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

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. И. Легалов
подпись инициалы, фамилия
« ____» _____ 2016 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01.09 – «Технологии разработки программного обеспечения»
код и наименование направления

Разработка, компиляция и выполнение программ на языке C++
с использованием Web интерфейса
тема

Пояснительная записка

Руководитель	_____	проф., д.т.н.	А. И. Легалов
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		А. В. Марудов
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____		В. И. Иванов
	подпись, дата		инициалы, фамилия

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мы живем в век стремительного развития информационных технологий, когда каждый студент в карманах и сумке имеет несколько устройств, обладающих вычислительной мощностью, которая даже не снилась создателям первых суперкомпьютеров. Но в какое русло мы направляем всю эту мощность?

Сидя на лекции, очень сложно сконцентрироваться и слушать несколько часов напролет речь преподавателя. Поэтому студенты засыпают или достают смартфоны, планшеты и тратят производительность своего устройства на развлечения, общение друг с другом или пролистывание новостной ленты в социальных сетях. Но если бы была возможность объединить прослушивание теоретического материала с интерактивным закреплением его на практике, с помощью неких инструментальных средств, используя подручные устройства, студенты почувствовали бы вовлеченность в процесс обучения и с большим интересом усваивали преподаваемый материал.

Рассмотрим принцип работы системы на простом примере проведения контрольной работы. Преподаватель пишет задание, не на доске, откуда плохо видно и постоянно требуются фломастеры или мел, а в системе. Задание отображается на экранах устройств у студентов. Обучающиеся пишут решение данного задания, тут же имея возможность проверить работоспособность своего решения и отправить его преподавателю. Преподаватель в свою очередь отслеживает успешность выполнения задания каждым студентом.

Использование такой системы на начальных этапах обучения программированию позволило бы уйти от написания программного кода на бумажных носителях и вести более продуктивный диалог между студентом и преподавателем во время лекций. Сделать проведение контрольных и тестов более гибкими и простыми, как для преподавателя, так и для студента.

Целью выпускной квалификационной работы является создание системы, которая бы дополнила проведение учебного процесса интерактивностью. Для преподавателя это возможность давать задания студентам и контролировать их

успеваемость онлайн. А для студента это возможность выполнения заданий преподавателя, заключающееся в написании программного кода и его последующей компиляции и отправке преподавателю.

Для достижения поставленной цели в работе решались следующие задачи:

– анализ существующих решений;
– на основе анализа существующих решений выделить функционал, необходимый для реализации цели;
– поиск возможного решения с открытым кодом для последующей доработки;
– разработка оболочки пользователя, обеспечивающей выполнение требуемых функций;
– разработка серверной части;
– разработка методов взаимодействия оболочки пользователя с сервером;
– разработка демонстрационного примера.

1 Анализ предметной области

Реализуемая система должна внести удобство в проведение лекций по программированию, как для преподавателя, так и для студента, путем их интерактивного взаимодействия, которое более подробно будет рассмотрено ниже.

1.1 Возможности преподавателя в системе

Преподаватель в системе имеет свое рабочее пространство, в котором ему предоставляется доступ к списку задач в виде заголовков, включающих номер задания и его название, все задачи хранятся на сервере в базе данных. Выбрав задачу из списка, открывается ее условие, шаблон кода, предоставляемый студентам, время, отведенное на выполнение данной задачи, а также кнопка запуска задания на выполнение студентами. После нажатия на кнопку для студентов открывается доступ к получению задания, а у преподавателя отображается окно текущей задачи с запущенным таймером обратного отсчета, кнопкой экстренного обнуления таймера и кнопкой демонстрации правильного решения данной задачи.

После обнуления таймера выдается сообщение в виде модального окна о том, что время на выполнение задачи истекло, и преподаватель вновь возвращается к выбору задач для возможного продолжения процесса.

Создание заданий в базе данных и регистрация пользователей осуществляется с помощью интерфейса доступного преподавателю из его рабочего пространства. Данный подход избавит от использования сторонних СУБД для редактирования базы данных.

В ходе работы преподавателя ведется журнал, в который записывается выдача заданий и вариант их завершения, что обеспечивает дополнительный контроль учебного процесса. Просмотреть данный журнал преподаватель

может в любой момент из своего рабочего пространства.

1.2 Возможности студента в системе

Вход в систему осуществляется после получения студентом адреса страницы с идентификацией. Для входа используется логин и пароль, которые идентифицируют студента в системе, что в дальнейшем позволит связать с ним процесс выполнения заданий.

После входа в систему отображается окно с заданием и кнопкой, при нажатии на которую студент запустит выполнение данного задания, если преподаватель не установил задание, кнопка будет не активна. Нажав на кнопку выполнения задания, студент переходит в свое рабочее пространство. Рабочее пространство студента содержит окно с условием задачи, окно с запущенным таймером обратного отсчета, окно редактирования программного кода, окно исходных данных, окно вывода программы, кнопка отправки задания на компиляцию и выполнение, кнопка завершения выполнения задания и кнопка отказа от выполнения задания.

По истечении таймера появится модальное окно, информирующее о завершении времени выделенного на задание и перенаправит студента к начальному окну ожидания задания. Время, выделенное на задание, указывается преподавателем индивидуально для каждой задачи, в зависимости от ее сложности.

Окно редактирования программного кода выполнено с подсветкой синтаксиса языка, в окне может содержаться шаблон кода, прикрепленный к выполняемому заданию.

Окно исходных данных, окно вывода программы, кнопка компиляции и выполнения задания, при нажатии на которую результат компиляции и выполнения программы отобразится в окне вывода программы, необходимы студенту для тестирования работоспособности программы. Данная процедура

может повторяться многократно до завершения отладки программы или до истечения времени, отведенного на выполнение задания.

Если студент уверен, что задание выполнено правильно, он нажимает кнопку завершения задания. После этого выдается модальное сообщение о завершении задании с указанием правильности его выполнения, закрыв которое студент автоматически переходи в окно ожидания следующего задания.

Находясь в окне ожидания задания, студент может выйти из системы, нажав кнопку выхода.

В ходе работы студента ведется журнал, в котором фиксируется ход выполнения заданий.

1.3 Обработка действий пользователя

Действия, выполняемые преподавателем и студентами, обрабатываются системой, которая реализуется на основе веб-сервера, развернутого на компьютере преподавателя. Процессы системы можно разбить на две группы:

– Процессы, поддерживающие работу преподавателя.

– Процессы, поддерживающие работу студента.

1.3.1 Поддержка работы студента

При запуске задания на выполнение система формирует запись в базе данных содержащую идентификатор студента, задачи и время получения задания.

При отправке решения на компиляцию формируется новая запись об успешности компиляции, содержащая идентификатор студента, задачи и переменную, отвечающую за успешность компиляция программы. Подробная информация о результатах компиляции отправляется на страницу студента.

При отказе от выполнения задания, пришедшего от студента, система формирует соответствующую запись в базе данных и передает клиенту окно ожидания следующего задания.

В случае получения сигнала о завершении задания система прогоняет полученную программу с тестовыми данными преподавателя, сравнивая каждый раз результат работы с эталонным результатом. При прохождении всех тестов формируется информация об успешном прохождении задания, которая передается студенту. Если хотя бы один из тестов не проходит, студент получает информацию о неудаче. Результаты тестирования заносятся в базу данных студента и могут в дальнейшем использоваться для анализа и получения статистических данных. После отображения сообщения о результате завершения в модальном окне сервер переводит клиента в состояние ожидания следующего задания.

1.3.2 Поддержка работы преподавателя

При выборе задания преподавателем сервер передает на соответствующее рабочее место список заданий и их описания. Выбор задания сопровождается порождением и выдачей на рабочее место преподавателя окна с условием задания и запущенным таймером обратного отчета. Сервер принимает сигнал о запуске таймера на странице преподавателя и дублирует его на странице студента. При обнулении таймера сервер перенаправляет пользователей на их первоначальные страницы, у студента это страница ожидания задания, а у преподавателя страница выбора задания из списка.

При создании задания, сервер осуществляет связь с базой данных и вносит в нее новый элемент.

Ведение журнала о действиях преподавателя осуществляется сервером, через связь с базой данных и занесении в нее отчетов о его действиях.

1.4 Анализ существующих решений

Для того чтобы система не уступала аналогам в удобстве интерфейса и быстродействии, необходимо провести анализ существующих решений, частично или полностью реализующих необходимый нам функционал, и выявить минусы и плюсы данных решений.

1.4.1 Интегрированная среда разработки

Рабочее пространство студента по функционалу имеет сходство с интегрированной средой разработки.

Из существующих онлайн ИСР, были выделены несколько веб-приложений: SourceLair, codingground, codepad, ideone, sonny lab. Основными критериями отбора служат эргономика пользовательского интерфейса и полная функциональность. Необходимым набором функций для реализации нашей цели служат: текстовый редактор, для ввода программного кода, желательно с подсветкой синтаксиса C++, окно предназначенное для ввода входных данных, а также окно отвечающее за консольный вывод результата компиляции введенного программного кода.

Первым на очереди стоит SourceLair [4], функционал данного сервиса превосходит наши требования и ориентирован на опытных разработчиков. Данная ИСР позволяет работать с несколькими проектами одновременно, является образцом эргономичности, поддерживает не так много языков и, к сожалению, в этом списке нет C++.

Без внимания нельзя оставить веб-приложение Codingground [5], поддерживающий большое количество языков, и при этом довольно простой и понятный в использовании. Для начинающего программиста на сайте имеется обучающий материал по каждому языку программирования.

Следующим на очереди стоит веб-приложение Codepad [6]. Сервис

привлекает простотой оформления, но отталкивает отсутствием необходимого функционала, а именно: отсутствие окна отвечающего за ввод входных данных, отсутствие подсветки синтаксиса, что значительно усложняет написание программного кода.

Неплохим функционалом обладает веб-приложение Ideone [7], здесь есть и подсветка синтаксиса, и все необходимые окна. Стоит выделить довольно удобную особенность, зарегистрировавшись, открывается доступ к сохранению программного кода на сервере, а также к скачиванию файла кода.

Единственным веб-приложением с MIT лицензией, позволяющей использовать код, является ИСР от компании Sonnylab [8]. Пользовательский интерфейс эргономичен, присутствует подсветка синтаксиса, нумерация строк программного кода, что способствует удобству при редактировании, поиске и устранении ошибок. Строка, на которой в данный момент находится курсор, подсвечивается, данная функция не даст потерять место, на котором вы закончили работу. В текстовом редакторе программного кода поддерживаются горячие клавиши работы с текстом. Размер окон можно изменять, что очень удобно при написании объемного программного кода. Минусом веб-приложения является невозможность использования в процессе обучения из-за недостатка функционала.

На основе проведенного анализа, было принято решение об интеграции продукта от компании Sonnylab в реализуемую среду обучения.

1.4.2 Сервис для онлайн-обучения

Цели, которые преследует реализация данной системы, схожи с курсами онлайн-обучения языкам программирования.

Яркими представителями подобных курсов являются GeekBrains [19], Code Avengers [20] и Stepic [21].

Перечисленные сервисы имеют набор бесплатных уроков, по нескольким изучаемым курсам.

Большое количество курсов доступно для прохождения на GeekBrains, по завершению которых, обучающемуся выдается сертификат о прохождении курса и возможность прохождения практики в IT компании. Занятия проходят в виде трансляций уроков через интернет, после которых обучающийся получает домашние задания.

Особенностью Code Avengers является представление обучающего материала в играющей форме, в рамках бесплатного курса ученик конструирует свою мини игру. Изменяя код, мы сразу можем увидеть результат его работы, что очень удобно в подобных системах.

Stepic - сервис, предоставляющий возможность зарегистрированным пользователям конструировать и добавлять курсы. Благодаря этой возможности доступно большое количество бесплатных курсов по различным предметам. Значительным плюсом является предоставление данных курсов, как на английском, так и русском языке. Минусом сервиса является предоставление курсов в определенное время, что доставляет неудобство людям, совмещающим работу и обучение.

2 Архитектура разработанной системы

2.1 Общая архитектура приложения

Система реализована в виде трехуровневой архитектурной модели.

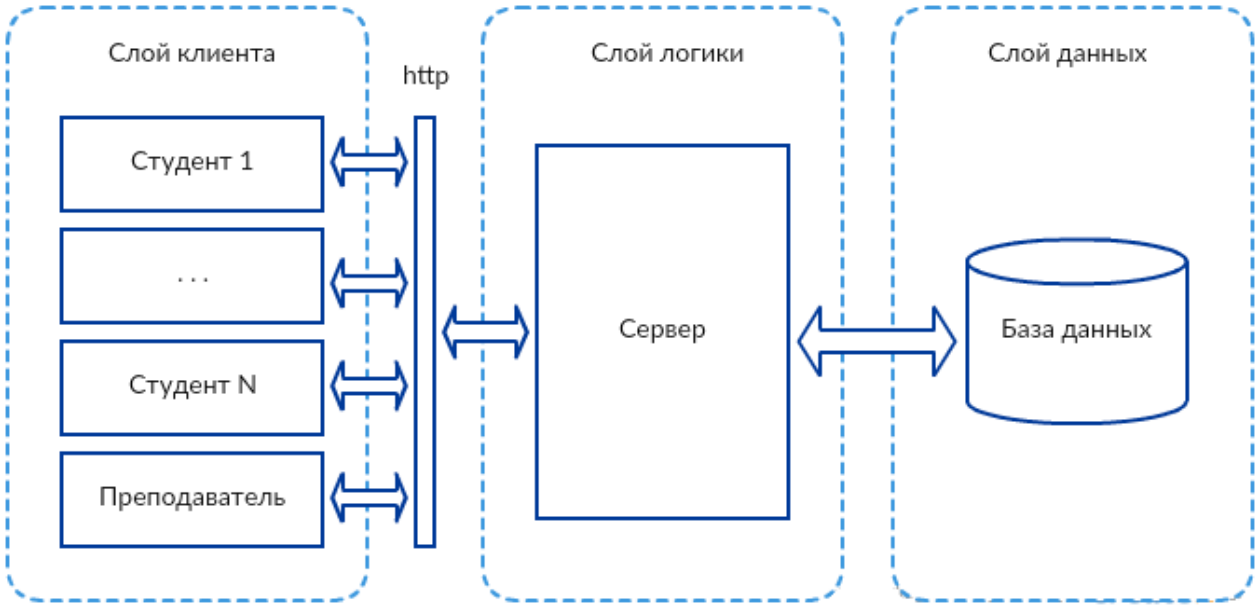

Рисунок 2.1 – Структура приложения

Слой клиента – та часть, которая доступна пользователю. Слой клиента должен быть легким для восприятия. Главная задача слоя обеспечить пользователю доступ к функционалу системы. Слой работает в среде любого веб-браузера.

Слой логики – то, что скрыто от пользователя, отвечает за корректную работу функционала, обрабатывает запросы пользователя, работает с базой данных. Слой реализован в виде веб-сервера.

Слой данных – необходим для хранения информации. Информация корректируется, добавляется через слой логики.

2.2 Средства разработки

Для разработки клиентской части системы использован язык разметки HTML5 [11] в связке с CSS3 [12]. Использование данных средств сделает страницы системы легковесными, что значительно ускорит обработку их браузером. Еще одним плюсом является адаптивный дизайн, что позволит страницам нормально отображаться на любом устройстве

Серверная часть выполнена на базе программной платформы Node.js [9], что обеспечивает высокую производительность системы и простоту реализации серверной части.

В качестве базы данных используется SQLite [10]. Преимуществами данной БД является высокая производительность и надежность. В node имеются встроенные модули для работы с данной базой данных, что сделает взаимодействии с сервером в разы проще.

2.3 Описание прецедентов

Пользователи системы делятся на две категории - студенты и преподаватель. Такое разделение необходимо для скрытия части прецедентов от студентов во избежание непредвиденных ситуаций.

Диаграмма на рисунке 2.3.1 демонстрирует иерархию актеров взаимодействующих с системой.

Функциональные возможности системы приведены в виде диаграмм прецедентов на рисунке 2.3.2 и 2.3.3

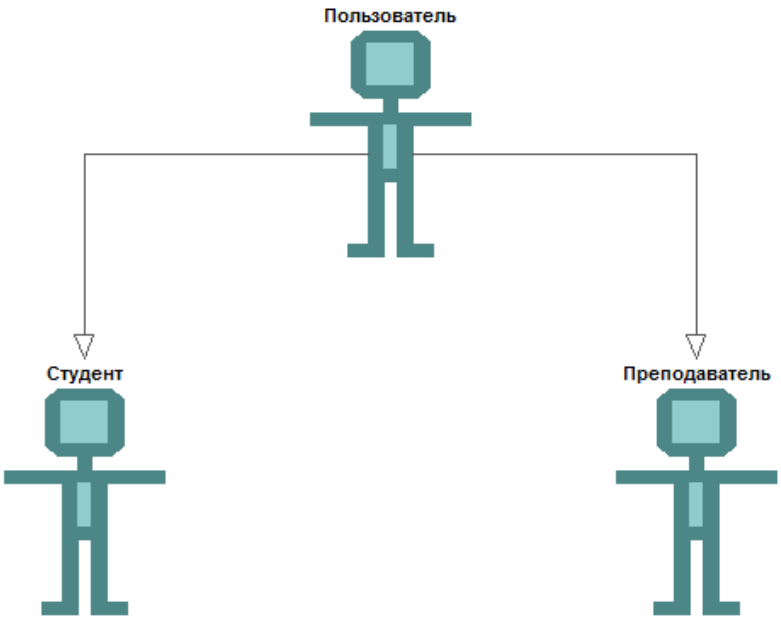

Рисунок 2.3.1 – Диаграмма иерархии актеров

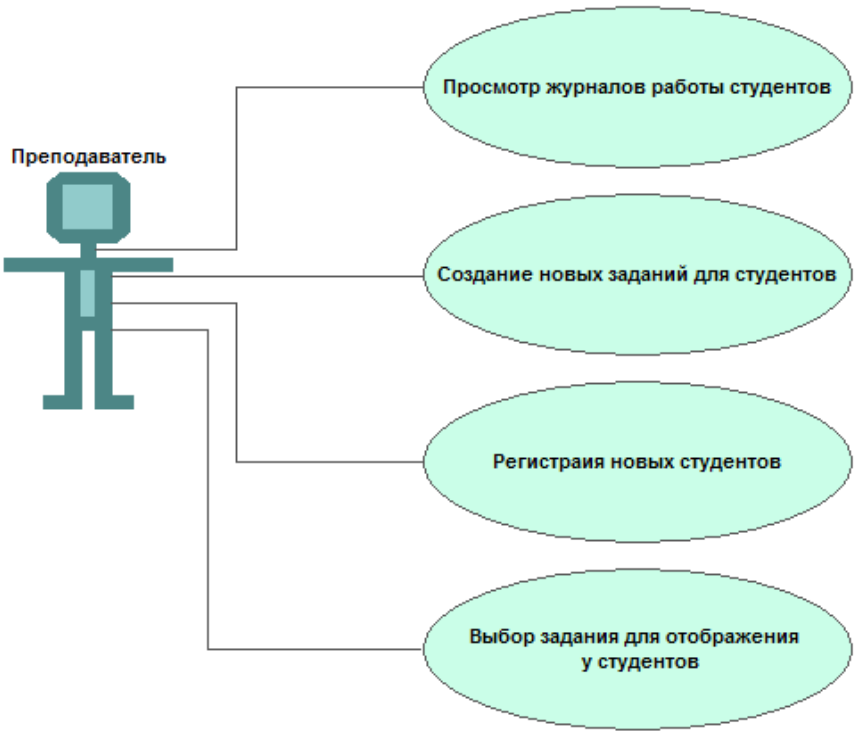

Рисунок 2.3.2 – Диаграмма прецедентов актера преподаватель

Прецедент «Просмотр файлов студента» доступен только преподавателю, используется для контроля успеваемости студентов путем просмотра файлов, отправленных ими.

Прецедент «Создание задания для студентов» необходим преподавателю для создания и отправки задания студентам.

Прецедент «Регистрация новых студентов» необходим преподавателю для добавления студентов как категории пользователей системы.

Прецедент «Выбор задания для отображения у студентов» позволяет преподавателю выбрать задание из списка имеющихся и предоставить студентам для дальнейшего решения.

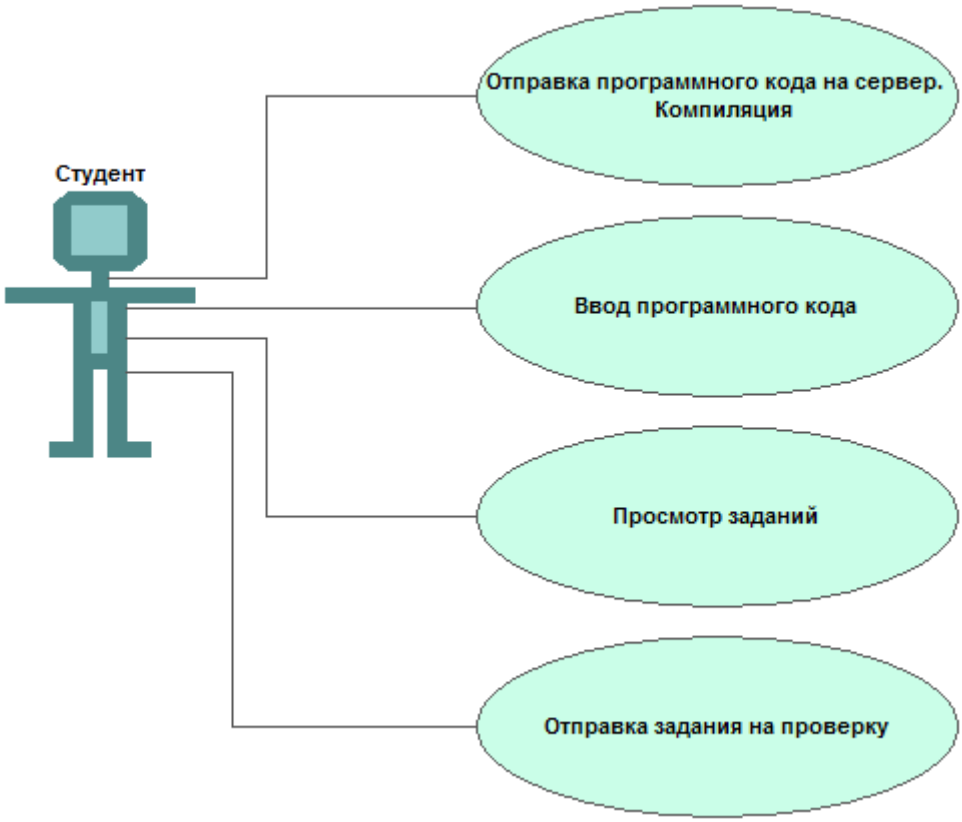

Рисунок 2.3.3 – Диаграмма прецедентов актера студент

Прецедент «Ввод программного кода» позволяет пользователям использовать редактор программного кода с подсветкой синтаксиса языка C++.

Прецедент «Отправка программного кода на сервер. Компиляция» необходим для вывода результата компиляции пользователю.

Прецедент «Просмотр заданий», необходим студенту для получения задания от преподавателя.

Прецедент «Отправка задания на проверку» позволит студенту произвести отправку решения задания системе на проверку правильности работы данного решения. Проверка правильности работы осуществляется в виде подмены ввода программы на тесты, хранящиеся в системе, с последующим сравнением выходных данных с теми же тестами.

2.4 Модульная структура приложения

Одним из плюсов node.js является обилие подключаемых модулей, благодаря которым не приходится писать сервер с нуля.

Установка модулей осуществляется с помощью стандартного менеджера пакетов npm (аббр. node package manager) [13].

Модули, которые используются в написании системы, указываются в файле «package.json».

После установки модули хранятся в папке «node_modules». Подключение модулей осуществляется функцией «require(‘Название модуля’)».

Express [14] – модуль упрощающий операции по созданию сервера, работе с запросами и маршрутизацией.

Passport-local [15] – модуль, который значительно упрощает реализацию аутентификации пользователей по логину и паролю.

Socket.io [16] – модуль, реализующий коммуникацию сервера и клиента в реальном времени.

Cloudcmd [4] – модуль, добавляющий возможность использования сервера в роли облачного файлового менеджера.

Bcrypt [17] – модуль, осуществляющий шифровку пароля пользователя.

SQLite3[18] – модуль для работы с базой данных SQLite.

2.5 Список функций для взаимодействия слоя логики и слоя данных

Sequelize – обеспечивает подключение к базе данных SQLite.

sync – создает таблицу в базе данных.

register – создает пользователя в базе данных.

logout – завершает сессию пользователя.

create – создает запись в таблице БД.

2.6 Список функций для взаимодействия слоя логики и слоя клиента

get() / post() – обрабатывают запросы клиентской части, выполняя действия прописанные для данных запросов.

redirect() – перенаправляет пользователя на другой адрес.

render() – выдает клиенту визуальное представление страницы.

2.7 Данные используемые системой

Файлы, формируемые при отправке пользователем решения задачи, имеют формат имени «Логин пользователя + Номер попытки» и расширение «cpp». Содержимое файлов - программный код, введенный пользователем на языке C++ .

Database.sqlite – база данных, содержит в себе таблицу пользователей, зарегистрированных в системе, таблицу заданий, добавленных преподавателем, таблицу тестов для проверки правильности ответов пользователей на задания, таблицу ответов на задания для просмотра статистики выполнения заданий пользователями. Полная структура таблиц базы данных приведена в Приложении Б.

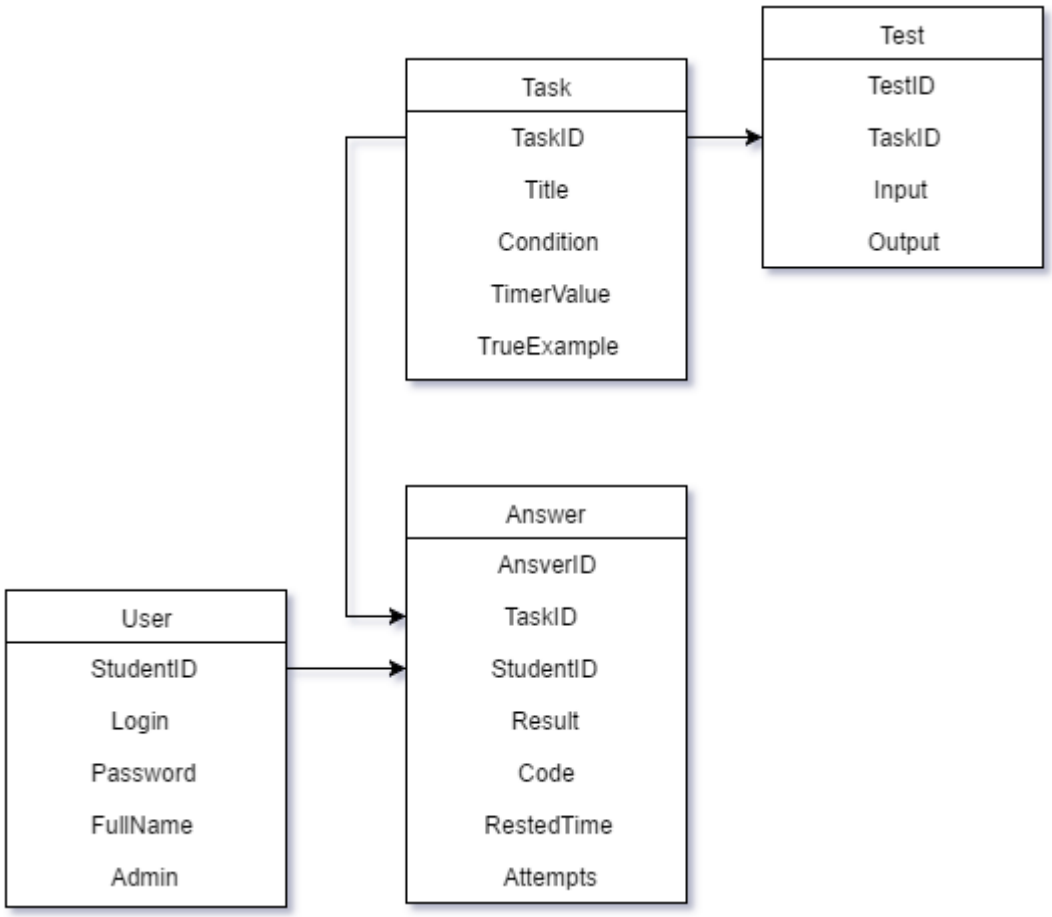

Рисунок 2.8.1 – Структура части таблиц реализуемых в базе данных

3 Реализация разрабатываемой системы

3.1 Реализация взаимодействия слоя логики и слоя данных

Ниже представлена реализация работы с базой данных SQLite.

```
var mydb = new Sequelize('database', 'username', 'password', {
  dialect: 'sqlite',
  storage: 'database.sqlite'
});
```

}); - подключение к базе данных SQLite.

User.sync(); - создание таблицы пользователей в базе данных.

```
User.register("ALegalov", "12345", function(err, user) {
 if (err) return;
  user.set('admin', true);
  user.save();
});
```

}); - функция создания пользователя преподаватель с правами администратора, где «ALegalov» - логин, «12345» - пароль.

```
var Task = mydb.define('task', {
      title: Sequelize.STRING,
                  description: Sequelize.TEXT,
                  deadline: Sequelize.INTEGER,
                  input: Sequelize.INTEGER,
                  output: Sequelize.INTEGER

});
```

}); - описание структуры таблицы заданий в БД

3.2 Реализация взаимодействия слоя логики и слоя клиента

Для того что бы слой логики корректно отвечал на запросы пользователей к конкретному адресу необходима реализация маршрутизации.

Маршрутизация осуществляется с помощью функций get и post.

```
app.get('/login',
 function(req, res) {
 if (req.isAuthenticated()) {
        if (req.user.admin) {
           res.redirect('/admin');
           return;
        }
        res.redirect('/compile');
        return;
  }
 res.render("login.html");
```

}) – функция обрабатывает запрос пользователя по адресу «/login». Пользователь, который еще не вошел в систему, получает страницу «login.html». Если пользователь уже произвел вход, то сервер перенаправляет его в свое рабочее пространство, а именно, студент направляется по адресу «/compile», преподаватель по адресу «/admin».

```
app.get('/compile',
        function(req, res) {
      if (!req.isAuthenticated()) {
        res.redirect('/login');
        return;
```

```
    }
  res.render('index.html, { user: req.user.username, task: task });
```

}); - функция обрабатывает запрос пользователя по адресу «/compile». Если пользователь прошел аутентификацию, то выдает ему страницу «index.html», иначе возвращает на страницу ввода логина и пароля.

```
app.post('/register', function(req, res) {
  var user = req.body.username;
  var account = req.body.password;
  User.register(user, account, function(error, user){
    if (error) {
      res.redirect('/register');
      return;
    }
  });
```

}); – функция получает введенные данные по адресу «/register», на основании которых добавляет нового пользователя в базу данных.

```
app.get('/admin',
      function(req, res) {
    if (!req.isAuthenticated()) {
      res.redirect('/login');
      return;
    }
  if (user.admin) {
res.render('admin.html', { user: req.user.username, task: task });
                  return;
              };
```

res.redirect ('/ compile'');

}); - функция обрабатывает запрос пользователя по адресу «/admin». Если пользователь – преподаватель, то ему выдается страница «admin.html».

```
app.use(function( req, res, next) {
        res.status(404).render("404.html");
        next();
```

}); - функция обрабатывает запрос пользователя по неизвестному для системы адресу и выдает страницу ошибки «404.html».

3.3 Разработка клиентской части

Внешний вид системы – первое, на что обратит внимание пользователь, поэтому стоит подойти к реализации клиентской части со всей ответственностью.

Для внешнего представления пользователям полного функционала системы, необходимо создать 4 страниц: страница входа пользователей, страница рабочего пространство студента, рабочего пространства преподавателя и страница, выдаваемая пользователю при возникновении ошибки работы системы.

Для того чтобы избежать использования стороннего ПО для редактирования базы данных, было решено создать окно для добавления задания. Реализована кнопка, при нажатии на которую, данные со страницы передаются на сервер и записываются в БД.

```
<h3>Создание задания</h3>
<textarea name="TaskTitle" id="TaskTitle" cols="1" rows="1"
placeholder="Заголовок"></textarea>
<textarea name="Task" id="Task" cols="30" rows="25"
```

placeholder="Условие"></textarea>

<textarea name="TaskTime" id="TaskTime" cols="1" rows="1" placeholder="Время на выполнение"></textarea>

<h3>Формирование теста</h3>

<textarea name="TaskInput" id="TaskInput" cols="1" rows="1" placeholder="Входные данные"></textarea>

<textarea name="TaskOutput" id="TaskOutput" cols="1" rows="1" placeholder="Вывод программы"></textarea>

<button class="btn" type="submit">Добавить задание</button>

Для возможности выбора заданий из базы данных для отправки студентам, было реализовано выпадающее окно со списком имеющихся заданий.

<h3>Список заданий</h3>

<select id = "select-task" name ="TaskTitle"required>

<option>Выберите задание из списка</option>

<% for (var i = 0; i < data.length; i++) { %>

<option name = <%= i%>><%= data[i].id %>.

<%= data[i].title %></option>

<% } %>

</select>

Для ограничения выполнения задания, реализован таймер обратного отсчета, по истечении которого блокируется окно отправки программного кода. Функция таймера реализована с помощью javascript, для того чтобы значение таймера изменялось в фоновом режиме, не обновляя активной страницы пользователя.

<script>

function updateTimer() {

$.getJSON("/timer", function(data) {

```
                console.log(data);
                var min = Math.floor(data.timer / 1000 / 60);
                var sec = Math.floor((data.timer - min * 1000 * 60) / 1000);
                if (!data.timer || data.timer < 0) {
                        $("#timer").text("Время закончилось");
                } else {
                        if(sec>=10){
                                $("#timer").text(min + " : " + sec );
                        }
                        else{
                                $("#timer").text(min + " : 0" + sec );
                        }
                }
        });
}
$(document).ready(function() {
window.setInterval(function() { updateTimer() }, 1000);
        $("#resetTimer").click(function() {
                $.getJSON("/timer/reset/");
        });
});
</script>
```

Для того чтобы применить стили CSS к шаблону страницы html, необходимо сначала их подключить или описать непосредственно в шаблоне. Подключение стилей осуществляется кодом, представленным ниже.

```
<head>
        <meta charset="UTF-8">
        <title>Рабочее пространство преподавателя</title>
        <link rel="stylesheet" href="/styles/compiler/grid.css">
        <link rel="stylesheet" href="/styles/compiler/style.css">
        <script src="/scripts/jquery.js"></script>
         <style>
</head>
```

3.4 Структура каталогов и файлов системы

Ниже представлена древовидная структура проекта. Папка «views» содержит html-шаблоны слоя клиента, отвечающие за визуальное представление системы. Папка «node_modules» содержит модули, используемые для функционирования слоя логики. В «public» хранятся стили CSS, изображения и шрифты, используемые страницами. Файлы, отправляемые студентами, хранятся в папке «users», отдельно для каждого студента. В корне проекта находятся файлы обеспечивающие корректную работу сервера это «server.js», «child.js» и «child-win.js». Остальные файлы: база данных database.sqlite и pacage.json, хранящий записи о модулях, которые используются системой для корректного функционирования.

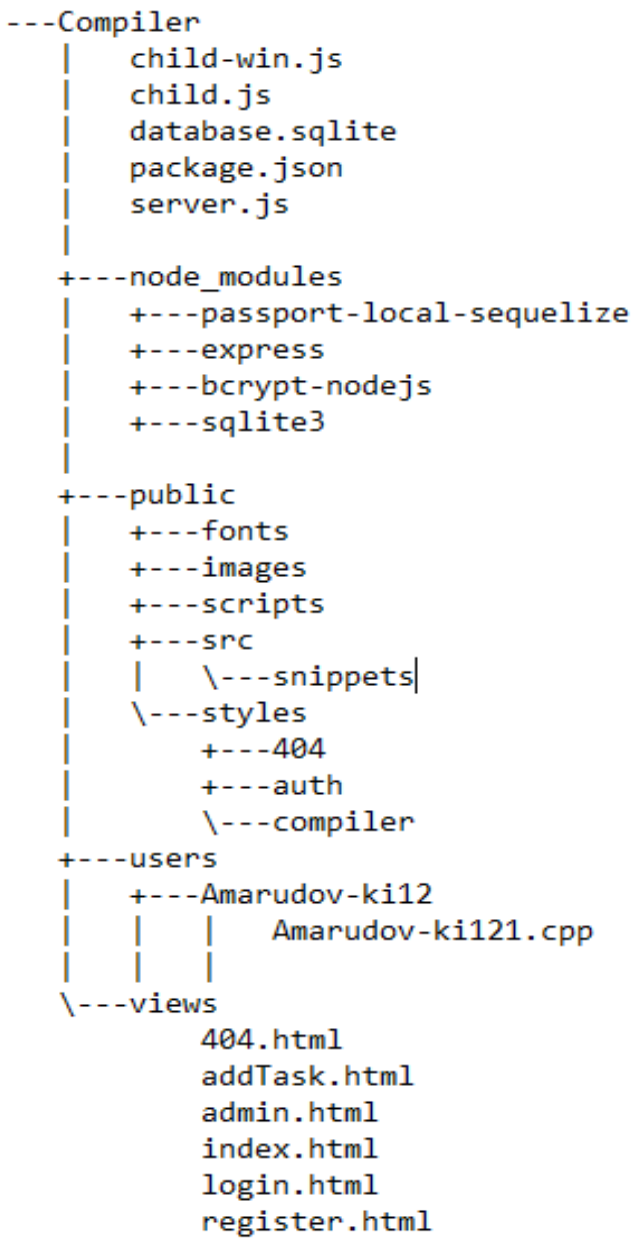

Рисунок 3.4.1 – Древовидная структура каталогов и файлов системы

4 Функционирование системы

4.1 Вход пользователя в систему

Для того, чтобы начать работу с системой пользователю необходимо ввести в адресной строке браузера адрес: «localhost:3000» (ввиду того, что разработка велась на тестовом сервере Node, отсутствует домен второго и первого уровней). После перехода по данному адресу система вызовет выполнение функции, отвечающей за отображение страницы аутентификации (рисунок 4.1).

Рисунок 4.1.1 – Страница аутентификации пользователя

После верного заполнения полей логина и пароля, система определяет права доступа пользователя. Если пользователем является студент, система перенаправит его на страницу «/compile» (рисунок 4.2), если пользователем является преподаватель, система перенаправит его на страницу «/admin» (рисунок 4.3).

4.2 Рабочее пространство студента

После того как студент вошел в систему он попадает на страницу «/compile» (рисунок 4.2).

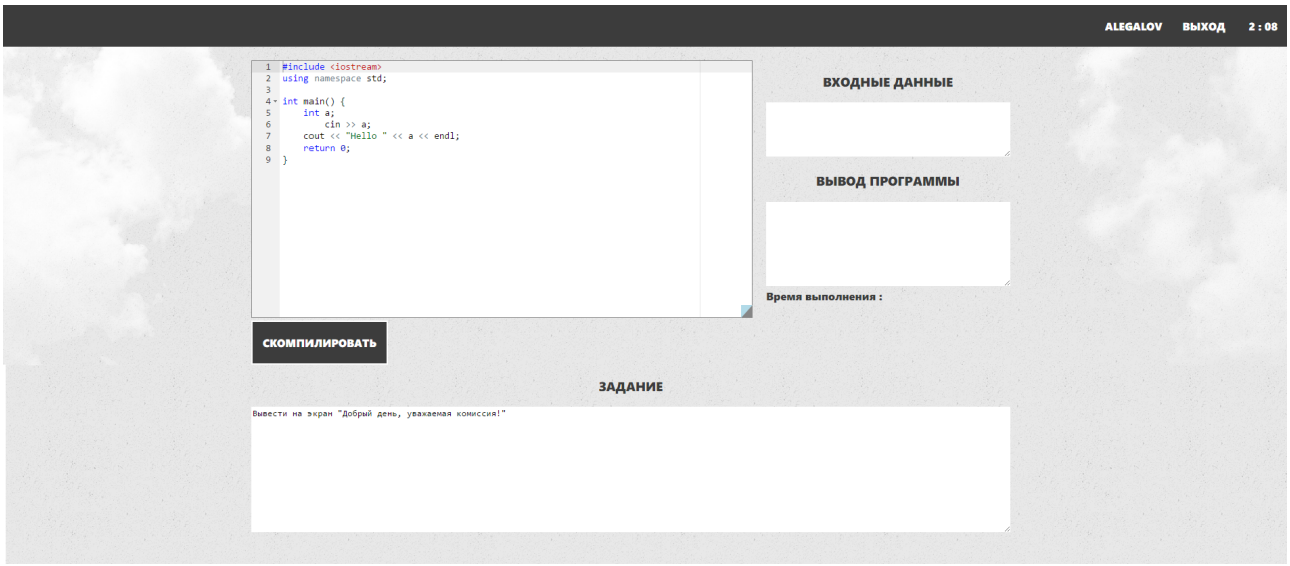

Рисунок 4.2.1 – Страница рабочего пространства студента «/compile»

Переходя на страницу «/compile», пользователь получает доступ к таким функциям системы как: ввод программного кода (функция реализована в виде окна ввода с подсвечиванием синтаксиса языка C++, расположенного над кнопкой «Скомпилировать»), просмотр задания от преподавателя (функция реализована в виде окна вывода, расположенного под кнопкой «Скомпилировать»), отправка программного кода на сервер и компиляция (функция реализована в виде кнопки «Скомпилировать», расположенной между окном ввода программного кода и окном вывода задания).

После того как пользователь ввел программный код и нажал кнопку «Скомпилировать», система произвела компиляцию программного кода и вывела результат в окно «Вывод программы».

При необходимости использования входных данных в написанном программном коде, необходимо ввести данные в поле «Входные данные».

Возможность отправлять попытки решения заданий продолжаются до тех пор, пока таймер обратного отсчета, который находится в верхнем правом углу, не обнулится.

4.3 Рабочее пространство преподавателя

После того как преподаватель вошел в систему он попадает на страницу «/admin» (рисунок 4.3).

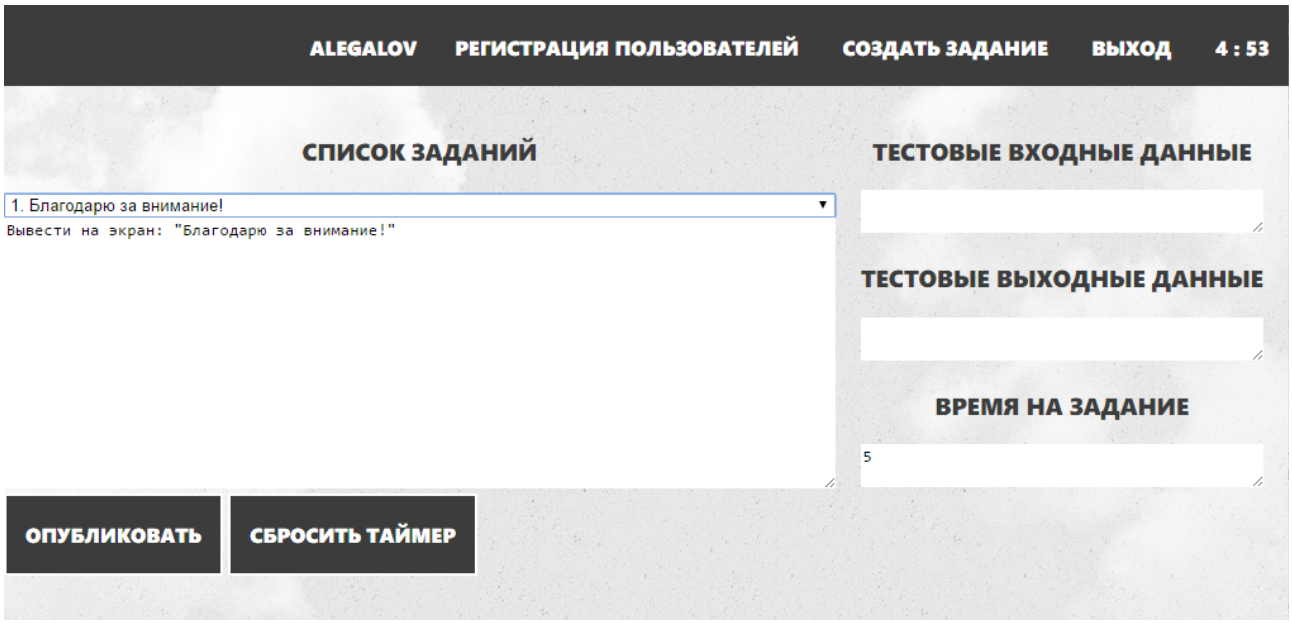

Рисунок 4.3 – Страница рабочего пространства преподавателя «/admin»

Переходя на страницу «/admin», преподавателю открывается возможность выбора задания для студентов (реализовано в виде выпадающего окна). Рассылка задания пользователям и запуск таймера происходят при нажатии на кнопку «Опубликовать».

Кнопка «Сбросить таймер» отвечает за обнуление таймера обратного отсчета, что ведет к завершению выполнения задания. Таймер расположен в верхнем правом углу экрана, как и на странице рабочего пространства студента

Для добавления пользователей в систему преподавателю необходимо перейти по нажатию кнопки «Регистрация пользователей» на страницу «/register» (рисунок 4.4).

Реализована возможность не используя сторонние СУБД добавить новое задание в базу данных SQL, для этого необходимо перейти по адресу «/addtask», нажав на кнопку «Создать задание» (рисунок 4.5).

Благодаря реализованной шапке страницы, легко вернуться обратно в рабочее пространство преподавателя, нажав на отображение своего логина.

Поля «Тестовые входные данные» и «Тестовые выходные данные» необходимы для создания теста, проверяющего корректность работы программного кода студента.

4.4 Регистрация пользователей

На рисунке 4.4 представлена страница добавления пользователя с последующей записью в БД.

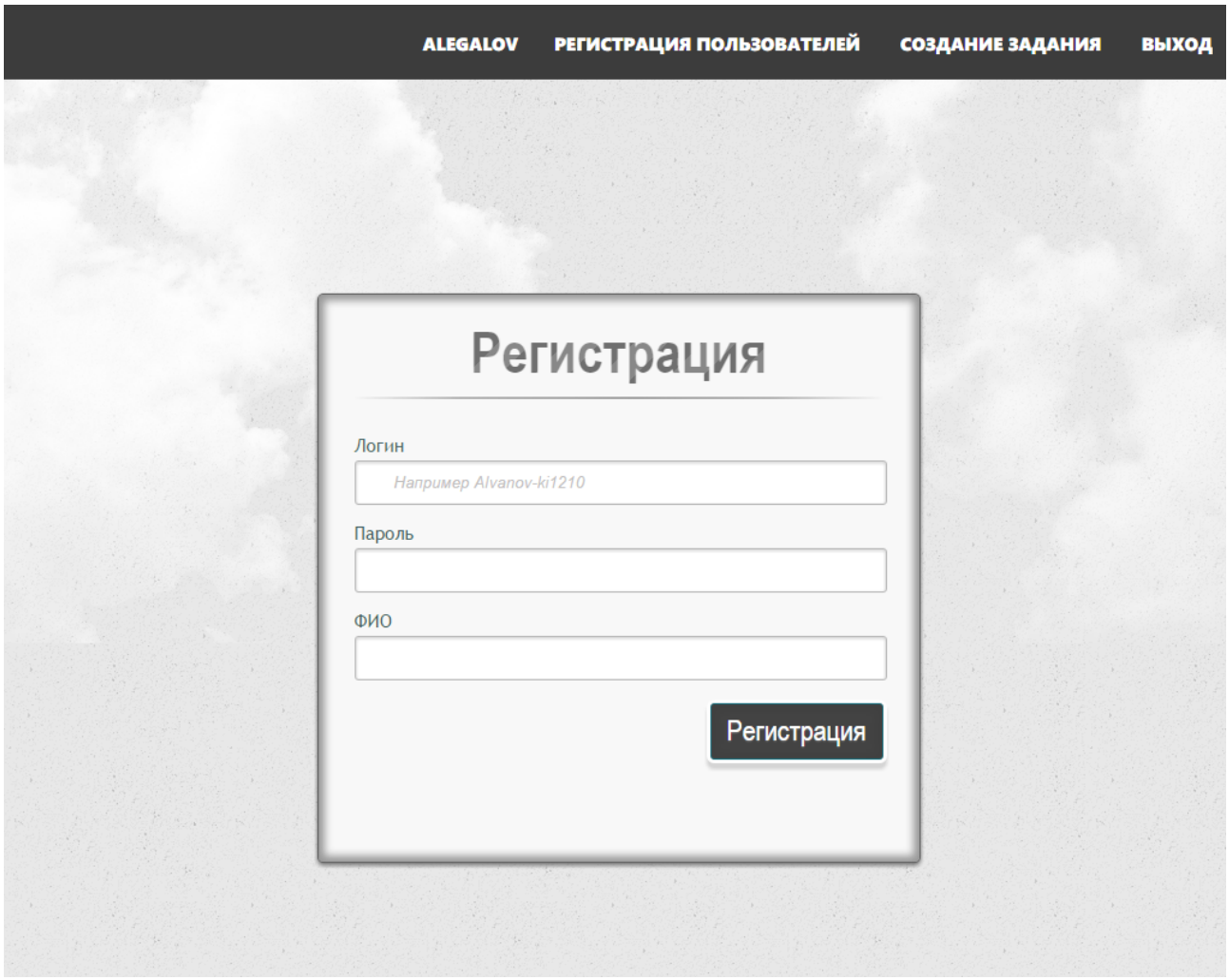

Рисунок 4.4 – Создание пользователя «/register»

4.5 Создание задания

Указав заголовок, условие задачи и время, отведенное на выполнение данной задачи, преподаватель добавит задание в базу данных.

Для проверки системой отправляемых заданий, необходимо указать эталон входных данных и данных вывода программы.

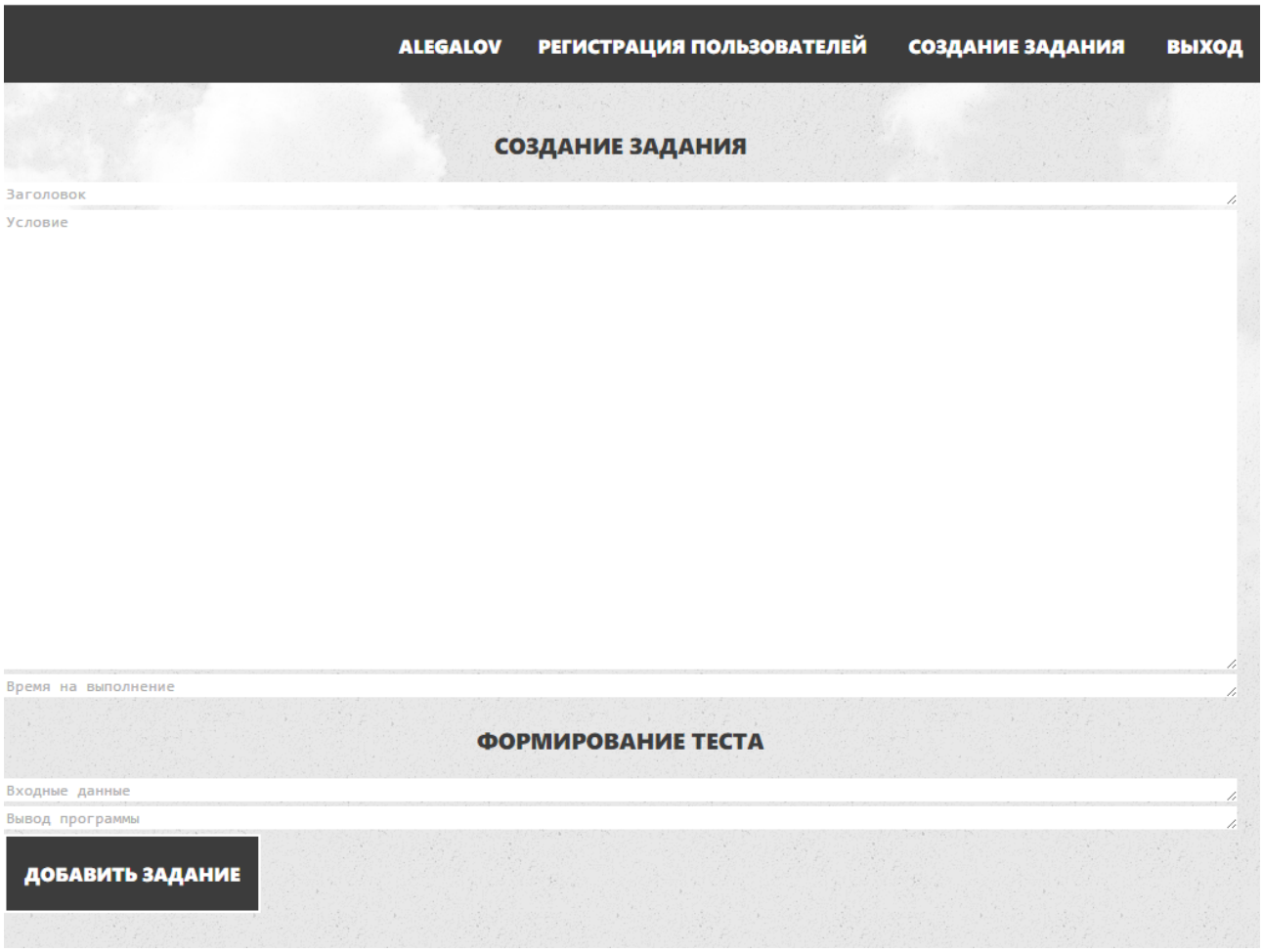

Рисунок 4.5 – Создание задания «/addtask»

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы создана система, которая дополняет проведение учебного процесса функционалом, обеспечивающим интерактивное общение студента и преподавателя.

Для достижения поставленной цели в ходе выполнения работы были решены следующие задачи:

- проанализированы существующие решения;
- на основе анализа существующих решений выделен функционал, необходимый для реализации цели;
- разработана оболочка пользователя, обеспечивающая выполнение требуемых функций
- разработана серверная часть, обрабатывающая действия пользователей в системе;
- разработан метод взаимодействия оболочки пользователя с сервером;
- разработан демонстрационный пример работы системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Наливкин А. В. Электроника и микропроцессорная техника [Электронный ресурс]: Электронный учебник. Петербургский Институт точной механики и оптики — Режим доступа: http://de.ifmo.ru/bk_netra/page.php?dir=1&tutindex=25&index=83&layer=1

2. Игнатенко В. Онлайн компиляторы Си и C++ [Электронный ресурс]: Техно старец. Обзор сервисов Онлайн компиляторов Си и C++ — Режим доступа: https://www.techold.ru/review/services/online-compiler/

3. Manuel Kiessling. The Node Beginner Book [Электронный ресурс]: перевод книги на русский язык: ARTOD — Режим доступа: http://nodebeginner.ru/#javascript-and-nodejs

4. SourceLair. Frictionless development in your browser [Электронный ресурс]: Develop software from any device using Python, Node.js, PHP, HTML5 and more. — Режим доступа: https://www.sourcelair.com/home

5. Codingground. Compile and execute c++ online [Электронный ресурс]: Онлайн компилятор — Режим доступа: http://www.tutorialspoint.com/compile_cpp_online.php

6. Codepad. [Электронный ресурс]: Онлайн компилятор — Режим доступа: http://codepad.org

7. Ideone. [Электронный ресурс]: Онлайн компилятор — Режим доступа: http://ideone.com

8. SonnyLab. [Электронный ресурс]: Онлайн компилятор — Режим доступа: http://sonnylab.com/api/compiler

9. Node.js. [Электронный ресурс]: About Node.js — Режим доступа: https://nodejs.org/en/about/

10. SQLite. [Электронный ресурс]: About SQLite — Режим доступа: http://www.sqlite.org/about.html

11. WHATWG, W3C. [Электронный ресурс]: HTML Living Standart —

Режим доступа: https://html.spec.whatwg.org/multipage/

12. W3C. [Электронный ресурс]: — Режим доступа: https://www.w3.org

13. NPM. [Электронный ресурс]: Package manager for JavaScript — Режим доступа: https://www.npmjs.com

14. Express. [Электронный ресурс]: Быстрый, гибкий, минималистичный веб-фреймворк для приложений Node.js — Режим доступа: http://expressjs.com/ru/

15. Passport. [Электронный ресурс]: Simple, unobtrusive authentication for Node.js — Режим доступа: http://passportjs.org

16. Socket.io. [Электронный ресурс]: FEATURING THE FASTEST AND MOST RELIABLE REAL-TIME ENGINE— Режим доступа: http://socket.io

17. Npmjs. Bcrypt. [Электронный ресурс]: A bcrypt library for NodeJS. — Режим доступа: https://www.npmjs.com/package/bcrypt

18. Npmjs. SQLite3. [Электронный ресурс]: Asynchronous, non-blocking SQLite3 bindings. — Режим доступа: https://www.npmjs.com/package/sqlite3

19. GeekBrains. [Электронный ресурс]: Основы программирования — Режим доступа: https://geekbrains.ru

20. Code Avengers. [Электронный ресурс]: Learn to build websites, apps and games in a fun and effective way. — Режим доступа: https://www.codeavengers.com

21. Stepic. [Электронный ресурс]: Открыт для знаний. Онлайн-конструктор уроков. Платформа для открытых курсов. Инструмент для распространения образовательных материалов.— Режим доступа https://stepic.org.

ПРИЛОЖЕНИЕ А

Программный код, реализующий рабочее пространство преподавателя

```
<!doctype html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>Рабочее пространство преподавателя</title>
        <link rel="stylesheet" href="/styles/compiler/grid.css">
        <link rel="stylesheet" href="/styles/compiler/style.css">
        <script src="/scripts/jquery.js"></script>
          <style>
  select {
   width: 100%;
  }
 </style>

</head>
<body>

        <div id="menu">
                <div class="container">
                        </div>
                        <ul class="menu">

                        <% if (user) { %>
                                <li><a href="/login"><%= user %></a><li>
                        <% } %>
                                <li><a href="/register">Регистрация пользователей
                                </a></li>
                                <li><a href="/addTask">Создать задание </a></li>
                                <li><a href="/logout">Выход </a></li>
                                <li><a id="timer"></a></li>
```

```
                    </ul>
                </div>
                <div class="article">
                    <div class="grid">
                        <form method="post" id="myCode" action="/admin">

                            <div class="col-2-3">
                                <h3>Список заданий</h3>
                                <select id = "selecttask" name = "TaskTitle"
required><option >Выберите задание</option>
                                    <% for (var i = 0; i < data.length; i++) {
%>
                                        <option value=" <%= i%>">
                                            <%= data[i].id %>. <%=
data[i].title %>
                                        </option>
                                    <% } %>
                                </select>

                                <textarea  name="description"
id="description" cols="15" rows="14"></textarea>
                            </div>

                            <div class="col-1-3">
                                <h3>Тестовые входные данные</h3>
                                <textarea id = "taskInput" name="taskInput"
cols="3" rows="2"></textarea>
                                <h3>Тестовые выходные данные</h3>
                                <textarea id = "taskOutput" name="taskOutput"
cols="3" rows="2"></textarea>
                            </div>

                            <div class="col-1-3">
                                <h3>Время на задание</h3>
                                <textarea name="timerInput" id="timerInput"
```

```
cols="1" rows="2"></textarea>
                                    </div>
                                    <div class="col-11-12">
                                        <button class="btn"
type="submit">Опубликовать</button>
                                        <button class="btn" id="resetTimer">Сбросить
таймер</button>
                                        <div id="timer"></div>
                                    </div>
                                </form>
                            </div>

                        </div>
                        <script>
                        var tasks = <%- JSON.stringify(data) %>;
                        console.log(tasks);

                        function updateTimer() {
                            $.getJSON("/timer", function(data) {
                                var min = Math.floor(data.timer / 1000 / 60);
                                var sec = Math.floor((data.timer - min * 1000 * 60) / 1000);

                                if (!data.timer || data.timer < 0) {
                                    $("#timer").text("Время закончилось");
                                } else {
                                    if(sec>=10){
                                        $("#timer").text(min + " : " + sec );
                                    }
                                    else{
                                        $("#timer").text(min + " : 0" + sec );
                                    }
                                }
                            });
                        }
                        $(document).ready(function() {
```

```
            window.setInterval(function() { updateTimer() }, 1000);
            $("#resetTimer").click(function() {
                $.getJSON("/timer/reset/");
            });

            $("#selecttask").change(function() {
                var selected = tasks[parseInt(this.value)];
                //console.log(selected, this.value);
                $("#description").text(selected.description);
                $("#timerInput").val(selected.deadline);
            });
        });

        </script>
        </body>
</html>
```

ПРИЛОЖЕНИЕ Б

Структура базы данных системы