Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра прикладной математики

РАБОТА ПРОВЕРЕНА
Рецензент,
_____ /Сартасов Е.М./
«27» 06 2016 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____ Л.А.Прокудина
«18» 06 2016 г.

Разработка сайта для поддержки трудоустройства студентов
ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–231000.2016.075.ПЗ ВКР

Руководитель работы, доцент кафедры прикладной математики
_____ / А.К. Демидов
«18» июня 2016 г.
Автор работы
Студент группы ММиКН-474
_____ / А.В. Беляев
«18» июня 2016 г.
Нормоконтролер, доцент кафедры прикладной математики, к.ф.-м.н.
_____ / С.У. Турлакова
«18» июня 2016 г.

Челябинск 2016

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Факультет математики, механики и компьютерных наук
Кафедра прикладной математики
Направление подготовки Программная инженерия

УТВЕРЖДАЮ
Заведующий кафедрой
_____ (Прокудина)
_____ 2015 г.

З А Д А Н И Е

на выпускную квалификационную работу студента
Беляева Арсения Викторовича
Группа ММиКН-474

1. **Тема работы** Разработка сайта для поддержки трудоустройства студентов утверждена приказом по университету от « 15 » апреля 2016 г. № 661
2. **Срок сдачи студентом законченной работы** « 1» **июня 2016 г.**
3. **Исходные данные к работе**
   3.1. Язык программирования PHP.
   3.2. Язык разметки HTML, CSS.
   3.3. Примеры запросов работодателей к уровню подготовки студентов.
4. **Перечень вопросов, подлежащих разработке**
   4.1. Обзор существующих систем для поиска вакансий и резюме соискателей.
   4.2. Разработка пользовательского интерфейса.
   4.3. Проектирование базы данных.
   4.4. Проектирование и разработка программы.
   4.5. Оценка результатов экспериментального внедрения программы.
   4.6. Разработка программной документации.
5. **Иллюстративный материал** (плакаты, альбомы, раздаточный материал, макеты, электронные носители и др.)
   5.1. Схема структуры базы данных. Демонстрационный плакат – 1 л.
   5.2. Мультимедийная презентация – 11 слайдов.
   Общее количество иллюстраций – 12.
6. **Дата выдачи задания** «1» октября 2015 г.

Руководитель _____ /А.К. Демидов/
(подпись)

Задание принял к исполнению _____ /А.В. Беляев/
(подпись)

7. Календарный план

Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Отметка о выполнении руководителя
1. Обзор литературы и постановка задачи	01.10.15 – 01.01.16	
2. Разработка пользовательского интерфейса	01.02.16 – 08.02.16	
3. Проектирование базы данных	09.02.15 – 16.02.16	
4. Проектирование и разработка алгоритма	17.02.16 – 17.04.16	
5. Отладка и тестирование программы	18.04.16 – 25.04.16	
6. Подготовка пояснительной записки дипломной работы	26.04.16 – 15.05.16	
7. Разработка программной документации	01.05.16 – 15.05.16	
8. Проверка работы руководителем, исправление замечаний	16.05.16 – 19.05.16	
9. Нормоконтроль	20.05.16 – 26.05.16	
10. Подготовка иллюстративного материала и доклада	23.05.16 – 28.05.16	
11. Рецензирование, представление зав. кафедрой	10.06.16 – 18.06.16	

Заведующий кафедрой _____ (подпись) / Прокудина Н.Ю. /

Руководитель работы _____ (подпись) / А.К. Демидов /

Студент _____ (подпись) / А.В. Беляев /

АННОТАЦИЯ

Беляев А.В. Разработка сайта для поддержки трудоустройства студентов.– Челябинск: ЮУрГУ, ММиКН-474, 170 с., 41 ил.,8 табл., библиогр. список – 10 наим., 1 прил., 1 л. плакатов ф. А1.

В работе проанализированы существующие информационные системы трудоустройства. С учётом их плюсов и минусов была сформулирована задача разработки веб-сайта для поддержки трудоустройства студентов. В соответствии с поставленной задачей была создана структура базы данных, разработан интерфейс, в котором наглядно представлены возможные варианты выбора и запрос формируется с помощью перетаскивания. При проектировании архитектуры систему использовался подход MVC. Также в работе описывается реализация методов и функций системы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Информационные технологии играют значимую роль во всех сферах нашей жизни и позволяют автоматизировать и упростить выполнение многих задач. Для чего раньше надо было прилагать массу усилий и тратить много времени, сейчас можно сделать в несколько кликов мышкой.

Эти изменения коснулись и сферы поиска работы. Раньше соискателям приходилось просматривать множество газет в поисках объявлений о работе, или же обзванивать различные фирмы, узнавая про наличие вакансий. Теперь же достаточно просто зайти на специальный сайт и просмотреть предлагаемые вакансии и связаться с работодателем через сайт или напрямую. При этом есть возможность фильтровать список вакансий по нужным вам параметрам, например, по зарплате или по району, в котором находится офис.

Такие сайты упрощают задачу поиска сотрудников для работодателя. Зайдя на сайт, работодатель может достаточно быстро просмотреть большое количество анкет и затем уже сам связаться с лучшими, по его мнению, кандидатами, это очень удобно, если нужно как можно быстрее найти человека на появившуюся вакансию. Для упрощения и ускорения поиска работодатель может фильтровать резюме.

Такие сайты для поиска работы уже существуют в большом количестве. Они обладают как плюсами, так и минусами.

Цель работы:

Создать веб-сайт для поиска вакансий и резюме. Система должна обеспечивать регистрацию работодателя, добавление резюме и вакансий, связь работодателя и соискателя работы, отображение списка вакансий и резюме, их фильтрацию. Требуется обеспечить администратору возможность регистрации группы студентов, контроля над зарегистрированными пользователями и добавляемыми резюме и вакансиями.

Для достижения данной цели необходимо решить следующие задачи:

1) Проектирование базы данных, в которой будут храниться данные о пользователях, вакансиях и резюме. Реализация в СУБД.
2) Разработка интерфейса сайта.
3) Разработка архитектуры сайта в соответствии с шаблоном MVC.
4) Отладка и тестирование сайта.

В качестве языка программирования необходимо использовать PHP.

1. ОБЗОР СУЩЕСТВУЮЩИХ ИНФОРМАЦИОННЫХ СИСТЕМ ТРУДОУСТРОЙСТВА

1.1. Сайт Работа74.ру

Сайт [1] предоставляет возможность размещать вакансии или резюме, а на их основе искать себе сотрудника или работодателя (рисунок 1.1).

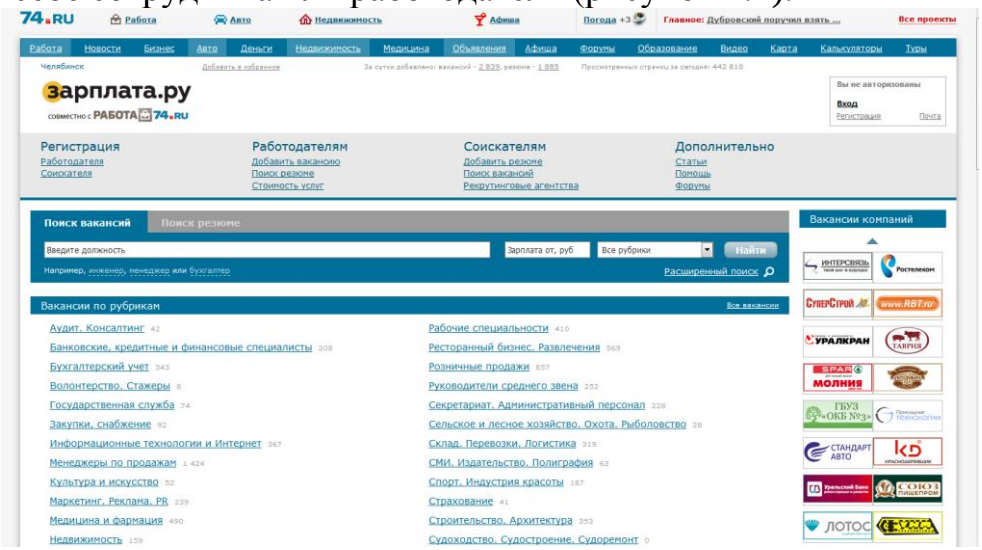

Рисунок 1.1 – Общий вид сайта Работа74.ру

Для размещения вакансии или резюме требуется регистрация на сайте. При размещении вакансии можно загрузить информацию о компании, логотип компании, информацию о предоставляемой работе, требования к соискателю, условия работы, контактную информацию и дополнительные файлы. При размещении резюме можно загрузить информацию о себе, фотографию, информацию о своих навыках, о предыдущем месте работы, о пожеланиях к работе, контактную информацию, а также дополнительные файлы (рисунок 1.2).

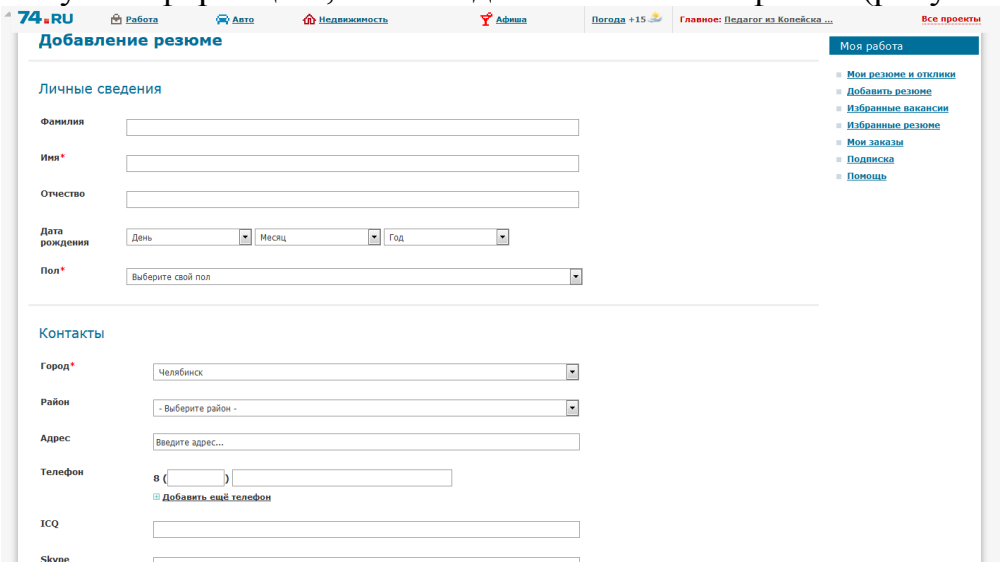

Рисунок 1.2 – Форма добавления резюме

Работа74.ру предоставляет возможность по продвижению вакансии или резюме вверх списка, за определенную плату. Для просмотра контактной информации соискателя, нужно быть зарегистрированным на сайте

работодателем. Вакансии и резюме разбиты по рубрикам, по которым их можно просматривать (рисунок 1.3).

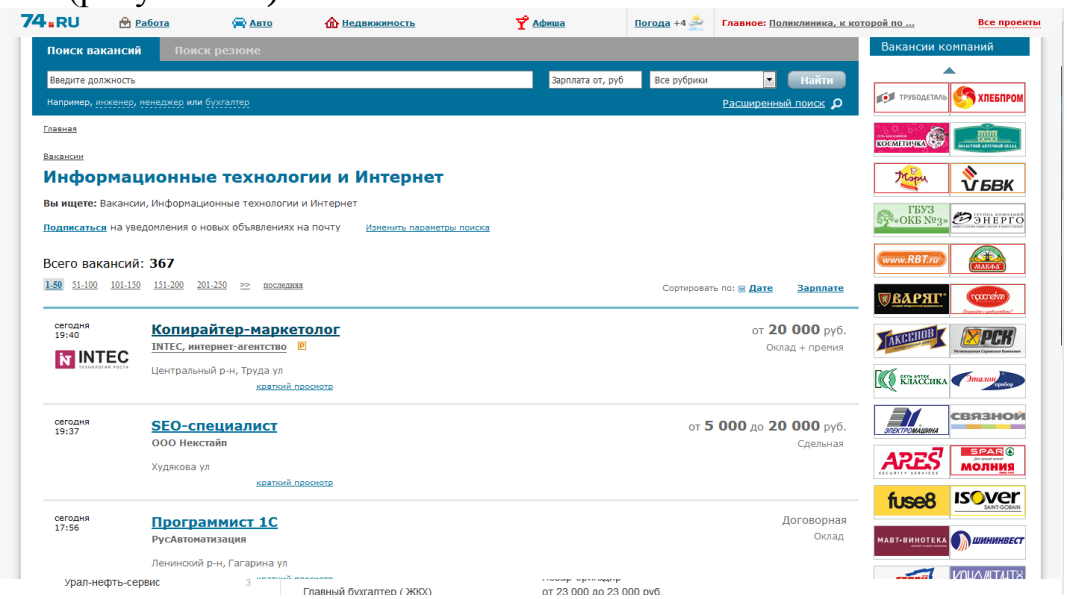

Рисунок 1.3 – Список вакансий из категории «информационные технологии и интернет» на сайте Работа74.ру

1.2. Сайт hh.ru

Сайт [2] является одним из крупнейших сайтов для поиска работы (рисунок 1.4).

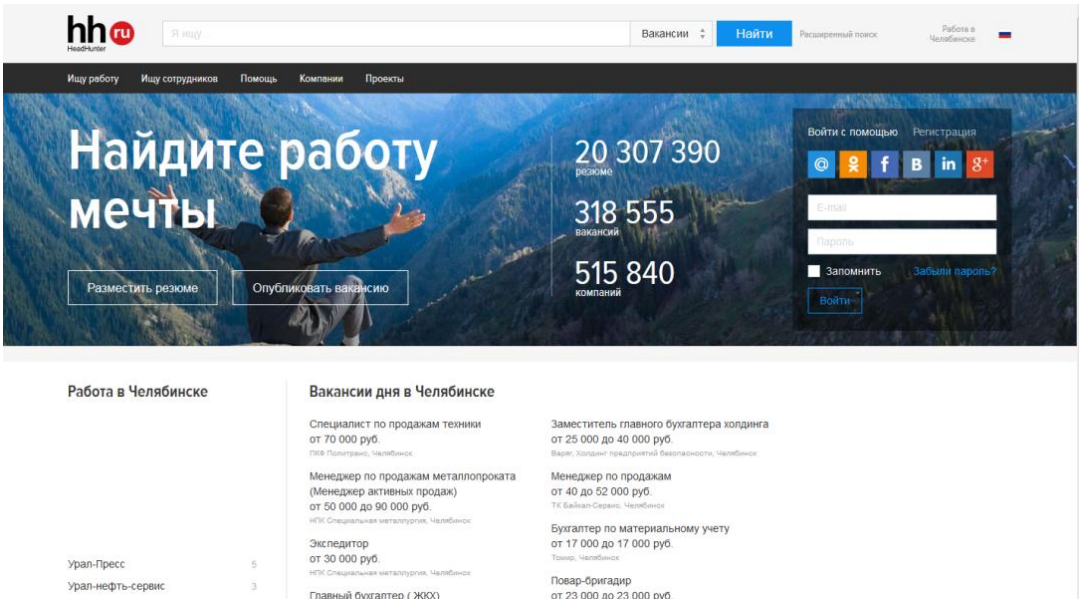

Рисунок 1.4 – Общий вид сайта hh.ru

Он позволяет размещать вакансии и резюме, смотреть уже размещенные, но для связи с соискателем или работодателем необходима регистрация, как и для размещения вакансий и резюме (рисунок 1.5).

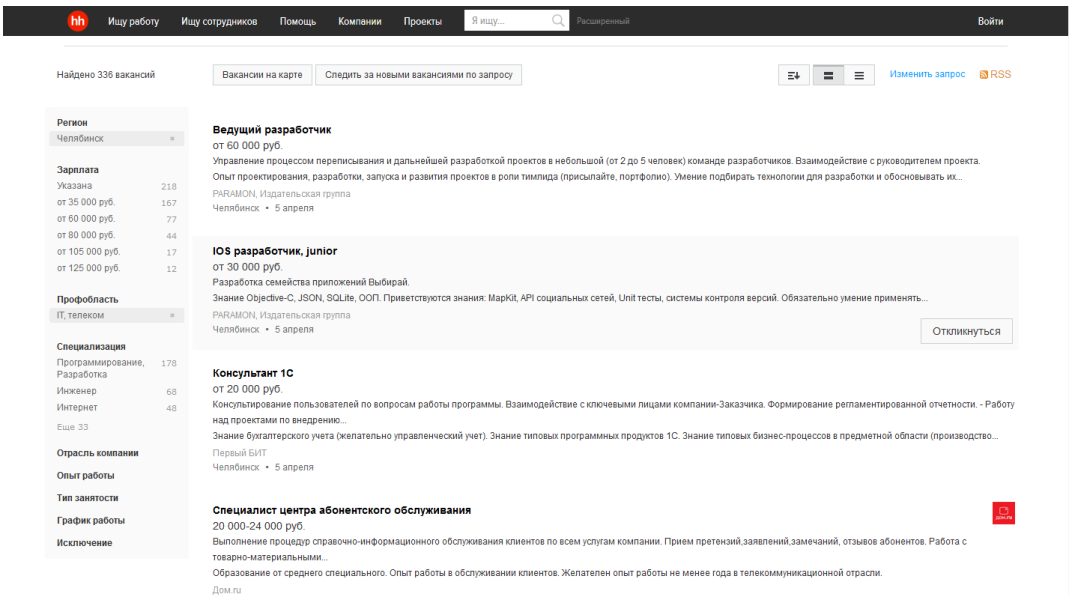

Рисунок 1.5 – Список вакансий на сайте hh.ru

Вакансии и резюме разбиты по категориям. При размещении вакансии можно загрузить информацию о компании, логотип компании, информацию о предоставляемой работе, требования к соискателю, условия работы, контактную информацию. При размещении резюме можно загрузить информацию о себе, фотографию, информацию о своих навыках, о предыдущем месте работы, о пожеланиях к работе, контактную информацию. Работодатель должен оплачивать доступ к контактной информации соискателей.

Кроме того сайт имеет современный и удобный интерфейс, удобную динамическую систему поиска, функцию просмотра списка вакансий на карте для того чтобы находить работу ближе к дому (рисунок 1.6).

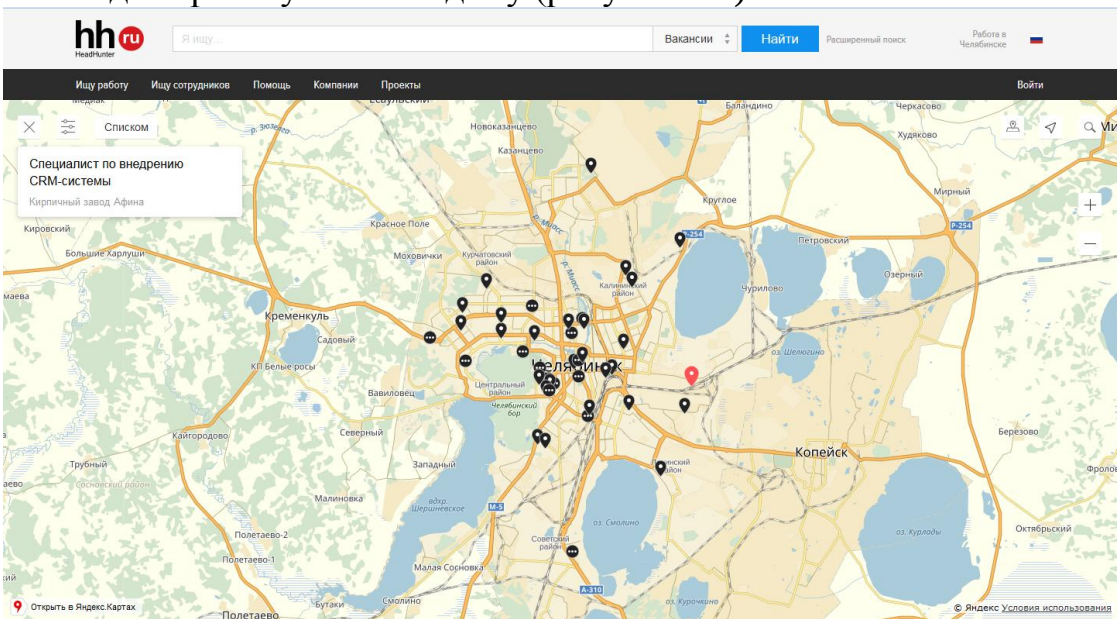

Рисунок 1.6 – Список вакансий в виде карты на сайте hh.ru

На сайте имеется очень большая база резюме (более 20 миллионов), вакансий (более 300 тысяч) и зарегистрированных компаний (более 500 тысяч). Сайт имеет разделение по регионам, включая страны ближнего зарубежья.

1.3. Центр содействия трудоустройству выпускников ИГУ

Сайт [3] является сайтам по поиску работы для выпускников Иркутского государственного университета (рисунок 1.7).

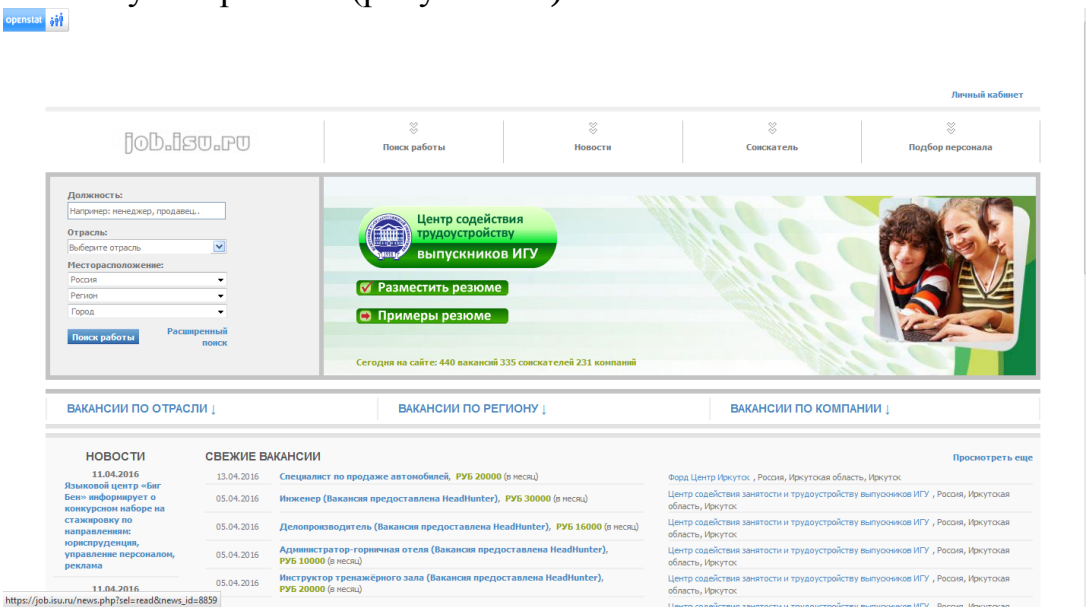

Рисунок 1.7 – Общий вид сайта job.isu.ru

Он позволяет размещать вакансии (рисунок 1.8) и резюме. Размещенные вакансии доступны для просмотра всем посетителям, резюме же доступны только работодателям. На сайте можно зарегистрироваться как в роли работодателя, так и в роли соискателя. Контакты для связи с работодателем доступны всем посетителям, но для связи через сайт, необходима регистрация. Вакансии можно просматривать по категориям, компаниям и регионам.

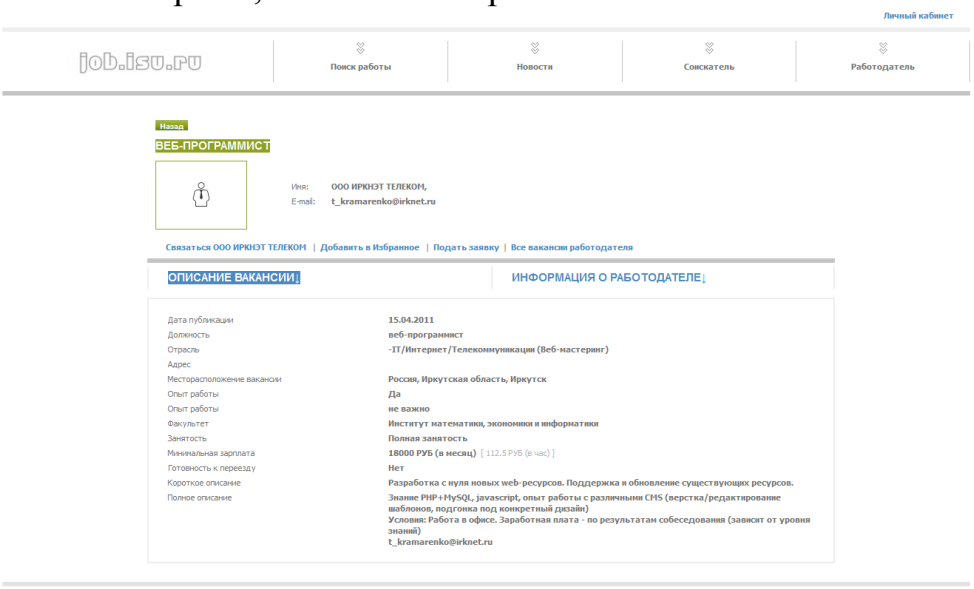

Рисунок 1.8 – Вакансия на сайте job.isu.ru

При размещении вакансии нужно указать должность, категорию к которой относится вакансия, факультет, на который рассчитана вакансия, зарплату, занятость, краткое и полное описание, необходимость в переезде для соискателя, опыт работы.

На сайте присутствует большое количество вакансий предоставленных сайтом [2], для связи с работодателем, разместившим такие вакансии, может потребоваться регистрация на сайте [2].

1.4. Региональный центр трудоустройства выпускников ЮУрГУ

Сайт [4] является сайтом по поиску работы для выпускников Южно-Уральского государственного университета (рисунок 1.9).

Сайт позволяет размещать вакансии и резюме. В резюме можно указать только фамилию, имя, отчество, телефон, e-mail и дополнительную информацию в виде текста. В вакансии можно указать название организации, город, предлагаемую должность, заработную плату, контактную информацию, вид занятости и дополнительную информацию. Все резюме и вакансии не сайте разбиты по специальностям и квалификациям (бакалавриат, магистратура). На сайте можно произвести поиск по нужной специальности.

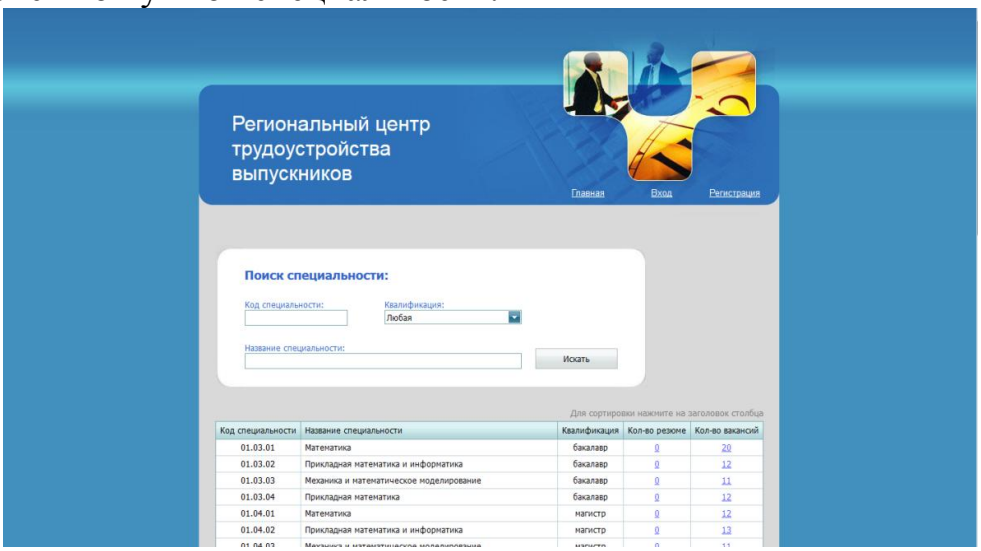

Рисунок 1.9 – Общий вид сайта univeris.susu.ru/job/

Выводы по разделу 1

Были рассмотрены существующие сайты для поиска работы.

Отметим их плюсы и минусы.

Из плюсов можно отметить широкий функционал, большие базы вакансий и резюме. Главным минусом таких сайтов является, то, что они стараются охватить, как можно большую аудиторию и из-за этого на таких сайтах невозможно произвести поиск вакансий и резюме по конкретным параметрам (например, по языкам программирования), приходится тратить больше времени и усилий для поиска. Ещё одним минусом для выпускников и студентов ВУЗов является, то что, приходя на такой сайт им очень сложно найти себе подходящую вакансию, так как работодатели чаще всего набирают людей с опытом работы, либо со знанием каких-то специфических технологий, не преподаваемых в университете. Для работодателя же минусом таких сайтов является, то, что с него разными способами пытаются взять денежную плату, либо за контактную информацию соискателя, либо за, то чтобы его вакансия была выше в списке.

2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Процесс проектирования базы данных можно разбить на 3 этапа:

1. Концептуальный – определение и анализ данных, требований к ним.
2. Логическое проектирование – преобразование требований полученных на первом этапе в структуры данных.
3. Физическое проектирование – реализация базы данных в СУБД.

На первом этапе необходимо сформировать понятия о предметах, с которыми будет работать данная система. Для этого будем использовать ER-модель. С помощью неё можно выделить ключевые сущности и атрибуты сущности (свойства объекта), после чего установим связи между ними. На втором этапе мы получим схему реляционной базы данных. И на заключительном этапе реализуем базу данных в СУБД.

2.1. Выделение сущностей предметной области

Выделим следующие сущности: пользователь, компания, резюме, вакансия, навык, категория.

Сущность пользователь должна обладать следующими свойствами:

- Идентификатор;
- Email, будет использоваться в качестве логина;
- Пароль;
- Имя;
- Фамилия;
- Отчество;
- Номер телефона;
- Уникальный идентификатор, для безопасной авторизации;
- Дата регистрации;
- Роль;
- Флаг активирована ли учетная запись.

Сущность компания имеет следующие свойства:

- Идентификатор;
- Название;
- Ссылка на сайт;
- Адрес;
- Краткая информация о компании;
- Логотип;
- Идентификатор пользователя, который является представителем компании.

Сущность вакансия обладает следующими свойствами:

- Идентификатор;
- Должность;
- Тип работы;
- График работы;
- Зарплата;
- Условия работы;

- Обязанности;
- Образование
- Дата размещения;
- Идентификатор пользователя.

Сущность резюме обладает следующими свойствами:

- Идентификатор;
- Личная информация о соискателе;
- Возраст;
- Должность;
- Тип работы;
- График работы;
- Зарплата;
- Образование;
- Пожелания к работе;
- Дата размещения;
- Идентификатор пользователя.

Сущность навык обладает следующими свойствами:

- Идентификатор;
- Название навыка;
- Идентификатор категории, к которой он относится.

Сущность категория обладает следующими свойствами:

- Идентификатор;
- Название категории.

2.2. ER-модели сущностей

Отношение между пользователем и резюме изображено на рисунке 2.1. У пользователя может быть множество опубликованных резюме, но каждое резюме относится только к одному пользователю.

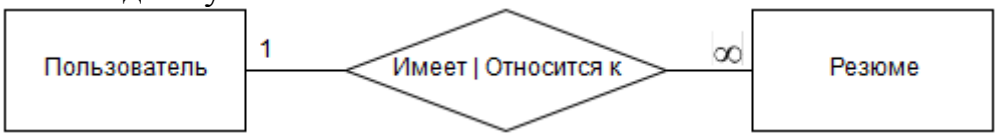

Рисунок 2.1

Отношение между пользователем и вакансией изображено на рисунке 2.2. У пользователя может быть множество опубликованных вакансий, но каждая вакансия относится только к одному пользователю.

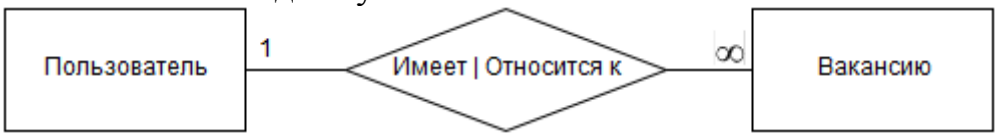

Рисунок 2.2

Отношение между пользователем и компанией изображено на рисунке 2.3. Пользователь может состоять только в одной компании. Компания может иметь только одного пользователя.

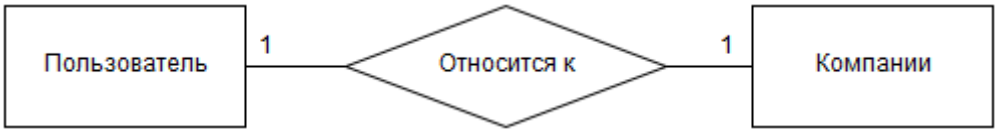

Рисунок 2.3

Отношение между вакансией и навыком изображено на рисунке 2.4. К одной вакансии может относиться множество навыков. К одному навыку может относиться множество вакансий.

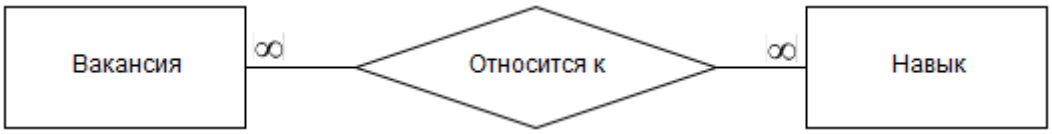

Рисунок 2.4

Отношение между резюме и навыком изображено на рисунке 2.5. К одному резюме может относиться множество навыков. К одному навыку может относиться множество резюме.

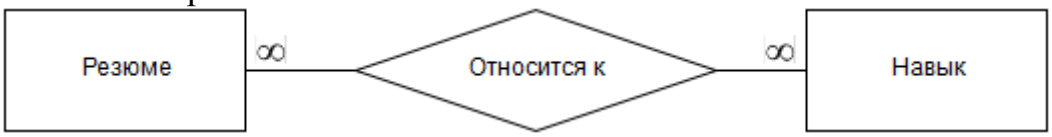

Рисунок 2.5

Отношение между навыком и категорией изображено на рисунке 2.6. К одному навыку может относиться одна категория. К одной категории может относиться множество навыков.

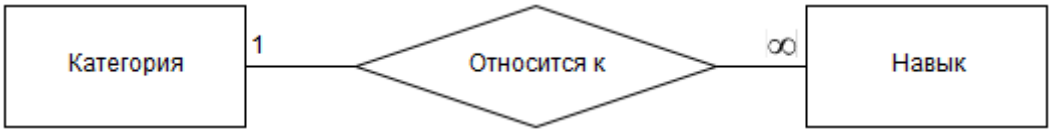

Рисунок 2.6

У нас есть связь многие ко многим между сущностями резюме и навык, и вакансия и навык. Нужно избавиться от этого вида связи, для этого создадим промежуточную сущность. Для связи резюме и навык создадим промежуточную сущность НавРез. Сущность НавРез будет иметь следующие свойства:

- Идентификатор;
- Идентификатор резюме;
- Идентификатор навыка.

Для связи вакансия и навык создадим промежуточную сущность НавВак. Сущность НавВак будет иметь следующие свойства:

- Идентификатор;
- Идентификатор вакансии;
- Идентификатор навыка.

Полная ER-диаграмма представлена на рисунке 2.7.

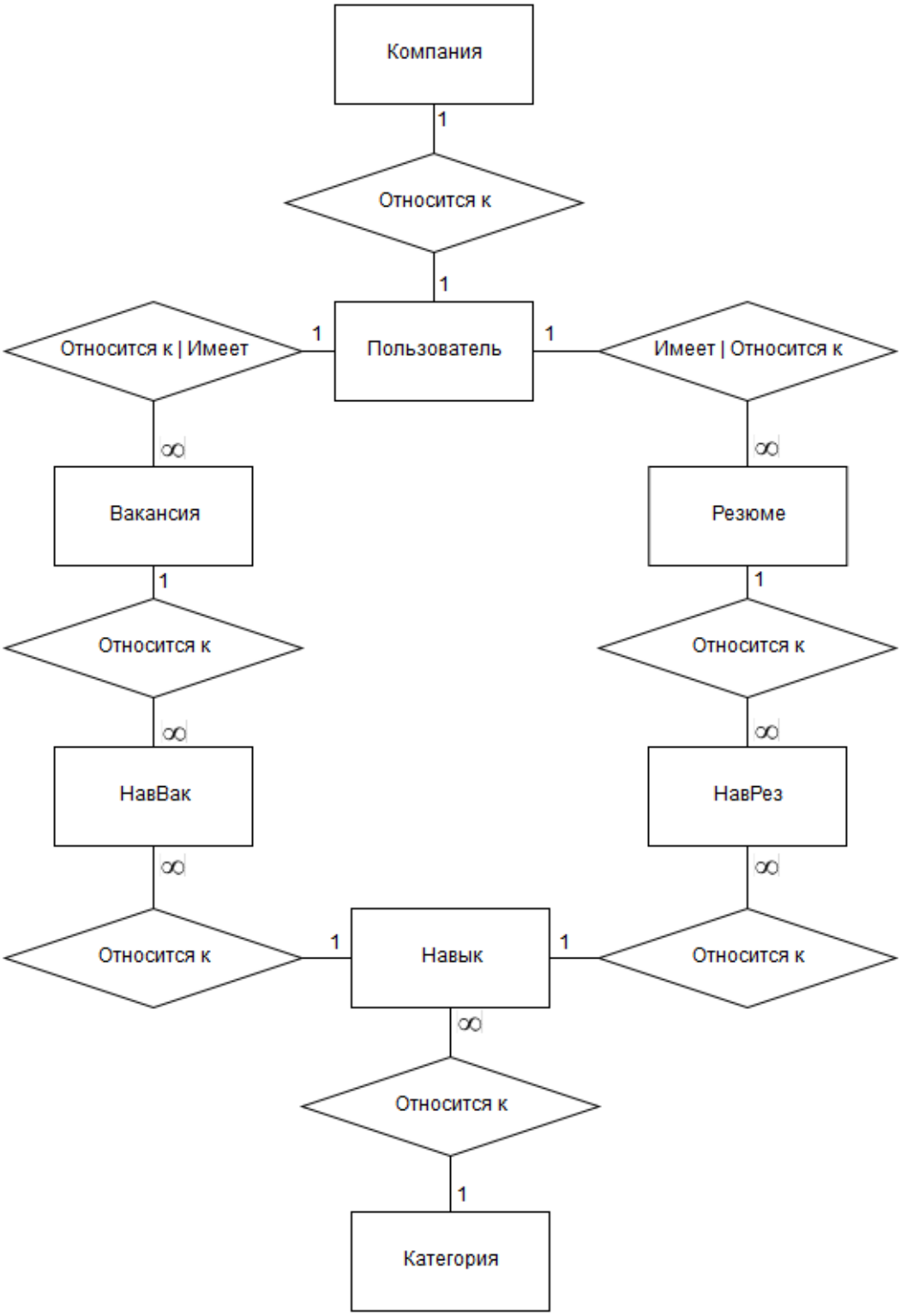

Рисунок 2.7

Для преобразования ER-диаграмм к схеме базы данных, сущности были преобразованы в таблицы.

В таблице 2.1 представлена сущность пользователь.

Таблица 2.1

Ключ	Атрибут	Тип данных	NULL
PK	user_id	Int	NOT NULL
	email	Varchar(50)	NOT NULL
	password	Varchar(32)	NOT NULL
	fullname	Varchar(100)	NOT NULL
	phone	Varchar(11)	NOT NULL
	uniq_id	Varchar(32)	NOT NULL
	regdate	Datetime	NOT NULL
	role	Int	NOT NULL
	active	Tinyint	NOT NULL

В таблице 2.2 представлена сущность вакансия.

Таблица 2.2

Ключ	Атрибут	Тип данных	NULL
PK	vacancy_id	Int	NOT NULL
	post	Varchar(150)	NOT NULL
	type	Varchar(100)	NOT NULL
	shedule	Varchar(100)	NOT NULL
	salary	Varchar(30)	NOT NULL
	conditions	Text	NOT NULL
	duties	Text	NOT NULL
	date	Datetime	NOT NULL
FK	user_id	Int	NOT NULL

В таблице 2.3 представлена сущность резюме.

Таблица 2.3

Ключ	Атрибут	Тип данных	NULL
PK	resume_id	Int	NOT NULL
	about_me	Text	NOT NULL
	age	Int	NOT NULL
	post	Varchar(150)	NOT NULL
	type	Varchar(100)	NOT NULL
	shedule	Varchar(100)	NOT NULL
	salary	Varchar(30)	NOT NULL
	suggestions	Text	NOT NULL
	education	Varchar(50)	NOT NULL
	date	Datetime	NOT NULL
FK	user_id	Int	NOT NULL

В таблице 2.4 представлена сущность компания.

Таблица 2.4

Ключ	Атрибут	Тип данных	NULL
PK	company_id	Int	NOT NULL
	company_name	Varchar(25)	NOT NULL
	site	Varchar(50)	NOT NULL
	info	Text	NOT NULL
	address	Varchar(100)	NOT NULL
	avatar	Varchar(20)	NOT NULL
FK	user_id	Int	NOT NULL

В таблице 2.5 представлена сущность навык.

Таблица 2.5

Ключ	Атрибут	Тип данных	NULL
PK	skill_id	int	NOT NULL
	skill_name	Varchar(100)	NOT NULL
FK	category_id	int	NOT NULL

В таблице 2.6 представлена сущность категория.

Таблица 2.6

Ключ	Атрибут	Тип данных	NULL
PK	category_id	Int	NOT NULL
	category_name	Varchar(100)	NOT NULL

В таблице 2.7 представлена сущность НавВак.

Таблица 2.7

Ключ	Атрибут	Тип данных	NULL
PK	id	int	NOT NULL
FK	vacancy_id	int	NOT NULL
FK	skill_id	int	NOT NULL

В таблице 2.8 представлена сущность НавРез.

Таблица 2.8

Ключ	Атрибут	Тип данных	NULL
PK	id	int	NOT NULL
FK	resume_id	int	NOT NULL
FK	skill_id	int	NOT NULL

2.3. Физическое проектирование

В качестве СУБД будем использовать MySQL, так как она является самой популярной СУБД при разработке сайтов, проста в работе и обладает достаточной функциональностью для разрабатываемого проекта. На рисунке 2.8 представлена диаграмма базы данных.

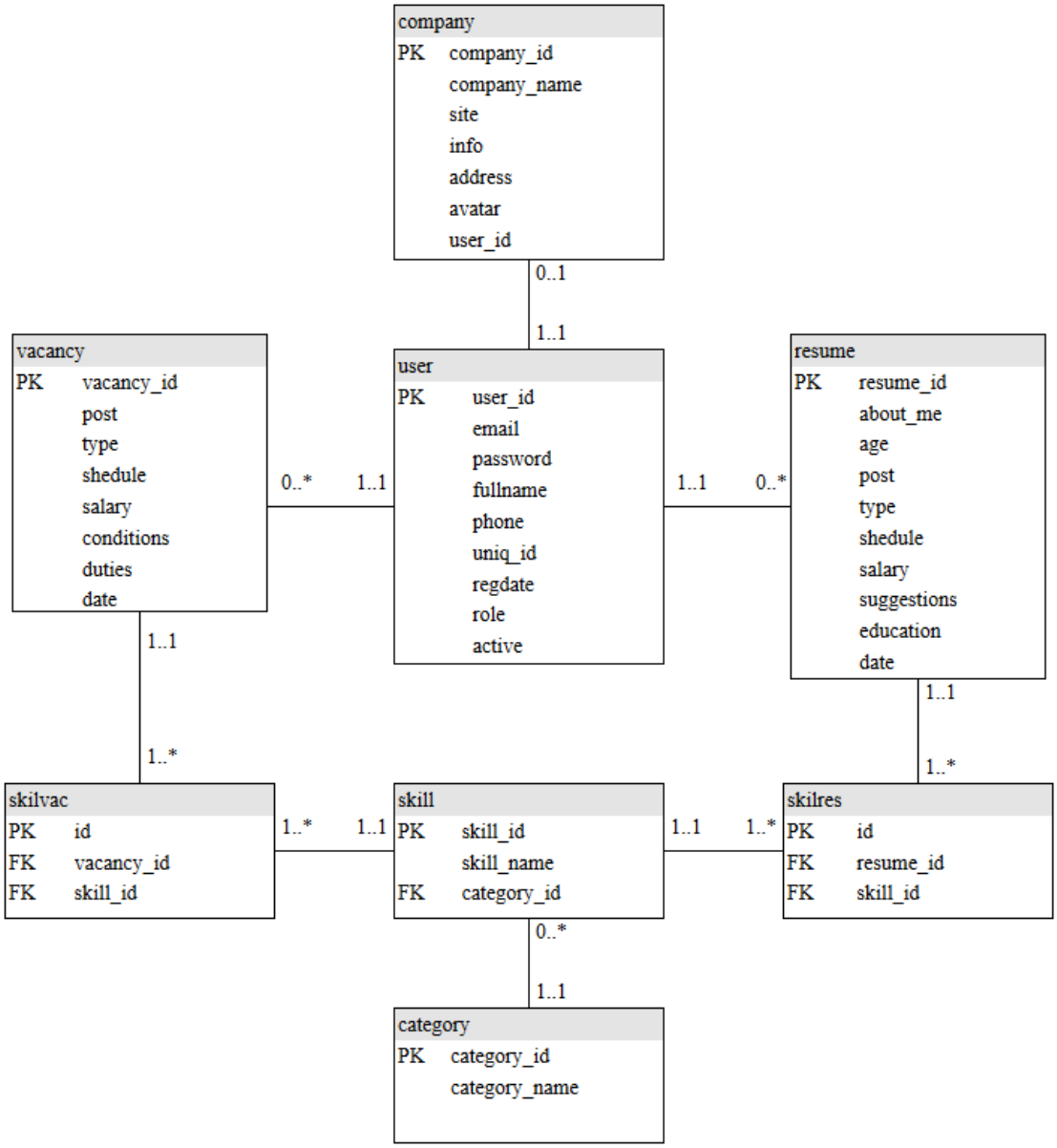

Рисунок 2.8

Выводы по разделу 2

В процессе разработки базы данных выделены сущности и их атрибуты, составлены ER-диаграммы. Получена схема базы данных. База данных нормализована. Она находится в нормальной форме Бойса-Кодда.

3. РАЗРАБОТКА ИНТЕРФЕЙСА

3.1. Разработка персонажей

Для разработки интерфейса будет применён метод персон описанный в книге [5, с. 109-146]. Суть метода в том, что нужно придумать персонажа из реальной целевой аудитории разрабатываемого продукта, но этот персонаж не должен быть реальным человеком.

Персонаж работодатель.

Иванова Ирина Александровна. Профессия – руководитель проекта в компании «Прикладные технологии». 30-летняя женщина, высшее образование. Она каждый день работает с компьютером, пишет e-mail, использует Word и занимается разработкой программ. Ей, как руководителю нового проекта нужно найти студента, который обладает нужными навыками. Для этого она может либо разместить вакансию с указанием требований к соискателю и ждать резюме, либо она может просмотреть размещенные резюме для того, чтобы подобрать персонал, с соответствующими знаниями.

Персонаж соискатель.

Березуцкий Вадим Генадьевич. Студент 4-го курса. Учиться на факультете математики, механики и компьютерных наук, в ЮУрГУ, пользуется компьютером каждый день: смотрит фильмы, работает с документами в Word, Excel, любит играть в компьютерные игры, сидит в интернете, пишет программы, изучает языки программирования, которые не преподают в университете. Цель - устроиться на работу. Ему необходимо просмотреть список вакансий и выбрать подходящую ему по знаниям и интересам вакансию, после чего ему нужно связаться с работодателем.

Персонаж администратор.

Ефремов Дмитрий Денисович. Преподаватель факультета математики, механики и компьютерных наук. Хорошо пользуется компьютером, ведет занятия по программированию. Ему необходимо зарегистрировать студентов группы.
Ещё администратору может понадобиться добавить, отредактировать навыки и или категории умений доступных для резюме и вакансий или же сортировать навыки, добавленные пользователями. Также ему может понадобиться проверить

новые резюме и вакансии на правильность заполнение, отредактировать или удалить их в случае каких-либо ошибок.

У нас есть 3 ключевых персонажа. Мы должны создать интерфейс, который будет включать функции для каждого персонажа. В зависимости от пользователя, ему будут необходимы разные элементы управления. Необходимо определить какому пользователю, какие функции нужны. Для этого можно использовать диаграммы вариантов использования.

3.2. Разработка диаграмм вариантов использования

Ещё для разработки интерфейса понадобится использовать диаграмму вариантов использования. Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе.

Диаграмма вариантов использования администратора (рисунок 3.1).

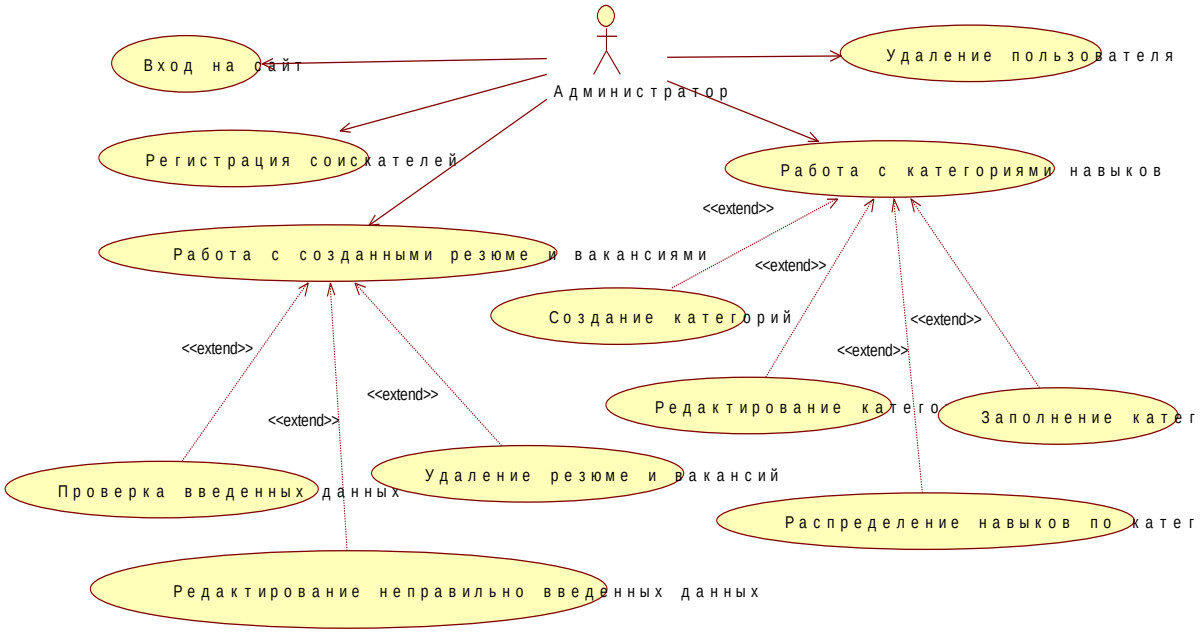

Рисунок 3.1

Вход на сайт – пользователь вводит свой логин и пароль. В качестве логина он использует свой email. Это наиболее предпочтительный вариант, потому что пользователю не придётся придумывать новый логин и запоминать его.

Регистрация соискателей – только администратор может регистрировать студентов на сайте. Он вводит email и ФИО каждого студента для регистрации.

Проверка введенных данных – администратор проверяет правильность добавленных резюме и вакансий. При необходимости он может отредактировать или удалить резюме или вакансию. Так как работодатель регистрируется сам, то в случае нарушения им правил сайта, администратор может удалить его.

Создание новых категорий – навыки, перечисляемые в резюме и вакансиях, распределены по категориям, администратор может добавлять эти категории.

Заполнение категорий – администратор может добавлять в каждую категорию новые навыки.

Распределение навыков по категориям – администратор может перемещать навыки по категориям, эта функция может пригодится, если пользователи будут добавлять свои навыки, которые будут помещаться в категорию разные.

Диаграмма вариантов использования соискателя (рисунок 3.2).

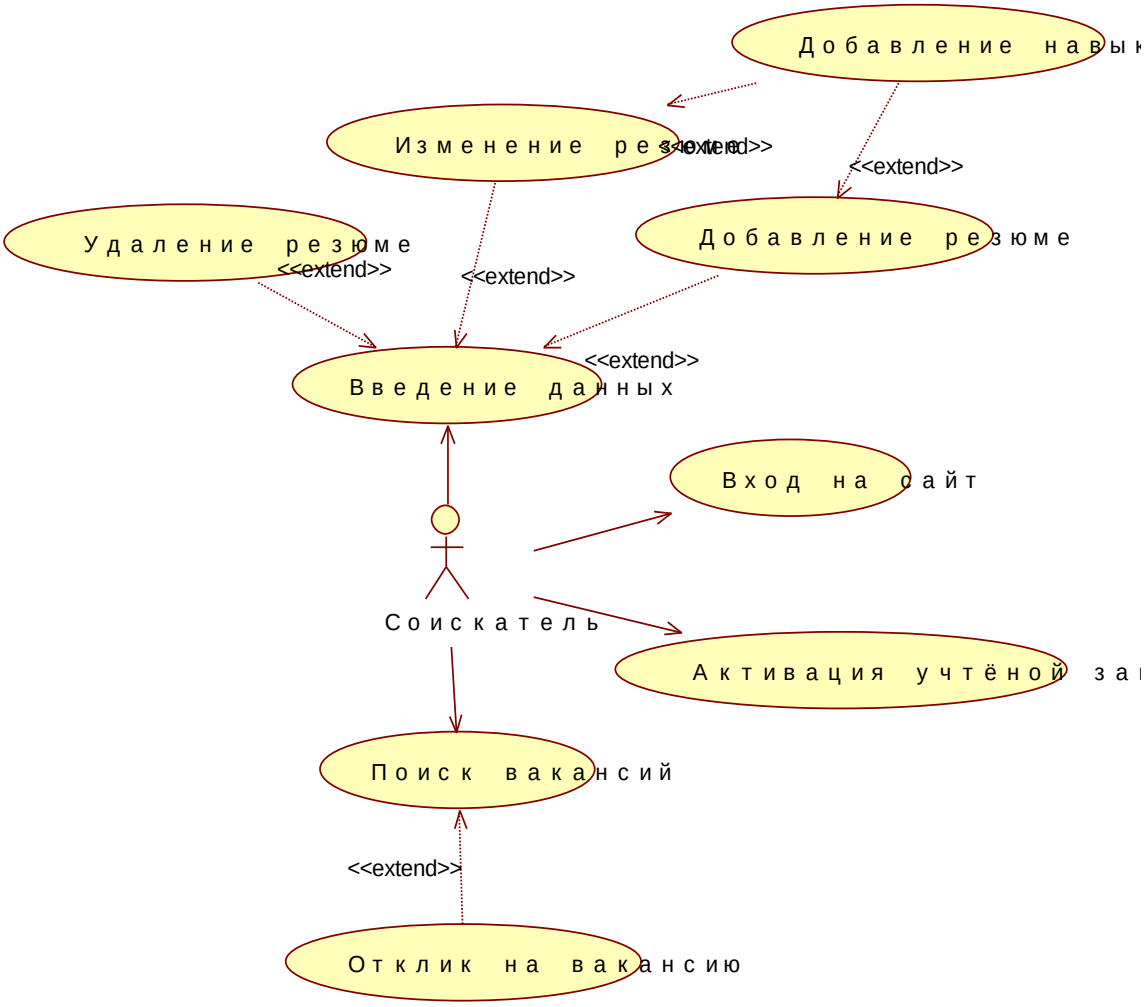

Рисунок 3.2

Добавление навыков – при добавлении резюме или вакансии у пользователя может возникнуть потребность добавить навыки, которых нет в базе.

Вход на сайт – аналогично администратору.

Отклик на вакансию – соискатель может просмотреть контактную информацию работодателя в понравившейся вакансии и связаться с ним.

Активация учётной записи – после того, как администратор зарегистрирует студентов, им на email высылается пароль и при первом входе на сайт, соискатель должен активировать учетную запись, введя новый пароль и номер своего телефона.

Диаграмма вариантов использования работодателя (рисунок 3.3).

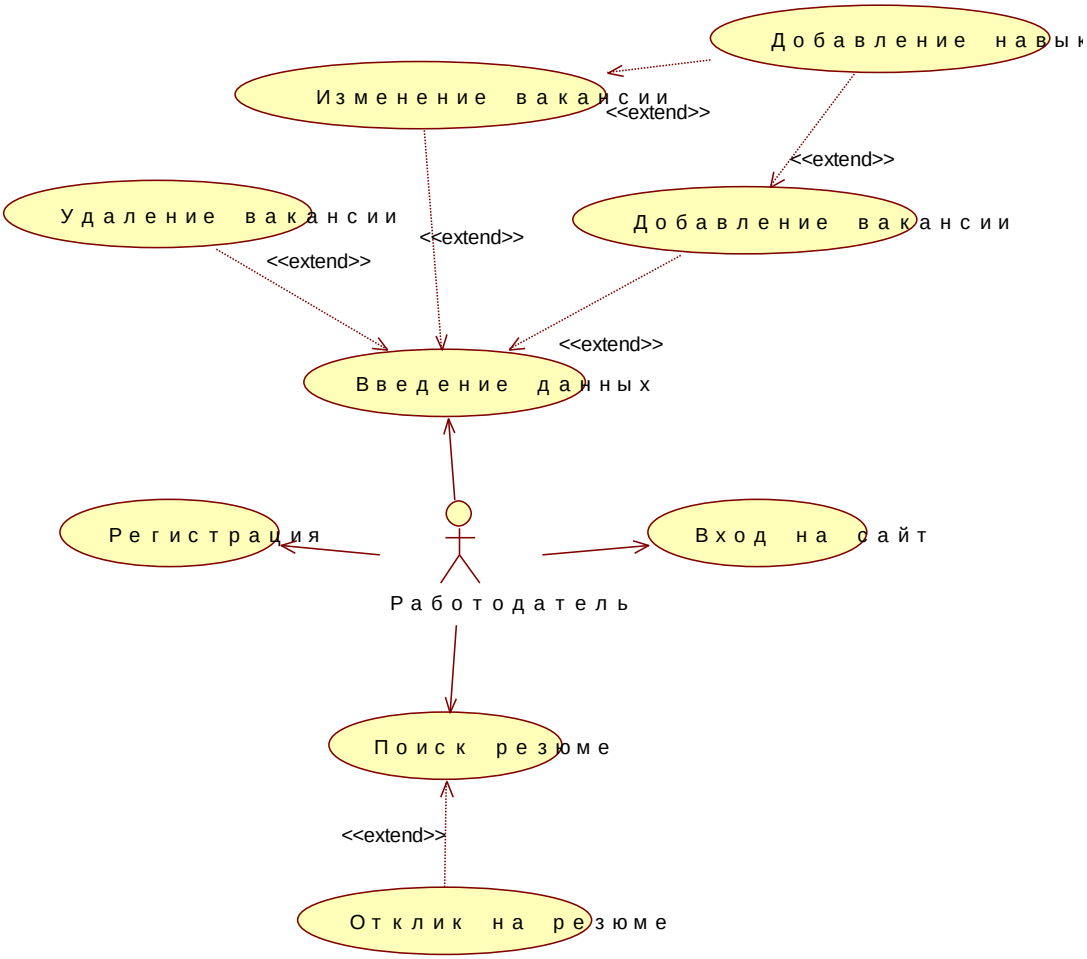

Рисунок 3.3

Добавление навыков – аналогично соискателю.

Регистрация – работодатель сам регистрируется на сайте, после чего ему придет письмо на указанный email адрес, в котором содержится ссылка для активации профиля.

Отклик на резюме – работодатель не может просматривать контактную информацию пользователя опубликовавшего резюме, но он может отправить ему email через форму на сайте.

3.3. Инструменты для разработки интерфейса

Интерфейса сайта должен разрабатываться на HTML 5 и CSS 3 (при изучении CSS 3 использовалась книга [6]), однако кроме них есть смысл использовать сторонние библиотеки: Twitter Bootstrap и Jquery UI.

Twitter Bootstrap это набор инструментов, основанный на HTML 5, CSS 3, JavaScript, для создания интерфейса. Он позволяет экономить время разработки, просто в использовании, совместим с различными браузерами, и позволяет с лёгкостью создавать адаптивный веб-дизайн.

Jquery UI это библиотека JavaScript с открытым исходным кодом для упрощения создания веб-интерфейса. В ней нам понадобится плагины Droppable и Draggable, при их изучении была использована статья [7]. Аналогично Bootstrap она ускоряет и упрощает создание интерфейса для сайта.

3.4. Разработанный интерфейс

Главная страница

На главной странице (рисунок 3.4), как и на всех страницах сайта, есть меню. Пункт вакансии включает в себя подпункты: разместить вакансию, поиск вакансии и список вакансий. Аналогичные подпункты включает в себя пункт резюме. Кроме того на главной странице присутствует информация о количестве вакансий, резюме и компаний на сайте.

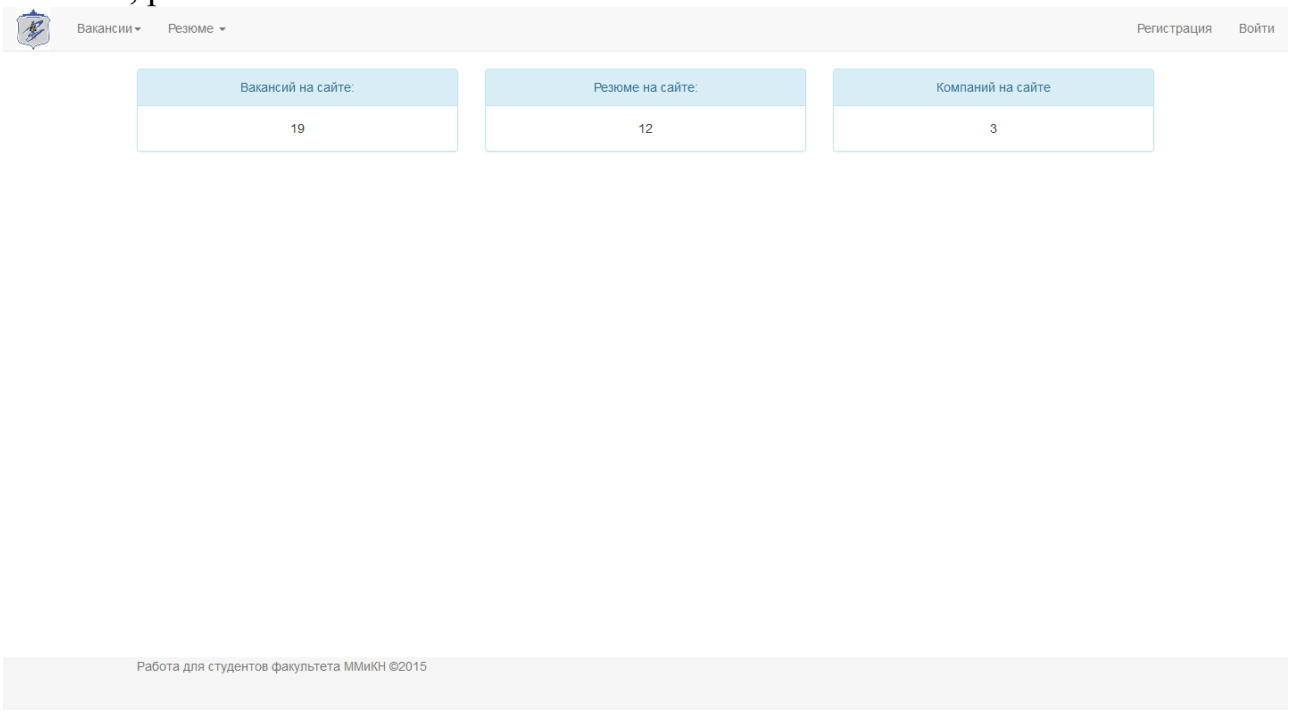

Рисунок 3.4

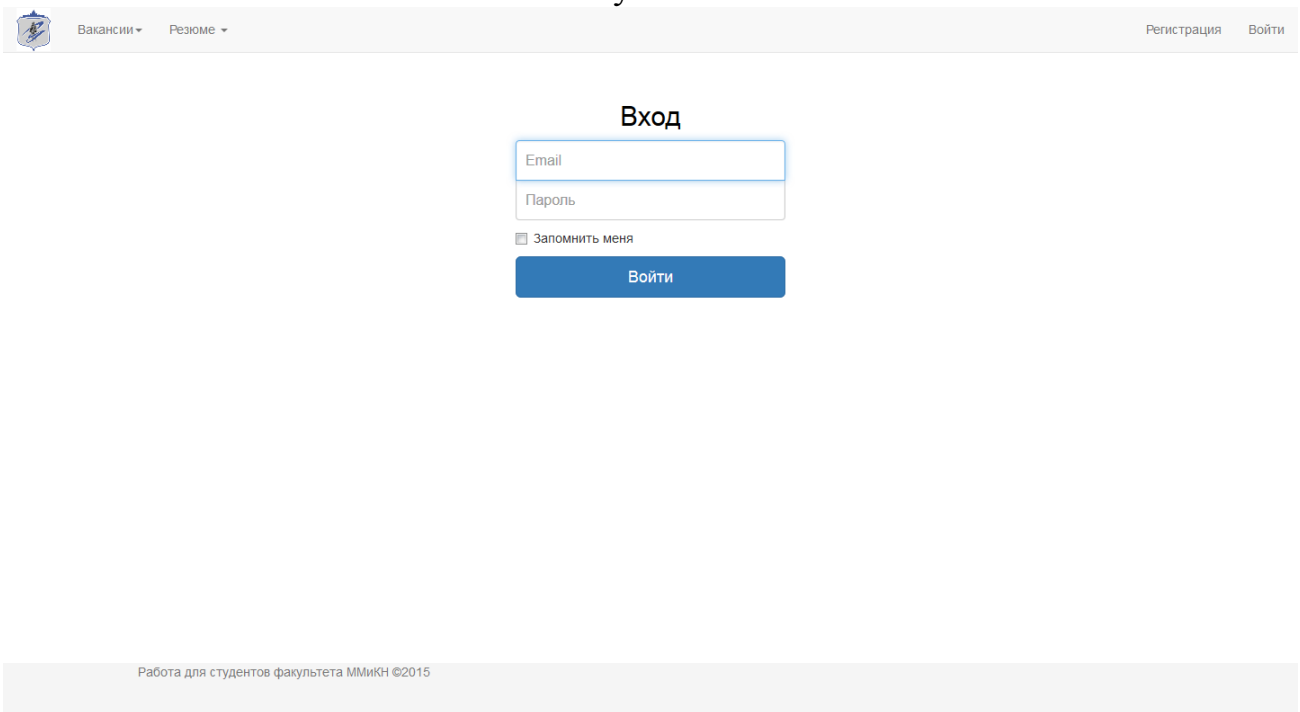

Рисунок 3.5

Страница входа

На странице входа (рисунок 3.5) находится форма для авторизации пользователя.

Страница регистрации

На странице регистрации (рисунок 3.6) находится форма для регистрации пользователя.

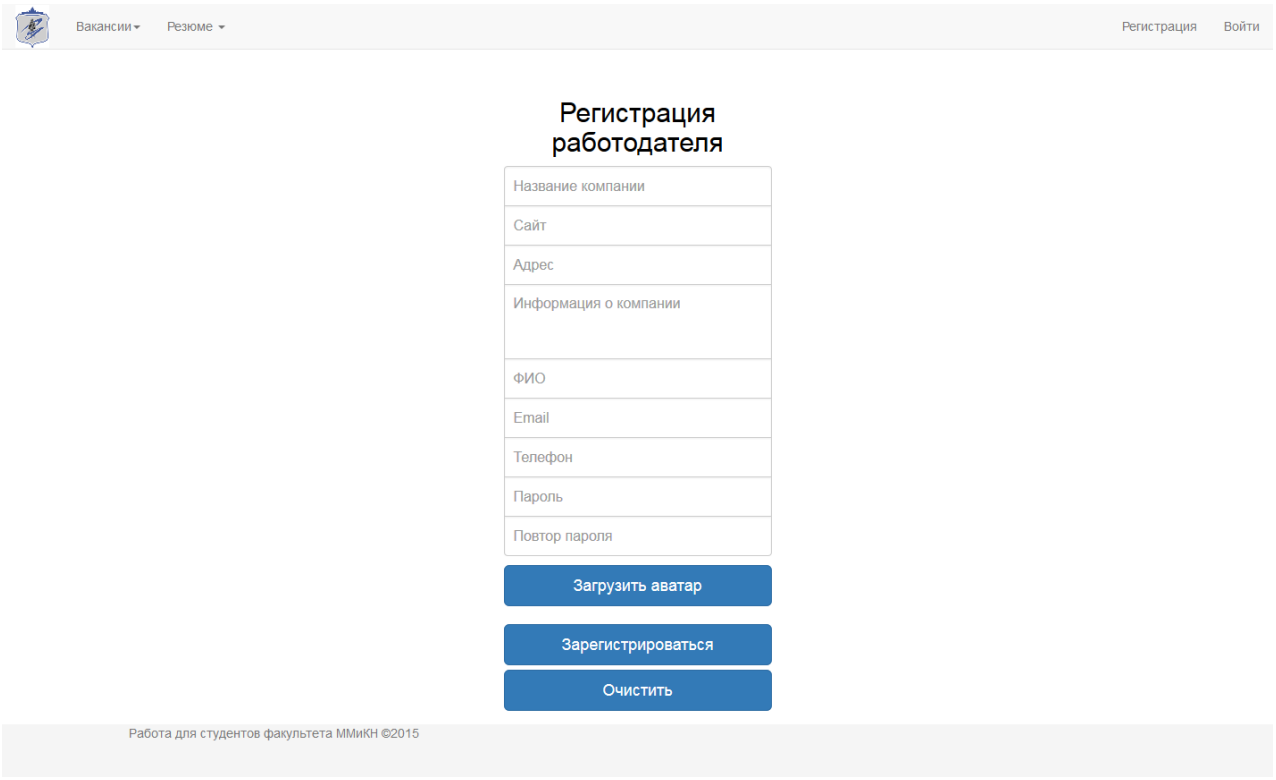

Рисунок 3.6

Рисунок 3.7

Страница ошибки 404

На странице ошибки 404 (рисунок 3.7) размещена картинка с информацией о том, что страница не найдена.

Страница ошибки доступа

На странице ошибки доступа (рисунок 3.8) размещена информация о том, что страница не доступна.

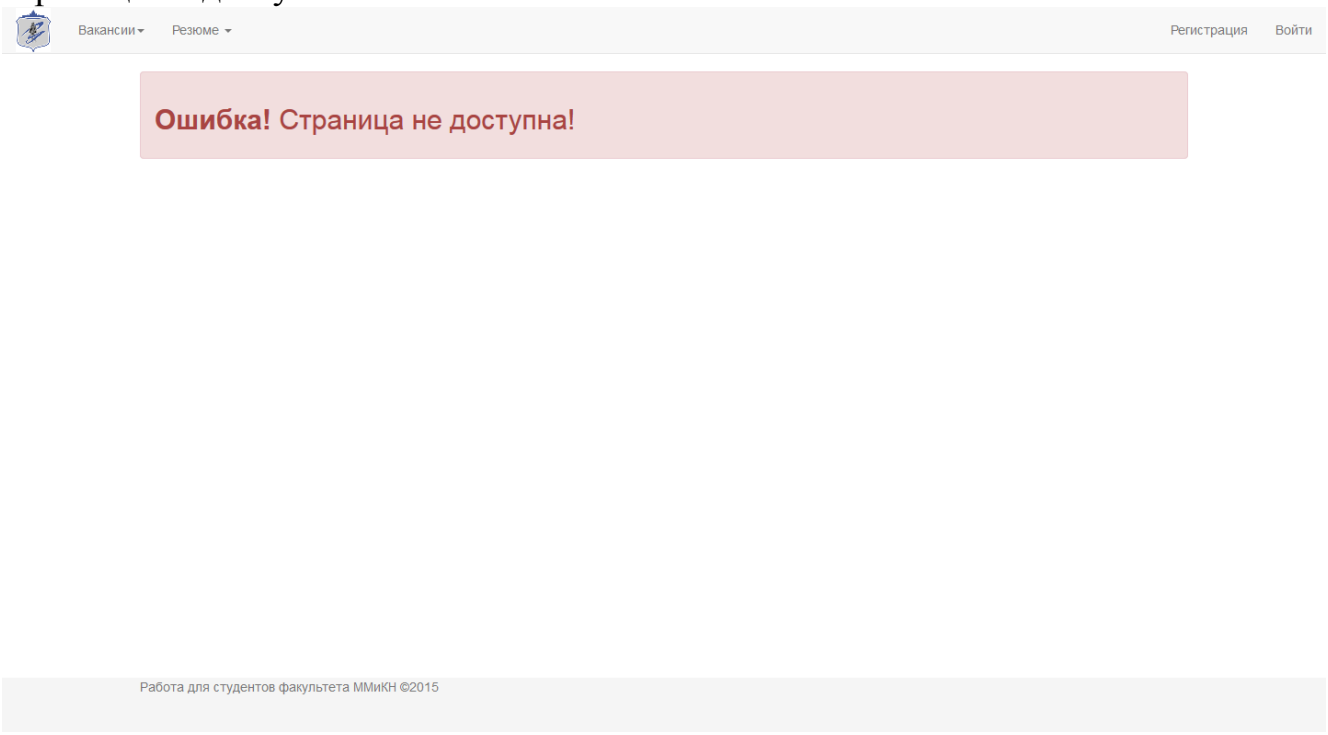

Рисунок 3.8

Страница со списком вакансий

На странице со списком вакансий (рисунок 3.9) размещена информация о пяти вакансиях. Также внизу имеется переключатель страниц, если есть непоказанные на странице вакансии.

Страница добавления вакансии

На странице добавления вакансии (рисунок 3.10) находится форма для добавления вакансии и список навыков, которые необходимо переносить в соседнюю ячейку для добавления их к вакансии.

Страница вакансии

На странице вакансии (рисунок 3.11) размещена вся информация о вакансии и информация, о компании, предлагающей вакансию.

Страница вакансии для незарегистрированного пользователя

Страница вакансии для незарегистрированного пользователя (рисунок 3.12) аналогична странице вакансии, но вместо контактной информации размещено сообщение о её недоступности.

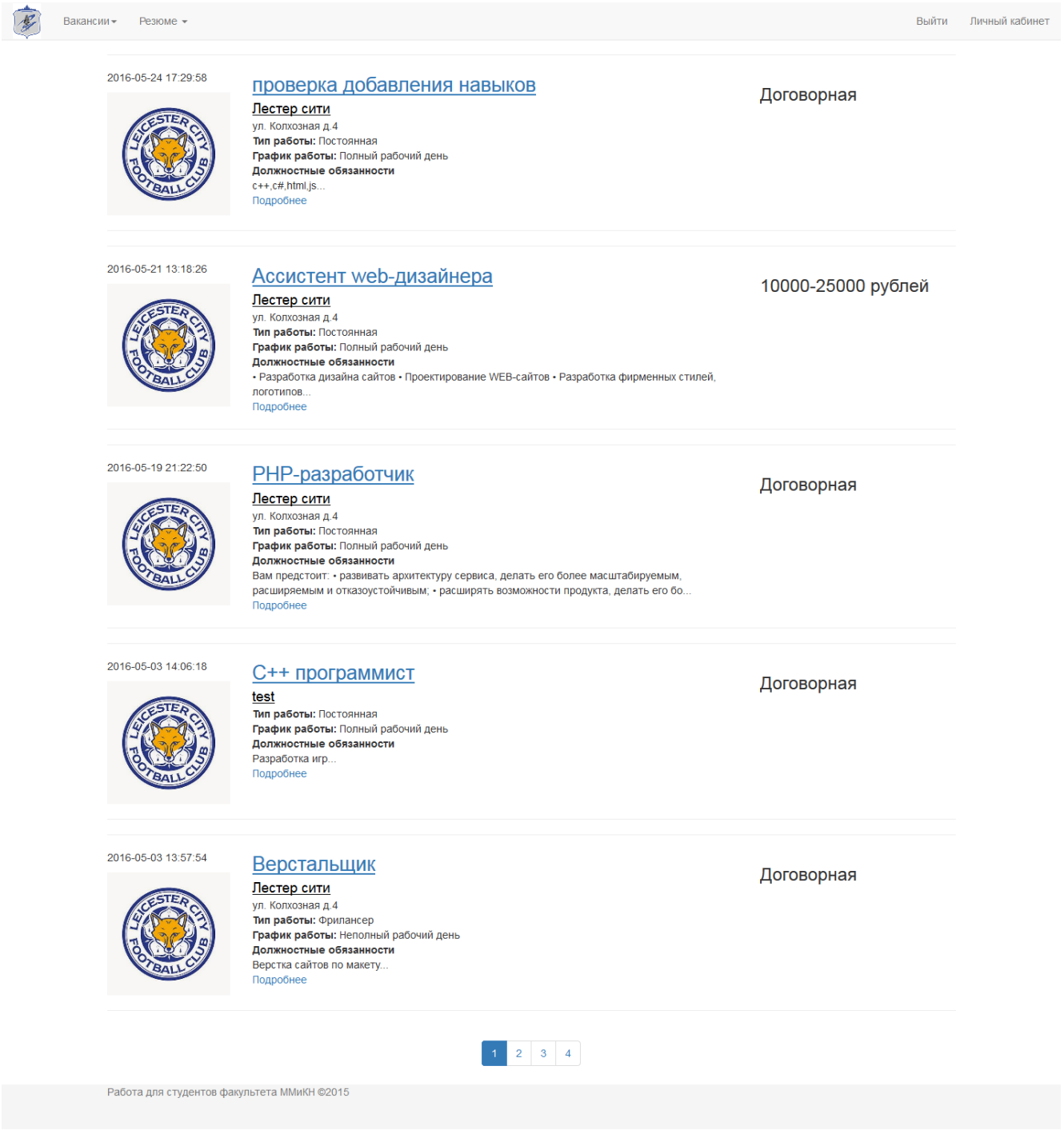
Вакансии
Резюме
Выйти
Личный кабинет
2016-05-24 17:29:58
проверка добавления навыков
Лестер сити
ул. Колхозная д.4
Тип работы: Постоянная
График работы: Полный рабочий день
Должностные обязанности
c++,c#,html,js...
Подробнее
Договорная
2016-05-21 13:18:26
Ассистент web-дизайнера
Лестер сити
ул. Колхозная д.4
Тип работы: Постоянная
График работы: Полный рабочий день
Должностные обязанности
• Разработка дизайна сайтов • Проектирование WEB-сайтов • Разработка фирменных стилей, логотипов...
Подробнее
10000-25000 рублей
2016-05-19 21:22:50
PHP-разработчик
Лестер сити
ул. Колхозная д.4
Тип работы: Постоянная
График работы: Полный рабочий день
Должностные обязанности
Вам предстоит: • развивать архитектуру сервиса, делать его более масштабируемым, расширяемым и отказоустойчивым; • расширять возможности продукта, делать его бо...
Подробнее
Договорная
2016-05-03 14:06:18
C++ программист
test
Тип работы: Постоянная
График работы: Полный рабочий день
Должностные обязанности
Разработка игр...
Подробнее
Договорная
2016-05-03 13:57:54
Верстальщик
Лестер сити
ул. Колхозная д.4
Тип работы: Фрилансер
График работы: Неполный рабочий день
Должностные обязанности
Верстка сайтов по макету...
Подробнее
Договорная
1
2
3
4
Работа для студентов факультета ММиКН ©2015

Рисунок 3.9

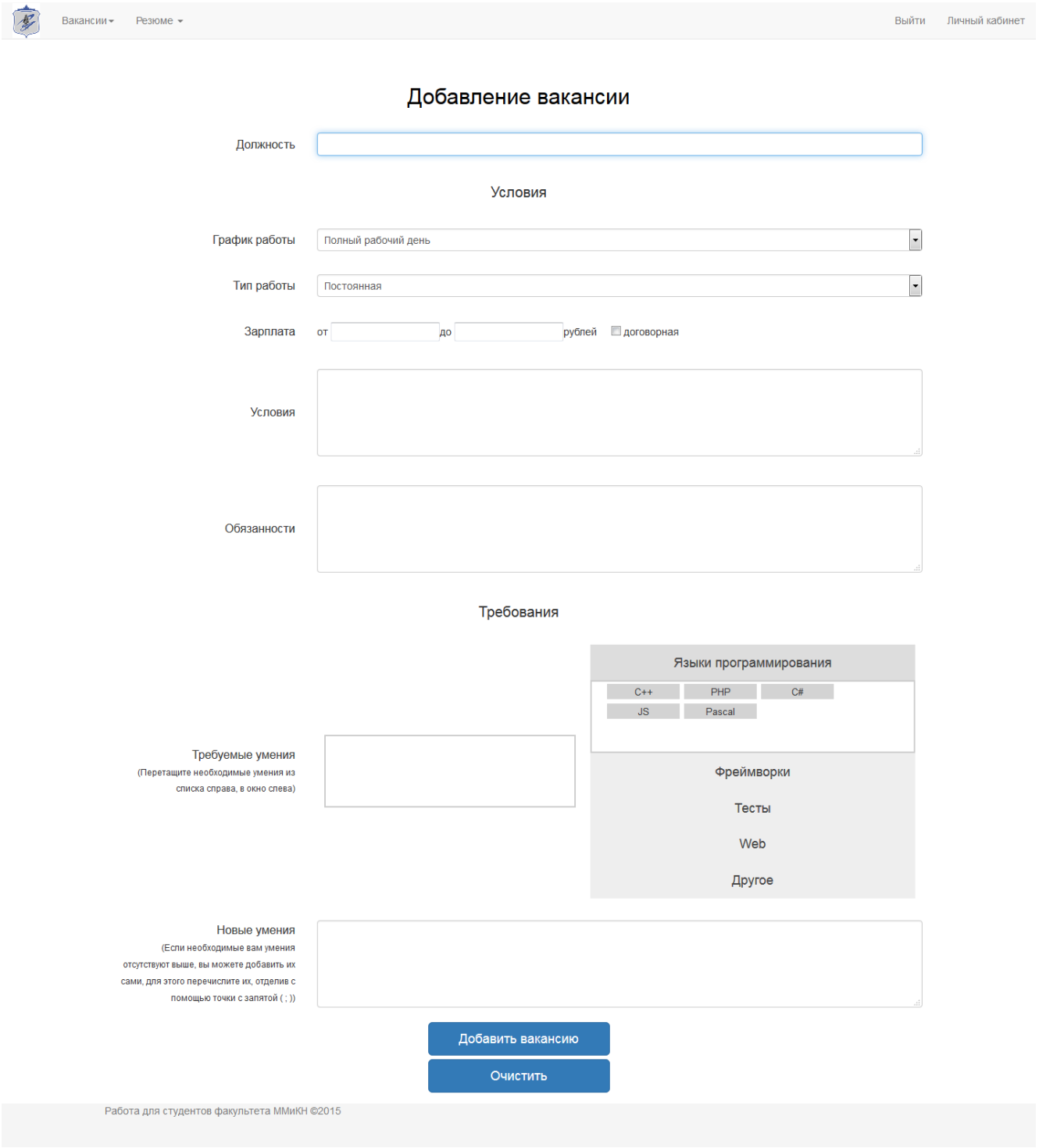
Вакансии
Резюме
Выйти
Личный кабинет
Добавление вакансии
Должность
Условия
График работы
Полный рабочий день
Тип работы
Постоянная
Зарплата
от
до
рублей
договорная
Условия
Обязанности
Требования
Требуемые умения
(Перетащите необходимые умения из списка справа, в окно слева)
Языки программирования
C++
PHP
C#
JS
Pascal
Фреймворки
Тесты
Web
Другое
Новые умения
(Если необходимые вам умения отсутствуют выше, вы можете добавить их сами, для этого перечислите их, отделив с помощью точки с запятой (;))
Добавить вакансию
Очистить
Работа для студентов факультета ММиКН ©2015

Рисунок 3.10

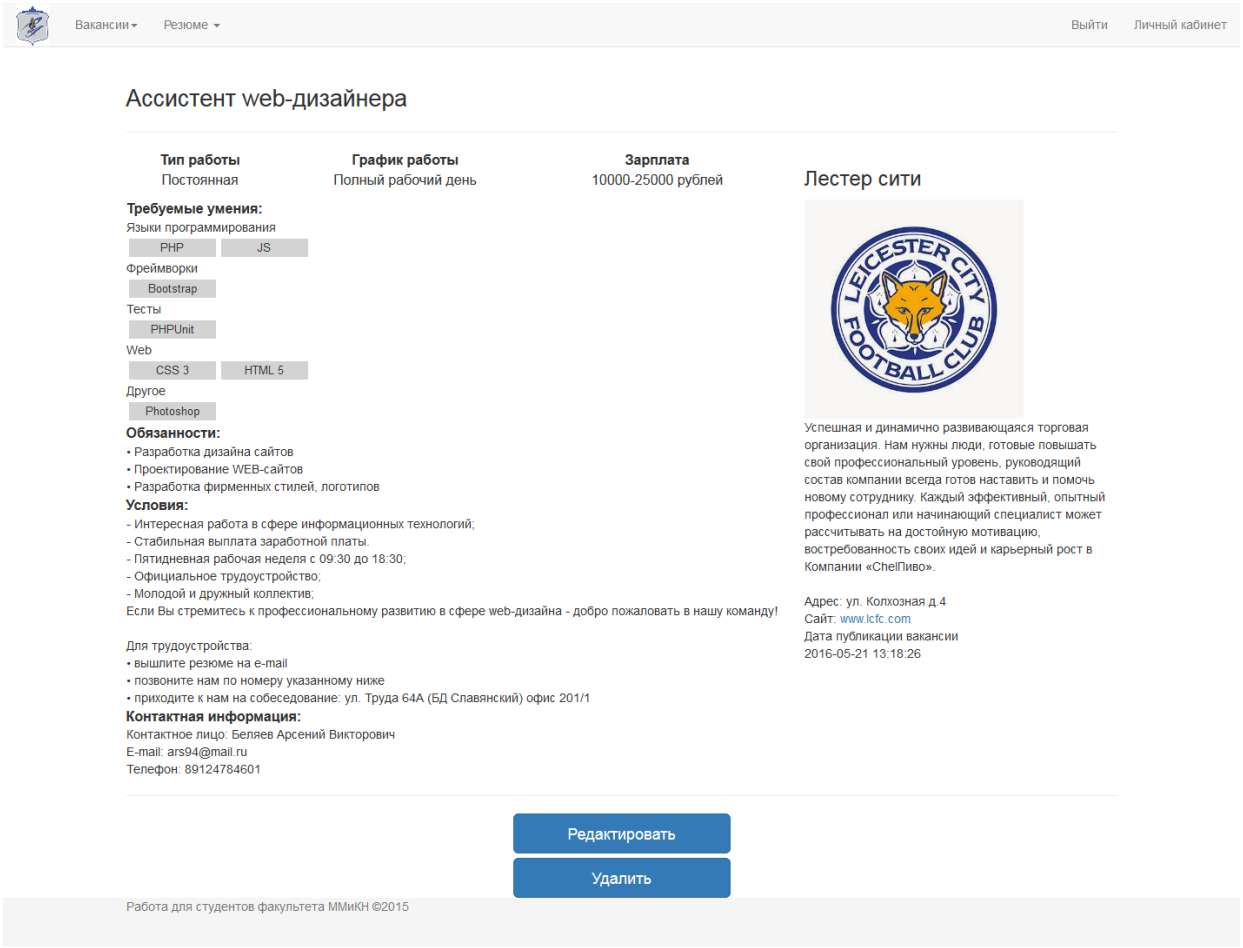

Рисунок 3.11

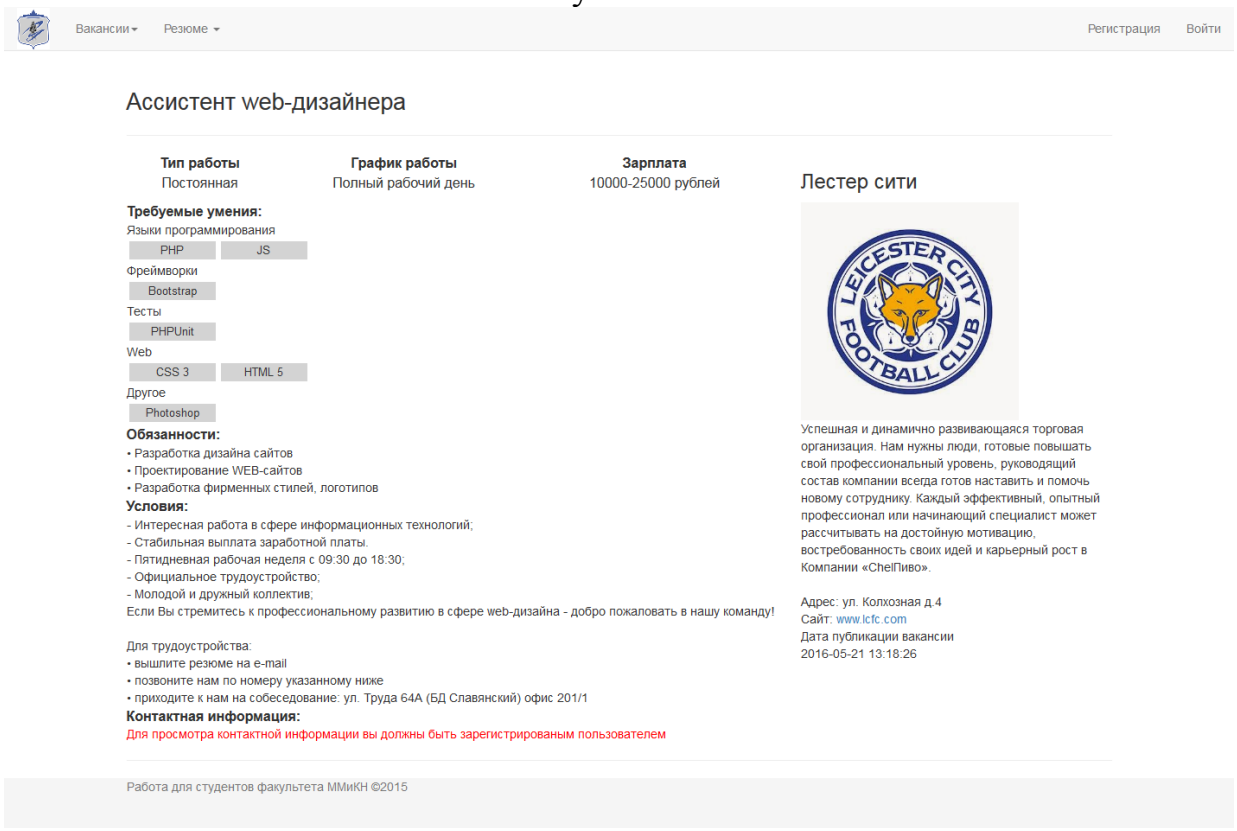

Рисунок 3.12

Страница поиска вакансий

На странице поиска вакансий (рисунок 3.13) размещены все навыки, перемещая, которые в ячейку рядом можно производить поиск.

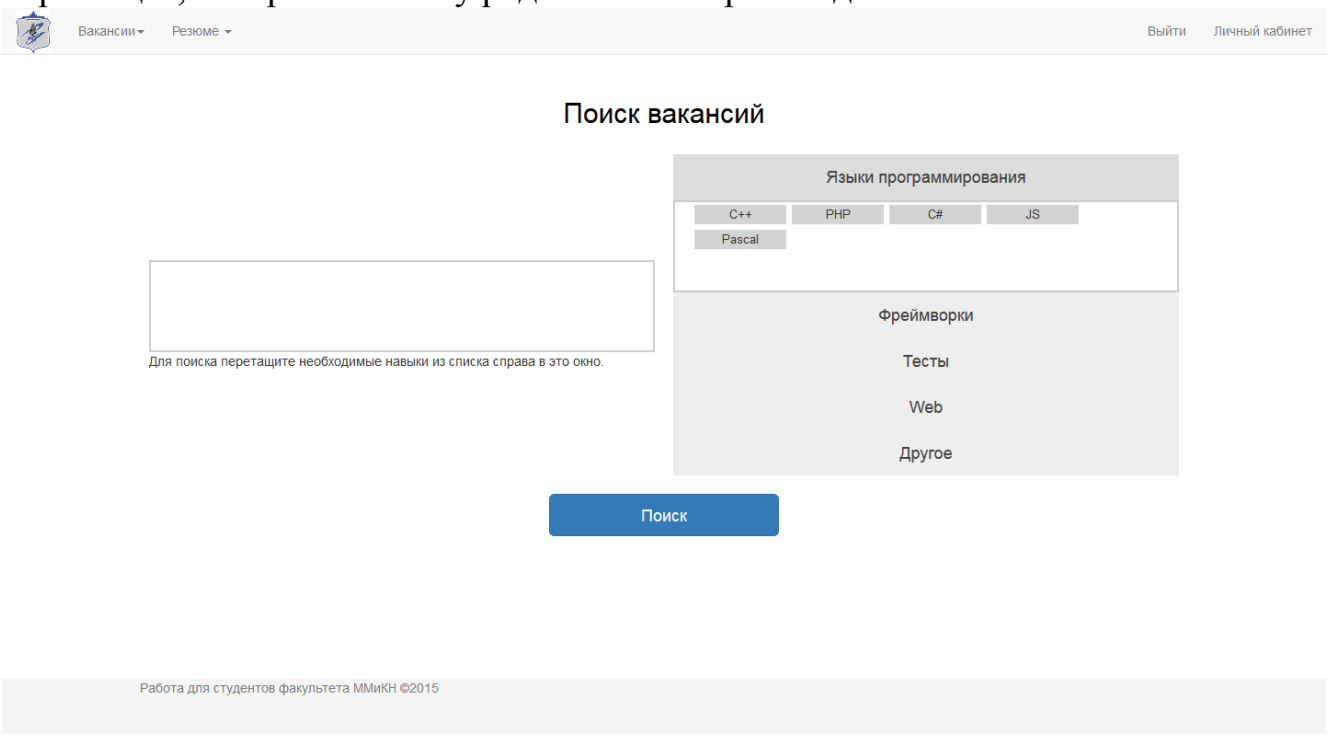

Рисунок 3.13

Страница со списком резюме

На странице со списком резюме (рисунок 3.14) размещена информация о пяти резюме. Также внизу имеется переключатель страниц, если есть непоказанные на странице резюме.

Страница добавления резюме

На странице добавления вакансии (рисунок 3.15) находится форма для добавления вакансии и список навыков, которые необходимо переносить в соседнюю ячейку для добавления их к вакансии.

Страница резюме

На странице резюме (рисунок 3.16) размещена вся информация о резюме.

Страница редактирования категорий и навыков

На странице редактирования категорий и навыков (рисунок 3.17) размещена форма для добавления и удаления навыков и категорий, а также для сортировки навыков.

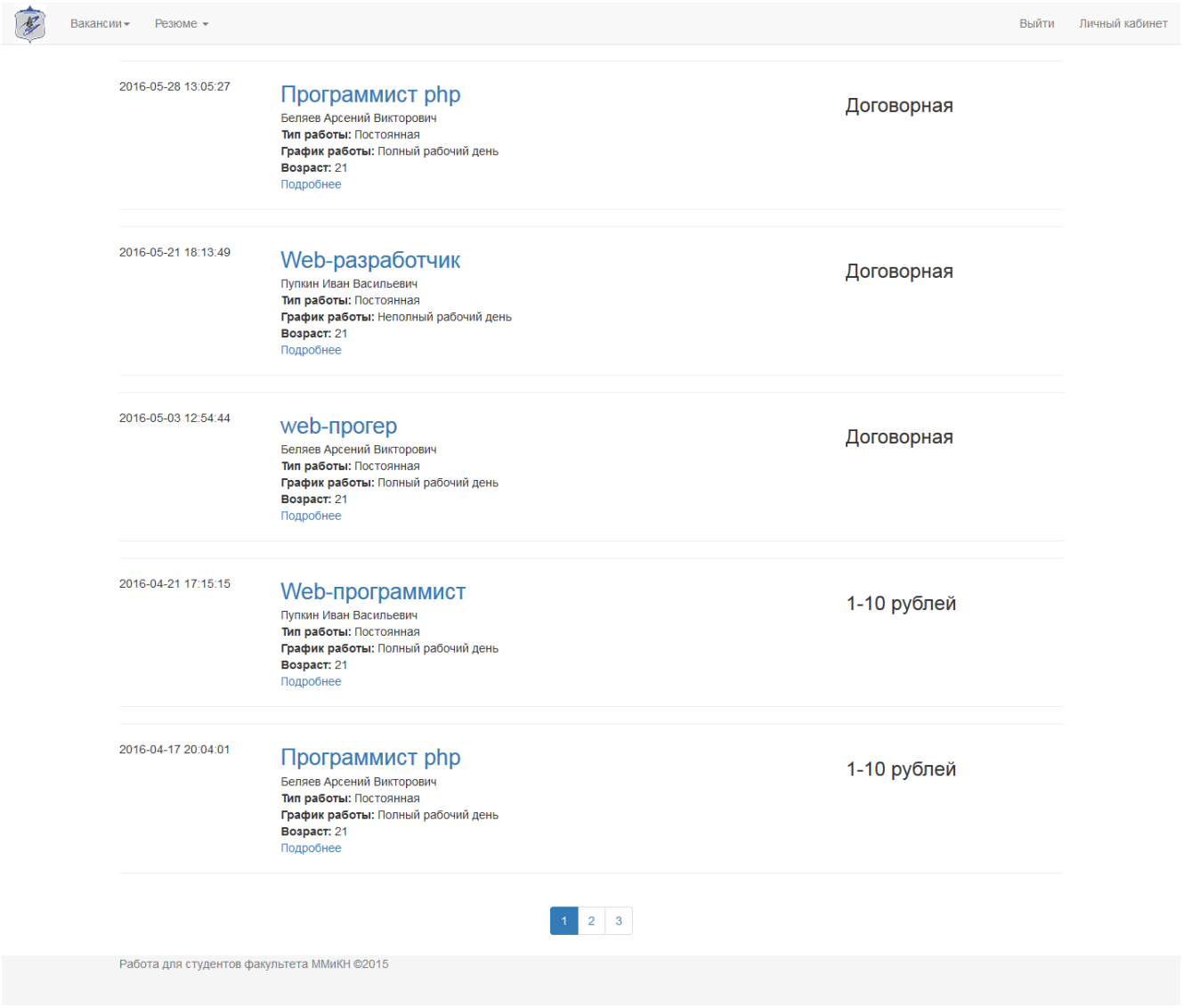

Рисунок 3.14

Страница изменения личных данных

На странице изменения личных данных (рисунок 3.18) размещены формы для изменения email, телефона, пароля.


Вакансии
Резюме
Выйти
Личный кабинет
Добавление резюме
Информация о себе
О себе
Возраст
Навыки
Языки программирования
C++
PHP
C#
JS
Pascal
Профессиональные навыки
(Перетащите необходимые навыки из списка справа, в окно слева)
Фреймворки
Тесты
Web
Другое
Новые навыки
(Если необходимые вам навыки отсутствуют выше, вы можете добавить их сами, для этого перечислите их, отделив с помощью точки с запятой (;))
Образование
Пожелания к работе
Должность
График работы
Полный рабочий день
Тип работы
Постоянная
Пожелания к работе
Зарплата
от
до
рублей
договорная
Добавить резюме
Очистить
Работа для студентов факультета ММиКН ©2015

Рисунок 3.15

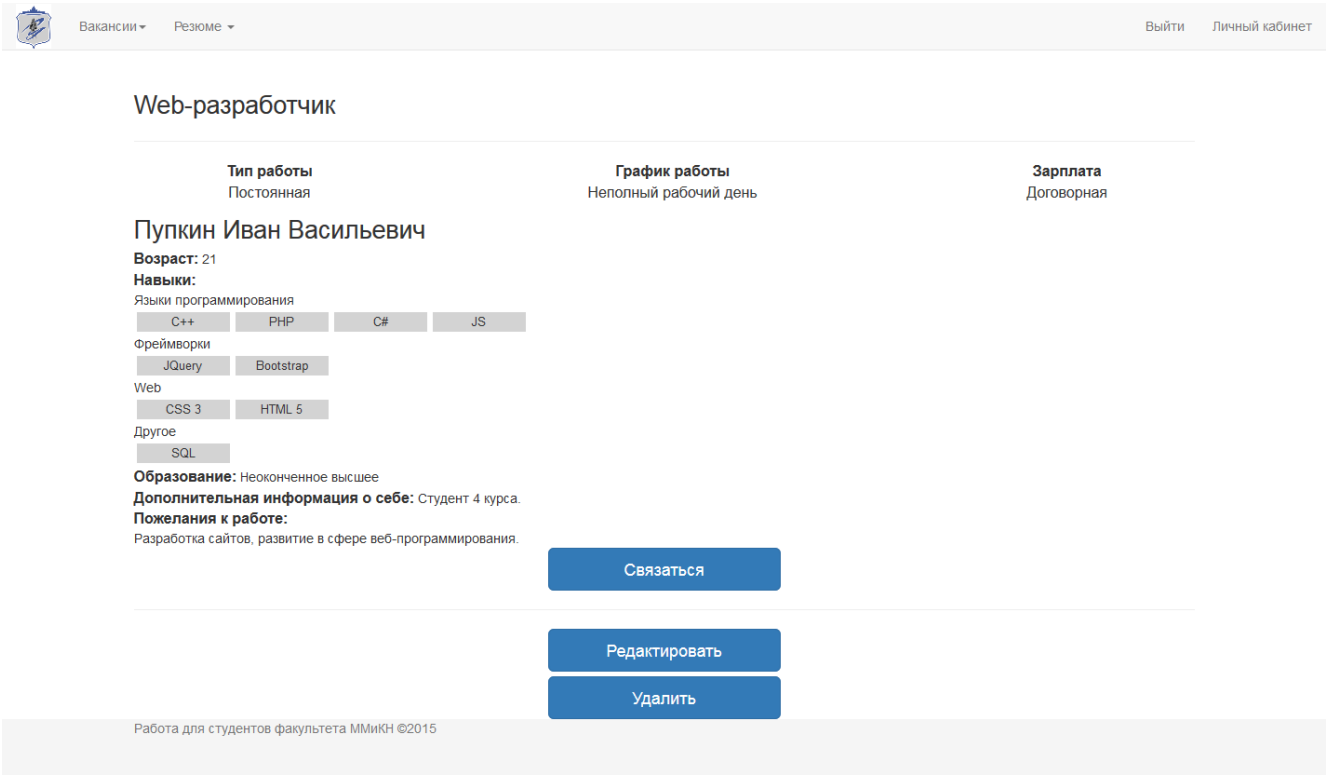

Рисунок 3.16

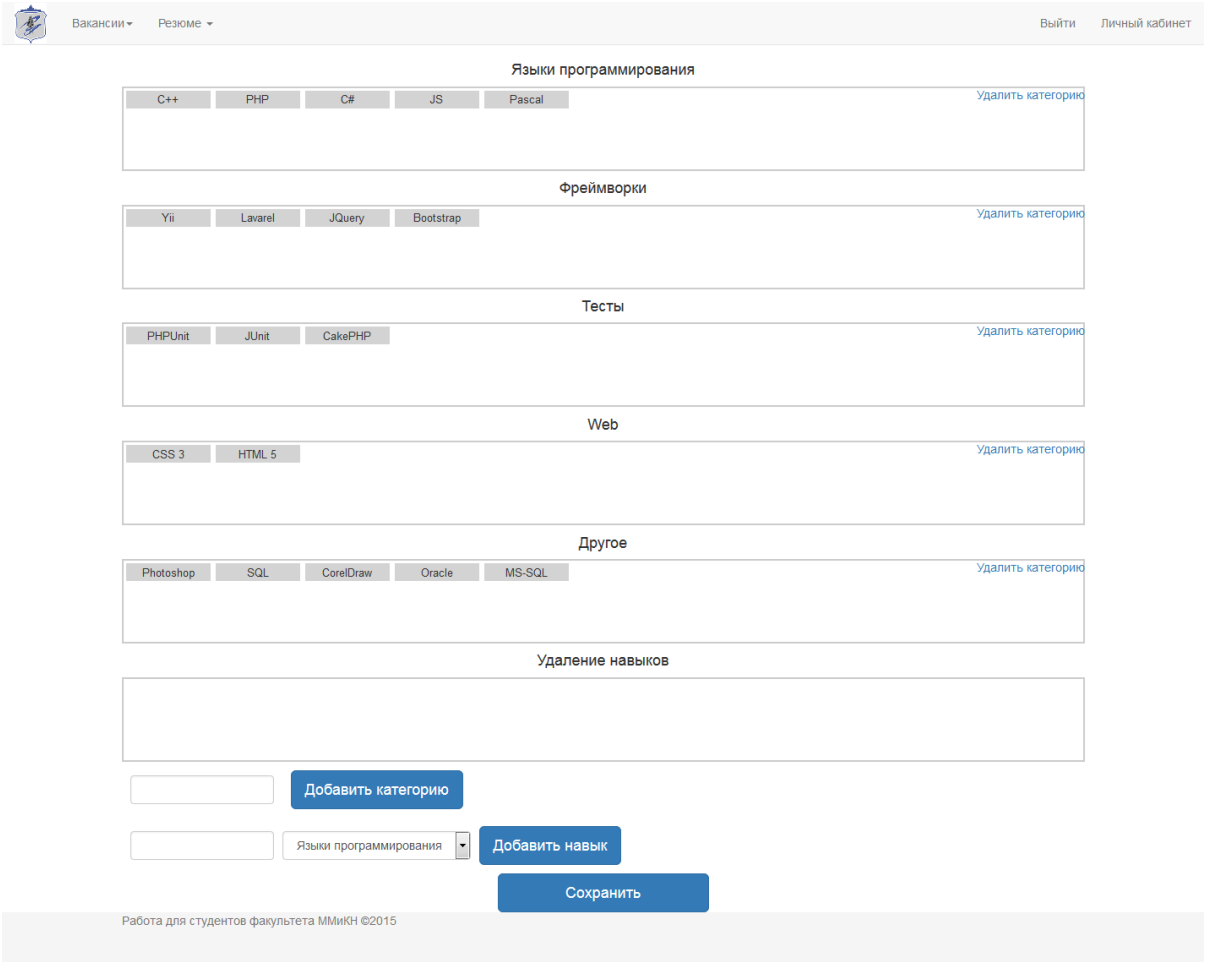

Рисунок 3.17

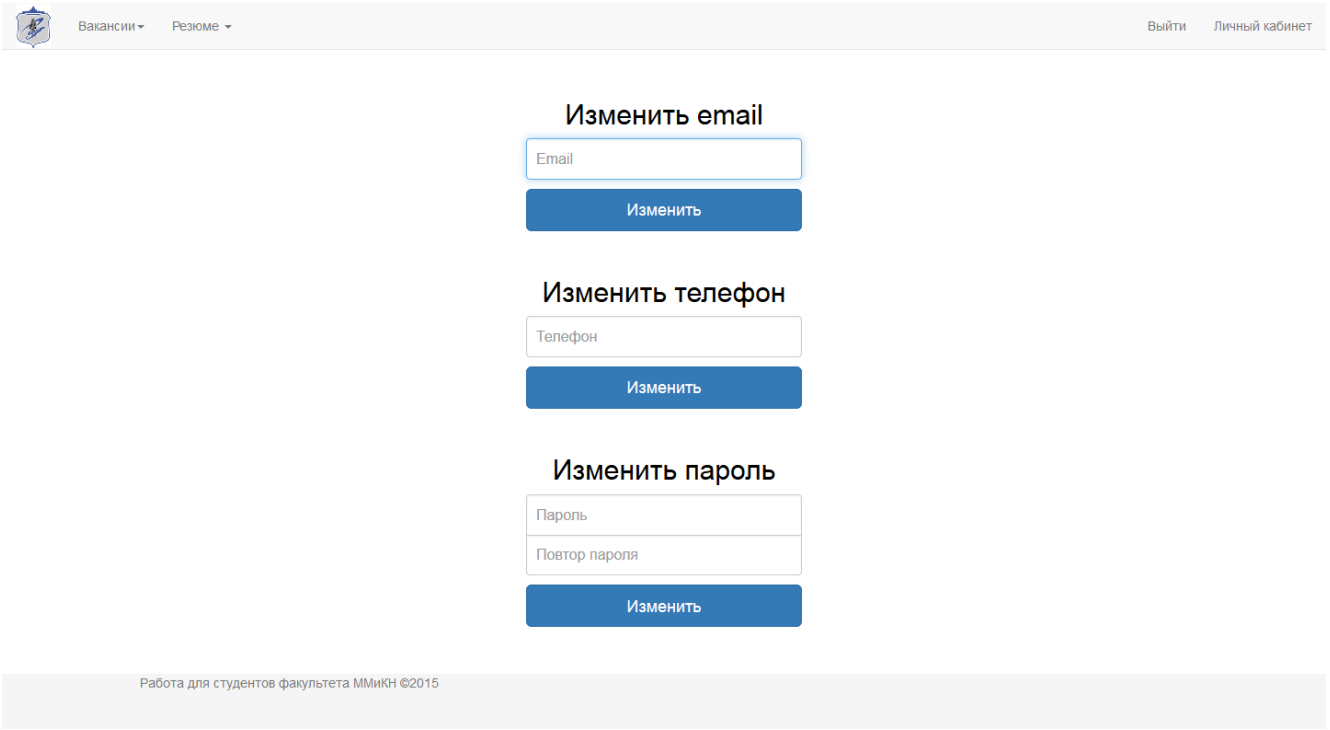

Рисунок 3.18

Страница связи работодателя с соискателем

На странице связи работодателя (рисунок 3.19) с соискателем размещена форма для отправки email сообщения соискателю.

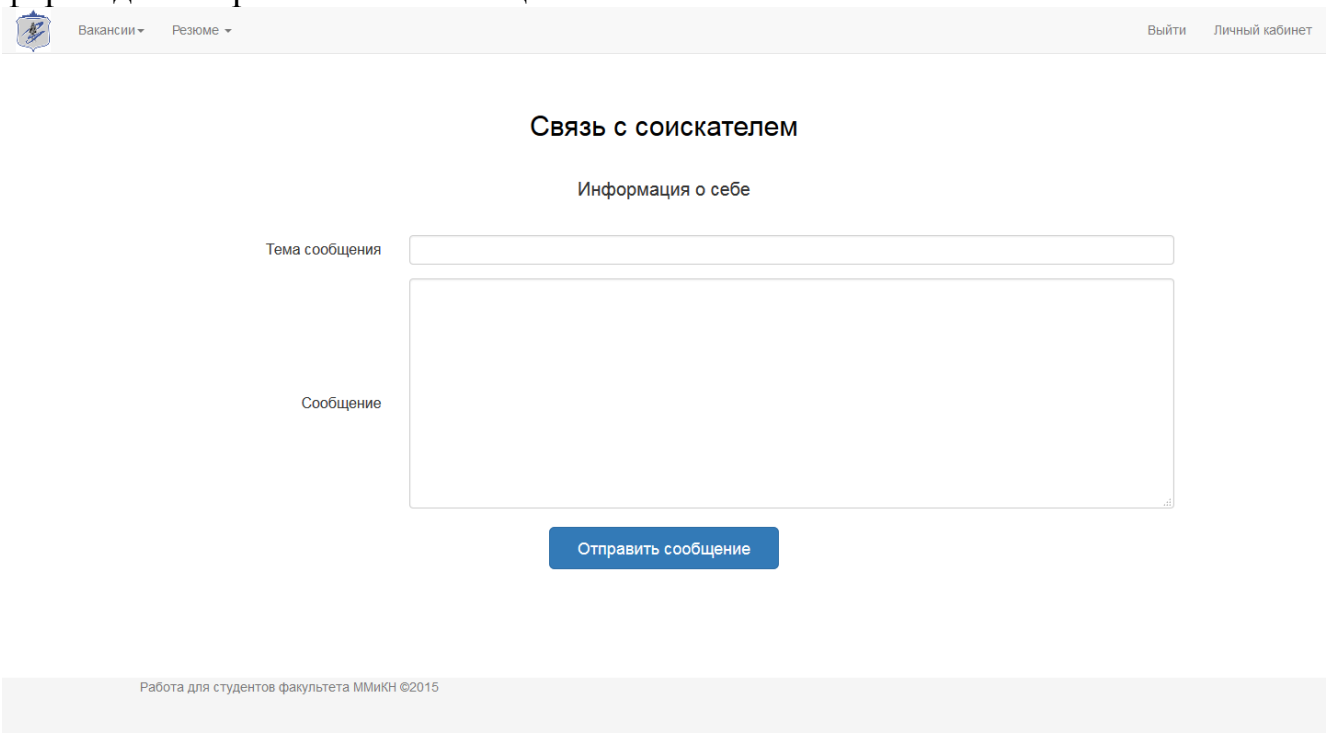

Рисунок 3.19

Выводы по разделу 3

Были разработаны персонажи и диаграммы вариантов использования. На их основе был разработан удобный пользовательский интерфейс, для которого были использованы HTML 5, CSS 3, Twitter Bootstrap и Jquery UI. Особенностью интерфейса является удобная работа с навыками, осуществляемая переносом мышкой. Также в этом разделе представлены скриншоты получившегося интерфейса.

4. РАЗРАБОТКА АЛГОРИТМА

Алгоритм будет разрабатываться с использованием шаблона MVC (model-view-controller), который изучен в статье на сайте [8]. Суть данного шаблона в том, что модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на отдельные компоненты так, что модификация одного компонента оказывает минимальное воздействие на остальные. Главным преимуществом данного шаблона является масштабируемость приложения. Контроллер — связующее звено между моделями, представлениями и другими компонентами. Контроллер отвечает за обработку запросов пользователя.

Представление (вид) — используется для задания внешнего отображения данных, полученных из контроллера и модели. Представления содержат HTML-разметку и небольшие вставки PHP-кода для обхода, форматирования и отображения данных.

Модель — содержит бизнес-логику приложения и включает методы выборки, обработки (например, правила проверки на правильность) и предоставления конкретных данных. Модель не должна напрямую взаимодействовать с пользователем.

Основная схема работы шаблона MVC представлена на рисунке 4.1.

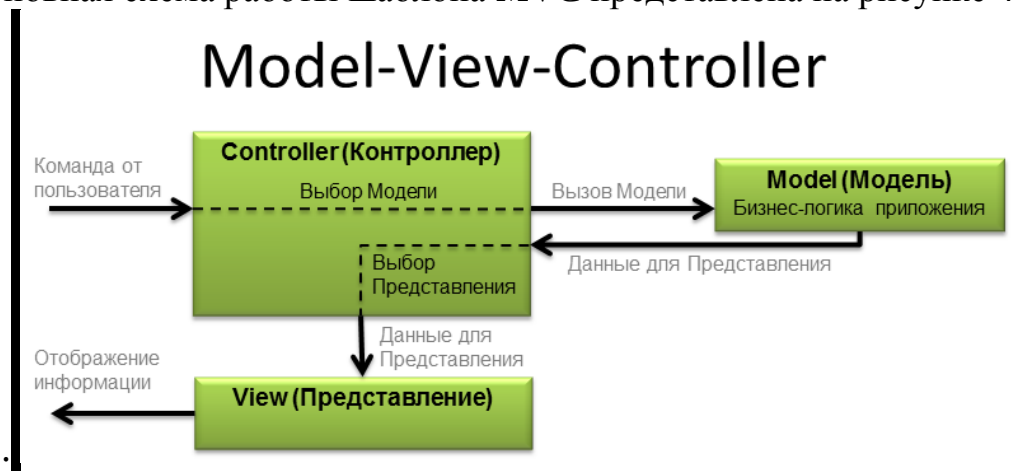

Рисунок 4.1

Ещё одной важной частью реализации данного шаблона является маршрутизация. За неё отвечает роутер. Задача роутера обрабатывать адрес, по которому обратился пользователь, и запускать необходимый контроллер.

Приложение будет написано на языке php, для его изучения использовались книги [9].

Структура файлов и папок приложения представлена на рисунке 4.2.

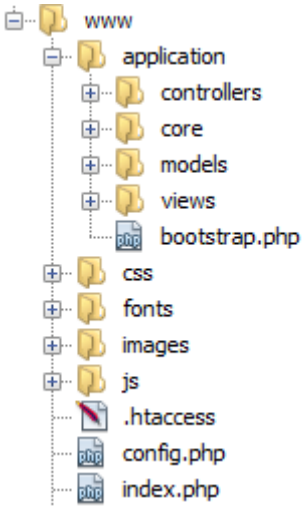

Рисунок 4.2

В приложении будет использоваться концепция единой точки входа, которой будет являться модуль index.php, он подключает модуль bootstrap.php. Модуль bootstrap.php инициирует загрузку приложения, подключая необходимые модули, создавая сессии и запуская функцию роутера start().

Приложения содержит модуль config.php, в котором есть класс Config с полями *$URL, $HOST, $USER, $PASS, $DB*. Этот класс используется для хранения переменных подключения к базе данных.

В папках controllers, model лежат контроллеры и модели с именами в виде controller_названиемодуля.php и model_названиемодуля.php, они содержат классы названия которых имеют вид: Controller_Названиемодуля.

В папке css содержаться файлы стиля, в папке js содержаться файлы со скриптами, в папке fonts файлы шрифтов, в папке images содержаться картинки необходимые для оформления.

В папке views содержаться представления, которые будет подцеплять контроллер.

В папке core лежит ядро приложения, которое состоит из следующих модулей:

- модуль controller.php
- модуль model.php
- модуль view.php
- модуль route.php
- модуль database.php
- модуль user.php
- модуль pagenav.php
- модуль category.php

Модуль controller.php – содержит класс Controller. Этот класс имеет поля *$model* и *$view*, для модели и представления. Конструктор инициализирует *$view*. Также он содержит следующие методы:

- метод *action_index()* — это действие, вызываемое по умолчанию, оно будет переопределяться при реализации классов потомков.

- метод *action_available()* – это действие, для генерации представления с ошибкой недоступности страницы для пользователя.
- метод *action_active()* – это действие, для генерации представления с ошибкой о не совершённой активации учетной записи пользователя.

Модуль model.php – содержит базовый класс Model. Он содержит поле *$db*, для хранения экземпляра класса DataBase, конструктор инициализирующий поле *$db*.

Модуль view.php – содержит класс View. Этот класс содержит метод *generate($content_view, $template_view, $title = null, $data = null, $pagenav = null, $skills = null)*, который предназначен для формирования вида. В него передаются следующие параметры:

1. $content_file — виды отображающие контент страниц;
2. $template_file — общий для всех страниц вид;
3. $data — массив, содержащий элементы контента страницы. Заполняется в моделе.
4. $pagenav – панель навигации, если необходимо.
5. $skills – массив навыков для выбора.

Модуль route.php – модуль содержит класс Route. Этот класс имеет два статических метода. Метод *start()* который разбивает адрес, по которому обратился пользователь, а на основе этого разбиения подключает необходимый файл с контроллером и запускает нужную функцию этого контроллера. Метод *ErorrPage404()*, перенаправляющий на страницу ошибки 404.

Модуль database.php – модуль содержит класс DataBase наследованный от Config. Класс содержит поля *$connect* и *$db*, в которых содержаться подключение и единственный экземпляр класс, чтобы не создавать множество подключений. Кроме того класс содержит несколько методов. Методы *Insert($query), update($query), delete ($query)*, для выполнения запросов к БД. Метод *connection()* для соединения с базой данных. И метод *getDB()* для создания или получения уже созданного экземпляра класса.

Модуль user.php – модуль содержит класс User, в котором есть статические методы для работы с пользователями сайта: проверка правильности заполнения формы (*check_name(), check_companyName(), check_site(), check_email(), check_phone(), check_password()*), генерация уникального идентификатора (*generate_uniqID($length=10)*), проверка состояние учетной записи пользователя (*isRemember(), isActive(), isAuth()*) и его роли (*isCompany(), isStudent(), isAdmin()*), проверка принадлежности резюме и вакансии пользователю (*isYourVacancy($id), isYourResume($id)*).

Модуль pagenav.php – содержит класс PageNav. У этого класса 4 поля: *$page* – номер текущей страницы, *$total* – общее количество записей в базе данных, *$limit* – количество записей на странице, *$allpage* – количество страниц. Также класс содержит один метод *generate_PageNav($type)*, который генерирует панель постраничной навигации.

Модуль category.php – содержит класс Category, в котором есть поля *$category, $skills*. Конструктор класса делает запрос к базе данных и заполняет эти поля категориями и навыками, либо если конструктору передаётся идентификатор вакансии или резюме, то поля класса заполняются выбранными навыками и категориями. Также класс содержит следующие методы:

- метод *get_category()* – для получения массива категорий.
- метод *get_skills_by_category($id)* ± для получения массива навыков, относящихся к определенной категории.
- метод *get_last_category_number()* – для получения номера последней категории.

Описанные модули являются основой приложения. Далее разработаем контроллеры, модели и представления для разделов сайта.

Главная страница

Модуль controller_main.php – содержит класс Controller_Main наследованный от Controller, который содержит метод *action_index()*. Этот метод генерирует представление главной страницы, используя метод *generate($content_view, $template_view, $title, $data)* класса view передавая ему имена файлов общего шаблона и вида с контентом страницы.

Модуль model_main.php – содержит класс Model_Main наследованный от Model, который содержит метод *get_data()*, для получения количества вакансий, кампаний и резюме на сайте.

Модуль main_view.php – содержит разметку главной страницы.

Для написания двух следующих модулей был изучен механизм реализации авторизации на сайте в статье [10].

Страница входа

Модуль controller_login – содержит класс Controller_Login наследованный от Controller. Этот класс содержит методы:

- *action_index()* ±генерирует представление страницы входа, используя метод *generate($content_view, $template_view, $title)* класса view.
- *action_login()* – используя метод *valid_data()* класса Model_Login() проверяет правильность введенных данных, затем выполняет вход пользователя используя метод *logIn()* класса Model_Login(). После выполнения входа перенаправляет на страницу профиля, либо для не активированных пользователей на страницу активации.
- *action_logout()* – используя метод *logout()* класса Model_Login выполняет выход пользователя и перенаправляет его на страницу входа.
- *action_loginforgot()* – генерирует страницу для отправки нового пароля пользователю.
- *action_newpassword()* ±с помощью метода *newpassword()* класса ModelLogin генерирует новый пароль для пользователя и отправляет ему на email.

Модуль model_login – содержит класс Model_Login наследованный от Model. Этот класс содержит поля *$role, $uniq_id, $user_id*. Также этот класс содержит следующие методы:

- *valid_data()* – проверяет правильность введенных данных, ищет пользователя в базе данных и получает его user_id, uniq_id и role.
- *logIn()* – авторизует пользователя на сайте.
- *logout()* – выполняет выход пользователя.
- *newpassword()* - генерирует новый пароль для пользователя и отправляет ему на email.

Модуль login_view – содержит разметку страницы входа.

Модуль login_forgot_view – содержит разметку страницы восстановления пароля.

Модуль newpassword_view – содержит разметку страницы с сообщением, что новый пароль выслан на email.

Страница регистрации

Модуль controller_registr – содержит класс Controller_Registr наследованный от Controller. Этот класс содержит методы:

- *action_index()* ±генерирует представление страницы регистрация, используя метод *generate($content_view, $template_view, $title)* класса view.
- *action_registr()* – используя метод *valid_data()* класса Model_Registr() проверяет правильность введенных данных, в случаи их правильности создаёт нового пользователя в базе данных, используя метод create_user*()* класса Model_Registr(). После создания пользователя происходит отправка сообщения ему на email со ссылкой на активацию пользователя, а затем перенаправляет на страницу с сообщением об успешной регистрации.
- *action_activation($hash)* – активирует учётную запись пользователя-работодателя, используя метод *activation($hash)* класса Model_Registr(), после чего генерирует страницу с сообщением об успешной активации или о неудачной активации. Если же пользователь уже активировал учётную запись, то метод activation($hash) не запускается, а происходит генерация страницы с сообщением, что активация уже выполнена.
- *action_activcompany()* – отправляет email сообщение пользователю со ссылкой для активации учетной записи и генерирует страницу с возможностью повторной отправки.
- *action_emailagain($hash)* – метод для повторной посылки email сообщения для активации учетной записи.
- *action_activstudent()* – генерирует страницу для активации учётной записи пользователя-студента.
- *action_addactivstudent()* – проверяет введённые данные с помощью метода *valid_activdata()* класса Model_Registr, в случае их правильности с помощью метода *active_student()* выполняется активация учётной записи пользователя-студента, после чего генерируется страница с сообщением об успешной активации.
- *action_changedata()* – генерируется страницу изменения данных пользователя.

- *action_changeemail()* ± с помощью метода *changeemail()* класса Model_Registr меняет email пользователя, в случае успеха генерирует страницу с сообщением об успешном изменении данных.
- *action_changephone()* ± с помощью метода *changephone()* класса Model_Registr меняет телефон пользователя, в случае успеха генерирует страницу с сообщением об успешном изменении данных.
- *action_changepassword()* ± с помощью метода *changeemail()* класса Model_Registr меняет пароль пользователя, в случае успеха генерирует страницу с сообщением об успешном изменении данных.
- *action_changecompanyname()* ± с помощью метода *changecompanyname()* класса Model_Registr меняет название компании пользователя, в случае успеха генерирует страницу с сообщением об успешном изменении данных.
- *action_changesite()* ± с помощью метода *changesite()* класса Model_Registr меняет адрес сайта компании пользователя, в случае успеха генерирует страницу с сообщением об успешном изменении данных.
- *action_changeaddress()* ± с помощью метода *changeaddress()* класса Model_Registr меняет адрес компании пользователя, в случае успеха генерирует страницу с сообщением об успешном изменении данных.
- *action_changeinfo()* ± с помощью метода *changeinfo()* класса Model_Registr меняет информацию о компании пользователя, в случае успеха генерирует страницу с сообщением об успешном изменении данных.

Модуль model_registr – содержит класс Model_Registr наследованный от Model. Этот класс содержит поле *$uniq_id*. Также этот класс содержит следующие методы:

- *valid_data()* – проверяет правильность введенных данных при регистрации, отсутствие пользователя с таким же email адресом, шифрует пароль.
- *valid_activdata()* – проверяет правильность введенных данных при активации пользователя-студента, шифрует пароль.
- *activ_student()* – активирует учётную запись пользователя-студента.
- *create_user()* – создаёт пользователя-работодателя в базе данных.
- *activation($hash)* – активирует учётную запись пользователя-работодателя.
- *changeemail()* – проверяет правильность введённых данных и отсутствие такого email адреса в базе данных, в случае успеха изменяет email адресс пользователя в базе данных.
- *changephone()* – проверяет правильность введённых данных, в случае успеха изменяет телефон пользователя в базе данных.
- *changepassword()* – проверяет правильность введённых данных, в случае успеха изменяет пароль пользователя в базе данных.
- *changecompanyname()* – проверяет правильность введённых данных, в случае успеха изменяет название компании пользователя в базе данных.
- *changesite()* – проверяет правильность введённых данных, в случае успеха изменяет адрес сайта компании пользователя в базе данных.

- *changeaddress()* – проверяет правильность введённых данных, в случае успеха изменяет адрес компании пользователя в базе данных.
- *changeinfo()* – проверяет правильность введённых данных, в случае успеха изменяет информацию о компании пользователя в базе данных.
- *emailbyuser_id()* – возвращает email пользователя по его user_id.

Модуль registr_view – содержит разметку страницы регистрации.

Модуль success_registr_view – содержит разметку страницы успешной регистрации.

Модуль activcompany_view – содержит разметку страницы повторной отправки email сообщение для активации профиля.

Модуль already_active_view – содержит разметку страницы с сообщением, о том, что учётная запись уже активирована.

Модуль success_active_view – содержит разметку страницы успешной активации.

Модуль nosuccess_active_view – содержит разметку страницы с сообщением, что пользователь для активации не найден.

Модуль activstudent_view – содержит разметку страницы активации учётной записи студента.

Модуль changedata_view – содержит разметку страницы изменения данных.

Модуль success_change_view – содержит разметку страницы успешного изменения данных.

Страница вакансии

Модуль controller_vacancy – содержит класс Controller_Vacancy наследованный от Controller. Этот класс содержит методы:

- *action_page($page)* – с помощью методов *total_page()* и *get_page($page, $total)* модели Model_Vacancy получает информацию о странице вакансий и генерирует, страницу с номером *$page* со списком вакансий. Кроме того создаёт объект PageNav (постраничная навигация) и передаёт её на представление.
- *action_id($id)* – генерирует страницу вакансии по id.
- *action_new()* – генерирует страницу для добавления вакансии.
- *action_add()* – проверяет введённые данные при добавлении вакансии, в случае их правильности добавляет вакансию в базу данных и перенаправляет на страницу добавленной вакансии.
- *action_edit($id)* – генерирует страницу для редактирования вакансии.
- *action_addedit($id)* – проверяет введённые данные при редактировании вакансии, в случае их правильности редактирует вакансию в базе данных и перенаправляет на страницу отредактированной вакансии.
- *action_delete($id)* – удаляет вакансию из базы данных и генерирует страницу с сообщением об успешном удалении.

Модуль model_vacancy – содержит класс Model_Vacancy наследованный от Model. Этот класс содержит поле *$limit*. Также этот класс содержит следующие методы:

- *get_page($page, $total)* – на основе запроса к базе данных формирует массив с данными для страницы со списком вакансий.
- *total_page()* – возвращает количество вакансий в базе данных.
- *get_id($id)* – на основе запроса к базе данных формирует массив с данными для страницы вакансии.
- *valid_data()* – проверяет данные введенные пользователем при добавлении и редактировании вакансии.
- *create_vacancy()* – создаёт вакансию в базе данных.
- *update_vacancy($id)* – изменяет вакансию в базе данных, после редактирования пользователем.
- *delete_vacancy($id)* – удаляет вакансию из базы данных.

Модуль vacancy_list_view – содержит разметку страницы со списком вакансий.

Модуль vacancy_id_view – содержит разметку страницы одной вакансии.

Модуль vacancy_new_view – содержит разметку страницы добавления вакансии.

Модуль vacancy_edit_view – содержит разметку страницы изменения вакансии.

Модуль vacancy_success_delete_view – содержит разметку страницы с сообщением об успешном удалении вакансии.

Страница резюме

Модуль controller_resume – содержит класс Controller_Resume наследованный от Controller. Этот класс содержит методы:

- *action_page($page)* – с помощью методов *total_page()* и *get_page($page, $total)* модели Model_Resume получает информацию о странице резюме и генерирует, страницу с номером *$page* со списком резюме. Кроме того создаёт объект PageNav (постраничная навигация) и передаёт её на представление.
- *action_id($id)* – генерирует страницу резюме по id.
- *action_new()* – генерирует страницу для добавления резюме.
- *action_add()* – проверяет введённые данные при добавлении резюме, в случае их правильности добавляет резюме в базу данных и перенаправляет на страницу добавленного резюме.
- *action_edit($id)* – генерирует страницу для редактирования резюме.
- *action_addedit($id)* – проверяет введённые данные при редактировании резюме, в случае их правильности редактирует резюме в базе данных и перенаправляет на страницу отредактированного резюме.
- *action_delete($id)* – удаляет резюме из базы данных и генерирует страницу с сообщением об успешном удалении.
- *action_linkpage($resume_id)* – генерирует страницу для связи с соискателем.
- *action_link($resume_id)* – отправляет сообщение работодателя на email соискателя и генерирует страницу об успешной отправке, в случае успеха.

Модуль model_resume – содержит класс Model_Resume наследованный от Model. Этот класс содержит поле *$limit*. Также этот класс содержит следующие методы:

- *get_page($page, $total)* – на основе запроса к базе данных формирует массив с данными для страницы со списком резюме. Схема данной функции представлена на рисунке 4.3.

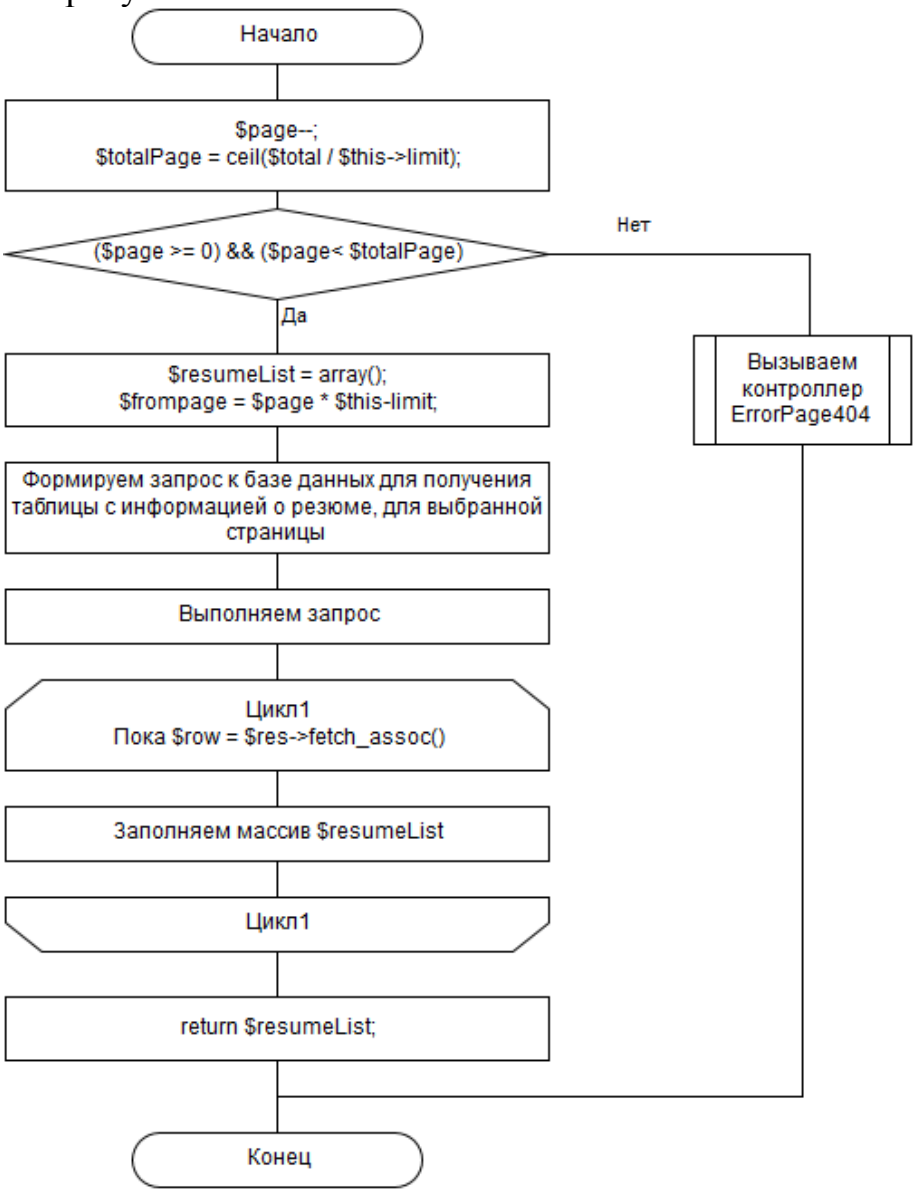

Рисунок 4.3

- *total_page()* – возвращает количество резюме в базе данных.
- *get_id($id)* – на основе запроса к базе данных формирует массив с данными для страницы резюме.
- *valid_data()* – проверяет данные введенные пользователем при добавлении и редактировании резюме.
- *create_resume()* – создаёт резюме в базе данных. Схема данной функции представлена на рисунке 4.4.

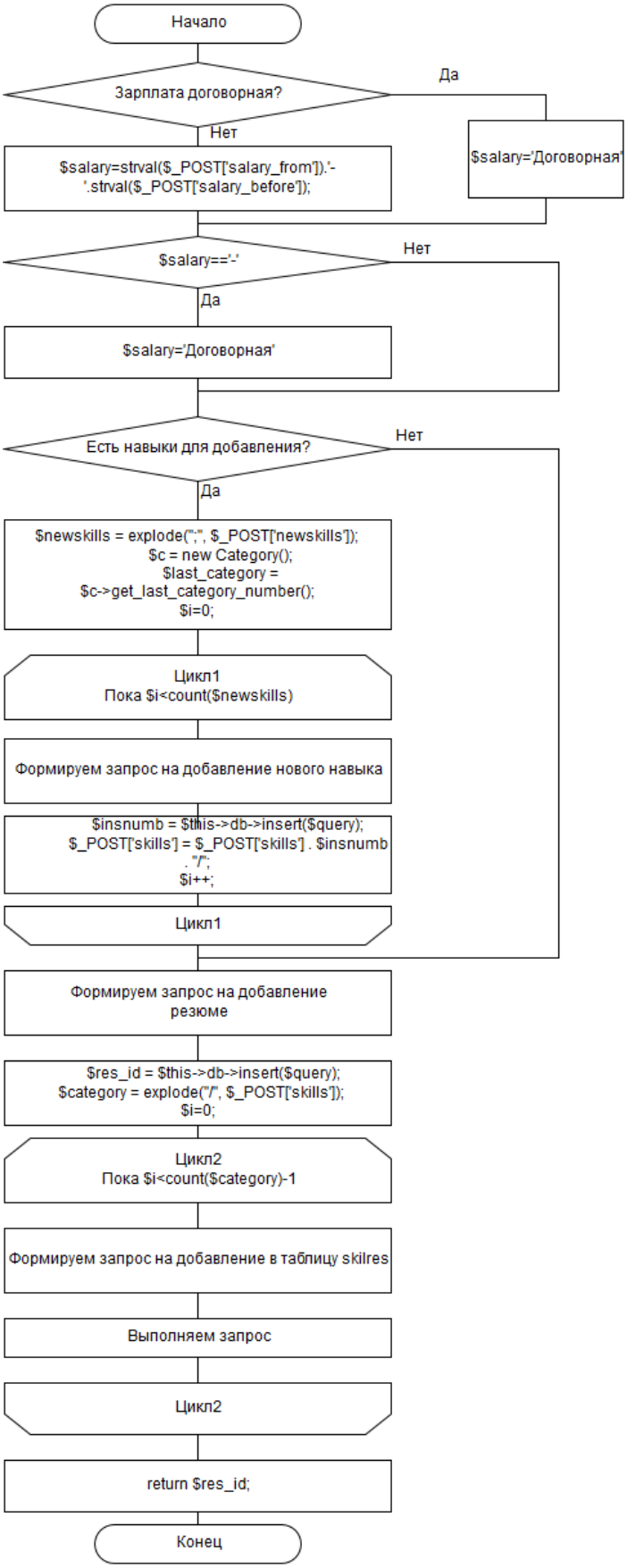

Рисунок 4.4 – Схема функции добавления резюме

- *update_resume($id)* – изменяет резюме в базе данных, после редактирования пользователем.
- *delete_resume($id)* – удаляет резюме из базы данных.

Модуль resume_list_view – содержит разметку страницы со списком резюме.

Модуль resume _id_view – содержит разметку страницы одного резюме.

Модуль resume _new_view – содержит разметку страницы добавления резюме.

Модуль resume _edit_view – содержит разметку страницы изменения резюме.

Модуль resume _success_delete_view – содержит разметку страницы с сообщением об успешном удалении резюме.

Модуль link_view – содержит разметку страницы для отправки сообщения.

Модуль success_mail_view – содержит разметку страницы с сообщением об успешной отправке сообщения.

Страница компании

Модуль controller_company – содержит класс Controller_Company наследованный от Controller. Этот класс содержит метод *action_id($id)*, который генерирует страницу резюме по id.

Модуль model_company – содержит класс Model_Company наследованный от Model. Этот класс содержит следующие методы:

- *get_id($id)* – на основе запроса к базе данных формирует массив с данными для страницы компании, на которой показывается список её вакансий.
- *get_companyname($id)* – получает имя компании по идентификатору компании..

Модуль company_view – содержит разметку страницы компании.

Страница личного кабинета

Модуль controller_profile – содержит класс Controller_Profile наследованный от Controller. Этот класс содержит метод *action_index()*, который генерирует страницу профиля пользователя, при этом содержимое страницы будет отличаться, в зависимости от роли пользователя.

Модуль model_profile – содержит класс Model_Profile наследованный от Model. Этот класс содержит следующие методы:

- *get_data()* – на основе запроса к базе данных формирует массив с данными о пользователи.
- *get_vacancy()* – на основе запроса к базе данных формирует массив с данными о вакансиях пользователя.
- *get_resume()* – на основе запроса к базе данных формирует массив с данными о резюме пользователя.

Модуль profile_view – содержит разметку страницы профиля.

Модуль profile_vacancy_view – содержит разметку для отображения списка вакансий пользователя в его профиле.

Модуль profile_resume_view – содержит разметку для отображения списка резюме пользователя в его профиле.

Страница ошибки 404

Модуль controller_404 – содержит класс Controller_400 наследованный от Controller. Этот класс содержит метод *action_index()*, который генерирует страницу с ошибкой 404.

Модуль 404_view – содержит разметку страницы с ошибкой 404.

Страница поиска

Модуль controller_search – содержит класс Controller_Search наследованный от Controller. Этот класс содержит методы:

- *action_vacancy()* – генерирует страницу для поиска вакансий.
- *action_resume()* – генерирует страницу для поиска резюме.
- *action_countuservacancy()* – выводит количество пользователей удовлетворяющих критериям поиска вакансий.
- *action_countuserresume()* – выводит количество пользователей удовлетворяющих критериям поиска резюме.
- *action_donevacancysearch()* – генерирует страницу с результатами поиска по вакансиям.
- *action_doneresumesearch()* – генерирует страницу с результатами поиска по резюме.

Модуль model_search – содержит класс Model_Search наследованный от Model. Этот класс содержит следующие методы:

- *count_uservacancy()* – на основе запроса к базе данных подсчитывает количество пользователей удовлетворяющих критериям поиска вакансий.
- *count_userresume()* – на основе запроса к базе данных подсчитывает количество пользователей удовлетворяющих критериям поиска резюме.
- *vacancy_search()* – на основе запроса к базе данных формирует массив с результатами поиска вакансий.
- *resume_search()* – на основе запроса к базе данных формирует массив с результатами поиска резюме.

Модуль searchnull_view – содержит разметку страницы с информацией о том, что поиск не дал результатов.

Модуль search_vacancy_view – содержит разметку для отображения страницы поиска вакансий.

Модуль search_resume_view – содержит разметку для отображения страницы поиска резюме.

Модуль search_vacancy_done_view – содержит разметку для отображения страницы с результатами поиска вакансий.

Модуль search_resume_done_view – содержит разметку для отображения страницы с результатами поиска резюме.

Страница администратора

Модуль controller_admin – содержит класс Controller_Admin наследованный от Controller. Этот класс содержит методы:

- *action_regstudent()* – генерирует страницу для регистрации студентов.

- *action_addregstudent()* – выполняет проверку введённых данных и в случае правильности добавляет пользователя в базу данных используя метод *create_user()* класса Model_Admin().
- *action_category()* – генерирует страницу для добавления, удаления, редактирования навыков и категорий, а также сортировки навыков.
- *action_savecategskills()* – с помощью метода *save_categ_skills()* модели Model_Admin сохраняет распределение навыков по категориям, а также удаляет навыки помещенные в корзину и обновляет страницу.
- *action_savecateg()* – с помощью метода *save_categ()* модели Model_Admin добавляет новую категорию в базу данных и обновляет страницу.
- *action_editcateg()* – с помощью метода *edit_categ()* модели Model_Admin редактирует название категории в базе данных и обновляет страницу.
- *action_delcateg()* – с помощью метода *del_categ()* модели Model_Admin удаляет категорию из базы данных и обновляет страницу.
- *action_saveskill()* – с помощью метода *save_skill()* модели Model_Admin добавляет новый навык в базу данных и обновляет страницу.
- *action_editskill()* – с помощью метода *edit_skill()* модели Model_Admin редактирует название навыка в базе данных и обновляет страницу.

Модуль model_search – содержит класс Model_Search наследованный от Model. Этот класс содержит следующие методы:

- *valid_data()* – проверяет правильность введенных данных при регистрации студентов, отсутствие пользователя с таким же email адресом.
- *create_student()* – создаёт пользователя-работодателя в базе данных.
- *save_categ_skills()* – сохраняет распределение навыков по категориям. Схема данной функции представлена на рисунке 4.5.
- *save_skill()* – добавляет новый навык в базу данных.
- *edit_skill()* – редактирует название навыка в базе данных.
- *save_categ()* – добавляет новую категорию в базу данных.
- *edit_categ()* – редактирует название категории в базе данных.
- *del_categ()* – удаляет категорию из базы данных.

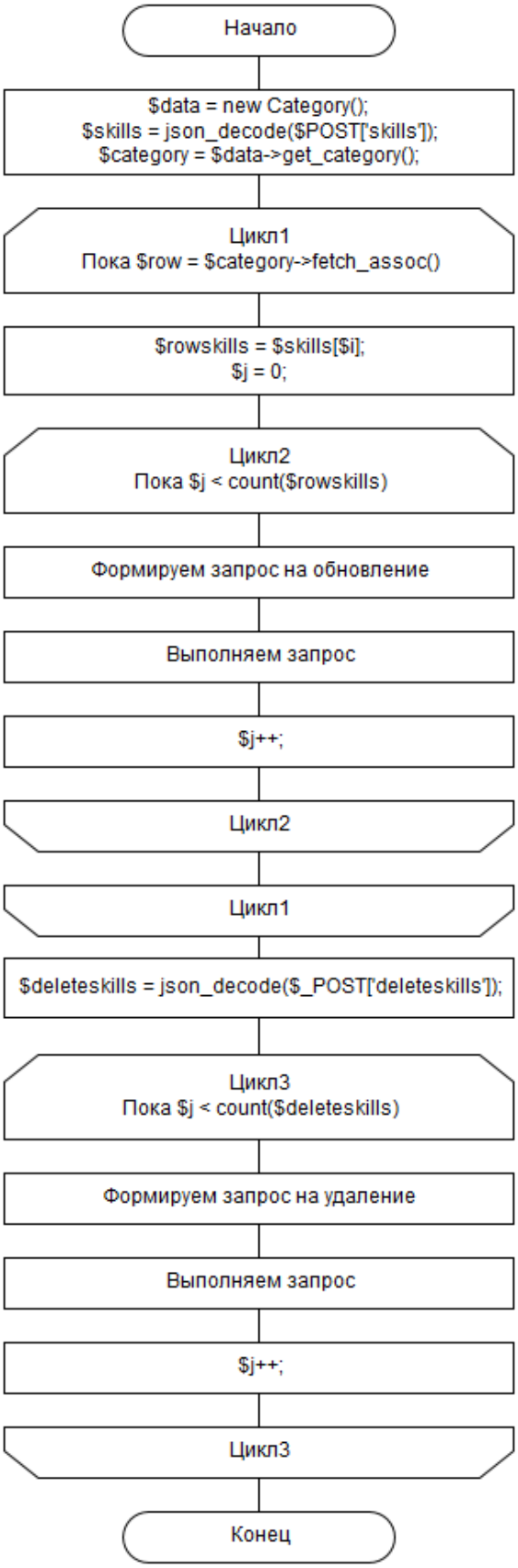

Рисунок 4.5 – Схема функции сохранения распределения навыков по категориям

Модуль regstudent_view – содержит разметку страницы для регистрации студентов.

Модуль category_view – содержит разметку для отображения страницы для добавления, удаления, редактирования навыков и категорий, а также сортировки навыков.

Выводы по разделу 4

В результате разработана архитектура сайта на основе MVC-шаблона. Описаны все модули сайта. Всего написынх модулей: 70.

ЗАКЛЮЧЕНИЕ

В результате работы были решены поставленные задачи:

1) Рассмотрены существующие информационные системы трудоустройства и сформулированы требования к системе для поддержки трудоустройства студентов.

2) Спроектирована база данных, хранящая информацию о навыках, резюме, вакансиях и пользователях веб-сайта.

3) Разработан интерфейс с использованием методики персонажей и диаграмм использования. Особенностью интерфейса является удобная работа с навыками, осуществляемая переносом мышкой.

4) Разработана архитектура сайта на основе шаблона MVC.

5) Было произведено тестирование сайта, в ходе которого были выявлены и исправлены некоторые ошибки.

Разработанный сайт был запущен в тестовом режиме на сервере факультета «Математики, механики и компьютерных наук».

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Работа в Челябинске – вакансии на 74.ru работа (Зарплата.ру). – URL: http://chelyabinsk.zarplata.ru/ (дата обращения: 07.03.16)

2 Работа, вакансии, база резюме, поиск работы на HeadHunter (hh.ru). – URL: https://hh.ru/ (дата обращения: 08.03.16)

3 Центр содействия трудоустройству выпускников ИГУ. – URL: https://job.isu.ru/ (дата обращения: 09.03.16)

4 ИАС Универис - Региональный центр трудоустройства выпускников. – URL: https://univeris.susu.ru/job/ (дата обращения: 10.03.2016)

5 Купер, А. Алан Купер об интерфейсе. Основы проектирования взаимодействия / Купер А., Рейман Р., Кронин Д. // – Пер. с англ. – СПб.: Символ-Плюс. – 2009. – 688 с.

6 Макфарланд, Д. Большая книга CSS3 / Макфарланд Д. // – СПб.: Питер. – 2014. – 608 с.

7 jQuery UI | Перетаскивание элементов (Drag and drop). – URL: http://professorweb.ru/my/javascript/jquery/level4/4_12.php (дата обращения: 05.04.2016)

8 Реализация MVC паттерна на примере создания сайта-визитки на PHP. – Дата обновления: 27.09.2012. URL: https://habrahabr.ru/post/150267/ (дата обращения: 07.02.2016).

9 Зандстра М. PHP:объекты, шаблоны и методики программирования / Зандстра М. // – Москва.: “И.Д. Вильямс”. – 2011. – 560 с.

10 Авторизация (php + mysql) и запоминание пользователей для начинающих. – URL: http://werther.wwgroup.in.ua/2010/07/07/авторизация-php-mysql-и-запоминание -пользов/ (дата обращения: 27.02.2016)