**ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ**

Государственное образовательное учреждение высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» имени В.И. УЛЬЯНОВА (ЛЕНИНА)»

(СПБГЭТУ)

Кафедра ВТ

ОТЧЕТ ПО ПРАКТИКЕ

Тема практики:

«Изучение современных технологий веб-программирования»

Выполнил Краснов Д.С.

группа 9811

Руководитель от кафедры Чугунов Л.А.

Санкт-Петербург

2013

Оглавление

[1. Техническое задание 4](#_Toc355624925)

[2. Обзор современных веб-технологий 5](#_Toc355624926)

[2.1 Введение 5](#_Toc355624927)

[2.2 Языки программирования 6](#_Toc355624928)

[2.2.1 Серверные языки 6](#_Toc355624929)

[2.2.1.1 PHP 6](#_Toc355624930)

[2.2.1.2 ASP.NET 6](#_Toc355624931)

[2.2.1.3 JSP 7](#_Toc355624932)

[2.2.1.4 Несколько слов о CGI 7](#_Toc355624933)

[2.2.1.5 Python 7](#_Toc355624934)

[2.2.1.6 Perl 8](#_Toc355624935)

[2.2.1.7 Ruby 8](#_Toc355624936)

[2.2.2 Клиентские языки программирования 8](#_Toc355624937)

[2.2.2.1 JavaScript 8](#_Toc355624938)

[2.2.2.2 Отдельно об AJAX 9](#_Toc355624939)

[2.2.2.3 Java 9](#_Toc355624940)

[2.2.2.4 ActionScript 9](#_Toc355624941)

[2.2.2.5 VBScript 9](#_Toc355624942)

[2.3 Языки разметки 10](#_Toc355624943)

[2.3.1 HTML 10](#_Toc355624944)

[2.3.2 CSS 10](#_Toc355624945)

[2.3.3 XML 10](#_Toc355624946)

[2.3.4 XSL 11](#_Toc355624947)

[2.3.5 SVG 11](#_Toc355624948)

[2.4 Хранение данных 12](#_Toc355624949)

[2.4.1 SQL 12](#_Toc355624950)

[2.4.2 NoSQL 12](#_Toc355624951)

[2.4.2.1 Не используется SQL 12](#_Toc355624952)

[2.4.2.2 Неструктурированность 13](#_Toc355624953)

[2.4.2.3 Представление данных в виде агрегатов 13](#_Toc355624954)

[2.4.2.4 Распределённые системы 13](#_Toc355624955)

[2.4.2.5 Молодость 14](#_Toc355624956)

[2.4.2.6 Заключение 14](#_Toc355624957)

[2.5 HTTP-серверы 15](#_Toc355624958)

[2.5.1 Apache 15](#_Toc355624959)

[2.5.2 IIS 15](#_Toc355624960)

[2.5.3 Nginx 16](#_Toc355624961)

[2.6 Связанные технологии 17](#_Toc355624962)

[2.6.1 Кэширование 17](#_Toc355624963)

[2.6.2 Межсерверное взаимодействие 17](#_Toc355624964)

[2.7 В завершение обзорной части 17](#_Toc355624965)

[3. Проектирование 19](#_Toc355624966)

[3.1 Выбор связки технологий для написания приложения 19](#_Toc355624967)

[3.1.1 Веб-технологии 19](#_Toc355624968)

[3.1.2 Программное обеспечение 19](#_Toc355624969)

[3.2 Структура приложения 20](#_Toc355624970)

[3.2.1 Архитектура 20](#_Toc355624971)

[3.2.2 Структура базы данных 20](#_Toc355624972)

[3.2.3 Каркас веб-приложения 21](#_Toc355624973)

[3.2.4 Класс для работы с БД 22](#_Toc355624974)

[3.2.5 Страницы 22](#_Toc355624975)

[3.2.6 Система авторизации и привилегии пользователей 22](#_Toc355624976)

[3.2.7 Защита от взлома 22](#_Toc355624977)

[4. Результат работы 24](#_Toc355624978)

[5. Заключение 28](#_Toc355624979)

[6. Приложение 29](#_Toc355624980)

[6.1 index.php 29](#_Toc355624981)

[6.2 inc.header.php 30](#_Toc355624982)

[6.3 inc.mainarea.php 31](#_Toc355624983)

[6.4 styles.css 32](#_Toc355624984)

[6.5 scripts.js 35](#_Toc355624985)

[6.6 createDB.php 36](#_Toc355624986)

[6.7 class.IQuestsDB.php 37](#_Toc355624987)

[6.8 class.QuestsDB.php 38](#_Toc355624988)

[6.9 quests.php 43](#_Toc355624989)

[6.10 done.php 44](#_Toc355624990)

[6.11 comments.php 45](#_Toc355624991)

[6.12 editQuest.php 47](#_Toc355624992)

[6.13 editComment.php 48](#_Toc355624993)

[6.14 addQuest.php 49](#_Toc355624994)

[6.15 addComment.php 50](#_Toc355624995)

[6.16 deleteQuest.php 51](#_Toc355624996)

[6.17 deleteComment.php 52](#_Toc355624997)

[6.18 markDone.php 53](#_Toc355624998)

[6.19 markUndone.php 54](#_Toc355624999)

[6.20 register.php 55](#_Toc355625000)

[6.21 login.php 56](#_Toc355625001)

[6.22 logout.php 57](#_Toc355625002)

[6.23 addUser.php 58](#_Toc355625003)

# Техническое задание

Разработать серверную часть веб-приложения – планировщика задач для использования внутри отдела.

В приложении реализовать функционал для добавления, редактирования и комментирования заданий, систему аутентификации, возможность пометить задание выполненным и возможность раздельного просмотра текущих и выполненных заданий.

# Обзор современных веб-технологий

## Введение

С 6 августа 1991 года, когда на первом в мире веб-сервере был размещён первый в мире веб-сайт, веб-технологии проделали огромный путь и сильно ушли вперёд: от единственного языка разметки (тогда ещё чисто логической) HTML 1.0, единственного браузера Mosaic и единственного веб-сервера CERN httpd до бесчисленного множества разнообразных по своему назначению стандартов, протоколов передачи данных, серверов, языков программирования и программ.

Разобраться во всём этом разнообразии технологий и охватить его целиком в рамках данной практики не представляется возможным, но давайте сделаем попытку разложить по полочкам хотя бы основные и наиболее часто используемые из них.

Ниже представлена примерная и неполная схема, в которой я постарался разделить веб-технологии на некоторые категории по признаку их назначения и привёл в каждой категории наиболее частоупотребимые и значимые из них.

Далее я собираюсь рассказать понемногу о каждой выделенной мной категории.

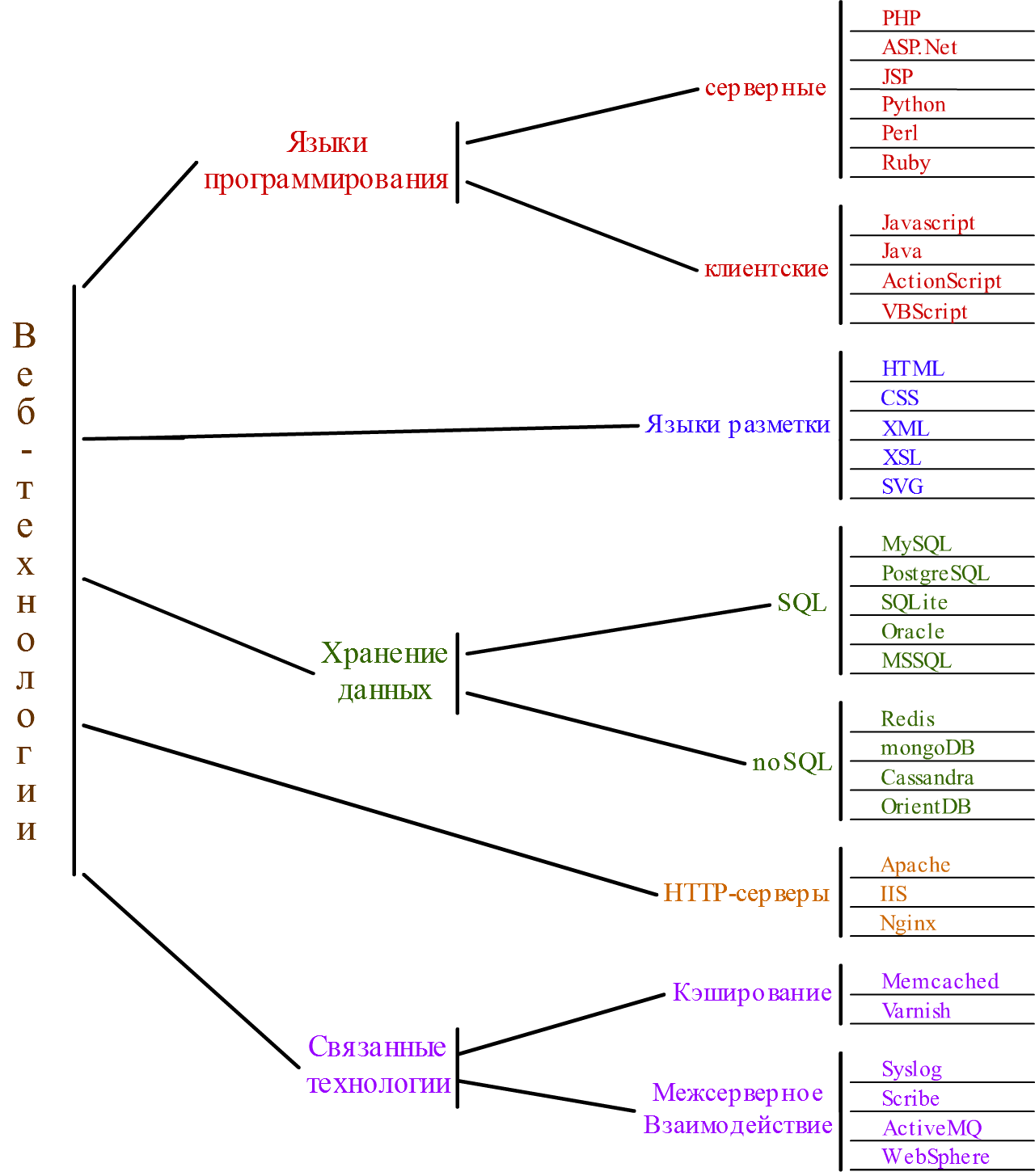


Рис. 1 – Схема основных веб-технологий

## Языки программирования

Все языки веб-программирования можно условно разделить на 2 большие группы: серверные и клиентские.

Задача **серверных языков** веб-программирования – обработка страниц на сервере перед отдачей их пользователю: когда пользователь даёт запрос на страницу, сервер сначала выполняет все связанные с ней программы, и только потом отдаёт пользователю итоговый файл страницы.

Важной особенностью серверных языков являются встроенные в них средства взаимодействия с серверами баз данных, что позволяет в любой момент использовать хранящуюся на них информацию.

Программы на **клиентских языках** выполняются на стороне пользователя (обычно браузером). Клиентские языки позволяют манипулировать внешним видом страниц и выполнять разнообразные проверки (например, правильности ввода данных в поля форм) без отправки данных на сервер, что позволяет сэкономить время на пересылке этих данных и упростить работу серверных скриптов. Однако важно помнить, что с помощью клиентского языка невозможно повлиять на файлы, хранящиеся на сервере.

Далее будут кратко рассмотрены основные языки обоих категорий.

### Серверные языки

#### PHP

Первая версия языка PHP была разработана датчанином Расмусом Лердорфом в 1995 году для собственных нужд. Вторая версия, выпущенная в 1997 году, использовалась уже на 1% интернет-доменов мира (около 50 тысяч). Благодаря своей простоте, скорости выполнения и богатой функциональности язык PHP снискал большую популярность и сейчас является одним из самых распространённых языков создания динамических веб-страниц. Его синтаксис подобен синтаксису С, а в качестве основных его особенностей называются:

* автоматическое извлечение POST и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
* взаимодействие с большим количеством различных систем управления базами данных (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape и Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird / InterBase, Paradox File Access, MaxDB, Интерфейс PDO);
* автоматизированная отправка HTTP-заголовков;
* работа с HTTP-авторизацией;
* работа с cookies и сессиями;
* работа с локальными и удалёнными файлами, сокетами;
* обработка файлов, загружаемых на сервер;
* работа с XForms.

#### ASP.NET

ASP.NET – технология создания веб-приложений и сервисов, выпущенная Microsoft как часть .NET Framework и основанная на Common Language Runtime (CLR) – среде выполнения кода на языке CIL, в который компиляторы .NET переводят программы, написанные на любом из более 100 поддерживаемых .NET языков.

Такой подход даёт программисту возможность писать серверную часть своего сайт на том языке, на котором он привык работать, хотя на практике чаще всего используются языки Visual Basic.NET и C#. Также, он позволяет значительно ускорить по сравнению со скриптовыми технологиями выполнение кода, т.к. код компилируется один раз при первом обращении и впоследствии только исполняется, не требуя времени на парсинг (синтаксический анализ). Кроме того, если знать модель классов .NET, написание веб-приложений и сервисов становится очень простым, быстрым и логичным.

И тут начинаются минусы. На изучение модели классов и всех необходимых функций требуется гораздо больше времени и сил, чем на изучение того же PHP. Родная среда ASP.NET – IIS, сервер от Microsoft, работающий под управлением Windows, и хотя существуют решения, позволяющие ASP.NET-приложениям работать под Linux, вся разработка таких приложений всё равно должна вестись в пакете Visual Studio от Microsoft, стоящем немало денег.

Из всего этого можно сделать вывод, что технология ASP.NET применима в основном для крупных проектов, для маленьких же – не приспособлена.

#### JSP

Технология JSP (JavaServer Pages) позволяет создавать страницы, имеющие как статические, так и динамические компоненты. При первом обращении весь код такой страницы транслируется в java-код, затем компилируется в байт-код виртуальной машины java (JVM), что позволяет значительно уменьшить время ответа при следующих обращениях. Отмечу, что для разработки на JSP не требуются платные программы, а также то, что JSP является платформонезависимой, переносимой и легко расширяемой.

#### Несколько слов о CGI

CGI (Common Gateway Interface) – стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программа, работающая с веб-сервером по такому интерфейсу, может быть написана на любом языке программирования, который может работать со стандартными потоками ввода-вывода. Когда приходит запрос от пользователя, веб-сервер в соответствии со стандартом помещает его параметры в соответствующие переменные среды и запускает CGI-программу (скрипт), которая получает их, обрабатывает, формирует код страницы, которая будет отправлена пользователю и выдаёт его в стандартный поток вывода, откуда её забирает сервер и отправляет в качестве ответа на запрос.

Описанные далее языки Python, Perl и Ruby применяются в контексте веб-программирования именно с использованием CGI.

#### Python

Python, в отличие от всего вышеперечисленного, является языком программирования общего назначения. Он ориентирован на повышение производительности разработчика и читаемости кода, его синтаксис минималистичен, возможности при этом широки, а стандартная библиотека включает много полезных функций, при этом из-за удобочитаемости многократное использование и обслуживание кода на python значительно проще, чем при использовании других скриптовых языков, а объём этого кода в 3-5 раз меньше, чем эквивалентного, например, на C++ или Java, что означает меньший объём ввода с клавиатуры, меньшее количество времени на отладку и меньший объем трудозатрат на сопровождение. Благодаря своим качествам python имеет много поклонников.

Python – интерпретируемый, скриптовый язык.

Для применения python в веб-разработке существуют различные фреймворки, самый популярный из которых – Django, использующийся во многих крупных проектах.

#### Perl

Perl — интерпретируемый скриптовый язык программирования общего назначения, один из самых распространённых в области веб-программирования. Основной особенностью языка считаются его богатые возможности для работы с текстом, в том числе работа с регулярными выражениями, встроенная в синтаксис. Перл также знаменит огромной коллекцией дополнительных модулей на CPAN - крупнейшем хранилище модулей.

Из недостатков можно указать некоторые сложности при установке и работе с дополнительными модулями под Windows.

#### Ruby

Ruby — интерпретируемый скриптовый язык высокого уровня для быстрого и удобного объектно-ориентированного программирования. Ruby имеет большое количество средств для обработки текстов, для решения системных задач, перенесён на множество платформ. Многие особенности синтаксиса и семантики Ruby заимствованы его создателем, японцем Юкихиро Мацумото, из языка Perl. По словам создателя, целью разработки было создание «настоящего объектно-ориентированного», лёгкого в разработке, интерпретируемого языка программирования.

Язык Ruby обладает многими интересными особенностями, которые не встречаются в других языках программирования. Например, программист может динамически добавлять методы классам и даже отдельным их экземплярам (таким образом, может быть легко реализован, например, шаблон «одиночка»).

Самым известным и широко применяемым фреймворком на Ruby является Ruby on Rails, используемый во многих крупных проектах.

### Клиентские языки программирования

#### JavaScript

JavaScript - прототипно-ориентированный сценарный язык программирования. Он применяется для придания интерактивности веб-страницам. Сценарии, написанные на JavaScript, встраиваются прямо в код страницы (либо располагаются в отдельных файлах, указанных в коде страницы, которые браузер загружает с сервера вместе с ней) и исполняются самим браузером на компьютере клиента.

При помощи JavaScript можно манипулировать внешним видом, положением и содержанием отдельных частей страницы, обрабатывать события (нажатия кнопок мыши и клавиатуры, перемещение указателя мыши, изменение значения элементов форм, изменение размеров окна и т.д.), проверять правильность ввода данных форм и многое другое.

Язык JavaScript можно по праву назвать лидером среди остальных клиентских языков веб-программирования, а в общем случае используется и в других областях. Его интерпретатор интегрирован в любой современный браузер, а сценарии, написанные на нём, присутствуют в том или ином объёме практически на любой странице в интернете.

Следует упомянуть так же о популярной библиотеке JavaScript, называемой jQuerry, значительно облегчающей доступ к элементам веб-страницы, их атрибутам и содержимому и манипулирование ими. Также библиотека jQuery предоставляет удобный API по работе с AJAX, о котором далее.

#### Отдельно об AJAX

AJAX («Asynchronous Javascript and XML») не является языком клиентского программирования, это набор техник построения интерактивных веб-интерфейсов, с помощью которых можно организовать фоновый обмен данными между клиентом и сервером, что позволяет обновлять содержимое страницы без её перезагрузки. Такой подход позволяет сократить трафик, снизить таким образом нагрузку на сервер и ускорить реакцию интерфейса.

#### Java

Java — язык общего назначения, на нем можно писать самые разные программы. Для интернет-страниц есть особая возможность - написание апплетов.

Апплет — это программа на языке Java, которую можно подключить к странице. Апплет выполняется в отдельной части страницы, в прямоугольном «контейнере». Все действия пользователя внутри него обрабатывает апплет. Контейнер, впрочем, может быть и спрятан, если апплету нечего показывать.

Конечно, для этого на компьютере должна быть установлена и включена среда выполнения Java. Статистика показывает, что примерно на 80% компьютеров Java будет работать, но этого всё равно не достаточно для того, чтобы Java-апплеты использовались повсеместно. Их применяют только в специфических ситуациях, например когда требуемый функционал нельзя реализовать средствами JavaScript из-за его ограничений безопасности.

Для запуска Java-апплета требуется разрешение пользователя, запрашиваемое фиксированного вида окном.

#### ActionScript

ActionScript — объектно-ориентированный язык программирования, который добавляет интерактивность, обработку данных и многое другое в содержимое Flash-приложений. ActionScript компилируется в байт-код и включается в SWF-файл, исполняемый Flash Player-ом. Flash Player существует в виде плагина к веб-браузеру. SWF-файл можно подключить к HTML-странице и запустить в прямоугольном контейнере.

Эта технология интересна тем, что позволяет делать многое, что JavaScript пока не умеет, например работа с микрофоном, камерой, с буфером обмена. Она работает безопасно и не требует разрешения посетителя.

JavaScript и ActionScript могут вызывать функции друг друга, поэтому обычно сайты используют JavaScript, а там, где он не справляется - можно подумать о Flash.

#### VBScript

VBScript — это язык программирования от компании Microsoft, предназначенный для создания сценариев (скриптов). Он является подмножеством языка Visual Basic и широко используется при создании административных сценариев в системе Windows. На нём так же могут быть написаны клиентские скрипты на веб-страницах, но работать они будут только в браузере Internet Explorer от Microsoft.

Развитие языка и включение в него новых возможностей продолжалось до появления фреймворка .NET; на этом этапе он был встроен в ASP.NET, и выпуск новых версий языка прекратился.

## Языки разметки

Язык разметки в компьютерной терминологии — набор символов или последовательностей, вставляемых в текст для передачи информации о его выводе или строении. Текстовый документ, написанный с использованием языка разметки, содержит не только сам текст, но и дополнительную информацию о различных его участках, например, указание на заголовки, выделения, списки и т. д. В более сложных случаях язык разметки позволяет вставлять в документ интерактивные элементы и содержание других документов.

В веб-программировании используются 2 основных языка разметки, HTML и XML, и, в связке с ними, каскадные таблицы стилей CSS – язык описания внешнего вида документа, написанного с использованием языка разметки. Сам по себе CSS языком разметки не является, но я включил его в этот раздел ввиду его неразрывной связанности с темой раздела.

#### HTML

Язык HTML (HyperText Markup Language — «язык разметки гипертекста») является стандартным языком разметки страниц в интернете. Большинство веб-страниц размечаются именно при помощи него.

В языке HTML определён небольшой набор структурных и семантических элементов - дискрипторов («тегов»). Текст, заключенный между открывающим и соответствующим ему закрывающим тегами, приобретает определённое семантическое значение и одновременно становится элементом структуры страницы. Существуют также пустые элементы (без текста) и одиночные элементы, не требующие закрывающего тега. Имя тега задаёт тип элемента. От этого типа зависит то, как элемент будет отображаться в браузере. Элементы можно вкладывать один в другой, задавая таким образом их иерархию и структуру. Элементы также обладают атрибутами, задающими их свойства.

#### CSS

Как уже было сказано ранее, CSS (Cascading Style Sheets – «каскадные таблицы стилей») сам по себе языком разметки не является. CSS – язык описания внешнего вида HTML-страниц. Он был разработан с целью разделения описания логической структуры веб-страницы и описания её внешнего вида. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление или чтение голосом.

CSS может работать как со всей страницей целиком, так и с отдельными её элементами. Вот некоторые из его возможностей: управление размерами и положением блоков, их формой, рамками, цветом, отступами и фоном; управление текстом (выбор шрифта, его размера, цвета и т.д.); управление видимостью и прозрачностью элементов; и даже рисование2D-изображений.

Код CSS может быть включен в код страницы либо содержаться в отдельных файлах, имеющих формат CSS и подключаемых к странице при помощи тега <LINK>.

#### XML

Язык XML (eXtensible Markup Language — расширяемый язык разметки) разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка.

Целью создания XML было обеспечение совместимости при передаче структурированных данных между разными системами обработки информации, особенно при передаче таких данных через Интернет. XML предназначен для хранения структурированных данных, для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки (например, XHTML).

Основным недостатком XML является избыточность синтаксиса: при некоторых условиях оболочка данных может быть больше, чем сами данные. В подобных случаях предпочтение обычно отдаётся альтернативным языкам, таким, как JSON или YAML.

#### XSL

XSL (eXtensible Stylesheet Language) - семейство рекомендаций консорциума W3C, описывающее языки преобразования и визуализации XML-документов. Состоит из трех частей:

* XSL Transformations (XSLT) — язык преобразований XML-документов.
* XSL Formatting Objects (XSL-FO) — язык разметки типографских макетов и иных предпечатных материалов.
* XPath — язык путей и выражений, используемый в XSLT для доступа к отдельным частям XML-документа.

С помощью XSL можно трансформировать XML-документ в любой вид, будь то HTML, WML, RTF, PDF, SQL, или даже, а также XSL. XSL несёт в себе информацию о том, как будет оформлен документ, где и как должны располагаться данные.

При этом XSL – это не “CSS для XML”, т.к. в отличие от CSS все преобразования происходят на стороне сервера, а пользователь получает результат.

#### SVG

SVG (Scalable Vector Graphics — «масштабируемая векторная графика») - это язык описания масштабируемой векторной графики в формате XML, позволяющий создавать и анимировать графические объекты, текст, а также обеспечивать интерактивный режим взаимодействия пользователя с этими объектами.

## Хранение данных

Ранее, когда речь шла про CSS, мы говорили о разделении описания логической структуры страницы и её внешнего вида. Но кроме внешнего вида и логической структуры страницы содержат ещё и так называемый «контент» - информационную составляющую сайта, которую тоже удобно было бы отделить от каркаса сайта. Преимуществ у такого подхода много, для примера можно назвать более простое администрирование сайта, сравнительную лёгкость доработки функционала и дизайна. Кроме того, когда объёмы контента, т.е. данных, растут, возникает проблема скорости доступа к ним.

Для хранения контента сайта используют базы данных. Сервер базы данных хранит данные и предоставляет к ним доступ по запросу. Запросы осуществляет серверная сторона сайта, формируя на основе ответов сервера БД итоговые страницы, отдаваемые пользователю.

Обычно база данных представляет собой набор таблиц, в каждой из которых хранятся данные своего типа. Система управления базой данных (СУБД) предоставляет возможность делать выборки из этих таблиц, изменять либо удалять существующие данные и добавлять новые. Сами таблицы при этом загружаются в память сервера, что ускоряет доступ к данным

### SQL

Большинство современных баз данных, используемых для веб-разработки – реляционные, запросы к ним осуществляются на языке SQL (Structured Query Language — «язык структурированных запросов») – универсальном языке, применяемом для работы с реляционными базами данных.

Из самых популярных в применении к веб-разработке СУБД можно назвать MySQL, PostgreSQL, Oracle, MS SQL и SQLite. У каждой из них свои преимущества и недостатки, останавливаться на которых в рамках данной работы я не хочу. Скажу только, что сервер базы данных Oracle платный, реализация MS SQL существует только для Windows, а SQLite в отличие от остальных хранит всю базу в одном файле и представляет собою не сервер, а библиотеку функций доступа к этому файлу.

### NoSQL

NoSQL (not only SQL – «не только SQL») - термин, обозначающий ряд подходов, проектов, направленных на реализацию моделей баз данных, имеющих существенные отличия от используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.

Описание схемы данных в случае использования NoSQL-решений может осуществляться через использование различных структур данных: хеш-таблиц, деревьев и других. При этом NoSQL не подразумевает отрицание языка SQL, это лишь попытка расширить возможности БД там, где SQL недостаточно гибок, например при работе с данными очень большого объема и в проектах с высокой нагрузкой.

Под термином NoSQL скрывается множество очень разнообразных и разнородных систем, таких, как mongoDB, Cassandra, Voldemort, OrientDB, Redis, но тем не менее можно попробовать выделить ряд направлений, которым все они придерживаются, и особенностей, которыми обладают. Далее я постараюсь рассмотреть их поподробнее.

#### Не используется SQL

Имеется в виду ANSI SQL DML, так как многие базы пытаются использовать query languages похожие на общеизвестный любимый синтаксис, но полностью его реализовать не удалось никому и вряд ли удастся.

#### Неструктурированность

Жесткая регламентированность структуры данных реляционных БД не позволяет им быть достаточно гибкими для решения некоторых современных задач. NoSQL базы данных стараются решить эту проблему. В NoSQL БД, в отличие от реляционных, в отдельной строке или документе можно добавить произвольное поле без предварительного декларативного изменения структуры всей таблицы. Таким образом, если появляется необходимость поменять модель данных, то единственное достаточное действие — отразить изменение в коде приложения.

Недостатками такого подхода можно считать накладные расходы в коде приложения при смене модели данных, отсутствие ограничений со стороны базы и сложности понимания и контроля структуры данных при одновременной работе с базами разных проектов, однако если от базы данных нужна гибкость, с ними можно смириться.

#### Представление данных в виде агрегатов

В отличие от реляционной модели, которая сохраняет логическую сущность приложения в различные физические таблицы в целях нормализации, NoSQL хранилища оперируют с этими сущностями как с целостными объектами. Например, в стандартной модели «заказ — позиции заказа — платежи — продукт» удобно было бы добавлять часть информации о заказанных товарах (название, id и количество) прямо в сущность заказа. Такая денормализация может порождать избыточные данные в базе, но зато позволяет не запрашивать информацию о каждом продукте при извлечении заказа, что соответствует главному правилу распределённых систем – минимум объединений разных таблиц в одном запросе («джоинов»).

Таким образом, представление в виде агрегатов удобно при организации базы в распределённую систему, т.к. позволяет значительно уменьшить количество «джоинов» и, таким образом, увеличить скорость работы базы, но при этом такая база требует очень тщательной и продуманной оптимизации под наиболее частые запросы. Минусом такого подхода является то, что оптимизируя базу под одни запросы можно значительно увеличить сложность и время выполнения других, но в случае распределённых систем это компромисс, на который можно пойти.

#### Распределённые системы

Это, возможно, главный лейтмотив развития NoSQL баз. С лавинообразным ростом информации в мире и необходимости ее обрабатывать за разумное время встала проблема вертикальной масштабируемости — рост скорости процессора остановился на 3.5 Ггц, скорость чтения с диска также растет тихими темпами, плюс цена мощного сервера всегда больше суммарной цены нескольких простых серверов. В этой ситуации обычные реляционные базы, даже кластеризованные на массиве дисков, не способны решить проблему скорости, масштабируемости и пропускной способности. Единственный выход из ситуации — горизонтальное масштабирование, когда несколько независимых серверов соединяются быстрой сетью и каждый владеет/обрабатывает только часть данных и/или только часть запросов на чтение-обновление. В такой архитектуре для повышения мощности хранилища (емкости, времени отклика, пропускной способности) необходимо лишь добавить новый сервер в кластер — и все. Процедурами шардинга, репликации, обеспечением отказоустойчивости (результат будет получен даже если одна или несколько серверов перестали отвечать), перераспределения данных в случае добавления ноды занимается сама NoSQL база. Вкратце представлю основные свойства распределенных NoSQL баз:

**Репликация** — копирование данных на другие узлы при обновлении. Позволяет как добиться большей масштабируемости, так и повысить доступность и сохранность данных. Принято подразделять на два вида:

* Master-slave - предполагает хорошую масштабируемость на чтение, но немасштабируемую запись
* Peer-to-peer - предполагается, что все узлы равны и могут обслуживать как запросы на чтение, так и на запись

**Шардинг** — разделение данных по узлам. Каждый узел отвечает за определённый набор данных и непосредственно обрабатывает запросы клиента на изменение/чтение. Шардинг часто использовался как “костыль” к реляционным базам данных в целях увеличения скорости и пропускной способности: пользовательское приложение партицировало данные по нескольким независимым базам данных и при запросе соответствующих данных пользователем обращалось к конкретной базе. В NoSQL базах данных шардинг, как и репликация, производятся автоматически самой базой и пользовательское приложение обособленно от этих сложных механизмов.

#### Молодость

NoSQL базы данных в основном созданы относительно недавно, поэтому не могут похвастаться многолетними наработками, не все их проблемы ещё решены и не все задумки реализованы. Однако несмотря на это разработчики всё чаще используют их в своих проектах, в том числе достаточно крупных, как замену реляционных баз данных либо дополнение к ним, потому что есть ряд задач, которые не под силу традиционным реляционным базам.

#### Заключение

NoSQL движение набирает популярность гигантскими темпами. Однако это не означает, что реляционные базы данных становятся рудиментом или чем-то архаичным. Скорее всего они будут использоваться и использоваться по-прежнему активно, но все больше в симбиозе с ними будут выступать NoSQL базы. Мы вступаем в эр, когда для различных потребностей используются разные хранилища данных. Теперь нет монополизма реляционных баз данных, как безальтернативного источника данных. Все чаще архитекторы выбирают хранилище исходя из природы самих данных и того, как мы ими хотим манипулировать, какие объемы информации ожидаются.

## HTTP-серверы

Веб- или HTTP-сервер – это сервер, принимающий HTTP-запросы от клиентов (обычно веб-браузеров), и выдающий им HTTP-ответы, обычно содержащие HTML-страницы, изображения, файлы, медиа-потоки или другие данные.

Веб-сервером может называться как программа, выполняющая эти функции, так и сам компьютер, на котором она установлена.

Существует достаточно большое количество веб-серверов.

#### Apache

Самым популярным из веб-серверов на сегодняшний день является Apache - свободный веб-сервер, используемый на 51% сайтов (04.2013, netcraft.com), для которых есть подобная информация.

Хоть Apache и будет работать на любой операционной системе, его намного чаще используют в комбинации с операционной системой Linux, которая также распространяется с открытым исходным кодом. Эти два компонента, объединенные с MySQL, базой данных PHP и языком сценариев, образуют популярное решение для веб-сервера под названием LAMP (комплекс серверного программного обеспечения).

Основные преимущества Apache и серверных систем LAMP:

* низкие затраты, так как нет необходимости приобретать лицензии на программное обеспечение;
* гибкое программирование из-за открытого исходного кода;
* улучшенная безопасность, так как Apache был разработан для операционной системы не-Windows (а большинство вредоносных программ написано под ОС от Microsoft), то он всегда имел репутацию более безопасной системы, чем IIS производства Microsoft.

#### IIS

IIS от Microsoft – второе по популярности (20%, 04.2013, netcraft.com) программное обеспечение для веб-сервера. Он состоит из серии служб, (FTP, HTTP, SMTP и другие), которые позволяют системе Windows управлять веб-сайтами. Последняя версия также включает различные модули для безопасности, кэширования, журналирования, сжатия и диагностики.

Поскольку IIS работает только с ОС Windows, то выбор IIS подразумевает под собой и выбор Windows Server, что, несомненно, увеличивает эксплуатационные расходы. Плюс Windows подвержен большему количеству вредоносных атак на программное обеспечение и имеет репутацию менее защищенного сервера, чем Apache.

Плюсы IIS:

* Windows и IIS поддерживаются корпорацией Microsoft.
* IIS поддерживает платформу.NET Microsoft и сценарии ASPX.
* модули позволяют включить потоковую передачу аудио и видеоконтента.

Минусы:

* Стоимость лицензий
* Меньшая, чем у Apache, безопасность
* Работа только под ОС Windows

#### Nginx

Третье место (15%, 04.2013, netcraft.com) по популярности у nginx разработки российского программиста Игоря Сысоева. Nginx является кроссплатформенным, но также чаще используется с ОС Linux. Это простой, быстрый и надёжный сервер, разработанный для проектов с большой нагрузкой. Часто на сайтах с большой нагрузкой он используется, как front-end к Apache, т.к. оптимизирован для раздачи статического контента большому количеству клиентов при незначительном потреблении системных ресурсов.

Если среди всех сайтов у nginx 3 место, то в сегменте популярных сайтов с большой нагрузкой он обходит IIS почти в два раза, имея 31,6% рынка и отставая от Apache всего на 5% (по данным w3techs.com)

## Связанные технологии

Мы разобрали основные технологии, используемые в том или ином варианте на большинстве веб-сайтов, но существует также много других технологий и программ, к использованию которых прибегают реже, в случае необходимости и в основном в крупных проектах. Такие программы обычно решают проблемы оптимизации, межсерверного взаимодействия, упрощают работу программистов или решают другие специфические проблемы.

Подобных программ и технологий существует масса, в рамках этой работы я расскажу только про несколько из них.

#### Кэширование

В общем случае кэширование представляет собой механизм сохранения информации в буфер с быстрым доступом и извлечения её оттуда.

В веб-разработках кэширование обычно применяется для частых запросов к базам данных. Это делается для уменьшения нагрузки на них и ускорения доступа к данным. Многие СУБД имеют собственные механизмы кэширования, их использование позволяет снизить время отклика БД.

Другое применение кэширования – кэширование результатов работы скриптов (в т.ч. и скриптов, работающих с БД). Оно имеет смысл при частом использовании тяжеловесных скриптов, т.к. позволяет значительно снизить нагрузку на процессор и время ответа (но повысит расход памяти). Самые популярные решения для этой задачи – сервера кэширования **memcached** и **Varnish**.

#### Межсерверное взаимодействие

В проектах с высокой нагрузкой одному, даже самому мощному, серверу с нагрузкой не справиться. В подобных проектах обычно создаётся целая серверная инфраструктура, в которой обязанности чаще всего разделены между серверами или группами серверов. Сложность тут в организации взаимодействия между компонентами системы и в постоянном отслеживании состояния узлов и системы в целом.

Для решения этих проблем также существует специальные решения, например **syslog -** стандарт отправки и регистрации сообщений о происходящих в системе событиях для компьютерных сетей, работающих по протоколу ip и одноимённое программное обеспечение (сервер логирования). Также для этих же целей часто используется **Scribe** - сервер логирования разработки facebook.

Кроме того, стоит привести здесь информацию о Messages queue. MQ - это архитектура и ПО промежуточного уровня, которое занимается сбором, хранение и маршрутизацией (распределением) сообщений между компонентами в асинхронном режиме, причём в качестве компонентов тут могут выступать как целые серверы, так и отдельные компоненты веб-приложений. Среди наиболее распространённых проектов MQ-серверов можно назвать **Apache ActiveMQ** (используемый, например, на Большом Адронном Коллайдере) и **WebSphere** от IBM.

## В завершение обзорной части

В первой части я постарался рассмотреть только наиболее значимые для веб-разработки технологии. В процессе поиска материала я нашёл большое количество различных интересных программ, фреймворков, технологий, каждая из которых достойна упоминания, но не стал писать о них потому, что в ином случае работа грозила бы неприлично разрастись, ведь сегодня новые разработки появляются постоянно и выбор постоянно растёт.

Также хочу отметить, что во многих крупных проектах предпочтение редко отдаётся какой-либо одной технологии каждого класса. Каждой технологии, каждому языку программирования присущи свои особенности, делающие их более или менее пригодными для разных типов задач, поэтому в крупном проекте из соображений оптимизации за разные данные могут отвечать разные СУБД, разные модули могут быть написаны на разных языках программирования, разные HTTP-сервера дополняют друг друга, деля запросы между собой с учётом своих архитектурных особенностей.

# Проектирование

## Выбор связки технологий для написания приложения

### Веб-технологии

**Веб-сервер**

В качестве веб-сервера будет использоваться сервер nginx последней версии ввиду его бесплатности, легковесности и достаточности для поставленных задач.

**Языки программирования**

Языком программирования серверной части выбран PHP. Он прост в изучении, к тому же у меня уже есть определённый опыт его использования.

**Языки разметки**

Для разметки страницы буду использовать стандартный набор HTML+CSS

**Сервер базы данных**

В качестве сервера базы данных будет использоваться MySQL. Большой нагрузки на систему не предполагается, задача вполне стандартна, поэтому разумно использовать наиболее распространённую среди небольших проектов, бесплатную и хорошо документированную БД с большим сообществом разработчиков.

### Программное обеспечение

**Операционная система**

Вся работа выполняется на ПК под управлением Windows 7 Professional x64

**Текстовый редактор**

Для написания кода используется текстовый редактор Notepad++ v6.3.2

**Браузер**

Веб-приложение тестируется в браузерах IE 9.0, IE 7.0, Firefox 20.01, Opera 12.15 и Chrome 26.0

## Структура приложения

#### Архитектура

Архитектура веб-приложения стандартна для небольших веб-приложений (рис. 3.2.1.1)

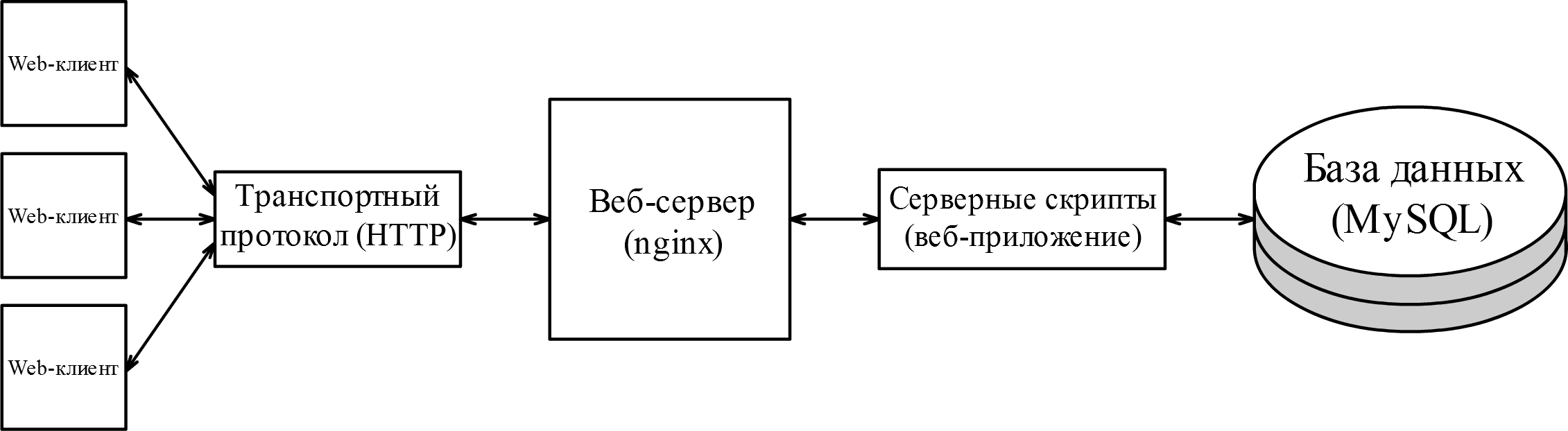


Рис. 3.2.1.1 – Архитектура

Клиенты по протоколу HTTP присылают запросы веб-серверу, тот выполняет соответствующие скрипты веб-приложения и отдаёт по протоколу HTTP результирующие страницы клииентам. Веб-приложение имеет связь с сервером БД, откуда во время выполнения скриптов при необходимости извлекает данные.

#### Структура базы данных

Структура базы отражена в рисунке 3.2.2.1.

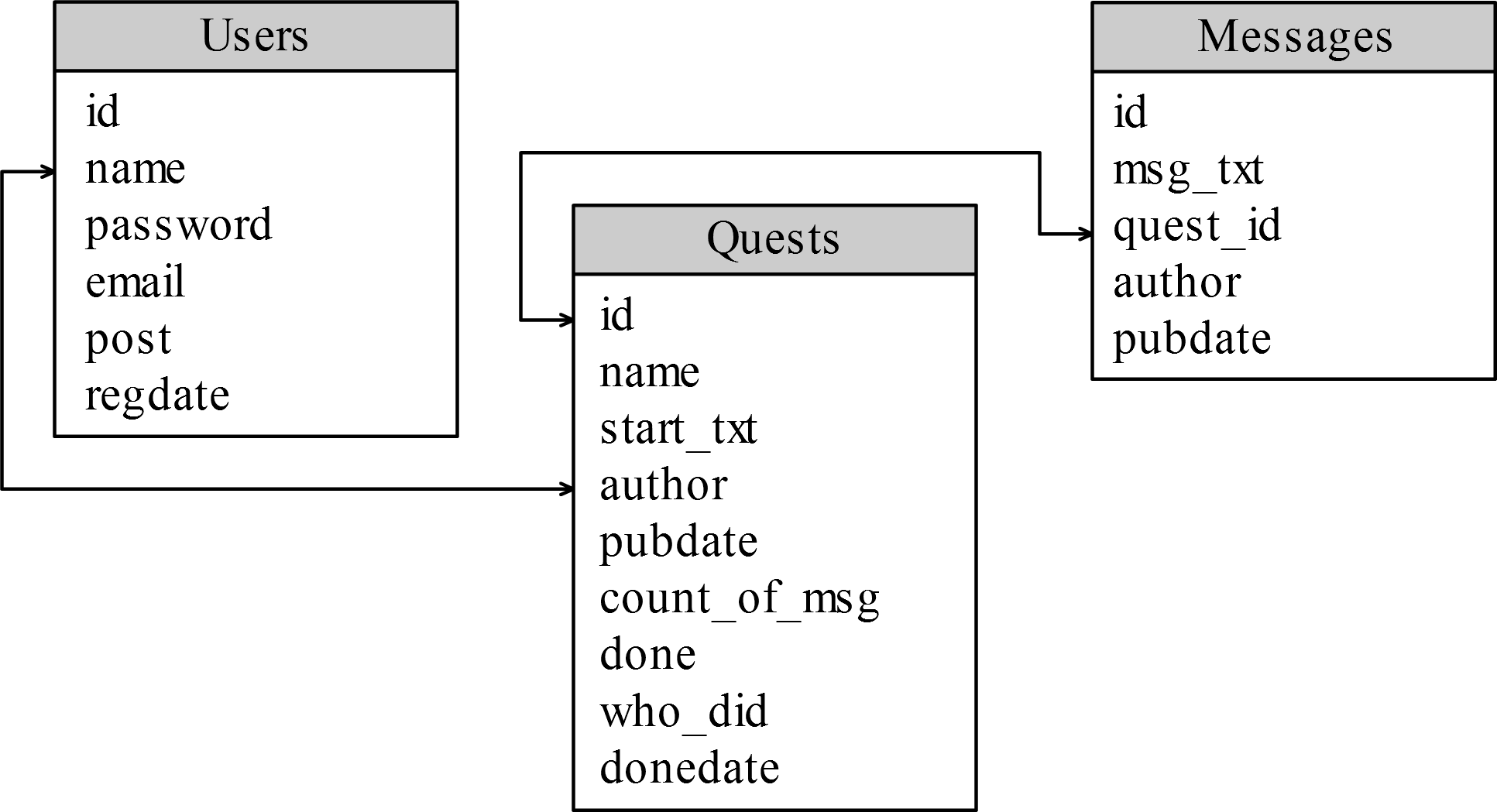


Рис. 3.2.2.1 – структура БД

В качестве данных в веб-приложении будут храниться информация о пользователях, заданиях и комментариях к ним. В базе данных созданы 3 соответствующие таблицы:

1. Users – таблица учётных записей пользователей. Поля таблицы:

* Id – номер записи
* Name – имя пользователя
* Password – хэш пароля
* Email – e-mail пользователя
* Post – должность
* Regdate – дата регистрации

1. Quests – таблица заданий и информации о них. Поля таблицы:

* Id – номер записи
* Name - заголовок
* Start\_txt – текст задания
* Author - автор
* Pubdate – дата публикации
* Count\_of\_msg – кол-во комментариев
* Done – флаг выполнения
* Who\_did – выполнивший
* Donedate – дата выполнения

1. Messages – таблица комментариев. Поля таблицы:

* Id – номер записи
* Msg\_txt – текст комментария
* Quest\_id – номер задания, к которому принадлежит комментарий
* Author – автор комментария
* Pubdate – дата написания комментария

#### Каркас веб-приложения

Структура каркаса отображена на рис. 3.2.3.1.

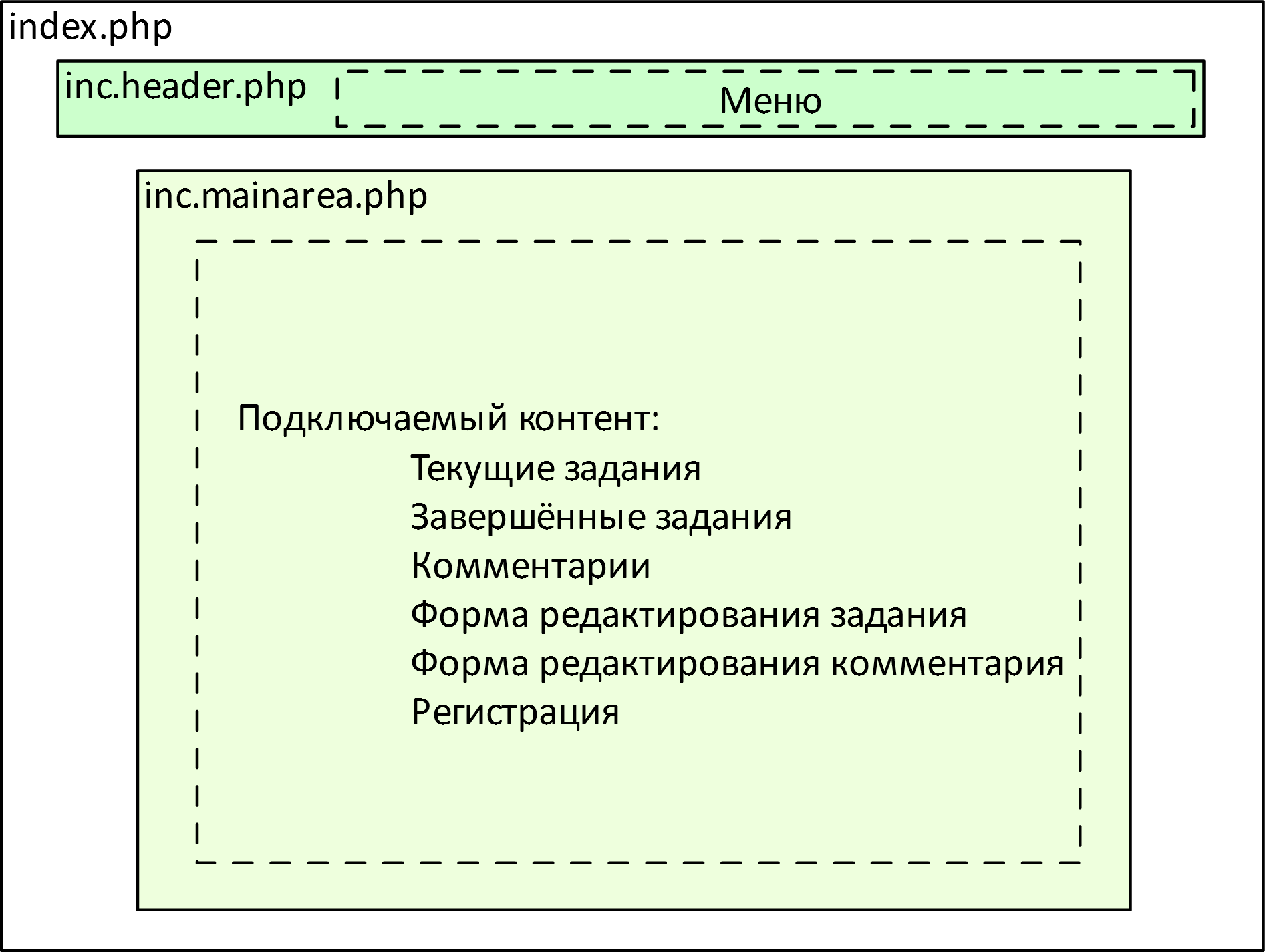


Рис. 3.2.3.1 – структура каркаса веб-приложения

В головной файл index.php подключаются файл заголовка inc.header.php, содержащий навигационные ссылки и блок авторизации, и файл основной области inc.mainarea.php, куда в зависимости от параметров URL подгружается те или иные данные или формы.

#### Класс для работы с БД

За всю работу с базой данных отвечает класс QuestsDB, имеющий следующие публичные методы:

login($name, $password) - авторизация

addUser($name,$password,$email,$post) - добавляет нового пользователя

getQuestCount($which) - возвращает кол-во заданий указанного типа

getQuests($which,$page=0, $id=0) - возвращает массив заданий указанного типа либо задание с указанным id

addQuest($name,$text,$author) - добавляет новое задание в базу

editQuest($questId,$newName,$newTxt,$author) - редактирует указанное задание

deleteQuest($questID) - удаляет указанное задание

markDone($questId,$username, $undone=0) - помечает указанное задание сделанным/несделанным

getComments($id,$commentId=0) - возвращает массив комментариев к указанному заданию

addComment($text, $questId, $author) - добавляет новый комментарий

editComment($commentId,$newTxt,$author) - редактирует указанный комментарий

deleteComment($msgId,$author) - удаляет указанный комментарий

#### Страницы

Различные страницы веб-приложения подключаются в главную область в зависимости от параметра id GET-запроса. Всего приложение имеет 6 страниц:

* Текущие задания (выводится по умолчанию) - quests.php
* Выполненные задания (id = done) - done.php
* Комментарии к заданию (id = comments) - comments.php
* Редактирование задания (id = editQuest) - editQuest.php
* Редактирование комментария (id = editComment) - editComment.php
* Регистрация нового пользователя (id = register) - register.php

В случае, когда значение параметра id отлично от указанных выше, подключается страница по умолчанию.

#### Система авторизации и привилегии пользователей

В веб-приложении существует система авторизации, позволяющая ограничить доступ пользователей к содержимому. Существуют следующие уровни доступа:

**Неавторизованный пользователь** может просматривать текущие и завершённые задания, комментарии к ним, а также может зарегистрироваться и авторизоваться.

**Авторизованный пользователь** может просматривать текущие и завершённые задания, комментарии к ним, добавлять новые задания и комментарии, помечать задания выполненными/невыполненными, а также редактировать и удалять комментарии, оставленные им.

#### Защита от взлома

Пароли пользователей хранятся в БД в захэшированном с использованием уникальной для каждого пользователя соли виде.

Все переменные, задаваемые пользователем и используемые в запросах к базе данных, экранируются, если должны иметь строковый тип, либо явно приводятся к требуемому числовому типу.

Все параметры GET-запросов проверяются. Заведомо неверные игнорируются, допустимые проверяются на доступность пользователю.

# Результат работы

Результатом работы является веб-приложение Quests, представленное на скриншотах ниже (в целях экономии тонера принтера задний тёмный фон был убран).

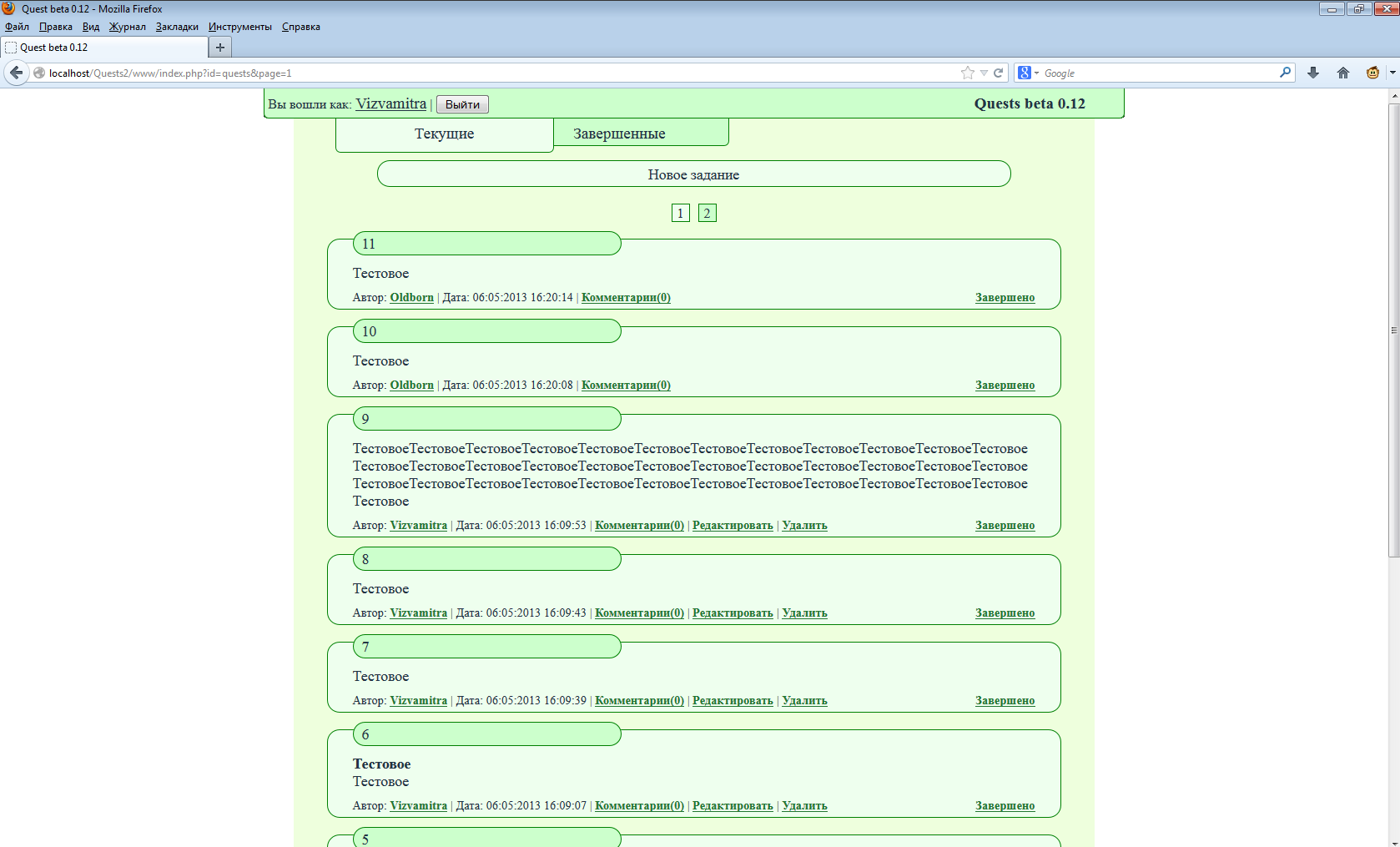


Рис. 4.1 – Текущие задания

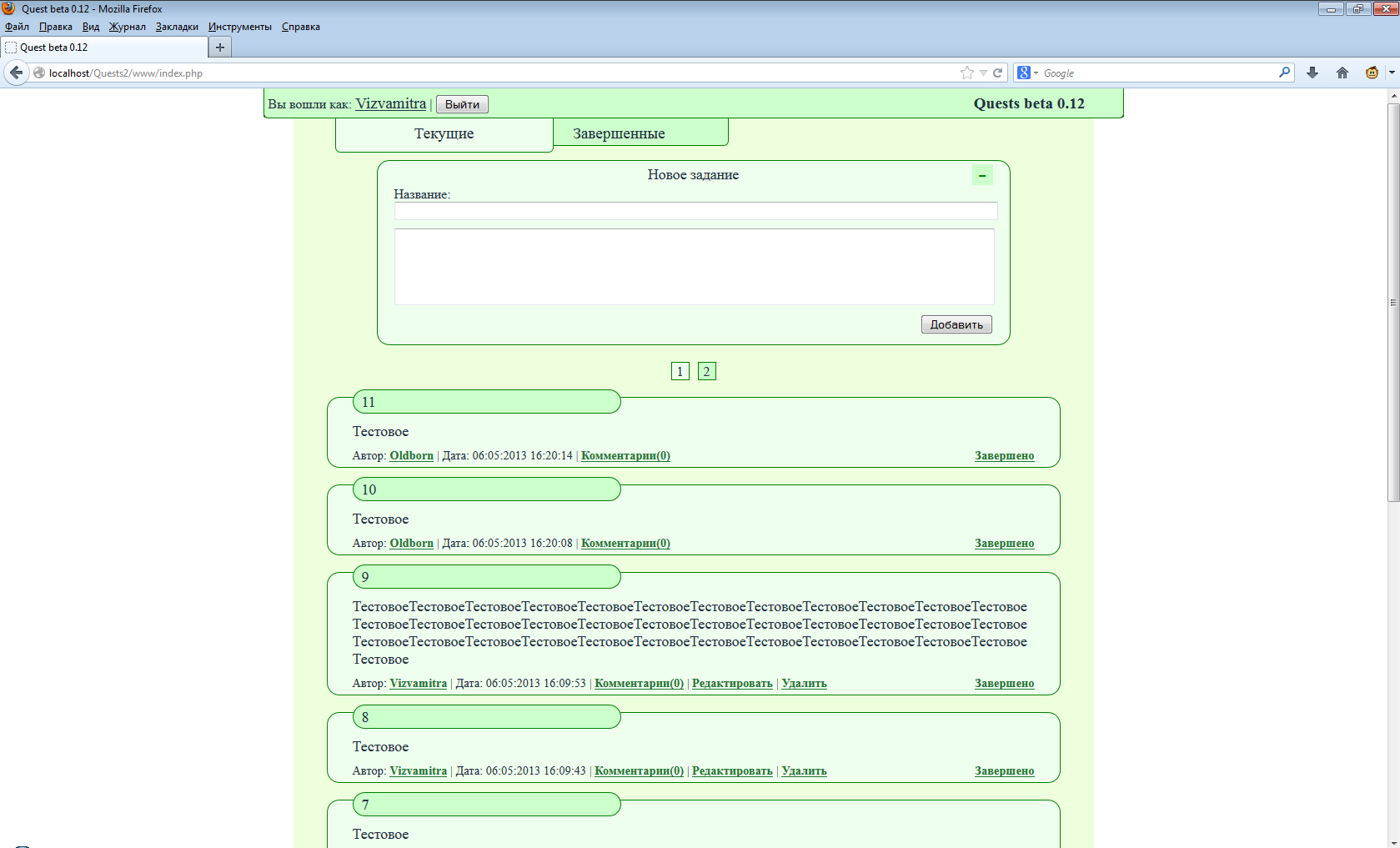


Рис. 4.2 – Добавление нового задания

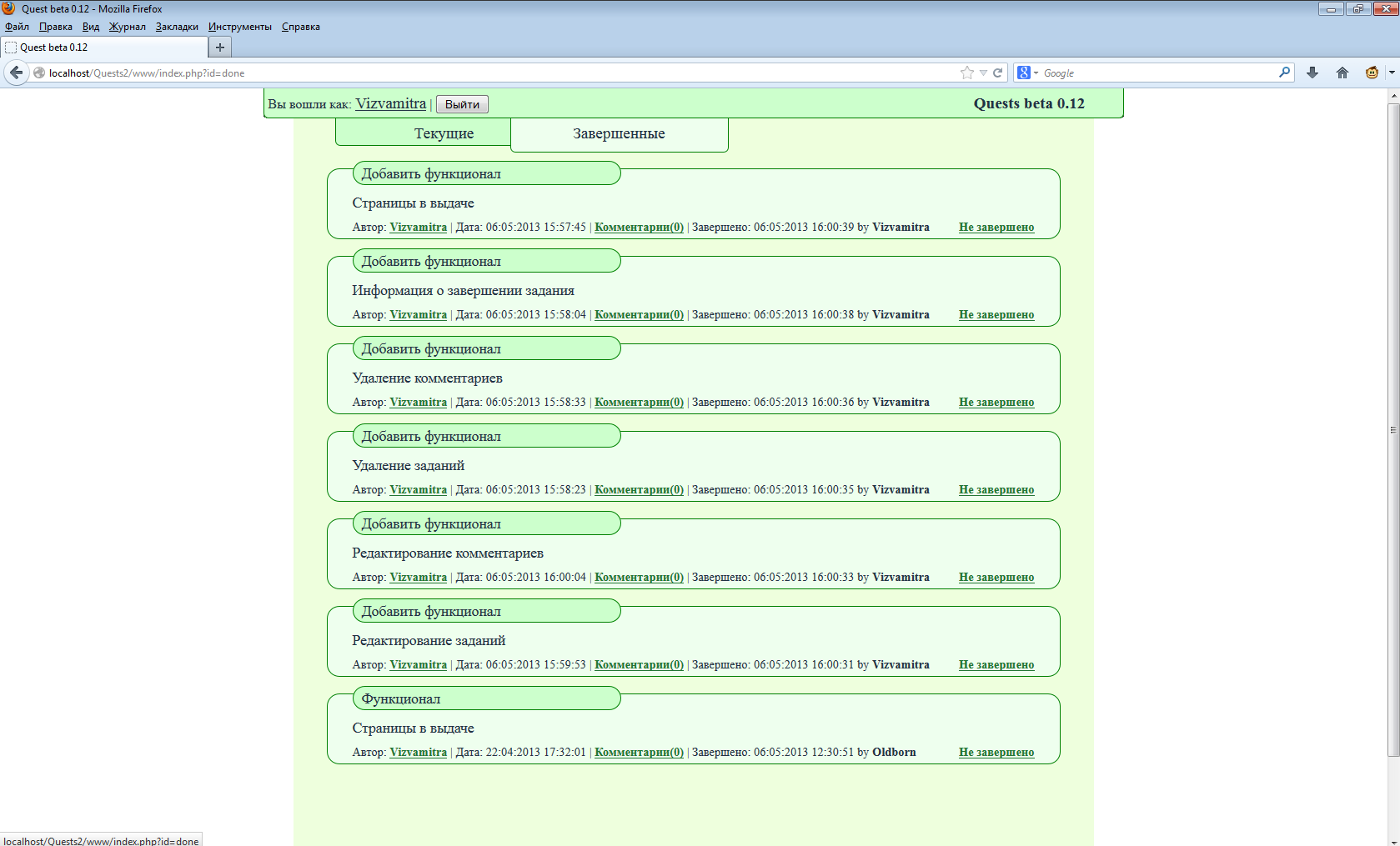


Рис. 4.3 – Завершенные задания

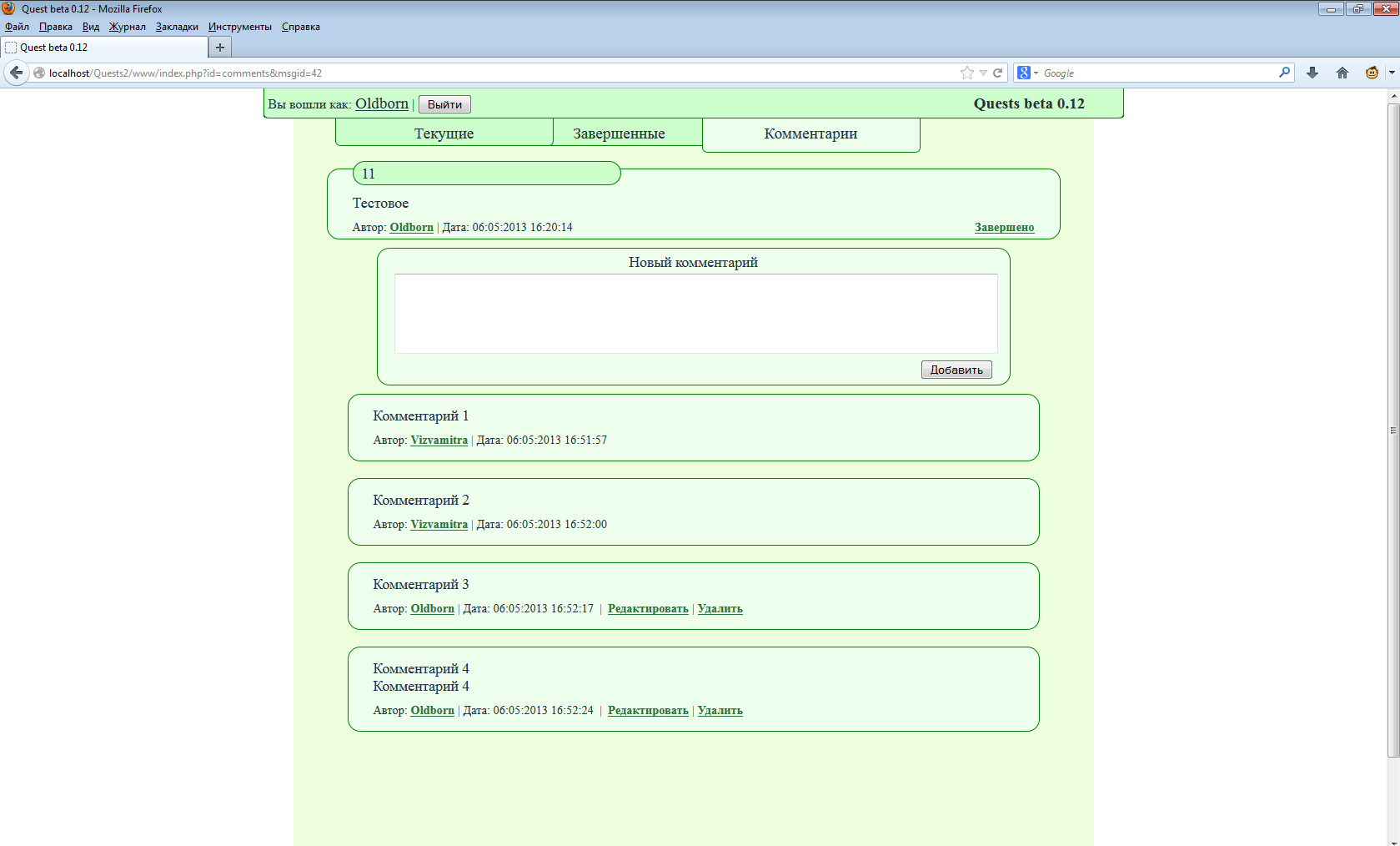


Рис. 4.4 – Комментарии

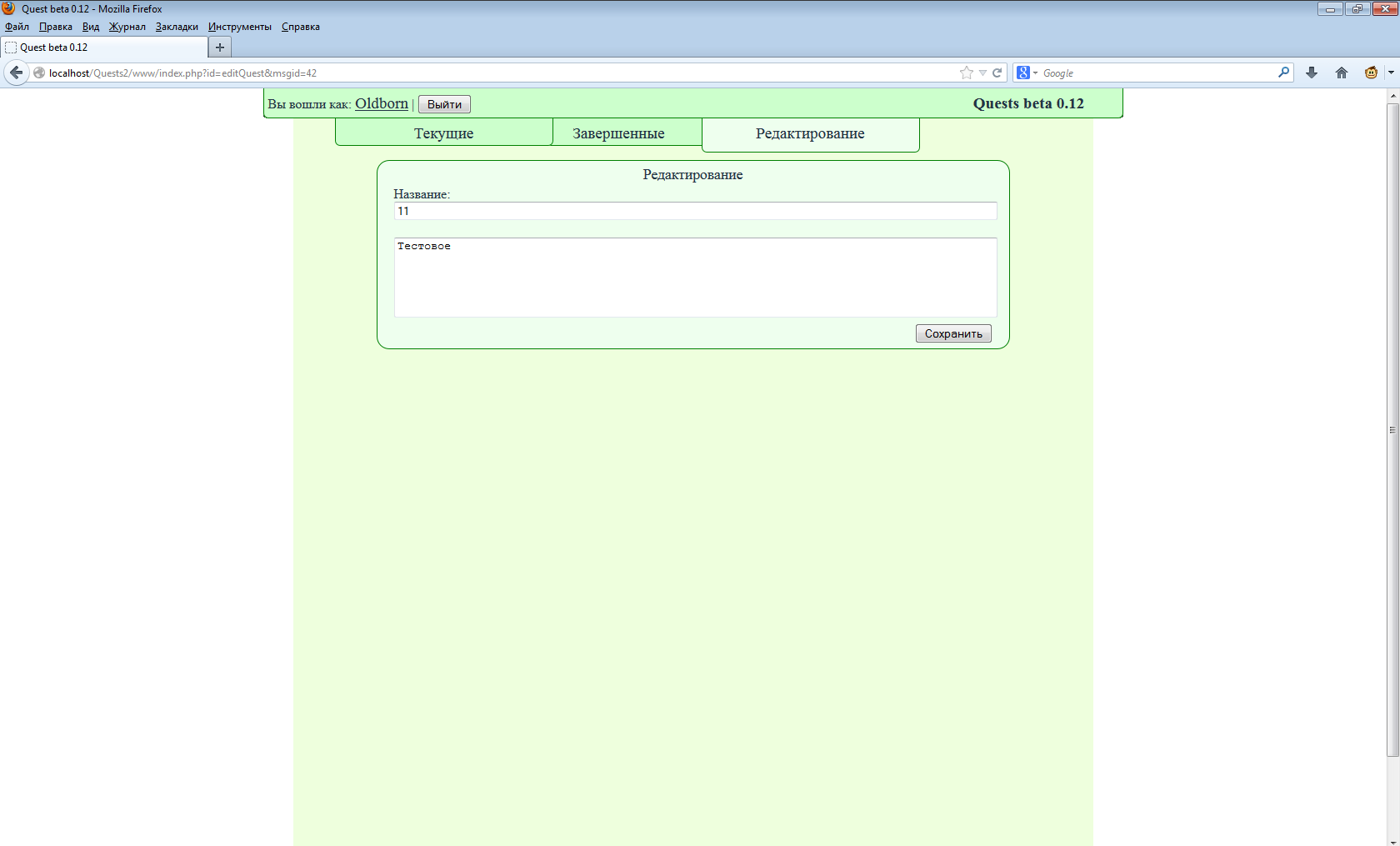


Рис. 4.5 – Редактирование задания

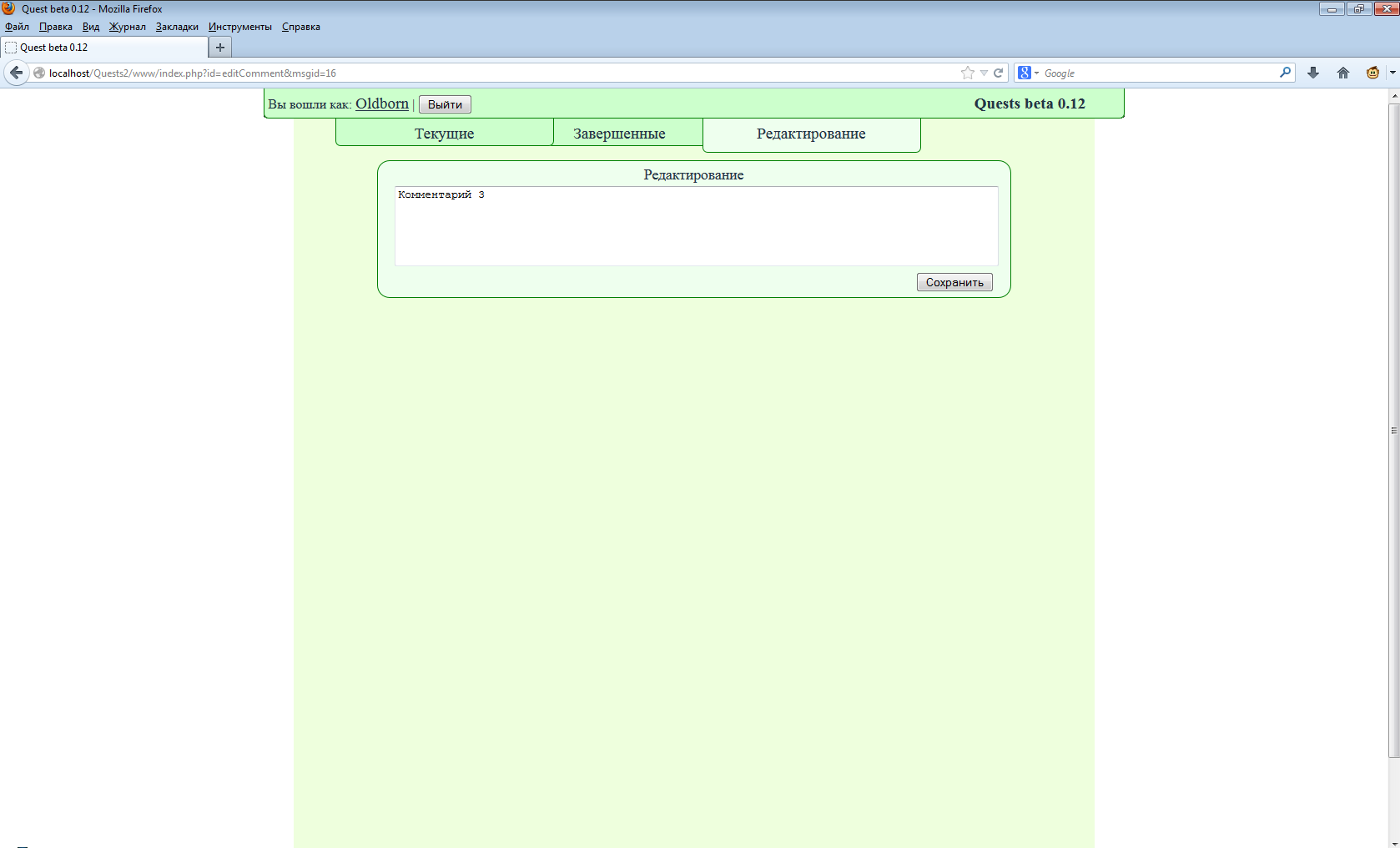


Рис. 4.6 – Редактирование комментария

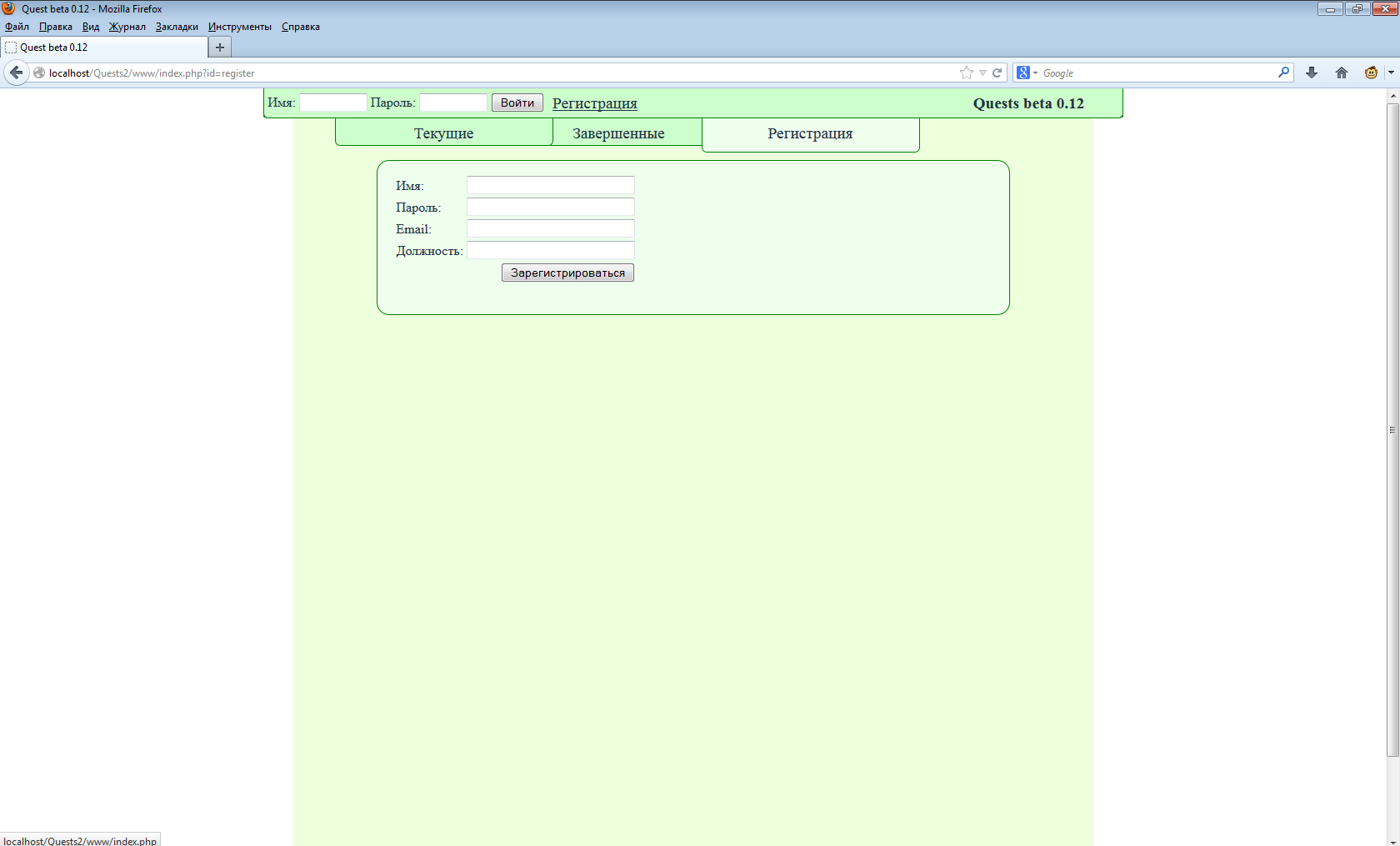


Рис. 4.7 – Регистрация

# Заключение

В процессе прохождения практики мною были рассмотрены существующие технологии веб-программирования и на основе связки PHP+MySQL разработана серверная часть веб-приложения – планировщика задач для использования внутри отдела.

# Приложение

#### index.php

<?php

**session\_start**();

**if** (!$\_SESSION[**'auth\_type'**])

$\_SESSION[**'auth\_type'**]=**"anonimous"**;

**require\_once**(**"class.QuestsDB.php"**);

$db = QuestsDB::getInstance();

?>

<!DOCTYPE HTML>

<HTML>

<HEAD>

<META CHARSET=**"utf-8"**/>

<LINK rel=**"stylesheet"** type=**"text/css"** href=**"css/styles.css"**/>

<SCRIPT type=**"text/javascript"** src=**"js/jquery-1.9.1.min.js"**></SCRIPT>

<SCRIPT type=**"text/javascript"** src=**"js/scripts.js"**></SCRIPT>

<TITLE>**Quest beta 0.12**</TITLE>

</HEAD>

<BODY>

<DIV id=**"header"**>

<?php

**include**(**"inc.header.php"**);

?>

</DIV>

<DIV id=**"main"**>

<?php

**include**(**"inc.mainarea.php"**);

?>

</DIV>

</BODY>

</HTML>

#### inc.header.php

<DIV id=**"login"**>

<?php

**switch**($\_SESSION[**'auth\_type'**]){

**case** **'anonimous'**:?><FORM id=**'loginForm'** method=**'POST'** action=**'login.php'">**

**<SPAN>Имя:</SPAN>**

**<INPUT id="**enteredLogin" type=**'text'** name=**'name'** size=**'10'** maxlength=**'30'**>

<SPAN>**Пароль:**</SPAN>

<INPUT id=**'enteredPassword'** type=**'password'** name=**'password'** size=**"10"** maxlength=**'30'**>

<INPUT type=**'submit'** value=**'Войти'**>

<INPUT type=**'hidden'** name=**'id'** value=**'**<?=$\_GET[**'id'**]?>**'**>

<INPUT type=**'hidden'** name=**'msgid'** value=**'**<?=$\_GET[**'msgid'**]?>**'**>

</FORM>

<A href=**'index.php?id=register'**>**Регистрация**</A><?

**break**;

**case** **'user'**:?>

<FORM method=**'POST'** action=**'logout.php'**>

<SPAN>**Вы вошли как:** <A href=**'index.php?id=profile&userid=**<?=$\_SESSION[**'userID'**]?>**'**><?=$\_SESSION[**'userName'**]?></A> **|** </SPAN>

<INPUT type=**'submit'** value=**'Выйти'**>

<INPUT type=**'hidden'** name=**'id'** value=**'**<?=$\_GET[**'id'**]?>**'**>

<INPUT type=**'hidden'** name=**'msgid'** value=**'**<?=$\_GET[**'msgid'**]?>**'**>

</FORM><?

**break**;

}

?>

</DIV>

<SPAN id=**"title"**>**Quests beta 0.12**</SPAN>

<DIV id=**'tabs'**>

<A id=**'questsTab'** href=**'index.php'**>**Текущие**</A>

<A id=**'doneTab'** href=**'index.php?id=done'**>**Завершенные**</A>

<A id=**'commentsTab'** href=**'#'**>**Комментарии**</A>

<A id=**'editTab'** href=**'#'**>**Редактирование**</A>

<A id=**'registerTab'** href=**'#'**>**Регистрация**</A>

</DIV>

#### inc.mainarea.php

<?php

$id = **strip\_tags**($\_GET[**'id'**]);

**if** ($id!=**'register'**){

$\_SESSION[**'nameError'**]=**''**;

$\_SESSION[**'passwordError'**]=**''**;

$\_SESSION[**'emailError'**]=**''**;

}

**switch** ($id){

**case** **'done'**: **include** **"done.php"**;**break**;

**case** **'register'**: **include** **"register.php"**;**break**;

**case** **'profile'**: **include** **"profile.php"**;**break**;

**case** **'comments'**: **include** **"comments.php"**;**break**;

**case** **'editQuest'**: **include** **"editQuest.php"**;**break**;

**case** **'editComment'**: **include** **"editComment.php"**;**break**;

**default**: **include** **"quests.php"**;

}

?>

#### styles.css

body **{**

**margin:0px;**

**width:100%;**

**background-color:#121;**

**color:#234;**

**}**

a**:link,** a**:visited{**

**color:#234;**

**}**

H1**{**

**margin:0px;**

**}**

**.**error**{**

**color:red;**

**}**

pre**{**

**font:inherit;**

**padding:0;**

**margin:0;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Заголовок \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#header{**

**width:1020px;**

**margin:0px auto;**

**background-color:#cfc;**

**padding:5px 5px 0px 5px;**

**height:30px;**

**border:1px solid green;**

**border-top:0px;**

**position:relative;**

**z-index:3;**

border-radius**:0 0 6px 6px;**

-webkit-border-radius**:0 0 6px 6px;**

-moz-border-radius**:0 0 6px 6px;**

-khtml-border-radius**:0 0 6px 6px;**

**}**

**#header** a**{**

**font-size:1.2em;**

**}**

**#login{**

**float:left;**

**position:absolute;**

**}**

**#header** **#title{**

**display:block;**

**float:right;**

**padding-right:40px;**

**font-size:1.2em;**

**font-weight:bold;**

**text-align:center;**

**}**

**#loginForm{**

**float:left;**

**margin-right:10px;**

**}**

**#tabs{**

**height:41px;**

**text-align:center;**

**}**

**#questsTab,** **#doneTab,** **#commentsTab,** **#editTab,** **#registerTab{**

**position:relative;**

**border:1px solid green;**

**border-top:0px none;**

**padding:5px;**

**text-decoration:none;**

**font-size:20px;**

border-radius**:0 0 6px 6px;**

**width:250px;**

**}**

**#questsTab{**

**display:block;**

**left:60px;**

**height:30px;**

**margin:31px 20px 0 20px;**

**background-color:#efe;**

**z-index:2;**

**}**

**#doneTab{**

**display:block;**

**top:-72px;**

**left:330px;**

**height:22px;**

**margin:31px 20px 0 -40px;**

**background-color:#cfc;**

**z-index:1;**

**}**

**#commentsTab,** **#editTab,** **#registerTab{**

**display:none;**

**top:-135px;**

**left:620px;**

**margin:31px 20px 0 -100px;**

**background-color:#cfc;**

**z-index:3;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Главная часть \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Окно информации \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#main{**

**padding:40px 0 20px 0;**

**width:960px;**

**margin:0px auto 20px auto;**

**background-color:#efd;**

**min-height:900px;**

border-radius**:0 0 6px 6px;**

-webkit-border-radius**:0 0 6px 6px;**

-moz-border-radius**:0 0 5px 5px;**

-khtml-border-radius**:0 0 10px 10px;**

**}**

**#pages{**

**margin: 20px 40px 0 40px;**

**text-align:center;**

**}**

**.**pageNum**{**

**display:inline-block;**

**width:20px;**

**height:20px;**

**background-color:#cfc;**

**border:1px solid green;**

**text-align:center;**

**margin: 0 5px;**

**text-decoration:none;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задачи \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**.**questDIV**{**

**margin:20px 40px 0px 40px;**

**padding:15px 30px 5px 30px;**

**background-color:#efe;**

**border:1px solid green;**

border-radius**:15px;**

-webkit-border-radius**:15px;**

-moz-border-radius**:15px;**

-khtml-border-radius**:15px;**

**}**

**.**questTitle**{**

**width:300px;**

**background-color:#cfc;**

**border:1px solid green;**

**margin:-25px 0 0 0;**

border-radius**:25px;**

**padding:3px 10px 3px 10px;**

**font-size:1.1em;**

**white-space:pre-wrap;**

**}**

**.**questText**{**

**white-space:pre-wrap;**

word-wrap**:break-word;**

**font-size:1.1em;**

**text-align:justify;**

**margin-top:10px;**

**}**

**.**markDone**{**

**text-align:right;**

**margin-top:10px;**

**}**

**.**markDone a**{**

**float:right;**

**font-weight:bold;**

**font-size:0.9em;**

**color:#273;**

**}**

**.**questInfo**{**

**margin-top:10px;**

**font-size:0.9em;**

**}**

**.**questInfo a**:link,** **.**questInfo a**:visited,** **.**controls a**:link,** **.**controls a**:visited{**

**color:#273;**

**font-weight:bold;**

**font-size:inherit;**

**}**

**.**controls**{**

**margin-top:10px;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Комментарии \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**.**comment**{**

**position:relative;**

**counter-increment:index;**

**margin:10px 65px 20px 65px;**

**padding:5px 30px 15px 30px;**

**background-color:#efe;**

**border:1px solid green;**

border-radius**:15px;**

-webkit-border-radius**:15px;**

-moz-border-radius**:15px;**

-khtml-border-radius**:15px;**

**}**

**.**control**{**

**text-align:right;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* формы \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#registerForm{**

**margin:10px 100px 10px 100px;**

**padding:15px 20px 35px 20px;**

**background-color:#efe;**

**border:1px solid green;**

border-radius**:15px;**

-webkit-border-radius**:15px;**

-moz-border-radius**:15px;**

-khtml-border-radius**:15px;**

**}**

**#registerForm** input**[**type='submit'**]{**

**float:right;**

**}**

**.**form**{**

**margin:10px 100px 10px 100px;**

**padding:5px 20px 35px 20px;**

**background-color:#efe;**

**border:1px solid green;**

border-radius**:15px;**

-webkit-border-radius**:15px;**

-moz-border-radius**:15px;**

-khtml-border-radius**:15px;**

**}**

**.**form textarea**{**

resize**:none;**

**width:100%;**

**height:90px;**

**}**

**.**form input**[**type="submit"**]{**

**float:right;**

**margin-top:5px;**

**margin-bottom:5px;**

**}**

**.**form input**[**type='text'**]{**

**width:100%;**

**}**

**.**form span**:first-child{**

**display:block;**

**text-align:center;**

**margin-bottom:3px;**

**font-size:1.1em;**

**}**

div**.**form**{**

**padding:15px 20px 15px 20px;**

**text-align:center;**

**}**

**#newQuest{**

**height:0px;**

**padding:5px 20px 25px 20px;**

**overflow:hidden;**

**cursor:pointer;**

**}**

**#newQuest** input **[**type="text"**]{**

**margin-bottom:20px;**

**width:716px;**

**}**

**#newQuest** textarea**{**

**margin-top:10px;**

**margin-bottom:5px;**

**padding:0;**

**}**

**#minus{**

**display:none;**

**height:25px;**

**width:25px;**

**font-size:1.8em;**

**color:green;**

**text-align:center;**

**line-height:0.7;**

**font-weight:900;**

**float:right;**

**margin:-25px 0 0 0;**

**background-color:#cfc;**

**cursor:pointer;**

**}**

#### scripts.js

$**(**document**).**ready**(*function*(){**

$**("#newQuest").**click**(**openForm**);**

$**("#minus").**click**(**closeForm**);**

**})**

***function*** openForm**(){**

$**("#newQuest").**css**({"cursor":"auto"}).**animate**({"height":"190px"},** 400**);**

$**("#minus").**css**({"display":"block"});**

**}**

***function*** closeForm**(){**

$**("#newQuest").**css**({"cursor":"pointer"}).**animate**({"height":"0px"},** 400**);**

$**("#minus").**css**({"display":"none"});**

***return*** ***false*;**

**}**

#### createDB.php

<?php

// Создание структуры Базы Данных гостевой книги

**define**(**"DB\_HOST"**, **"localhost"**);

**define**(**"DB\_LOGIN"**, **"oldborn"**);

**define**(**"DB\_PASSWORD"**, **"00060029"**);

**define**(**"DB\_NAME"**, **"oldborn\_quests"**);

**mysql\_connect**(DB\_HOST, DB\_LOGIN, DB\_PASSWORD) **or** **die**(**mysql\_error**());

**mysql\_query**(**"CREATE DATABASE "**.DB\_NAME);

**mysql\_select\_db**(DB\_NAME) **or** **die**(**mysql\_error**());

$sql = **"**

**CREATE TABLE quests (**

**id int(11) NOT NULL auto\_increment,**

**name varchar(33) NOT NULL default '',**

**start\_txt text,**

**author varchar(30),**

**pubdate int(11) NOT NULL default 0,**

**count\_of\_msg int(11) NOT NULL default 0,**

**done int(1) NOT NULL default 0,**

**who\_did varchar(30),**

**donedate int(11) NOT NULL default 0,**

**PRIMARY KEY (id)**

**)"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$sql = **"**

**CREATE TABLE messages (**

**id int(11) NOT NULL auto\_increment,**

**msg\_txt text,**

**quest\_id int(11),**

**author varchar(30),**

**pubdate int(11) NOT NULL default 0,**

**PRIMARY KEY (id)**

**)"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$sql = **"**

**CREATE TABLE users (**

**id int(11) NOT NULL auto\_increment,**

**name varchar(30) NOT NULL default '',**

**password varchar(40) NOT NULL default '',**

**email text,**

**post text,**

**regdate int(11) NOT NULL default 0,**

**PRIMARY KEY (id)**

**)"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

**mysql\_close**();

**print** **'<p>Структура базы данных успешно создана!</p>'**;

?>

#### class.IQuestsDB.php

<?php

/\*\*

\* IQuestsDB

\* содержит основные методы для работы с Базой данных

\*/

**interface** IQuestsDB {

**function** login($name, $password);

**function** addUser($name,$password,$email,$post);

**function** getQuestCount($which);

**function** getQuests($which,$page=0, $id=0);

**function** addQuest($name,$text,$author);

**function** editQuest($questId,$newName,$newTxt,$author);

**function** deleteQuest($questID);

**function** markDone($questId,$username, $undone=0);

**function** getComments($id,$commentId=0);

**function** addComment($text, $questId, $author);

**function** editComment($commentId,$newTxt,$author);

**function** deleteComment($msgId,$author);

}

?>

#### class.QuestsDB.php

<?php

**require**(**"class.IQuestsDB.php"**);

**class** QuestsDB **implements** IQuestsDB {

**const** DB\_HOST = **"localhost"**;

**const** DB\_LOGIN = **"oldborn"**;

**const** DB\_PASSWORD = **"00060029"**;

**const** DB\_NAME = **"oldborn\_quests"**;

**private** **static** $\_instance;

**private** $\_conn;

**private** **function** \_\_construct(){}

**private** **function** \_\_clone(){}

**private** **function** **\_\_wakeup**(){}

**public** **static** **function** getInstance(){

**if** (self::$\_instance === **null**) {

self::$\_instance = **new** self;

}

**return** self::$\_instance;

}

**private** **function** clearData($data, $type=**"s"**){

**switch**($type){

**case** **"s"**: **return** **"'"**.**mysql\_real\_escape\_string**(trim(**strip\_tags**($data))).**"'"**;

**case** **"i"**: **return** intval($data);

**case** **"ui"**: $data = intval($data);

**if** ($data<0) $data=-$data;

**return** $data;

**case** **"sf"**: **return** **"'"**.trim(**strip\_tags**($data)).**"'"**;

**case** **"stext"**:**return** **"'"**.trim(**strip\_tags**($data, **'<b><a><u><pre><i><tt><s>'**)).**"'"**;

}

}

**private** **function** db2Array($data){

$arr = **array**();

**while** ($row = **mysql\_fetch\_assoc**($data))

$arr[] = $row;

**return** $arr;

}

**private** **function** connectDb(){

$\_conn = **mysql\_connect**(self::DB\_HOST,self::DB\_LOGIN,self::DB\_PASSWORD);

**mysql\_select\_db**(self::DB\_NAME);

$sql = **"SET NAMES utf8"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$sql = **"SET CHARACTER SET utf8"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

**return** $\_conn;

}

**function** login($name, $password){

$\_conn = $this->connectDb();

$name = $this->clearData($name);

$password = **sha1**( **md5**( **md5**($password . **sha1**($name)) ) );

$sql = **"SELECT id, name**

**FROM users**

**WHERE name = $name**

**AND password = '$password'**

**"**;

$result = **mysql\_query**($sql) **or** **die**(**mysql\_error**());

**mysql\_close**($\_conn);

$user = **array**();

$user = **mysql\_fetch\_assoc**($result);

**return** $user;

}

**function** addUser($name,$password,$email,$post){

$i = 0;

$\_conn = $this->connectDb();

$name = $this->clearData($name);

$email = $this->clearData($email);

$post = $this->clearData($post);

$password = **"'"**.**sha1**( **md5**( **md5**($password . **sha1**($name)) ) ).**"'"**;

$sql = **"SELECT name, email FROM users**

**WHERE name = $name**

**OR email = $email"**;

$result=**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$result=$this->db2array($result);

**foreach**($result **as** $user){

**if** (**"'"**.**strtoupper**($user[**'name'**]).**"'"**==**strtoupper**($name))

$i+=10;

**if** (**"'"**.**strtoupper**($user[**'email'**]).**"'"**==**strtoupper**($email))

$i+=1;

}

**if** ($i!=0){

**mysql\_close**($\_conn);

**return** $i;

}

**else**{

$sql = **"INSERT INTO users (name, password, email, post, regdate)**

**VALUES ($name,$password,$email,$post,"**.**time**().**")"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$sql = **"SELECT id, name FROM users WHERE name = $name"**;

$result = **mysql\_query**($sql) **or** **die**(**mysql\_error**());

$result = **mysql\_fetch\_assoc**($result);

}

**mysql\_close**($\_conn);

**return** $result;

}

**function** getQuestCount($which){

$\_conn = $this->connectDb();

$sql = **"SELECT count(\*) FROM quests"**;

**switch**($which){

**case** **"actual"**: $sql .= **" WHERE done=0"**; **break**;

**case** **"done"**: $sql .= **" WHERE done=1"**; **break**;

**case** **"all"**: **break**;

**default**: **die**(**"Error useing getQuestCount: wrong argument.\n"**);

}

$result = **mysql\_query**($sql) **or** **die**(**mysql\_error**());

**mysql\_close**($\_conn);

$result = $this->db2array($result);

**return** intval($result[0][**'count(\*)'**]);

}

**function** getQuests($which, $page=0, $id=0){

$\_conn = $this->connectDb();

$id = $this->clearData($id,**'i'**);

$page = $this->clearData($page,**'ui'**);

$sql = **"SELECT id, name, start\_txt, author, pubdate, count\_of\_msg, done, donedate, who\_did FROM quests"**;

**if** ($id)

$sql .= **" WHERE id = $id"**;

**else** {

**switch**($which){

**case** **"actual"**: $sql .= **" WHERE done = 0"**; **break**;

**case** **"done"**: $sql .= **" WHERE done = 1"**; **break**;

**case** **"all"**: **break**;

**default**: **die**(**"Error using getQuests: 1st argument is wrong.\n"**);

}

}

$sql .= **" ORDER BY "**;

**switch**($which){

**case** **"done"**: $sql .= **"donedate"**; **break**;

**default**: $sql .= **"pubdate"**; **break**;

}

$sql .= **" DESC LIMIT "**.($page\*10).**", "**.(10+$page\*10);

$result = **mysql\_query**($sql) **or** **die**(**mysql\_error**());

**mysql\_close**($\_conn);

**return** $this->db2array($result);

}

**function** addQuest($name,$text,$author){

$\_conn = $this->connectDb();

$name = $this->clearData($name,**'stext'**);

**if** ($name==**''**)

$name=**"'e砭ৢ୨'"**;

$text = $this->clearData($text,**'stext'**);

$sql = **"INSERT INTO quests (name, start\_txt, author, pubdate)**

**VALUES ($name, $text, '"**.$author.**"', "**.**time**().**")"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

**mysql\_close**($\_conn);

}

**function** editQuest($questId,$newName,$newTxt,$author){

$\_conn = $this->connectDb();

$questId = $this->clearData($questId,**'i'**);

$sql = **"SELECT id, author FROM quests**

**WHERE id=$questId"**;

$result=**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$result=**mysql\_fetch\_assoc**($result);

**if** ($result[**'author'**]==$author){

$newName = $this->clearData($newName,**'sf'**);

$newTxt = $this->clearData($newTxt,**'stext'**);

$sql = **"UPDATE quests**

**SET name=$newName,**

**start\_txt=$newTxt**

**WHERE id=$questId"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

}

**mysql\_close**($\_conn);

}

**function** deleteQuest($questId){

$\_conn = $this->connectDb();

$questId = $this->clearData($questId,**'i'**);

$sql = **"DELETE FROM messages**

**WHERE quest\_id=$questId"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$sql = **"DELETE FROM quests**

**WHERE id=$questId"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

**mysql\_close**($\_conn);

}

**function** markDone($questId, $username, $undone=0){

$\_conn = $this->connectDb();

$questId = $this->clearData($questId,**'i'**);

$username = $this->clearData($username);

$sql = **"SELECT id, done FROM quests**

**WHERE id=$questId"**;

$result=**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$result=**mysql\_fetch\_assoc**($result);

**switch**($undone){

**case** 0: **if** (!$result || $result[**'done'**]==1) **return** -1;

$sql = **"UPDATE quests**

**SET done=1,**

**who\_did=$username,**

**donedate="**.**time**().**"**

**WHERE id=$questId"**;

**break**;

**case** 1: **if** (!$result || $result[**'done'**]==0) **return** -1;

$sql = **"UPDATE quests**

**SET done=0,**

**who\_did=0,**

**donedate=0**

**WHERE id=$questId"**;

}

$result=**mysql\_query**($sql) **or** **die**(**mysql\_error**());

**return** 0;

}

**function** getComments($id, $page=0, $commentId=0){

$\_conn = $this->connectDb();

$id = $this->clearData($id,**"i"**);

$commentId = $this->clearData($commentId,**"i"**);

$sql = **"SELECT id, msg\_txt, author, pubdate, quest\_id**

**FROM messages"**;

**if** ($id!=0)

$sql .= **" WHERE quest\_id = $id"**;

**elseif** ($commentId!=0)

$sql .= **" WHERE id = $commentId"**;

$sql .= **" LIMIT "**.($page\*10).**", "**.(10+$page\*10);

$result = **mysql\_query**($sql) **or** **die**(**mysql\_error**());

**mysql\_close**($\_conn);

**return** $this->db2array($result);

}

**function** addComment($text, $questId, $author){

$\_conn = $this->connectDb();

$text = $this->clearData($text,**'stext'**);

$questId = $this->clearData($questId,**'i'**);

$sql = **"INSERT INTO messages (msg\_txt, quest\_id, author, pubdate)**

**VALUES ($text, $questId, '"**.$author.**"', "**.**time**().**")"**;

$result = **mysql\_query**($sql) **or** **die**(**mysql\_error**());

$sql = **"UPDATE quests**

**SET count\_of\_msg = count\_of\_msg+1**

**WHERE id = $questId"**;

$result = **mysql\_query**($sql) **or** **die**(**mysql\_error**());

**mysql\_close**($\_conn);

}

**function** editComment($commentId,$newTxt,$author){

$\_conn = $this->connectDb();

$commentId = $this->clearData($commentId,**'i'**);

$sql = **"SELECT id, author FROM messages**

**WHERE id=$commentId"**;

$result=**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$result=**mysql\_fetch\_assoc**($result);

**if** ($result[**'author'**]==$author){

$newTxt = $this->clearData($newTxt,**'stext'**);

$sql = **"UPDATE messages**

**SET msg\_txt=$newTxt**

**WHERE id=$commentId"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

}

**mysql\_close**($\_conn);

}

**function** deleteComment($msgId,$author){

$\_conn = $this->connectDb();

$msgId = $this->clearData($msgId,**'i'**);

$sql = **"SELECT author, quest\_id**

**FROM messages**

**WHERE id=$msgId"**;

$result=**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$result=**mysql\_fetch\_assoc**($result);

**if** ($result[**'author'**]==$author){

$sql = **"DELETE FROM messages**

**WHERE id = $msgId"**;

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

$sql = **"UPDATE quests**

**SET count\_of\_msg = count\_of\_msg-1**

**WHERE id = "**.$result[**'quest\_id'**];

**mysql\_query**($sql) **or** **die**(**mysql\_error**());

}

**mysql\_close**($\_conn);

**return** $result[**'quest\_id'**];

}

}

?>

#### quests.php

<?php

$count = $db->getQuestCount(**"actual"**);

**if** ($\_GET[**'page'**] == **NULL**) $page = 0;

**else** $page = $\_GET[**'page'**]-1;

$quests = $db->getQuests(**"actual"**,$page);

?>

<? **if** ($\_SESSION[**'auth\_type'**]==**'user'**){?>

<FORM class=**'form'** id=**'newQuest'** method=**'POST'** action=**'addQuest.php'**>

<SPAN>**Новое задание**</SPAN>

<SPAN id=**'minus'**>**-**</SPAN>

<SPAN>**Название:**</SPAN>

<INPUT type=**'text'** name=**'questName'** maxlength=**'33'**>

<TEXTAREA rows=**'6'** name=**'questText'**></TEXTAREA>

<INPUT type=**'submit'** value=**'Добавить'**>

</FORM><?

}

**if** ($count>10){

**if** (intval($\_GET[**'page'**])<=ceil($count/10)){?>

<DIV id=**'pages'**>

<?**for** ($i = 1; $i<=ceil($count/10); $i++){

**echo** **"<A class='pageNum'"**;

**if** ($i == $\_GET[**'page'**] || (!$\_GET[**'page'**] && $i == 1))

**echo** **" style='background-color:#efe'"**;

**echo** **" href='index.php?id=quests&page=$i'>$i</A>"**;

}?>

</DIV><?

}

}

**foreach** ($quests **as** $quest){?>

<DIV class=**'questDIV'**>

<DIV class=**'questTitle'**><?=$quest[**'name'**]?></DIV>

<DIV class=**'questText'**><?=$quest[**'start\_txt'**]?></DIV>

<DIV class=**'questInfo'**>

**Автор:** <A href=**'index.php?id=profile&user=**<?=$quest[**'author'**]?>**'**><?=$quest[**'author'**]?></A> **|**

**Дата:** <?=**date**(**"d:m:Y H:i:s"**,$quest[**'pubdate'**])?> **|**

<A class=**'controls'** href=**'index.php?id=comments&msgid=**<?=$quest[**'id'**]?>**'**>**Комментарии(**<?=$quest[**'count\_of\_msg'**]?>**)**</A><?

**if** ($quest[**'author'**]==$\_SESSION[**'userName'**]){?>

**|***&nbsp;*<A class=**'controls'** href=**'index.php?id=editQuest&msgid=**<?=$quest[**'id'**]?>**'**>**Редактировать**</A>

**|***&nbsp;*<A class=**'controls'** href=**'deleteQuest.php?msgid=**<?=$quest[**'id'**]?>**'**>**Удалить**</A><?

}?>

<?**if** ($\_SESSION[auth\_type]==**'user'**){?>

<SPAN class=**'markDone'** ><A href=**'markDone.php?msgid=**<?=$quest[**'id'**]?>**'**>**Завершено**</A></SPAN>

<?}?>

</DIV>

</DIV>

<? }

**if** ($count>10){

**if** (intval($\_GET[**'page'**])<=ceil($count/10)){?>

<DIV id=**'pages'**>

<?**for** ($i = 1; $i<=ceil($count/10); $i++){

**echo** **"<A class='pageNum'"**;

**if** ($i == $\_GET[**'page'**] || (!$\_GET[**'page'**] && $i == 1))

**echo** **" style='background-color:#efe'"**;

**echo** **" href='index.php?id=quests&page=$i'>$i</A>"**;

}?>

</DIV><?

}

}

?>

#### done.php

<style>

**#questsTab{**

**height:22px;**

**background-color:#cfc;**

**}**

**#doneTab{**

**z-index:3;**

**height:30px;**

**top:-64px;**

**background-color:#efe;**

**}**

</style>

<?php

$count = $db->getQuestCount(**"done"**);

**if** ($\_GET[**'page'**] == **NULL**) $page = 0;

**else** $page = $\_GET[**'page'**]-1;

$quests = $db->getQuests(**"done"**,$page);

**if** ($count>10){

**if** (intval($\_GET[**'page'**])<=ceil($count/10)){?>

<DIV id=**'pages'**>

<?**for** ($i = 1; $i<=ceil($count/10); $i++){

**echo** **"<A class='pageNum'"**;

**if** ($i == $\_GET[**'page'**] || (!$\_GET[**'page'**] && $i == 1))

**echo** **" style='background-color:#efe'"**;

**echo** **" href='index.php?id=quests&page=$i'>$i</A>"**;

}?>

</DIV><?

}

}

**foreach** ($quests **as** $quest){?>

<DIV class=**'questDIV'**>

<DIV class=**'questTitle'**><?=$quest[**'name'**]?></DIV>

<DIV class=**'questText'**><?=$quest[**'start\_txt'**]?></DIV>

<DIV class=**'questInfo'**>

**Автор:** <A href=**'index.php?id=profile&user=**<?=$quest[**'author'**]?>**'**><?=$quest[**'author'**]?></A> **|**

**Дата:** <?=**date**(**"d:m:Y H:i:s"**,$quest[**'pubdate'**])?> **|**

<A class=**'controls'** href=**'index.php?id=comments&msgid=**<?=$quest[**'id'**]?>**'**>**Комментарии(**<?=$quest[**'count\_of\_msg'**]?>**)**</A> **|**

**Завершено:** <?=**date**(**"d:m:Y H:i:s"**,$quest[**'donedate'**])?> **by** <B><?=$quest[**'who\_did'**]?></B>

<?**if** ($\_SESSION[auth\_type]==**'user'**){?>

<SPAN class=**'markDone'**><A href=**'markUndone.php?msgid=**<?=$quest[**'id'**]?>**'**>**Не**&nbsp**завершено**</A></SPAN>

<?}?>

</DIV>

</DIV>

<? }

**if** ($count>10){

**if** (intval($\_GET[**'page'**])<=ceil($count/10)){?>

<DIV id=**'pages'**>

<?**for** ($i = 1; $i<=ceil($count/10); $i++){

**echo** **"<A class='pageNum'"**;

**if** ($i == $\_GET[**'page'**] || (!$\_GET[**'page'**] && $i == 1))

**echo** **" style='background-color:#efe'"**;

**echo** **" href='index.php?id=quests&page=$i'>$i</A>"**;

}?>

</DIV><?

}

}

?>

#### comments.php

<STYLE>

**#questsTab{**

**height:22px;**

**background-color:#cfc;**

**}**

**#doneTab{**

**top:-64px;**

**}**

**#commentsTab{**

**display:block;**

**height:30px;**

**top:-128px;**

**background-color:#efe;**

**}**

</STYLE>

<?php

**if** ($\_GET[**'msgid'**]){

$quest = $db->getQuests(**""**,0,$\_GET[**'msgid'**]);

$count=$quest[0][**'count\_of\_msg'**];

**if** ($\_GET[**'page'**] == **NULL**) $page = 0;

**else** $page = $\_GET[**'page'**]-1;

$comments = $db->getComments($\_GET[**'msgid'**], $page);

?>

<DIV class=**'questDIV'**>

<DIV class=**'questTitle'**><?=$quest[0][**'name'**]?></DIV>

<DIV class=**'questText'**><?=$quest[0][**'start\_txt'**]?></DIV>

<DIV class=**'questInfo'** style=**'float:none;'**>

**Автор:** <A href=**'index.php?id=profile&user=**<?=$quest[0][**'author'**]?>**'**><?=$quest[0][**'author'**]?></A> **|**

**Дата:** <?=**date**(**"d:m:Y H:i:s"**,$quest[0][**'pubdate'**])?>

<?**if** ($quest[0][**'done'**]==1){

**echo** **"| Завершено: "**.**date**(**"d:m:Y H:i:s"**,$quest[0][**'donedate'**]).**" by <B>"**.$quest[0][**'who\_did'**].**"</B>"**;

}

**if** ($\_SESSION[auth\_type]==**'user'**){

**if** ($quest[0][**'done'**]==1){?>

<SPAN class=**'markDone'**><A href=**'index.php?id=markUndone&msgid=**<?=$quest[0][**'id'**]?>**'**>**Не**&nbsp**завершено**</A></SPAN>

<?} **else** {?>

<SPAN class=**'markDone'** ><A href=**'index.php?id=markDone&msgid=**<?=$quest[0][**'id'**]?>**'**>**Завершено**</A></SPAN>

<?}

}?>

</DIV>

</DIV>

<?**if** ($\_SESSION[auth\_type]==**'user'**){?>

<FORM class=**'form'** method=**'POST'** action=**'addComment.php'**>

<SPAN>**Новый комментарий**</SPAN>

<TEXTAREA rows=**'6'** name=**'newComment'**></TEXTAREA><BR>

<INPUT type=**'submit'** value=**'Добавить'**>

<INPUT type=**'hidden'** name=**'questId'** value=**'**<?=$\_GET[**'msgid'**]?>**'**>

</FORM><?

}

**else**{?>

<DIV class=**'form'**>

<SPAN>**Чтобы оставлять комментарии, Вы должны** <A href=**'index.php?id=register'**>**зарегистрироваться**</A></SPAN>

</DIV><?

}

**if** ($count>10){

**if** (intval($\_GET[**'page'**])<=ceil($count/10)){?>

<DIV id=**'pages'**>

<?**for** ($i = 1; $i<=ceil($count/10); $i++){

**echo** **"<A class='pageNum' href='index.php?id=comments&msgid=**{$quest[0]['id']}**&page=$i'>$i</A>"**;

}?>

</DIV><?

}

}?>

<DIV id=**'comments'**><?

**foreach**($comments **as** $comment){?>

<DIV class=**'comment'**>

<DIV class=**'questText'**><?=$comment[**'msg\_txt'**]?></DIV>

<DIV class=**'questInfo'**>

**Автор:** <A href=**'index.php?id=profile&user=**<?=$comment[**'author'**]?>**'**><?=$comment[**'author'**]?></A> **|**

**Дата:** <?=**date**(**"d:m:Y H:i:s"**,$comment[**'pubdate'**])?><?

**if** ($\_SESSION[**'auth\_type'**]==**'user'** && $\_SESSION[**'userName'**]==$comment[**'author'**]){?>

*&nbsp;***|***&nbsp;*

<A class=**'controls'** href=**'index.php?id=editComment&msgid=**<?=$comment[**'id'**]?>**'**>**Редактировать**</A> **|**

<A class=**'controls'** href=**'deleteComment.php?msgid=**<?=$comment[**'id'**]?>**'**>**Удалить**</A><?

}?>

</DIV>

</DIV><?

}?>

</DIV><?

**if** ($count>10){

**if** (intval($\_GET[**'page'**])<=ceil($count/10)){?>

<DIV id=**'pages'**>

<?**for** ($i = 1; $i<=ceil($count/10); $i++){

**echo** **"<A class='pageNum' href='index.php?id=comments&msgid=**{$quest[0]['id']}**&page=$i'>$i</A>"**;

}?>

</DIV><?

}

}

}

?>

#### editQuest.php

<STYLE>

**#questsTab{**

**height:22px;**

**background-color:#cfc;**

**}**

**#doneTab{**

**top:-64px;**

**}**

**#editTab{**

**display:block;**

**height:30px;**

**top:-128px;**

**background-color:#efe;**

**}**

</STYLE>

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

$db = QuestsDB::getInstance();

**if** ($\_SERVER[**'REQUEST\_METHOD'**]==**'GET'**){

$quest = $db->getQuests(**""**,0,$\_GET[**'msgid'**]);

**if** ($quest[0][**'author'**]==$\_SESSION[**'userName'**]){?>

<FORM class=**"form"** method=**'POST'** action=**'editQuest.php'**>

<SPAN>**Редактирование**</SPAN>

<SPAN>**Название:**</SPAN>

<INPUT type=**'text'** name=**'questName'** maxlength=**'100'** value=**'**<?=$quest[0][**'name'**]?>**'**><BR><BR>

<TEXTAREA rows=**'6'** name=**'questText'**><?=$quest[0][**'start\_txt'**]?></TEXTAREA><BR>

<INPUT type=**'submit'** value=**'Сохранить'**>

<INPUT type=**'hidden'** name=**'msgid'** value=**'**<?=$quest[0][**'id'**]?>**'**>

</FORM><?

}

**else**

header(**"location:index.php?id=quests"**);

}

**elseif**($\_SERVER[**'REQUEST\_METHOD'**]==**'POST'**){

$quest = $db->getQuests(**""**,0,$\_POST[**'msgid'**]);

**if** ($quest[0][**'author'**]==$\_SESSION[**'userName'**]){

$db->editQuest($quest[0][**'id'**],$\_POST[**'questName'**],$\_POST[**'questText'**],$\_SESSION[**'userName'**]);

}

header(**"location:index.php?id=quests"**);

}

?>

#### editComment.php

<STYLE>

**#questsTab{**

**height:22px;**

**background-color:#cfc;**

**}**

**#doneTab{**

**top:-64px;**

**}**

**#editTab{**

**display:block;**

**height:30px;**

**top:-128px;**

**background-color:#efe;**

**}**

</STYLE>

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

$db = QuestsDB::getInstance();

**if** ($\_SERVER[**'REQUEST\_METHOD'**]==**'GET'**){

$comment = $db->getComments(0,0,$\_GET[**'msgid'**]);

**if** ($comment[0][**'author'**]==$\_SESSION[**'userName'**]){

**if** ($\_SERVER[**'REQUEST\_METHOD'**]==**'GET'**){?>

<FORM class=**"form"** method=**'POST'** action=**'editComment.php'**>

<SPAN>**Редактирование**</SPAN>

<TEXTAREA rows=**'6'** name=**'commentText'**><?=$comment[0][**'msg\_txt'**]?></TEXTAREA><BR>

<INPUT type=**'submit'** value=**'Сохранить'**>

<INPUT type=**'hidden'** name=**'msgid'** value=**'**<?=$comment[0][**'id'**]?>**'**>

</FORM><?

}

}

**else**

header(**"location:index.php?id=quests"**);

}

**elseif**($\_SERVER[**'REQUEST\_METHOD'**]==**'POST'**){

$comment = $db->getComments(0,$\_POST[**'msgid'**]);

**if** ($comment[0][**'author'**]==$\_SESSION[**'userName'**]){

$result = $db->editComment($comment[0][**'id'**],$\_POST[**'commentText'**],$\_SESSION[**'userName'**]);

}

header(**"location:index.php?id=comments&msgid="**.$comment[0][**'quest\_id'**]);

}

?>

#### addQuest.php

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

**if** ($\_SERVER[**'REQUEST\_METHOD'**]==**'POST'**){

$db = QuestsDB::getInstance();

$db->addQuest($\_POST[**'questName'**],$\_POST[**'questText'**],$\_SESSION[**'userName'**]);

}

header(**"Location:index.php?id=quests"**);

?>

#### addComment.php

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

**if** ($\_SERVER[**'REQUEST\_METHOD'**]==**'POST'**){

$db = QuestsDB::getInstance();

$db->addComment($\_POST[**'newComment'**],$\_POST[**'questId'**],$\_SESSION[**'userName'**]);

}

header(**"Location:index.php?id=comments&msgid="**.$\_POST[**'questId'**]);

?>

#### deleteQuest.php

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

$db = QuestsDB::getInstance();

$quest = $db->getQuests(**""**,0,$\_GET[**'msgid'**]);

**if** ($quest[0][**'author'**]==$\_SESSION[**'userName'**]){

$db->deleteQuest($\_GET[**'msgid'**]);

}

header(**"location:index.php"**);

?>

#### deleteComment.php

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

**if** ($\_SESSION[**'auth\_type'**]==**'user'**){

$db = QuestsDB::getInstance();

$questId=$db->deleteComment($\_GET[**'msgid'**],$\_SESSION[**'userName'**]);

}

header(**"Location:index.php?id=comments&msgid=$questId"**);

?>

#### markDone.php

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

$db = QuestsDB::getInstance();

**if**($\_SERVER[**'REQUEST\_METHOD'**]==**'GET'** && $\_SESSION[**'auth\_type'**] == **'user'**){

$db->markDone($\_GET[**'msgid'**],$\_SESSION[**'userName'**]);

}

header(**"location:index.php?id=quests"**);

?>

#### markUndone.php

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

$db = QuestsDB::getInstance();

**if**($\_SERVER[**'REQUEST\_METHOD'**]==**'GET'** && $\_SESSION[**'auth\_type'**] == **'user'**){

$db->markDone($\_GET[**'msgid'**],$\_SESSION[**'userName'**],1);

}

header(**"location:index.php?id=done"**);

?>

#### register.php

<?php

**if** ($\_SESSION[**'auth\_type'**]==**'user'**)

header(**"Location:index.php"**);

?>

<STYLE>

**#questsTab{**

**height:22px;**

**background-color:#cfc;**

**}**

**#doneTab{**

**top:-64px;**

**}**

**#registerTab{**

**display:block;**

**height:30px;**

**top:-128px;**

**background-color:#efe;**

**}**

</STYLE>

<FORM id=**'registerForm'** method=**"POST"** action=**'addUser.php'**>

<TABLE>

<TR>

<TD><SPAN>**Имя:**</SPAN></TD>

<TD><INPUT id=**"enteredLogin"** type=**'text'** name=**'name'** size=**'30'** maxlength=**'30'**></TD>

<TD class=**'error'**><?=$\_SESSION[**'nameError'**]?></TD>

</TR>

<TR>

<TD><SPAN>**Пароль:**</SPAN></TD>

<TD><INPUT id=**'enteredPassword'** type=**'password'** name=**'password'** size=**"30"** maxlength=**'30'**></TD>

<TD class=**'error'**><?=$\_SESSION[**'passwordError'**]?></TD>

</TR>

<TR>

<TD><SPAN>**Email:**</SPAN></TD>

<TD><INPUT id=**'enteredEmail'** type=**'text'** name=**'email'** size=**"30"**></TD>

<TD class=**'error'**><?=$\_SESSION[**'emailError'**]?></TD>

</TR>

<TR>

<TD><SPAN>**Должность:**</SPAN></TD>

<TD><INPUT id=**'enteredPost'** type=**'text'** name=**'post'** size=**"30"**></TD>

</TR>

<TR>

<TD></TD>

<TD><INPUT type=**'submit'** value=**'Зарегистрироваться'**></TD>

<TD></TD>

</TR>

</TABLE>

</FORM>

#### login.php

<?php

**session\_start**();

**require\_once**(**"class.QuestsDB.php"**);

$db = QuestsDB::getInstance();

**if** ($\_SERVER[**'REQUEST\_METHOD'**]==**"POST"**){

**if** ($\_POST[**'name'**] && $\_POST[**'password'**]){

$user = $db->login($\_POST[**'name'**], $\_POST[**'password'**]);

**if** ($user[**"name"**] && $user[**"id"**]){

$\_SESSION[**'auth\_type'**]=**'user'**;

$\_SESSION[**'userName'**]=$user[**"name"**];

$\_SESSION[**'userID'**]=$user[**"id"**];

}

}

**if** ($\_POST[**'id'**] && $\_POST[**'id'**]!=**"register"**){

$queryString = **"?id=**{$\_POST['id']}**"**;

**if** ($\_POST[**'msgid'**])

$queryString .= **"&msgid=**{$\_POST['msgid']}**"**;

}

header(**"Location:index.php"**.$queryString);

**exit**;

}

?>

#### logout.php

<?php

**session\_start**();

$\_SESSION[**'auth\_type'**]=**'anonimous'**;

$\_SESSION[**'userName'**]=**''**;

$\_SESSION[**'userID'**]=**''**;

$queryString = **"?id=**{$\_POST['id']}**"**;

**if** ($\_POST[**'id'**]){

$queryString = **"?id=**{$\_POST['id']}**"**;

**if** ($\_POST[**'msgid'**])

$queryString .= **"&msgid=**{$\_POST['msgid']}**"**;

}

header(**"Location:index.php"**.$queryString);

**exit**;

?>

#### addUser.php

<?php

**session\_start**();

**if** ($\_SERVER[**'REQUEST\_METHOD'**]==**"POST"**){

$\_SESSION[**'nameError'**]=**""**;

$\_SESSION[**'emailError'**]=**""**;

**require\_once**(**"class.QuestsDB.php"**);

$db = QuestsDB::getInstance();

**if** ($\_POST[**'name'**] && $\_POST[**'password'**] && $\_POST[**'email'**]){

$result = $db->addUser($\_POST[**'name'**], $\_POST[**'password'**], $\_POST[**'email'**], $\_POST[**'post'**]);

**if** (is\_array($result)){

$\_SESSION[**'auth\_type'**]=**'user'**;

$\_SESSION[**'userName'**]=$result[**"name"**];

$\_SESSION[**'userID'**]=$result[**"id"**];

header(**"Location:index.php"**);

**exit**;

}

**else**{

**switch** ($result){

**case** 11: $\_SESSION[**'nameError'**]=**"Логин занят"**;

**case** 1: $\_SESSION[**'emailError'**]=**"Email занят"**;**break**;

**case** 10: $\_SESSION[**'nameError'**]=**"Логин занят"**;

}

header(**"Location:index.php?id=register"**);

**exit**;

}

}

}

header(**"Location:index.php?id=register"**);

?>