.3. Алгоритъм за регистрация на потребител.

При опит на потребител да влезе в системата, формата се обработва от скрипта members.php. Тук се извиква функцията `Page::login()`, която прави заявка към базата данни и проверява съвпадение за потребителско име и парола. Инициализират се сесийните променливи, извеждат се подходщите съобщения за успешно влизане.

Извеждат се подходящи съобщения за потребителя чрез променливата $members_text, която се подава на функцията `Page::user_menu()`. Тук е важно да се спомене функцията `user_menu($members_text)`, в която се изобразява подходящия изглед, в зависимост от ролята на потребителя, като за администратора в менюто са включени опции за редактиране на съдържанието. При успешен вход в системата се записва сесийна променлива с ID на потребителя, която се използва за идентификация на потребителя в другите страници от сайта.

При забравяне на парола се изпълнява скрипта password_recovery.php. Неговата роля е да провери в базата данни съвпадението за потребителско име и електронен адрес. При съвпадение се генерира нова парола, тъй като в БД се съхраняват само хешове на пароли. Функцията `Registration::generate_new_password()` генерира парола, съобразена с изискваната дължина, съдържаща букви (главни и малки) и цифри. Обновява се БД и се изпраща мейл на потребителя.

За смяна на паролата от потребителя се грижи скрипта change_password.php. Проверява се: влидността на потребителското име; съвпадение при двете въвеждания на новата парола; формата на паролата. При успех се вика функцията `Registration::change_password($username, $old_password, $new_password)` –

тук се проверява и дали текущото потребителско име съвпада с въведената стара парола и при успех се обновява базата данни.

4.3.2. Реализация на вход и изход

Настоящата разработка предоставя възможност потребителите да добавят информация на сървъра. Това определя специални мерки, които се взимат, за да се предпази сървъра от въвеждането на потребителска информация, която би му навредила или би променила външния вид на сайта.

Функциите, които реализират входни и изходни операции с данни от сървъра са дефинирани основно в класа Base.

`Base::db_connect()` – осъществява конекция към базата данни. Създава се конекция с вградената PHP функция `mysql_connect()`.

Функцията `db_connect` прави опит да се свърже с име и парола към сървъра, като за целта акаунтът трябва да е предварително създаден в MySQL. Като резултат от изпълнението си връща идентификатор на връзката към базата данни при успех и `false` при неуспех.

`Base::userstring2DB()` – филтрира текст въведен от потребителя, без да позволява никакви HTML тагове. Използва вградените функции htmlspecialchars()и addslashes().

`Base::admintext2DB()` – обработва текст от потребител с по-големи права като му дава пълна свобода да използва HTML тагове и само поставя символи „\“ преди всеки специален за SQL символ, който може да доведе до неуспешно изпълнена заявка.

`Base::userstring2display()` - преобразува текста, съхраняван в базата данни до вид за представяне на потребителя. Използва функцията stripslashes() за да премахне всички поставени символи „\“ от функцията addslashes().

За съхранение на файлове се използва метода `FilesAdmin::upload_file()` – Методът записва на сървъра файл от избрана директория на компютъра на клиента (потребителя). Като резултат връща `true` при успех или грешката – при неуспешно записване. Трансферът на файл се извършва с HTTP протокол, чрез форма изпращяща POST заявка в кодиране тип `enctype="multipart/form-data"`.

От съображения за сигурност не се дава възможност на потребителя да избира името на файла и директорията за съхранение, а те се избират от програмата.

Оставено е само разширението на файловете, защото някои операционни системи (напр. MS Windows) го използват за определяне на типа на файла.

При качването на файл, той се запазва в стандартната временна директория на сървъра, след което се премества в предварително зададена на сървъра директория. Функцията реагира на всички изключителни ситуации (грешки), които могат да възникнат при качването на файла. В резултат от изпълнението и се връща стойност `true` или съобщение за грешка.

Скриптът resize_image.php **п**роменя размерите на дадено jpeg изображение. Приема GET параметри за името на файла с изображението и желаните максимална ширина и височина на показваното изображение. Изчислява пропорциите на оригиналното изображение, за да го преоразмери пропорционално.

По този начин на клиента се изпраща само преоразмереното изображение и се спестява трафик по мрежата. Скриптът помага и да се се подобри форматирането на текста на страницата като премахва опасността клиентите да зададат изображения, които да развалят външния вид на сайта.

4.3.3. Реализация на управление на съдържанието

Управлението на съдържанието се извършва от страниците: categories_manage.php, categories_edit.php, topics_edit.php, categories_delete.php, topics_delete.php. Всеки от тези скриптове е предназначен за промяна на съдържанието на сайта. Извършват се промени в базата данни, за които е привилегирован само администраторът на системата, менажиращ наличното съдържание. Въпреки, че от графичния интерфейс връзки се предоставят само за администратора, в началото на всеки един скрипт също се прави проверка за ролята на потребителя, попаднал на страницата за редакция.

От страницата categories_manage.php администраторът избира тема за редактиране или добавя нова. Наличната форма предава с POST параметри на скриптовете categories_edit.php и topics_edit.php избраните тема и/или подтема. Изобразява се текстбокс, в който администраторът въвежда съдържанието. В началото на скрипта се прави проверка с функцията `Page::check_admin()` дали потребителят има тези права. `Page::ategories_select($current_category_ID)` и `topics_select($category_ID,$current_topic_ID)` правят съответните заявки към базата данни и връщат избраната тема и наличните към нея в момента подтеми.

За предоставяне на повече възможности се използва външния клас FCKEditor. Обект на класа на FCKEditor се създава с:

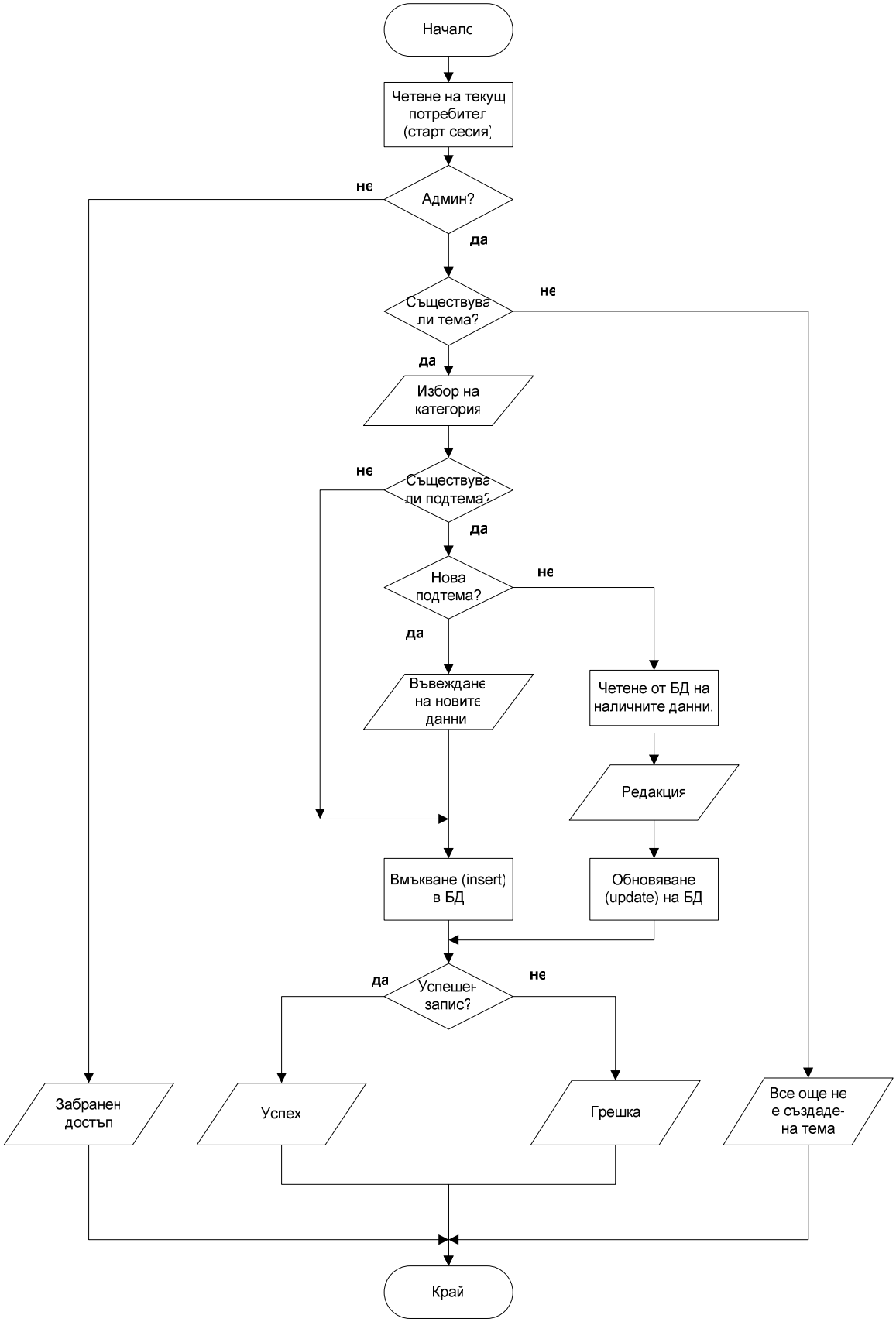
```
$oFCKeditor = new FCKeditor('introduction') ;
$oFCKeditor->BasePath = 'FCKeditor/';
$oFCKeditor->Value = "$introduction";
$oFCKeditor->Create();
```

Въведените от администратора данни се вкарват в базата данни (чрез `TextsAdmin::insert_category(...)` или `TextsAdmin::update_category(...)`. В зависимост от резултата се изпълнява скрипта success.php или error.php, като се показва на потребителя съответно съобщение за успех или грешка.

При избор за редакция на подтема се попада в страницата topics_edit.php. `TextsAdmin::update_topics_form($category_ID)` проверява дали въвеждаме нова тема или редактираме. В случай на редакция се прави заявка към базата данни и се изобразява съдържанието на полето description от таблицата topics. При създаване на нова тема се появява празна форма. Алгоритъмът е представен с блок схема на фиг. 4.4.

Изтриването на налична тема се осъществява от скрипта categories_delete.php. Тук отново стандартно се извиква проверката за наличните права. За изтриване на категория необходимо всички свързани с нея глави да бъдат изтрити. Скрипта за изтриване обработва резултатите върнати от страницата categories_manage.php, преди да бъде изпълнен се прави проверка дали към избраната категория за триене има въведени теми. При наличие на такива се съобщава на потребителя, че те трябва да се изтрият предварително. За изтриването на тема се изпълнява метода `TextsAdmin::delete_category()`, той обновява таблица categories от базата данни, като дава стойност active=0 на съответната категория. Накрая се изпълнява success.php при успех и error.php при грешка.

При изтриване на тема от дадена категория, автоматично се изтриват и всички свързани с нея картинки, линкове, файлове и въпроси от теста. Това се извършва от функцията `delete_topic()`. Функцията прави последователно заявки към таблиците questions, answers, pictures, files, links и накрая topics, като изтрива принадлежащите на темата и накрая самата тема.



Фиг. 4.4. Алгоритъм за създаване/редактиране на подтема.

4.3.4. Реализация на управление на тестове

Това е друг важен аспект от дейността на системата. Реализиран е в quiz_edit.php. Чрез избор на линка ‘Quiz management’, потребителят с необходимите права вижда форма за редакция на въпросите към тестовете.

Тук отоново се започва с проверка на ролята на потребителя, функцията е вече споменатата `Page::check_admin()`. С `Admin::edit_quiz_form()` се прави заявка към таблиците questions, answers и topics. Извличат се необходимите атрибути за показването на теста към избраната глава, като е маркиран и верния отговор:

```
SELECT questions.question_ID,questions.question_text,questions.has_pic,
  answers.answer_ID,answers.answer_text,answers.is_correct, topics.title
FROM    questions,answers,topics
WHERE   topics.category_ID='$quiz_category_ID'
AND topics.topic_ID=questions.topic_ID
AND questions.question_ID=answers.question_ID
AND questions.active='1' AND answers.active='1'
ORDER BY questions.question_ID;
```

След като е избран вече тест се предлагат следните възможности:

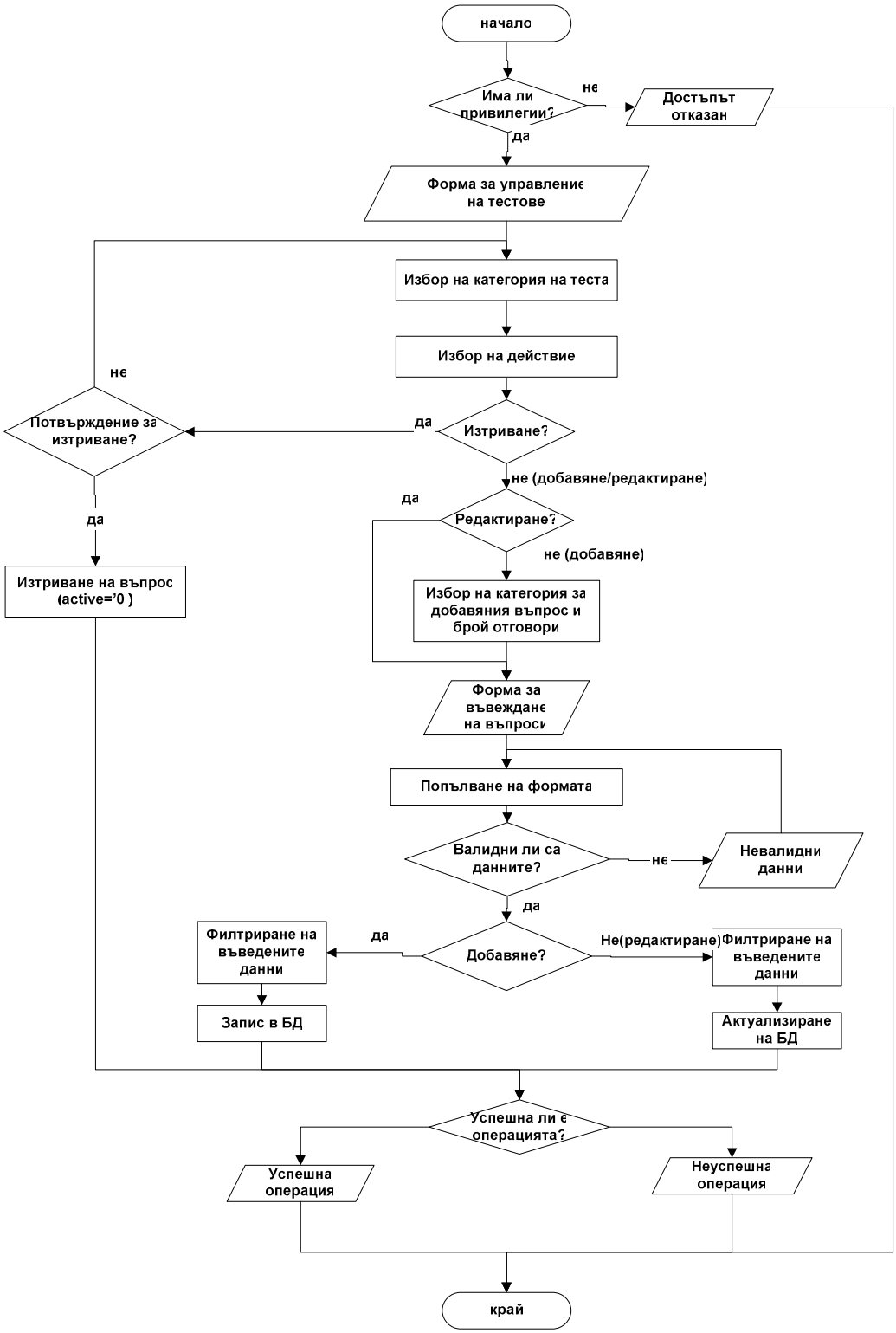
– Преглед на резултатите на положилите теста потребители – осъществява се с линк отварящ страницата quiz_admin_reports.php;

добавяне на нов въпрос – отваря се страницата quiz_addquestion.php. На първата стъпка се изобразява форма за избор на броя отговори, които ще се предложат на потребителя към тестовия въпрос. Следващата стъпка води към страницата quiz_editquestion.php, където се избира подтема, въвежда се текста на въпроса и отговорите и се посочва коректният отговор.

– Редакция на съществуващ въпрос – тази форма се обраотва отново от скрипта quiz_editquestion.php, но тук в появилата се форма се показват досегашните стойности на полетата.

– Изтриване на въпрос – формата се обработва от quiz_deletequestion.php. Тук се прави ъпдейт на базата данни, като се засяга както таблицата questions, така и таблица answers. На потребителя се връща съответното съобщение чрез скриптовете success.php или error.php, в зависимост от резултата.

Блок-схема на алгоритъма за съставяне и редактиране на тест е представена на фиг. 4.5.



Фиг.4.5. Алгоритъм за създаване/редактиране на тест.

4.3.5. Реализация на управление на допълнителните ресурси.

Системата позволява прибавяне на прикачени файлове, връзки, изображения към всяка от темите. За всеки един от изброените ресурси се предлага интерфейс за добавяне, редактиране и изтриване. Методите за управление са включени в

класовете Admin и FlesAdmin. Техни инстанции се правят във всеки един файл за менажиране на съдържанието. Ето по-важните от тези методи:

`Admin::edit_links_form($category_ID)` – прави заявка към БД за извличане на необходимите стойности за показване на текущото състояние на връзките: към коя катогория са, подтема, описващ текст, URL.

`Admin::add_edit_links($link_ID_or_new,$category_ID)`– прави проверка дали се добавя нова връзка или се редактира налична. Ако ще се променя наличен линк следва заявка към БД за извличане на текущите му параметри, показва формата с полетата за редакция. При добавяне на нов, формата е празна.

`Admin::insert_link($topic_ID,$url,$url_text,$link_text)` – служи за добавяне на нов ред във таблица links на базата данни. Връща `true` при успех.

`Admin::update_link($link_ID,$topic_ID,$link_text,$url,$url_text)` – променя налични данни в таблица links.

`Admin::delete_link($link_ID)` – Извиква се при изтриване на налични връзки.

Съществуват аналогични на последните три функции за останалите категории материали – изображения, файлове. Като се извикват с подходящите параметри в зависимост от атрибутите си.

`show_topic_pictures($topic_ID)` – проверява в базата наличието на снимки в дадена тема.

`upload_file($HTTP_POST_FILES,$destination_file_name)` – проверява дали избраният за добавяне файл в действително е изображение. Това е от изключителна важност за сигурността на сървъра.

На избраните от предлаганата форма за навигация изображения се променят размерите преди записването им в обозначена директория на сървъра, в случая /images/topics. Обработката се извършва от скрипта resize_image.php.

4.3.6. Реализация на полагане и проверка на тест.

Тази функционалност се осигурява от класа Quiz. Ключова роля има функцията `show_questions()`. Функцията приема за параметри `$quiz_category_ID` и `$email_result`. Прави заявка към таблиците questions, answers, topics и categories на базата данни. Целта е да се извлекат текстовете на въпросите заедно с прилежащите им отговори. Заявката е следната:

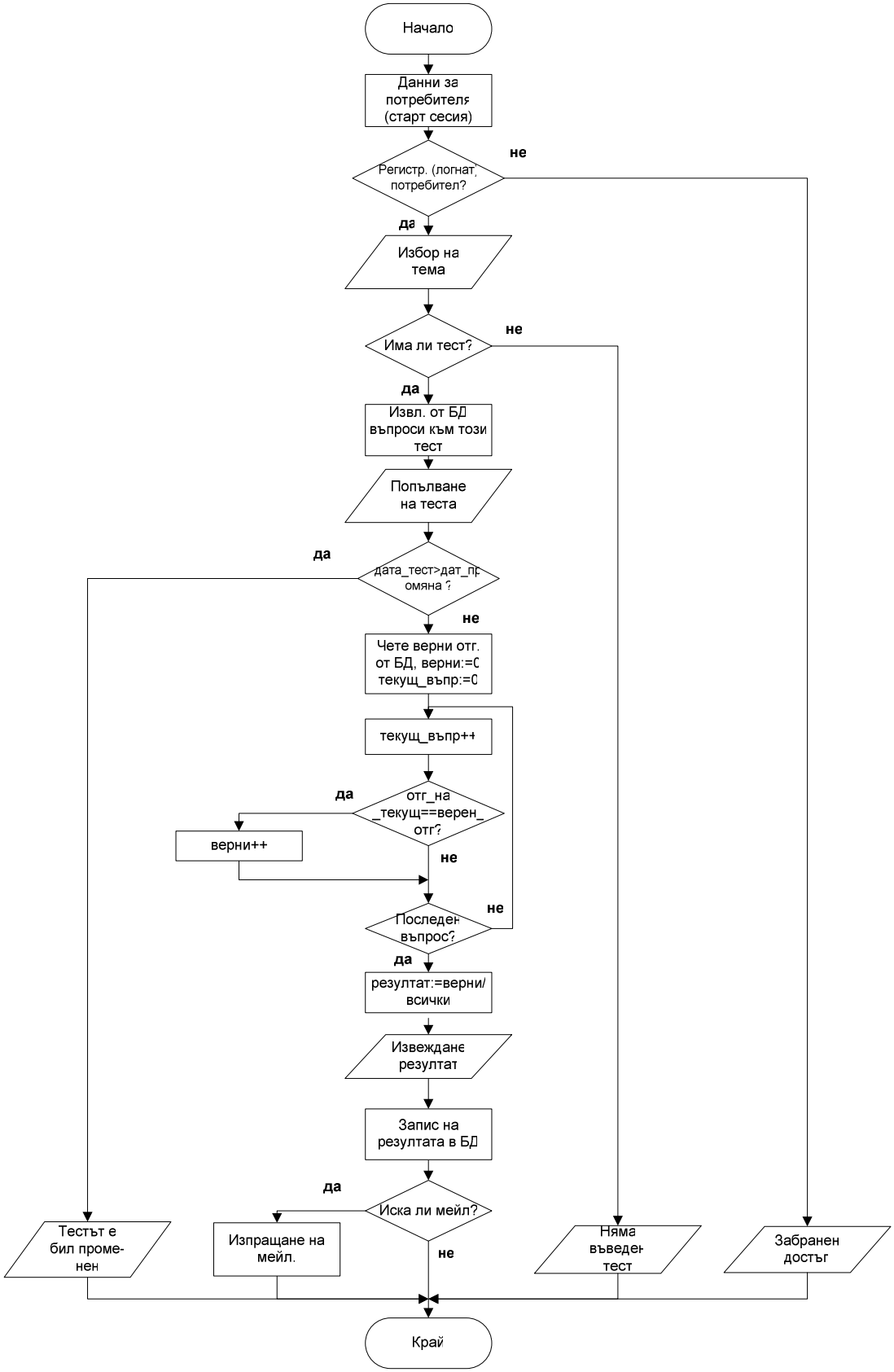
```
SELECT categories.quiz_update_date,questions.question_ID,
       questions.question_text,questions.has_pic,
       answers.answer_ID,answers.answer_text
FROM  questions,answers,topics,categories
WHERE categories.category_ID='$quiz_category_ID'
AND topics.category_ID=categories.category_ID
AND topics.topic_ID=questions.topic_ID
AND questions.question_ID=answers.question_ID
AND questions.active='1'
AND answers.active='1'
ORDER BY questions.question_ID;
```

Изобразява се форма с теста, след попълване се обработва от скрипта quiz_results.php. Важен момент е дали теста е бил променян от администратора, докато някой го е попълвал. Това би довело до некоректни резултати, затова са взети мерки да се избегне. Функцията `check_quiz_compatibility()` прави нова заявка към базта данни и сравнява прочетената дата на последна промяна на теста с прочетената стойност на това поле при заявката за извличане на теста.

Проверката и изчисляването на резултата се осъществява от функцията `check_quiz_result()`. Потребителят е изпратил формата, в която е попълнил масив от своите отговори. В същата форма, в скрито поле се записва и датата, в която последно е променян теста (`quiz_update_date`). Функцията check_quiz_result извлича със заявка до базата данни масив от верните отговори за съответния тест и датата на последната му промяна. С функция `check_quiz_compatability()` се проверява дали теста не е бил променян по време на работата на потребителя:

```
if($quiz_update_date_current===$quiz_update_date_taken)
return true;
else return false;
```

Извлечената от базата данни дата се сравнява с предадената през формата и ако няма съвпадение се връща съобщение за грешка, без да продължи проверката на резултата. При съвпадащи дати се прави проверка. В масив `wrong_questions[]` се пази информация за грешните въпроси на потребителя. След обхождането на целия масив, тази информация се запазва в таблица results. Блок схема на алгоритъма изобразява фигура 4.6.



Фиг. 4.6. Алгоритъм при правене на тест и изчисляване на резултата.

Потребителят разполага с ограничено време за полагането на теста. Отчитането на времето е направено от страна на клиента (за удобство и по-малко трафик) и от страна на сървъра (за сигурност).

Клиентската част е реализирана в скрипта `timer.js`. При стартиране на страницата за теста, в метода `onLoad` на HTML елемента `<body>` се зарежда Javascript, който брои (с брояч `timeoutSecs`) до определена стойност (`timerTimeout`) и когато я достигне, автоматично предава формата за теста: `document.timerForm.submit();`.

Във формата на теста има поле (`<input type="text" name="timerField" size=10 value =""/>`), което показва колко време е минало и се управлява от javascript-а. В скрито поле `<input type="hidden" name="timerTimeout" value ="$secondsTimeout"/>` се пази колко секунди ще е продължителността на теста – определя се от броя на отговорите на теста.

Недостатък на клиентската част е, че потребителят може да изключи Javascript-а или да го измени, затова е реализирана и сървърна част. При показването на теста се пази сесийна променлива с текущото време(`start_time`) (в секунди) и още една сесийна променлива за това колко ще продължи теста (`secondsTimeout`)

```
$HTTP_SESSION_VARS['secondsTimeout']=$secondsTimeout;
if(!$HTTP_SESSION_VARS['start_time'])HTTP_SESSION_VARS['start_time']=time();
```

Ако `start_time` вече е зададено, не му се дава нова стойност, по този начин когато се презареди страницата, няма да се нулира времето на теста (въпреки, че javascript-ът ще започне да показва часовника от 0).

В quiz_result.php се взима нова стойност `end_time` и се изчислява времето, за което е правен теста в секунди: като е отчетено и времето за трансфер клиент-сървър, през което потребителят реално не попълва теста:

```
$timeWorked=$end_time-$start_time-30;
```

Ако това време е повече от допустимото за теста, то се генерира грешка и не се проверява теста (в този случай не би трябвало да се попадне при нормална работа със системата и JavaScript-а).

В случай, че се стигне до quiz_result.php времето на теста вече се нулира:

```
$HTTP_SESSION_VARS['start_time']=0;
```

Всеки потребител има възможност да проверява резултатите си. Скрипта quiz_reports.php представя съхранения в базата резултат за текущия потребител. Съхранява се последния резултат от тест към дадена тема. От формата с резултати се

предоставя линк към страницата quiz_report_detail.php. Функцията `show_quiz_admin_reports($category_ID)` чете от таблица `Results` на базата данни сгрешените от потребителя въпроси, отговора избран от него и верния отговор. Изобразяват се само сгрешените въпроси, като се посочва линк към текста на материала, обясняващ тематиката на въпроса. Линкът води до подтемата, в която е изяснен въпроса.

4.3.7. Реализация на търсенето в системата

Скриптът се извиква при натискане на бутона "Search" от главното меню, ако потребителят е въвел фраза за търсене. Самата фраза се подава като GET параметър, така че заявката за търсене може да бъде написана директно в URL адреса през браузъра. С JavaScript се проверява дали потребителят е въвел фраза преди да се изпрати към страницата с резултата. Въведената фраза се филтрира като се премахват всички излишни празни символи, с помощта на вградената в PHP функция `trim()`, както и всички символи които могат да попречат на правилното изпълнение на заявката в MySQL. Функция `show_search_result()` визуализира намерените в базата данни редове, в които се среща търсената фраза. Реализира се от скрипта search.php.

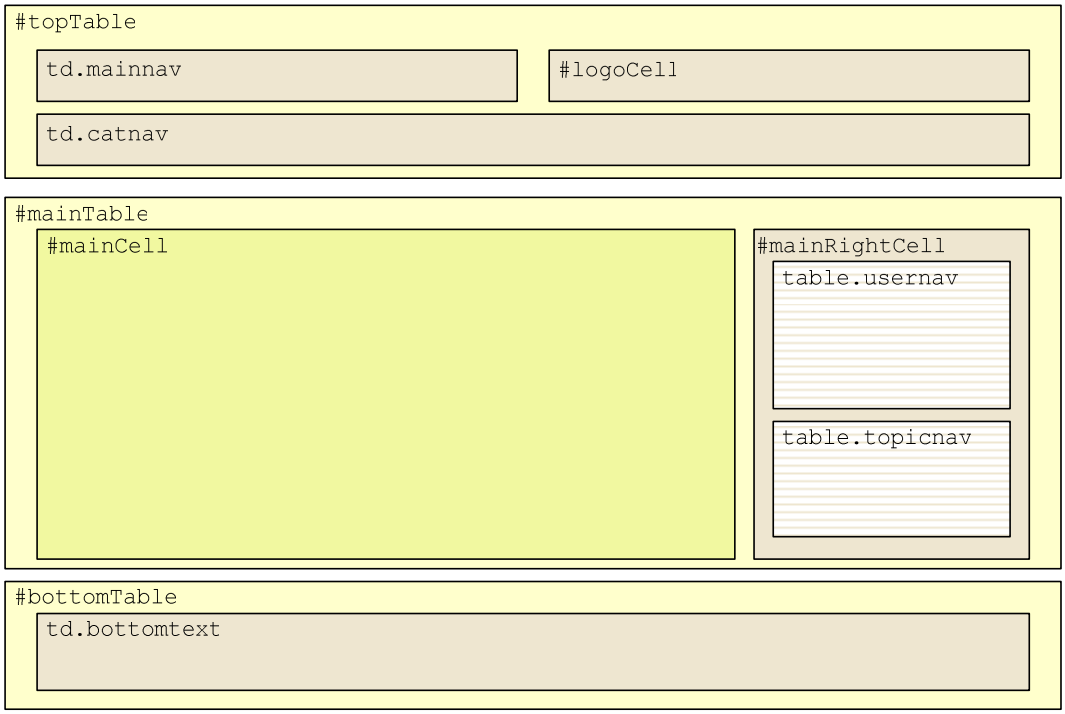
4.3.8. Дизайн на уеб интерфейса на системата

Дизайнът на системата е проектиран така, че клиентската част да е отделена от бизнес логиката на приложението. Клинетката страна на уеб интерфейса на системата е реализиран с HTML, CSS и JavaScript.

HTML код се връща като резултат от изпълнението на PHP скриптовете. Основните менюта, форми и статични части от страницата са отделени от самата програмам във външни файлове. Клас Page съдържа методи за обединението на тези външни страници за всяка страница.

Например, функция `Page::print_main_heading()`извиква файла header.inc, където са записани началните HTML тагове, дефинирани скриптове, икона, и мета данни във страницата, които са еднакви навсякъде.

Статично на сървъра се съхранява файл stle.css, където са описани каскадните стилове, които използва документа.



Фиг. 4.7. Дизайн на уеб интерфейса.

Самото съдържание на страниците е подредено с помощта на HTML таблици (фиг. 4.7). Таблиците имат фиксирана големина, която определя и общите граници на страницата. Всяка таблица има свой идентификатор записан в CSS файла и по този начин лесно може да се променя изгледа й. За изгледа на основните клетки на главните таблици са дефинирани класове в CSS. По този начин отново лесно, без да се променя основния код, може да се измени външния вид на страницата.

4.3.9. Дизайн на интерфейса за мобилни браузъри

Интерфейсът за мобилни браузъри е съобразен с по-малките системни възможности на мобилните телефони. Премахнати са някои мултимедийни файлове, които не могат да се визуализират правилно от мобилните устройства и се зареждат тежко. Изгледът на страниците е съставен с XHTML MP и WCSS по подобен начин на уеб интерфейса. В клас Mobile, който наследява Page, са предефинирани много от методите за извеждане на съдържание. Каскадните стилове са описани във файла mstyle.css, който се съхранява на сървъра. Ще разгледаме по-подробно как дългите текстове се разделят на няколко части.

Функция `Mobile:topic_description($topic_ID,$page,$page_size)` приема за параметри ID на текущата тема, текуща страница и големина на една страница. Чрез GET заявка при зареждане на страницата m-topics.php се изпраща на сървъра номера на текущата страница (ако не се изпрати нищо, значи това е първа страница).

Номерата на страниците системата отчита с начало 0. В параметър show_start се записва номера на първия символ от съдържанието, който ще бъде показан в текущата страница. Задължително условие е никоя дума от низа за показване да не е разделена в две страници. От една страна такова разделяне ще е неприятно за потребителите. Освен това низът съдържа и XHTML форматиращи тагове, разделянето на такъв таг, ще направи изходното съдържание невалиден XML и то няма да може да се визуализира от мобилния браузър. Поради тази причина show_start трябва винаги да е в началото или на позицията на празен символ („ „).

а) при първа страница show_start сочи в началото (show_start=0)

б) за друга страница, show_start взима позицията на първия срещнат празен символ („ „), след определения брой символи за текущата страница ($page *$page_size)

```
if($page==0) $show_start=0;
else $show_start=strpos($description,' ',$page*$page_size);
```

Позицията на последният низ от текущо показаната страница се записва в параметър show_end. Важно условие е show_end на дадена страница винаги да има еднаква стойност с show_start от следващата страница (за да не се отрязва съдържание или да се изписва по два пъти). Ако страницата не е достатъчно голяма или е стигнато до послдна страница, show_end взима позицията на последния низ в текста. Определянето на стойността на show_end става със следната извадка от код:

```
$show_end=strpos($description,' ',($page+1)*$page_size);
if($show_end==false) $show_end=strlen($description);
```

Като резултат, на екрана на мобилния телефон се публикува съдържанието, заключено между позициите на show_start и show_end.

```
$description_show=substr($description,$show_start,$show_end-$show_start);
```

4.3.10. Реализация на системата за съобщения

В приложението е предвидена възможност за заявка за данни чрез SMS и изпращане на SMS и MMS. Самото приложение не обработва получаването на SMS в SMS центъра, а само текста на полученото съобщение.

За симулиране на изпращане на съобщения към системата се използва формата sms_form.php. В текстово поле се въвежда текст, който генерира външно приложение за обработка на входящи SMS.

Самото изпращане на съобщение се прави чрез SMPP конекция към SMS център. Клас SMPPClass реализира методи за имплеметиране на SMPP протокола. В конструктура на класа са инициализирани основните параметри на съобщението –

време на изпращане и номер в идентификатор в опашката. Функция `Start($host, $port, $username, $password, $system_type)` отваря сокет към съответния порт и предава данните за автентификация на SMSC за стартиране на SMPP сесия. В класа е реализирано изпращането на необходимите PDU към сървъра:

- BIND_TRANSMITTER – за отваряне на сесията към SMSC;
- SUBMIT_SM – за изпращане на текста на съобщението;
- SUBMIT_MULTI – за изпращане на мултикаст суобщение;
- UNBIND – за прекъсване на сесията към SMSC;
- ENQUIRELINL – за проверка на състоянието на сесията.

Самото съобщение се предава с и се реализира с функцията `Send($to, $text, $unicode = false)`, която предава SMPP PDU SUBMIT_SM към SMS центъра. Клас SMPPSMS се грижи също за разделянето на големите съобщения, както и за кодирането им в Unicode.

В клас Message са реализирани методите за изпрашането на съобщения в системата. Обикновено текстово съобщение се изпраща с функцията `SMS_send($destination, $text,$type)`. Извършват се следните дейтвия:

1. `$oSMPP->Start($smpphost, $smppport, $systemid, $password, $system_type)` – стартира конекция към SMS центъра
2. `$oSMPP->TestLink()` – проверява дали е успешно отворена SMPP сесия
3. В зависимост от типа на съобщението предава самото съобщение. Типа може да бъде „single” – за обикновено съобщение (`$oSMPP->Send($destination,$text)`), „multi” – за съобщения към много потребители едновременно (`$oSMPP->SendMulti($destination, $text)`), “unicode” – за съобщение кодирано в Unicode (`$oSMPP->Send($destination, $text, true)`).
4. `$oSMPP->End()` – прекратява SMPP сесията.

Изпращането на Push съобщение от фунция `Push_SMS_Send($destination,$title,$push_url)` се извършва като се изпраща съобщение кодирано в WBXML. Кодирането се извършва от функция `encode_wappush`.

Функция MMS_Send($destination,$text,$img,$regions,$id) реализра изпращане на MMS – текст и изображения към определена дестинация. Функцията създава SMIL файл и го съхранява на сървъра, заедно с текстови файл на съобщението. След това изпраща към потребителя Push SMS с URL на SMIL файла.

За обработка на заявките на потребителите и генериране на необходимите съобщения се грижи функция `SMS_handler($source_number,$sms_text)`. Тя приема като параметър получен от потребител SMS, заедно с неговия номер. Първоначално се проверява дали номерът на потребителя е регистриран в системата. Ако номерът е регистриран, се изпращат няколко типа съобщения в зависимост от заявката:

- при получено съобщение с текст „DEFINE <topic>” – системата връща MMS с текста и картинка от съответната тема или резюме на няколко теми, ако ключовата дума се съдържа в заглавието на повече от една тема;
- при получено съобщение с текст „RESULT <category>” - системата връща MMS с детайлна справка за избраната категория или статистика от всички тестове, ако не е избрана категория или е избраната категория не съществува;
- при получено съобщение с текст „HELP”, системата връща SMS с описание на SMS командите;
- при получено друго съобщение, системата връща съобщение за грешка.

Глава 5. Ръководство за потребителя и приложения на проекта

5.1. Системни изисквания

Клиент:

- Процесор – минимум Intel 486 с тактова честота 75 MHz. Препоръчва се Intel Pentium 200 MHz, AMD K5 166 MHz или по-бърз.
- Оперативна памет – минимум 8 MB. Препоръчва се 64 MB или повече.
- Свободно място на твърд диск – минимум 20 MB. Препоръчва се 100 MB или повече.
- Дисплей – минимум разделителна способност 640x480 пиксела, 256 цвята. Препоръчва се 1024x768 пиксела и 16 бита (65 000) или повече цвята.
- Операционна система – без ограничения. Системата е изпробвана и работи на MS Windows 98, MS Windows 2000, MS Windows XP, Mac OS 10.4, Linux Mandrake 8.1, Linux GNU 1.6, Knoppix 3.4
- Интернет връзка – минимум 9,6 kbps симетричен трафик, препоръчва се 256 kbps или повече.
- Интернет браузър – необходимо е браузърът са спазва спецификациите на W3C и да поддържа JavaScript и CSS – системата няма да работи правилно върху версии на Internet Explorer преди 4.0 и на Netscape Navigator преди версия 3.0. За най-ефективна работа се препоръчва използването на браузър Mozilla Firefox 1.0+.
- Мобилен браузър версия WAP 1.2 или по-нова и активна услуга WAP или i-mode при мобилния оператор.
- Мобилен телефон, поддържащ SMS, WAP Push SMS и MMS и активна услуга SMS при мобилния оператор.

Сървър:

- Процесор – минимум Intel Pentium с тактова честота 1 GHz. Препоръчва се двуядрен процесор.
- Оперативна памет – минумум 1 GB. Препоръчва се 2 GB или повече.
- Твърд диск – минимум 50 MB, препоръчва се 100 MB или повече.
- Операционна система – без ограничения. Препоръчва се Linux Fedora Core 7.

- Интернет връзка – минимум 512 kbps upload. Препоръчва се 2 Mbps или повече.

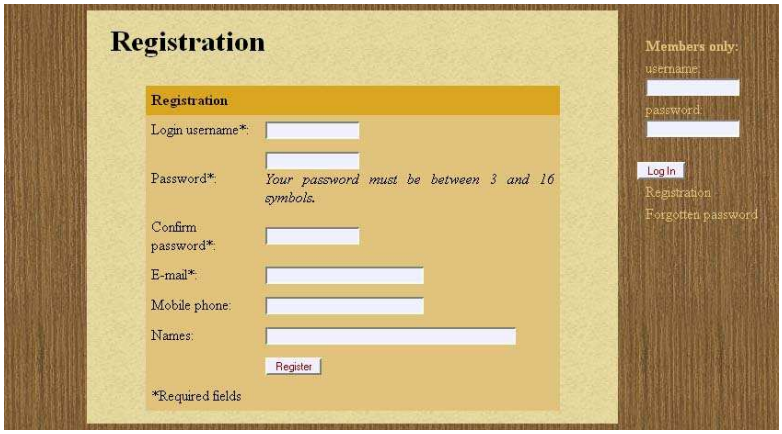
5.2. Потребителски интерфейс

Ще опишем действията, при които потребителят би имал някакви колебания или затруднения. Няма да се спираме подробно на възможностите за навигация в съдържанието на сайта, поради интуитивността на действията, който трябва да се предприемат.

5.2.1. Регистрация

1. Заредете сайта на приложението във Вашия Интернет браузър
2. Изберете регистрация от връзката в дясната колона на коя да е страница от сайта. (Фиг. 5.1.)
3. Изберете уникално потребителско име, e-mail и мобилен телефон. Полетата, отбелязани със звездички са задължителни. Потребителското име може да съдържа кирилица, латиница, "_" и цифрите от 0 до 9, паролата трябва да е с минимум 3 символа. Електронният адрес и телефон трябва да бъдат във валиден формат.(Фиг. 5.2.)

Фиг. 5.1. Меню за вход и регистрация.



Фиг. 5.2. Форма за попълване на данните при регистрация.

4. Натиснете бутона „Register". Извежда се съобщение показващо резултата. Ако данните не са въведени коректно се извежда в червен текст съобщения за полетата, които не са попълнени правилно.

5.2.2. Влизане в системата

1. Заредете сайта на приложението във Вашия Интернет браузър.

2. Попълнете поптребителското си име и парола на Вашия акаунт във формата за влизане в лявата част на страницата. (Фиг. 5.1.)

3. Натиснете бутона за потвърждение.

5.2.3. Възстановяване на изгубена парола.

1. Изберете връзката „Forgotten password“ в дясната част на страницата. (Фиг. 5.1.)

2. Попълнете формата с потребителското си име и e-mail за генериране на нова парола. (Фиг5.3.)

Фиг. 5.3. Форма за възстановяване на парола.

3. Отворете изпратения от системата e-mail от Вашата пощенска кутия.

4. Влезте в системата с Вашето потребителско име и получената временна парола.

5. Изберете връзката „Change password“ от главното меню. (Фиг. 5.4.)

6. Въведете желана от вас нова парола.

Фиг. 5.4. Главно потребителско меню.

5.2.4. Управление на съдържанието.

Съдържанието в системата се управлява от администратора, като до менютата и формите може да се достигне интуитивно по различни начини. Ще покажем как това се извършва като се започва от главното меню за администратора.

Добавяне и редактиране на текстове.

1. Влезте с администраторски акаунт.

2. Изберете „Content management“ от менюто на администратора. (Фиг. 5.5.)

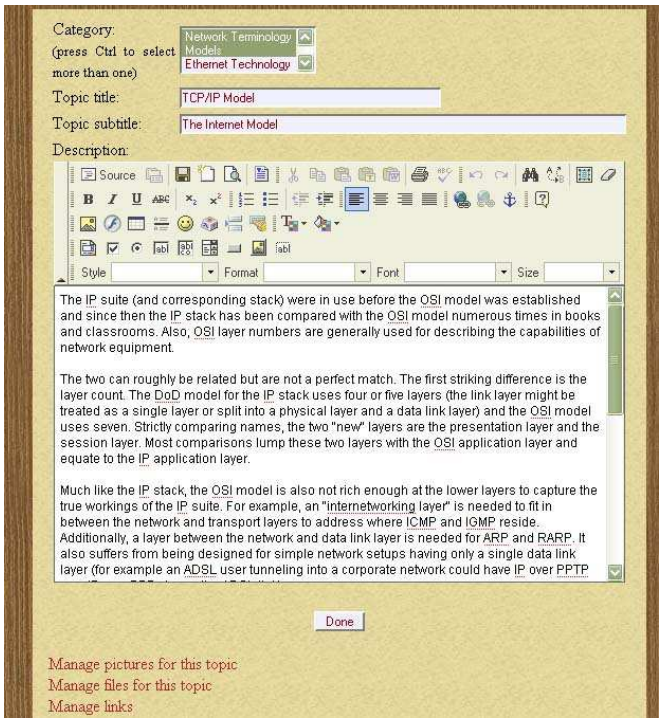
3. Изберете желаната от вас категория и тема от менюто за упрваление на съдържанието. (Фиг. 5.6.)

Фиг. 5.5. Меню за администратор.

Фиг. 5.6. Меню за управление на съдържанието.

4. Изберете бутона за добавяне, редактиране или изтриване менюто за управление на съдържанието, в зависимост от действието, което искате да извършите.

5. Добавете/редактирайте съдържание в предоставената форма. (Фиг. 5.7.)



Фиг. 5.7. Форма за въвеждане и редактиране на съдържанието.

6. Изберете бутона Done, за да завършите редакцията или Delete, ако желаете да изтриете съответната страница.

Управление на файлове.

1. Влезте с администраторски акаунт.

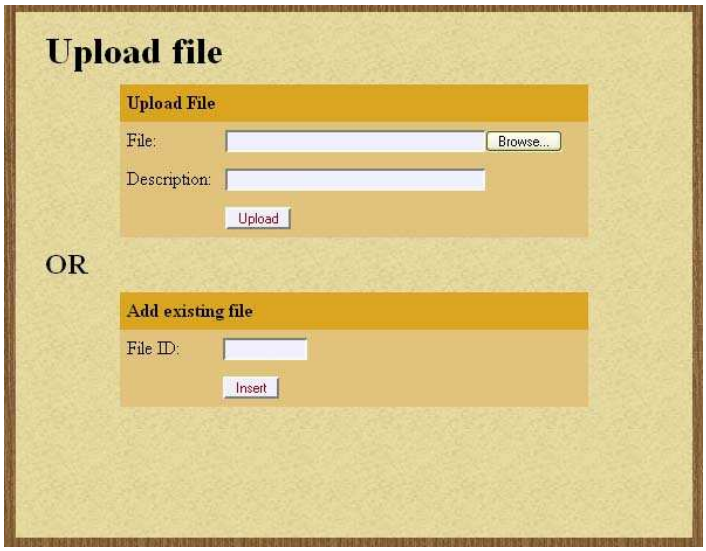
2. Изберете „Files management“ от менюто на администратора. (Фиг. 5.5.)

3. Изберете от падащото меню категория и тема, за която искате да добавяте файл. (Фиг. 5.8.)

Фиг. 5.8. Меню за управление на файлове.

4. Изберете връзката Add new file, разположена под падащото меню.

5. а) Натиснете бутона Browse от появилата се форма (фиг. 5.9) и добавете файл за прикачване от вашия компютър и добавете описание.



Фиг. 5.9. Меню за добавяне на файлове.

5 б) Изберете ID на файл, който вече е качен в системата, за да го използвате отново в тази страница.

6. Натиснете Upload или Insert, за да завършите добавянето.

Изтриване на файлове.

1. Влезте с администраторски акаунт.

2. Изберете „Files management“ от менюто на администратора. (Фиг. 5.5.)

3. Изберете от падащото меню категория и тема, за която искате да управявате файловете (фиг. 5.8.), ще се покажат всички файлове, които са свързани с категорията.

4. а) ако искате да изтриете файл само от тази тема, изберете бутон „Remove from this topic” под съответния файл. (Фиг. 5.8.)

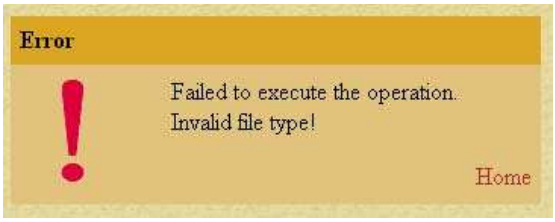
4. б) ако искате да изтриете файла от системата, изберете опция „Delete”. (Фиг. 5.8.)

5. Отговорете с OK на менюто за потвърждение за изтриване. (Фиг. 5.10.)

Фиг. 5.10. Потвърждение за изтриване на файл от тема.

Добавяне и изтриване на изображения.

Извършва се по аналогичен начин на добавянето на файлове. Извършва се проверка дали прикаченият файл е изображение и ако файлът е друг тип, той не се прикачва и излиза съобщение за грешка. (Фиг. 5.11.)



Фиг. 5.11. Съобщение за неуспешно прикачване.

Управление на линкове.

1. Влезте с администраторски акаунт.

2. Изберете „Links management“ от менюто на администратора. (Фиг. 5.5.)

3. Изберете от падащото меню категория и тема, за която искате да управявате линковете, ще се покажат всички линкове, които са свързани с категорията.

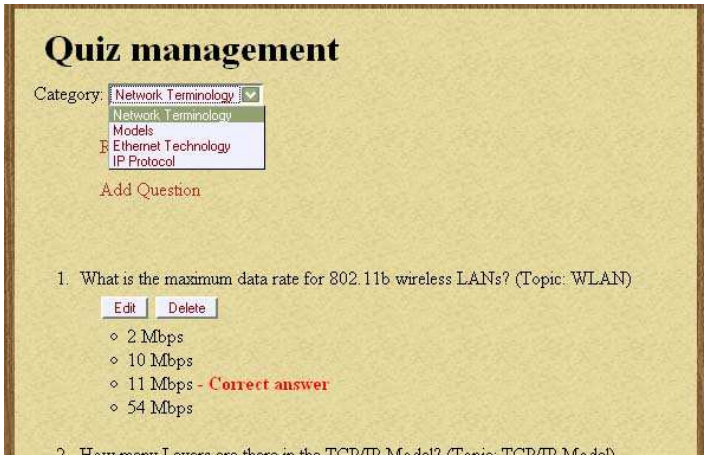
4. Изберете върху даден линк Delete за изтриване, Edit, за промяна или Add Link, за да добавите нова връзка.

5. Въведете данните за връзката и натиснете бутона Done. (Фиг. 5.12.)

Фиг. 5.12. Форма за въвеждане на линк.

Създаване на тест.

1. Влезте с администраторски акаунт.
2. Изберете „Quiz management“ от менюто на администратора. (Фиг. 5.5.)
3. Изберете от падащото меню категория на теста. (Фиг. 5.13.)

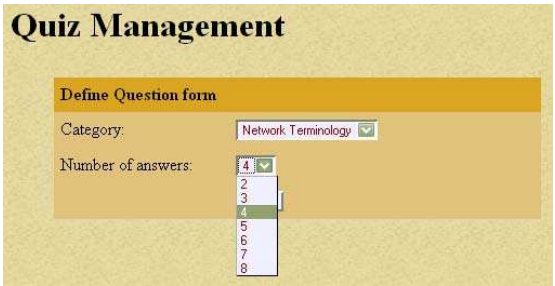


Фиг. 5.13. Форма за редактиране на тест.

4. Появяват се текущите въпроси от теста и линк за добавяне на нов въпрос (фиг. 5.12.), изберете да редактирате въпрос да изтриете или да добавите нов от съответните бутони.

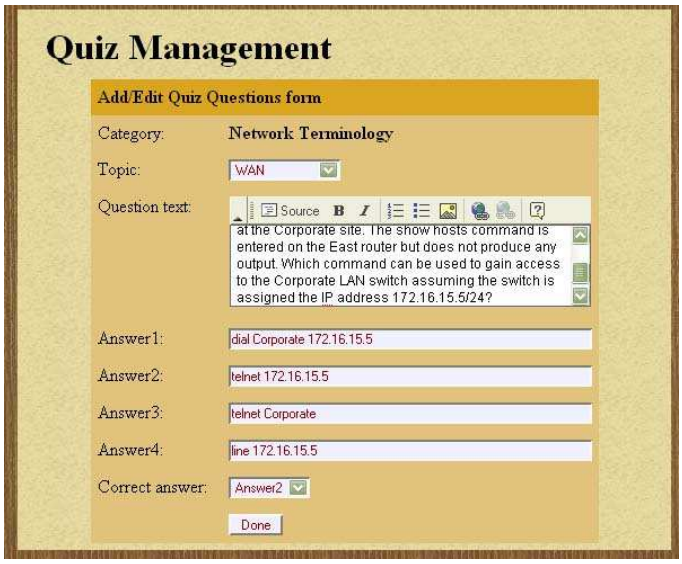
5. Изберете броя възможни отговори за въпроса и натиснете бутона Next. (Фиг.5.14.)

6. Изберете темата (topic), за който се отнася логически въпроса. Въведете текста



Фиг. 5.14. Съставяне на въпрос - 1

на въпоса и отговорите. От падащото меню изберете кой да бъде верният отговор. (Фиг. 5.15.)



Фиг. 5.15. Съставяне на въпрос - 2

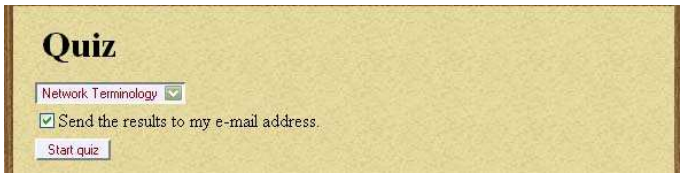
7. Изберете бутон Done, за да вкарате въпроса в системата.

5.2.5. Решаване на тест

1. Влезте в системата с Вашия акаунт.

2. Изберете връзката Quiz от потребителското меню. (Фиг. 5.4.)

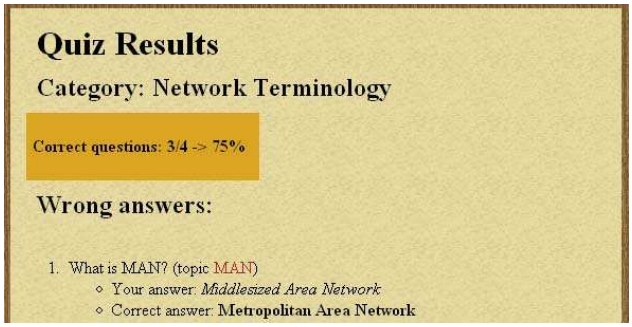
3. Изберете от падащото меню категория, на която да положите теста. (Фиг. 5.16.)



Фиг. 5.16. Меню за избор на тест.

4. Изберете отметката отдолу, ако желаете да получите писмо с резултата. Натиснете „Start quiz”, разположен под падащото меню. (Фиг. 5.15.)

5. Попълнете теста, като селектирате верният отговор. Непопълнените полета ще се отчетат като грешни. Натиснете Submit.



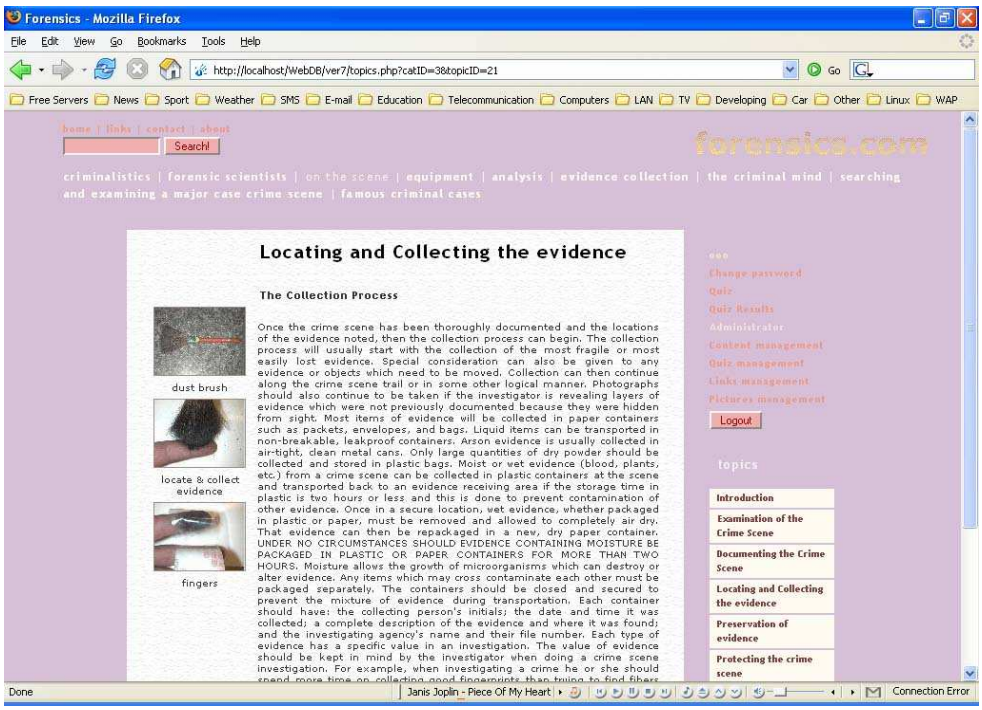
Фиг. 5.17. Резултати от тест.

6. Ще видите Вашия резултат и връзка към подтемата, където е обяснен сгрешеният въпрос. (Фиг. 5.17.)

5.3. Приложения на проекта

Настоящият проект е прилаган в различни предметни области за няколко проекта за обучение.

5.3.1. Forensics



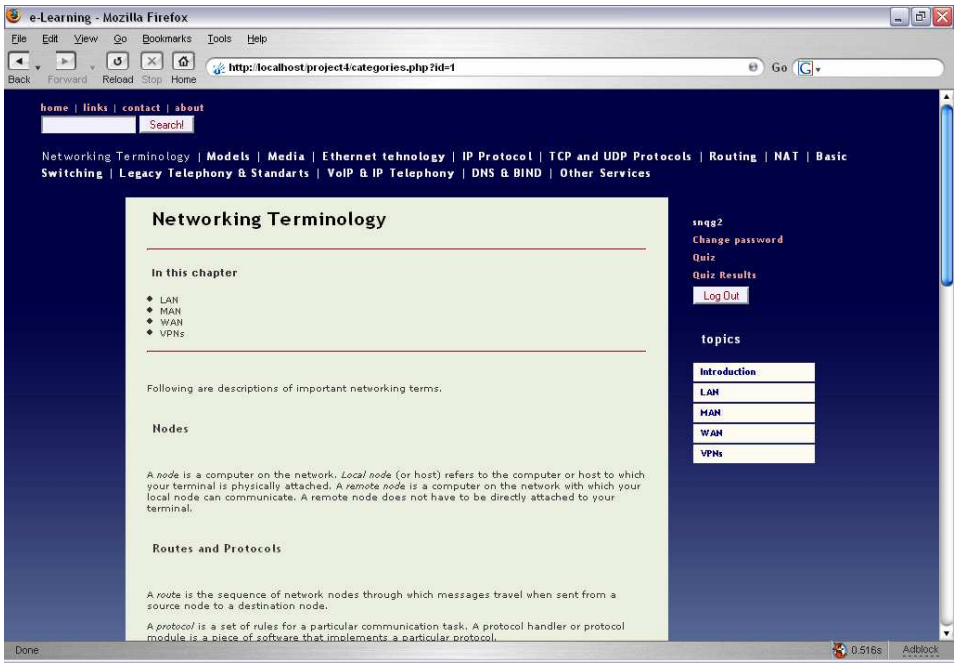
Фиг. 5.18. Графичен интерфейс на Forensics.

Проекта Forensics има за цел да систематизира и представи информация за различните технилогии и подходи в съвременната Криминалистика. В категориите са групирани различните етапи при работата по разкриването на криминално престъпление. Подробностите по всеки етап са записани в теми. Изгледът на системата е показан на фиг. 5.18.

5.3.2. eClass

Този проект се използва за курс за обучение на нови служители в Център за управлението на мрежата на телекомуникационен оператор. В отделните категории са поместени областите, които обхваща работата на служителите. Като теми се

записват описания на термини и различни подходи при отстраняване на повреди. Преподавателят използва възможността за прикачване на файлове, за да добавя различни PDF документи, с които е необходимо новите служиетели да се запознаят. Фирмата има възможност да използва и системата за тестове, за да проверява знанията на служителите си.



Фиг. 5.19. Графичен интерфейс на eClass.

5.3.3. Babylon

В този проект системата е използвана за създаването на Интернет страница на курс за обучение по немски език. Отделните категории представялват уроците на курса. Съдържанието на всеки урок е разделено на части(страници), всяка от които е записана като „тема” в системата. Тестовете се използват за проверка на знанията на учениците.

Заключение и възможности за разширение

Изработеното приложение постигна първоначално поставените задачи и успешно реализира информационна система с разнообразни интерфейси за достъп, чрез съвременни комуникационни технологии. Системата е успешно приложена в няколко проекта от различно естество, с което постигна и другата основна цел – еднакъв начин на работа, независим от предметната област.

За усъвършенстване на настоящия продукт биха могли да се направят следните разширения:

– Интерфейс за комуникация с IP телефонна централа (напр. Asterisk) и поддръжка на интерактивно гласово меню (Interactive Voice Response – IVR). По този начин информацията в системата ще бъде достъпна от мобилни и цифрови стационарни телефони с обикновено телефонно обаждане до определен номер.

– Възможност за самостоятелна обработка на получените SMS. В момента системата разчита на външно приложение да комуникира с мобилния оператор и обработва само текста на получените съобщения. За да обработва директно съобщенията, трябва да поддържа постоянно отворена сесия за асинхронни заявки от страна на SMSC.

– Интерфейс за Интернет браузър за управление на привилегиите на потребителите – в момента ролите на потребителите се променят чрез директни заявки към базата данни. Управлението на ролите директно през браузъра ще направи системата по-удобна за използване от страна на администратора.

– Изграждане на криптирана връзка (SSL) при изпращане на секретни данни до сървъра.

– Предоставяне на статистическа информация относно най-използваните в сайта материали.

Използвана литература

1. ATIS Committee Telecom Glossary – http://www.atis.org/tg2k/.

2. John F. Rockart et. Al (1996) Eight imperatives for the new IT organization; MIT Sloan Management review.

3. PHPXref.com Content Management Systems – http://phpxref.com/cms/.

4. Wikipedia – the free encyclopedia – http://en.wikipedia.org.

5. Heppell, S. (2007) Virtually There: Learning Platforms; Cleveratom.

6. Официална страница на Joomla – http://joomla.org.

7. Официална страница на проекта Mambo – http://mambo-foundation.org.

8. Официална страница на Елидо Медиа – http://elido.com.

9. Светлин Наков (2004) Интернет програмиране с Java; Фебер.

10. Атанас Георгиев (2006) Лекции по Уеб интерфейс към база данни, ФМИ.

11. Netcraft (2007) Web Server Survey – http://news.netcraft.com.

12. Люк Уелинг, Лаура Томсн (2004) – Разработване на проекти за Web с PHP и MySQL; СофтПрес.

13. Developer's Home website – http://developershome.com.

14. Open Mobile Alliance (2007) XHTML Specification; http://openmobilealliance.org.

15. Васил Георгиев (2006) Упражнения по WML/WAP/XHTML; ФМИ.

16. Matthew Caesar (2002) Architecture of i-mode; University of California.

17. GSM Doc 28/85 (1985) Services and Facilities to be provided in the GSM System.

18. SMPP Forum (1999) Short Message Peer to Peer Protocol Specification v3.4.

19. WAP Forum (2001) Binary XML Content Format Specification Version 1.3.

20. W3C Recommendation (2005) Synchronized Multimedia Integration Language 2.1.

21. C. J. Date (2003) An Introduction to Database Systems: Addison Wesley.

Приложение

Упражнения за курса по

WAP/WML/XHTML