Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Легалов
подпись инициалы, фамилия
«___» _____ 2017 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специализации

Программное обеспечение лабораторных работ по дисциплине «Высокопроизводительные вычисления на графическом процессоре»
тема

Пояснительная записка

Руководитель	_____	доцент, к.т.н	Ю.В. Удалова
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		И.С. Сироткин
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	доцент, к.т.н	В. И. Иванов
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время все большее внимание уделяется развитию новых видов получения образования. Одной из таких технологий является дистанционное обучение. Дистанционное обучение - это новая технология получения образования, основанная на самостоятельном изучении предмета учащимся [1]. Такого рода технология обладает рядом преимуществ. Это облегчает процесс усвоения материала обучающимся, потому что нет географической привязки к месту учебы и нет необходимости личного присутствия преподавателя. Слушатель может в любом месте и в удобное для себя время заниматься изучением курса, для этого нужен лишь компьютер и доступ в интернет. Экономия времени и денежных средств налицо. Также стоит отметить, что обучение индивидуально для каждого ученика, так как слушатель не ограничен временными рамками, и может многократно изучать материал и выполнять разного рода практические работы.

Одной из основных форм дистанционного обучения являются веб-занятия – дистанционные уроки, конференции, семинары, деловые игры, лабораторные работы, практикумы и другие формы учебных занятий, проводимых с помощью средств телекоммуникаций и других возможностей Интернета [2].

Поэтому программное обеспечение, которое разрабатывается в данной работе, будет актуально для нашего времени. Человек без труда сможет перейти на страницу веб-приложения и воспользоваться его возможностями в любом месте в любое время.

1 Анализ задания на проектирование

Целью данной бакалаврской работы является создание веб-приложения для выполнения практикума лабораторных работ по дисциплине «разработка приложений, использующих компьютерную графику на основе спецификации OpenGL».

Из поставленной цели были выявлены следующие задачи:

– определения понятия – лабораторный практикум, анализ понятия в соотношении с целью работы;

– обзор существующих систем на основании понятия «лабораторный практикум»;

– изучить материалы необходимые для разработки веб-приложения;

– обзор существующих средств разработки и выбор оптимальных;

– разработка технического задания в соответствии с уже решенными задачами;

– реализация технического задания.

1.1 Понятие «Лабораторный практикум»

Лабораторный практикум – элемент учебного процесса, заключающийся в том, что обучающийся выполняет практические действия в определенной области. Как правило, это курс лабораторных работ, которые последовательно выполняются обучающимся, тем самым приобретаются систематизированные знания.

Каждая лабораторная работа должна состоять из теоретической части, и инструкцией к выполнению работы. Необязательным, но полезным атрибутом является наличие примера по данной теме.

1.2 Обзор существующих систем

На сегодняшний день существует большое количество различных электронных лабораторных практикумов, применяемых в разных дисциплинах. Ниже приведены некоторые из них:

1. Лабораторный практикум по информатике Сибирского государственного университета геосистем и технологий [3];
2. Лабораторный практикум по работе с Open Graphics Library (OpenGL) [4];
3. Лабораторный практикум по работе с HyperText Markup Language (HTML) [5].

1.2.1 Лабораторный практикум по информатике

Лабораторный практикум по информатике Сибирского государственного университета геосистем и технологий содержит список из 33 лабораторных работ, в которых содержится теоретическая информация и задания к выполнению. Пример лабораторной работы приведен на рисунке 1.

Электронный учебно-методический комплекс
кафедры прикладной информатики
и информационных систем

Лабораторная работа № 1.
Представление информации в ЭВМ

Время выполнения

2 часа

Цель работы

Научиться переводить числа в те системы счисления, которые использует ЭВМ, подсчитывать объем занимаемой данными информации и уметь переводить значения количества информации из одних единиц измерения в другие.

Задачи лабораторной работы

После выполнения работы студент должен знать и уметь:

1. знать основные приемы работы с позиционными системами счисления;
2. уметь переводить числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную;
3. производить обратный перевод из этих систем в десятичную;
4. уметь переводить значения из одних единиц измерения информации в другие.

Перечень обеспечивающих средств

Для обеспечения выполнения работы необходимо иметь компьютер с операционной системой и методические указания по выполнению работы.

Общие теоретические сведения

Система счисления – это способ представления чисел цифровыми знаками и соответствующие ему правила действий над числами.

Системы счисления можно разделить:

- непозиционные системы счисления;
- позиционные системы счисления.

В непозиционной системе счисления значение (величина) символа (цифры) не зависит от положения в числе.

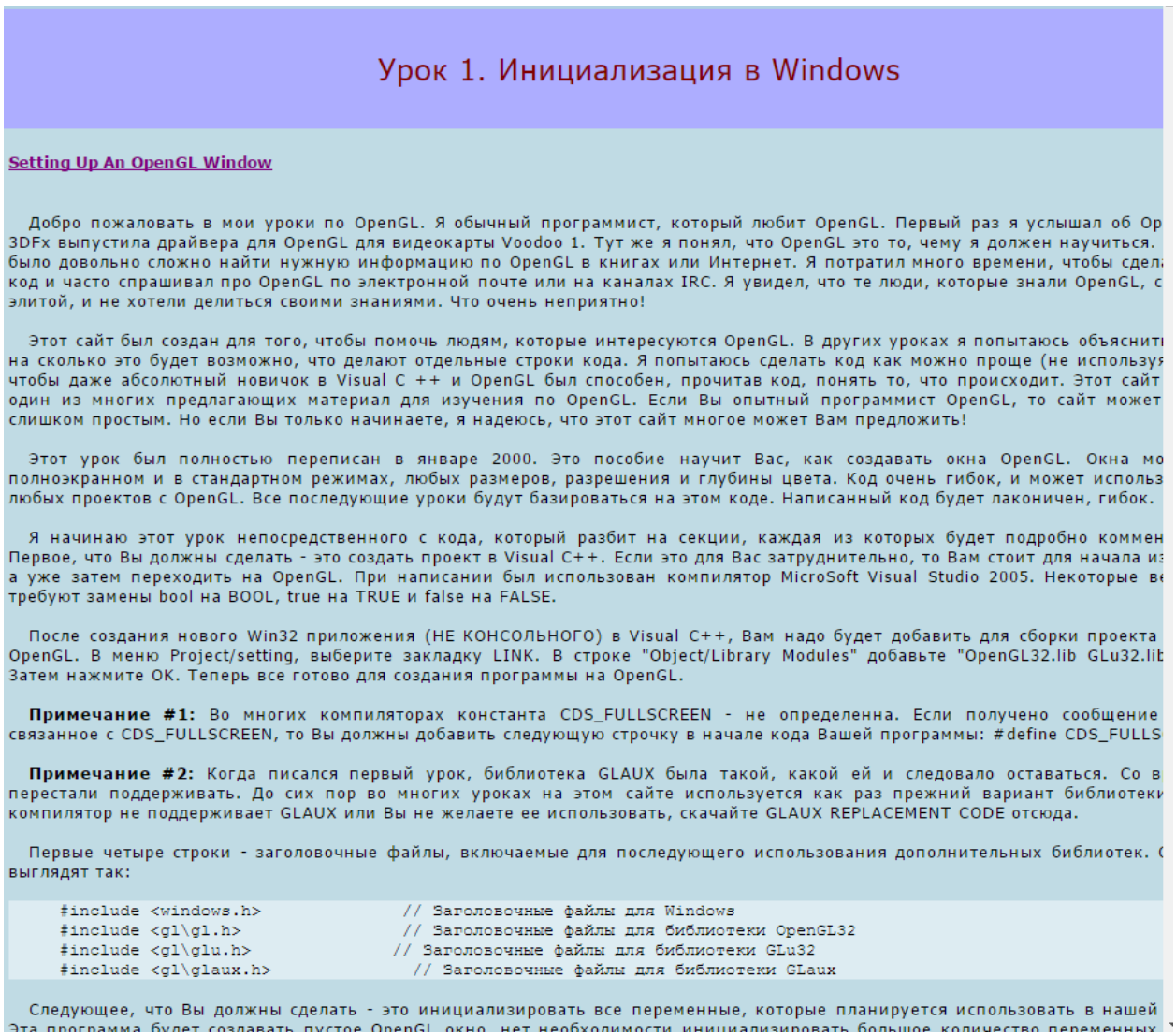
Самой распространенной непозиционной системой счисления является **римская**.

Рисунок 1 – Пример лабораторной работы по информатике

1.2.2 Лабораторный практикум по работе с OpenGL

Лабораторный практикум по работе со спецификацией OpenGL содержит список лабораторных работ, напротив которых изображена картинка с примерным результатом выполнения данной работы. Внутри лабораторной работы содержится теоретическая часть, в которой присутствуют элементы кода для написания программы OpenGL. В конце работы содержится ссылка для скачивания файла с исходным кодом, если слушатель не справился с выполнением работы. Из недостатков можно отметить отсутствие отладки и

компиляции файлов прямо из веб-приложения. Пример лабораторной работы приведен на рисунке 2.



Урок 1. Инициализация в Windows

Setting Up An OpenGL Window

Добро пожаловать в мои уроки по OpenGL. Я обычный программист, который любит OpenGL. Первый раз я услышал об Op 3DFx выпустила драйвера для OpenGL для видеокарты Voodoo 1. Тут же я понял, что OpenGL это то, чему я должен научиться. было довольно сложно найти нужную информацию по OpenGL в книгах или Интернет. Я потратил много времени, чтобы сдел код и часто спрашивал про OpenGL по электронной почте или на каналах IRC. Я увидел, что те люди, которые знали OpenGL, с элитой, и не хотели делиться своими знаниями. Что очень неприятно!

Этот сайт был создан для того, чтобы помочь людям, которые интересуются OpenGL. В других уроках я попытаюсь объясниті на сколько это будет возможно, что делают отдельные строки кода. Я попытаюсь сделать код как можно проще (не используя чтобы даже абсолютный новичок в Visual C ++ и OpenGL был способен, прочитав код, понять то, что происходит. Этот сайт один из многих предлагающих материал для изучения по OpenGL. Если Вы опытный программист OpenGL, то сайт может слишком простым. Но если Вы только начинаете, я надеюсь, что этот сайт многое может Вам предложить!

Этот урок был полностью переписан в январе 2000. Это пособие научит Вас, как создавать окна OpenGL. Окна мо полноэкранном и в стандартном режимах, любых размеров, разрешения и глубины цвета. Код очень гибок, и может использ любых проектов с OpenGL. Все последующие уроки будут базироваться на этом коде. Написанный код будет лаконичен, гибок.

Я начинаю этот урок непосредственного с кода, который разбит на секции, каждая из которых будет подробно коммен Первое, что Вы должны сделать - это создать проект в Visual C++. Если это для Вас затруднительно, то Вам стоит для начала из а уже затем переходить на OpenGL. При написании был использован компилятор MicroSoft Visual Studio 2005. Некоторые ве требуют замены bool на BOOL, true на TRUE и false на FALSE.

После создания нового Win32 приложения (НЕ КОНСОЛЬНОГО) в Visual C++, Вам надо будет добавить для сборки проекта OpenGL. В меню Project/setting, выберите закладку LINK. В строке "Object/Library Modules" добавьте "OpenGL32.lib GLu32.lib Затем нажмите OK. Теперь все готово для создания программы на OpenGL.

Примечание #1: Во многих компиляторах константа CDS_FULLSCREEN - не определенна. Если получено сообщение связанное с CDS_FULLSCREEN, то Вы должны добавить следующую строчку в начале кода Вашей программы: #define CDS_FULLS

Примечание #2: Когда писался первый урок, библиотека GLAUX была такой, какой ей и следовало оставаться. Со в перестали поддерживать. До сих пор во многих уроках на этом сайте используется как раз прежний вариант библиотеки компилятор не поддерживает GLAUX или Вы не желаете ее использовать, скачайте GLAUX REPLACEMENT CODE отсюда.

Первые четыре строки - заголовочные файлы, включаемые для последующего использования дополнительных библиотек. выглядят так:

```
#include <windows.h>        // Заголовочные файлы для Windows
#include <gl\gl.h>          // Заголовочные файлы для библиотеки OpenGL32
#include <gl\glu.h>         // Заголовочные файлы для библиотеки GLu32
#include <gl\glaux.h>       // Заголовочные файлы для библиотеки GLaux
```

Следующее, что Вы должны сделать - это инициализировать все переменные, которые планируется использовать в нашей Эта программа будет создавать пустое OpenGL окно, нет необходимости инициализировать большое количество переменных

Рисунок 2 – Пример лабораторной работы по OpenGL

1.2.3 Лабораторный практикум по работе с HTML

Лабораторный практикум по работе с HTML от «htmlacademy.ru» содержит 14 уроков, каждый из которых содержит приветственное окно с теоретической информацией по работе, далее работа продолжается в основном окне, где расположены 3 окна: правое окно отвечает за инструкцию к выполнению работы, а левое верхнее и нижнее окна являются редакторами

кода для написания программ на HTML и CSS. Основными плюсами можно отметить компиляцию исходного кода прямо на сайте, а также удобный интерфейс. Пример лабораторной работы приведен на рисунке 3.

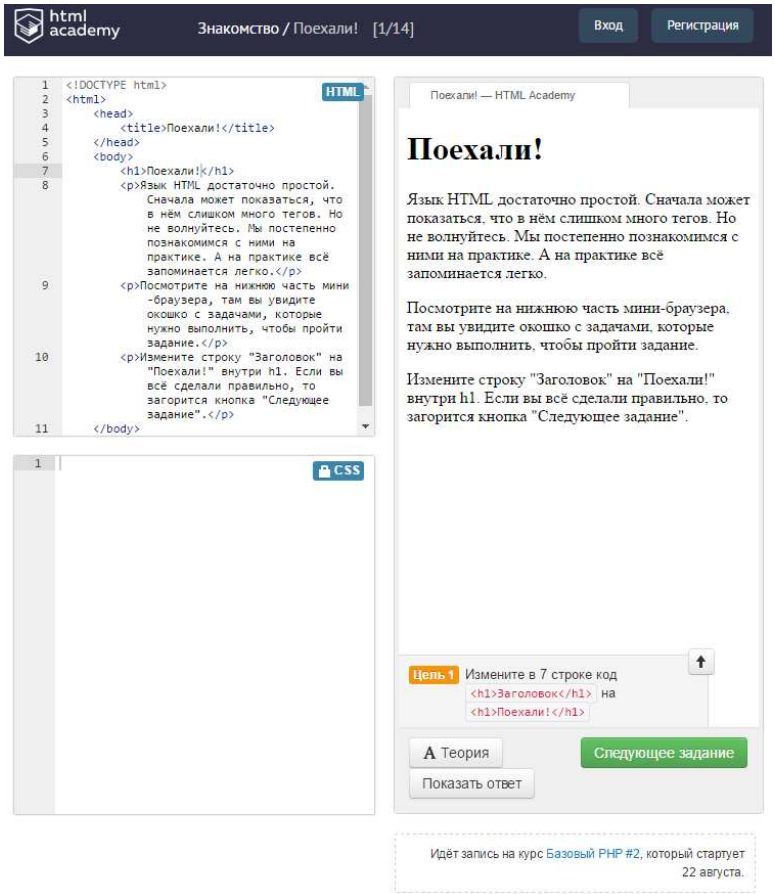


Рисунок 3 – Пример лабораторной работы по HTML

1.3 Выбор существующих средств разработки

1.3.1 Обзор систем управления базами данных

Одной из главных задач при проектировании является выбор системы управления базами данных (СУБД). Это влияет на быстродействие системы, ее дальнейшую модернизацию, а также на стоимость использования системы. Далее кратко описано наиболее используемые СУБД в настоящее время.

1.3.1.1 Oracle

Oracle включает СУБД и средства разработки и анализа данных. Oracle включает базу данных (БД), интеграционную платформу, сервер приложений, инструменты управления неструктурированными данными и аналитики. СУБД Oracle Database позволяет автоматизировать задачи администрирования, обеспечивает безопасность и соответствие нормативно-правовым актам защиты информации, содержит функции управления и самодиагностики. К характеристикам системы относится управление большими объемами данных с помощью использования компрессии и распределенных таблиц, эффективная защита данных, возможность интеграции геофизических данных и полного восстановления и так далее [6].

Основными достоинствами можно отметить много дополнительных возможностей, а также версионный сервер. Недостатком данной системы является высокая стоимость сервера и технической поддержки.

1.3.1.2 Microsoft SQL Server

Microsoft SQL Server (MS SQL) предлагает обширный спектр услуг администрирования. Это позволяет использовать ее в информационных системах для среднего бизнеса и компьютерных информационных системах. В основе платформы MS SQL Server используется среда Windows. Главное преимущество программы – интеграция с программными продуктами от Microsoft и возможность экспорта/импорта данных в большинство распространенных форматов данных, что позволяет использовать MS SQL Server как центральное хранилище данных [7].

Основными достоинствами является то, что данная СУБД быстро развивается и быстро приближается к конкурентам, а также ее средняя стоимость. Недостатком является системы является существование только для одной платформы Windows.

1.3.1.3 MongoDB

MongoDB – это система управления базами данных, «заточенная» под веб-приложения и инфраструктуру Интернета. Модель данных и стратегия их постоянного хранения спроектированы для достижения высокой пропускной способности чтения и записи и обеспечивает простую масштабируемость с автоматическим переходом на резервный ресурс в случае отказа [8].

Основными достоинствами является бесплатное распространение и высокая скорость работы. Основными минусами является уникальный язык запросов и неустойчивость к разного рода атакам.

1.3.2 Обзор средств для разработки логики сервера

Еще одной важной частью разработки веб-приложения является выбор средств разработки логики сервера. Самыми популярными и удобными являются Javascript, Ruby и Python.

1.3.2.1 JavaScript и Node.js

Язык программирования JavaScript создан для определения поведения веб-страниц. В базовой структуре JavaScript определён минимальный набор библиотек для работы с текстом, массивами, датами и регулярными выражениями, но средства ввода-вывода данных в него не входят. Основные архитектурные черты: динамическая типизация, автоматическое управление памятью [9]. Основными достоинствами является простота синтаксиса, полезные функциональные настройки, взаимодействие с приложением может осуществляться через любые текстовые редакторы. Основными недостатками являются пониженный уровень безопасности ввиду открытости популярных скриптов.

Node.js – программная платформа, основанная на движке V8, компилирующая JavaScript во внутренний машинный код. Достоинствами Node.js является простота установки и настройки, независимость Node.js от платформы, однопоточность Node позволяет упростить задачу написания веб-приложений, а задача получения высокой производительности решается с помощью серверного параллелизма [10].

1.3.2.2 Ruby

Ruby – объектно-ориентированный динамический язык программирования. Особенностями Ruby является интерпретируемость, гибкий и мощный синтаксис, автоматический сбор «мусора», строгая динамическая типизация, а также встроенная поддержка Unicode, возможность создания консольных, графических, мобильных и веб-приложений [11]. Основным недостатком является меньшая производительность по сравнению с другими языками, которые используются для написания веб-приложений.

1.3.2.3 Python

Python – язык общего назначения с открытым исходным кодом, применением множества парадигм, поддержкой структур объектно-ориентированного, функционального и процедурного программирования. К преимуществам Python можно отнести простоту изучения, удобство синтаксиса языка и читаемости кода, переносимость программ и интеграция их компонентов [12]. Недостатками являются ограниченность средств для работы с СУБД, производительность языка по сравнению с Java ниже.

1.3.3 Обзор графической системы OpenGL

Графическая система OpenGL – это программный интерфейс к графическому оборудованию. Она позволяет создавать интерактивные программы, которые формируют движущиеся цветные изображения в трехмерном пространстве. Для понимания работы данной спецификации необходимо иметь начальные знания по математике (геометрии, тригонометрии, линейной алгебре и т.д.).

OpenGL содержит порядка 250 отдельных команд, которые используются для задания объектов и операций, необходимых для создания интерактивных приложений трехмерной графики.

OpenGL разработан как аппаратно-независимый интерфейс для работы на различных аппаратных платформах. Поэтому в OpenGL не включены команды для управления окнами и организации пользовательского ввода. Вся такая работа ведётся через операционную систему. Аналогично, OpenGL не имеет высокоуровневых команд для описания трехмерных сложных моделей.

Задача OpenGL – помочь программисту создать модель из небольшого набора графических примитивов: точек, линий и многоугольников. Высокоуровневые средства предоставляются библиотеками, являющимися надстройками над OpenGL. Для моделирования кривых и поверхностей предназначена библиотека OpenGL Utility Library (GLU), имеющая множество инструментов [13].

После обзора приведенных выше средств разработки были выбраны и использованы такие средства как:

- HTML и CSS, отвечающие за разработку веб-интерфейса;
- NodeJS, отвечающий за разработку логики сервера;
- MongoDB, отвечающий за хранение лабораторных работ в БД;
- OpenGL, отвечающий за содержание лабораторных работ.

1.4 Техническое задание

1.4.1 Наименование системы

Программное обеспечение лабораторных работ по дисциплине «Высокопроизводительные вычисления на графическом процессоре на базе спецификации OpenGL».

1.4.2 Назначение и цель создания системы

Назначением является обучение слушателей курса работе со спецификацией OpenGL.

Основной целью создания системы является дистанционное электронное выполнение лабораторных работ по спецификации OpenGL с возможностью отладки и компиляции непосредственно из приложения.

1.4.3 Требования к системе

1.4.3.1 Требования к структуре и функциональной части веб-приложения

Функциональная структура веб-приложения должна включать в себя:

– простой и понятный интерфейс;
– быстроту работы;
– малую ресурсоемкость;
– возможность запуска приложения в любом браузере.

Система должна обладать следующими возможностями:

– возможность редактирования пользователем исходного кода лабораторной работы, а также его запуск непосредственно из веб-приложения;

– хранение в базе данных созданных и отредактированных лабораторных работ;

– поддержка администрирования системы: создание лабораторных работ, изменение лабораторных работ, удаление лабораторных работ.

1.4.3.2 Требования к способам и средствам связи для информационного обмена между компонентами системы

Информационный обмен между модулями логики сервера системы и веб-интерфейсами системы должен осуществляться через всемирную систему объединённых компьютерных сетей для хранения и передачи информации – интернет посредством использования стандартизированных протоколов и форматов обмена данными.

1.4.3.3 Перспективы развития системы

Веб-приложение должно быть выполнено с возможностью дальнейшей модернизации. Веб-приложение должно допускать изменение дизайна и интерфейса, а также должно обеспечивать возможность замены частей исходного кода с целью улучшения работоспособности.

1.4.3.4 Требования к надежности

Время восстановления работоспособности веб-приложения при любых сбоях и отказах не должно превышать одного рабочего дня, исключая случаи неисправности серверного оборудования.

1.4.3.5 Требования по эргономике и технической эстетике

При разработке визуальной части программы должны учитываться психофизиологические особенности человека:

- не слишком яркая цветовая палитра;
- информация должно выводится простым и понятным способом;
- положение визуальной составляющей веб-приложения должно соответствовать структуре сайта.

1.4.4 Требования к видам обеспечения

1.4.4.1 Требования к лингвистическому обеспечению

Языки программирования

Разработка программного обеспечения должна вестись с использованием языков высокого уровня.

Языки взаимодействия пользователей и системы

Основным языком взаимодействия пользователей и системы является русский язык. Язык исходного кода, представленного в лабораторных работах – английский.

1.4.4.2 Требования к программному обеспечению

Программное обеспечение должно быть представлено в виде веб-приложения, которое поддерживается всеми современными браузерами, а также комплект программ для развертывания базы данных для хранения лабораторных работ и компиляции исходного кода OpenGL непосредственно из веб-приложения.

1.4.4.3 Требования к аппаратному обеспечению

Операционная система

- Windows XP, Vista, 7,8,8.1, Server 2003, Server 2008
- Ubuntu 12.04 или более поздней версии
- Debian 7 или более поздней версии
- OpenSuSE 12.2 или более поздней версии
- Fedora Linux 17 или более поздней версии
- Mac OS X 10.6 и более поздней версии.

Процессор

Процессор с тактовой частотой 233 мегагерца (МГц) или выше.

Свободное место на диске

Не более 350 МБ.

Оперативная память

512 МБ.

Устройство чтения дисков

Дисковод компакт-дисков (если установка выполняется с компакт-диска).

Дисплей

Монитор с разрешением Super VGA (800 x 600) или более высоким и поддержкой 256 цветов.

Периферийные устройства

- Модем или подключение к Интернету;
- Мышь или другое совместимое указывающее устройство.

2 Разработка архитектуры и основных технических решений

2.1 Функциональная модель веб-приложения

Цель работы – разработка веб-приложения для дистанционного электронного выполнения лабораторных работ по спецификации OpenGL. С помощью данного веб-приложения учащиеся могут ознакомиться с предоставляемым материалом и выполнить лабораторный практикум. В каждой лабораторной работе будет предоставлена возможность редактировать код программы, а также производить отладку и компиляцию программы непосредственно из веб-приложения.

При входе в приложение авторизации не происходит, дальнейшая авторизация администратора возможна при входе в настройки лабораторного практикума.

2.2 Информационная структура веб-приложения

2.2.1 Общая архитектура системы

При разработке веб-приложения была выбрана клиент-серверная модель. Её преимущества заключаются в том, что функции вычислительной системы можно разделить между различными компьютерами, также при клиент-серверной модели все данные хранятся на централизованном сервере, что обеспечивает высокую степень целостности и безопасности данных.

Рисунок 4 – Архитектура «Клиент-сервер»

Работа данной архитектуры заключается в обмене информацией между сервером и клиентом. В данной модели клиент (пользователь) через интерфейс HTML-страницы в браузере выполняет запрос к серверу, и получает список лабораторных работ к выполнению. После, при выборе лабораторной работы происходит загрузка данных, связанных с заданием. Принцип выполнения задания заключается в написании или изменении кода программы, которая рисует графику с помощью спецификации OpenGL. В процессе выполнения веб-приложение посылает запрос к серверу о компиляции и запуске программы, после чего формирует ответ клиенту в виде окна с программой, демонстрирующей графику. При неверном выполнении лабораторной работы выводится лог с ошибками.

2.2.2 Диаграмма прецедентов

В приложении предусмотрено 2 роли, такие как администратор (преподаватель) и пользователь (учащийся). Диаграмма прецедентов изображена на рисунке 5.

Функции пользователя:

- просмотр лабораторных работ;
- редактирование лабораторных работ в процессе выполнения;
- компиляция и запуск лабораторных работ.

Функции администратора:

- просмотр лабораторных работ;
- компиляция и запуск лабораторных работ;
- добавление, изменение и удаление лабораторных работ.

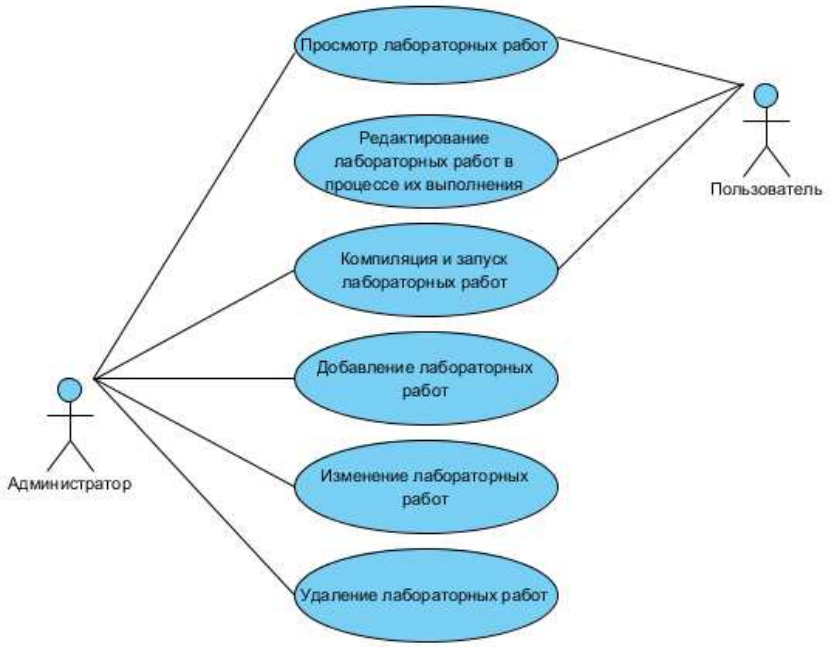


Рисунок 5 – Диаграмма прецедентов

2.3 Разработка структуры базы данных

Для организации работы системы требуется создание базы данных лабораторных работ в виде списка. Каждая лабораторная работа содержит 5 полей:

– поле идентификатора лабораторной работы (id);
– поле названия лабораторной работы (name);
– поле описания лабораторной работы (descr);
– поле листинга лабораторной работы (code);
– поле первоначального листинга лабораторной работы (defcode).

Последнее поле необходимо для того, чтобы была возможность восстановить первоначальный код лабораторной работы после его редактирования пользователем.

2.4 Разработка интерфейса веб-приложения

Веб-приложение будет состоять из нескольких страниц:

– страница приветствия;

– страница списка лабораторных работ;

– страница выполнения лабораторной работы;

– страница настроек компилятора и входа в страницу администратора;

– страница администратора.

2.5 Программная реализация веб-приложения

Для начального запуска сервера был создан файл **app.js**, с помощью которого приложение подключает необходимые стили, библиотеки, редакторы текста, а также поддержку русского языка. Также был создан файл **router.js**, с помощью которого происходит вся маршрутизация, а также реализуется весь необходимый функционал веб-приложения. Подробное содержание файла продемонстрировано в приложении.

2.5.1 Разработка страницы приветствия

При входе в веб-приложение пользователя встречает страница приветствия, на которой пользователь может получить первичную информацию о данном приложении, а также перейти в необходимый ему пункт меню, который расположен слева (рисунок 6).

Для создания данной страницы был разработан файл **index.ejs**, а также файл **head.ejs**, который отвечает за шапку веб-страницы, и файл **leftpanel.ejs**, который отвечает за боковое меню веб-приложения.

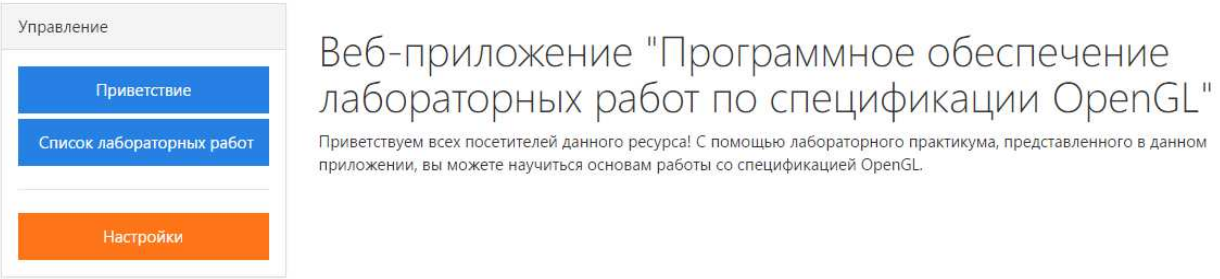


Рисунок 6 – Страница приветствия веб-приложения

2.5.2 Разработка страницы списка лабораторных работ

При переходе на вкладку «Список лабораторных работ» открывается список лабораторных работ, подлежащих выполнению. Каждая лабораторная работа пронумерована и имеет своё уникальное название (рисунок 7).

Для разработки данной веб-страницы был разработан файл **lablist.ejs**. Также для отображения лабораторных работ, хранимых в базе данных, в файле **router.js** производится подключение к базе данных и сортировка списка лабораторных работ по их идентификационному номеру (id). За данные операции отвечает метод **router.get('/labs', function(req,res)**.

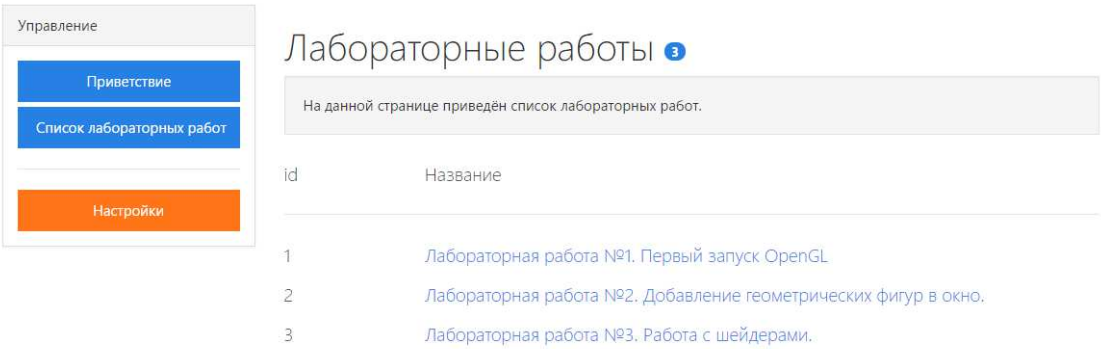


Рисунок 7 – Окно списка лабораторных работ

2.5.3 Разработка страницы содержимого лабораторной работы

Страница содержимого лабораторной работы предназначена для ознакомления с описанием данной работы, работы с кодом программы, а также для его последующей компиляции и запуске непосредственно из приложения (рисунок 8,9).

Для создания данной страницы был создан файл **lab.ejs**.

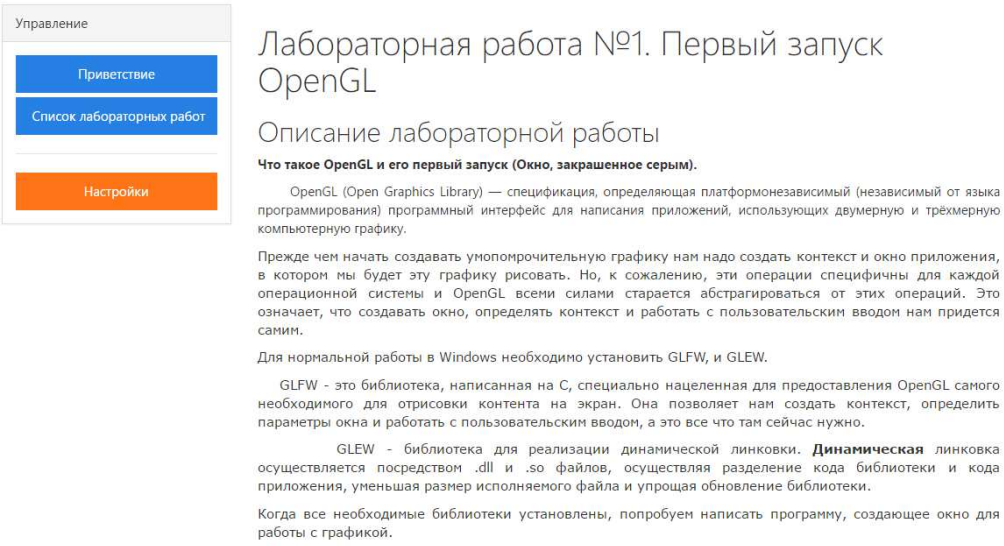


Рисунок 8 – Описание лабораторной работы

Код программы

Редактировать | Восстановить код | Компилировать

```
// GLEW нужно подключать до GLFW.
// GLEW
#define GLEW_STATIC
#include <GL/glew.h>
// GLFW
#include <GLFW/glfw3.h>
#include <iostream>

//Прототип функции
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode);

int main()
{
        //Инициализация GLFW
        glfwInit();
        //Настройка GLFW
        //Задается минимальная требуемая версия OpenGL.
        //Мажорная
        glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
        //Минорная
        glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
        //Установка профайла для которого создается контекст
        glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
        //Выключение возможности изменения размера окна
        glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);\

        //создание объекта окна
        GLFWwindow* window = glfwCreateWindow(800, 600, "LearnOpenGL", nullptr, nullptr);
        if (window == nullptr)
        {
                std::cout << "Failed to create GLFW window" << std::endl;
                glfwTerminate();
                return -1;
        }
        glfwMakeContextCurrent(window);

        //инициализация GLEW
```

Рисунок 9 – Листинг программы лабораторной работы и меню редактора кода

Для редактирования кода лабораторной работы (кнопка «редактировать») в файле **router.js** был создан метод **router.post('/labs/:id/edit', function(req,res)**.

Кнопка «Восстановить код» необходима для того, чтобы вернуться к исходному коду программы, если пользователь неправильно изменил код лабораторной работы. При восстановлении кода текст изменённой лабораторной работы (переменная **code**) заменяется на первоначальный код (переменная **defcode**), который присутствует в базе данных при создании данной лабораторной работы. За данную операцию отвечает метод **router.get('/labs/:id/default',function(req,res)** файле **router.js**.

При редактировании кода лабораторной работы открывается текстовый редактор. В редакторе кода возможно редактирование исходного кода, отменить или вернуть действие, а также компилировать исходный код. После редактирования необходимо нажать кнопку «Сохранить код» (рисунок 10).

Код программы

Сохранить код | Компилировать | Выбрать все | Отменить | Повторить

```
1  // GLEW нужно подключать до GLFW.
2  // GLEW
3  #define GLEW_STATIC
4  #include <GL/glew.h>
5  // GLFW
6  #include <GLFW/glfw3.h>
7  #include <iostream>
8
9  //Прототип функции
10 void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode);
11
12 int main()
13 {
14     //Инициализация GLFW
15     glfwInit();
16     //Настройка GLFW
17     //Задается минимальная требуемая версия OpenGL.
18     //Мажорная
19     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
20     //Минорная
21     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
22     //Установка профайла для которого создается контекст
23     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
24     //Выключение возможности изменения размера окна
25     glfwWindowHint(GLFW_RESIZABLE, GL_FALSE);\
26
27     //создание объекта окна
28     GLFWwindow* window = glfwCreateWindow(800, 600, "LearnOpenGL", nullptr, nullptr
29     if (window == nullptr)
30     {
31         std::cout << "Failed to create GLFW window" << std::endl;
32         glfwTerminate();
33         return -1;
34     }
35     glfwMakeContextCurrent(window);
36
```

Рисунок 10 – Редактирование информации пользователем

За кнопки на данной странице отвечает функция **edit**, а также метод **ajaxcompile**, который позволяет компилировать исходный код без сохранения с возможностью редактирования, и метод **ajaxsave**, который позволяет сохранить отредактированный код. Данные методы и функции находятся в файле **lab.ejs.**

2.5.4 Компиляция исходного кода лабораторной работы

Кнопка «Компилировать» необходима для отладки и запуска лабораторных работ. При правильном написании программы открывается окно с графикой, создаваемой при помощи OpenGL (рисунок 11), иначе выводится лог ошибок (рисунок 12). Для отображения страницы с логом ошибок компилятора был создан файл **compilelog.ejs**.

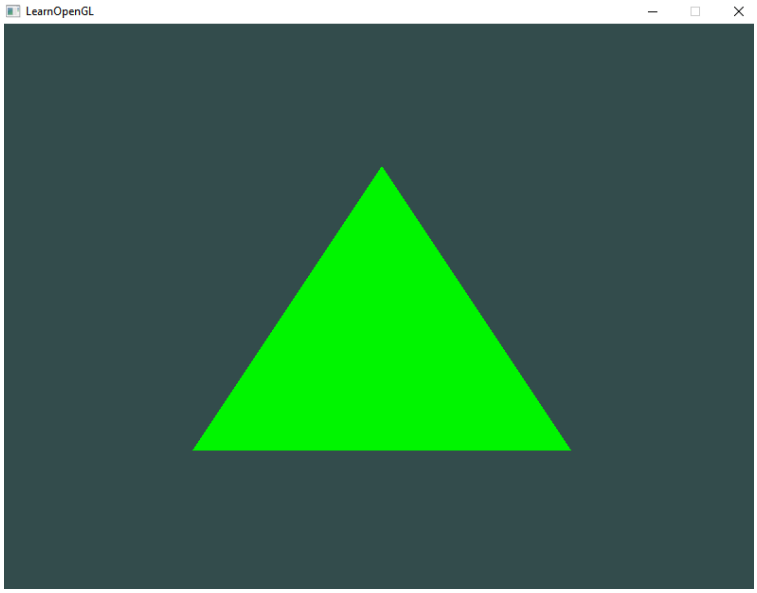


Рисунок 11 – Успешный результат компиляции исходного кода лабораторной работы №3

Рисунок 12 – Вывод ошибок при неправильном написании кода программы

Компиляция проекта происходит следующим образом: при нажатии кнопки «Компилировать» код программы в лабораторной работе сохраняется в виде файла **lastsource.cpp**, далее формируется команда компиляции, которая состоит из названия компилятора, исполняемого файла, и параметров компиляции. При успешной компиляции программа запускается, иначе выводится лог ошибок. Ниже приведена реализация метода **router.get('/run/:id',function(req,res)** в файле **router.js**.

```
//Компиляция проекта
router.get('/run/:id',function(req,res)
{
	//Настройки компиляции
	//Загрузка параметров компиляции из settings.json
	var compile = JSON.parse(fs.readFileSync('settings.json', 'utf8'));
	//Подключение к БД с лабораторными работами
	MongoClient.connect(dburl, function(err, db)
	{
		var collection = db.collection('labs')
		.findOne({id:req.params.id},function(err,docs)
		{
			//Запись кода лабораторной работы в файл lastsource.cpp
			 saved = fs.writeFileSync('compile/lastsource.cpp', docs.code, 'utf8');
			console.log('СОХРАНЕНО');
			//Формирование команды компиляции
			var cmd = compile.path + "compile/lastsource.cpp" + compile.params+"
			&& a.exe";
			//Выполнение команды компиляции
			 exec(cmd, function(error, stdout, stderr)
			{
				//Вывод лога ошибок при их наличии
				if (error) res.render('compilelog',{err:stderr,command:cmd});
				else res.render('success');
```

```
                    });
                });
            });
})
```

2.5.5 Разработка страницы настроек лабораторного практикума

Последним пунктом главного меню является страница настроек лабораторного практикума. На данной странице возможно изменение настроек компилятора для запуска программ лабораторных работ, а также вход на страницу администратора (рисунок 13). Для отображения данной страницы был создан файл **options.ejs**.

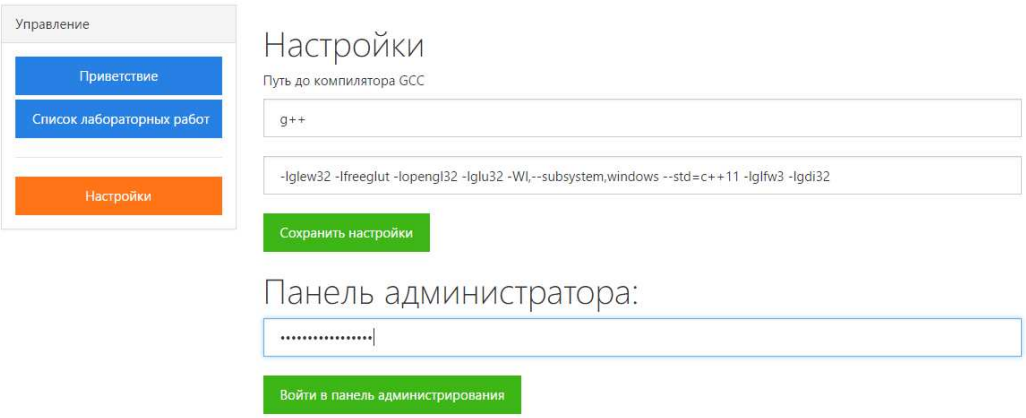


Рисунок 13 – Страница настроек лабораторного практикума

При вводе верного пароля (пол умолчанию «ADMIN») в строке ввода осуществляется переход на страницу администратора (рисунок 14), где возможно производить добавление, а также изменение/удаление любой лабораторной работы.

Переход к данной странице, подключение базы данных, а также проверка пароля производится с помощью метода **router.post('/admin',function(req,res)** в файле **router.js**.

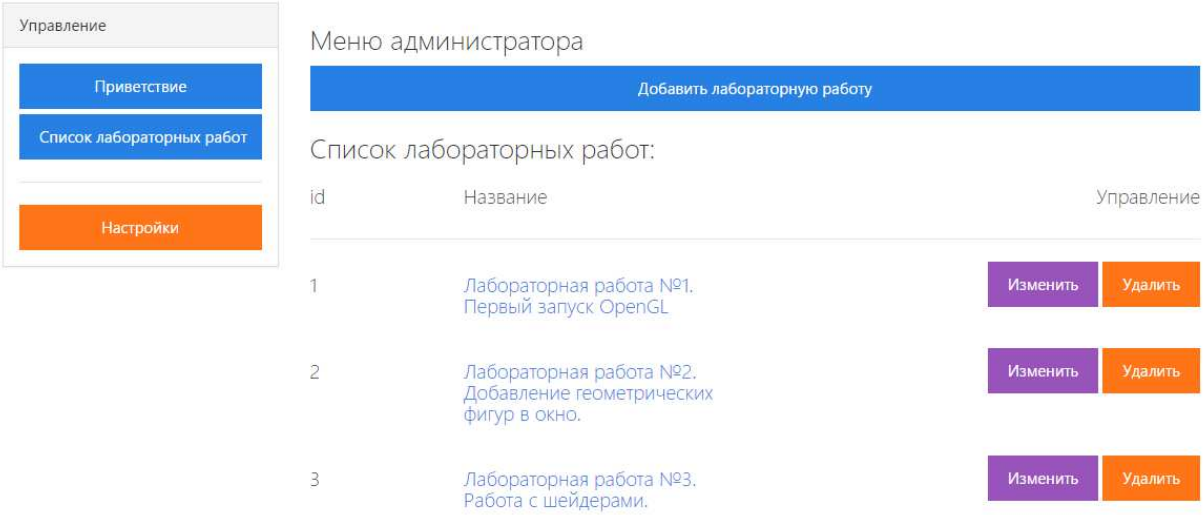


Рисунок 14 – Меню администратора

Как уже отмечалось ранее, администратор может добавить новую лабораторную работу, редактировать или удалять уже существующие лабораторные работы.

2.5.6 Разработка страницы добавления лабораторной работы

При добавлении работы необходимо заполнить её порядковый номер, название, словесное описание и код программы (рисунок 15). Для отображения данной страницы был создан файл **add.ejs**, а для работы добавления лабораторной работы в базу данных отвечает метод **router.post('/add',function(req,res)** в файле **router.js**.

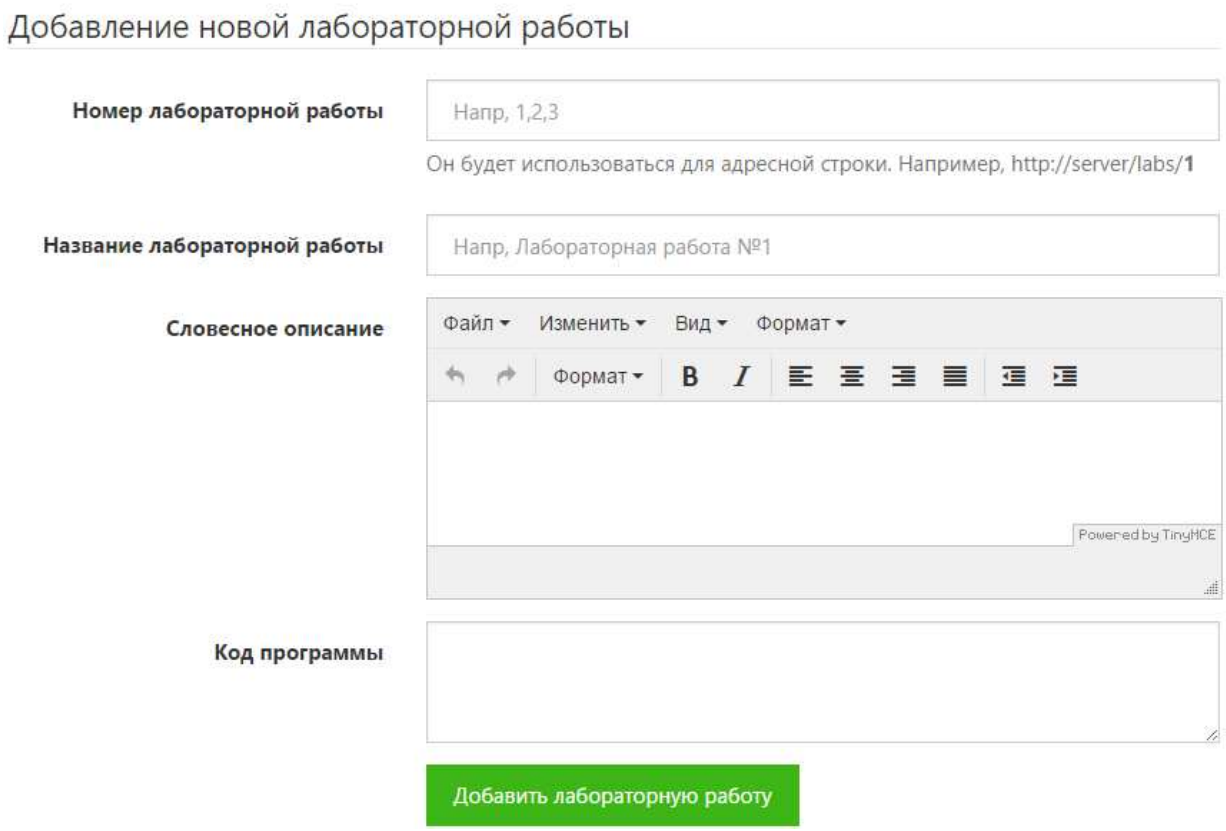


Рисунок 15 – Страница добавления лабораторной работы

2.5.7 Разработка страницы редактирования лабораторных работ

При нажатии кнопки «Изменить» напротив соответствующей лабораторной работы администратор может начать её редактирование (рисунок 16). За отображение страницы отвечает файл **edit.ejs** в папке «views/admin». Для работы обновления лабораторной работы был создан метод **router.get('/admin/edit/:id/:pass',function(req,res)**, а также метод **router.post('/admin/edit',function(req,res)** в файле **router.js**.

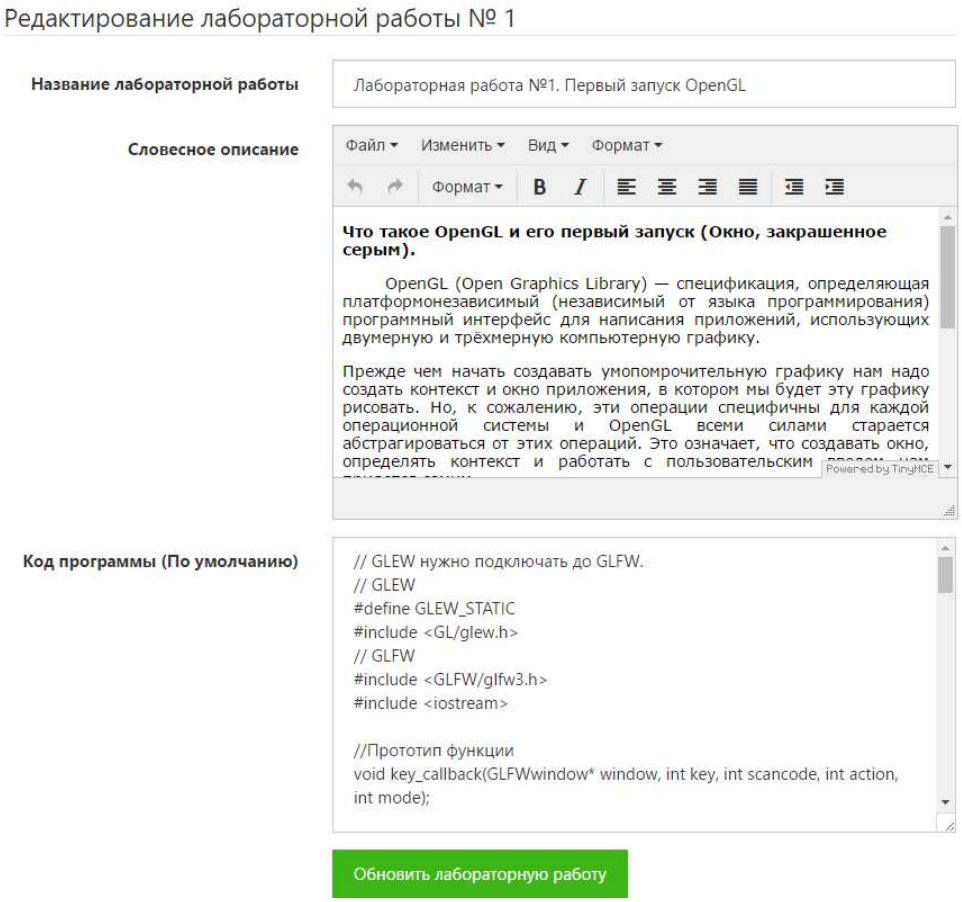


Рисунок 16 – Страница редактирования лабораторной работы

2.5.8 Удаление лабораторных работ

В меню администратора возможно удаление лабораторных работ при нажатии кнопки «Удалить». За реализацию данной функции отвечает метод **router.get('/admin/delete/:id/:pass',function(req,res)** в файле **router.js**.

2.6 Выводы по разделу

В ходе выполнения второй главы были решены технические вопросы, связанные с разработкой системы. Был описан весь функционал работы веб-приложения, включая работу с базой данных, компиляцию проектов OpenGL, а также решения базовых графических частей.

3 Руководство пользователя

Для правильной работы веб-приложения необходимо выполнить ряд действий:

– установить и запустить сервер на Node.js;

– установить и запустить базу данных;

– установить пакет программ для компиляции проектов OpenGL.

3.1 Запуск серверной части на ОС Linux

Далее приведен порядок действий для запуска сервера на ОС Linux:

– скопировать на машину пользователя папку «LearnOpenGL»;

– открыть терминал Linux;

– на системах «CentOS» Node.js необходимо установить через инструмент управления пакетами Yum с помощью команд «**curl --silent --location https://rpm.nodesource.com/setup_6.x | bash -**» и «**yum -y install nodejs**».

– на системах «Arch Linux» Node.js необходимо установить с помощью команды «**pacman –S nodejs npm**»

– на системах «Ubuntu», «Debian», «Elementary» Node.js устанавливается с помощью команды «**sudo apt-get install -y nodejs**»;

– перейти в каталог с сервером «**cd** %путь до каталога LearnOpenGL%»;

– установить все зависимости для сервера командой «**npm install**»;

– запустить сервер командой «**node app.js**»;

3.2 Запуск серверной части на ОС Windows

Далее приведен порядок действий для запуска сервера на ОС Windows:

– скопировать на машину пользователя папку «LearnOpenGL»;

– скачать Node.js с официального сайта («www.nodejs.com»);

– установить Node.js на серверную машину;

– открыть командную строку от администратора (**cmd.exe**);

– перейти в каталог с сервером командой «**cd** %путь до папки LearnOpenGL%»;

– установить все зависимости для сервера командой «**npm install**»;

– запустить сервер командой «**node app.js**», либо запустить пакетный файл **START.bat**, который находится в папке «LearnOpenGL».

3.3 Установка и запуск базы данных

Для подключения базы данных к веб-приложению нужно выполнить ряд действий:

– установить систему управления базами данных MongoDB (запустить файл **mongodb.msi**, находящийся в каталоге «LearnOpenGL/Database»);

– установить (не обязательно) графический клиент для управления базами данных Robomongo (запустить файл **robomongo.exe**, находящийся в каталоге «LearnOpenGL/Database»);

– скопировать содержимое папки «data», которая находится в каталоге «LearnOpenGL/Database» в папку установки MongoDB;

– открыть командную строку от администратора (**cmd.exe**);

– перейти в каталог с БД с помощью команды «cd %путь до папки MongoDB%»;

– ввести команду «**mongod.exe -- dbpath data**».

3.4 Установка пакета программ для компиляции проектов OpenGL

Для компиляции проектов OpenGL необходимо выполнить ряд действий:

– установить компилятор MinGW (каталог /LearnOpenGL/compiler/codeblocks-mingw.exe/);

– скопировать содержимое архивов «glew.zip», «glfw.zip», «glut.rar» (каталог /LearnOpenGL/compiler) в папку установки MinGW;

– в свойствах системы перейти на вкладку «Переменные среды» и добавить в системную переменную «path» путь до папки «MinGW/bin»;

– при использовании других инструментов для работы с OpenGL алгоритм выполнения действий аналогичен.

ЗАКЛЮЧЕНИЕ

В ходе работы были сформированы требования к системе, проанализированы средства для реализации электронного лабораторного практикума. В результате формирования требований к системе была разработана общая архитектура системы и выбрана клиент-серверная модель организации системы, что позволило снизить требования к ресурсам клиентской аппаратной части.

Также была разработана архитектура базы данных позволяющая создавать, изменять и компилировать лабораторные работы. Для связи веб-интерфейса с СУБД были разработаны соответствующие методы на языке Node.js, с помощью которых осуществляются процессы управления работами и процессы выполнения лабораторных работ.

Разработанная система позволяет осуществлять выполнение работ студентами дистанционно, без участия преподавателя. Разработанная система имеет возможность адаптации под различные дисциплины.

СПИСОК СОКРАЩЕНИЙ

OpenGL – Open Graphics Library;

СУБД – система управления базами данных;

HTML – HyperText Markup Language;

ВКР – выпускная квалификационная работа;

БД – база данных;

MS – Microsoft;

GLU – OpenGL Utility Library.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дистанционное обучение, Сибирский государственный университет имени М.Ф. Решетнева [Электронный ресурс] – Режим доступа: http://www.sibsau.ru/index.php/distantsionnoe-obuchenie (дата обращения: 20.03.2017).
2. Дистанционное обучение [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/дистанционное_обучение (дата обращения: 25.04.2017).
3. Электронный учебно-методический комплекс кафедры прикладной информатики и информационных систем Сибирского государственного университета геосистем и технологий [Электронный ресурс] – Режим доступа: http://informatics.ssga.ru/informatika/practics (дата обращения 25.04.2017).
4. Работа с OpenGL [Электронный ресурс] – Режим доступа: http://pmg.org.ru/nehe/index.html (дата обращения 25.04.2017).
5. Интерактивные онлайн-курсы «Html Academy» [Электронный ресурс] – Режим доступа: https://htmlacademy.ru/courses/4 (дата обращения 25.04.2017).
6. Обзор современных реляционных СУБД [Электронный ресурс] – Режим доступа: https://author24.ru/spravochniki/bazy_dannyh/yazyk_sql_osnovy_raboty_s_relyacionnymi_subd_osnovy_yazyka_sql/obzor_sovremennyh_relyacionnyh_subd/ (дата обращения 26.04.2017).
7. Карманович И. И. Обзор современных СУБД / И. И. Карманович, В. С. Модин – Чебоксары: ЦНС «Интерактив плюс», 2016. – 176 с.
8. Бэнкер К. MongoDB в действии / К. Бэнкер, Пер. с англ. А.А. Слинкина. – Москва: ДМК Пресс, 2012. – 394 с.
9. Флэнаган Д. JavaScript: карманный справочник / Д. Флэнаган. – Москва: ООО «И.Д. Вильямс», 2013. – 320 с.

10. Браун И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript / И. Браун. – Санкт-Петербург: Питер, 2017. – 336 с.

11. Флэнаган Д. Язык программирования Ruby / Д. Флэнаган, Ю. Мацумото. – Санкт-Петербург: Питер, 2011. – 496 с.

12. Лутц М. Python. Карманный справочник / М. Лутц – Москва: ООО «И.Д. Вильямс», 2015. – 320 с.

13. Девис Т. OpenGL. Руководство по программированию. Библиотека программиста / Т. Девис, Дж. Нейдер, Д. Шрайнер. – Санкт-Петербург: Питер, 2006. – 624 с.

ПРИЛОЖЕНИЕ

Файл router.js

Файл router.js содержит набор методов, с помощью которых происходит вся маршрутизация, а также реализуется весь необходимый функционал веб-приложения. Далее приведен программный код на языке JavaScript.

```
var express = require('express');
var router = express.Router();
//Настройки для базы данных
var MongoClient = require('mongodb').MongoClient;
var dburl = 'mongodb://localhost:27017/lablist';
var fs = require('fs');
//Переменная для выполнения команды компиляции
var exec = require('child_process').exec;
//Добавление переменной пароля администратора
var Admin = new Object();
Admin.pass = "ADMIN";
//Главная страница (приветствия)
router.get('/',function(req,res)
{
        res.render('index');
})
//Страница со списком лабораторных работ
router.get('/labs',function(req,res)
{
        //Подключение к базе данных
        MongoClient.connect(dburl, function(err, db)
        {       if (err) throw err;
        //Поиск лабораторных работ в базе данных и их сортировка по id
                var collection = db.collection('labs')
                .find({})
```

```
                .sort([['id',1]])
                .toArray(function(err,docs)
                {
                        res.render('lablist',{list:docs});
                });
        })
})
//Страница с работой
router.get('/labs/:id',function(req,res)
{
        MongoClient.connect(dburl, function(err, db)
        {
                if (err) throw err;
                var collection = db.collection('labs')
                .findOne({id:req.params.id},function(err,docs)
                {
                        res.render('lab',docs);
                });
        })
})
//Запись лабораторной работу в файл (Редактирование)
router.post('/labs/:id/edit',function(req,res)
{
        MongoClient.connect(dburl, function(err, db)
        {
                var collection = db.collection('labs');
                //Редактирование информации
                collection.updateOne({ id : req.params.id }
                , { $set: { code : req.body.code } }, function(err, result)
                {
                        if (err) throw err;
                        res.redirect('/labs/'+req.params.id);
                });
        });
})
```

//Восстановление кода по-умолчанию

```
router.get('/labs/:id/default',function(req,res)
{
	MongoClient.connect(dburl, function(err, db)
	{
		var collection = db.collection('labs')
```

/*Замена кода, отредактированного пользователем (code) на код по умолчанию (defcode).*/

```
		.findOne({id:req.params.id},function(err,docs)
		{
			var collection2 = db.collection('labs')
			collection2.updateOne({ id : req.params.id }
			, { $set: { code : docs.defcode } }, function(err, result)
			{
				if (err) throw err;
				res.redirect('/labs/'+req.params.id);
			});
		});
	});
})
```

//Парсинг файла настроек компилятора

```
router.get('/options',function(req,res)
{
	var data = JSON.parse(fs.readFileSync('settings.json', 'utf8'));
	res.render('options',data);
})
```

//Сохранение настроек компилятора

```
router.post('/saveoptions',function(req,res)
{
	var data =
	{
		path:req.body.path,
		params:req.body.params
	}
```

```
        saved = fs.writeFileSync('settings.json', JSON.stringify(data), 'utf8');
        res.redirect('/options');
})
//Компиляция проекта
router.get('/run/:id',function(req,res)
{
        //Настройки компиляции
        //Загрузка параметров компиляции из settings.json
        var compile = JSON.parse(fs.readFileSync('settings.json', 'utf8'));
        //Подключение к БД с лабораторными работами
        MongoClient.connect(dburl, function(err, db)
        {
                var collection = db.collection('labs')
                .findOne({id:req.params.id},function(err,docs)
                {
                        /*Запись кода лабораторной работы в файл
                lastsource.cpp*/
                        saved = fs.writeFileSync('compile/lastsource.cpp', docs.code, 'utf8');
                        console.log('СОХРАНЕНО');
                        //Формирование команды компиляции
                        var cmd = compile.path + "compile/lastsource.cpp" +
                        compile.params+" && a.exe";
                        //Выполнение команды компиляции
                         exec(cmd, function(error, stdout, stderr)
                        {
                                //Вывод лога ошибок при их наличии
                                if (error) res.render('compilelog',{err:stderr,command:cmd});
                                else res.render('success');
                        });
                });
         });
})
//Вход в панель администратора
```

```
router.post('/admin',function(req,res)
{
        //Проверка пароля
         if (req.body.pass == Admin.pass)
         /*Подключение к базе данных и вывод отсортированного списка лабораторных работ*/
        MongoClient.connect(dburl, function(err, db)
        {
                 if (err) throw err;
                var collection = db.collection('labs')
                .find({})
                .sort([['id',1]])
                .toArray(function(err,docs)
                {
                        docs.pass = req.body.pass;
                        res.render('admin/index',{list:docs});
                });
        })
        else res.redirect('/options');
})
//Страница добавления лабораторной работы (проверка пароля)
router.get('/add/:pass',function(req,res)
{
        if (req.params.pass != Admin.pass)
        res.redirect('/');
        res.render('add');
})

//Добавление лабораторной работы
router.post('/add',function(req,res)
{
        //Информация, которую будем передавать в базу данных
        var data =
        {
```

```
            id: req.body.id,
            name: req.body.name,
            code: req.body.code,
            defcode: req.body.code,
            descr: req.body.descr
        }
        //Добавление новой лабораторной работы в БД
        MongoClient.connect(dburl, function(err, db) {
        if (err) throw err;
        var collection = db.collection('labs')
        collection.insertOne(data,function(err,result)
        {
            if (err) throw err;
            res.redirect('/labs/'+req.body.id);
        });
        })
})
//Изменение лабораторной работы администратором
router.get('/admin/edit/:id/:pass',function(req,res)
{
        //Защита от несанкционированного доступа
        if (req.params.pass != Admin.pass)
        res.redirect('/');
        MongoClient.connect(dburl, function(err, db)
        {
            if (err) throw err;
            var collection = db.collection('labs')
            collection.findOne({id:req.params.id},function(err,result)
            {
                if (err) throw err;
                res.render('admin/edit',result);
            });
        })
});
```

```
//Обновление изменения лабораторной работы
router.post('/admin/edit',function(req,res)
{
        MongoClient.connect(dburl, function(err, db)
        {
                var collection = db.collection('labs');
                collection.updateOne({ id : req.body.id }
                , { $set: { code : req.body.code, name: req.body.name, descr:req.body.descr }
                }, function(err, result)
                {
                        if (err) throw err;
                        res.redirect('/labs/'+req.body.id);
                });
        });
})
//Удаление лабораторной работы
router.get('/admin/delete/:id/:pass',function(req,res)
{       //Защита от несанкционированного доступа
         if (req.params.pass != Admin.pass)
        res.redirect('/');
        MongoClient.connect(dburl,function(err,db)
        {
                var collection = db.collection('labs');
                collection.deleteOne({ id : req.params.id }, function(err, result)
                {res.redirect('/options');});
   })
})

module.exports = router;
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Легалов
подпись инициалы, фамилия
«14» 06 2017 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специализации

Программное обеспечение лабораторных работ по дисциплине «Высокопроизводительные вычисления на графическом процессоре»
тема

Пояснительная записка

Руководитель	09.06.17	доцент, к.т.н	Ю.В. Удалова
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	09.06.17		И.С. Сироткин
	подпись, дата		инициалы, фамилия
Нормоконтролер	14.06.17	доцент, к.т.н	В. И. Иванов
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2017